

Big Data Analytics for Semantic Data BigSem Tutorial

Module 1: Libraries for Analytics and ML in Python

Chelmis Charalampos, Bedirhan Gergin
University at Albany, SUNY

ISWC 2024

What is Jupyter Notebook?

- Interactive platforms to write and run code, visualize results, and document work—perfect for experiments and tutorials



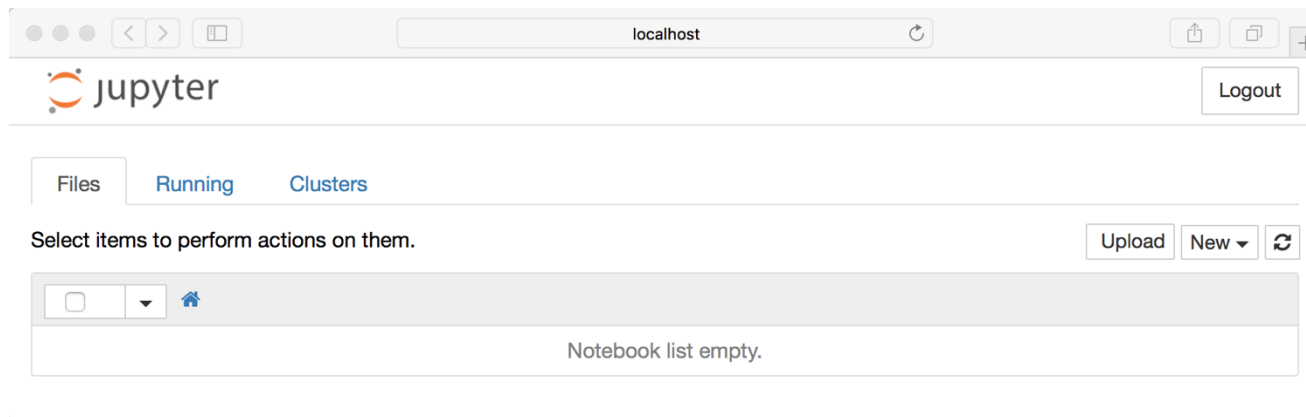
How to Access Jupyter Notebook

- Install and launch Jupyter through Anaconda, or via the terminal with `jupyter notebook`. The notebook server will open in your browser

```
Shell
$ pip install jupyter

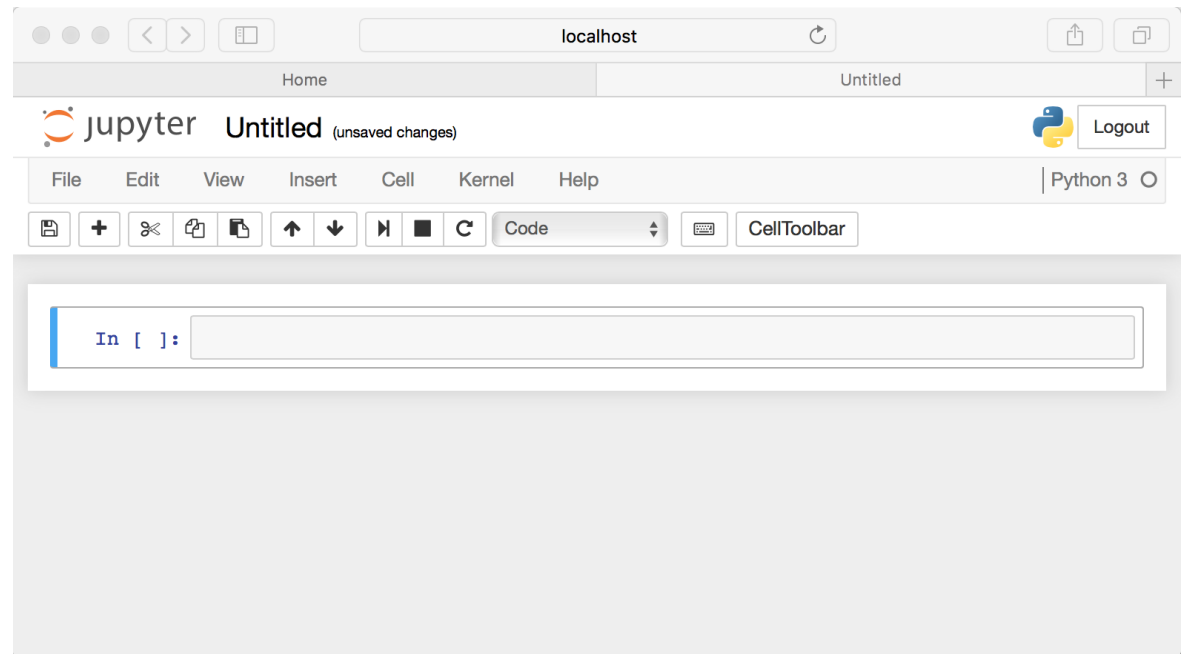
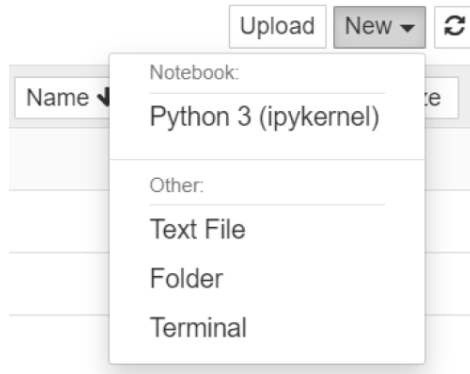
Shell
$ jupyter notebook
```

- This will start up Jupyter and your default browser should start (or open a new tab) to the following URL: <http://localhost:8888/tree>



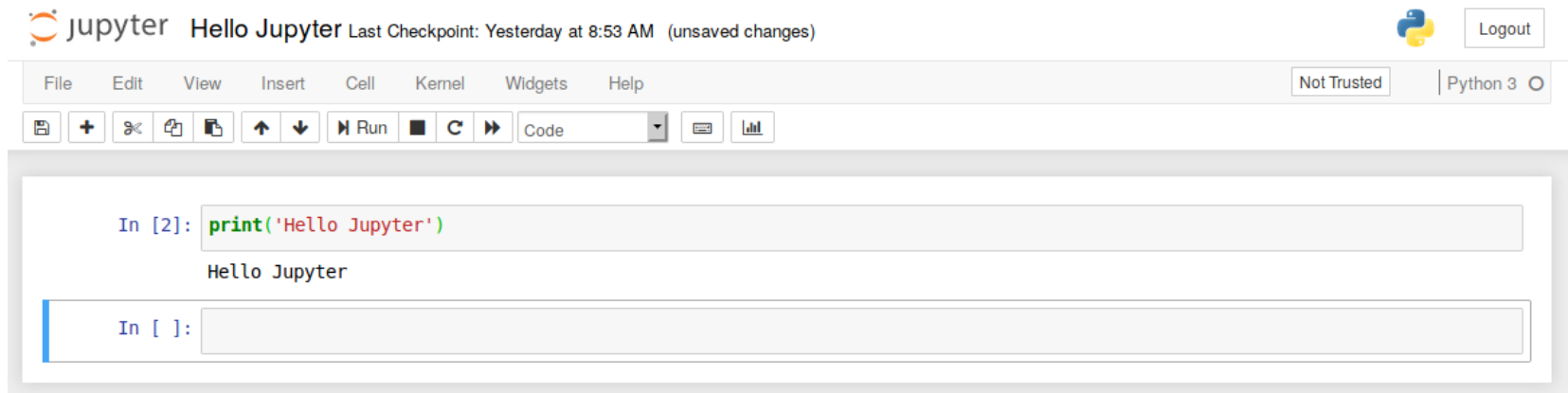
Creating a Notebook

- Click on the New button (upper right), and it will open up a list of choices. Let's choose Python 3



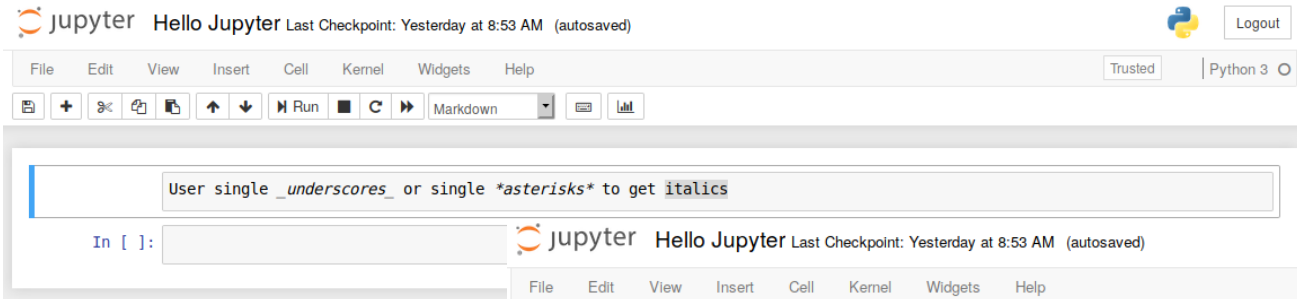
Notebook Structure

- Notebooks are organized into cells, which can contain code, text, or visualizations
- Run each cell by pressing the 'Run' button or Shift + Enter to execute the code and see results immediately

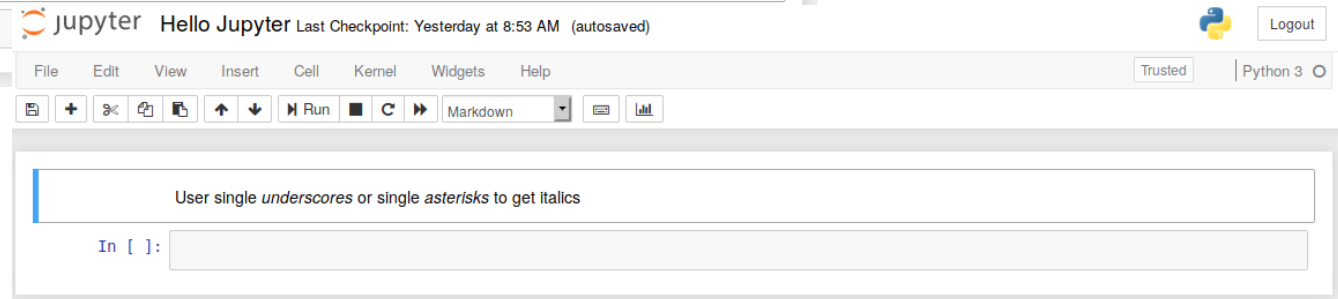


Writing Text with Markdown

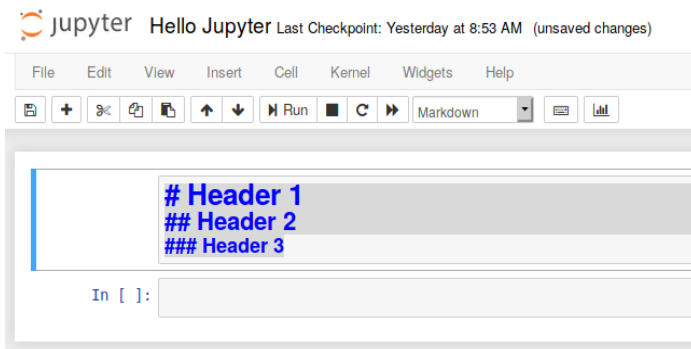
- Use markdown cells to add headings, lists, and formatted text to explain your code



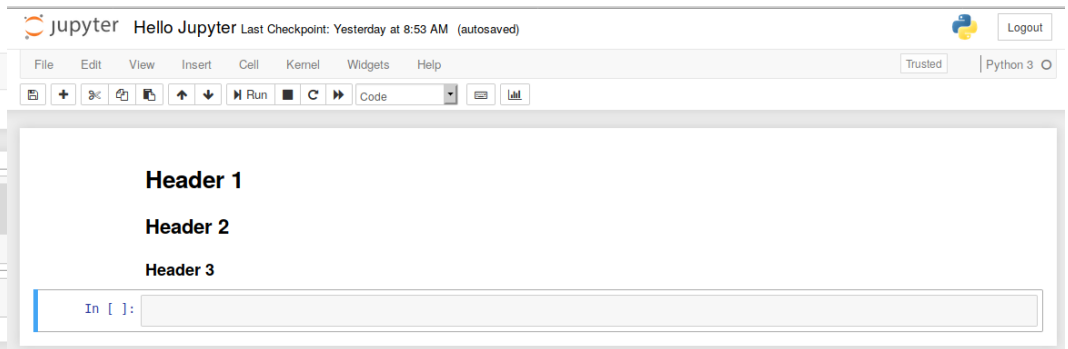
The image shows a Jupyter Notebook interface. The top bar includes the Jupyter logo, the text "Hello Jupyter", and "Last Checkpoint: Yesterday at 8:53 AM (autosaved)". The menu bar contains "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The toolbar has icons for saving, adding, deleting, and running cells, along with a dropdown menu currently set to "Markdown". The main area contains a single Markdown cell with the text: "User single underscores or single *asterisks* to get *italics*". Below the cell is a prompt "In []:".



This image is a second instance of the Jupyter Notebook interface, identical to the one above, showing a Markdown cell with the text: "User single underscores or single *asterisks* to get *italics*".



The image shows a Jupyter Notebook interface. The top bar includes the Jupyter logo, the text "Hello Jupyter", and "Last Checkpoint: Yesterday at 8:53 AM (unsaved changes)". The menu bar contains "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The toolbar has icons for saving, adding, deleting, and running cells, along with a dropdown menu currently set to "Markdown". The main area contains a single Markdown cell with the text: "# Header 1", "## Header 2", and "### Header 3". Below the cell is a prompt "In []:".



This image shows the rendered output of the Jupyter Notebook from the previous block. The top bar is identical, but the dropdown menu is now set to "Code". The main area displays the rendered headers: "Header 1" (large bold), "Header 2" (medium bold), and "Header 3" (small bold). Below the rendered text is a prompt "In []:".

Visualizing Data

- Jupyter Notebooks can display charts and plots right next to your code output

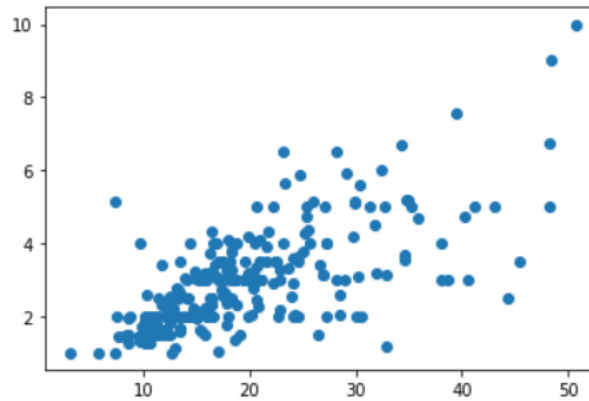
```
In [1]: import pandas as pd  
import matplotlib.pyplot as plt
```

```
In [2]: tips_data = pd.read_csv("tips.csv")  
print(tips_data.head())
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

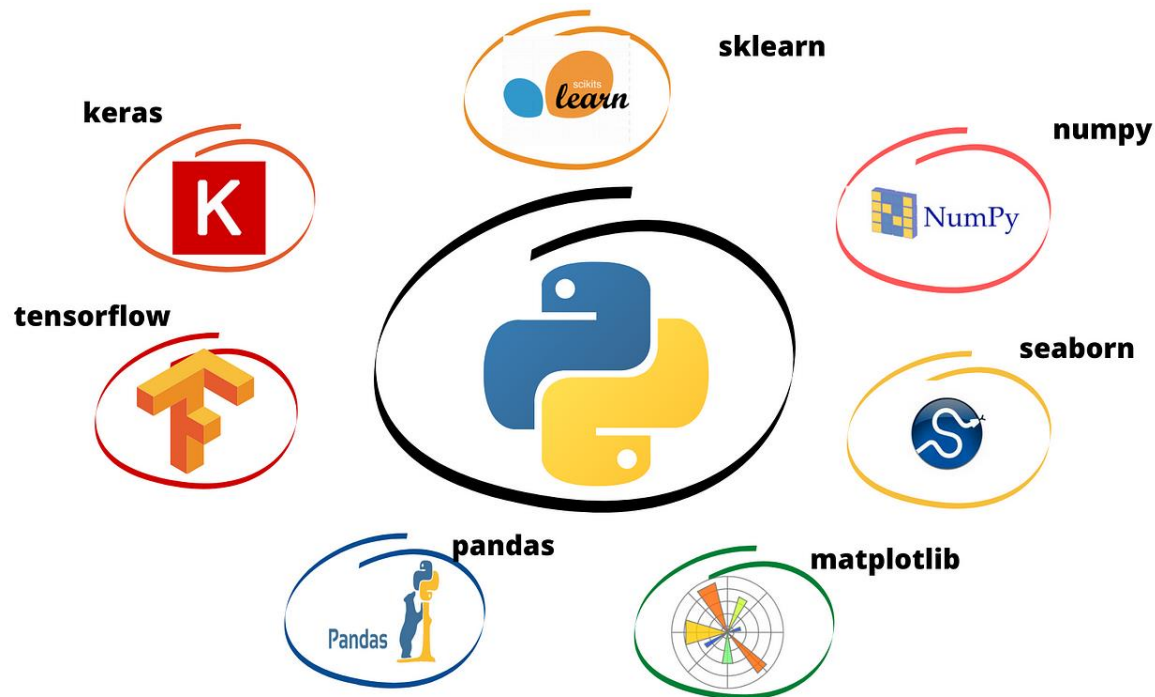
```
In [3]: plt.scatter(tips_data['total_bill'], tips_data['tip'])
```

```
Out[3]: <matplotlib.collections.PathCollection at 0x259d905c6d8>
```



Data Analysis and ML Libraries in Python

- Plethora of libraries to manipulate, analyze, and model data for data science
- Let's start with three key libraries: NumPy, Pandas, and Scikit-learn



NumPy – Handling Numerical Data

- Fast operations on arrays and matrices
- Tools for mathematical and statistical calculations



```
import numpy as np
```

NumPy Example

- Some samples on how to create NumPy arrays

```
[3] a = np.array([1, 2, 3]) # Create a rank 1 array
```

`np.array([1,2,3])`



1
2
3

`np.array([[1,2],[3,4]])`



1	2
3	4

`np.array([[[1,2],[3,4]],
 [[5,6],[7,8]]])`



		5	6
1	2		
3	4	8	

Numpy Notebook

- Feel free to explore more on Numpy!!
- You can find the hands-on [notebook for numpy](#) in our Github repository.

Pandas – Working with DataFrames

- Simplifies working with structured data
- Facilitates loading, manipulating, and analyzing tabular datasets



Pandas Data Structures

- The primary two components of pandas are the Series and DataFrame
- A Series is essentially a column, and a DataFrame is a multi-dimensional table made up of a collection of Series

Series		Series		DataFrame		
	apples		oranges		apples	oranges
0	3	+	0	=	0	0
1	2		3		1	3
2	0		7		2	7
3	1		2		3	2

Pandas Example

- An example of reading data from a csv file and loading into a pandas dataframe.

```
[1]: import pandas as pd
```

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

```
[1]:
```

	Name	Age	City	Country
0	Alice	28	New York	USA
1	Bob	22	Los Angeles	USA
2	Charlie	35	London	UK
3	David	30	Toronto	Canada

```
[2]: df.describe()
```

```
[2]:
```

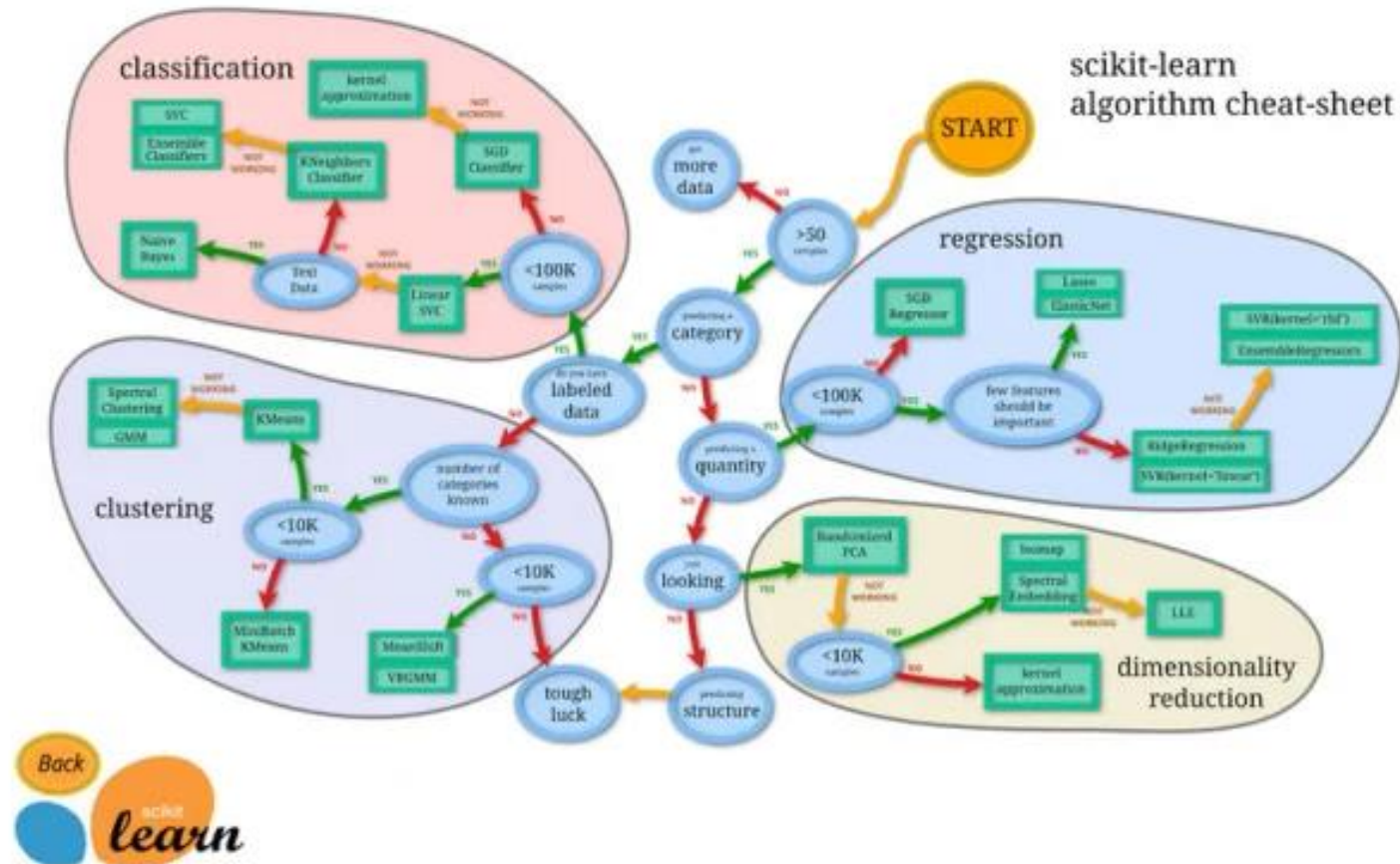
	Age
count	4.000000
mean	28.750000
std	5.377422
min	22.000000
25%	26.500000
50%	29.000000
75%	31.250000
max	35.000000

Pandas Notebook

- Feel free to explore more on Pandas!!
- You can find the hands-on [notebook for pandas](#) in our Github repository.

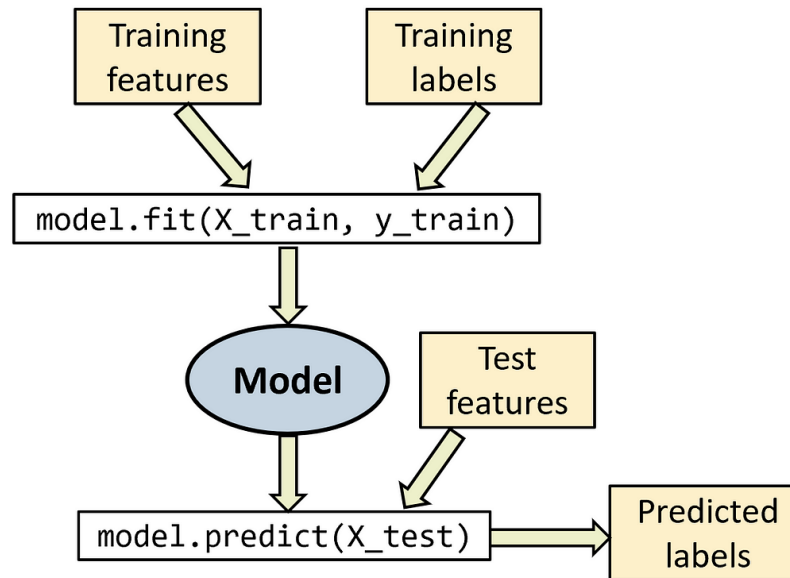
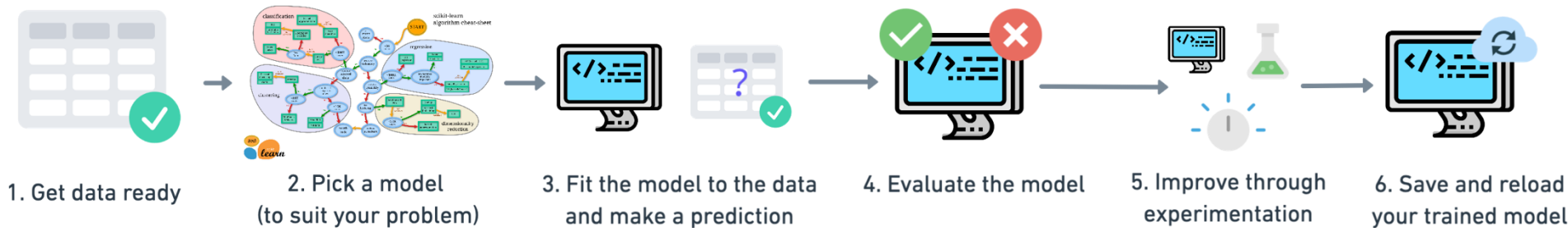
Scikit-learn – Machine Learning Made Simple

- Tools for building ML models, such as classification and regression



Scikit-learn

- Basic workflow of scikit-learn



Scikit-learn Example

- As an example dataset, heart-disease dataset
- Patient medical records and whether or not they have heart disease or not

```
import pandas as pd
heart_disease = pd.read_csv('../data/heart-disease.csv')
heart_disease.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Scikit-learn Example

- The target column indicates whether the patient has heart disease (target=1) or not (target=0), this is our "label" column, the variable we're going to try and predict

```
# Create X (all the feature columns)
X = heart_disease.drop("target", axis=1)

# Create y (the target column)
y = heart_disease["target"]

# Check the head of the features DataFrame
X.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2

Scikit-learn Example

- Split the dataset and create the model

```
# Split the data into training and test sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    Y,
                                                    test_size=0.25) # by default train_test_split uses 25% of the

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
# Since we're working on a classification problem, we'll start with a RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()
```

```
clf.fit(X=X_train, y=y_train)
```

```
# Use the model to make a prediction on the test data (further evaluation)
y_preds = clf.predict(X=X_test)
```

Scikit-learn Example

- Evaluation

```
# Evaluate the model on the test set
test_acc = clf.score(X=X_test, y=y_test)
print(f"The model's accuracy on the testing dataset is: {test_acc*100:.2f}%")
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Create a classification report
print(classification_report(y_test, y_preds))
```

	precision	recall	f1-score	support
0	0.81	0.60	0.69	35
1	0.72	0.88	0.79	41
accuracy			0.75	76
macro avg	0.76	0.74	0.74	76
weighted avg	0.76	0.75	0.74	76

```
# Create a confusion matrix
conf_mat = confusion_matrix(y_test, y_preds)
conf_mat
```

```
array([[21, 14],
       [ 5, 36]])
```

```
# Compute the accuracy score (same as the score() method for classifiers)
accuracy_score(y_test, y_preds)
```

```
0.75
```

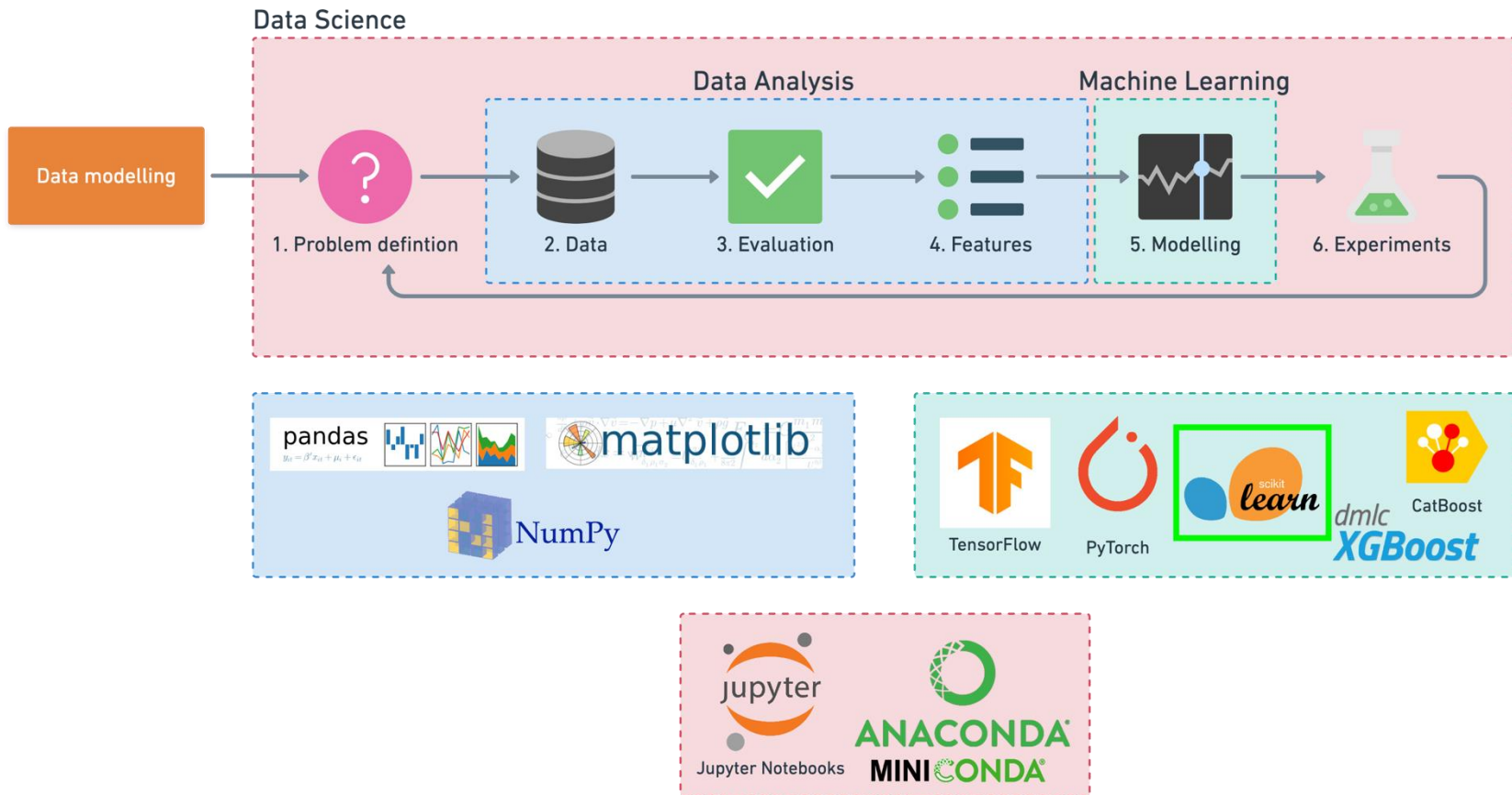
Scikit-learn Notebook

- Feel free to explore more on Scikit-learn!!
- You can find the hands-on [notebook for Scikit-learn](#) in our Github repository.

Putting It All Together

- NumPy for numerical operations, Pandas for data manipulation, and Scikit-learn for ML

Tools you can use



Off to Module 2

- Any questions??

