Milestone 5: Deployment and Monitoring/Maintenance

CRISP-DM Phase 6 - Health and Demographic Patterns in South Africa (HDPSA)

Group A

17 October 2025

Table of Contents

# Executive Summary

This report documents the deployment of the HDPSA Random Forest model as a web-based Survey Data Analytics Tool, completing CRISP-DM Phase 6 (Deployment).

**Key Deliverables:**

* Deployment Strategy (Task 1)
* Functional web deployment at <http://localhost:3000/project> (Task 2)
* Comprehensive Maintenance Plan (Task 3)
* Automated Monitoring Plan (Task 4)

**Model Performance:** R² = 0.997, RMSE = 0.0554, MAE = 0.0382 (Milestone 4 validation)

**Target Users:** Health researchers, DHS survey planners, data quality teams

**Use Cases:** Survey data validation, gap-filling, consistency analysis

# Task 1: Plan Deployment (Business Strategy)

## 1.1 Stakeholder Analysis

### Primary Stakeholders

**Health Ministry Data Quality Teams**

* Need: Validate survey results against historical patterns
* Benefit: Reduced manual review time, improved data integrity

**DHS Survey Planners**

* Need: Estimate expected outcomes for budgeting and sample planning
* Benefit: Better resource allocation and planning efficiency

**Academic Researchers**

* Need: Fill gaps in incomplete historical datasets
* Benefit: More complete datasets for trend analysis

### Business Value

1. Detect data anomalies faster than manual review
2. Support survey planning efficiency
3. Enable longitudinal trend analysis
4. Demonstrate end-to-end ML deployment capability

## 1.2 Model Performance Summary

Model Performance Summary

| Metric | Value | Interpretation |
| --- | --- | --- |
| R² Score | 0.997 | Excellent variance explained |
| RMSE | 0.0554 | Low error on log-scaled target |
| MAE | 0.0382 | Mean absolute error |
| Training Samples | 560 | DHS survey records (1998-2016) |
| Features | 27 | Engineered features from Milestone 2 |

**Key Limitation:** The model achieves excellent fit on held-out data but reflects pattern-matching within the training distribution (1998-2016 aggregated survey statistics) rather than predictive power for individual outcomes or future time periods.

## 1.3 Deployment Tool Selection

### Tool Comparison

Deployment Tool Comparison (Scores out of 10)

| Tool | Scalability | UI\_Quality | Infrastructure\_Reuse | Total\_Score |
| --- | --- | --- | --- | --- |
| Next.js + Flask + Python | 9 | 10 | 10 | 8.8 |
| R Shiny | 4 | 6 | 2 | 5.5 |
| Python Streamlit | 6 | 7 | 4 | 6.3 |

### Selected: Next.js + Flask + Python

**Justification:**

1. **Infrastructure Reuse** - Leverages existing Next.js dashboard and Flask API
2. **Scalability** - Microservices architecture allows independent scaling
3. **Professional UI** - React provides polished, responsive interface
4. **Separation of Concerns** - UI, API, and ML logic are independent
5. **Team Familiarity** - Consistent with existing technology stack

**Architecture:**

Next.js UI (Port 3000)  
 ↓ HTTP REST  
Flask API (Port 5001)  
 ↓ Python call  
ML Service (Python module)

## 1.4 User Access and Documentation

### Access

* **Development:** <http://localhost:3000/project>
* **Production (Proposed):** Belgium Campus internal server or cloud hosting (Vercel + Render)

### Documentation

* User Guide with step-by-step instructions
* Model Card with limitations and ethics
* API Documentation with endpoint reference
* FAQ and troubleshooting guide

### Support

* Self-service: FAQ and in-app help
* Email support with 2-day SLA
* Quarterly stakeholder feedback sessions

## 1.5 Success Metrics

### Technical Metrics

Technical Performance Monitoring

| Metric | Threshold | Frequency |
| --- | --- | --- |
| R² Score | >= 0.95 | Weekly |
| RMSE | <= 0.07 | Weekly |
| MAE | <= 0.05 | Weekly |
| API Uptime | >= 99.5% | Daily |
| Response Latency | <= 500ms | Daily |

### Usage Metrics

* Active users per month (target: 10+ by Month 3)
* Prediction requests per week (target: 50+ by Month 2)
* Return user rate (target: 40% retention)

### Business Impact

* Time to validate survey batch (target: 25% reduction)
* Survey planning cycle time (target: 15% reduction)
* Research papers citing the tool

# Task 2: Deploy Model (Technical Implementation)

## 2.1 Architecture Overview

The model has been deployed using a three-tier microservices architecture:

**Frontend (Next.js):**

* Location: app/project/
* Dashboard: Model status, training interface
* Predictions: CSV upload, results display

**Backend API (Flask):**

* Location: 02\_Project/api/app.py
* Endpoints: /api/ml/train, /api/ml/predict-csv, /api/ml/model-info
* Features: Validation, error handling, CORS

**ML Service (Python):**

* Location: 02\_Project/api/ml\_service.py
* Methods: train\_model(), predict\_from\_dataframe(), inverse\_transform\_predictions()

### Data Flow

1. User uploads CSV via Next.js UI
2. Next.js sends file to Flask API
3. Flask validates and calls ML service
4. ML service preprocesses, predicts, and inverse transforms
5. Flask returns predictions to Next.js
6. Next.js displays results and enables download

## 2.2 Key Features

### Model Artifacts

**Saved Files (02\_Project/api/ml/outputs/):**

* final\_random\_forest\_model.pkl - Trained Random Forest (750 trees)
* model\_metadata.json - Training date, metrics, sample count
* feature\_importance.json - Feature rankings
* categorical\_mappings.json - Encoding dictionaries
* value\_log\_scaler.json - Mean/std for inverse transformation

**Version:** v1.0.0 (initial deployment)

### Input/Output Format

**Input Requirements:**

* CSV with 27 engineered features matching training schema
* Required fields: precision, indicator encoding, sample size tier, data quality scores

**Output Format:**

* predicted\_value\_log\_scaled - Scaled prediction for validation
* predicted\_value - Original percentage/rate (inverse-transformed)

**Example:**

Input: BCG vaccine, Age 12-23, High precision, Large sample  
Output: 95.2% vaccination rate

## 2.3 User Interface

### Dashboard Page (/project)

* Model Status Card with current metrics (R², RMSE, training date)
* “Train Model” button using local training data
* Navigation to Predictions page
* Quick Start instructions

### Predictions Page (/project/predict)

**Features:**

1. File upload (drag-and-drop or click)
2. Preview mode (first 10 predictions)
3. Full CSV download with predictions
4. Results table showing predicted values as percentages
5. Help information and input format guidance

**Results Display:**

* Table columns: Row, Predicted Value (%), Actual Value (%), Difference, Type, Importance
* Color-coded differences (green < 5%, yellow 5-10%, red > 10%)
* Bar chart showing prediction errors

## 2.4 Technology Stack

Technology Stack

| Component | Technology | Version |
| --- | --- | --- |
| Frontend | Next.js | 15.1.0 |
| Backend API | Flask | 3.0.0 |
| ML Framework | scikit-learn | 1.3.2 |
| Data Processing | pandas | 2.1.0 |

### Running Locally

Execute Z\_start-dev.bat which:

1. Installs npm dependencies
2. Creates Python virtual environments
3. Starts all 3 servers (Next.js, Flask Task API, Flask ML API)

**Access URLs:**

* ML Dashboard: <http://localhost:3000/project>
* Flask ML API: <http://localhost:5001>

# Task 3: Plan Maintenance

## 3.1 Retraining Procedures

### Retraining Triggers

1. **Performance Degradation:** RMSE > 0.07 or R² < 0.95 for two consecutive weeks
2. **Data Drift:** PSI > 0.2 or KS test p < 0.05 for multiple features
3. **New Data:** Quarterly national survey releases
4. **Business Changes:** Revised KPIs or policy focus

### Retraining Process

1. Collect new/updated data
2. Re-run Milestone 2 (Data Preparation) pipeline
3. Re-run Milestone 3 (Modeling) with hyperparameter tuning
4. Re-run Milestone 4 (Evaluation) on holdout sets
5. Compare to baseline and approve if metrics improve
6. Document and deploy new version

## 3.2 Model Versioning

### Semantic Versioning

**Format:** vMAJOR.MINOR.PATCH

* **MAJOR:** Breaking schema or architecture changes
* **MINOR:** New data, tuning, or non-breaking feature updates
* **PATCH:** Bug fixes with no material metric change

### Model Registry

**Location:** 02\_Project/Milestone\_5/model\_registry.csv

**Fields:** version, date, dataset\_hash, hyperparameters, rmse, mae, r2, psi, owner, artifact\_uri

**Retention Policy:**

* Active: Current production model
* Backup: Previous version (for rollback)
* Archive: Older versions (cold storage)

## 3.3 Model Retirement

### Retirement Criteria

1. Persistently low performance for > 3 months
2. Structural data changes undermining learned relationships
3. Documented bias not correctable by retraining
4. Ethical or compliance risks

### Retirement Process

1. Generate final performance report
2. Notify stakeholders
3. Hold formal review meeting
4. Archive model artifacts and logs
5. Disable prediction endpoint
6. Record decision in retirement log

## 3.4 Governance Schedule

Quarterly Governance Schedule

| Quarter | Focus | Outcome |
| --- | --- | --- |
| Q1 | Retraining and rollback drill | Validated runbook |
| Q2 | Backup and disaster recovery test | Restores verified |
| Q3 | Documentation and registry audit | Metadata updated |
| Q4 | Lifecycle and roadmap review | Continue / Retire decision |

# Task 4: Plan Monitoring

## 4.1 Performance Monitoring

### Core Metrics

Performance Monitoring Metrics

| Metric | Baseline | Alert\_Threshold | Frequency |
| --- | --- | --- | --- |
| RMSE | 0.0554 | > 0.07 | Weekly |
| MAE | 0.0382 | > 0.05 | Weekly |
| R² | 0.997 | < 0.95 | Weekly |

### Alerting

* **Green:** All metrics within thresholds
* **Amber:** Single metric near threshold
* **Red:** Threshold breach or multiple amber signals
* **Escalation:** Two consecutive amber or one red creates maintenance ticket

## 4.2 Data Drift Detection

### Drift Tests

**Input Feature Drift (Kolmogorov-Smirnov Test):**

* Compare current vs training distribution
* Test each top feature
* Flag if p < 0.05

**Population Stability (PSI):**

* Compute PSI per feature
* Flag if PSI > 0.2 (material shift) or > 0.3 (critical shift)

**Output Drift:**

* Track prediction mean/variance over 4-week window
* Flag if shift > 15% relative to baseline

### Remediation

Drift Remediation Matrix

| Severity | Criteria | Action |
| --- | --- | --- |
| Minor | PSI < 0.2 | Log and monitor next cycle |
| Moderate | PSI 0.2-0.3 | Schedule data refresh and review |
| Severe | PSI > 0.3 | Trigger retraining |

## 4.3 Automation Workflow (n8n)

### Workflow Overview

Weekly automated monitoring using n8n:

1. **Scheduler** - Triggers Mondays 07:00 SAST
2. **Fetch Data** - Calls Flask /api/metrics/window
3. **Compute Metrics** - Calculates RMSE, MAE, R²
4. **KS & PSI Tests** - Runs drift detection
5. **Render Report** - Generates weekly report
6. **Alert** - Sends Slack/email if thresholds breached
7. **Dashboard Refresh** - Updates monitoring pages

**Storage:** MongoDB for logs, metrics, and drift results

**Security:** API keys stored in n8n credentials (encrypted)

## 4.4 Ethical Monitoring

### Fairness Tests

**Subgroup Error Parity:**

* Compute MAE/RMSE across provinces, urban/rural, indicator domains
* Flag if subgroup MAE deviates > 25% from overall MAE

**Calibration Parity:**

* Quarterly review of calibration plots
* Ensure predictions not systematically biased

### Privacy (POPIA Compliance)

* No personal identifiers processed
* Aggregated data only
* Logs de-identified
* 12-month retention, then deletion

# Ethical Considerations

## Scope and Limitations

### In Scope

* Survey indicator value prediction based on metadata
* Data quality validation
* Historical gap-filling
* Performance monitoring

### Out of Scope

* Individual health outcome prediction (not household-level data)
* Causal analysis (cannot link domains like water → health)
* Time series forecasting beyond 2016
* Real-time streaming (batch processing only)

### Critical Limitations

* Aggregated statistics, not individual records
* Cannot establish causal relationships
* Limited to 27 specific features
* Historical patterns (1998-2016) may not generalize

## Fairness and Bias

**Mitigation Strategies:**

* Subgroup error analysis across provinces
* Document limitations in Model Card
* Alert users when input differs from training distribution
* Clear documentation that predictions are population-level only

## Transparency

**Model Card includes:**

* Purpose, training data sources, performance metrics
* Known limitations and failure modes
* Ethical considerations and recommended use cases
* Contact information for concerns

# Risk Assessment

Risk Assessment Summary

| Risk | Likelihood | Impact | Mitigation |
| --- | --- | --- | --- |
| Model predictions degrade over time | Medium | High | Automated monitoring, quarterly retraining reviews |
| Low user adoption | Medium | Medium | Clear documentation, stakeholder demos, training |
| Misuse for unsupported cases | Medium | Critical | Prominent disclaimers, user education, access restrictions |
| Security vulnerabilities | Medium | High | Input sanitization, file limits, security audit |

# Conclusion

## Key Accomplishments

This Milestone 5 report documents successful deployment of the HDPSA Random Forest model, completing CRISP-DM Phase 6:

1. **Strategic Planning** - Stakeholder analysis, tool selection, success metrics
2. **Technical Implementation** - 3-tier microservices architecture with professional UI
3. **Maintenance Strategy** - Retraining procedures, versioning, retirement policy
4. **Monitoring Plan** - Automated performance tracking and drift detection

## Model Positioning

**Legitimate Use Cases:**

* Survey data quality validation
* Gap-filling for incomplete datasets
* Survey planning and estimation
* Consistency analysis (1998-2016 distribution)

**Acknowledged Limitations:**

* Population-level patterns only
* No causal inference
* Historical context bound (1998-2016)
* Pattern-matching within training distribution

## Recommendations

### Short-Term (3 Months)

1. Complete MongoDB logging integration
2. Implement n8n monitoring workflow
3. Conduct stakeholder training
4. Security hardening and testing

### Medium-Term (6-12 Months)

1. Model retraining with additional data (2017-2024)
2. Add confidence intervals on predictions
3. Docker containerization
4. Video demos and case studies

### Long-Term (12+ Months)

1. Explore domain-specific models
2. Integration with DHS data platforms
3. Research collaboration and publications

# Appendices

## Appendix A: Model Card Summary

* **Model:** HDPSA Random Forest v1.0.0
* **Performance:** R² = 0.997, RMSE = 0.0554
* **Training Data:** 560 DHS survey records (1998-2016)
* **Use Cases:** Validation, planning, gap-filling
* **Limitations:** Population-level only, no causal inference

## Appendix B: API Endpoints

Flask API Endpoints

| Method | Endpoint | Description |
| --- | --- | --- |
| GET | /api/health | Service health check |
| GET | /api/ml/model-info | Model metadata and metrics |
| POST | /api/ml/train | Train model using local data |
| POST | /api/ml/predict-csv | Full CSV predictions |
| POST | /api/ml/predict-sample | Preview (10 rows) predictions |

## Appendix C: File Structure

02\_Project/  
├── api/  
│ ├── app.py (Flask API)  
│ ├── ml\_service.py (ML service)  
│ └── ml/outputs/ (Model artifacts)  
├── Milestone\_5/  
│ ├── Task\_1\_Deployment\_Strategy.md  
│ ├── Task\_3\_Maintenance\_Plan.md  
│ ├── Task\_4\_Monitoring\_Plan.md  
│ └── Milestone\_5\_Report.Rmd  
app/  
└── project/  
 ├── page.tsx (Dashboard)  
 └── predict/page.tsx (Predictions)

**Document Version:** 1.0 **Date:** 17 October 2025 **Status:** Submitted for Assessment