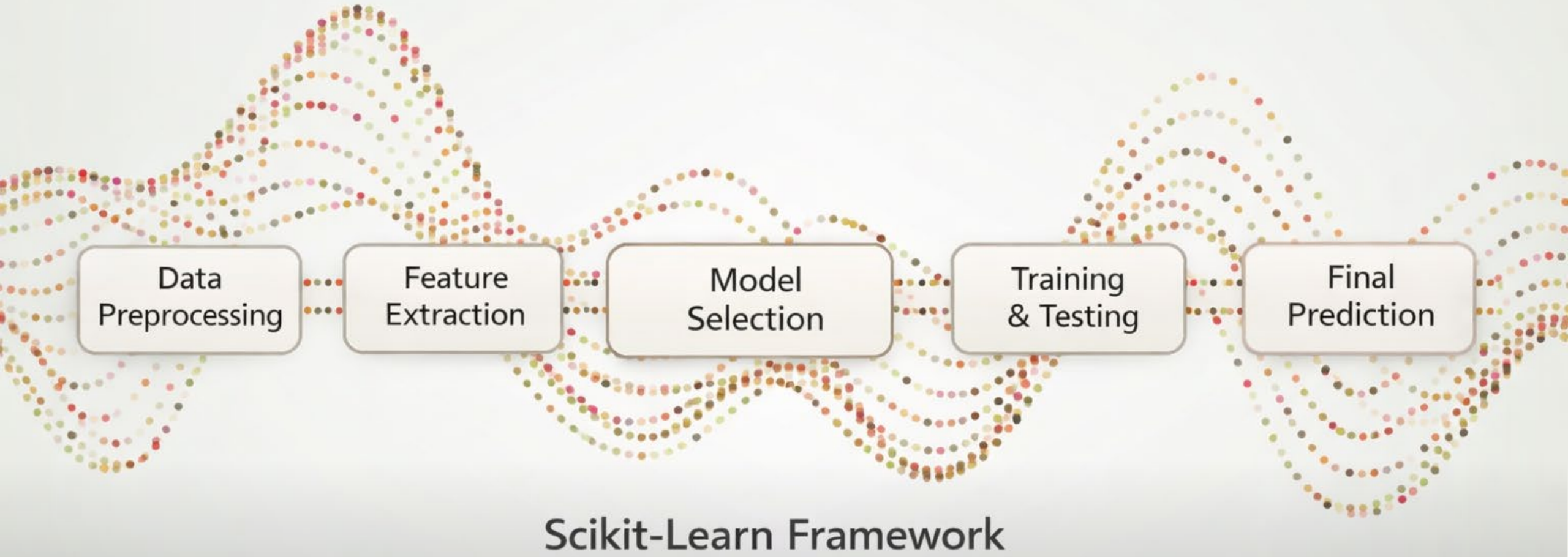

Scikit-Learn Framework

INTRODUCTION TO SCIKIT-LEARN

WHAT IS SCIKIT-LEARN

- Scikit-Learn is one of the best-known Python packages for machine learning
- scikit-learn.org
- It provides efficient versions of many common ML algorithms
- It has a clean, uniform, and streamlined API
  - Once you understand the syntax of Scikit-Learn, it is straightforward to use any model inside the library
- In Python, Scikit-Learn is called with `sklearn`
  - **`import sklearn`**

SCIKIT-LEARN DESIGN

- **Consistency**:
  * All objects share a consistent and sensible interface

- **Inspection**:
  * All hyperparameter and learned parameters are directly accessible (as attributes of the object)
  * Learn parameters have an underscore suffix

- **Nonproliferation of classes**:
  * Datasets and hyperparameters are Pandas, Numpy, or base Python objects

- **Composition**:
  * Existing building blocks are reused as much as possible

- **Sensible defaults**:
  * Most parameters have reasonable default values

SCIKIT-LEARN DESIGN: CONSISTENCY

- **Estimators**:
  - Any object that can estimate some parameter based on a dataset
  - The `.fit()` method performs the estimation
- **Transformers**:
  - Estimators that can also transform a dataset
  - Transformation is performed with the `.transform()` method
  - The `.fit_transform()` method fits and transforms in one step
  - Example: `StandardScaler`
- **Predictors**:
  - Estimators that are capable of making predictions
  - Has a `.predict()` method
  - Usually have a built-in `.score()` method
  - Example: `KNeighborsClassifier`

OBJECT-ORIENTED MODELING

- A model is an **object**:
  - It has **state** (learned parameters)
  - It has **methods** (fit, predict)
- This allows:
  - Reuse
  - Inspection
  - Composition (pipelines)

```
from sklearn.linear_model import
LinearRegression

# Model instantiation
model = LinearRegression()

#Model fit
model.fit(X, y)
model.predict(X_new)

#Model learned parameters:
coefficients = model.coef_
```

SIMPLE SCIKIT-LEARN USAGE

1. Prepare the data into a feature matrix and a target vector
2. Choose a model and import the estimator ***class*** from Scikit-Learn
3. Instantiate the model class and choose ***hyperparamters***
   (not all models classes have hyperparameters)
4. Fit the model by calling the `.fit()` method
5. Apply the model with the `.predict()` or the `.transform()` method

Class: Scikit-Learn makes extensive use of Python classes and object oriented programming principles

Hyperparameters: *Also called tuning parameters are parameters of the learning algorithm and not the model. That is, its value is used to control the learning process and is chosen by the modeler rather than learned from the data.*

PREPARING YOUR DATA

- The data should be in tabular form
    - Generally, Numpy or Pandas objects
    - The rows should be the observations or samples
- The feature matrix (X) should be a 2D Numpy array or a Pandas DataFrame
    - This applies even when there is only 1 feature
    - **Note**: if only one feature, it often must be reshaped using `.reshape(-1, 1)`
- The target (Y) should be a 1D Numpy array or a Pandas Series

EXAMPLE

```
import numpy as np
from sklearn.linear_model import LinearRegression


# Create simple data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)  # feature matrix (n_samples, n_features)
y = np.array([2, 4, 6, 8, 10])                # target vector


# Instantiate and fit the model
model = LinearRegression()
model.fit(X, y)


# Make predictions
y_pred = model.predict(X)


# Inspect learned parameters
print("Slope:", model.coef_[0])
print("Intercept:", model.intercept_)
```

BECOME FRIENDS WITH THE DOCUMENTATION

- A large part of learning Scikit-Learn is getting comfortable with the documentation
- The documentation is vast and sometimes intimidating but learning how to read it will help you know how to implement the right model for your situation
- https://scikit-learn.org/stable/