Machine Learning in Python

Software Engineering for AI systems - DAT821

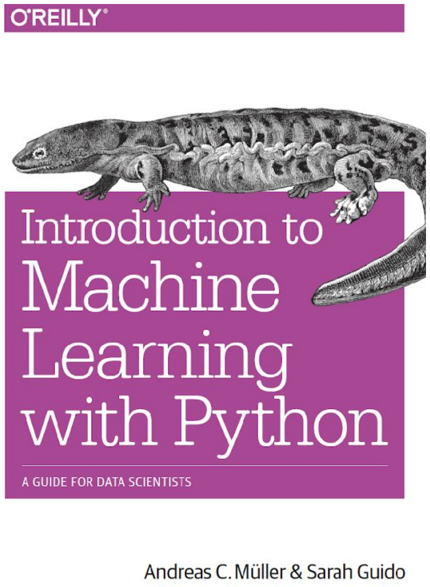
AI system lifecycle

Objectives

Learn how subtasks of machine learning map unto Python

- **Understand** and **visualize** your training data before training
- **Apply** ML algorithm on your training data
- **Evaluate** your ML model on your test data

Great References BOOKS & INTERNET


O'REILLY
Introduction to Machine Learning with Python
A GUIDE FOR DATA SCIENTISTS
Andreas C. Müller & Sarah Guido

Python documentation
https://docs.python.org/3/

Python tutorials

- *Learn Python: step-by-step tutorial* (Link)
- *Several others e.g., listed here* (Link)

Machine learning with Python tutorials

- Scipy- lectures (Link)
- Others - *search machine learning with Python*

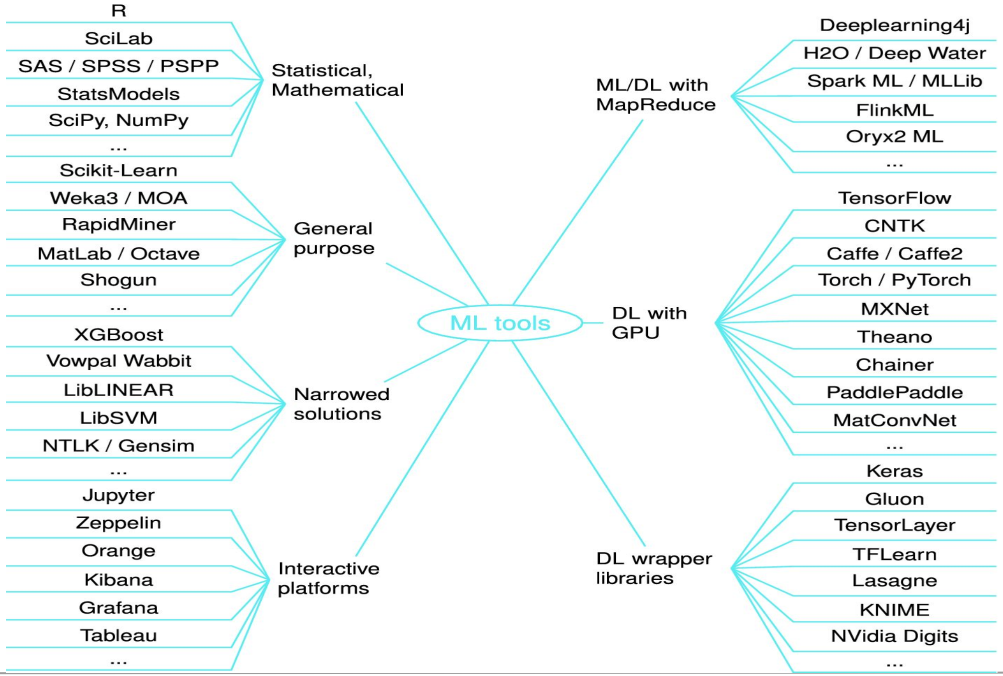
What is, and why Python?

- Python is a dynamic, interpreted language
- Has no type declarations of variables, parameters, functions, or methods in source code
- It tracks the types of all values at runtime and flags code that does not make sense as it runs
- Has a growing and dominant ecosystem for machine learning

What is, and why Python?

Overview of Machine Learning frameworks and libraries

The number of ML algorithms and their software implementation is quite large



ML tools

Statistical, Mathematical: R, SciLab, SAS / SPSS / PSPP, StatsModels, SciPy, NumPy, ...

General purpose: Scikit-Learn, Weka3 / MOA, RapidMiner, MatLab / Octave, Shogun, ...

Narrowed solutions: XGBoost, Vowpal Wabbit, LibLINEAR, LibSVM, NTLK / Gensim, ...

Interactive platforms: Jupyter, Zeppelin, Orange, Kibana, Grafana, Tableau, ...

ML/DL with MapReduce: Deeplearning4j, H2O / Deep Water, Spark ML / MLLib, FlinkML, Oryx2 ML, ...

DL with GPU: TensorFlow, CNTK, Caffe / Caffe2, Torch / PyTorch, MXNet, Theano, Chainer, PaddlePaddle, MatConvNet, ...

DL wrapper libraries: Keras, Gluon, TensorLayer, TFLearn, Lasagne, KNIME, NVidia Digits, ...

Nguyen, et al., (2019). Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey. Artificial Intelligence Review, 52(1), 77-124
scipy-lectures

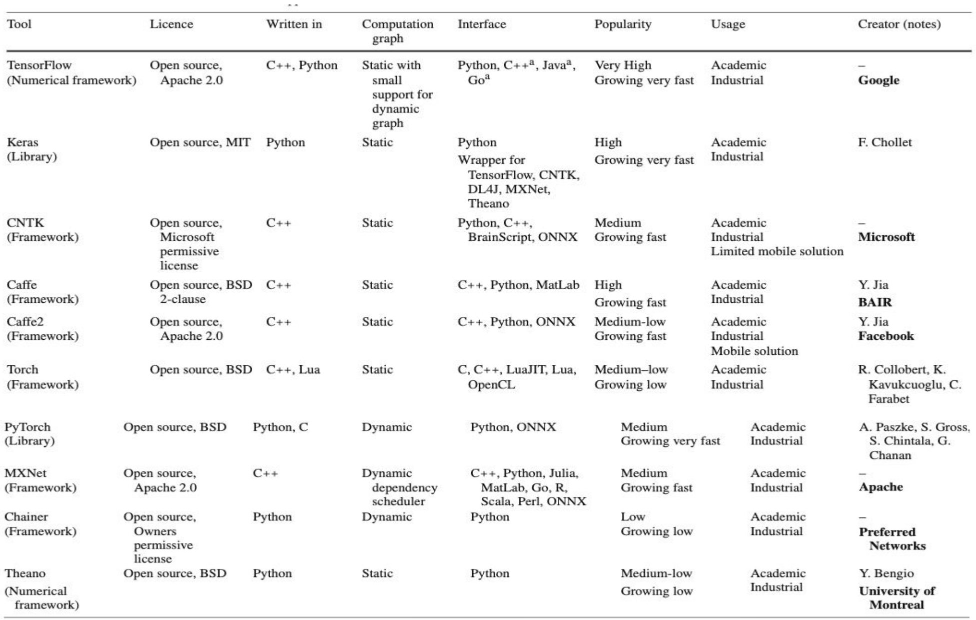
What is, and why Python?

Overview of Machine Learning frameworks and libraries

Findings from a survey:

- Python is the most popular programming language for data mining, Machine Learning and Deep Learning applications

- The majority of frameworks and libraries are either Python based or support Python interfaces



Tool	Licence	Written in	Computation graph	Interface	Popularity	Usage	Creator (notes)
TensorFlow (Numerical framework)	Open source, Apache 2.0	C++, Python	Static with small support for dynamic graph	Python, C++^{a}, Java^a, Go^a	Very High Growing very fast	Academic Industrial	– **Google**
Keras (Library)	Open source, MIT	Python	Static	Python Wrapper for TensorFlow, CNTK, DL4J, MXNet, Theano	High Growing very fast	Academic Industrial	F. Chollet
CNTK (Framework)	Open source, Microsoft permissive license	C++	Static	Python, C++, BrainScript, ONNX	Medium Growing fast	Academic Industrial Limited mobile solution	– **Microsoft**
Caffe (Framework)	Open source, BSD 2-clause	C++	Static	C++, Python, MatLab	High Growing fast	Academic Industrial	Y. Jia **BAIR**
Caffe2 (Framework)	Open source, Apache 2.0	C++	Static	C++, Python, ONNX	Medium-low Growing fast	Academic Industrial Mobile solution	Y. Jia **Facebook**
Torch (Framework)	Open source, BSD	C++, Lua	Static	C, C++, LuaJIT, Lua, OpenCL	Medium–low Growing low	Academic Industrial	R. Collobert, K. Kavukcuoglu, C. Farabet
PyTorch (Library)	Open source, BSD	Python, C	Dynamic	Python, ONNX	Medium Growing very fast	Academic Industrial	A. Paszke, S. Gross, S. Chintala, G. Chanan
MXNet (Framework)	Open source, Apache 2.0	C++	Dynamic dependency scheduler	C++, Python, Julia, MatLab, Go, R, Scala, Perl, ONNX	Medium Growing fast	Academic Industrial	– **Apache**
Chainer (Framework)	Open source, Owners permissive license	Python	Dynamic	Python	Low Growing low	Academic Industrial	– **Preferred Networks**
Theano (Numerical framework)	Open source, BSD	Python	Static	Python	Medium-low Growing low	Academic Industrial	Y. Bengio **University of Montreal**

Nguyen, et al., (2019). Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey. Artificial Intelligence Review, 52(1), 77-124
scipy-lectures

Python installation

- Check if Python is installed
- Installation guides Link
- Recommended installation
  - Anaconda Link
- Package installation
  - Using pip Link
  - Using conda Link
- Virtual environment
  - using venv Link
  - using conda Link

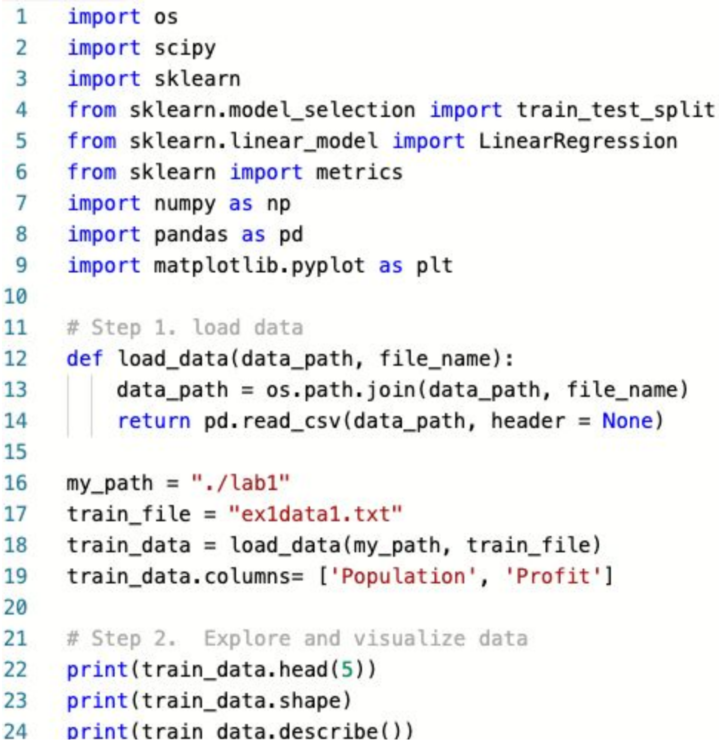
```
$ pip install numpy scipy
matplotib scikit-learn pandas
```

```
$ conda install numpy scipy
matplotib scikit-learn pandas
```

Python Packaging index (https://pypi.org/)

Python syntax

Example: Lab1 – Linear regression with one variable



```
import os
import scipy
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Step 1. load data
def load_data(data_path, file_name):
    data_path = os.path.join(data_path, file_name)
    return pd.read_csv(data_path, header = None)

my_path = "./lab1"
train_file = "ex1data1.txt"
train_data = load_data(my_path, train_file)
train_data.columns= ['Population', 'Profit']

# Step 2.  Explore and visualize data
print(train_data.head(5))
print(train_data.shape)
print(train_data.describe())
```

Pep8 – recommended style guideline for python code (https://www.python.org/dev/peps/pep-0008/#function-and-variable-names)

Python ecosystem for machine learning

- **NumPy** – For N-dimensional *array manipulation*
- **SciPy** – For advanced *mathematical routines*
- **Pandas** – For *data analysis* and *data structures*
- **Matplotlib** – For *2D plotting*
- **Scikit-learn** – For *machine learning* algorithms
- **PyTorch** – For *deep learning*

Python ecosystem for machine learning

Installation

Pip

```
$ pip install numpy scipy matplotib scikit-learn pandas
```

Conda

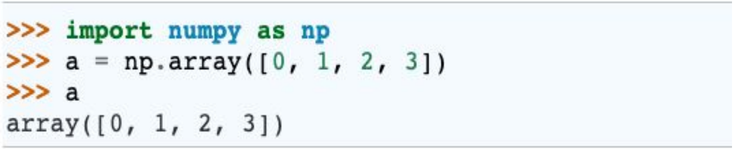
```
$ conda install numpy scipy matplotib scikit-learn pandas
```

NumPy

- A fundamental package for scientific computing in Python

- It contains functionality for multidimensional arrays, high-level mathematical functions such as linear algebra operations

- NumPy is a fundamental data structure on scikit-learn



```
>>> import numpy as np
>>> a = np.array([0, 1, 2, 3])
>>> a
array([0, 1, 2, 3])
```

Scipy-lectures

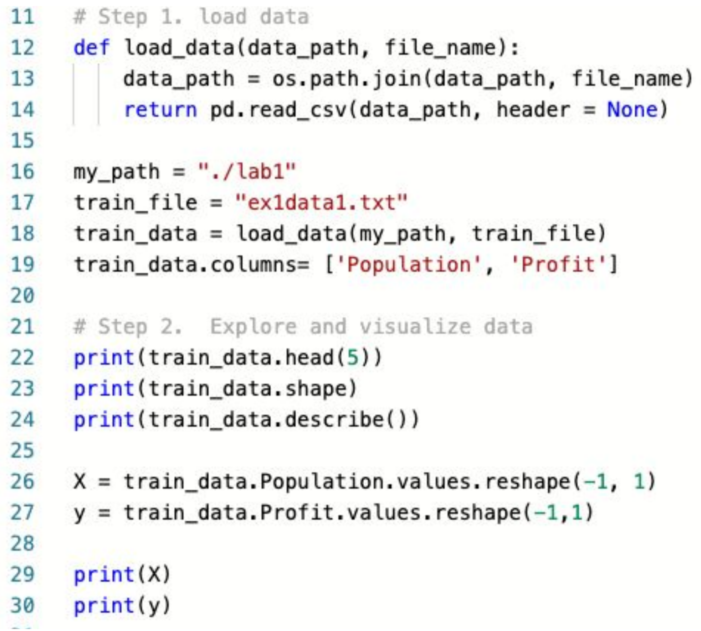
SciPy

- SciPy provides among other advanced linear algebra routines, mathematical function optimization, signal processing, special mathematical function and statistical distributions
- It is meant to operate efficiently on numpy arrays, so that numpy and scipy work hand in hand
- scikit-learn draws from SciPy’s collection of functions for implementing its algorithms

scipy- lectures

Pandas

- A library for data wrangling and analysing
- Built around a data structure called DataFrame, which is like a table (similar to Excel spreadsheet)
- Provides a wide range of methods to modify and operate on DataFrame
- Contrast to NumPy, pandas allows each column to have a separate data type
- Pandas gives the ability to ingest from a great variety of data files formats

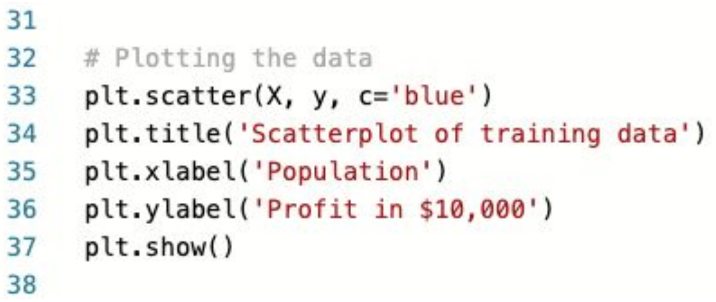


```python
# Step 1. load data
def load_data(data_path, file_name):
    data_path = os.path.join(data_path, file_name)
    return pd.read_csv(data_path, header = None)

my_path = "./lab1"
train_file = "ex1data1.txt"
train_data = load_data(my_path, train_file)
train_data.columns= ['Population', 'Profit']

# Step 2.  Explore and visualize data
print(train_data.head(5))
print(train_data.shape)
print(train_data.describe())

X = train_data.Population.values.reshape(-1, 1)
y = train_data.Profit.values.reshape(-1,1)

print(X)
print(y)
```

https://pypi.org/project/pandas/

Matplotlib

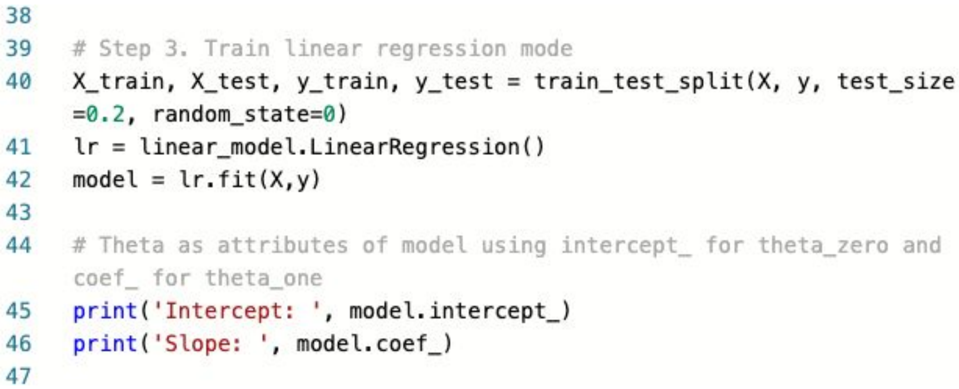
- Matplotlib is the primary scientific plotting library in Python
- It provides functions for making good quality visualizations, such as histograms, scatter plots etc.,
- Visualizations are important for giving insights into your data analysis



```

# Plotting the data
plt.scatter(X, y, c='blue')
plt.title('Scatterplot of training data')
plt.xlabel('Population')
plt.ylabel('Profit in $10,000')
plt.show()

```

Scikit-learn

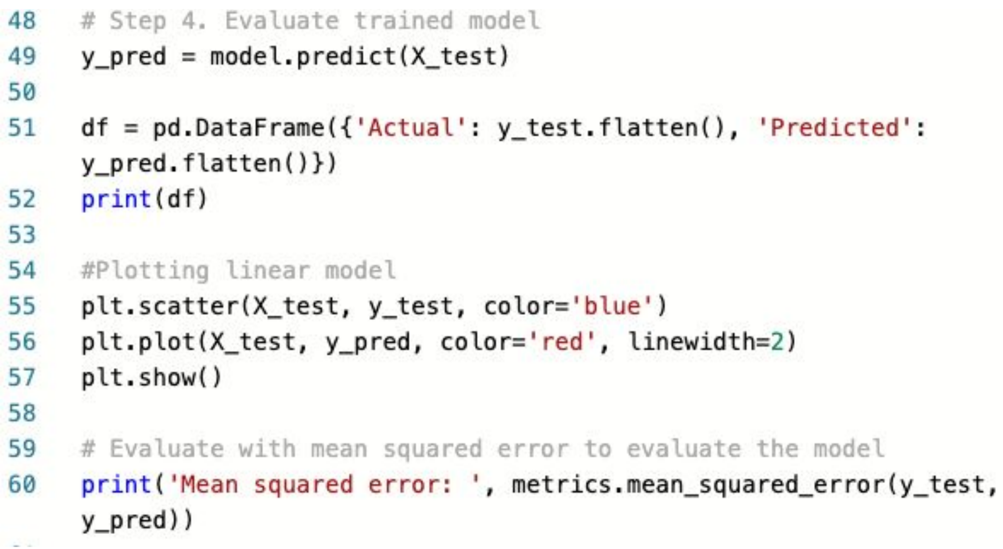
- Scikit-learn contains a number of machine learning algorithms for classification, regression, clustering ,as well as pre-processing
- Scikit-learn depends on two other Python packages NumPy and sciPy
- ML algorithms implemented in scikit-learn expect data to be stored in a **two-dimensional array or matrix** e.g., numpy arrays



```

# Step 3. Train linear regression mode
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
lr = linear_model.LinearRegression()
model = lr.fit(X,y)

# Theta as attributes of model using intercept_ for theta_zero and coef_ for theta_one
print('Intercept: ', model.intercept_)
print('Slope: ', model.coef_)

```

Scikit-learn

- Provides metrics functions for evaluating ML model performance



```
# Step 4. Evaluate trained model
y_pred = model.predict(X_test)

df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
print(df)

#Plotting linear model
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.show()

# Evaluate with mean squared error to evaluate the model
print('Mean squared error: ', metrics.mean_squared_error(y_test, y_pred))
```