Technical Specification

Feature Extraction Module – Project Atlas (Week 2)

---

1. Document Overview

This document specifies the design, architecture, and integration details of the **Feature Extraction Module** developed for processing unstructured text data (logs, notes) within the Analytics Engine.

2. Objective

The objective of this module is to:

- Enable processing of unstructured text data within the existing pipeline
- Extract lightweight, structured features suitable for downstream analytics
- Integrate with the current feature extraction stage without refactoring
- Ensure compliance with data governance policies (no PII exposure)
- Operate efficiently within shared Apache Spark cluster constraints

This implementation is a **prototype**, focused on integration readiness and compliance rather than advanced NLP techniques.

3. Scope

In Scope

- Unstructured text processing (logs, notes)
- Feature extraction using Python and Apache Spark
- Input validation and basic error handling
- Unit-testable extraction logic
- Design and integration documentation

Out of Scope

- Modifications to upstream ETL
- Changes to downstream analysis components
- UI or dashboard updates
- Schema migration
- External cloud or NLP APIs

4. System Context

Existing Pipeline Flow

Legacy S3 Storage

ETL & Validation Layer

Feature Extraction Stage (This Module)

Downstream Analytics Components

The current pipeline processes structured data only. This module extends the feature extraction stage to support unstructured text inputs.

5. Architecture Overview

Design Principles

- Lightweight and deterministic
- Spark-native transformations
- Minimal dependencies
- Governance-first approach

6. Input Specification

Input Source

- Legacy S3 buckets ingested by existing ETL pipeline

Input Format

- Apache Spark DataFrame
- Required column:
    - text (string): unstructured text content

Assumptions

- Core schema fields are stable
- Text may be empty or malformed
- Additional columns may exist but are ignored

7. Output Specification

Output Format

- Apache Spark DataFrame

Output Schema

Column Name	Type	Description
text	string	Original input text
text_length	int	Length of sanitized text
word_count	int	Number of words
keyword_density	float	Word count divided by text length
empty_text_flag	int	1 if text is empty, else 0

8. Feature Extraction Logic

Text Sanitization

To comply with data governance requirements:

- All text is converted to lowercase
- Email addresses are removed
- Numeric identifiers are removed
- Sanitized text is used for feature computation

Extracted Features

- **Text Length:** Indicates content size
- **Word Count:** Measures text complexity
- **Keyword Density:** Lightweight analytical signal
- **Empty Text Flag:** Data quality indicator

9. Validation and Error Handling

- Input schema validation ensures required columns exist
- Custom exceptions are raised for invalid inputs
- Spark-native operations ensure fault tolerance during batch execution

10. Integration Points

Upstream Integration

- Consumes DataFrame produced by existing ETL and validation stages
- No upstream changes required

Downstream Integration

- Outputs structured features in expected schema
- Downstream components remain unchanged
- No schema migration required

11. Performance Considerations

- Uses Spark columnar operations only
- Avoids Python UDFs
- Optimized for shared cluster execution
- Prototype benchmarks target throughput within 20% of estimated baseline

12. Security and Data Governance

Governance Measures

- No external APIs or SaaS tools
- PII indicators removed during sanitization
- No raw PII emitted as features
- Processing confined to internal Spark cluster

Compliance Alignment

- Adheres to Sparwix data governance framework
- Designed to pass pipeline validation checks

13. Testing Strategy

- Unit tests implemented using pytest
- Local Spark session for testing
- Covered scenarios:
    - Missing text column

    - Empty text input
    - PII-like patterns
    - Multi-row inputs
- Target ≥90% coverage for extraction logic

14. Trade-Offs and Design Decisions

Decision	Rationale
Rule-based extraction	Predictable, fast, resource-efficient
No advanced NLP	Avoids heavy compute and dependencies
Spark-native operations	Better scalability and integration
Simple feature set	Meets SLA and prototype scope

15. Future Enhancements

- Expanded keyword dictionaries
- Language detection
- Enhanced sentiment indicators
- Configurable sanitization rules
- Support for additional unstructured sources

16. Conclusion

This Feature Extraction Module prototype fulfills all requirements for Project Atlas. It delivers a compliant, scalable, and integration-ready solution for unstructured text processing while respecting architectural and governance constraints.