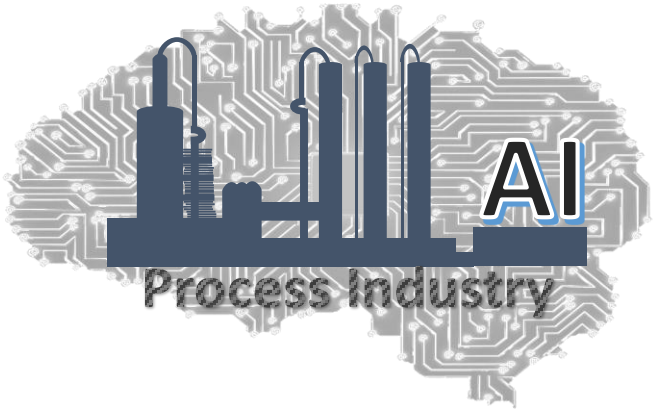
# Statistical Techniques for Monitoring Industrial Processes



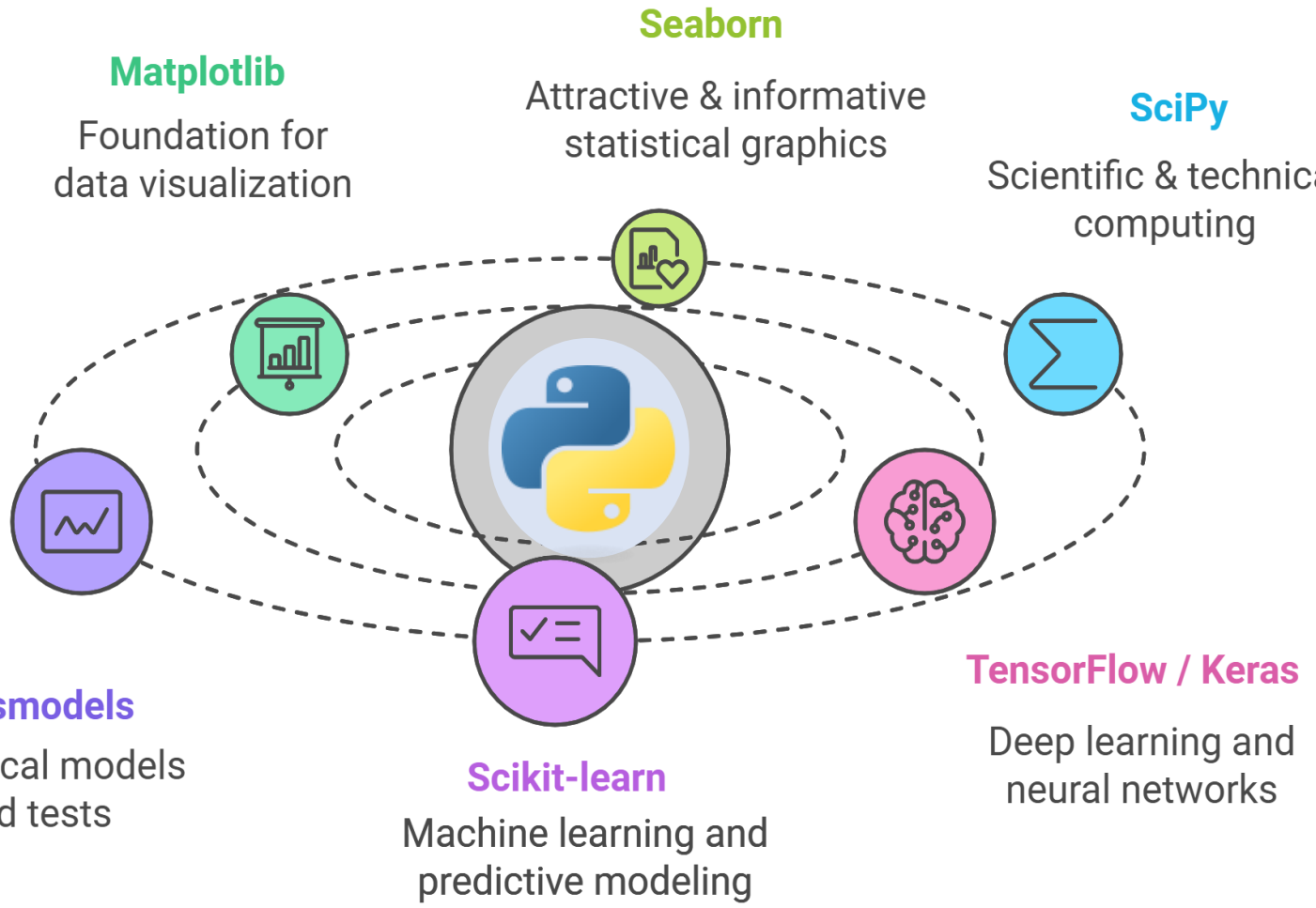

***Lecture :*** Other Useful Packages for Data Analytics

***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 Packages for Scientific Data Analysis



# Matplotlib

The foundational plotting library for Python

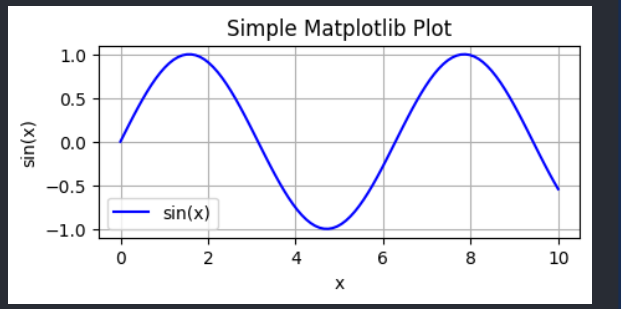
Key Features

- Publication-quality figures
- Highly customizable plots
- Multiple plot types:
  - Line plots, scatter plots
  - Bar charts, histograms
  - 3D plots
  - Contour plots

```python
import matplotlib.pyplot as plt
import numpy as np

# Create sample data
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Create plot
plt.figure(figsize=(5, 2))
plt.plot(x, y, 'b-', label='sin(x)')
plt.title('Simple Matplotlib Plot')
plt.xlabel('x'), plt.ylabel('sin(x)')
plt.legend(), plt.grid(True)
plt.show()
```



# Seaborn

Used for generating elegant and informative statistical graphics

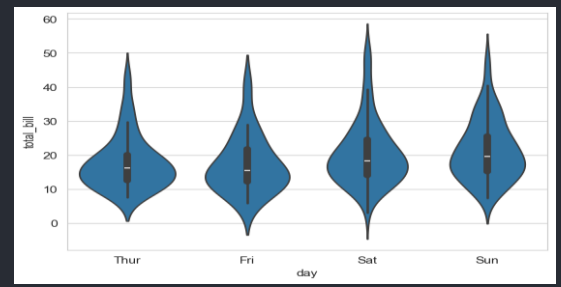
Key Features

- 🎨 Beautiful default styles
- 📈 Complex visualizations with one line:
  - Distribution plots
  - Regression plots
  - Heatmaps
  - Pair plots

```
import seaborn as sns

# Load built-in dataset
tips = sns.load_dataset('tips')

# Violin plot
sns.violinplot(data=tips, x='day',
                          y='total_bill')
```



# Scipy

Scientific computing powerhouse; provides utility functions for optimization, stats, signal processing, etc.

**Key Features**

- Linear algebra operations
- Statistical functions
- Optimization algorithms
- Signal and image processing
- Interpolation
- Numerical integration

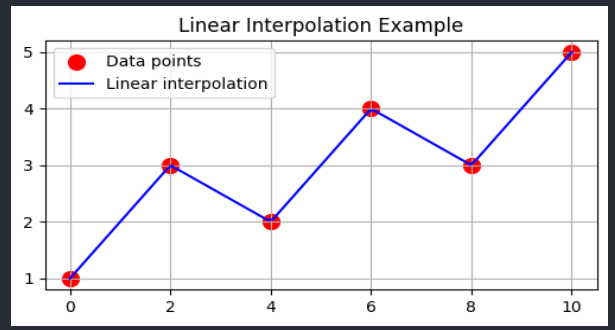
```python
import scipy, numpy as np, matplotlib.pyplot as plt

# Create sample points
x, y = np.array([0, 2, 4, 6, 8, 10]), np.array([1, 3, 2, 4, 3, 5])

# Create linear interpolation function
f = scipy.interpolate.interp1d(x, y)

# Create points for smooth curve
x_new = np.linspace(0, 10, 100)
y_new = f(x_new)

# Plot
plt.figure(figsize=(6, 3))
plt.scatter(x, y, color='red', label='Data points')
plt.plot(x_new, y_new, 'b-', label='Linear interpolation')
plt.title('Linear Interpolation Example'), plt.grid(True)
plt.legend(), plt.show()
```



# Statsmodels

Comprehensive statistical modeling

Key Features

- 📊 Regression Models
- 📈 Time Series Analysis
- 📉 Statistical Tests
- 🔍 Model Diagnostics

```python
import statsmodels.api as sm
import numpy as np

# Create sample data
X = np.random.normal(size=(100,2))
y = np.random.normal(size=(100,))

# Fit linear regression
model = sm.OLS(y, X).fit()
print(model.summary().tables[1])  # Print coefficient table
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.0756      0.095     -0.797      0.427      -0.264       0.113
x2            -0.0842      0.087     -0.962      0.338      -0.258       0.089
==============================================================================
```

# Scikit-learn

Comprehensive machine learning toolkit

Key Features

- Supervised Learning:
  - Classification
  - Regression
- Unsupervised Learning:
  - Clustering
  - Dimensionality reduction
- Model Selection:
  - Cross-validation
  - Hyperparameter tuning
- Data Preprocessing:
  - Scaling

```python
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load the iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Create and train the model
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X, y)

# Make predictions and compute accuracy
predictions = rf.predict(X)
accuracy = accuracy_score(y, predictions) * 100

print(f"Model Accuracy: {accuracy}%")
```

# Tensorflow and Keras

User-friendly framework for neural network modeling and deep learning

Key Features

🧠 Neural Network Types:
- Dense (Fully connected)
- Convolutional (CNN)
- Recurrent (RNN, LSTM)
- Transformer

🛠️ Built-in Tools:
- Data preprocessing
- Model visualization
- Training monitoring
- Model deployment

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.datasets import boston_housing
from sklearn.metrics import r2_score

# Load data
(X_train, y_train), (X_test, y_test) = boston_housing.load_data()

# Create neural network
model = Sequential([
    Dense(64, activation='relu', input_shape=(13,)),
    Dense(32, activation='relu'),
    Dense(1)])

# Compile and train model
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=0)

# Make predictions
predictions = model.predict(X_test, verbose=0)

# Calculate R-squared score
r2 = r2_score(y_test, predictions) * 100
print(f"R-squared Score: {r2}%")
```

# Statistical Techniques for Monitoring Industrial Processes

***Next Lecture :*** Introduction to Univariate SPM & Control Charts

***Module :*** Univariate SPM