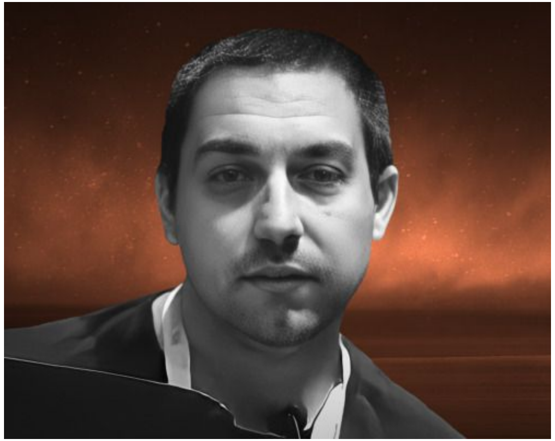
Tips and tricks for data science projects with Python

José Manuel Ortega
Python Developer



Jose Manuel Ortega

Software engineer,

Freelance

1. Introducing Python for machine learning projects
2. Stages of a machine learning project
3. Selecting the best python library for your project for each stage
4. Python tools for deep learning in data science projects

Introducing Python for machine learning projects

- Simple and consistent
- Understandable by humans
- General-purpose programming language
- Extensive selection of libraries and frameworks

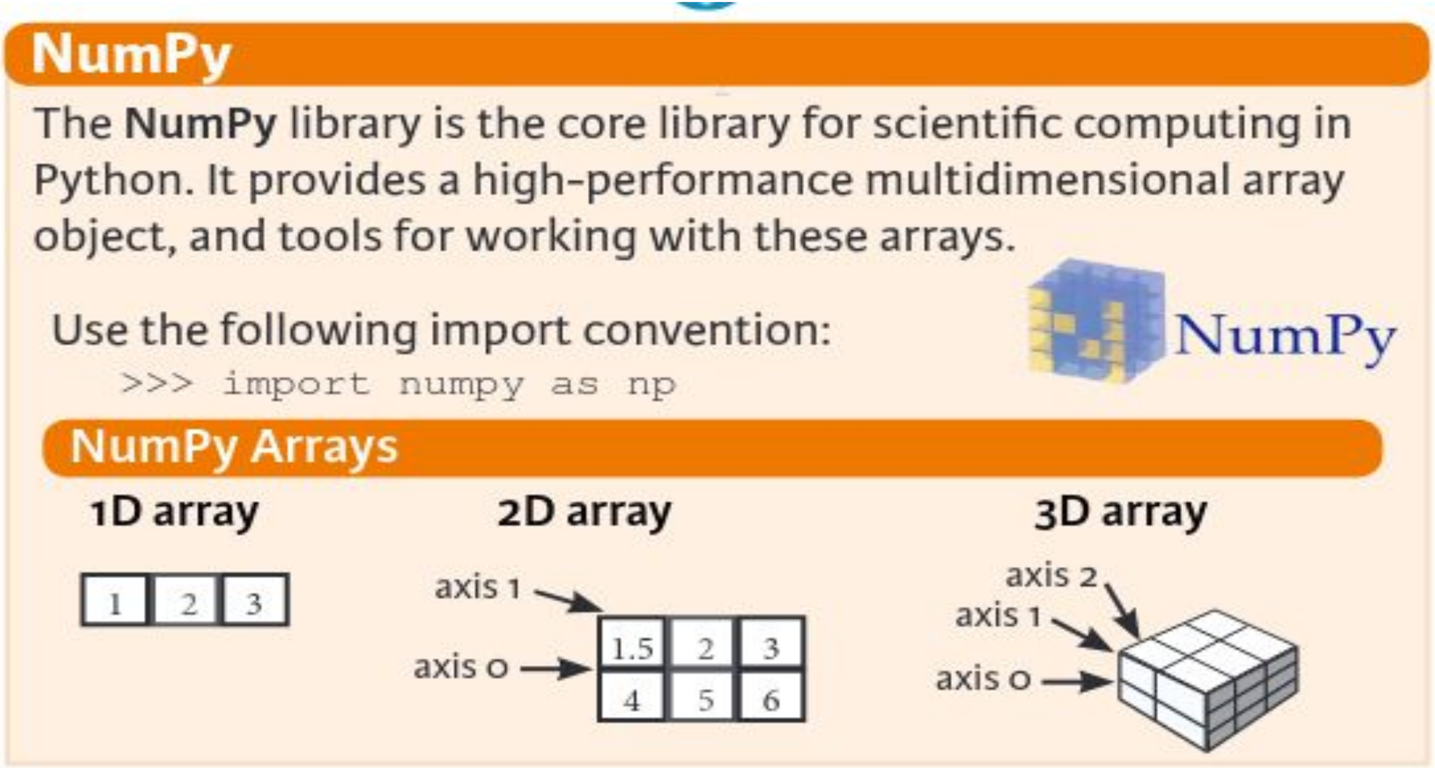
Introducing Python for machine learning projects

- Spam filters
- Recommendation systems
- Search engines
- Ppersonal assistants
- Fraud detection systems

Introducing Python for machine learning projects

● Machine learning	● Keras, TensorFlow, and Scikit-learn
● High-performance scientific computing	● Numpy, Scipy
● Computer vision	● OpenCV
● Data analysis	● Numpy, Pandas
● Natural language processing	● NLTK, spaCy

Introducing Python for machine learning projects



NumPy

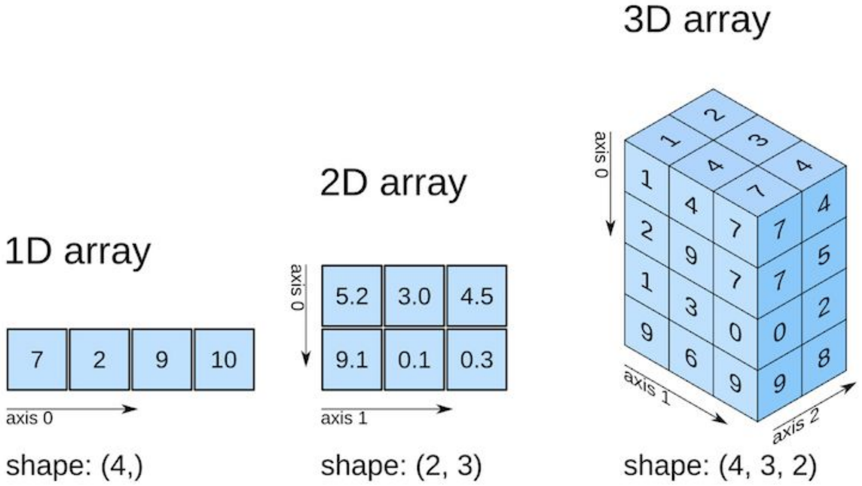
The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

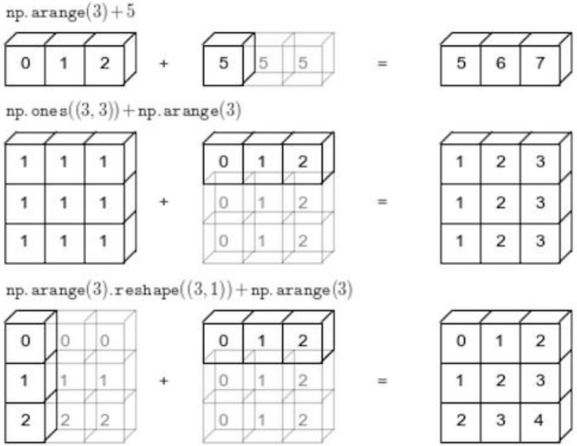
Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays

Introducing Python for machine learning projects





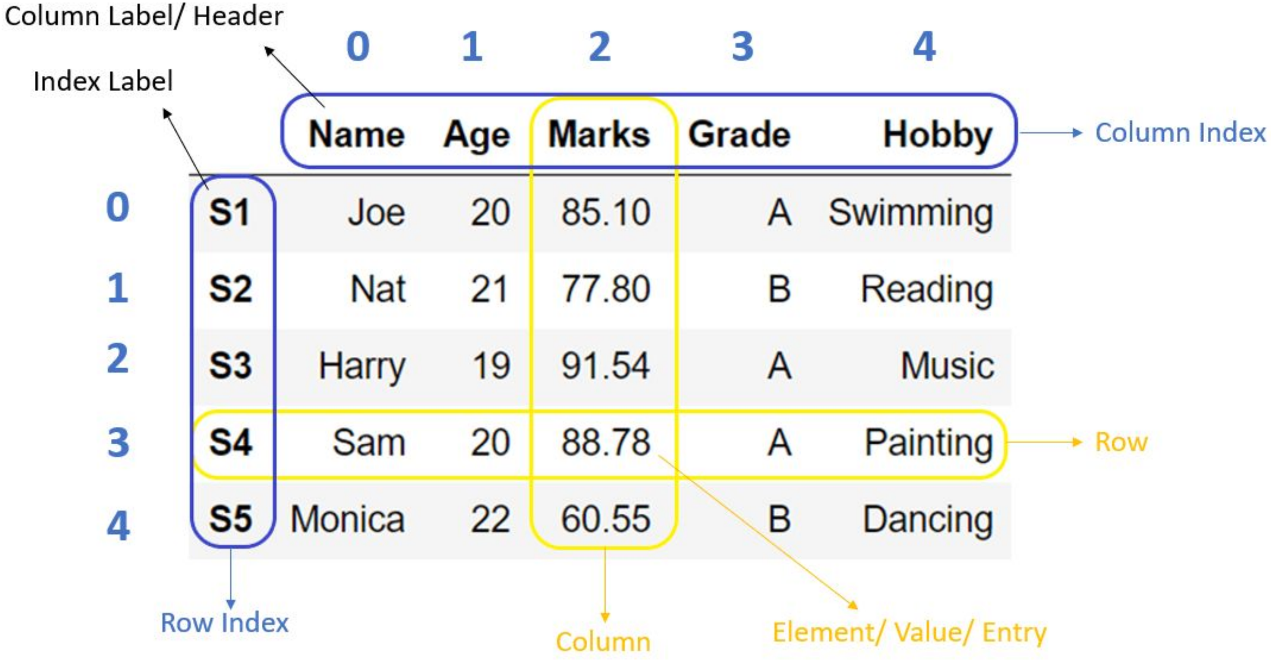
Introducing Python for machine learning projects

- Reading/writing many different data formats
- Selecting subsets of data
- Calculating across rows and down columns
- Finding and filling missing data
- Applying operations to independent groups within the data
- Reshaping data into different forms
- Combing multiple datasets together
- Advanced time-series functionality
- Visualization through Matplotlib and Seaborn

pandas

Introducing Python for machine learning projects

pandas

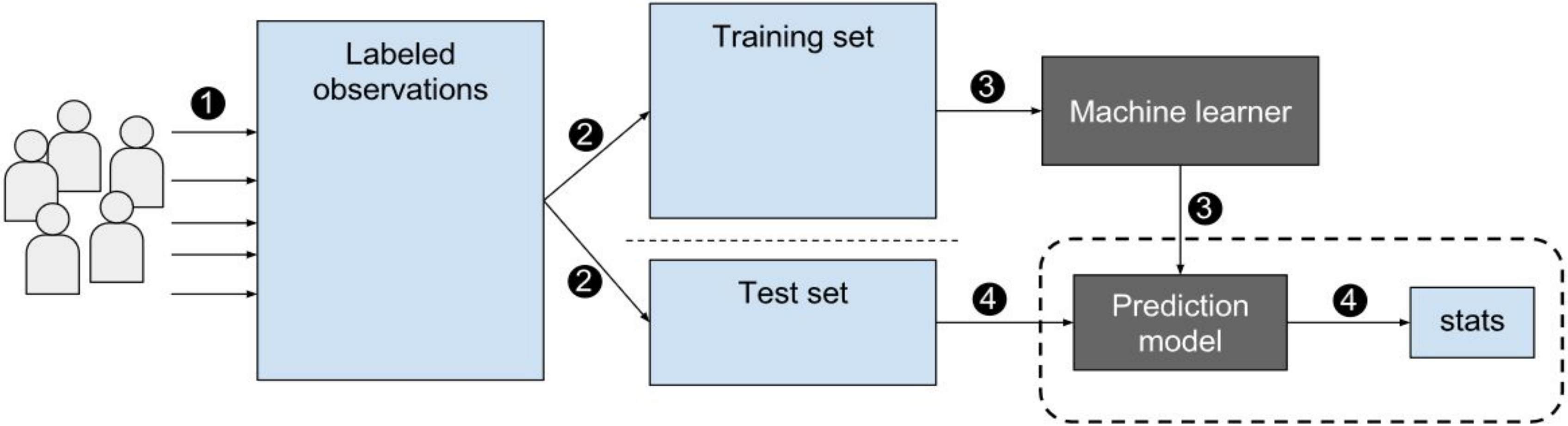


		0	1	2	3	4
		Name	Age	Marks	Grade	Hobby
0	S1	Joe	20	85.10	A	Swimming
1	S2	Nat	21	77.80	B	Reading
2	S3	Harry	19	91.54	A	Music
3	S4	Sam	20	88.78	A	Painting
4	S5	Monica	22	60.55	B	Dancing

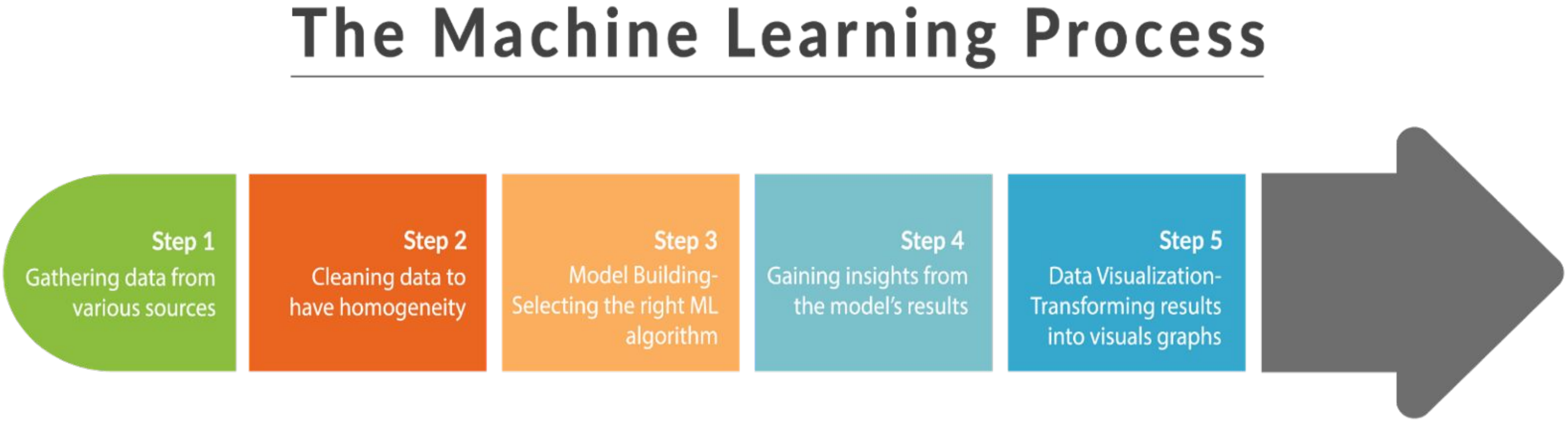
Introducing Python for machine learning projects

```
import pandas as pd
import pandas_profiling

# read the dataset
data = pd.read_csv('your-data')
prof = pandas_profiling.ProfileReport(data)
prof.to_file(output_file='output.html')
```

Stages of a machine learning project



Stages of a machine learning project



The Machine Learning Process

Step 1
Gathering data from various sources

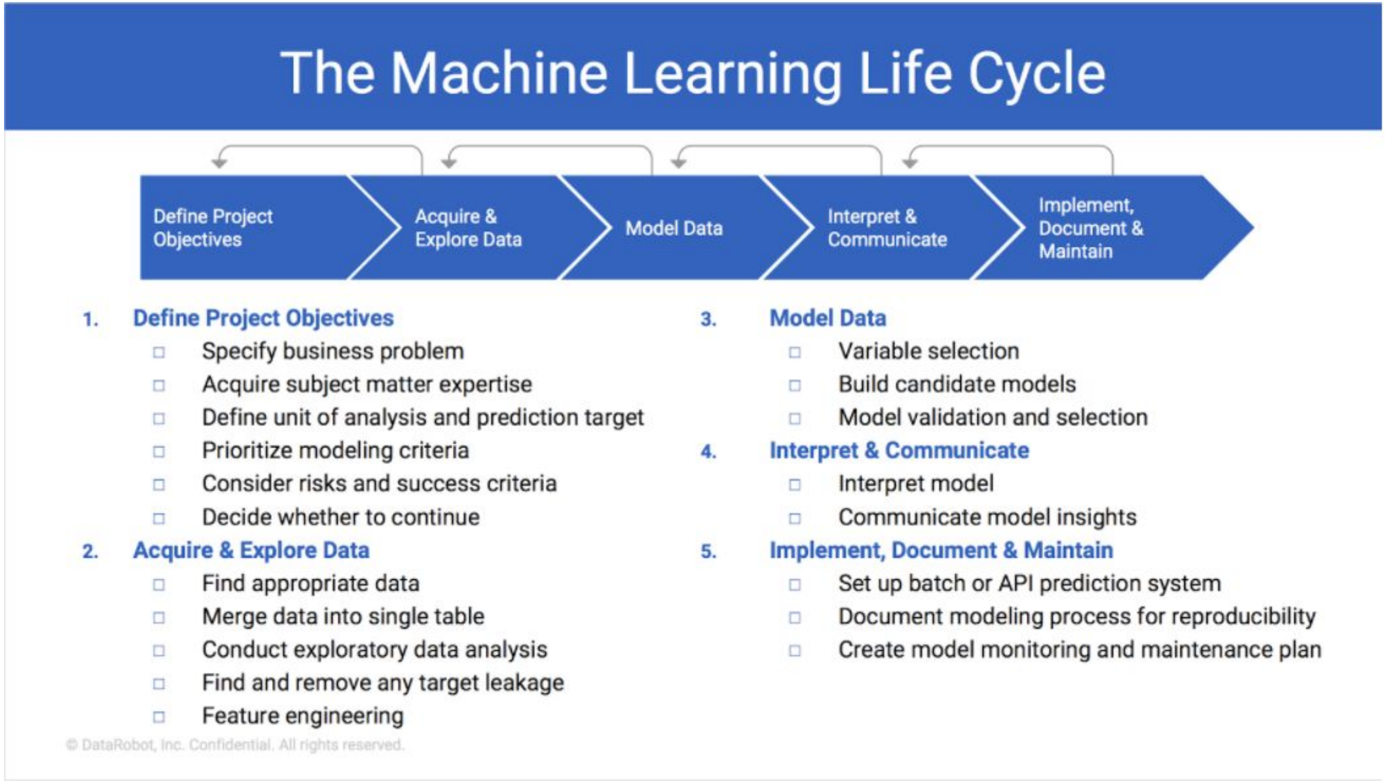
Step 2
Cleaning data to have homogeneity

Step 3
Model Building- Selecting the right ML algorithm

Step 4
Gaining insights from the model's results

Step 5
Data Visualization- Transforming results into visuals graphs

Stages of a machine learning project



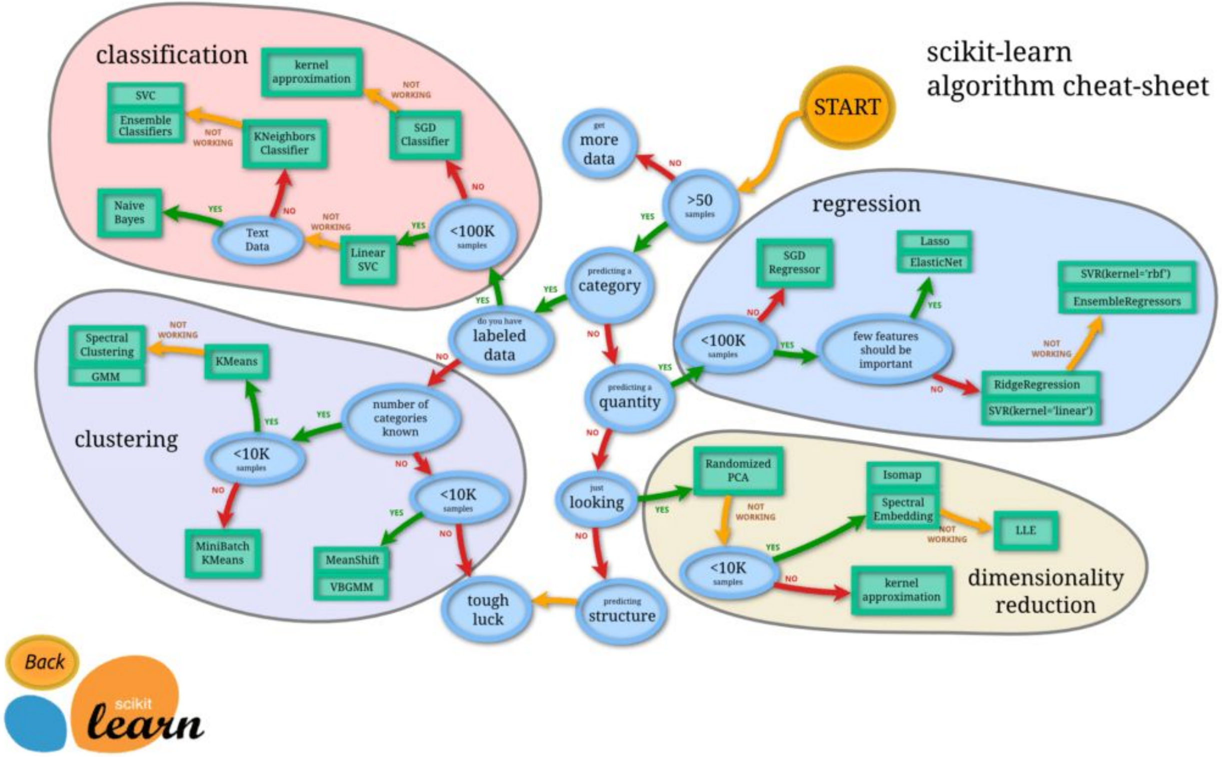
The Machine Learning Life Cycle

Define Project Objectives → Acquire & Explore Data → Model Data → Interpret & Communicate → Implement, Document & Maintain

1. **Define Project Objectives**
   - □ Specify business problem
   - □ Acquire subject matter expertise
   - □ Define unit of analysis and prediction target
   - □ Prioritize modeling criteria
   - □ Consider risks and success criteria
   - □ Decide whether to continue
2. **Acquire & Explore Data**
   - □ Find appropriate data
   - □ Merge data into single table
   - □ Conduct exploratory data analysis
   - □ Find and remove any target leakage
   - □ Feature engineering
3. **Model Data**
   - □ Variable selection
   - □ Build candidate models
   - □ Model validation and selection
4. **Interpret & Communicate**
   - □ Interpret model
   - □ Communicate model insights
5. **Implement, Document & Maintain**
   - □ Set up batch or API prediction system
   - □ Document modeling process for reproducibility
   - □ Create model monitoring and maintenance plan

© DataRobot, Inc. Confidential. All rights reserved.

Python libraries



Python libraries

scikit learn

- Supervised and unsupervised machine learning
- Classification, regression, Support Vector Machine
- Clustering, Kmeans, DBSCAN
- Random Forest

SciPy

NumPy

Python libraries

- Pipelines
- Grid-search
- Validation curves
- One-hot encoding of categorial data
- Dataset generators
- Principal Component Analysis (PCA)

Python libraries

Pipelines

```
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.naive_bayes import MultinomialNB
>>> from sklearn.preprocessing import Binarizer
>>> make_pipeline(Binarizer(), MultinomialNB())
Pipeline(steps=[('binarizer', Binarizer()),
('multinomialnb', MultinomialNB())])
```

http://scikit-learn.org/stable/modules/pipeline.html

Python libraries

Grid-search

estimator.get_params()

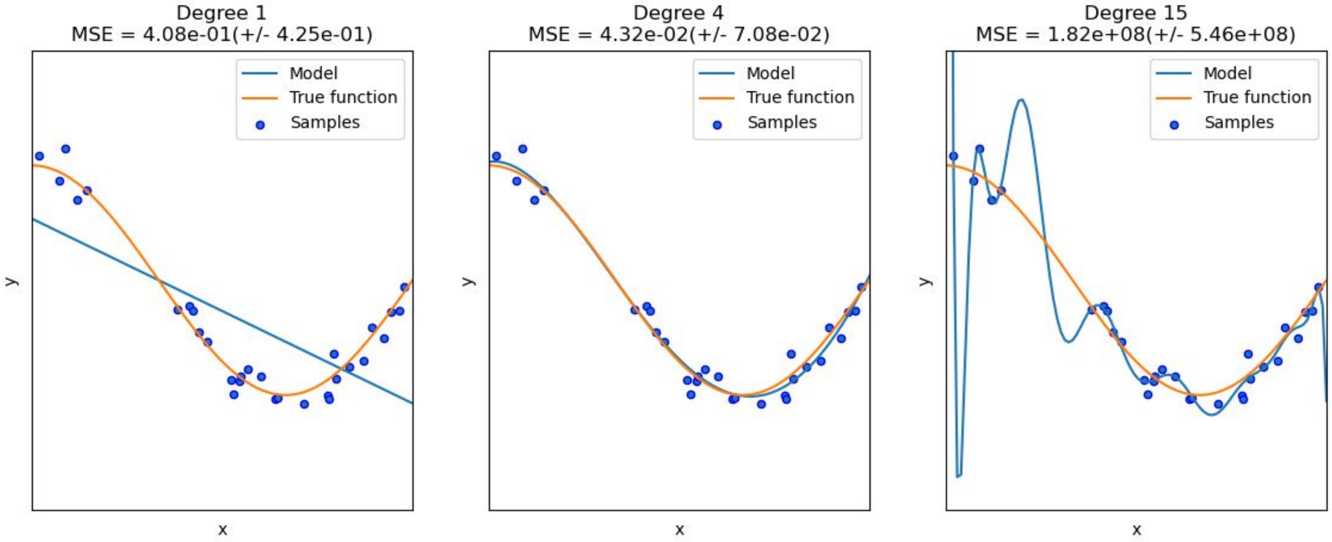
A search consists of:

- **an estimator (regressor or classifier such as sklearn.svm.SVC())**
- **a parameter space**
- **a method for searching or sampling candidates**
- **a cross-validation scheme**
- **a score function**

https://scikit-learn.org/stable/modules/grid_search.html#grid-search

Python libraries

Validation curves



https://scikit-learn.org/stable/modules/learning_curve.html

Python libraries

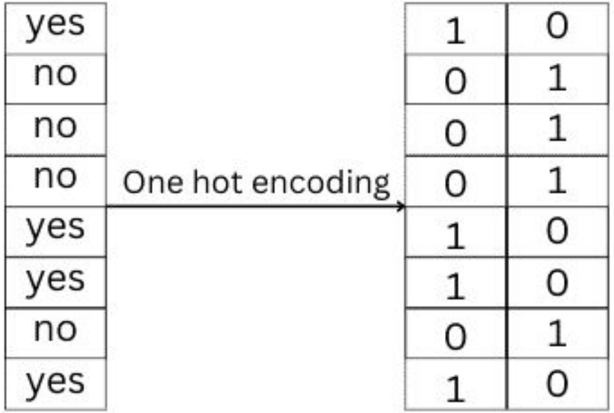
Validation curves

```
>>> train_scores, valid_scores = validation_curve(
...     Ridge(), X, y, param_name="alpha", param_range=np.logspace(-7, 3, 3),
...     cv=5)
>>> train_scores
array([[0.93..., 0.94..., 0.92..., 0.91..., 0.92...],
        [0.93..., 0.94..., 0.92..., 0.91..., 0.92...],
       [0.51..., 0.52..., 0.49..., 0.47..., 0.49...]])
>>> valid_scores
array([[0.90..., 0.84..., 0.94..., 0.96..., 0.93...],
 [0.90..., 0.84..., 0.94..., 0.96..., 0.93...],
[0.46..., 0.25..., 0.50..., 0.49..., 0.52...]])
```

Python libraries

One-hot encoding

```
# importing sklearn one hot encoding
from sklearn.preprocessing import
OneHotEncoder
# initializing one hot encoding
encoding = OneHotEncoder()
# applying one hot encoding in python
transformed_data =
encoding.fit_transform(data[['Status']])
# head
print(transformed_data.toarray())
```

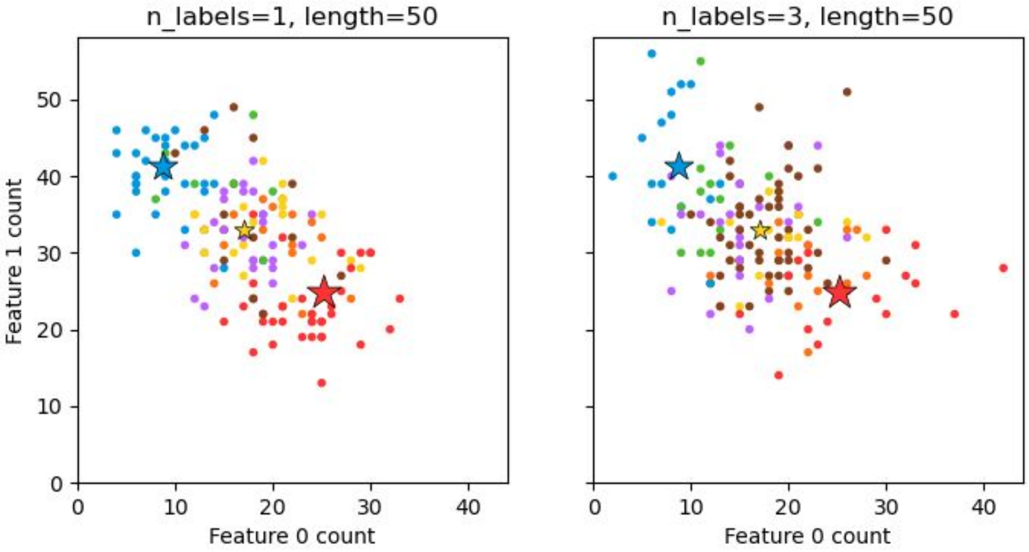


yes	1	0
no	0	1
no	0	1
no	0	1
yes	1	0
yes	1	0
no	0	1
yes	1	0

https://scikit-learn.org/stable/modules/preprocessing.html#encoding-categorical-features

Python libraries

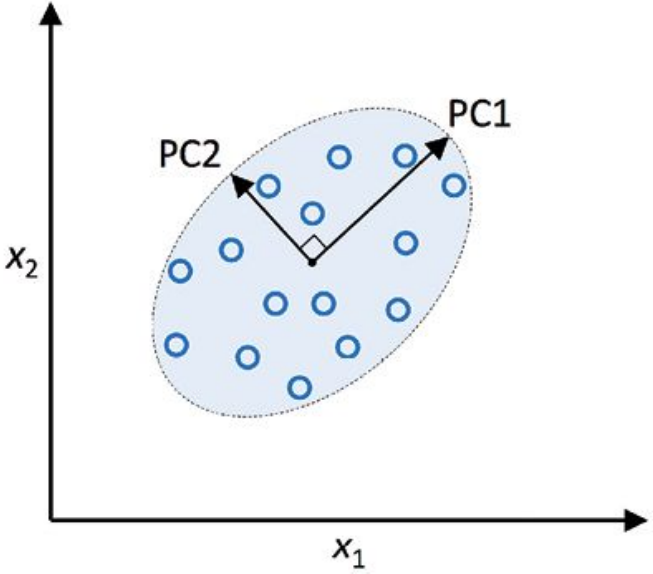
Dataset generators



https://scikit-learn.org/stable/datasets/sample_generators.html

Python libraries

Principal Component Analysis (PCA)



https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

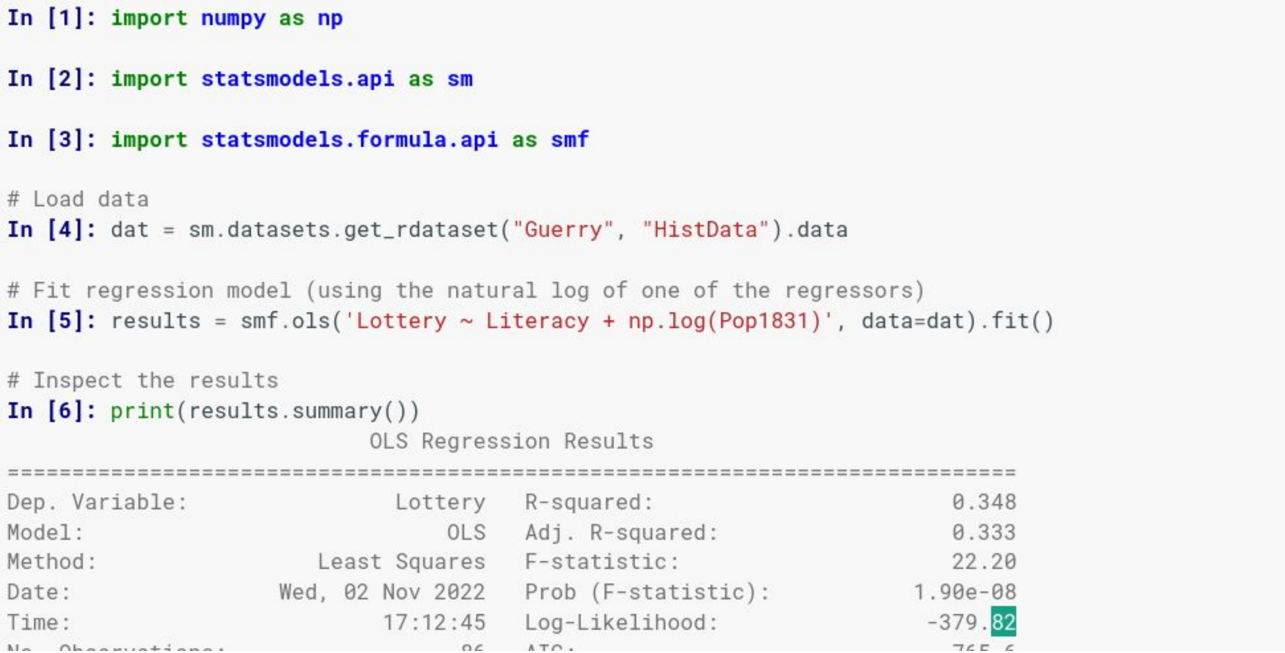
Python libraries

Principal Component Analysis (PCA)

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

Python libraries

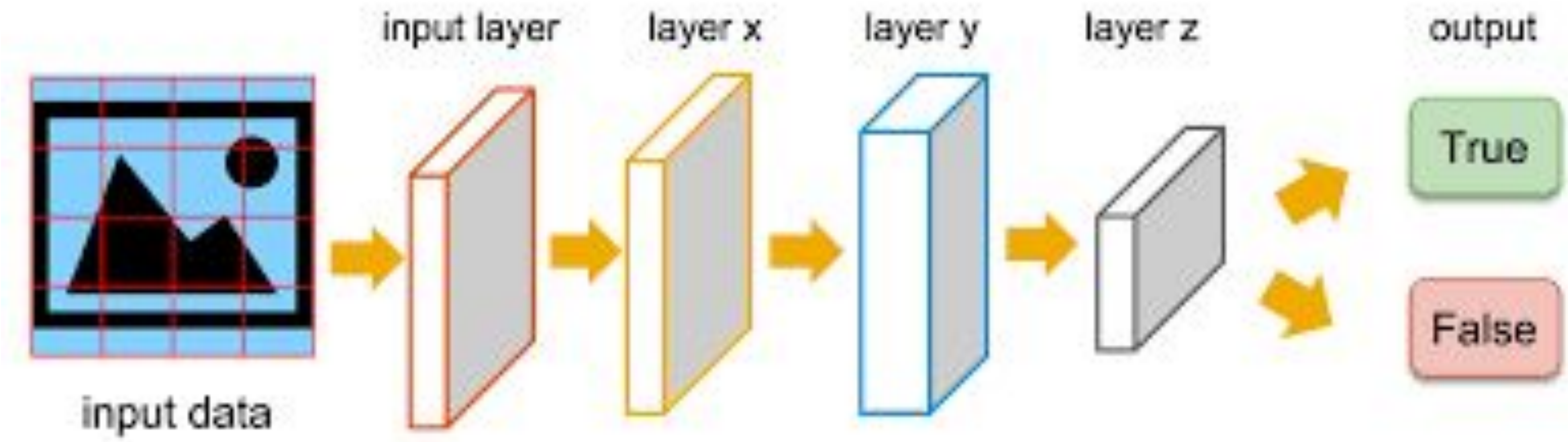
statsmodels



```
In [1]: import numpy as np

In [2]: import statsmodels.api as sm

In [3]: import statsmodels.formula.api as smf

# Load data
In [4]: dat = sm.datasets.get_rdataset("Guerry", "HistData").data

# Fit regression model (using the natural log of one of the regressors)
In [5]: results = smf.ols('Lottery ~ Literacy + np.log(Pop1831)', data=dat).fit()

# Inspect the results
In [6]: print(results.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                Lottery   R-squared:                       0.348
Model:                            OLS   Adj. R-squared:                  0.333
Method:                 Least Squares   F-statistic:                     22.20
Date:                Wed, 02 Nov 2022   Prob (F-statistic):           1.90e-08
Time:                        17:12:45   Log-Likelihood:                -379.82
No. Observations:                  86   AIC:                             765.6
```

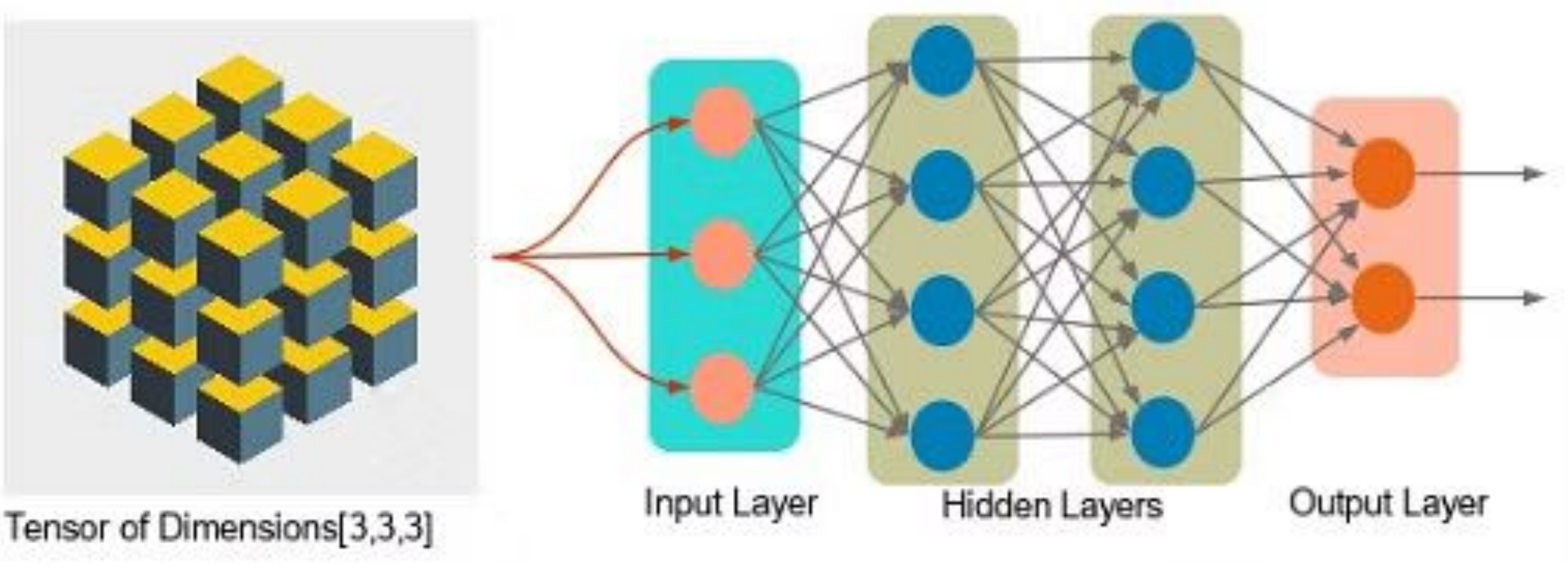
Python tools for deep learning

TensorFlow



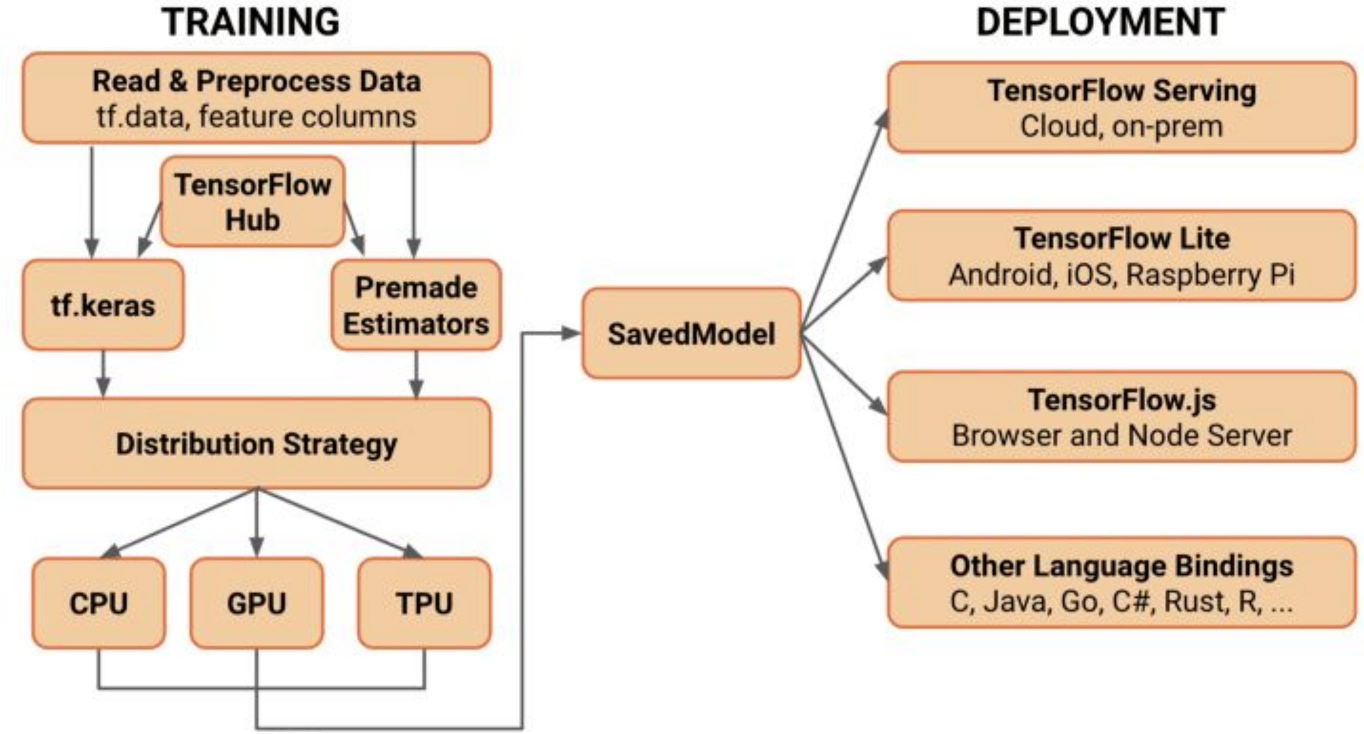
Python tools for deep learning

TensorFlow

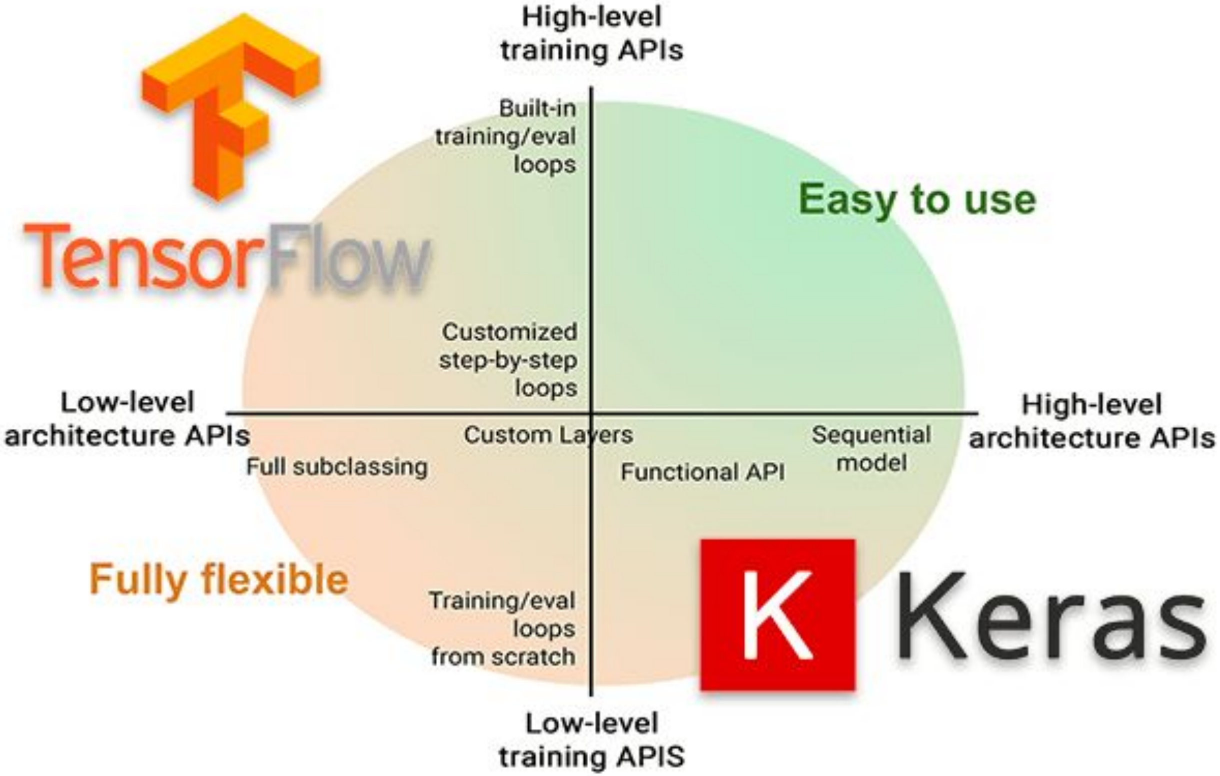


Python tools for deep learning

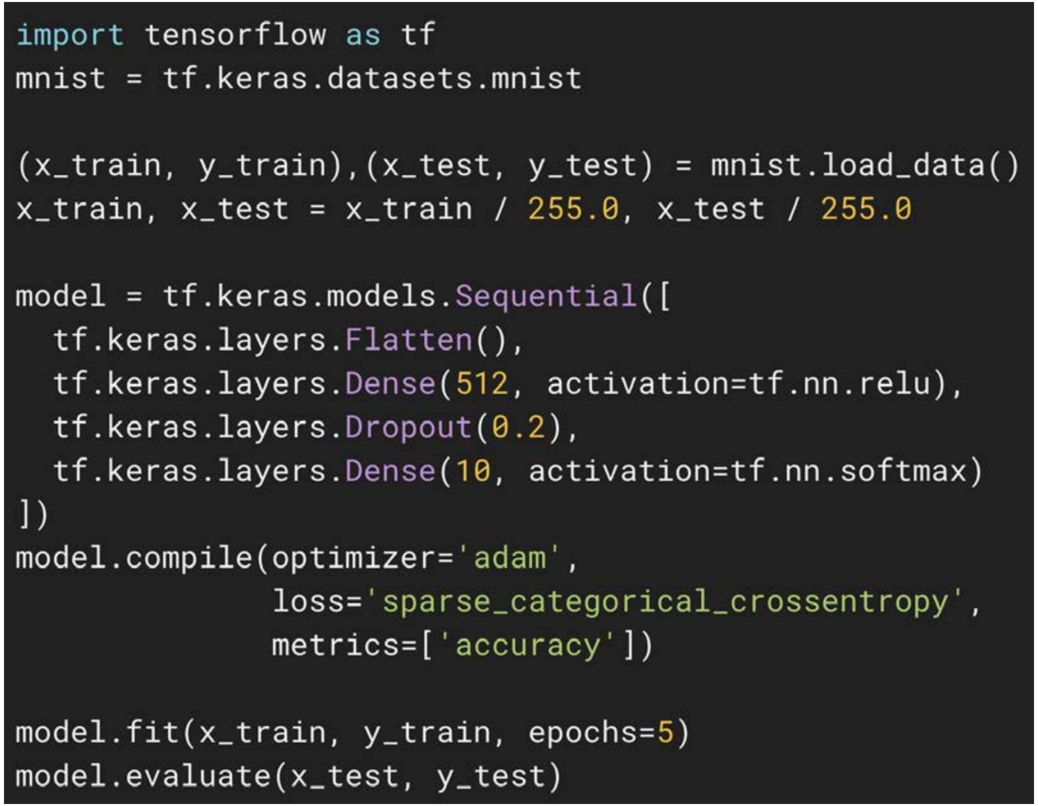
TensorFlow



Python tools for deep learning



Python tools for deep learning



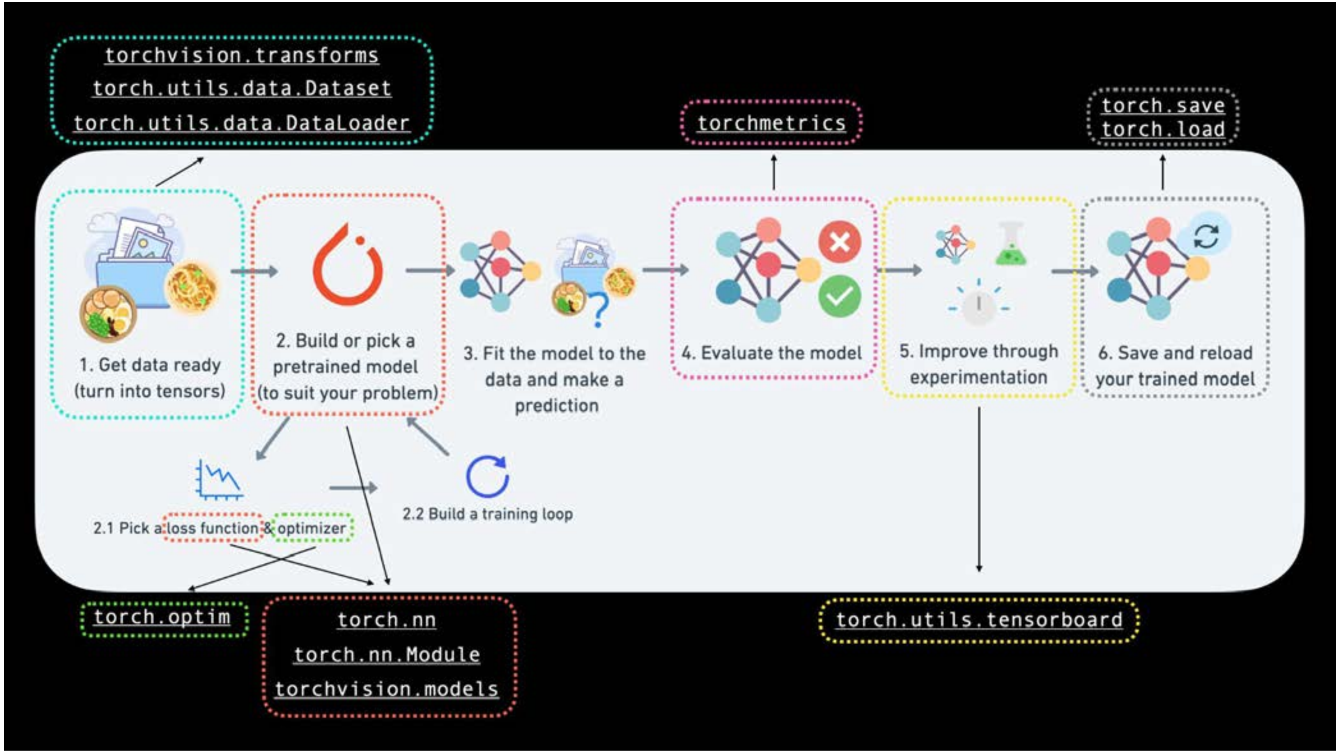
```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Keras

A deep learning library

Python tools for deep learning

PyTorch


torchvision.transforms
torch.utils.data.Dataset
torch.utils.data.DataLoader
torchmetrics
torch.save
torch.load
1. Get data ready (turn into tensors)
2. Build or pick a pretrained model (to suit your problem)
3. Fit the model to the data and make a prediction
4. Evaluate the model
5. Improve through experimentation
6. Save and reload your trained model
2.1 Pick a loss function & optimizer
2.2 Build a training loop
torch.optim
torch.nn
torch.nn.Module
torchvision.models
torch.utils.tensorboard

Python tools for deep learning

	TensorFlow	Keras	Pytorch
API Level	High and Low	High	Low
Architecture	Not easy to use	Simple, concise, readable	Complex, less readable
Speed	Fast, high-performance	Slow, low performance	Fast, high-performance
Trained Models	Yes	Yes	Yes

Python tools for deep learning

theano

- **tight integration with NumPy** – Use numpy.ndarray in Theano-compiled functions.
- **transparent use of a GPU** – Perform data-intensive computations much faster than on a CPU.
- **efficient symbolic differentiation** – Theano does your derivatives for functions with one or many inputs.
- **speed and stability optimizations** – Get the right answer for log(1+x) even when x is really tiny.
- **dynamic C code generation** – Evaluate expressions faster.
- **extensive unit-testing and self-verification** – Detect and diagnose many types of error

Python tools for deep learning

theano

- **Synkhronos** Extension to Theano for multi-GPU data parallelism
- **Theano-MPI** Theano-MPI a distributed framework for training models built in Theano based on data-parallelism.
- **Platoon** Multi-GPU mini-framework for Theano, single node.
- **Elephas** Distributed Deep Learning with Keras & Spark.

Tips and tricks for data science projects with Python

@jmortegac Linkedin

https://www.linkedin.com/in/jmortega1