AI Data Cleansing & Validation Service

API Contract Specification v1.0

Document Information

- **Service Name**: AI Data Cleansing & Validation Service
- **API Version**: 1.0.0
- **Document Date**: September 24, 2025
- **Document Owner**: Shireen Hussain J
- **Classification**: Internal Use Only
- **Technology Stack**: Python 3.11 + TensorFlow + FastAPI

---

1. Service Overview

1.1 Purpose

The AI Data Cleansing & Validation Service leverages machine learning algorithms to automatically improve ESG data quality, detect anomalies, identify duplicates, and provide confidence scoring for sustainability metrics across the Sweep platform.

1.2 Core Capabilities

- **Anomaly Detection**: ML-powered identification of outliers and suspicious data points
- **Duplicate Detection**: Fuzzy matching and similarity scoring to identify duplicate records
- **Data Quality Scoring**: Automated quality assessment with confidence intervals
- **Missing Data Imputation**: Intelligent filling of data gaps using statistical models
- **Data Categorization**: Automated classification and tagging of ESG metrics
- **Validation Rules Engine**: Configurable business logic validation

1.3 Base URLs

- **Production**: `https://api.sweep.com/v1/ai-cleansing`
- **Staging**: `https://api-staging.sweep.com/v1/ai-cleansing`
- **Development**: `https://api-dev.sweep.com/v1/ai-cleansing`

1.4 Authentication

- **Primary**: OAuth 2.0 Bearer Token (JWT format)
- **Service-to-Service**: API Key authentication

- **Token Expiry**: 24 hours
- **Required Roles**: ESG_ANALYST, SUSTAINABILITY_MANAGER, SYSTEM_ADMIN

---

2. API Endpoints

2.1 Health Check

GET /health

Purpose: Returns AI service health status and model information

Security: No authentication required

Response (200 OK):

```json
{
  "status": "healthy",
  "timestamp": "2025-09-23T15:45:00Z",
  "version": "1.0.0",
  "uptime": "48h 30m 15s",
  "models": {
    "anomalyDetection": {
      "status": "loaded",
      "version": "v2.1.0",
      "accuracy": 0.995,
      "lastTrained": "2025-09-15T10:00:00Z"
    },
    "duplicateDetection": {
      "status": "loaded",
      "version": "v1.8.0",
      "precision": 0.952,
      "recall": 0.948
    },
    "dataQualityScoring": {
      "status": "loaded",
      "version": "v3.0.0",
      "meanAccuracy": 0.967
    }
  },
  "processingQueue": {
    "pendingJobs": 12,
    "averageProcessingTime": "2.3s"
  }
}
```

2.2 Data Quality Assessment

POST /assess/quality

Purpose: Analyze data quality and generate comprehensive quality scores

Request Body:

```json
{
  "jobId": "quality-job-123abc",
  "organizationId": "org-123",
  "datasetId": "dataset-456def",
  "metrics": [
    {
      "id": "metric-001",
      "entityId": "facility-dubai-001",
      "metricType": "SCOPE_1_EMISSIONS",
      "value": 1200.5,
      "unit": "kg CO2e",
      "timestamp": "2025-09-23T12:00:00Z",
      "sourceSystem": "SAP_ERP_PROD"
    },
    {
      "id": "metric-002",
      "entityId": "facility-dubai-001",
      "metricType": "ENERGY_CONSUMPTION",
      "value": 15600.0,
      "unit": "kWh",
      "timestamp": "2025-09-23T12:00:00Z",
      "sourceSystem": "SAP_ERP_PROD"
    }
  ],
  "assessmentOptions": {
    "includeAnomalyDetection": true,
    "includeDuplicateCheck": true,
    "includeCompletenessCheck": true,
    "includeConsistencyCheck": true,
    "confidenceThreshold": 0.85
  }
}
```

Response (202 Accepted):

json

```json
{
  "jobId": "quality-job-123abc",
  "status": "PROCESSING",
  "metricsReceived": 2,
  "estimatedProcessingTime": "30-60 seconds",
  "statusUrl": "/assess/quality/status/quality-job-123abc",
  "submittedAt": "2025-09-23T15:30:00Z"
}
```

GET /assess/quality/status/{jobId}

Purpose: Check quality assessment job status and results

Response (200 OK):

json

```
{
  "jobId": "quality-job-123abc",
  "status": "COMPLETED",
  "progress": {
    "totalMetrics": 2,
    "processedMetrics": 2,
    "percentComplete": 100
  },
  "results": {
    "overallQualityScore": 94.2,
    "qualityDistribution": {
      "excellent": 1,
      "good": 1,
      "fair": 0,
      "poor": 0
    },
    "assessmentSummary": {
      "anomaliesDetected": 0,
      "duplicatesFound": 0,
      "completenessScore": 100.0,
      "consistencyScore": 96.8
    },
    "metricResults": [
      {
        "id": "metric-001",
        "qualityScore": 96.5,
        "confidence": "HIGH",
        "issues": [],
        "improvements": [
          {
            "type": "PRECISION_ENHANCEMENT",
            "suggestion": "Consider using more decimal places for higher precision",
            "impact": "LOW"
          }
        ]
      },
      {
        "id": "metric-002",
        "qualityScore": 91.9,
        "confidence": "HIGH",
        "issues": [
          {
            "type": "UNIT_INCONSISTENCY",
            "severity": "LOW",
            "description": "Unit format could be standardized",
            "suggestedFix": "Use 'kWh' instead of 'kwh'"
        }
      ]
    }
  ]
 },
 "completedAt": "2025-09-23T15:31:15Z"
}
```

2.3 Anomaly Detection

POST /detect/anomalies

Purpose: Identify outliers and suspicious data points using ML models

Request Body:

```
json
```

```json
{
  "jobId": "anomaly-job-789xyz",
  "organizationId": "org-123",
  "detectionConfig": {
    "algorithm": "ISOLATION_FOREST",
    "sensitivity": "MEDIUM",
    "contextWindow": "30_DAYS",
    "includeSeasonality": true,
    "customThresholds": {
      "SCOPE_1_EMISSIONS": {
        "upperBound": 5000.0,
        "lowerBound": 0.0,
        "expectedRange": [800, 1500]
      }
    }
  },
  "metrics": [
    {
      "id": "metric-003",
      "entityId": "facility-abu-dhabi-001",
      "metricType": "SCOPE_1_EMISSIONS",
      "value": 8900.5,
      "unit": "kg CO2e",
      "timestamp": "2025-09-23T14:00:00Z",
      "historicalContext": {
        "previous30Days": [1200, 1150, 1300, 1100, 1250],
        "sameMonthLastYear": 1180
      }
    }
  ]
}
```

Response (202 Accepted):

```json
{
  "jobId": "anomaly-job-789xyz",
  "status": "PROCESSING",
  "algorithm": "ISOLATION_FOREST",
  "metricsReceived": 1,
  "estimatedProcessingTime": "45-90 seconds",
  "statusUrl": "/detect/anomalies/status/anomaly-job-789xyz"
}
```

GET /detect/anomalies/status/{jobId}

Purpose: Get anomaly detection results

Response (200 OK):

```json
```

```
{
  "jobId": "anomaly-job-789xyz",
  "status": "COMPLETED",
  "results": {
    "totalMetrics": 1,
    "anomaliesDetected": 1,
    "overallAnomalyScore": 0.89,
    "detectionSummary": {
      "highSeverity": 1,
      "mediumSeverity": 0,
      "lowSeverity": 0
    },
    "anomalies": [
      {
        "metricId": "metric-003",
        "anomalyScore": 0.89,
        "severity": "HIGH",
        "confidence": 0.94,
        "anomalyType": "STATISTICAL_OUTLIER",
        "explanation": {
          "reason": "Value significantly exceeds historical patterns",
          "details": "Current value (8900.5) is 6.8 standard deviations above historical mean (1200)",
          "context": "30-day historical average: 1200 kg CO2e"
        },
        "recommendations": [
          {
            "action": "VERIFY_SOURCE_DATA",
            "priority": "HIGH",
            "description": "Check source system for data entry errors or equipment malfunction"
          },
          {
            "action": "INVESTIGATE_OPERATIONAL_CHANGE",
            "priority": "MEDIUM",
            "description": "Verify if operational changes justify the increase"
          }
        ],
        "flags": [
          "EXCEEDS_EXPECTED_RANGE",
          "STATISTICAL_OUTLIER",
          "REQUIRES_VALIDATION"
        ]
      }
    ]
  },
  "modelInfo": {
    "algorithm": "ISOLATION_FOREST",
    "version": "v2.1.0",
    "trainingData": "90_days_historical",
    "lastUpdated": "2025-09-15T10:00:00Z"
  },
  "completedAt": "2025-09-23T15:32:30Z"
}
```

2.4 Duplicate Detection

POST /detect/duplicates

Purpose: Identify duplicate records using fuzzy matching and similarity algorithms

Request Body:

json

```
{
  "jobId": "duplicate-job-456def",
  "organizationId": "org-123",
  "detectionConfig": {
    "algorithm": "FUZZY_MATCHING",
    "similarityThreshold": 0.85,
    "matchingFields": [
      "entityId",
      "metricType",
      "timestamp",
      "value"
    ],
    "timeWindowMinutes": 60,
    "valueTolerancePercent": 5.0
  },
  "metrics": [
    {
      "id": "metric-004",
      "entityId": "facility-dubai-001",
      "metricType": "SCOPE_1_EMISSIONS",
      "value": 1200.5,
      "unit": "kg CO2e",
      "timestamp": "2025-09-23T12:00:00Z",
      "sourceSystem": "SAP_ERP_PROD"
    },
    {
      "id": "metric-005",
      "entityId": "facility-dubai-001",
      "metricType": "SCOPE_1_EMISSIONS",
      "value": 1201.0,
      "unit": "kg CO2e",
      "timestamp": "2025-09-23T12:05:00Z",
      "sourceSystem": "MANUAL_ENTRY"
    }
  ]
}
```

Response (202 Accepted):

```
json
```

```json
{
  "jobId": "duplicate-job-456def",
  "status": "PROCESSING",
  "algorithm": "FUZZY_MATCHING",
  "metricsReceived": 2,
  "estimatedProcessingTime": "15-30 seconds"
}
```

GET /detect/duplicates/status/{jobId}

Purpose: Get duplicate detection results

Response (200 OK):

```
json
```

```
{
  "jobId": "duplicate-job-456def",
  "status": "COMPLETED",
  "results": {
    "totalMetrics": 2,
    "duplicateGroups": 1,
    "totalDuplicates": 1,
    "duplicateMatches": [
      {
        "groupId": "dup-group-001",
        "primaryMetric": {
          "id": "metric-004",
          "confidence": "PRIMARY",
          "reason": "Earlier timestamp, automated source"
        },
        "duplicates": [
          {
            "id": "metric-005",
            "similarityScore": 0.92,
            "matchingFactors": [
              {
                "field": "entityId",
                "similarity": 1.0
              },
              {
                "field": "metricType",
                "similarity": 1.0
              },
              {
                "field": "timestamp",
                "similarity": 0.92,
                "note": "5 minutes apart"
              },
              {
                "field": "value",
                "similarity": 0.96,
                "note": "0.4% difference"
              }
            ],
            "recommendation": "MERGE_WITH_PRIMARY",
            "confidence": 0.92
          }
        ]
      }
    ],
    "resolutionSuggestions": [
      {
        "groupId": "dup-group-001",
        "action": "KEEP_AUTOMATED_SOURCE",
        "reasoning": "SAP_ERP_PROD data typically more reliable than manual entry",
        "impacts": {
          "metricsToRemove": 1,
          "dataQualityImprovement": "+2.3%"
        }
      }
    ]
  },
  "completedAt": "2025-09-23T15:31:45Z"
}
```

2.5 Data Cleansing Operations

POST /cleanse/apply

Purpose: Apply AI-recommended data cleansing operations

Request Body:

```
json
```

```
{
  "jobId": "cleanse-job-abc123",
  "organizationId": "org-123",
  "cleansingOperations": [
    {
      "type": "REMOVE_DUPLICATES",
      "targetMetrics": ["metric-005"],
      "reasoning": "Identified as duplicate of metric-004"
    },
    {
      "type": "CORRECT_ANOMALY",
      "targetMetric": "metric-003",
      "correction": {
        "originalValue": 8900.5,
        "correctedValue": 1200.5,
        "correctionReason": "Suspected data entry error based on historical patterns"
      }
    },
    {
      "type": "STANDARDIZE_UNITS",
      "targetMetrics": ["metric-002"],
      "standardization": {
        "originalUnit": "kwh",
        "standardUnit": "kWh"
      }
    }
  ],
  "approvalRequired": true,
  "notifyUsers": ["user-john-doe", "user-jane-smith"]
}
```

Response (202 Accepted):

```json
```

```
{
  "jobId": "cleanse-job-abc123",
  "status": "PENDING_APPROVAL",
  "operationsScheduled": 3,
  "impactSummary": {
    "metricsToModify": 2,
    "metricsToRemove": 1,
    "estimatedQualityImprovement": "+5.2%"
  },
  "approvalUrl": "/cleanse/approve/cleanse-job-abc123",
  "expiresAt": "2025-09-24T15:30:00Z"
}
```

POST /cleanse/approve/{jobId}

Purpose: Approve or reject cleansing operations

Request Body:

```json
{
  "decision": "APPROVED",
  "approvedBy": "user-john-doe",
  "approvalNotes": "Operations look correct based on data review",
  "partialApproval": {
    "approvedOperations": ["REMOVE_DUPLICATES", "STANDARDIZE_UNITS"],
    "rejectedOperations": ["CORRECT_ANOMALY"],
    "rejectionReason": "Need additional verification for anomaly correction"
  }
}
```

Response (200 OK):

```json
{
  "jobId": "cleanse-job-abc123",
  "status": "APPROVED",
  "executionScheduled": "2025-09-23T15:35:00Z",
  "operationsToExecute": 2,
  "operationsRejected": 1,
  "executionUrl": "/cleanse/execute/cleanse-job-abc123"
}
```

2.6 Data Imputation

POST /impute/missing

Purpose: Fill missing data using statistical and ML models

Request Body:

```json
{
  "jobId": "impute-job-789def",
  "organizationId": "org-123",
  "imputationConfig": {
    "method": "MACHINE_LEARNING",
    "algorithm": "RANDOM_FOREST",
    "confidenceThreshold": 0.75,
    "useHistoricalData": true,
    "usePeerComparison": true,
    "historicalWindowDays": 90
  },
  "missingDataPoints": [
    {
      "entityId": "facility-riyadh-001",
      "metricType": "ENERGY_CONSUMPTION",
      "expectedTimestamp": "2025-09-22T12:00:00Z",
      "context": {
        "previousValues": [14800, 15200, 14900, 15100],
        "sameTimeLastMonth": 15050,
        "facilityOperatingStatus": "NORMAL"
      }
    }
  ]
}
```

Response (202 Accepted):

```json
{
  "jobId": "impute-job-789def",
  "status": "PROCESSING",
  "algorithm": "RANDOM_FOREST",
  "missingPointsReceived": 1,
  "estimatedProcessingTime": "2-5 minutes"
}
```

GET /impute/missing/status/{jobId}

Purpose: Get data imputation results

Response (200 OK):

```json
```

```
{
  "jobId": "impute-job-789def",
  "status": "COMPLETED",
  "results": {
    "totalMissingPoints": 1,
    "successfulImputations": 1,
    "failedImputations": 0,
    "imputations": [
      {
        "entityId": "facility-riyadh-001",
        "metricType": "ENERGY_CONSUMPTION",
        "timestamp": "2025-09-22T12:00:00Z",
        "imputedValue": 14975.0,
        "confidence": 0.87,
        "confidenceInterval": {
          "lower": 14650.0,
          "upper": 15300.0,
          "level": 95
        },
        "methodology": {
          "algorithm": "RANDOM_FOREST",
          "features": [
            "historical_trend",
            "seasonal_pattern",
            "facility_capacity",
            "operating_status",
            "weather_conditions"
          ],
          "featureImportance": {
            "historical_trend": 0.45,
            "seasonal_pattern": 0.28,
            "facility_capacity": 0.15,
            "operating_status": 0.08,
            "weather_conditions": 0.04
          }
        },
        "qualityIndicators": {
          "reliability": "HIGH",
          "uncertainty": "LOW",
          "basisStrength": "STRONG"
        },
        "flags": [
          "ML_IMPUTED",
          "HIGH_CONFIDENCE",
          "PEER_VALIDATED"
        ]
      }
    ]
  },
  "completedAt": "2025-09-23T15:34:20Z"
}
```

2.7 Model Management

GET /models/info

Purpose: Get information about loaded AI models

Response (200 OK):

```
json
```

```json
{
  "models": [
    {
      "name": "anomaly_detection",
      "version": "v2.1.0",
      "algorithm": "ISOLATION_FOREST",
      "status": "ACTIVE",
      "performance": {
        "accuracy": 0.995,
        "precision": 0.992,
        "recall": 0.997,
        "f1Score": 0.994
      },
      "trainingInfo": {
        "trainingDate": "2025-09-15T10:00:00Z",
        "trainingDataSize": 150000,
        "trainingDuration": "4h 23m",
        "validationScore": 0.991
      },
      "capabilities": [
        "STATISTICAL_OUTLIERS",
        "TEMPORAL_ANOMALIES",
        "CONTEXTUAL_OUTLIERS"
      ]
    },
    {
      "name": "duplicate_detection",
      "version": "v1.8.0",
      "algorithm": "FUZZY_MATCHING_ENSEMBLE",
      "status": "ACTIVE",
      "performance": {
        "precision": 0.952,
        "recall": 0.948,
        "f1Score": 0.950
      },
      "capabilities": [
        "EXACT_MATCHING",
        "FUZZY_MATCHING",
        "SEMANTIC_SIMILARITY"
      ]
    }
  ],
  "totalModelsLoaded": 2,
  "memoryUsage": "2.4GB",
  "gpuUtilization": "45%"
}
```

POST /models/retrain

Purpose: Request model retraining with updated data

Request Body:

```json
{
  "modelName": "anomaly_detection",
  "retrainConfig": {
    "useLatestData": true,
    "dataWindowDays": 180,
    "includeUserFeedback": true,
    "hyperparameterTuning": true,
    "validationSplit": 0.2
  },
  "priority": "MEDIUM",
  "notificationWebhook": "https://client.sweep.com/webhooks/model-training"
}
```

Response (202 Accepted):

```json
{
  "retrainJobId": "retrain-job-456abc",
  "modelName": "anomaly_detection",
  "status": "QUEUED",
  "estimatedDuration": "2-4 hours",
  "datasetSize": 180000,
  "queuePosition": 2
}
```

2.8 Feedback and Learning

POST /feedback/submit

Purpose: Submit feedback on AI predictions to improve model accuracy

Request Body:

```json
```

```
{
  "feedbackType": "ANOMALY_DETECTION",
  "predictionId": "anomaly-job-789xyz",
  "metricId": "metric-003",
  "feedback": {
    "wasCorrect": false,
    "actualSeverity": "LOW",
    "userClassification": "OPERATIONAL_CHANGE",
    "explanation": "Value increase due to planned equipment upgrade",
    "correctiveAction": "UPDATE_BASELINE"
  },
  "submittedBy": "user-john-doe",
  "organizationId": "org-123"
}
```

Response (201 Created):

```json
{
  "feedbackId": "feedback-123abc",
  "status": "ACCEPTED",
  "impactAssessment": {
    "modelRetrainingTriggered": false,
    "baselineUpdateScheduled": true,
    "similarCasesFound": 3,
    "estimatedAccuracyImprovement": "+0.2%"
  },
  "acknowledgedAt": "2025-09-23T15:35:00Z"
}
```

---

3. Error Handling

3.1 Standard Error Response Format

```json
```

```
{
  "error": {
    "code": "AI_PROCESSING_ERROR",
    "message": "Machine learning model processing failed",
    "details": "Insufficient historical data for anomaly detection",
    "timestamp": "2025-09-23T15:30:00Z",
    "requestId": "req-ai-123abc",
    "path": "/v1/ai-cleansing/detect/anomalies",
    "method": "POST",
    "modelInfo": {
      "modelName": "anomaly_detection",
      "version": "v2.1.0",
      "errorCode": "INSUFFICIENT_DATA"
    }
  }
}
```

3.2 AI-Specific Error Codes

Status Code	Error Code	Description
400	`INVALID_ALGORITHM`	Specified ML algorithm not supported
400	`INSUFFICIENT_DATA`	Not enough data for ML processing
400	`INVALID_THRESHOLD`	Confidence threshold out of valid range
422	`MODEL_PREDICTION_FAILED`	ML model unable to generate prediction
422	`FEATURE_EXTRACTION_ERROR`	Error extracting features from data
503	`MODEL_NOT_LOADED`	Required ML model not available
503	`GPU_UNAVAILABLE`	GPU resources unavailable for processing
507	`MEMORY_INSUFFICIENT`	Insufficient memory for large dataset processing

---

4. Performance & SLA

4.1 Processing Time Targets

Operation Type	Target Processing Time	SLA Percentile
Quality Assessment	< 1 minute/1000 records	95th percentile
Anomaly Detection	< 2 minutes/1000 records	95th percentile
Duplicate Detection	< 30 seconds/1000 records	95th percentile
Data Imputation	< 5 minutes/100 missing points	90th percentile

4.2 Model Performance Standards

Model	Accuracy	Precision	Recall	F1-Score
Anomaly Detection	≥ 99.0%	≥ 99.0%	≥ 99.5%	≥ 99.2%
Duplicate Detection	≥ 95.0%	≥ 95.0%	≥ 94.5%	≥ 94.7%
Quality Scoring	≥ 96.0%	≥ 95.5%	≥ 96.5%	≥ 96.0%

4.3 Scalability Limits

- **Max Batch Size**: 10,000 records per request
- **Concurrent Jobs**: 50 per organization
- **Memory Usage**: 8GB per processing job
- **GPU Utilization**: Max 80% sustained load

---

5. Model Information

5.1 Anomaly Detection Models

Isolation Forest Model

- **Algorithm**: Isolation Forest with ensemble optimization
- **Features**: Statistical measures, temporal patterns, contextual data
- **Training Data**: 150,000+ ESG metrics with expert annotations
- **Update Frequency**: Weekly with new data, monthly full retrain
- **Confidence Scoring**: Based on outlier score and historical validation

LSTM Temporal Anomaly Detection

- **Algorithm**: Long Short-Term Memory neural network
- **Features**: Time-series patterns, seasonal decomposition
- **Sequence Length**: 30-day windows
- **Applications**: Detecting temporal pattern anomalies

5.2 Duplicate Detection Models

Fuzzy Matching Ensemble

- **Algorithms**: Levenshtein distance, Jaro-Winkler, semantic similarity
- **Similarity Threshold**: Configurable (default: 0.85)
- **Field Weights**: Entity ID (0.3), Timestamp (0.2), Value (0.3), Type (0.2)
- **Performance**: 95.2% precision, 94.8% recall

5.3 Data Quality Scoring

Multi-dimensional Quality Assessment

- **Dimensions**: Completeness, accuracy, consistency, timeliness, validity
- **Weighting**: Configurable by metric type and business rules
- **Scoring Range**: 0-100 with confidence intervals
- **Validation**: Cross-validation with domain expert ratings

---

6. Integration Examples

6.1 Python Client Example

python

```python
import requests
import json
from datetime import datetime

class AICleansingClient:
    def __init__(self, access_token, base_url="https://api.sweep.com/v1/ai-cleansing"):
        self.base_url = base_url
        self.headers = {
            'Authorization': f'Bearer {access_token}',
            'Content-Type': 'application/json'
        }

    def assess_data_quality(self, metrics, organization_id):
        """Assess data quality for a batch of metrics"""
        payload = {
            'jobId': f'quality-job-{datetime.now().strftime("%Y%m%d-%H%M%S")}',
            'organizationId': organization_id,
            'metrics': metrics,
            'assessmentOptions': {
                'includeAnomalyDetection': True,
                'includeDuplicateCheck': True,
                'confidenceThreshold': 0.85
            }
        }

        response = requests.post(
            f'{self.base_url}/assess/quality',
            json=payload,
            headers=self.headers
        )
        return response.json()

    def detect_anomalies(self, metrics, organization_id, sensitivity='MEDIUM'):
        """Detect anomalies in ESG metrics"""
        payload = {
            'jobId': f'anomaly-job-{datetime.now().strftime("%Y%m%d-%H%M%S")}',
            'organizationId': organization_id,
            'detectionConfig': {
                'algorithm': 'ISOLATION_FOREST',
                'sensitivity': sensitivity,
                'contextWindow': '30_DAYS',
                'includeSeasonality': True
            },
            'metrics': metrics
        }
        response = requests.post(
            f'{self.base_url}/detect/anomalies',
            json=payload,
            headers=self.headers
        )
        return response.json()

    def get_job_status(self, job_id, job_type='quality'):
        """Get status of a processing job"""
        endpoints = {
            'quality': '/assess/quality/status',
            'anomaly': '/detect/anomalies/status',
            'duplicate': '/detect/duplicates/status'
        }

        response = requests.get(
            f'{self.base_url}{endpoints[job_type]}/{job_id}',
            headers=self.headers
        )
        return response.json()

    def submit_feedback(self, prediction_id, metric_id, feedback_data, user_id, org_id):
        """Submit feedback on AI predictions"""
        payload = {
            'feedbackType': 'ANOMALY_DETECTION',
            'predictionId': prediction_id,
            'metricId': metric_id,
            'feedback': feedback_data,
            'submittedBy': user_id,
            'organizationId': org_id
        }

        response = requests.post(
            f'{self.base_url}/feedback/submit',
            json=payload,
            headers=self.headers
        )
        return response.json()

# Usage Example
client = AICleansingClient('your-access-token')

# Assess data quality
metrics_data = [
    {
        'id': 'metric-001',
        'entityId': 'facility-dubai-001',
        'metricType': 'SCOPE_1_EMISSIONS',
        'value': 1200.5,
        'unit': 'kg CO2e',
        'timestamp': '2025-09-23T12:00:00Z'
    }
]

quality_result = client.assess_data_quality(metrics_data, 'org-123')
print(f"Quality assessment job: {quality_result['jobId']}")

# Check job status
import time
time.sleep(30)  # Wait for processing
status = client.get_job_status(quality_result['jobId'], 'quality')
print(f"Quality score: {status['results']['overallQualityScore']}")
```

6.2 JavaScript Example

```
javascript
```

```
class AICleansingClient {
  constructor(accessToken, baseUrl = 'https://api.sweep.com/v1/ai-cleansing') {
    this.baseUrl = baseUrl;
    this.headers = {
      'Authorization': `Bearer ${accessToken}`,
      'Content-Type': 'application/json'
    };
  }

  async detectAnomalies(metrics, organizationId, config = {}) {
    const payload = {
      jobId: `anomaly-job-${Date.now()}`,
      organizationId,
      detectionConfig: {
        algorithm: 'ISOLATION_FOREST',
        sensitivity: 'MEDIUM',
        contextWindow: '30_DAYS',
        includeSeasonality: true,
        ...config
      },
      metrics
    };

    const response = await fetch(`${this.baseUrl}/detect/anomalies`, {
      method: 'POST',
      headers: this.headers,
      body: JSON.stringify(payload)
    });

    return response.json();
  }

  async detectDuplicates(metrics, organizationId) {
    const payload = {
      jobId: `duplicate-job-${Date.now()}`,
      organizationId,
      detectionConfig: {
        algorithm: 'FUZZY_MATCHING',
        similarityThreshold: 0.85,
        matchingFields: ['entityId', 'metricType', 'timestamp', 'value'],
        timeWindowMinutes: 60,
        valueTolerancePercent: 5.0
      },
      metrics
    };
    const response = await fetch(`${this.baseUrl}/detect/duplicates`, {
      method: 'POST',
      headers: this.headers,
      body: JSON.stringify(payload)
    });

    return response.json();
  }

  async pollJobStatus(jobId, jobType, maxAttempts = 30) {
    const endpoints = {
      quality: '/assess/quality/status',
      anomaly: '/detect/anomalies/status',
      duplicate: '/detect/duplicates/status'
    };

    for (let i = 0; i < maxAttempts; i++) {
      const response = await fetch(
        `${this.baseUrl}${endpoints[jobType]}/${jobId}`,
        { headers: this.headers }
      );

      const result = await response.json();

      if (result.status === 'COMPLETED' || result.status === 'FAILED') {
        return result;
      }

      // Wait 2 seconds before next poll
      await new Promise(resolve => setTimeout(resolve, 2000));
    }

    throw new Error('Job polling timeout');
  }
}

// Usage
const client = new AICleansingClient('your-access-token');

async function processESGData() {
  const metrics = [
    {
      id: 'metric-001',
      entityId: 'facility-abu-dhabi-001',
      metricType: 'SCOPE_1_EMISSIONS',
      value: 8900.5, // Potentially anomalous value
      unit: 'kg CO2e',
    timestamp: '2025-09-23T14:00:00Z'
   }
  ];

  try {
   // Detect anomalies
   const anomalyJob = await client.detectAnomalies(metrics, 'org-123');
   console.log('Anomaly detection started:', anomalyJob.jobId);

   // Poll for results
   const anomalyResults = await client.pollJobStatus(anomalyJob.jobId, 'anomaly');

   if (anomalyResults.results.anomaliesDetected > 0) {
    console.log('Anomalies found:', anomalyResults.results.anomalies);

    // Handle anomalies (e.g., flag for review, auto-correct, etc.)
    for (const anomaly of anomalyResults.results.anomalies) {
     console.log(`Anomaly in ${anomaly.metricId}: ${anomaly.explanation.reason}`);
    }
   }

  } catch (error) {
   console.error('Error processing ESG data:', error);
  }
}

processESGData();
```

7. Advanced Configuration

7.1 Custom Anomaly Detection Rules

```
json
```

```
{
  "customRules": [
    {
      "name": "energy_consumption_facility_type",
      "metricType": "ENERGY_CONSUMPTION",
      "conditions": {
        "facilityType": "MANUFACTURING",
        "expectedRange": {
          "min": 10000,
          "max": 50000,
          "unit": "kWh"
        },
        "seasonalAdjustment": true,
        "peakHours": ["08:00-18:00"]
      }
    },
    {
      "name": "scope1_emissions_fuel_correlation",
      "metricType": "SCOPE_1_EMISSIONS",
      "correlationChecks": [
        {
          "correlateWith": "FUEL_CONSUMPTION",
          "expectedRatio": {
            "min": 2.0,
            "max": 2.5,
            "tolerance": 0.1
          }
        }
      ]
    }
  ]
}
```

7.2 Model Configuration Options

```
json
```

```json
{
  "modelConfigurations": {
    "anomalyDetection": {
      "isolationForest": {
        "contamination": 0.05,
        "nEstimators": 200,
        "maxSamples": 1000,
        "randomState": 42
      },
      "lstm": {
        "sequenceLength": 30,
        "hiddenUnits": 64,
        "dropoutRate": 0.2,
        "learningRate": 0.001
      }
    },
    "duplicateDetection": {
      "fuzzyMatching": {
        "jaroWinklerWeight": 0.4,
        "levenshteinWeight": 0.3,
        "semanticWeight": 0.3,
        "thresholds": {
          "exact": 1.0,
          "high": 0.9,
          "medium": 0.75,
          "low": 0.6
        }
      }
    }
  }
}
```

---

8. Monitoring & Observability

8.1 AI-Specific Metrics

The service exposes specialized metrics at `/metrics` endpoint:

- `ai_cleansing_model_predictions_total`: Total predictions by model and outcome
- `ai_cleansing_model_accuracy`: Current model accuracy scores
- `ai_cleansing_processing_duration_seconds`: Processing time by operation type
- `ai_cleansing_anomaly_detection_rate`: Anomaly detection rate over time
- `ai_cleansing_false_positive_rate`: False positive rates by model

- `ai_cleansing_model_memory_usage_bytes`: Memory usage per model
- `ai_cleansing_gpu_utilization_percent`: GPU utilization metrics

8.2 Model Performance Dashboards

json

```json
{
  "dashboards": {
    "anomalyDetection": {
      "metrics": [
        "detection_accuracy_over_time",
        "false_positive_trend",
        "processing_latency",
        "confidence_score_distribution"
      ],
      "alerts": [
        {
          "metric": "accuracy",
          "threshold": 0.95,
          "action": "RETRAIN_MODEL"
        },
        {
          "metric": "false_positive_rate",
          "threshold": 0.05,
          "action": "ALERT_TEAM"
        }
      ]
    }
  }
}
```

8.3 Logging Format

json

```json
{
  "timestamp": "2025-09-23T15:45:00Z",
  "level": "INFO",
  "service": "ai-cleansing",
  "operation": "ANOMALY_DETECTION",
  "requestId": "req-ai-123abc",
  "organizationId": "org-123",
  "modelInfo": {
    "name": "isolation_forest",
    "version": "v2.1.0",
    "processingTime": 2.3,
    "confidence": 0.94
  },
  "results": {
    "anomaliesDetected": 1,
    "severity": "HIGH",
    "actionRequired": true
  },
  "performance": {
    "memoryUsed": "245MB",
    "gpuUtilization": "67%",
    "cpuTime": "1.8s"
  }
}
```

---

9. Security & Compliance

9.1 AI Model Security

- **Model Protection**: Encrypted model storage and secure loading
- **Input Validation**: Sanitization of all input data before ML processing
- **Output Filtering**: Validation of ML outputs to prevent data leakage
- **Access Control**: Role-based access to different AI capabilities

9.2 Data Privacy in ML

- **Data Anonymization**: PII removal before model training
- **Federated Learning**: Support for distributed training without data sharing
- **Model Interpretability**: Explainable AI for regulatory compliance
- **Bias Detection**: Regular assessment for algorithmic bias

9.3 Model Governance

- **Version Control**: Complete versioning of all models and training data
- **Audit Trails**: Tracking of model decisions and user feedback
- **A/B Testing**: Controlled rollout of model updates
- **Rollback Procedures**: Ability to revert to previous model versions

---

10. Support & Troubleshooting

10.1 Common Issues and Solutions

Issue	Cause	Solution
Low Confidence Scores	Insufficient training data	Increase historical data window
High False Positive Rate	Model too sensitive	Adjust sensitivity threshold
Slow Processing	Large batch size	Reduce batch size or use streaming
Memory Errors	Dataset too large	Implement batch processing
Model Not Loaded	Service restart	Check model loading status

10.2 Performance Tuning

python

```python
# Example configuration for optimal performance
performance_config = {
    "batchProcessing": {
        "maxBatchSize": 1000,  # Optimal for memory usage
        "parallelJobs": 4,      # Based on CPU cores
        "memoryLimit": "2GB"    # Per job limit
    },
    "modelOptimization": {
        "useGPU": True,
        "precisionMode": "FP16",    # Faster inference
        "batchInference": True,     # Process multiple samples together
        "caching": {
            "enabled": True,
            "ttl": "1h",           # Cache results for 1 hour
            "maxSize": "500MB"
        }
    }
}
```

10.3 Error Recovery Procedures

1. **Model Failure**: Automatic fallback to previous stable version
2. **Memory Issues**: Graceful degradation to smaller batch sizes
3. **GPU Unavailable**: Fallback to CPU processing
4. **Training Data Corruption**: Use validated backup datasets
5. **Service Overload**: Queue management with priority handling

---

11. Regulatory & Ethical Considerations

11.1 AI Ethics Framework

- **Transparency**: Clear documentation of model decision-making processes
- **Fairness**: Regular bias testing across different entity types and regions
- **Accountability**: Human oversight for all automated decisions
- **Privacy**: Strict data handling procedures and anonymization

11.2 Regulatory Compliance

- **AI Act (EU)**: Classification as limited-risk AI system
- **GDPR Article 22**: Right to explanation for automated decisions
- **SOX Compliance**: Audit trails for financial ESG data processing
- **ISO/IEC 23053**: Framework for AI risk management

---

12. Roadmap & Future Enhancements

12.1 Short-term (3-6 months)

- **Enhanced NLP Models**: Processing of unstructured ESG reports
- **Real-time Streaming**: Continuous anomaly detection for IoT data
- **Advanced Visualization**: Interactive anomaly exploration dashboards
- **Multi-language Support**: ESG data processing in multiple languages

12.2 Medium-term (6-12 months)

- **Reinforcement Learning**: Self-improving anomaly detection
- **Causal Inference**: Understanding cause-effect relationships in ESG data
- **Federated Learning**: Collaborative model training across organizations
- **Automated Report Generation**: AI-generated ESG insights and summaries

12.3 Long-term (12+ months)

- **Quantum ML**: Exploration of quantum computing for complex optimizations
- **Graph Neural Networks**: Analysis of supply chain relationships
- **Foundation Models**: Large language models for ESG domain expertise
- **Predictive ESG Analytics**: Future ESG performance forecasting

---

Document Control

- **Created By**: Shireen Hussain J
- **AI/ML Review**: Data Science Team Lead
- **Security Review**: AI Ethics Committee
- **Last Updated**: September 24, 2025
- **Next Review**: December 23, 2025
- **Classification**: Confidential - Internal Use Only

This API contract represents the current state of AI capabilities. Model performance and features may be updated based on ongoing research and development.