Hi,

Just out of curiosity, I am researching on if we can use machine learning library in rust under no_std setting and SGX mode.

This time I tried using the following library.

GitHub

### AtheMathmo/rusty-machine

Machine Learning library for Rust. Contribute to AtheMathmo/rusty-machine development by creating an account on GitHub.

// Built-In Attributes

# ![no_std]

// Imports extern crate eng_wasm; extern crate eng_wasm_derive; extern crate serde; extern crate statistical; extern crate rusty_machine;

# [macro_use]

extern crate std;

use eng_wasm::*; use eng_wasm_derive::pub_interface; use serde::{Serialize, Deserialize}; use std::vec;

use rusty_machine::linalg::Matrix; use rusty_machine::linalg::Vector; use rusty_machine::learning::gp::GaussianProcess; use rusty_machine::learning::gp::ConstMean; use rusty_machine::learning::toolkit::kernel; use rusty_machine::learning::SupModel;

// Encrypted state keys static MILLIONAIRES: &str = "millionaires";

// Structs

# [derive(Serialize, Deserialize)]

pub struct Millionaire { address: H160, net_worth: U256, }

// Public struct Contract which will consist of private and public-facing secret contract functions pub struct Contract;

// Private functions accessible only by the secret contract impl Contract { fn get_millionaires() -> Vec { read_state! (MILLIONAIRES).unwrap_or_default() } }

// Public trait defining public-facing secret contract functions

# [pub_interface]

pub trait ContractInterface{ fn add_millionaire(address: H160, net_worth: U256); fn compute_richest() -> H160; } // Implementation of the public-facing secret contract functions defined in the ContractInterface // trait implementation for the Contract struct above impl ContractInterface for Contract { #[no_mangle] fn add_millionaire(address: H160, net_worth: U256) { let mut millionaires = Self::get_millionaires(); millionaires.push(Millionaire { address, net_worth, }); write_state! (MILLIONAIRES => millionaires); }

```
#[no_mangle]
fn compute_richest() -> H160 {
    let v = vec![1.1,2.1,3.1];
    match Self::get_millionaires().iter().max_by_key(|m| m.net_worth) {
        Some(millionaire) => {
            let _tst = statistical::mean(v.as_slice());
            // First we'll get some data.

            // Some example training data.
            let inputs = Matrix::new(3,3,vec![1.,1.,1.,2.,2.,2.,3.,3.,3.]);
            let targets = Vector::new(vec![0.,1.,0.]);

            // Some example test data.
            let test_inputs = Matrix::new(2,3, vec![1.5,1.5,1.5,2.5,2.5,2.5]);

            // Now we'll set up our model.
```

```
        // This is close to the most complicated a model in rusty-machine gets!

        // A squared exponential kernel with lengthscale 2, and amplitude 1.
        let ker = kernel::SquaredExp::new(2., 1.);

        // The zero function
        let zero_mean = ConstMean::default();

        // Construct a GP with the specified kernel, mean, and a noise of 0.5.
        let mut gp = GaussianProcess::new(ker, zero_mean, 0.5);


        // Now we can train and predict from the model.

        // Train the model!
        gp.train(&inputs, &targets).unwrap();

        // Predict output from test datae]
        let _outputs = gp.predict(&test_inputs).unwrap();
        millionaire.address
        //millionaire.address
      },
      None => H160::zero(),
    }
}

}
```

I just added some simple code into the tutorial without exporting the result to the enigma state.

Fortunately, it passed compile, migration, and test code and seems the secret contract can handle machine learning library!!!

I am really feeling Secret Contract will expand the horizon of blockchain world!!!