BigSem: Big Data Analytics for Semantic Data Tutorial

Module 1: Libraries for analytics and ML in Python

Chelmis Charalampos, Bedirhan Gergin
University at Albany, SUNY

ISWC 2024

goo.gl/6Nrc1r

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.

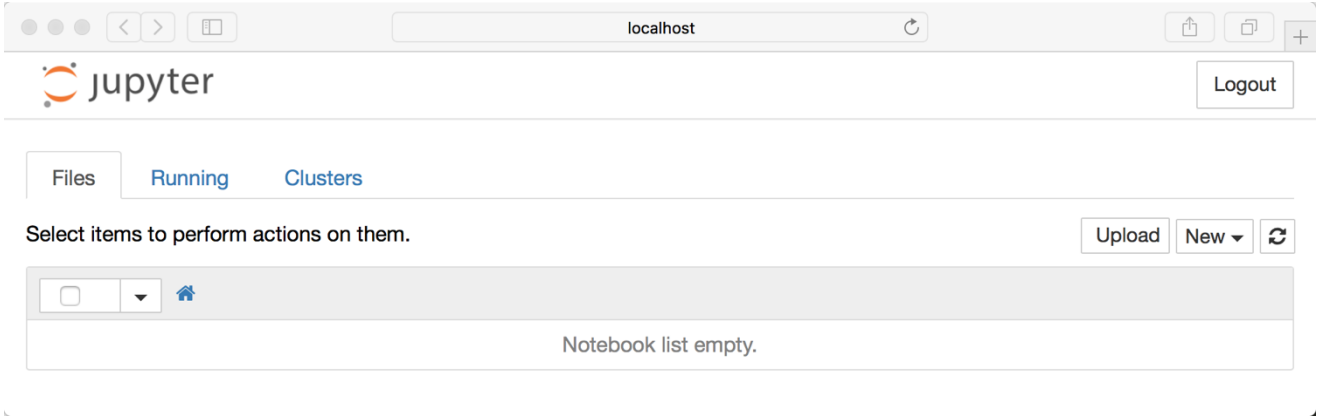
Shell

```
$ pip install jupyter
```

Shell

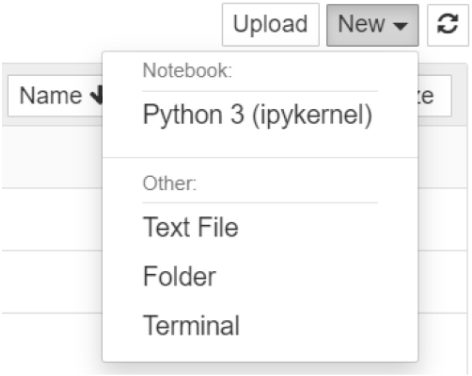
```
$ jupyter notebook
```

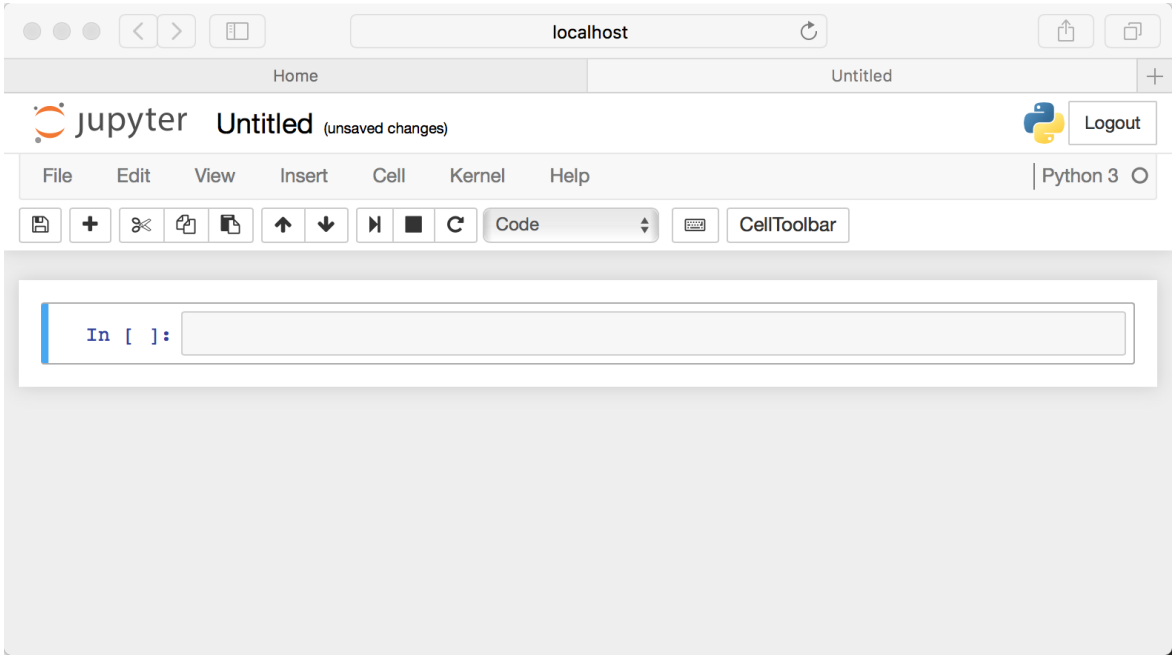
- This will start up Jupyter and your default browser should start (or open a new tab) to the following URL: http://localhost:8888/tree



Creating a Notebook

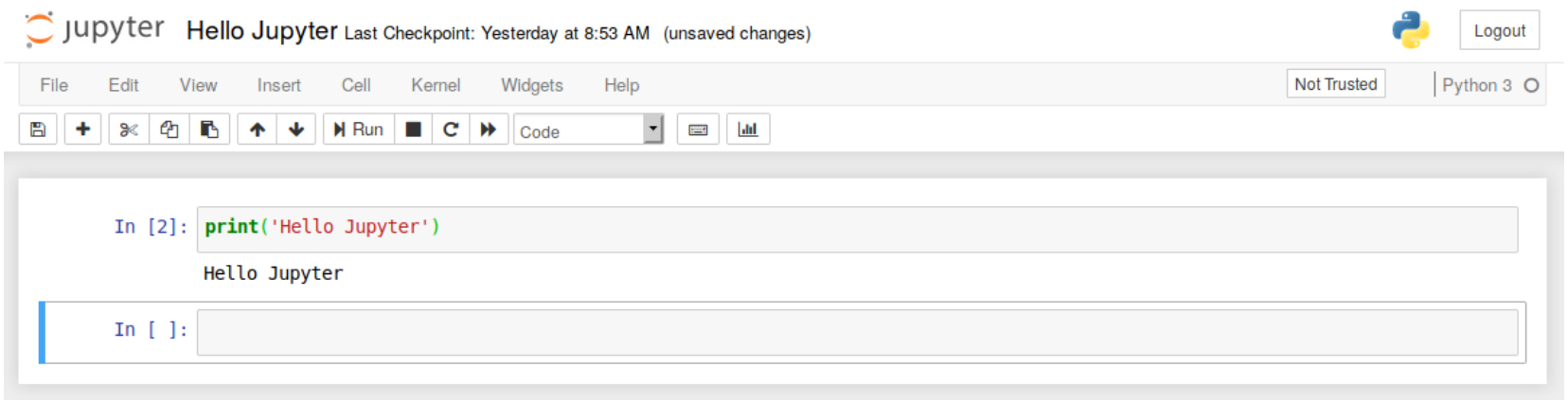
- Click on the New button (upper right), and it will open up a list of choices. Let's choose Python 3.





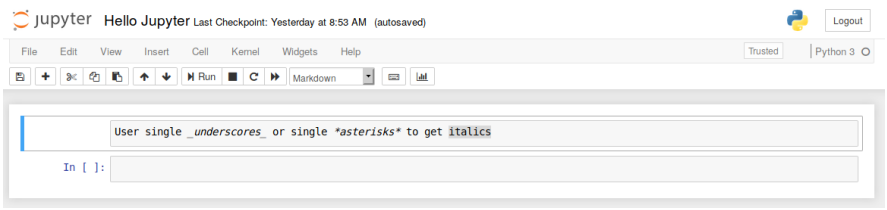
Notebook Structure

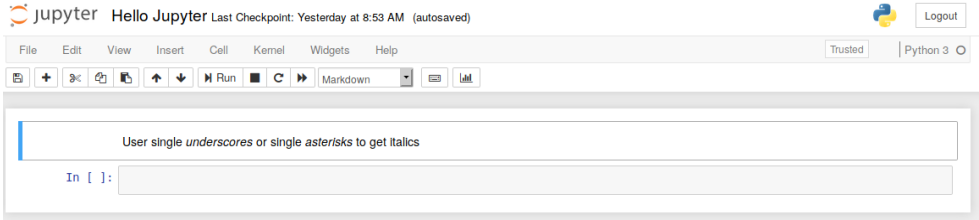
- Notebooks are organized into cells, which can contain code, text, or visualizations.
- Run each cell by pressing the ‘Run’ button or Shift + Enter to execute the code and see results immediately.

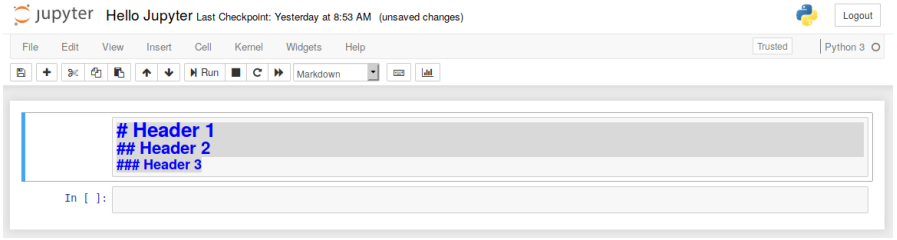


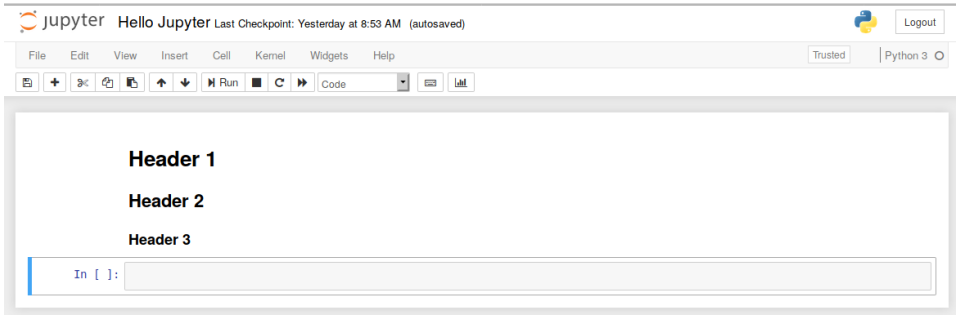
Writing Text with Markdown

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









Visualizing Data

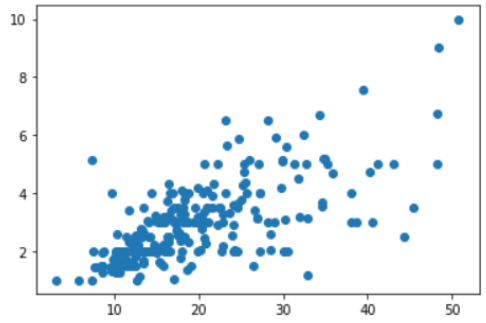
- Jupyter Notebooks can display charts and plots right next to your code output.

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

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

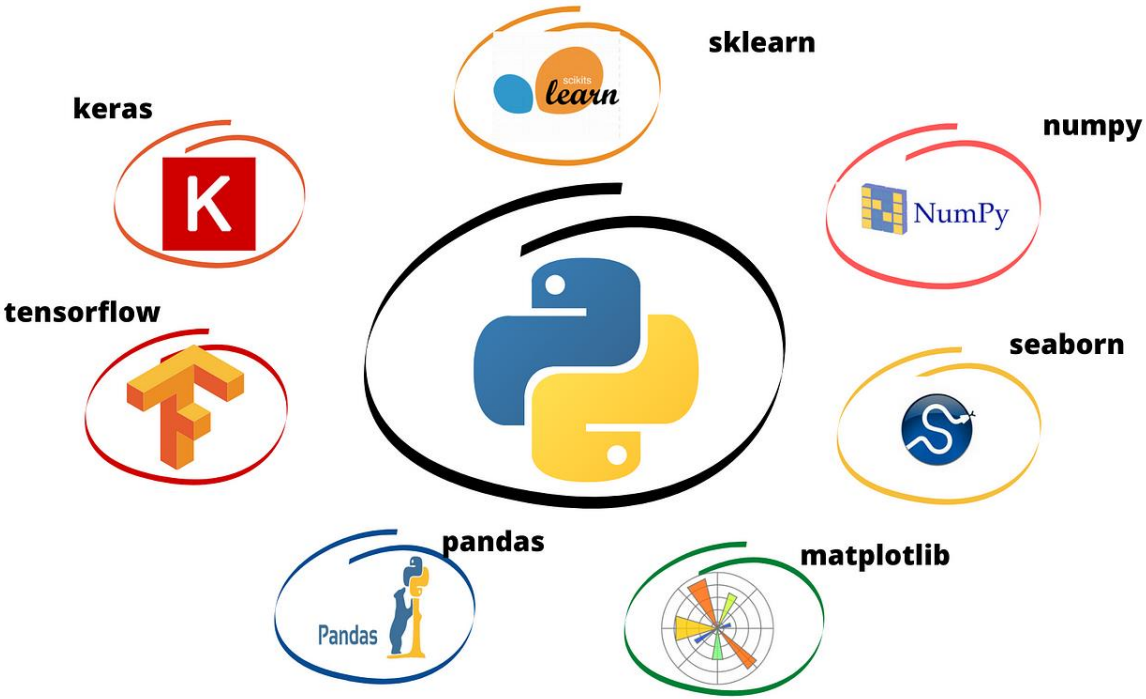
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
In [3]: plt.scatter(tips_data['total_bill'], tips_data['tip'])
Out[3]: <matplotlib.collections.PathCollection at 0x259d905c6d8>
```



Python: Data Analysis and ML Libraries

- In data science, we use powerful libraries to manipulate, analyze, and model data.
- Let's start with three key libraries: Pandas, NumPy, and Scikit-learn.



NumPy – Handling Numerical Data

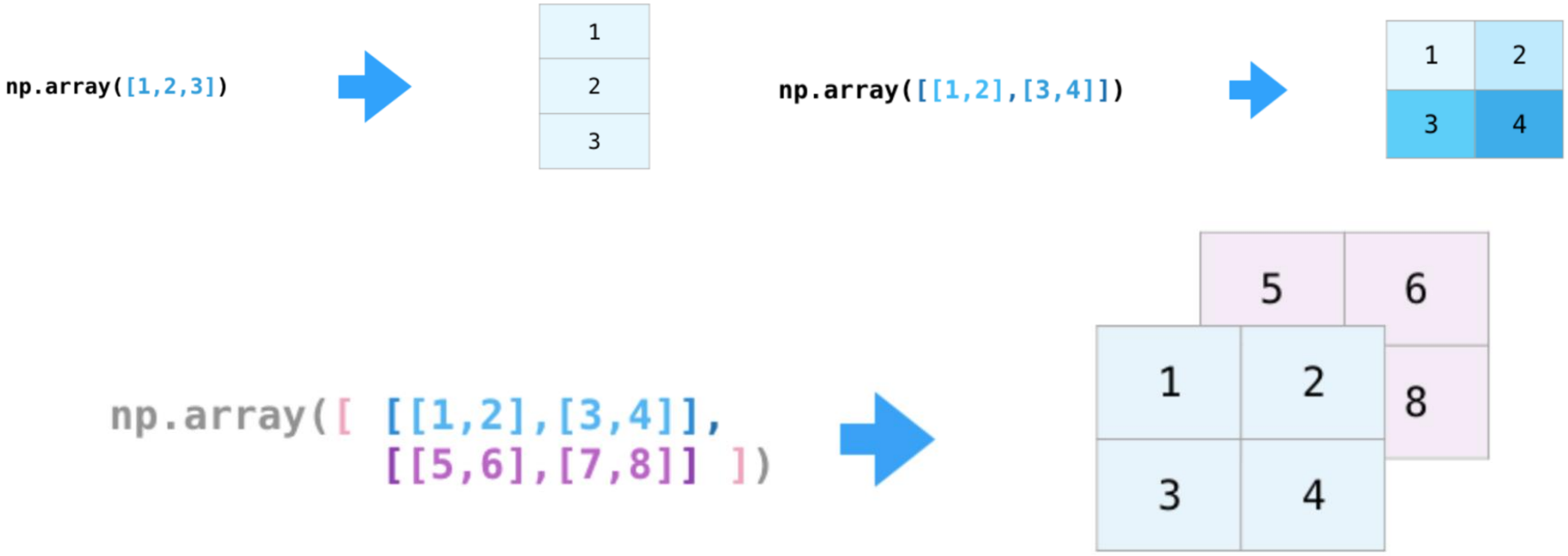
- NumPy is used for fast operations on arrays and matrices, providing tools for mathematical and statistical calculations.


```
import numpy as np
```

NumPy Example

- Here's some samples to create NumPy arrays

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

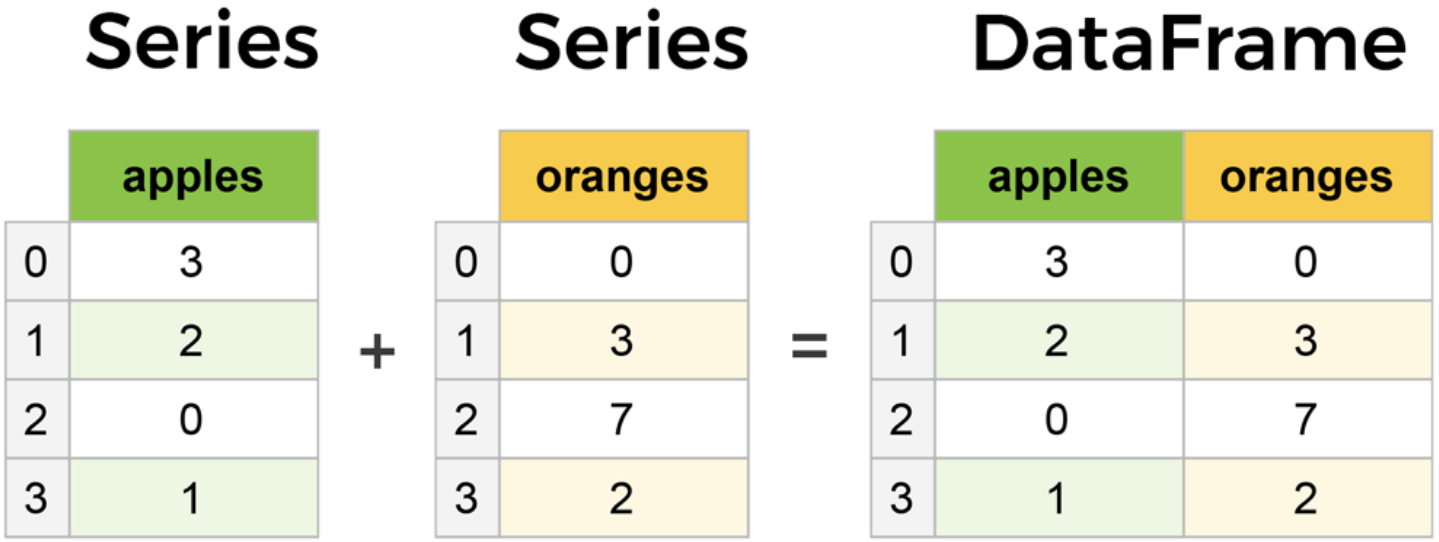


Pandas – Working with DataFrames

- Pandas makes working with structured data easy, allowing you to load, manipulate, and analyze 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.



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

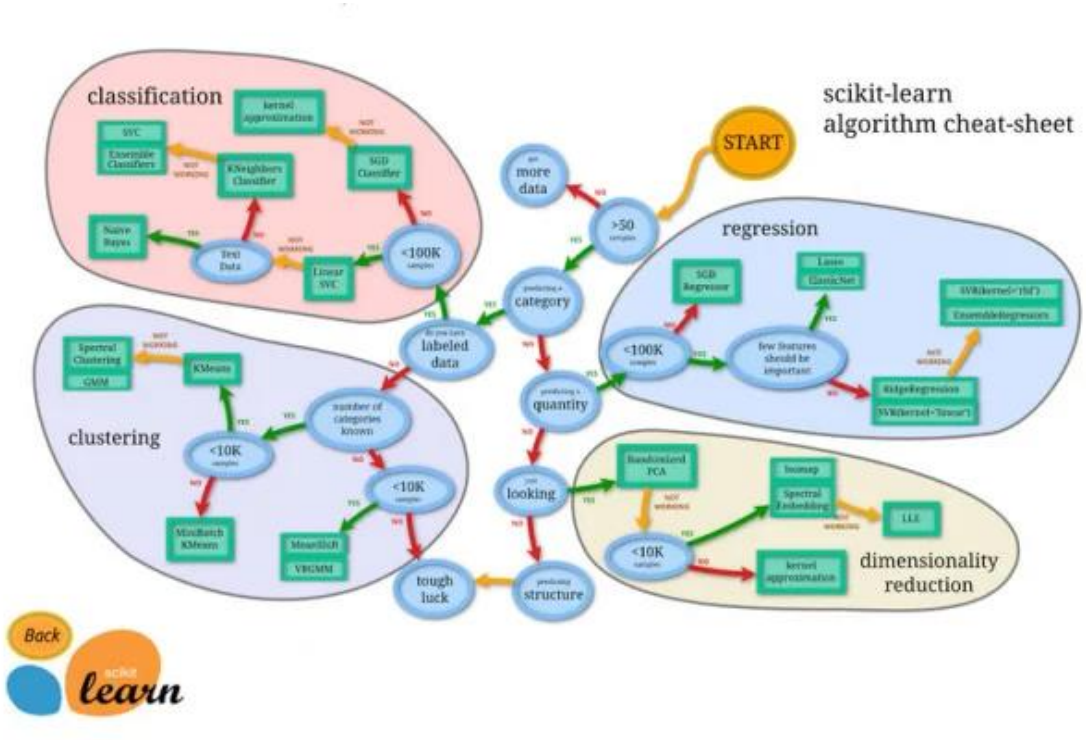
```
[2]: df.describe()
```

[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

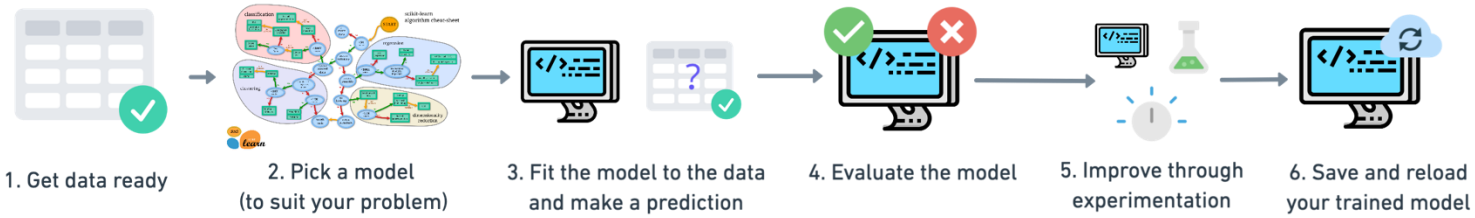
Scikit-learn – Machine Learning Made Simple

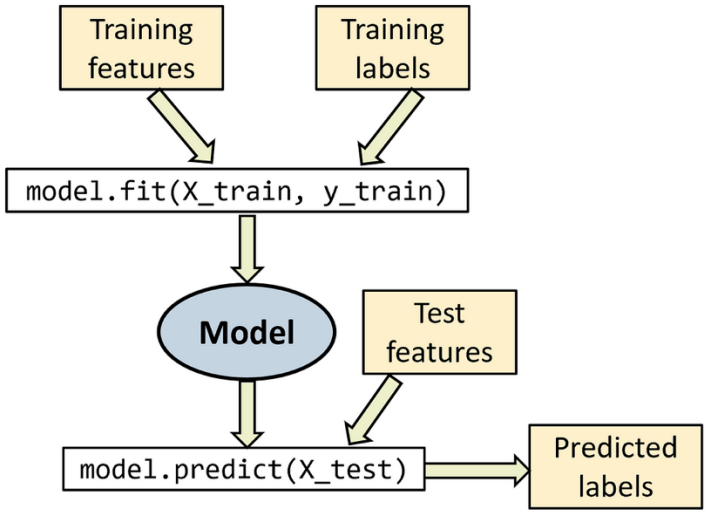
- Scikit-learn provides tools for building machine learning models like classification and regression.



Scikit-learn

- Basic workflow of scikit-learn.





Scikit-learn Example

- As an example dataset, we'll import heart-disease.csv.
- This file contains anonymized patient medical records and whether or not they have heart disease or not (this is a classification problem since we're trying to predict whether something is one thing or another).

```
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
```

Putting It All Together

- Combine NumPy for numerical operations, Pandas for data manipulation, and Scikit-learn for machine learning to build full data science workflows.