

Disclaimer

The analyses, their interpretation, and related information contained herein are made and provided subject to the assumptions, methodologies, caveats, and variables described in this report and are based on third party sources and data reasonably believed to be reliable. No warranty is made as to the completeness or accuracy of such third party sources or data.

As with any attempt to estimate future events, the forecasts, projections, conclusions, and other information included herein are subject to certain risks and uncertainties, and are not to be considered guarantees of any particular outcome.

All reproduction rights, quotations, broadcasting, publications reserved. No part of this presentation may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without express written consent of IQVIA.

Copyright © 2025 IQVIA. All rights reserved. IQVIA® is a registered trademark of IQVIA Inc. in the United States and various other countries.

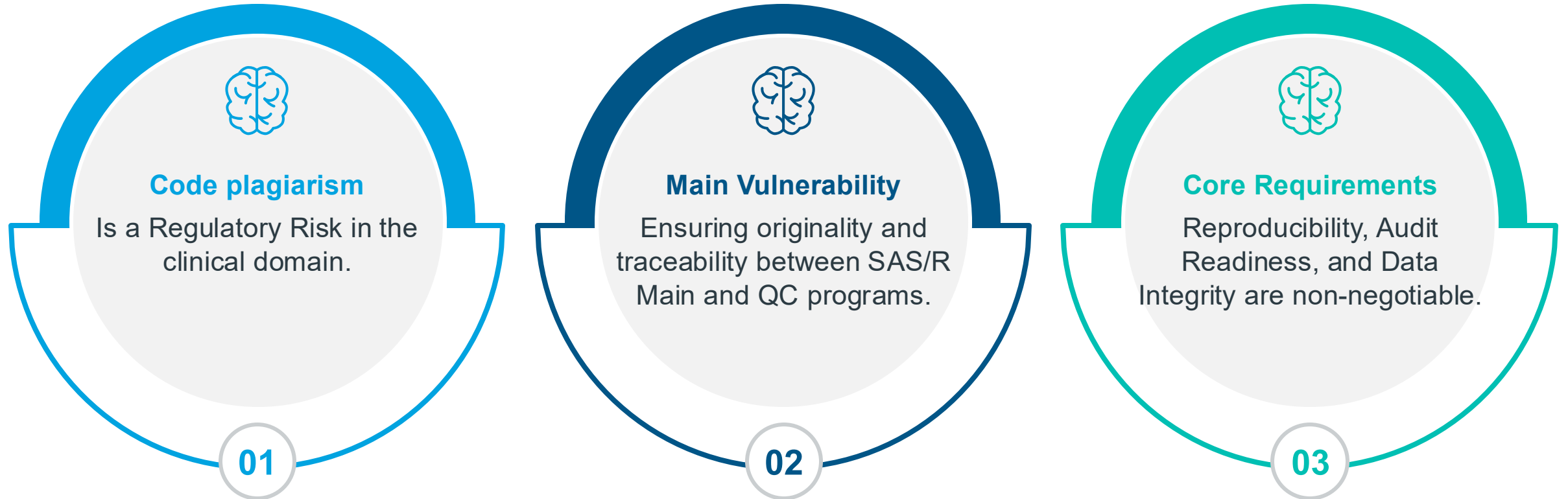


Code Compare -

A Machine Learning Based Framework for Code Plagiarism Analysis and Detection

Saroj Kumar Jena, IQVIA, Amsterdam, NL

The Problem: Code Integrity in Clinical Programming



Limitations of Traditional Solutions



Rely on Superficial String Matching.



Cannot handle Structural Variations (renamed variables, reordered logic).

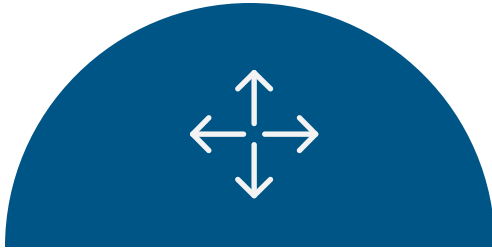


Lack Domain-Specific language support (SAS, R).



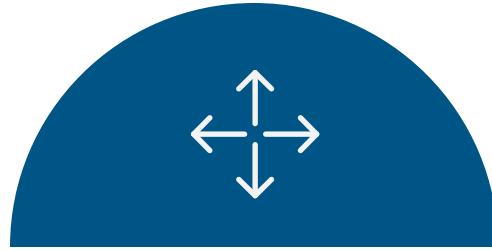
Result in Time-Consuming Manual Reviews.

Our Unique Contribution: Semantic Detection



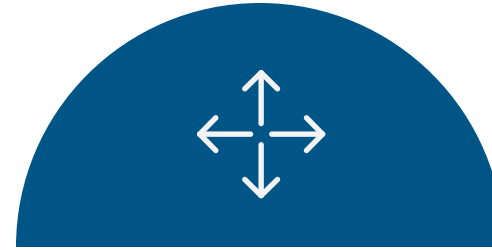
The Solution

A Machine Learning-based framework that analyzes Semantic Similarity.



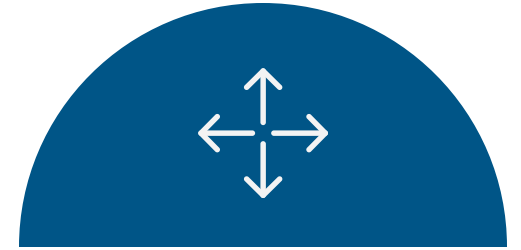
Robustness

Immune to superficial changes like formatting or variable renaming.



Scalable

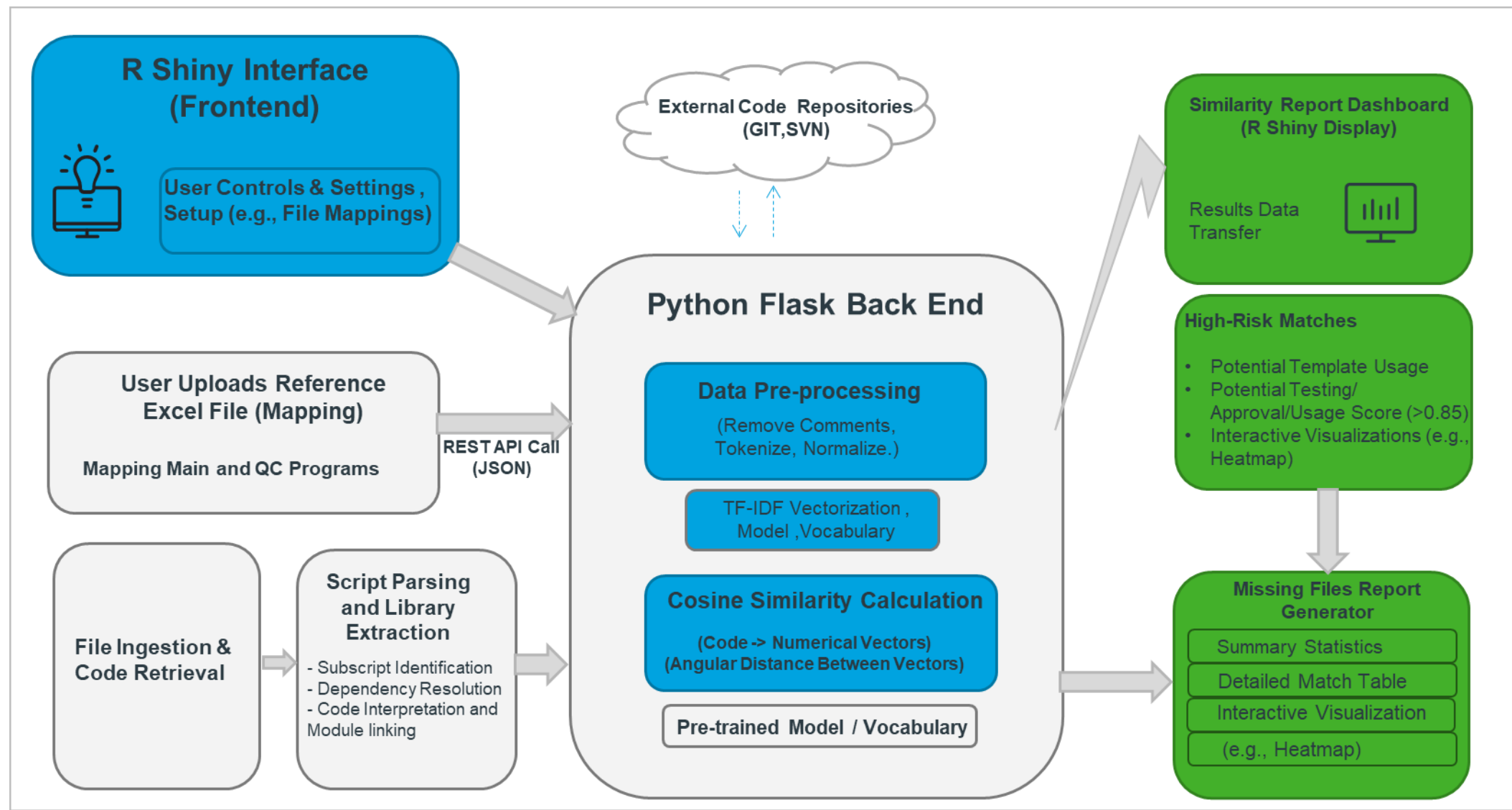
Deployed in a modular R Shiny/Python architecture.



Language-Agnostic Design

Supports SAS-to-SAS and R-to-R comparisons.

Framework Architecture Overview



Methodology: Data Preprocessing (The Cleanup)

Goal: Transform raw code into a clean, mathematical input.

- 1. Noise Removal:** Strip comments, extra whitespace, and non-essential tokens.
- 2. Tokenization & Normalization:** Break code into units (tokens) and standardize formatting.

The ML Core: TF-IDF Explained

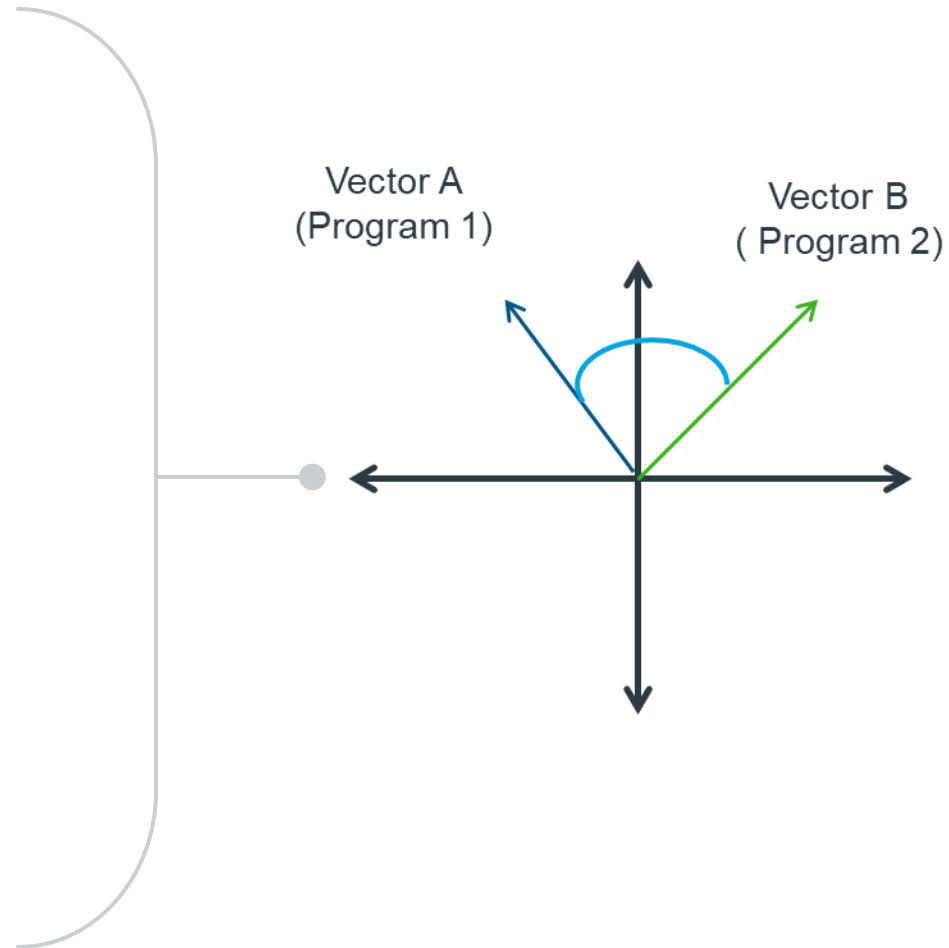
- **TF-IDF:** Term Frequency-Inverse Document Frequency.
- **TF (Term Freq.):** How often a token appears in *one program*.
- **IDF (Inverse Doc. Freq.):** How unique that token is across *all programs* (the corpus).
- **Result:** Code is converted into a **Numerical Vector** that weights the distinct, meaningful constructs.

The ML Core: Cosine Similarity

Key Metric: Measures the Angular Distance between two code vectors.

Why it's Robust: A small angle (score near 1.0) means the vector directions are close, indicating similar semantic structure, regardless of variable names.

Visual Concept:

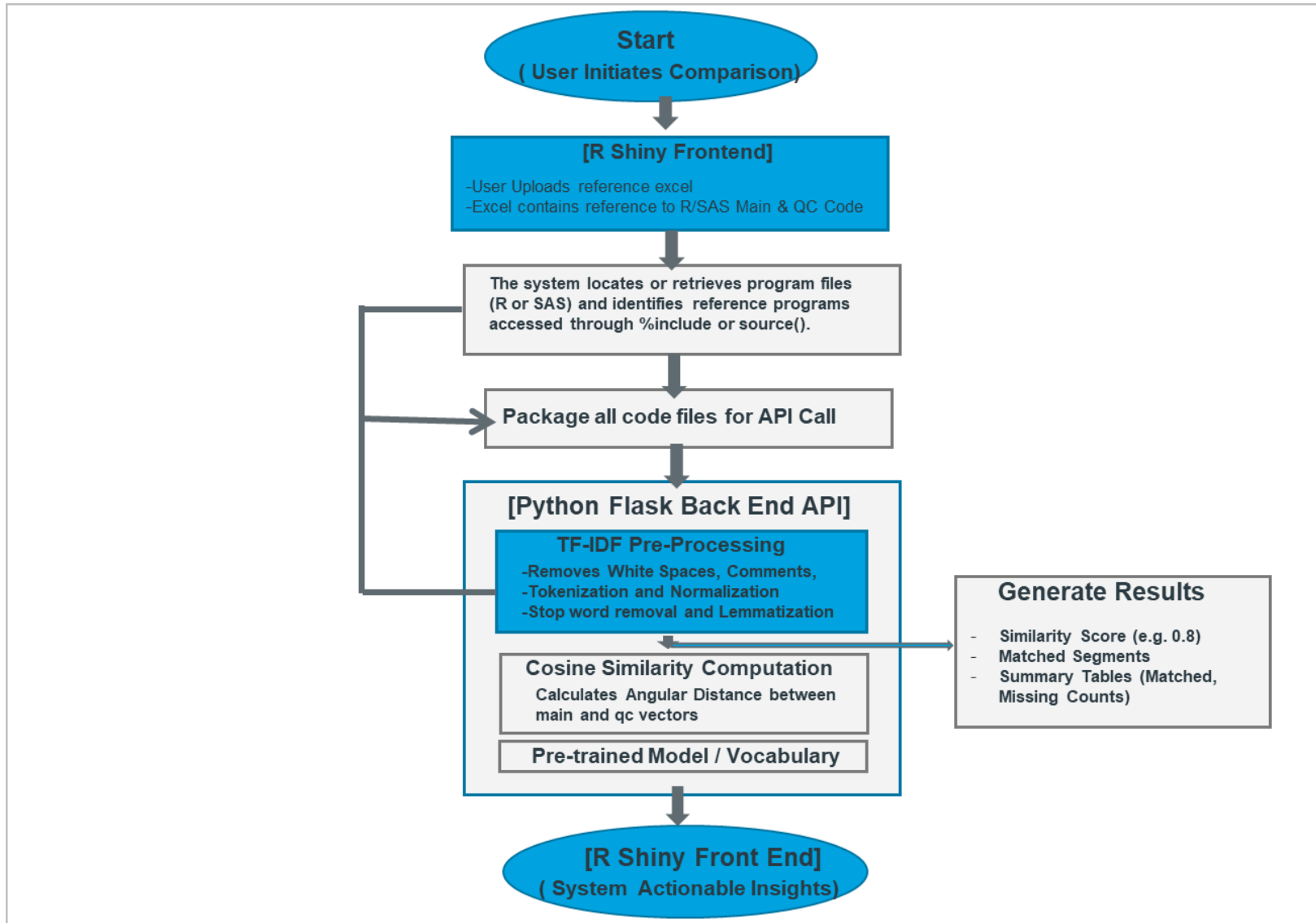


Cosine Similarity:

Higher Score(near 1.0) = Smaller Angle(Similar Logic)

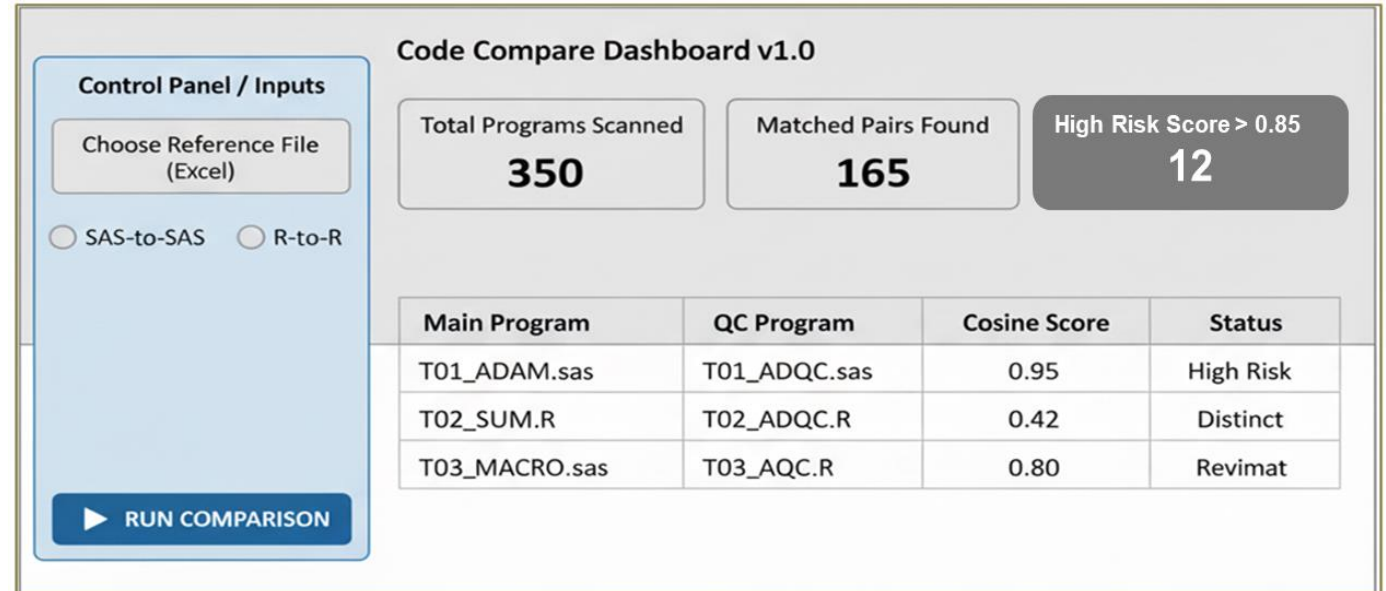
Lower Score(near 0.0) = Larger Angle(Dissimilar Logic)

The Automated Implementation Workflow



Key Results: Performance & Threshold

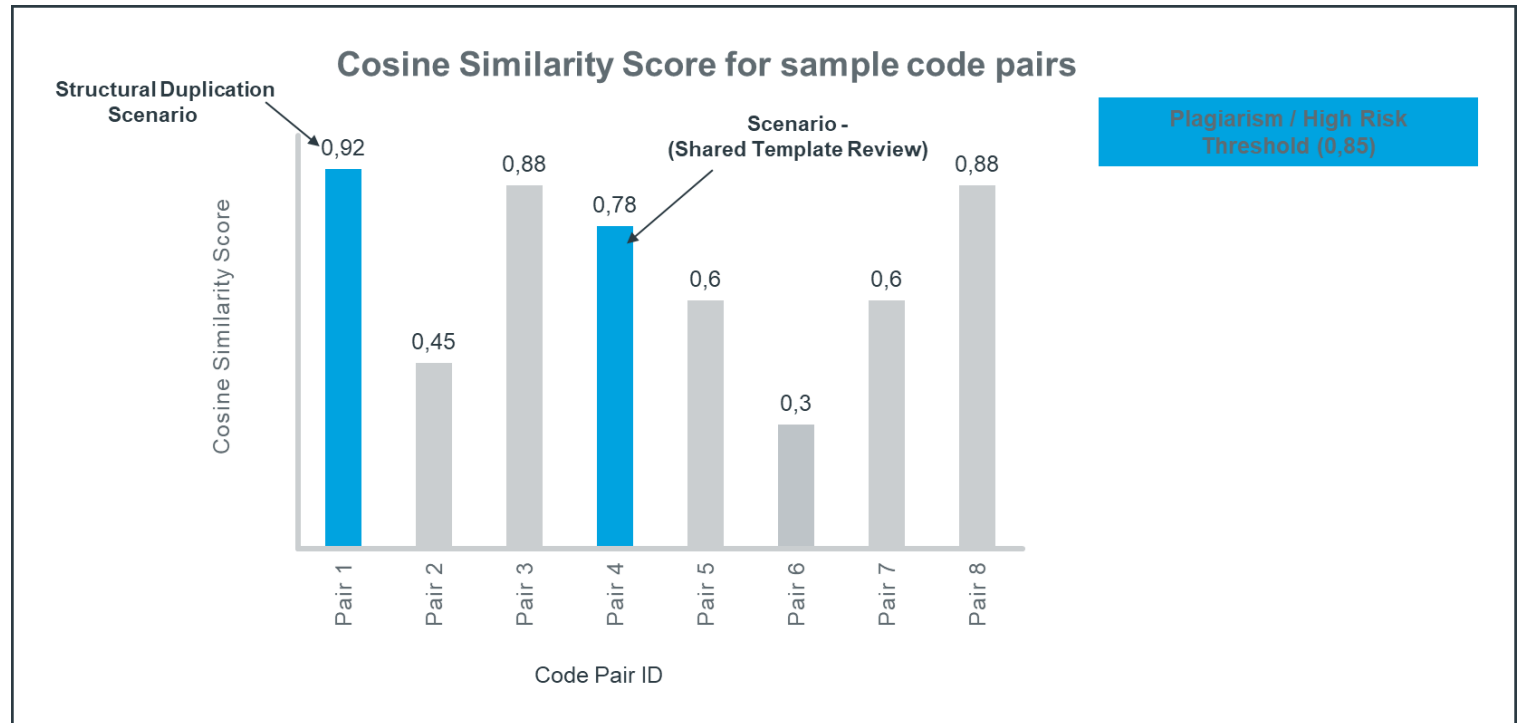
- **High Precision** demonstrated on curated SAS/R scripts.
- **Critical Threshold:** Cosine Similarity Scores **0.85** consistently correlate with high plagiarism risk.



Advantage: Faster processing and superior tolerance for changes compared to string-matching.

Case Studies: Putting Scores into Context

- **Internal Duplication Scenario (True Positive):**
 - » Score 0.92.
 - » Manual review confirmed structural duplication with variable renaming.
- **Template Usage Scenario (Audit Flag):**
 - » Score 0.78.
 - » Flagged shared use of a common macro/template, not plagiarism.
 - » Requires “Revimat” status review (Revimat: Review and Mandatory verification)



Core Takeaway: The tool is a flagging system that requires human expertise for final risk disposition.

Impact on the Clinical Workflow



Direct Validation Support

- The tool provides objective evidence of code originality between Main and QC programs, streamlining the required validation checks.



Enhanced Audit Readiness

- Every high-score match is flagged for a documented human review ("Revimat" status), creating a non-repudiable audit trail



Promotes Sponsor-CRO Quality

- Establishes objective, data-driven quality metrics for code, ensuring consistent programming standards are maintained across collaborating teams.



Focus on Logic, Not Styling

- By filtering out boilerplate, the system ensures that auditors focus their time only on programs with statistically significant logical similarity.

Critical Assessment: Strengths & Limitations

Strength

1. Semantic-level comparison.
2. Modular/scalable Python/R architecture.
3. Language-agnostic design (SAS/R).
4. User Friendly Interface (R-Shiny).

Weakness

1. No cross-language support (e.g., Python-to-R).
2. Potential for false positives on heavy boilerplate code.

Future Enhancements

1



Deep Learning Models

- Incorporate code embedding (transformers) for enhanced semantic depth.

2



Explainability

- Add features to highlight the exact matched logic blocks in the code.

3



Integration

- Enable automated triggering of comparisons via Version Control Systems (Git)
- Add batch processing capability for historical codebase analysis.

4



LLM Models

- Leveraging alternative algorithms (LLM Models)

Conclusion & Call to Action

Summary



- Code Compare provides a scalable, intelligent, and user-friendly solution.

Impact

- Empowers teams to uphold ethical coding standards and streamline audit readiness.



Brand and product names are trademarks of their respective companies.

Contact

Saroj Kumar Jena

Assoc. Director, Data Science

Saroj.jena@iqvia.com

+31 644 266 924