# Navi: High-Performance Machine Learning Serving Server in Rust

Navi is a high-performance, versatile machine learning serving server implemented in Rust and tailored for production usage. It's designed to efficiently serve within the Twitter tech stack, offering top-notch performance while focusing on core features.

## Key Features

- \*\*Minimalist Design Optimized for Production Use Cases\*\*: Navi delivers ultra-high performance, stability, and availability, engineered to handle real-world application demands with a streamlined codebase.

- \*\*gRPC API Compatibility with TensorFlow Serving\*\*: Seamless integration with existing TensorFlow Serving clients via its gRPC API, enabling easy integration, smooth deployment, and scaling in production environments.

- \*\*Plugin Architecture for Different Runtimes\*\*: Navi's pluggable architecture supports various machine learning runtimes, providing adaptability and extensibility for diverse use cases. Out-of-the-box support is available for TensorFlow and Onnx Runtime, with PyTorch in an experimental state.

## Current State

While Navi's features may not be as comprehensive as its open-source counterparts, its performance-first mindset makes it highly efficient.

- Navi for TensorFlow is currently the most feature-complete, supporting multiple input tensors of different types (float, int, string, etc.).

- Navi for Onnx primarily supports one input tensor of type string, used in Twitter's home recommendation with a proprietary BatchPredictRequest format.

- Navi for Pytorch is compilable and runnable but not yet production-ready in terms of performance and stability.

## Directory Structure

- `navi`: The main code repository for Navi

- `dr\_transform`: Twitter-specific converter that converts BatchPredictionRequest Thrift to ndarray

- `segdense`: Twitter-specific config to specify how to retrieve feature values from BatchPredictionRequest

- `thrift\_bpr\_adapter`: generated thrift code for BatchPredictionRequest

## Content

We have included all \*.rs source code files that make up the main Navi binaries for you to examine. However, we have not included the test and benchmark code, or various configuration files, due to data security concerns.

## Run

In navi/navi, you can run the following commands:

- `scripts/run\_tf2.sh` for [TensorFlow](https://www.tensorflow.org/)

- `scripts/run\_onnx.sh` for [Onnx](https://onnx.ai/)

Do note that you need to create a models directory and create some versions, preferably using epoch time, e.g., `1679693908377`.

so the models structure looks like:

models/

-web\_click

- 1809000

- 1809010

## Build

You can adapt the above scripts to build using Cargo.