Statistical Techniques for Monitoring Industrial Processes

Lecture : Python Installation & Development Environment

Module : Python Installation and Basics

Course TOC

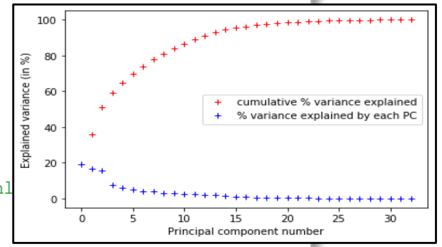
- ❑ Introduction to Statistical Process Monitoring (SPM)
- ❑ Python Installation and basics (optional)
  - ➢ Development environment; Scientific computing packages
- ❑ Univariate SPM & Control Charts
  - ➢ Shewhart Charts
  - ➢ CUSUM Charts
  - ➢ EWMA Charts
- ❑ Multivariate SPM
  - ➢ Fault detection using Principal Component Analysis (PCA)
  - ➢ Fault detection using Partial Least Squares (PLS) regression
  - ➢ Fault diagnosis using PCA/PLS contribution charts
  - ➢ Strategies for handling nonlinear, dynamic, multimode systems
- ❑ Deploying SPM solutions

Implemented using Python

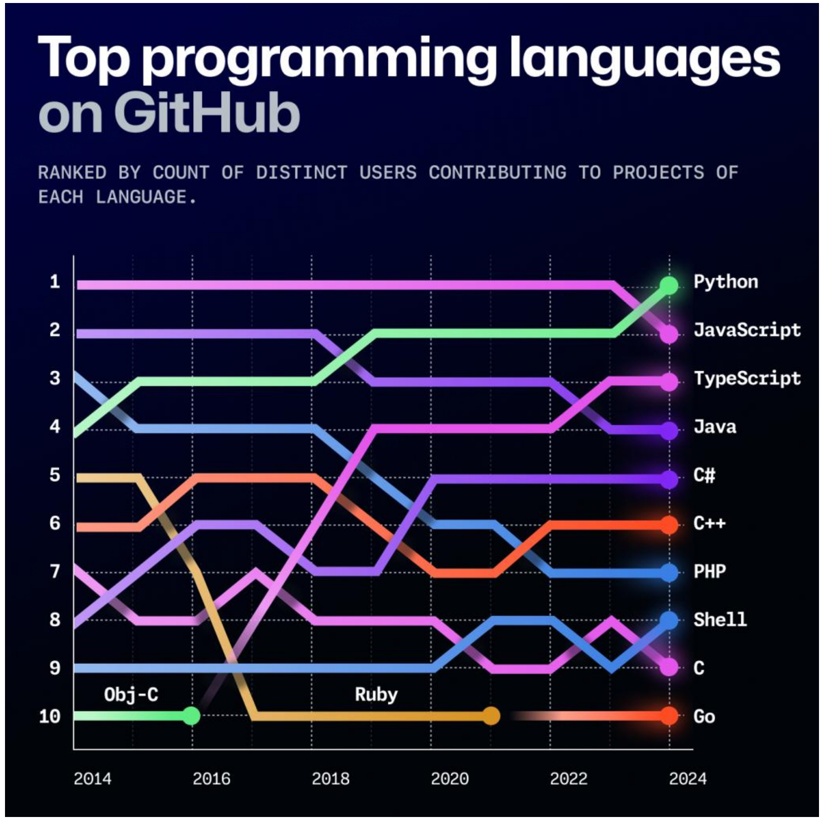
What is Python?

High-level modern programming language

```python
#%% import required packages
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

#%% fetch data
data = pd.read_excel('proc1a.xls', skiprows = 1,usecols = 'C:AI')

#%% separate train data
data_train = data.iloc[0:69,]

#%% scale data
scaler = StandardScaler()
data_train_normal = scaler.fit_transform(data_train)

#%% PCA
pca = PCA()
score_train = pca.fit_transform(data_train_normal)

#%% confirm no correlation
corr_coef = np.corrcoef(score_train,rowvar = False)
print('Correlation matrix: \n', corr_coef[0:3,0:3]) # printing onl

#%% visualize explained variance
import matplotlib.pyplot as plt

explained_variance = 100*pca.explained_variance_ratio_ # in percentage
cum_explained_variance = np.cumsum(explained_variance) # cumulative % variance explained

plt.figure()
plt.plot(cum_explained_variance, 'r+', label = 'cumulative % variance explained')
plt.plot(explained_variance, 'b+' , label = '% variance explained by each PC')
plt.ylabel('Explained variance (in %)'), plt.xlabel('Principal component number'), plt.legend()
```



De-facto language of choice for data science



*https://github.blog/news-insights/octoverse/octoverse-2024/

Why Python is so Popular?

Easy & Simple
- Gentle learning curve
- Easy to read; Low maintenance

Versatile
- ML, DL, scientific computing
- Web development
- GUI programming

Extensive Libraries
- Large standard library set for common tasks
- Ever-growing collection from Python community

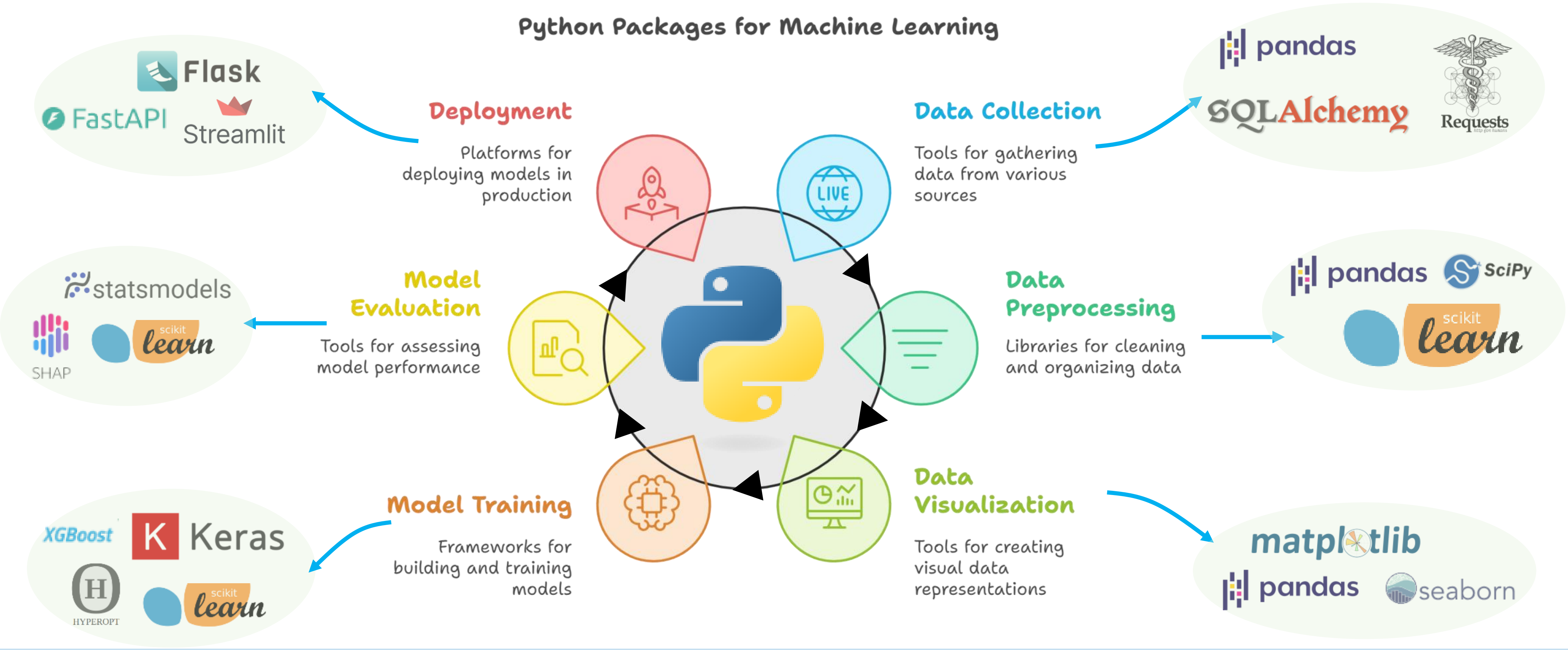
Community Support
- Vast community contributes to open-source knowledgebase
- Easy to find timely support

Documentation
- Well-written documentation
- Plenty of resources (guides, tutorials) for all level (beginner, expert)

Portable / Platform-independent
- Code written on Windows PC can be easily executed on a Linux/Mac/UNIX machine

Why Python is so Popular?

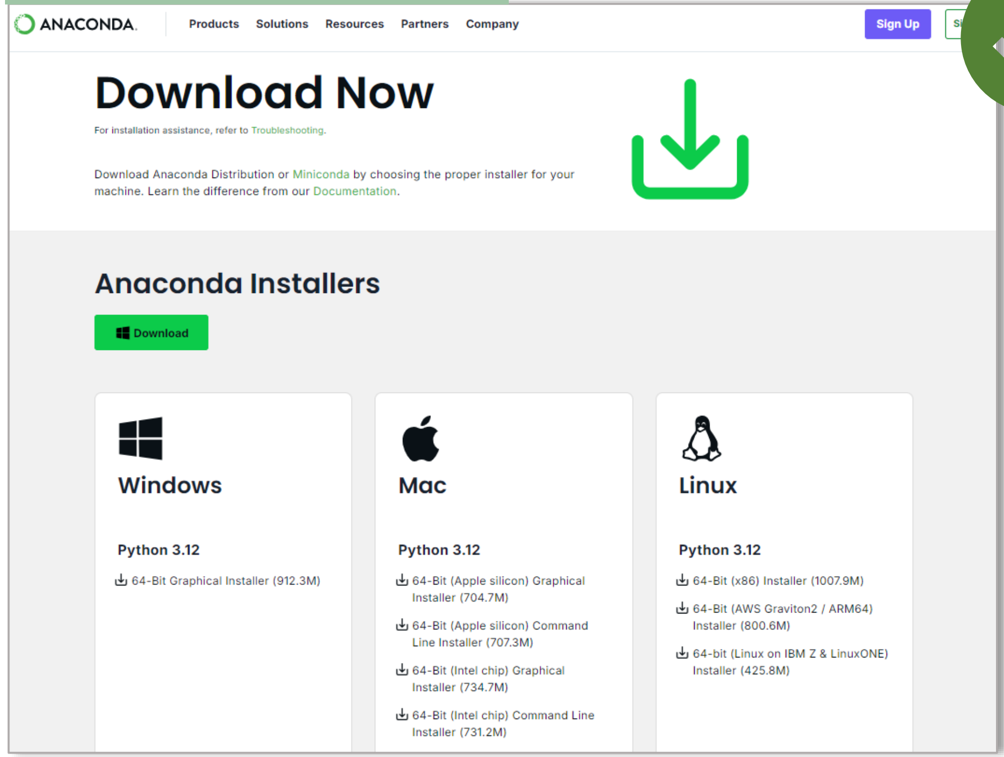
Python provides several tools to conveniently perform all steps of a data science project



Python Installation

Popular and convenient way

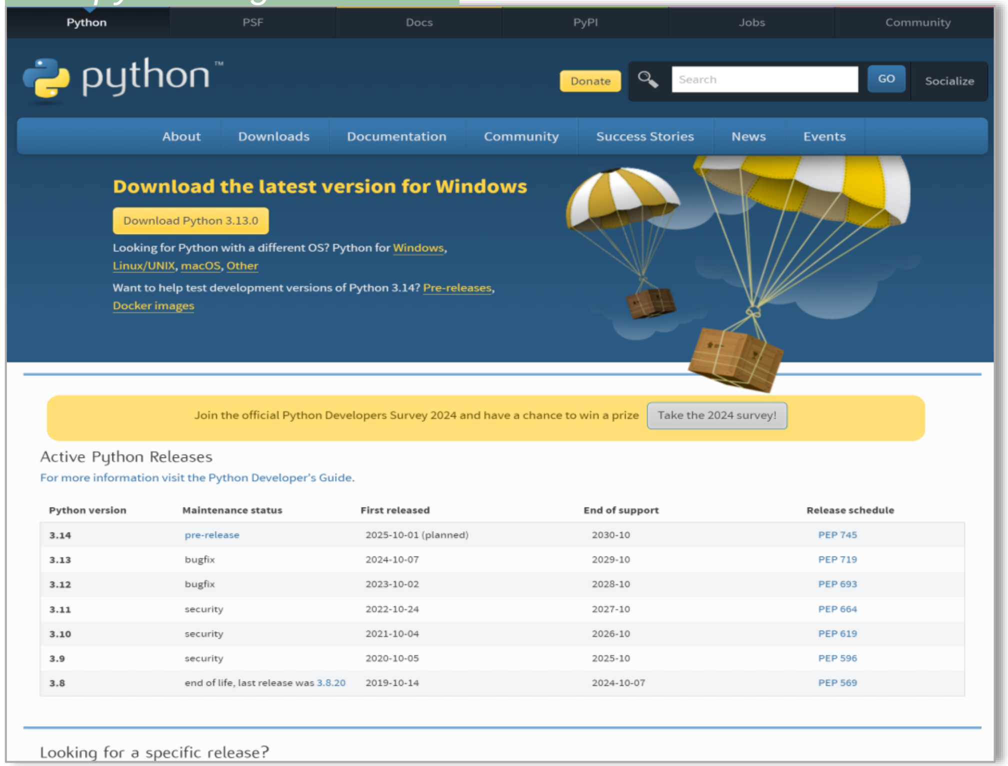
www.anaconda.com/download



➢ When you install Anaconda, several commonly used packages are installed along with Python

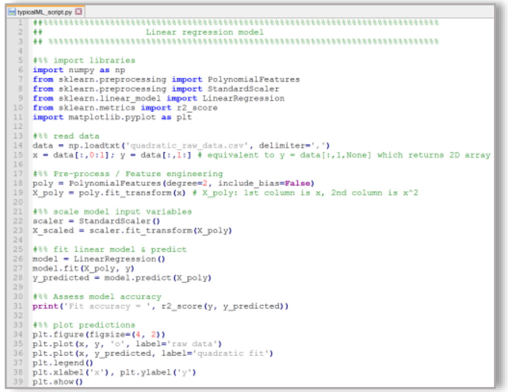
Using base Python installer

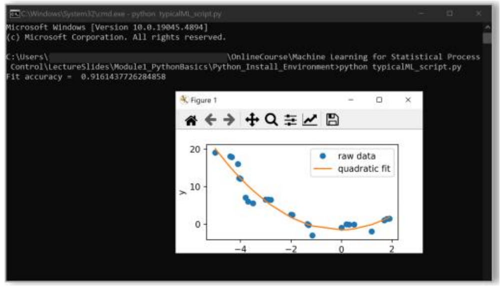
www.python.org/downloads



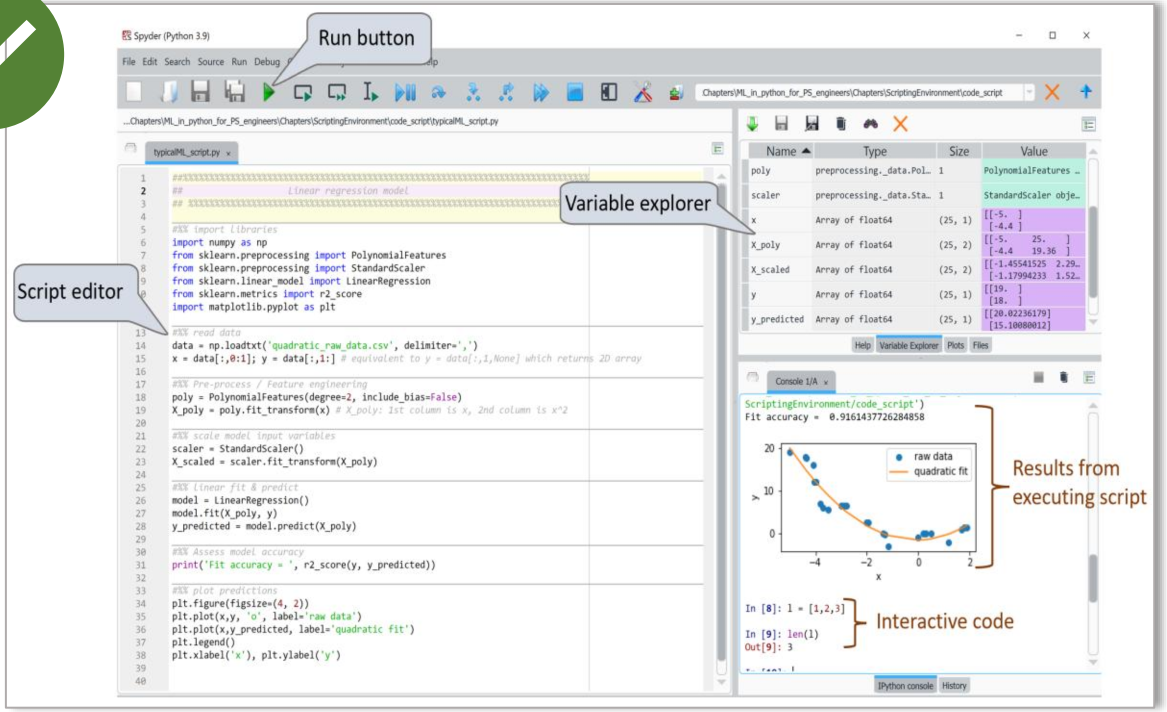
Where to Write Python Code?

Any text editor + command terminal execution





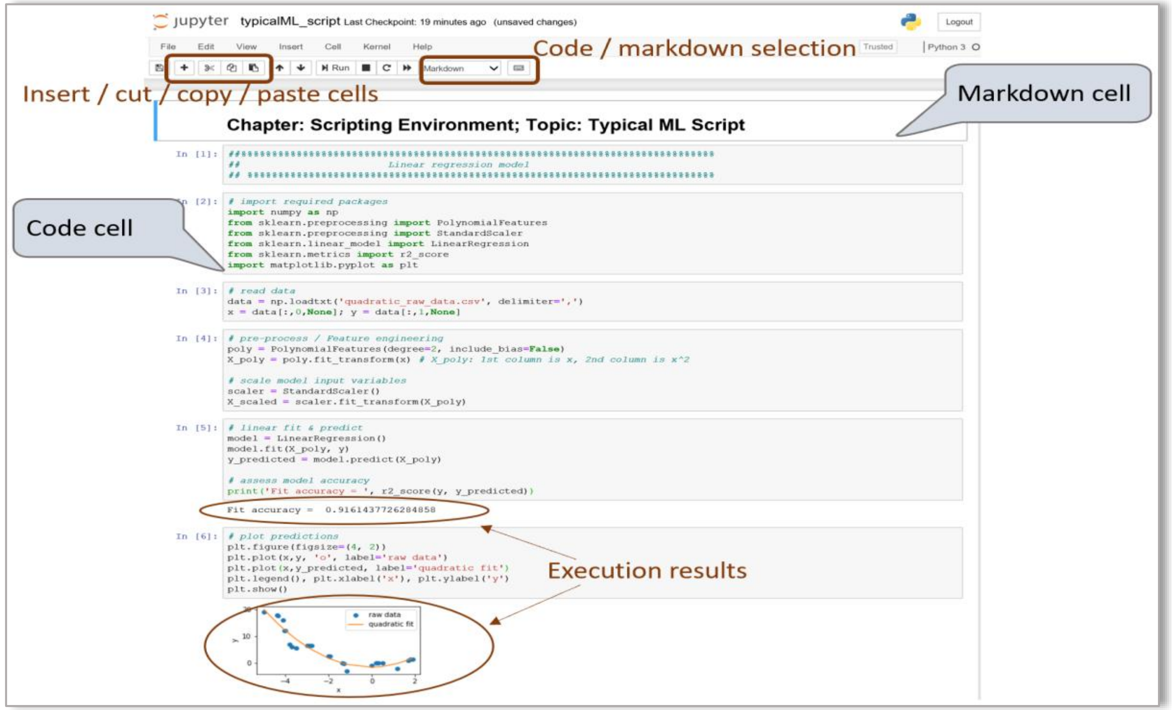
Using IDE (Integrated Development Environment)



- Allows you to write and execute your scripts, and see the results in the same environment
- Popular choices: PyCharm, Spyder, Visual Studio Code

Where to Write Python Code?

Using Jupyter Notebook



- ➢ Allows you to put code, results, plots, formatted text in the same document

Very popular among data scientists for interactive data exploration and modeling experimentations

Statistical Techniques for Monitoring Industrial Processes

Next Lecture : Python Language Basics

Module : Python Installation and Basics