

Development Framework for Python Data Scientist, AI, Machine Learning Practitioner

Development framework (workflow + tools + practices) that a **Python Data Scientist, AI, and Machine Learning Practitioner** should follow to build projects efficiently and professionally. Here's a structured breakdown:

1. Problem Definition & Requirement Gathering

- ◆ Understand the business or research problem.
 - ◆ Define **clear objectives** (classification, regression, clustering, recommendation, NLP, CV, etc.).
 - ◆ Identify constraints (data availability, compute resources, latency, interpretability).
-

2. Data Collection & Exploration

- ◆ Sources: APIs, databases, Kaggle, UCI ML, company logs.
 - ◆ Tools:
 - **Pandas / NumPy** – data handling.
 - **SQL / MongoDB** – data extraction.
 - **Web Scraping (BeautifulSoup, Scrapy, Selenium)** – if data is external.
-

3. Data Preprocessing & Feature Engineering

- ◆ Cleaning: Handle missing values, duplicates, anomalies.
 - ◆ Transformation: Scaling, normalization, encoding categorical features.
 - ◆ Feature engineering: Domain-specific feature creation.
 - ◆ Tools:
 - **Scikit-learn** – preprocessing pipeline.
 - **FeatureTools** – automated feature engineering.
 - **NLTK / spaCy** – text preprocessing.
 - **OpenCV / PIL** – image preprocessing.
-

4. Exploratory Data Analysis (EDA)

- ◆ Visualize data distributions, correlations, and trends.
 - ◆ Tools:
 - **Matplotlib / Seaborn / Plotly** – visual analytics.
 - **Pandas Profiling / Sweetviz / ydata-profiling** – automated EDA reports.
 - **Tableau / Power BI** (optional for dashboards).
-

5. Model Building & Training

- ◆ Traditional ML: **Scikit-learn, XGBoost, LightGBM, CatBoost**.
 - ◆ Deep Learning: **TensorFlow, Keras, PyTorch**.
 - ◆ NLP: **Hugging Face Transformers, spaCy**.
 - ◆ Computer Vision: **PyTorch Vision, TensorFlow Object Detection API, OpenCV**.
 - ◆ Reinforcement Learning: **Stable-Baselines3, RLLib**.
-

6. Model Evaluation & Validation

- ◆ Split: Train/Validation/Test.
 - ◆ Cross-validation for robustness.
 - ◆ Metrics:
 - Classification → Accuracy, Precision, Recall, F1, ROC-AUC.
 - Regression → RMSE, MAE, R².
 - Clustering → Silhouette Score, Davies-Bouldin Index.
 - ◆ Tools: **Scikit-learn metrics, Yellowbrick, MLflow tracking**.
-

7. Model Optimization

- ◆ Hyperparameter tuning: **GridSearchCV, RandomizedSearchCV, Optuna, Hyperopt**.
 - ◆ Feature selection: PCA, Lasso, Boruta.
 - ◆ Ensemble methods: Bagging, Boosting, Stacking.
-

8. Deployment & Integration

◆ Packaging models into APIs/services.

- **Flask / FastAPI / Django** – for REST APIs.
 - **Streamlit / Dash / Gradio** – for interactive ML apps.
 - ◆ Model deployment:
 - **Docker / Kubernetes** – containerization.
 - **AWS Sagemaker / GCP AI Platform / Azure ML** – managed ML services.
 - **ONNX / TensorRT** – for model optimization.
-

9. MLOps & Lifecycle Management

◆ Version Control: **Git/GitHub/GitLab**.

◆ Experiment tracking: **MLflow, Weights & Biases (wandb), DVC**.

◆ Continuous Integration/Deployment (CI/CD): **GitHub Actions, Jenkins**.

◆ Monitoring: **Prometheus, Grafana, EvidentlyAI**.

10. Documentation & Communication

◆ Well-structured **Jupyter Notebooks** (analysis, experiments).

◆ Reports & dashboards for stakeholders.

◆ Technical documentation (README, code comments, wiki).

◆ Storytelling with data → crucial for decision-making.

🛠 Tech Stack Summary

- **Core Languages:** Python, SQL
 - **Libraries:** Pandas, NumPy, Scikit-learn, TensorFlow, PyTorch, Matplotlib, Seaborn
 - **DevOps / MLOps:** Git, Docker, MLflow, DVC, Airflow
 - **Deployment:** Flask, FastAPI, Streamlit, AWS/GCP/Azure
-

🚀 Best Practices (Tips & Tricks)

- Always start with **baseline models** before deep learning.
 - Modularize code → reusable functions, pipelines.
 - Use **virtual environments** (`venv`, `conda`, `pipenv`) for dependency isolation.
 - Automate workflows (ETL + training + deployment).
 - Track experiments to avoid “model chaos.”
 - Focus on **explainability (SHAP, LIME)** when models affect critical decisions (finance, healthcare).
-

✓ This framework ensures you cover the **full lifecycle**: from data → models → deployment → monitoring.