SmartOpt — Simple AI-Powered Code Optimizer

Overview

SmartOpt is a **prototype compiler optimization advisor** for **C, C++**, and **Rust** programs.
It analyzes user-submitted code, compiles it with different optimization flags (like -O0, -O1, -O2, -O3, and -Os), and identifies which flag performs best based on **runtime, compile time, and binary size**.

This is a **simple, working prototype** — not a research-level mathematical optimizer.
It focuses on **small code samples** and **basic loops, operations, or functions**, not on advanced workloads (like matrix multiplication, sin/cos, or heavy math libraries).

Current Capabilities

- Supports **C, C++, and Rust**
- Benchmarks compile time, runtime, and binary size
- Provides **best optimization flag**
- Uses a simple **ML model** (RandomForest/XGBoost) for prediction
- Automated deployment via **Jenkins + Cloud Run**

Limitations (for prototype phase)

- Designed for **simple code snippets** only
- No mathematical or high-complexity workloads (e.g., sin, cos, matrix math)
- Binary sizes may appear similar for small programs
- Focused on functionality, not precision modeling
- Optional **LLM explanation** (Gemma-2B) for plain-language insight(Working on this)

Future Enhancements

- Add support for **mathematical and scientific workloads**
- Improve ML model using **code-level feature extraction**
- Introduce **GNN or RL-based learning** for better predictions
- Add advanced benchmarking and metrics visualization
- Integrate more languages and improve frontend UX

Purpose

The goal of this version is to **demonstrate concept and workflow** —
how DevOps automation (Jenkins + GCP) and Machine Learning can work together in compiler optimization.
It's not about mathematical depth — it's about building a **working, cloud-based AI compiler advisor**.

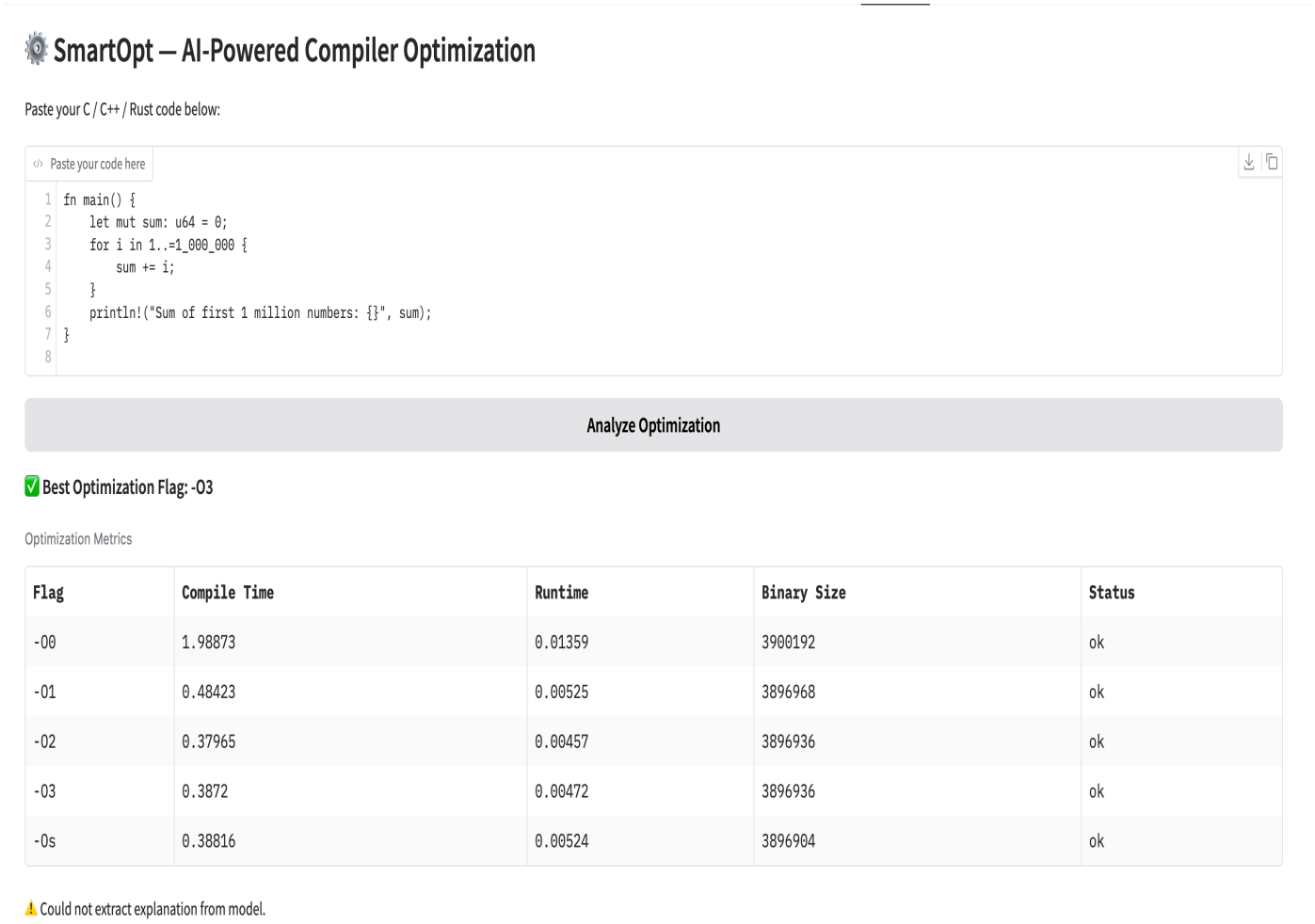
Sample Output :

Rust Outputs

Using Docker :

{"filename":"tmpqvui5_va.rs","language":".rs","best_flag":"-O1","flags":[{"flag":"-O0","compile_time":0.26019,"runtime":0.00061,"binary_size":3916608,"status":"ok"},{"flag":"-O1","compile_time":0.16952,"runtime":0.00058,"binary_size":3915952,"status":"ok"},{"flag":"-O2","compile_time":0.13158,"runtime":0.00059,"binary_size":3915952,"status":"ok"},{"flag":"-O3","compile_time":0.121,"runtime":0.00054,"binary_size":3915952,"status":"ok"},{"flag":"-Os","compile_time":0.12937,"runtime":0.00052,"binary_size":3915952,"status":"ok"}],"explanation":"⚠️ Could not extract explanation from model."}%

Using UI :

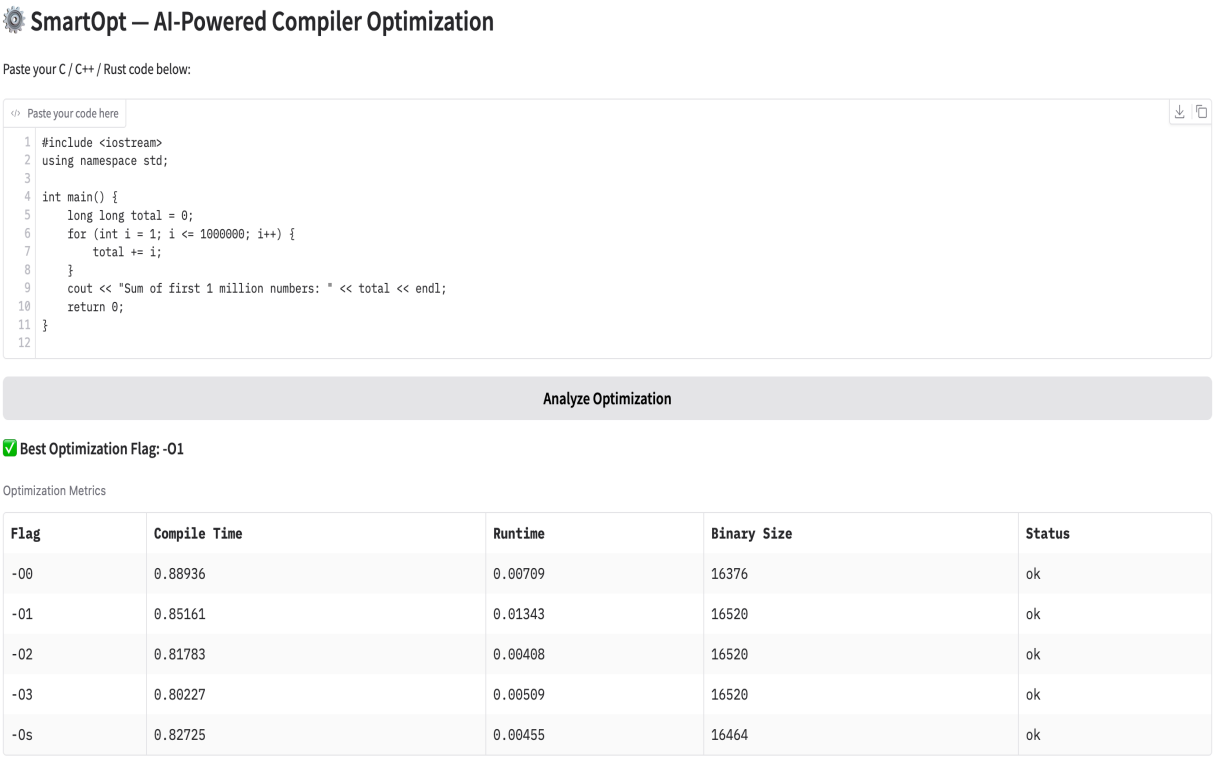


C++ Outputs

Using Docker :

"filename":"tmpk5guz6bg.cpp","language":".cpp","best_flag":"-O1","flags":[{"flag":"-O0","compile_time":0.22146,"runtime":0.00075,"binary_size":70600,"status":"ok"},{"flag":"-O1","compile_time":0.22051,"runtime":0.00073,"binary_size":70600,"status":"ok"},{"flag":"-O2","compile_time":0.20621,"runtime":0.0007,"binary_size":70600,"status":"ok"},{"flag":"-O3","compile_time":0.20739,"runtime":0.00069,"binary_size":70600,"status":"ok"},{"flag":"-Os","compile_time":0.20756,"runtime":0.00067,"binary_size":70600,"status":"ok"}],"explanation":"⚠️ Could not extract explanation from model."}%

Using UI:



C Outputs:

Using Docker:

{"filename":"tmpjb4xqts2.c","language":".c","best_flag":"-O1","flags":[{"flag":"-O0","compile_time":0.05148,"runtime":0.00041,"binary_size":70432,"status":"ok"},{"flag":"-O1","compile_time":0.04576,"runtime":0.00036,"binary_size":70432,"status":"ok"},{"flag":"-O2","compile_time":0.02968,"runtime":0.00036,"binary_size":70432,"status":"ok"},{"flag":"-O3","compile_time":0.03317,"runtime":0.00036,"binary_size":70432,"status":"ok"},{"flag":"-Os","compile_time":0.03471,"runtime":0.00045,"binary_size":70432,"status":"ok"}],"explanation":"⚠️ Could not extract explanation from model."}%

Using UI :