Modern Dataiku DSS Project Development Lifecycle

Table of Contents for V3.1_DIKU_DSS

Step 1: Business Requirements Definition (BRD) Use case scoping, stakeholder alignment, KPIs, strategic goals.Business Requirements Document (BRD)

Step 2: Onboarding Process Infrastructure setup, access provisioning, project creation. Onboard Use Cases

Step 3: Development Lifecycle:

Data acquisition, Model building, Testing/Validation.

Step 4: Deployment.

I proposed few more Steps but it will be optional

Step 5: Responsible AI Principles Fairness, accountability, transparency.

Step 6: Data Governance and Security Quality, privacy, compliance, platform hardening.

Step 7: Documentation and Reporting Audit readiness, explainability, automated reporting.

Dataiku Data Science Studio (DSS) provides a collaborative, end-to-end platform for developing, deploying, and managing data science and machine learning (ML) projects. As of 2025, the lifecycle emphasizes automation, governance, and integration with ecosystems like AWS SageMaker, Azure ML, Databricks, and Snowflake, enabling scalable MLOps practices. It supports both visual (low-code) and programmatic workflows, fostering collaboration across data scientists, engineers, and business stakeholders.

The lifecycle is iterative, with built-in tools for version control, experiment tracking, and CI/CD integration. Below is a structured overview of the key stages, drawn from Dataiku's documentation and best practices.

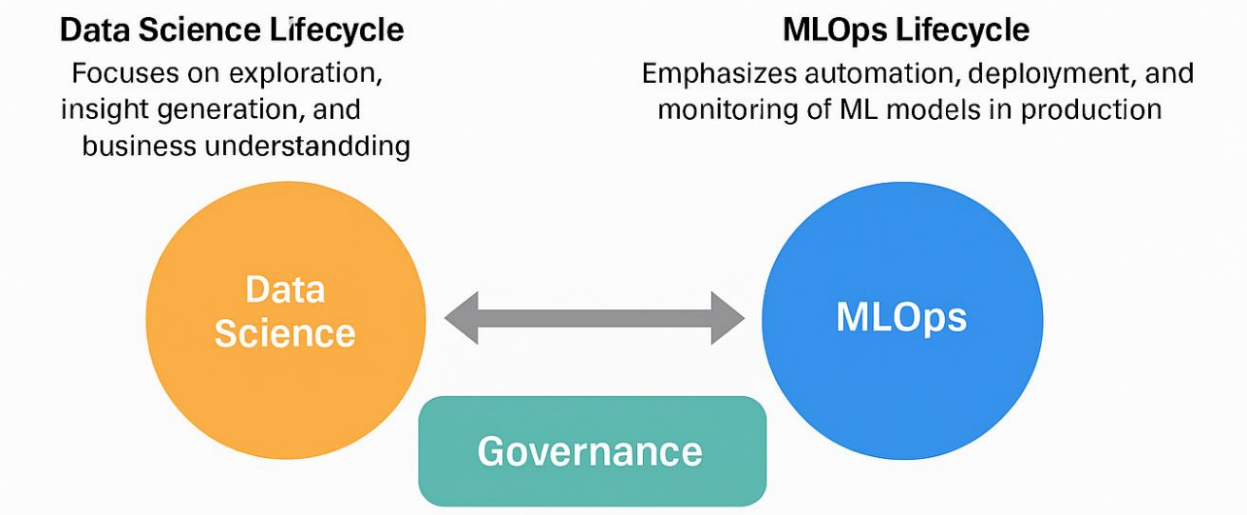
Data Science vs. MLOps in Dataiku — with Governance Perspective

Lifecycle Differences Recap

- **Data Science Lifecycle** focuses on **exploration, insight generation**, and **business understanding**.
- **MLOps Lifecycle** emphasizes **automation, deployment**, and **monitoring** of ML models in production.

Dataiku Combines Both. Dataiku enables seamless collaboration between Data Science and MLOps engineers by offering:

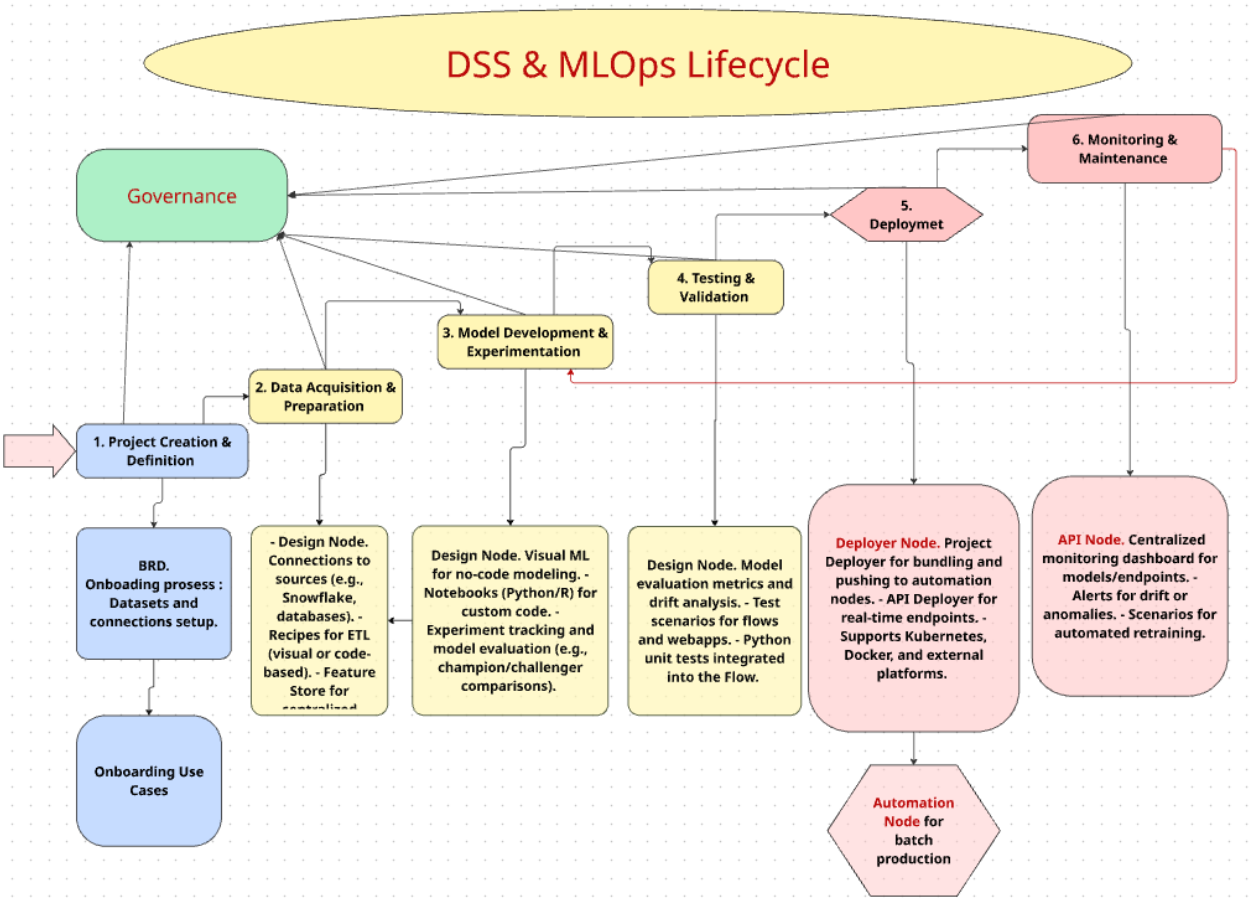
- Visual workflows for data prep and modeling
- AutoML and custom code support
- CI/CD integration for deployment
- Monitoring tools for drift and performance



Aspect	Dataiku Implementation
Strategy & Planning	Defined governance goals in project documentation and scenarios.
Data Management	Uses metadata, lineage tools, and secure access controls to ensure data integrity and security.
Model Development	Tracks experiments with MLflow and enforces ethical reviews and documentation during the development phase.
Deployment	Automates the process with CI/CD capabilities and provides tools to document deployment steps.
Monitoring & Improvement	Sets up drift detection, schedules audits, and supports continuous retraining of models

Visual Representation (Text-Based Diagram) ***DS & MLOps Lifecycle***

Diagram of the lifecycle, showing sequential flow with feedback loops for iteration (e.g., from Monitoring back to Data Preparation). This aligns with Dataiku's emphasis on continuous, reproducible workflows.



Development Lifecycle in Dataiku

Dataiku structures the AI development lifecycle around iterative project workflows in the **Flow** (a visual data pipeline editor). The lifecycle aligns with CRISP-DM or similar frameworks but is tailored for collaborative, code-optional environments:

1. **Project Definition**: Define objectives and resources.
2. **Data Acquisition**: Ingest and connect data sources (focus of this doc).
3. **Data Exploration**: Visualize and profile data.
4. **Data Preparation**: Clean, transform, and enrich data.

5. **Model Building & Experimentation**: Train and iterate models in the Visual ML or Lab.
6. **Evaluation & Iteration**: Validate performance and refine.
7. **Deployment**: Bundle into production-ready artifacts.
8. **MLOps**: Monitor, retrain, and govern post-deployment.

Key Stages of the Lifecycle

Stage	*Description*	*Key Dataiku Features/Tools*	*Best Practices (2025 Updates)*
1. Project Creation & Definition	*Define project scope, use cases, and requirements. This includes creating a new project in DSS as a container for datasets, flows, models, and notebooks. Incorporate business requirements (e.g., BRD) to align with KPIs.*	*- Project dashboard for overview. - Datasets and connections setup. - Visual Flow for initial data discovery.*	*Use project standards for governance (e.g., naming conventions, security). Integrate AI governance early for bias checks and compliance (e.g., GDPR).*
2. Data Acquisition & Preparation	*Ingest, clean, and engineer data. Curate features for reuse across projects.*	*- Design Node. Connections to sources (e.g., Snowflake, databases). - Recipes for ETL (visual or code-based). - Feature Store for centralized feature management.*	*Automate data quality checks with scenarios. Leverage 2025 enhancements in real-time data streaming for faster ingestion.*
3. Model Development & Experimentation	*Build, train, and iterate on ML models. Explore prototypes in a lab-like environment.*	*- Design Node. Visual ML for no-code modeling. - Notebooks (Python/R) for custom code. - Experiment tracking and model evaluation (e.g., champion/challenger comparisons).*	*Track experiments with metadata for reproducibility. Import external models for hybrid workflows. Focus on explainability with auto-generated documentation.*
4. Testing & Validation	*Evaluate model performance, run unit tests, and ensure quality. Stress-test for edge cases.*	*- Design Node. Model evaluation metrics and drift analysis. - Test scenarios for flows and webapps. - Python unit tests integrated into the Flow.*	*Implement automated checks for data/model drift. Use 2025's expanded integration with CI/CD tools (e.g., GitLab CI) for continuous validation.*
5. Deployment	*Package and deploy models/projects to production environments (batch or real-time).*	*- Deployer Node. Project Deployer for bundling and pushing to automation nodes. - API Deployer for real-time endpoints. - Supports Kubernetes, Docker, and external platforms.*	*Adopt lifecycle stages (Dev → Test → Prod) for staged rollouts. Automate with Python API for 10x faster operationalization, as seen in enterprise cases reducing deployment time by 90%.*
6. Monitoring & Maintenance	*Track performance post-deployment, detect issues, and retrain as needed.*	*- Govern Node. Centralized monitoring dashboard for models/endpoints. - Alerts for drift or anomalies. - Scenarios for automated retraining.*	*Enable unified visibility across hybrid deployments. Use version control for quick rollbacks. 2025 updates include enhanced scalability for 100s of models with minimal code.*

Stage 2. Data Acquisition & Preparation in Dataiku DSS

1. Ingest Data

Use **Design Node** to connect to:
Snowflake, databases, cloud storage, APIs
Governance Tip: Tag and document each data source with metadata (owner, refresh rate, sensitivity level).

2. Clean & Validate

Apply visual recipes (Prepare, Filter, Join) or code recipes (Python, SQL, Spark) for:

- Handling missing values, duplicates, and outliers
- Standardizing formats and encoding variables
- Creating derived columns and aggregations

Use Scenarios to automate: Schema checks, null thresholds, and alerts.
Governance Tip: Log all transformations and maintain version control using Flow documentation and Git integration.

3. Feature Engineering

- Feature Selection: Analyze correlations in the Statistics tab to remove redundant features and prevent overfitting.
- Avoid Leakage: Ensure features are prediction-time compatible, using only historical data for time-series (e.g., prior sales). Will be additional link to explore this topic.
- Transformations: Create meaningful features (e.g., age from birth dates) via Prepare recipes or visual ML Script tab.

Advanced Feature Engineering:

- Use time-series recipes, embeddings, and rolling aggregates
- Leverage custom plugins for domain-specific transformations

Governance Tip: Implement **access controls** and **approval workflows** for publishing features to the store.

Stage 3: Model Development in Dataiku DSS

1. **Model Selection & Configuration**

- Use Visual ML to explore algorithms (e.g., Random Forest, XGBoost) based on data characteristics (e.g., sample size, task type).
- Configure settings: Split data (e.g., 80/20 train/test), set hyperparameters (e.g., max depth = 10), and enable cross-validation.
- For custom models, use Jupyter Notebooks to custom code in Python or R.

Governance Tip: Use Dataiku Govern to define model qualification criteria and log selection rationale for auditability.

2. **Training & Experimentation**

- Train models in Visual ML by dragging prepared datasets (e.g., prepared_fraud) into the pipeline.
- Experiment with multiple algorithms and hyperparameter combinations, tracked via Experiment Tracking.
- Example (Python recipe):

```
from sklearn.ensemble import RandomForestClassifier
import dataiku

prepared_dataset = dataiku.Dataset("prepared_fraud")
df = prepared_dataset.get_dataframe()
X = df[['transaction_freq', 'amount']]
y = df['is_fraud']
model = RandomForestClassifier(n_estimators=100)
model.fit(X, y)
```

a. Analyze feature importance to refine Stage 2 features.

Governance Tip: Apply bias detection tools in Govern to ensure fairness during training and require stakeholder sign-offs.

3. Iteration & Optimization

- Iterate based on performance metrics (e.g., adjust features or models if accuracy is low).
- Use Spark Integration for scalable training on large datasets.
- Document experiments and save models in the Flow for reproducibility.

Governance Tip: Enforce version control on models and track changes with Audit Trails for transparency.

Stage 4: Testing & Validation in Dataiku DSS

1. **Data Splitting & Test Preparation**
   - Use a Split recipe to create a test set (e.g., 20%) from prepared_fraud.
   - Ensure test data is representative and free of training data leakage.
   - Governance Tip: Tag test datasets with metadata (e.g., creation date, purpose) and restrict access via RBAC.
2. **Model Evaluation**

- Evaluate in Visual ML with metrics (e.g., AUC for classification, RMSE for regression) on the test set.
- For custom evaluation, use a Python recipe:

```
from sklearn.metrics import roc_auc_score
import dataiku

test_dataset = dataiku.Dataset("test_fraud")
df_test = test_dataset.get_dataframe()
X_test = df_test[['transaction_freq', 'amount']]
y_test = df_test['is_fraud']
predictions = model.predict_proba(X_test)[:, 1]
auc = roc_auc_score(y_test, predictions)
print(f"AUC: {auc}")
```

- Perform k-fold cross-validation (e.g., k=5) to assess stability.

Governance Tip: Use Govern to automate drift analysis and flag performance degradation.

7. **Validation Against Business Goals**
   - Compare results against KPIs (e.g., fraud detection rate > 90%) using Charts or Score recipes.
   - Validate on new data samples to ensure generalizability.
   - Document findings in the project wiki.

   Governance Tip: Require Govern approval workflows and prevent deployment without validation sign-offs.
8. **Iteration & Finalization**
   - If validation fails, revisit Stage 2 (feature engineering) or Stage 3 (model tuning).
   - Finalize the model once KPIs are met, saving it for Stage 5 (Deployment).

   Governance Tip: Use Advanced Govern for scheduled sign-off resets and compliance checks before proceeding.

Stage 4.5: Model Review & Governance Gate (Pre-Deployment Approval)

Mandatory before Stage 5

1. **Model Documentation & Artifacts**
   a. Export **Model Card** from Visual ML (includes performance, features, bias checks, data lineage).
   b. Attach **Experiment Summary Report** (auto-generated from Experiment Tracking).

    c. Save **Final Saved Model Version** with semantic tag: v1.0-prod-candidate.
2. **Architecture Review Board (ARB) Sign-Off**
    a. **Submit to ARB via Confluence + Dataiku Govern Blueprint**:
        i. Model purpose, business KPI alignment, scalability plan, rollback strategy.
        ii. Confirm **no PII leakage**, **HIPAA-compliant features only**.
    b. **ARB Checklist**:
        i. Model meets AUC > 0.90 (or business threshold)
        ii. No high-risk drift flags in evaluation store
        iii. Code environment uses approved Delta Dental base image
        iv. Rollback bundle prepared and tested
3. **Dataiku Govern Approval Workflow**
    a. In **Govern Node**, create a **Model Qualification Process**:
        i. Assign reviewers: Data Steward, Security Lead, ML Engineer
        ii. Auto-checks:
            1. Bias metrics (disparate impact < 0.8)
            2. Data sensitivity tags (e.g., PHI, PCI)
            3. Audit trail completeness
    b. **Status**: Pending → In Review → Approved/Rejected
    c. **Only Approved models** are allowed to generate **production-ready bundles**.
4. **Bundle Creation (Post-Approval)**
    a. From Design Node: **Project > Actions > Create Bundle**
    b. Tag bundle: fraud-detection-v1.0-approved-2025-10-27
    c. **Publish to Project Deployer** → visible only to Automation Node admins.

Governance Tip: Use **Govern API** to block bundle publishing if status ≠ Approved.

python

```
if govern_status != "APPROVED": raise Exception("Deployment blocked by Govern")
```

Production Deployment Process (Stage 5)

Only Approved Bundles from Stage 4.5

Stage 5: Production Deployment on Automation Nodes

1. **Bundle Deployment via Project Deployer**
    a. **Project Deployer** (central control plane) connects **Design → Automation Nodes**.
    b. Select **approved bundle → Deploy to Automation Node: PROD-ML-01**.
    c. DSS automatically:
        i. Validates schema consistency
        ii. Preloads **code environment** (e.g., python-3.11-deltadental)
        iii. Restarts **Scenarios** and **API services**

2. **Deployment Modes**

Mode	Use Case	Execution Node
Batch Scoring	Daily fraud risk batch	Automation Node (Spark)
Real-time API	Transaction-time scoring	API Node (REST endpoint)
Hybrid	External model fallback	External Model plugin

3. **API Service Activation**
    a. From Deployer: **Activate API Service fraud-predict-v1**
    b. Endpoint: POST https://api.deltadental.com/ml/fraud/predict
    c. SLA: < 100ms p95 latency
    d. Auto-scaling: 2–10 containers via **API Node pool config**

4. **Rollback & Versioning**
    a. Keep **last 3 bundles** in Deployer
    b. Rollback: Deploy previous version → zero-downtime switch

Governance Integration:

- **Govern Node** receives deployment event → logs **"Model Live in Prod"**
- Auto-creates **compliance artifact** for audit (SOX/HIPAA)

Stage 6: Production Monitoring & Continuous Governance

1. **Unified Monitoring on Automation Node**
    a. Dashboards:
        i. Model AUC trend (daily)
        ii. API latency & error rate
        iii. Data drift (KS test on transaction_freq, amount)

b. Alert rules in **Scenarios**:
    i. AUC < 0.88 → Slack + PagerDuty
    ii. Drift > 0.15 → trigger retraining pipeline

2. **Model Evaluation Store (Prod)**
    a. Daily ground-truth ingestion (from claims system)
    b. Auto-log metrics:

python
```
eval_store.create_evaluation(date, {'auc': 0.91, 'drift_ks': 0.04, 'disparate_impact': 0.76})
```

3. **Automated Retraining Pipeline**
    a. Triggered by drift or performance drop
    b. Flow: Stage 2 → 3 → 4 → 4.5 (auto-ARB fast-track) → 5 (Deployer)
    c. **Canary Deployment**: 10% traffic → new model → full rollout if stable
4. **Govern Node Oversight**
    a. All prod models **publish status** to Govern every 6h
    b. Quarterly **Model Health Report** auto-generated
    c. **Decommission workflow**: archive model after 90 days inactivity

Key Features and Tools (Updated)

Feature/Tool	Description
Project Deployer	Central hub for **approved bundle promotion** from Design → Automation Nodes
Automation Node	Isolated production runtime for batch jobs, scoring, and monitoring
API Node	Scalable real-time inference endpoints with SLA enforcement
Dataiku Govern Node	**Single source of truth** for model approval, risk, and compliance across lifecycle
Model Evaluation Store	Time-series performance logging in production
ARB + Govern Workflow	Enforced **pre-deployment gate** with audit trail

Governance Tips Across Stages (Delta Dental)

Stage	Governance Action
4.5	**ARB + Govern sign-off required** before bundle creation
5	**Deployer blocks unapproved bundles**; Audit Trail logs deployment actor
6	**Govern auto-flags** models with AUC drop >5% or bias shift

Key Features and Tools

Feature/Tool	Description
Connections Manager	Centralized hub for defining secure, reusable data source connections with credentials and schemas (Stage 2).
Datasets	Core entity for managing tabular data, supporting schema enforcement and sampling during acquisition (Stage 2).
Visual Recipes	Drag-and-drop interface for transformations like cleaning, joining, and filtering during preparation (Stage 2).
Code Recipes	Python, R, or SQL scripts for custom data ingestion, preparation, and feature engineering (Stages 2-4).
Analyze & Charts	Interactive tools for data profiling, visualization, and exploration to identify trends and issues (Stage 2).
Flow	Visual pipeline editor to track lineage and manage the end-to-end process across all stages.
Visual ML	Automated machine learning tool for training, tuning, and evaluating models (Stages 3-4).
Jupyter Notebooks	Interactive coding environment in the Lab for custom model development and validation (Stages 3-4).
Experiment Tracking	Built-in system to log, compare, and reproduce model experiments with performance metrics (Stage 3).
Score Recipe	Applies trained models to test or new data to generate predictions and validate performance (Stage 4).
Scenarios	Automates validation runs, schema checks, and alerts for continuous monitoring (Stages 2-4).
Spark Integration	Scalable execution engine for training and processing large-scale datasets (Stages 3-4).

Dataiku Govern	Governance layer for risk assessment, bias detection, approval workflows, drift analysis, and compliance (All stages).
Audit Trails	Non-repudiation logging of user access and changes for transparency and accountability (All stages).
Advanced Govern	Customizable governance for complex requirements, including scheduled sign-offs and delegation (All stages).

Notes: The Implementation Process in Dataiku DSS—covering Stages 2, 3, and 4—streamlines data handling, model development, and validation, with Dataiku Govern ensuring governance, compliance, and trust. For advanced setups, refer to the Dataiku DSS Documentation.

Additional Insights

- **Iteration & Automation**: Unlike linear processes, Dataiku encourages loops via scenarios (e.g., retrain on drift detection) and webapps for stakeholder feedback.
- **Governance & Collaboration**: Built-in project standards, role-based access, and a unified platform reduce silos, supporting diverse teams.
- **Modern Enhancements (2025)**: Increased focus on hybrid cloud integrations, automated documentation, and ecosystem-wide monitoring for faster scaling (e.g., 75% less pipeline code in financial services use cases).