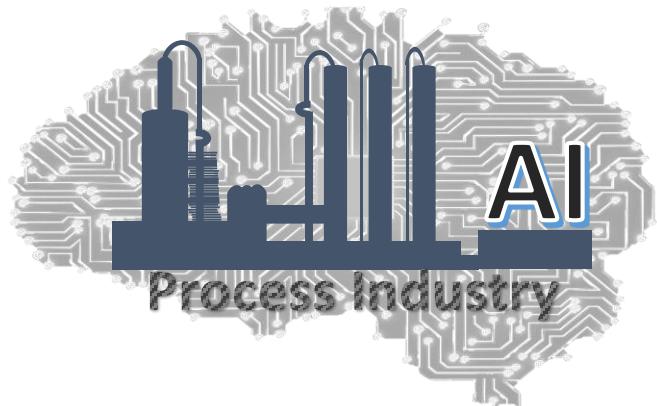


Statistical Techniques for Monitoring Industrial Processes



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

Python

Implemented using



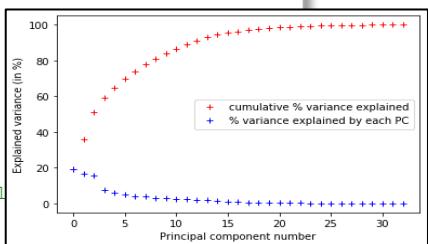
What is Python?

High-level modern
programming language

```

5  #%% import required packages
6  import numpy as np
7  import pandas as pd
8  from sklearn.preprocessing import StandardScaler
9  from sklearn.decomposition import PCA
10
11 #%% fetch data
12 data = pd.read_excel('procla.xls', skiprows = 1,usecols = 'C:AI')
13
14 #%% separate train data
15 data_train = data.iloc[0:69,:]
16
17 #%% scale data
18 scaler = StandardScaler()
19 data_train_normal = scaler.fit_transform(data_train)
20
21 #%% PCA
22 pca = PCA()
23 score_train = pca.fit_transform(data_train_normal)
24
25 #%% confirm no correlation
26 corr_coeff = np.corrcoef(score_train, rowvar = False)
27 print('Correlation matrix: \n', corr_coeff[0:3,0:3]) # printing only first 3 rows
28
29 #%% visualize explained variance
30 import matplotlib.pyplot as plt
31
32 explained_variance = 100*pca.explained_variance_ratio_ # in percentage
33 cum_explained_variance = np.cumsum(explained_variance) # cumulative % variance explained
34
35 plt.figure()
36 plt.plot(cum_explained_variance, 'r+', label = 'cumulative % variance explained')
37 plt.plot(explained_variance, 'b+', label = '% variance explained by each PC')
38 plt.ylabel('Explained variance (in %)'), plt.xlabel('Principal component number'), plt.legend()
39
40

```



De-facto language of choice
for data science

Top programming languages on GitHub

RANKED BY COUNT OF DISTINCT USERS CONTRIBUTING TO PROJECTS OF EACH LANGUAGE.



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)

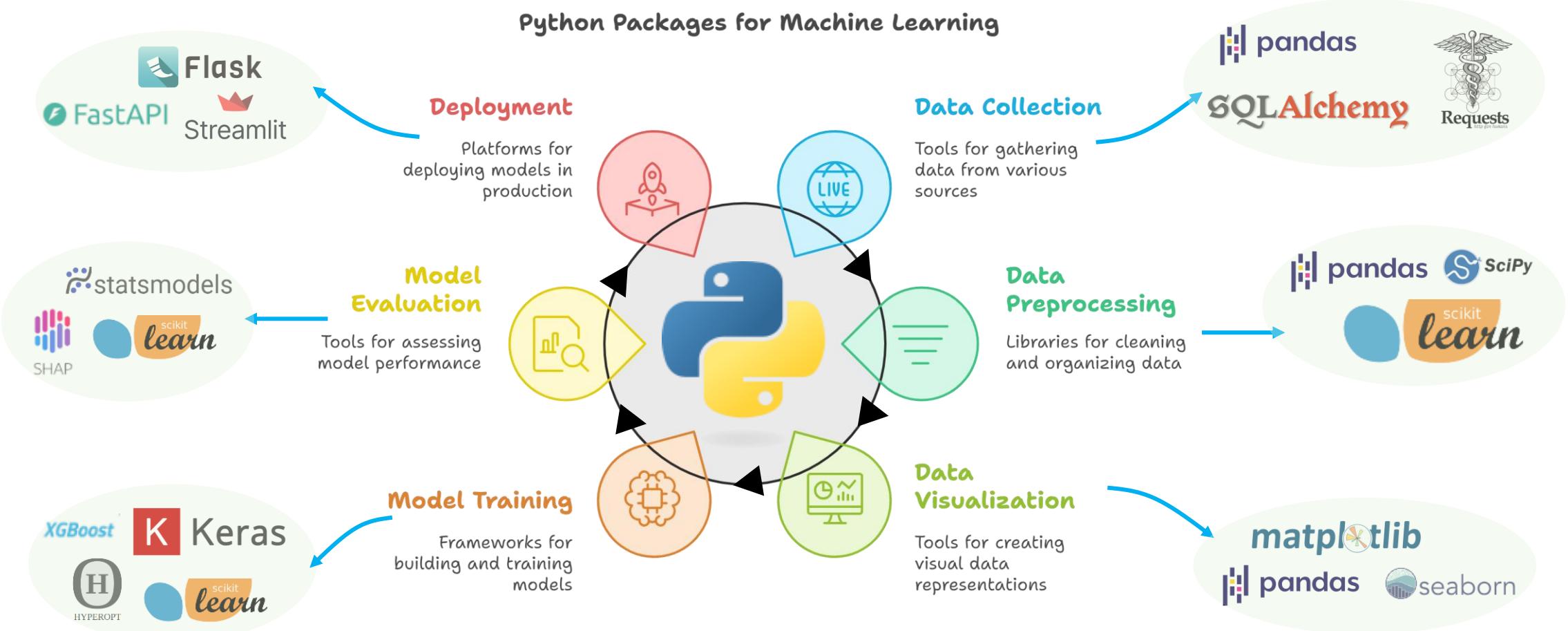


Portable / Platform-independent

- 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

The screenshot shows the Anaconda download page. At the top, there's a navigation bar with links for Products, Solutions, Resources, Partners, and Company. Below that, a large green button says "Download Now". A green checkmark icon is overlaid on the right side of the page. The main content area is titled "Anaconda Installers" and lists installers for Windows, Mac, and Linux. Each section includes a "Download" button and a list of available Python versions (e.g., Python 3.12 for Windows).

Using base Python installer

www.python.org/downloads

The screenshot shows the Python download page. At the top, there's a navigation bar with links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a large illustration of two boxes with yellow and white striped parachutes falling from the sky. A button labeled "Download Python 3.13.0" is prominently displayed. Below the illustration, there's a call-to-action for the Python Developers Survey 2024. The "Active Python Releases" table provides details on various Python versions, their maintenance status, release dates, support end dates, and release schedules.

Python version	Maintenance status	First released	End of support	Release schedule
3.14	pre-release	2025-10-01 (planned)	2030-10	PEP 745
3.13	bugfix	2024-10-07	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	security	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	end of life, last release was 3.8.20	2019-10-14	2024-10-07	PEP 569

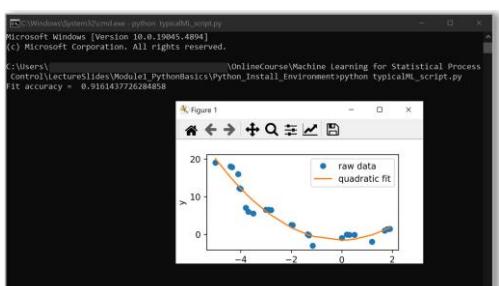
- When you install Anaconda, several commonly used packages are installed along with Python



Where to Write Python Code?

Any text editor + command terminal execution

```
typicalML_script.py
1 #%% Linear regression model
2
3 ## Import libraries
4 import numpy as np
5 from sklearn.preprocessing import PolynomialFeatures
6 from sklearn.preprocessing import StandardScaler
7 from sklearn.linear_model import LinearRegression
8 from sklearn.metrics import r2_score
9 import matplotlib.pyplot as plt
10
11 #%% read data
12 data = np.loadtxt('quadratic_raw_data.csv', delimiter=',')
13 x = data[:,0]; y = data[:,1] # equivalent to y = data[:,1,None] which returns 2D array
14
15 #%% Pre-process / Feature engineering
16 poly = PolynomialFeatures(degree=2, include_bias=False)
17 X_poly = poly.fit_transform(x) # X_poly: 1st column is x, 2nd column is x^2
18
19 #%% scale model input variables
20 scaler = StandardScaler()
21 X_scaled = scaler.fit_transform(X_poly)
22
23 #%% fit linear model & predict
24 model = LinearRegression()
25 model.fit(X_poly, y)
26 y_predicted = model.predict(X_poly)
27
28 #%% Assess model accuracy
29 print("Fit accuracy: ", r2_score(y, y_predicted))
30
31 #%% plot predictions
32 #plt.figure(figsize=(4, 2))
33 plt.plot(x, y, 'o', label='raw data')
34 plt.plot(x, y_predicted, label='quadratic fit')
35 plt.xlabel('x'), plt.ylabel('y')
36 plt.show()
```



Script editor

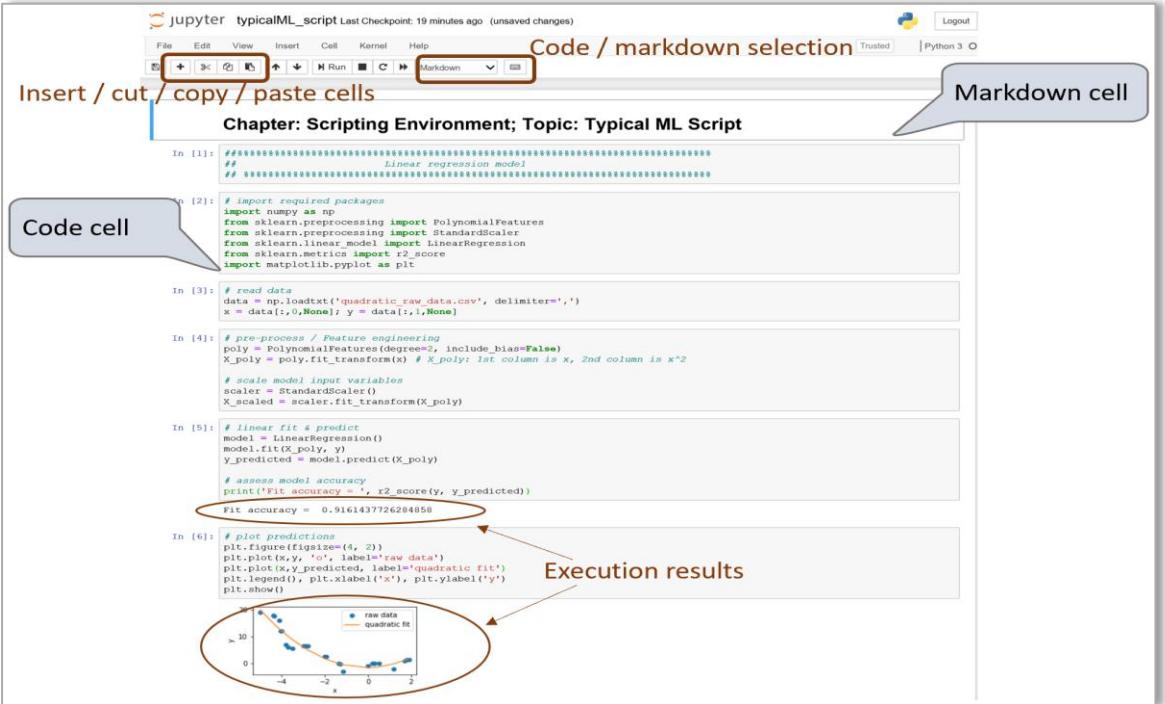
Using IDE (Integrated Development Environment)

The screenshot shows the Spyder IDE interface. The script editor contains the same quadratic regression code as the text editor. The variable explorer shows the data structures: `poly` (PolynomialFeatures), `scaler` (StandardScaler), `X` (array of float64), `X_poly` (array of float64), `X_scaled` (array of float64), `y` (array of float64), and `y_predicted` (array of float64). The results from executing the script are displayed in the 'Results from executing script' section, showing a scatter plot of raw data and a fitted quadratic curve. The interactive code console at the bottom shows the command `In [8]: 1 = [1,2,3]` and its output `Out[8]: 3`.

- 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



The screenshot shows a Jupyter Notebook interface with the following elements:

- Toolbar:** Includes File, Edit, View, Insert, Cell, Kernel, Help, and a dropdown menu for "Code / markdown selection".
- Header:** "jupyter typicalML_script Last Checkpoint: 19 minutes ago (unsaved changes)" and "Python 3".
- Cells:**
 - Code cell:** Contains code for a linear regression model.
 - Markdown cell:** Contains the text "Chapter: Scripting Environment; Topic: Typical ML Script".
 - Execution results:** Shows the output of the code, including data loading, feature engineering, model fitting, and accuracy printing.
 - Plot:** A scatter plot titled "raw data" with a fitted quadratic curve labeled "quadratic fit".

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

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

Statistical Techniques for Monitoring Industrial Processes



Next Lecture : Python Language Basics

Module : Python Installation and Basics

