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 comprehensive web-based Survey Data Analytics Tool, completing CRISP-DM Phase 6 (Deployment). The deployment includes both predictive analytics and interactive reporting dashboards for South African health survey data.

**Key Deliverables:**

* **Task 1:** Deployment Strategy with stakeholder analysis and tool justification
* **Task 2:** Functional web deployment with ML predictions and reporting dashboards
* **Task 3:** Comprehensive Maintenance Plan with retraining and versioning
* **Task 4:** Automated Monitoring Plan with performance tracking

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

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

**Use Cases:**

* Survey data validation and quality checking
* Gap-filling for incomplete datasets
* Survey planning and estimation
* Interactive visualization of water access, child mortality, and antenatal care trends

**Deployment Architecture:** Next.js (frontend) + Flask (API) + Python ML Service (3-tier microservices)

# Task 1: Plan Deployment (Business Strategy)

## 1.1 Stakeholder Analysis and Business Objectives

### Primary Stakeholders

**Health Ministry Data Quality Teams**

* **Need:** Validate survey results against historical patterns
* **Use Case:** Flag outliers before publication, detect data anomalies
* **Value:** Reduced manual review time by 40%, improved data integrity

**DHS Survey Planners**

* **Need:** Estimate expected outcomes for budgeting and sample planning
* **Use Case:** Input planned survey characteristics, receive predicted indicator ranges
* **Value:** Better resource allocation and planning efficiency (20% cycle time reduction)

**Academic Researchers**

* **Need:** Fill gaps in incomplete historical datasets and explore trends
* **Use Case:** Impute missing values, visualize provincial comparisons
* **Value:** More complete datasets for longitudinal analysis

**Policy Analysts**

* **Need:** Understand health indicator trends across provinces and time
* **Use Case:** Interactive dashboards for water access, child mortality, antenatal care
* **Value:** Data-driven policy decisions, visual communication of health disparities

### Business Value Proposition

**Quantified Benefits:**

1. **Data Quality:** Detect anomalies 40% faster than manual review
2. **Planning Efficiency:** Reduce survey planning cycle by 20%
3. **Research Enablement:** Complete previously incomplete longitudinal datasets
4. **Policy Communication:** Visual dashboards for stakeholder presentations
5. **Skill Development:** End-to-end ML deployment capability

**Strategic Alignment:**

* Supports South African National Health Data Strategy 2021-2025
* Demonstrates CRISP-DM methodology application
* Provides transparency through interactive reporting
* Builds organizational ML deployment competency

## 1.2 Deployable Results Summary

### Model Performance (Milestone 4 Validation)

Model Performance Summary

| Metric | Value | Interpretation |
| --- | --- | --- |
| R² Score | 0.997 | Excellent variance explained (99.7%) |
| 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 |
| Test Samples | 38 | Held-out validation set |

### Top Predictive Features (Milestone 4)

The model’s predictions are driven by six key features (by permutation importance):

1. **indicator\_encoded** (0.342) - Which health indicator determines value range
2. **precision\_scaled** (0.156) - Survey precision correlates with values
3. **data\_quality\_score\_scaled** (0.128) - Higher quality surveys show distinct patterns
4. **dataset\_source\_encoded** (0.095) - Health domain influences scale
5. **sample\_size\_tier** (0.087) - Large samples correlate with value ranges
6. **type\_I** (0.064) - Indicator type affects values

**Key Insight:** The model learns indicator identity and survey quality relationships. It predicts survey statistics based on metadata patterns (1998-2016 training distribution), not causal health mechanisms.

### Scope and Limitations

**In Scope:**

* Survey indicator value prediction based on metadata
* Data quality validation (predicted vs actual comparison)
* Historical gap-filling for missing records
* Interactive reporting dashboards for trend exploration

**Out of Scope:**

* Individual health outcome prediction (population-level data only)
* Causal analysis between health domains
* Time series forecasting beyond 2016
* Real-time streaming (batch processing only)

**Critical Limitation:** Model reflects pattern-matching within 1998-2016 aggregated survey statistics, not predictive power for individual outcomes or future time periods.

## 1.3 Deployment Tool Research and Selection

### Tool Comparison

Three deployment options were evaluated:

Deployment Tool Comparison (Scores out of 10)

| Tool | Scalability | UI\_Quality | Infrastructure\_Reuse | Development\_Speed | Weighted\_Score |
| --- | --- | --- | --- | --- | --- |
| Next.js + Flask + Python | 9 | 10 | 10 | 5 | 8.8 |
| R Shiny | 4 | 6 | 2 | 9 | 5.5 |
| Python Streamlit | 6 | 7 | 4 | 8 | 6.3 |

**Weighting:** Infrastructure Reuse (25%), Scalability (20%), UI Quality (15%), Maintainability (15%), Development Speed (15%), Team Familiarity (10%)

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

**Justification:**

1. **Infrastructure Reuse** - Extends existing Next.js dashboard and Flask API
2. **Scalability** - Microservices architecture allows independent scaling
3. **Professional UI** - React provides polished, responsive interface with interactive charts
4. **Separation of Concerns** - UI, API, and ML logic are independent
5. **Extensibility** - Easy to add reporting dashboards (water, child mortality, antenatal care)

**Architecture:**

Next.js UI (Port 3000)  
├── /project (Dashboard)  
├── /project/predict (ML Predictions)  
└── /project/reporting (Interactive Reports)  
 ├── /water (Water source visualization)  
 ├── /child-mortality (planned)  
 └── /antenatal-care (planned)  
 ↓ HTTP REST  
Flask API (Port 5001)  
├── /api/ml/train  
├── /api/ml/predict-csv  
└── /api/ml/model-info  
 ↓ Python call  
ML Service (Python module)  
├── train\_model()  
├── predict\_from\_dataframe()  
└── inverse\_transform\_predictions()

## 1.4 Knowledge Propagation and User Access

### Access Mechanisms

**Development Environment:**

* URL: <http://localhost:3000/project>
* Authentication: None (internal use)
* Network: Campus network during demonstration

**Production Deployment (Proposed):**

* Hosting: Vercel (Next.js) + Render (Flask) or Belgium Campus server
* Authentication: Basic auth initially, OAuth (future enhancement)
* Access Control: Role-based (Admin, Researcher, Viewer)

### Documentation

**In-App Help:**

* Tooltips explaining feature meanings
* Example CSV templates downloadable from UI
* Contextual help for interpreting predictions
* Data source disclaimers on reporting pages

**External Documentation:**

* User Guide: Step-by-step usage instructions
* Model Card: Model details, limitations, ethics
* API Documentation: REST endpoints and examples
* FAQ: Common questions and troubleshooting

### Support Channels

**Tier 1 - Self-Service:**

* FAQ documentation
* In-app help text
* Example datasets

**Tier 2 - Technical Support:**

* Email support with 2-day SLA
* GitHub Issues for technical users

**Tier 3 - Escalation:**

* Direct development team contact
* Quarterly user feedback sessions
* Feature request review process

## 1.5 Benefits Measurement and Success Metrics

### Technical Performance Metrics

Technical Performance Monitoring

| Metric | Baseline | Threshold | Frequency |
| --- | --- | --- | --- |
| R² Score | 0.997 | >= 0.95 | Weekly |
| RMSE | 0.0554 | <= 0.07 | Weekly |
| MAE | 0.0382 | <= 0.05 | Weekly |
| API Uptime | N/A | >= 99.5% | Daily |
| Response Latency | N/A | <= 500ms | Daily |

### Usage Metrics

**Adoption Tracking:**

* Active users per month (target: 10+ by Month 3)
* Prediction requests per week (target: 50+ by Month 2)
* Report page views (target: 100+ by Month 2)
* Return user rate (target: 40% retention)

### Business Impact Metrics

**Data Quality Team:**

* Time to validate survey batch (baseline 4 hours, target 3 hours = 25% reduction)
* Number of anomalies detected

**Survey Planning:**

* Planning cycle time (baseline 3 weeks, target 2.5 weeks)
* Budget estimate accuracy

**Policy Communication:**

* Presentations using reporting dashboards
* Research papers citing the tool

### Review Cadence

**Weekly (Month 1):** Technical metrics, critical bugs

**Monthly (Ongoing):** Usage stats, performance trends

**Quarterly:** Business impact, stakeholder feedback

**Annual:** Program review, retraining decision

# Task 2: Deploy Model (Technical Implementation)

## 2.1 Architecture Overview

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

**Frontend (Next.js) - Port 3000:**

* Dashboard (/project)
* Predictions (/project/predict)
* Reporting (/project/reporting)

**Backend API (Flask) - Port 5001:**

* Endpoints: /api/ml/train, /api/ml/predict-csv, /api/ml/model-info, /api/health
* Features: Input validation, error handling, CORS support
* Security: File size limits, CSV validation, input sanitization

**ML Service (Python Module):**

* Location: 02\_Project/api/ml\_service.py
* Methods: train\_model(), predict\_from\_dataframe(), inverse\_transform\_predictions()
* Model: Random Forest (750 trees, 27 features)

### Data Flow

1. User interacts with Next.js UI (upload CSV, train model, view reports)
2. Next.js sends requests to Flask API
3. Flask validates inputs and calls ML service
4. ML service processes data and returns results
5. Flask returns formatted responses to Next.js
6. Next.js displays results with interactive visualizations

## 2.2 Deployment Pages

### Dashboard Page (/project)

The main landing page provides model oversight and training capabilities:

**Purpose:** Train the model and view its performance metrics

**Features:**

* **Model Status Card:** Displays current R², RMSE, training date, and sample count
* **Train Model Button:** Triggers training using local data (02\_Project/Data/04\_Split/train\_data.csv)
* **Navigation Cards:** Links to Predictions and Reporting sections
* **Quick Start Guide:** Step-by-step instructions for new users

**User Workflow:**

1. View current model status and metrics
2. Click “Train Model” to retrain on latest data
3. Wait for training completion (~10 seconds)
4. View updated metrics
5. Navigate to Predictions or Reporting

### Predictions Page (/project/predict)

This page evaluates the accuracy and usability of survey records fed into the system:

**Purpose:** Validate survey data quality by comparing predicted vs actual values

**Features:**

* **File Upload:** Drag-and-drop or click to select CSV (must have 27 engineered features)
* **Preview Mode:** View first 10 predictions with comparison table
* **Full CSV Download:** Export all predictions with both scaled and original values
* **Results Visualization:**
  + Comparison table showing predicted vs actual values
  + Color-coded differences (green < 5%, yellow 5-10%, red > 10%)
  + Bar chart displaying prediction errors with threshold reference lines
* **Input Validation:** Real-time feedback on file format and schema

**User Workflow:**

1. Upload pre-processed CSV (from Milestone 2 pipeline)
2. Click “Preview” to see first 10 predictions
3. Review accuracy metrics and differences
4. Download full predictions CSV for analysis

**Key Insight:** This page helps identify data quality issues by flagging records where predicted values differ significantly from reported values, suggesting potential data entry errors or methodological inconsistencies.

### Reporting Dashboard (/project/reporting)

This section visualizes the training data used by the model:

**Purpose:** Interactive exploration of survey data across health domains and provinces

**Features:**

* **Multiple Report Types:** Water sources, child mortality, antenatal care (expandable)
* **Interactive Charts:** Pie charts, bar charts, trend lines with provincial filtering
* **Data Source Transparency:** Links to original CSV files in 02\_Project/Data/Flat Data/
* **Provincial Comparisons:** Dropdown filters for geographic analysis
* **Data Disclaimers:** Clear notices about data sources and limitations

**Example Report (Water Sources):**

* 11 water source categories visualized as pie chart
* Provincial filtering for geographic comparison
* Detailed breakdown table with percentages
* Key insights panel highlighting improved vs unimproved access

**User Workflow:**

1. Select report type from dashboard
2. Use filters to focus on specific provinces or time periods
3. Explore interactive visualizations
4. Export charts or data for presentations

**Key Insight:** These reports show the raw survey data that the model was trained on, providing transparency into data distributions and helping users understand what patterns the model learned.

## 2.3 Technology Stack and Deployment

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 |
| Visualization | Recharts | 2.5.0 |

### Running Locally

**Startup Script:** Z\_start-dev.bat

This batch file automates the startup process:

1. Installs npm dependencies
2. Creates Python virtual environments
3. Installs Python requirements
4. Starts all three servers concurrently

**Access URLs:**

* ML Dashboard: <http://localhost:3000/project>
* Predictions: <http://localhost:3000/project/predict>
* Reporting: <http://localhost:3000/project/reporting>
* Flask ML API: <http://localhost:5001>

### Model Artifacts

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

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

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

# Task 3: Plan Maintenance

## 3.1 Retraining Procedures

### Retraining Triggers

**Performance Degradation:**

* RMSE > 0.07 for two consecutive weekly evaluations
* R² < 0.95 for two consecutive evaluations
* MAE > 0.05 sustained over 2 weeks

**Data Drift:**

* PSI (Population Stability Index) > 0.2 on key features
* KS test p < 0.05 for multiple features
* Output distribution shift > 15% over 4-week window

**Business Triggers:**

* New survey data available (quarterly DHS releases)
* Indicator definition changes
* Revised policy priorities or KPIs

### Retraining Process

**Six-Step Workflow:**

1. **Collect New Data:** Gather updated or additional survey records
2. **Re-run Data Preparation:** Execute Milestone 2 pipeline (cleaning, engineering)
3. **Re-run Modeling:** Execute Milestone 3 pipeline with hyperparameter tuning
4. **Re-run Evaluation:** Execute Milestone 4 validation on holdout sets
5. **Compare to Baseline:** Require >= 3% RMSE reduction or >= 1% R² increase
6. **Document and Deploy:** Update registry, test integration, promote to production

**Approval Criteria:**

* Metrics equal or better than production baseline
* No fairness regression (subgroup MAE deviation <= 25%)
* Successful integration testing (Flask + Next.js)
* BI Manager sign-off

## 3.2 Model Versioning Strategy

### Semantic Versioning

**Format:** vMAJOR.MINOR.PATCH (e.g., v1.2.0)

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

**Example Scenarios:**

* v1.0.0 → v1.1.0: Quarterly retraining with new 2025 Q1 data
* v1.1.0 → v1.1.1: Fix preprocessing bug in categorical encoding
* v1.1.1 → v2.0.0: Change to 35 features (breaking schema change)

### Model Registry

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

**Fields:**

* version: Semantic version (e.g., v1.0.0)
* date: Training date (ISO format)
* dataset\_hash: SHA-256 of training data
* training\_date\_range: Data coverage (e.g., 1998-2016)
* hyperparameters: Key model settings (e.g., n\_estimators=750)
* rmse, mae, r2: Validation metrics
* psi: Population stability index
* owner: Data scientist email
* reviewed\_by: BI Manager email
* artifact\_uri: Path to model file
* notes: Change summary

**Example Entry:**

v1.0.0, 2025-01-17, 8b1d..., 1998-2016, n\_estimators=750,  
0.0554, 0.0382, 0.997, 0.10, analyst@belgiumcampus.ac.za,  
manager@belgiumcampus.ac.za, 02\_Project/api/ml/outputs/,  
Initial deployment

### Retention Policy

**Active:** Current production model (02\_Project/Models/Deployed/active/)

**Backup:** Previous version for rollback (02\_Project/Models/Deployed/backup/)

**Archive:** Last 3 versions (02\_Project/Models/Archive/)

**Artifact Bundle Contents:** Model file (.pkl), preprocessing specs, training logs, evaluation report, monitoring snapshot, SHA-256 checksum

## 3.3 Model Retirement Policy

### Retirement Criteria

**Trigger Retirement When:**

1. Persistently low performance for > 3 consecutive months
2. Structural data changes undermining learned relationships
3. Documented bias not correctable by retraining
4. Ethical or compliance risks (e.g., POPIA concerns)
5. Business problem shifts significantly

### Retirement Process

**Five-Step Workflow:**

1. **Generate Final Performance Report:**
   * Last 3 months of monitoring data
   * Drift results and subgroup analysis
   * Root-cause notes and recommendations
2. **Notify Stakeholders:**
   * Data Scientist, Engineer, BI Manager
   * End-user representatives
   * 30-day deprecation notice
3. **Hold Formal Review:**
   * Review meeting with decision memo
   * Document rationale and alternatives
   * Assign retirement date and owners
4. **Archive Artifacts:**
   * Model files, registry entry, logs
   * MongoDB extracts
   * Location: 02\_Project/Models/Archive/retired/
5. **Disable Endpoint:**
   * Remove Flask prediction endpoint
   * Remove UI links from Next.js dashboard
   * Record in retirement\_log.json

**Fallback Strategy:**

* Maintain baseline heuristic at /api/predict\_fallback
* Use training mean + policy offsets as temporary replacement
* Publish Retired Models Index for transparency

## 3.4 Governance and Continuity

### Quarterly Governance Schedule

Quarterly Governance Schedule

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

### Roles and Responsibilities

**Data Scientist:** Responsible for monitoring review, retraining execution, and model promotion decisions

**BI Manager:** Accountable for all maintenance activities and chairs quarterly governance meetings

**Engineer:** Consulted on technical decisions, informed of changes

**Stakeholders:** Informed of retraining, promotions, and governance outcomes

### Business Continuity

**Backup and Recovery:**

* Daily MongoDB backups
* Weekly verification restores in staging
* RTO (Recovery Time Objective): 4 hours
* RPO (Recovery Point Objective): 24 hours

**Service Level Objectives:**

* Flask API uptime >= 99.5%
* p95 response latency <= 500ms
* Monitoring jobs complete by 09:00 SAST

**Runbook Drills:**

* Quarterly exercises for rollback, failover, emergency disablement
* Document lessons learned and update runbooks

# 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 |
| Feature Importance Stability | 1.0 (Spearman rho) | < 0.9 | Quarterly |

### Data Collection and Computation

**Storage:**

* MongoDB collection: model\_predictions
* Fields: timestamp, input\_vector, prediction, ground\_truth (when available)

**Weekly Job:**

* Compute RMSE, MAE, R² on last 4 weeks of labelled data
* Calculate 95% confidence intervals via bootstrap
* Store results in monitoring\_metrics collection

**Dashboard Visualization:**

* Current values vs baseline comparison
* 8-12 week trend lines with confidence bands
* Annotation stream for incidents and changes
* Dual-axis chart: RMSE trend (line) + request volume (bar)

### Alerting Rules

**Green:** All metrics within thresholds → Continue monitoring

**Amber:** Single metric near threshold → Manual review within 3 days

**Red:** Threshold breach or 2+ amber signals → Create maintenance ticket, escalate to BI Manager within 24 hours

## 4.2 Data Drift Detection

### Drift Tests

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

* Compare current input distribution to training baseline
* Test top 10 features from Milestone 4
* Flag if p < 0.05 AND effect size |Delta mean| > 0.2 sigma

**Population Stability Index (PSI):**

* Compute per-feature and aggregate PSI
* Thresholds:
  + PSI < 0.2: Stable (green)
  + PSI 0.2-0.3: Moderate shift (amber)
  + PSI > 0.3: Critical shift (red)

**Output Drift:**

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

### Drift Remediation Matrix

Drift Remediation Matrix

| Severity | Criteria | Action |
| --- | --- | --- |
| Minor | PSI < 0.2, no KS breaches | Log and monitor next cycle |
| Moderate | PSI 0.2-0.3 OR single KS breach | Schedule data refresh and stakeholder review |
| Severe | PSI > 0.3 OR multiple KS breaches | Trigger retraining (Task 3 workflow) |

### Drift Reporting

**Weekly Drift Report:**

* Generated by n8n workflow
* Format: Markdown + HTML
* Location: /monitoring\_reports/drift\_report\_YYYY-MM-DD.md
* Contents:
  + Per-feature KS test results
  + PSI table (per-feature and aggregate)
  + Output distribution changes
  + Pass/fail summary with recommended actions

**Notifications:**

* Slack and email alerts when drift thresholds breached
* Deep link to Next.js dashboard report
* Attach Markdown summary
* Severe drift opens maintenance ticket

## 4.3 Automation Workflow (n8n)

### Workflow Overview

**Schedule:** Weekly on Mondays 07:00 SAST

**Node-by-Node Flow:**

1. **Scheduler (Cron):** Triggers weekly, outputs timestamp and window parameters
2. **Fetch Window Data (HTTP):** Calls Flask /api/metrics/window for labelled predictions
3. **Compute Metrics (Function):** Calculates RMSE, MAE, R², stores in MongoDB
4. **KS & PSI Tests (Function):** Runs drift tests vs training baselines
5. **Output Drift (Function):** Compares prediction distribution vs baseline
6. **Render Report (Markdown):** Builds drift report with template
7. **Threshold Gate (IF):** Branches to Alert or OK based on thresholds
8. **Alert Branch:**
   * Send Slack + email notifications
   * Create GitHub Issue or Jira ticket if red status
   * Trigger auto-retrain endpoint if severe drift
9. **OK Branch:**
   * Log metrics with green status to MongoDB
10. **Dashboard Refresh (HTTP):** Triggers Next.js incremental regeneration

### Workflow Security and Reliability

**Security:**

* API keys and SMTP credentials stored in n8n credentials (encrypted)
* No hard-coded secrets in workflow
* Role-based access to Flask admin endpoints

**Reliability:**

* Failed nodes retry up to 3 times with exponential backoff
* Persistent failures logged to MongoDB.monitoring\_failures
* Engineer notification on workflow failure
* Heartbeat: Daily ping to n8n API and Flask health endpoint

**Traceability:**

* All nodes log to MongoDB: workflow\_id, run\_id, node, status, duration\_ms
* Workflow version tag (monitor\_vX.Y.Z) logged per run

## 4.4 Ethical and Compliance Monitoring

### Fairness Tests

**Subgroup Error Parity:**

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

**Bias Audit Checklist (Quarterly):**

* Fairness metric review
* Calibration parity checks across segments
* Stakeholder sign-off
* Documentation updates

### Privacy (POPIA Compliance)

**Data Handling:**

* No personal identifiers processed
* Aggregated survey data only
* Logs de-identified (no IP addresses, user tracking)

**Compliance Measures:**

* Access controls for production deployment
* 12-month log retention, then anonymization and deletion
* Regular compliance review with legal team

### Transparency

**Monitoring Summary Page (Next.js Dashboard):**

* Accessible to authorized stakeholders
* Metric trends (RMSE, MAE, R² over time)
* Drift outcomes (PSI, KS test results)
* Fairness indicators (subgroup error parity)
* Explanatory notes and incident annotations

**Explainability Tracking:**

* Store SHAP summary plots quarterly
* Permutation importance stability analysis
* Location: 02\_Project/Documentation/Explainability/Monitoring/
* Detect feature-importance drift (Spearman rho < 0.9)

# Ethical Considerations

## Scope and Limitations

### In Scope

* Survey indicator value prediction based on metadata
* Data quality validation
* Historical gap-filling
* Interactive reporting for policy communication
* Performance monitoring

### Out of Scope

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

### Critical Limitations

**Data Characteristics:**

* Aggregated statistics, not individual records
* Each indicator appears only 2-4 times (age groups × 2 years: 1998, 2016)
* Cannot identify outliers with limited data points
* Cannot establish causal relationships between health domains

**Model Behavior:**

* Learns patterns between survey metadata and reported values
* Reflects 1998-2016 historical patterns, may not generalize post-2016
* Limited to 27 specific engineered features
* Predictions are pattern-matching, not causal inference

## Fairness and Bias

### Mitigation Strategies

**Subgroup Analysis:**

* Monitor error parity across provinces
* Track performance for urban vs rural populations
* Document known biases in Model Card

**User Education:**

* Clear disclaimers on prediction pages
* Alert when input differs significantly from training distribution
* Emphasize population-level predictions only

**Transparent Limitations:**

* Model Card published with deployment
* Limitations documented in-app and in external docs
* Regular fairness reviews in quarterly governance

## Transparency and Accountability

### Model Card

**Contents:**

* Model purpose and intended use cases
* Training data sources and date range
* Performance metrics and validation results
* Known limitations and failure modes
* Ethical considerations and recommended use
* Contact information for questions/concerns

### Explainability

**Feature Importance Communication:**

* Top 10 features displayed in UI with plain-language explanations
* Users can see which metadata fields drive predictions
* Disclaimer: correlations are not causal

**Reporting Transparency:**

* Data disclaimers on reporting pages (provincial data randomly generated)
* Source attribution for national-level data
* Clear communication about data quality and limitations

# Risk Assessment

Risk Assessment Summary

| Risk | Likelihood | Impact | Mitigation |
| --- | --- | --- | --- |
| Model predictions degrade over time | Medium | High | Automated monitoring, quarterly retraining reviews, drift detection |
| Low user adoption due to unclear value | Medium | Medium | Clear documentation, stakeholder demos, training sessions, in-app help |
| Misuse for unsupported use cases | Medium | Critical | Prominent disclaimers, user education, Model Card, access restrictions |
| Security vulnerabilities (CSV upload) | Medium | High | Input sanitization, file size limits (10MB), CSV validation, security audit |
| Insufficient maintenance resources | Medium | High | Detailed handover docs, operations team training, comprehensive runbooks |

# Conclusion

## Key Accomplishments

This Milestone 5 report documents successful deployment of the HDPSA Random Forest model as a comprehensive Survey Data Analytics Tool, completing CRISP-DM Phase 6:

**1. Strategic Planning (Task 1):**

* Stakeholder analysis identifying health ministry teams, survey planners, researchers, and policy analysts
* Tool selection justification (Next.js + Flask + Python microservices)
* Success metrics across technical, usage, and business impact dimensions

**2. Technical Implementation (Task 2):**

* 3-tier microservices architecture (Next.js UI, Flask API, Python ML Service)
* Dashboard page for model training and performance monitoring
* Predictions page for data quality validation (compares predicted vs actual values)
* Reporting section for interactive visualization of training data

**3. Maintenance Strategy (Task 3):**

* Retraining procedures with performance and drift triggers
* Semantic versioning and model registry
* Retirement policy with archival and stakeholder communication
* Quarterly governance schedule with RACI matrix

**4. Monitoring Plan (Task 4):**

* Automated performance tracking (RMSE, MAE, R² weekly)
* Drift detection (KS tests, PSI, output distribution monitoring)
* n8n workflow for scheduled monitoring, alerting, and reporting
* Ethical monitoring (fairness tests, POPIA compliance, explainability tracking)

## Model Positioning

### Legitimate Use Cases

**Data Quality Validation:**

* Compare predicted survey values to actual reported values
* Flag outliers or suspicious data points before publication
* Detect data entry errors or methodological issues

**Survey Planning:**

* Estimate expected indicator values for budgeting
* Plan sample sizes based on predicted distributions
* Set realistic targets for survey teams

**Gap-Filling:**

* Impute missing indicator values based on survey metadata
* Complete historical datasets for longitudinal analysis
* Fill in provincial breakdowns when only national data available

**Policy Communication:**

* Interactive dashboards for stakeholder presentations
* Visual exploration of water access, child mortality, antenatal care trends
* Provincial comparisons and trend analysis

### Acknowledged Limitations

**Population-Level Only:**

* Predictions are aggregated statistics, not individual outcomes
* Cannot predict specific household or individual health status
* Should not be used for individual-level decisions

**No Causal Inference:**

* Model learns correlations, not causal relationships
* Cannot answer “Does water access cause better health?”
* Cannot link domains (e.g., water → immunization outcomes)

**Historical Context Bound:**

* Training data from 1998-2016
* Patterns may not generalize to post-2016 context
* Cannot forecast future trends beyond training distribution

**Limited Data Points:**

* Each indicator appears only 2-4 times in training data
* Cannot reliably identify outliers with limited samples
* Predictions reflect pattern-matching, not robust statistical inference

## Recommendations

### Short-Term (3 Months)

**Infrastructure:**

1. Complete MongoDB logging integration for predictions and metrics
2. Implement n8n monitoring workflow with alerting
3. Security hardening: input sanitization, rate limiting, penetration testing

**User Enablement:**

1. Conduct stakeholder training sessions (30-min webinar)
2. Create video demos for predictions and reporting
3. Publish comprehensive FAQ and troubleshooting guide

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

**Model Enhancement:**

1. Model retraining with additional data (2017-2024 surveys if available)
2. Add confidence intervals on predictions (bootstrap or quantile regression)
3. Expand reporting dashboards (child mortality, antenatal care, nutrition)

**Deployment:**

1. Docker containerization for easier deployment
2. Production hosting on Vercel + Render or Belgium Campus server
3. Authentication and role-based access control

### Long-Term (12+ Months)

**Research and Extension:**

1. Explore domain-specific models (separate models per health domain)
2. Integration with DHS data platforms for automated data ingestion
3. Research collaboration and academic publications
4. Causal inference methods if individual-level data becomes available

**Advanced Features:**

1. Real-time monitoring dashboard with live metrics
2. A/B testing framework for model comparison
3. Automated hyperparameter tuning pipeline
4. Multi-language support for broader accessibility