











Rust 무작정 배워보기

1년에 하나 새로운 언어나 플랫폼을 배우자

아젠다

-  [과거 학습들](#)
-  [Rust 학습 요약](#)
-  [Rust 특징](#)
-  [결론](#)

게팅 스타티드

-  [Actix](#)
-  [Yew](#)
-  [Tokio](#)
-  [Linha](#)



과거 학습들

개발자로 살아온 10년 역지로 끼워 맞춘 년도 별 학습 현황

1. 2010년 - 안드로이드
2. 2011년 - 자체 서버 프레임워크 (자바 생태계)
3. 2012년 - 자바스크립트
4. 2013년 - 스프링 프레임워크
5. 2014년 - AngularJS 웹 프레임워크
6. 2015년 - Bash shell
7. 2016년 - TypeScript (Angular, Vue)
8. 2017년 - Go lang
9. 2018년 - Scala
10. 2019년 - C# (유니티)
11. 2020년 - Python, React 웹 프레임워크
12. 2021년 - Rust (올해가 가기전에...)



Rust 학습 요약

퇴근 후 1시간

- 관련 내용 검색 (1시간)
- YouTube로 학습 (3시간)
 - (2019년 이후 자료들)
<https://youtu.be/W9DO6m8JSSs> (mithradates)
 - <https://youtu.be/cUywpnH8o48> (팀 주피터)
- Rust 빠르게 읽어보기 (1시간)
 - <https://doc.rust-lang.org/book/> (러스트 공식 도서)
 - <http://www.yes24.com/Product/Goods/83075894> (한글 책 - 2019년 출판)
- 실습 (4시간)
 - Actix, Yew, Tokio, Linfa
- 발표 준비 (2시간)



Rust 특징

메모리 관리 (소유권과 수명)

- 메모리는 단 하나의 소유권을 가짐
- 필요하면 빌려 쓸 수 있음
- 코드 블록("{ code block}")이 끝나면 메모리도 소멸됨

높은 안정성, 고성능

- 강력한 컴파일로 안정성 보장 (Lint 수준의 문법 체크)
- C, C++ 수준의 고성능의 컴파일

개발 편의

- Visual Studio Code 로 개발하기 편함
- 프로젝트 생성, 빌드, 테스트 등 쉽게 개발 환경 구축 가능
- 최신 트렌드 문법을 마음껏 활용 할 수 있음

레퍼런스

- Libra (Meta)
- npm
- CloudFlare
- AWS Lambda
- FileSystem (Dropbox)

Actix

웹 서버 (Back-end)

A powerful, pragmatic, and extremely fast web framework for Rust

```
use actix_web::{get, post, web, App, HttpResponse, HttpServer, Responder};

#[get("/")]
async fn hello() -> impl Responder {
    HttpResponse::Ok().body("Hello world!")
}

#[post("/echo")]
async fn echo(req_body: String) -> impl Responder {
    HttpResponse::Ok().body(req_body)
}

async fn manual_hello() -> impl Responder {
    HttpResponse::Ok().body("Hey there!")
}

#[actix_web::main]
async fn main() -> std::io::Result<()> {
    HttpServer::new(|| {
        App::new()
            .service(hello)
            .service(echo)
            .route("/hey", web::get().to(manual_hello))
    })
    .bind("127.0.0.1:8080")?
    .run()
    .await
}
```

Yew

웹 어셈블리 (Front-end)

Yew is a modern Rust framework for creating multi-threaded front-end web apps.

```
use yew::prelude::*;

enum Msg {
    AddOne,
}

struct Model {
    link: ComponentLink<Self>,
    value: i64,
}

impl Component for Model {
    type Message = Msg;
    type Properties = ();

    fn create(_props: Self::Properties, link: ComponentLink<Self>) -> Self {
        Self {
            link,
            value: 0,
        }
    }

    fn update(&mut self, msg: Self::Message) -> ShouldRender {
        match msg {
            Msg::AddOne => {
                self.value += 1;
                true
            }
        }
    }
}
```

```

    }
}

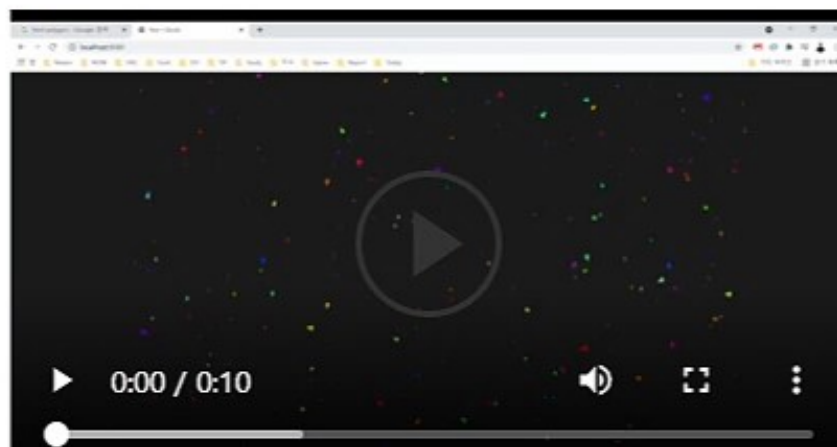
fn change(&mut self, _props: Self::Properties) -> ShouldRender {
    false
}

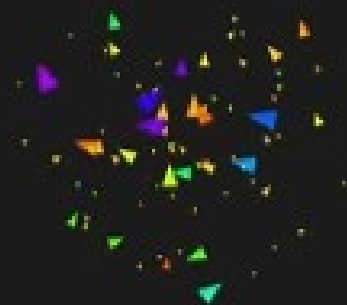
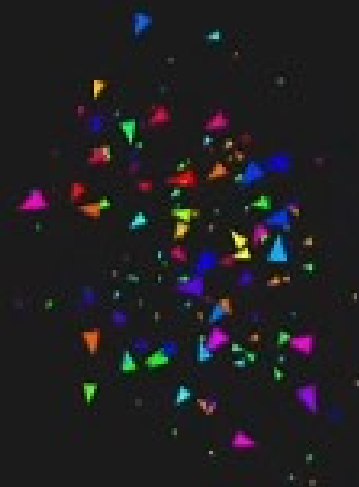
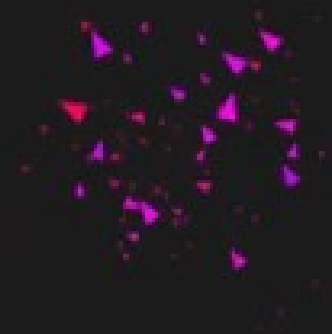
fn view(&self) -> Html {
    html! {
        <div>
            <button onclick=self.link.callback(|_| Msg::AddOne)>{ "+1" }</button>
            <p>{ self.value }</p>
        </div>
    }
}

fn main() {
    yew::start_app::<Model>();
}

```

Boids 샘플





Tokio

Network Application (Library)

```
use mini_redis::{Connection, Frame};
use tokio::net::{TcpListener, TcpStream};

#[tokio::main]
async fn main() {
    // Bind the listener to the address
    let listener = TcpListener::bind("127.0.0.1:6379").await.unwrap();

    loop {
        // The second item contains the ip and port of the new connection.
        let (socket, _) = listener.accept().await.unwrap();

        // A new task is spawned for each inbound socket. The socket is
        // moved to the new task and processed there.
        tokio::spawn(async move {
            process(socket).await;
        });
    }
}

async fn process(socket: TcpStream) {
    use mini_redis::Command::{self, Get, Set};
    use std::collections::HashMap;

    // A hashmap is used to store data
    let mut db = HashMap::new();

    // Connection, provided by `mini-redis`, handles parsing frames from
    // the socket
```

```

let mut connection = Connection::new(socket);

// Use `read_frame` to receive a command from the connection.
while let Some(frame) = connection.read_frame().await.unwrap() {
    let response = match Command::from_frame(frame).unwrap() {
        Set(cmd) => {
            // The value is stored as `Vec<u8>`
            db.insert(cmd.key().to_string(), cmd.value().to_vec());
            Frame::Simple("OK".to_string())
        }
        Get(cmd) => {
            if let Some(value) = db.get(cmd.key()) {
                // `Frame::Bulk` expects data to be of type `Bytes`. This
                // type will be covered later in the tutorial. For now,
                // `&Vec<u8>` is converted to `Bytes` using `into()`.
                Frame::Bulk(value.clone().into())
            } else {
                Frame::Null
            }
        }
    };
    cmd => panic!("unimplemented {:?}", cmd),
};

// Write the response to the client
connection.write_frame(&response).await.unwrap();
}
}

```



RunRust > tokio > examples > @ chat.rs

```

12  /// You can test this out by running:
13  ///
14  /// cargo run --example chat
15  ///
16  /// And then in another terminal run:
17  ///

```

문재 출력 터미널 디버그 콘솔

| | | | | | | | | |
|--|--|--|--|--|--|---|---|--|
| <pre> use r4: hi~ user2: good morning user5: nice to meet you user4: hello guest1 has left the chat lou has left the chat user5: a user5: sadsa user9 has joined the chat user10 has joined the chat user10: user10: hello~ </pre> | <pre> : 1 user7: user4: a 4: hi~ good morning user5: nice to meet you user4: hello guest1 has left the chat lou has left the chat user5: a user5: sadsa user9 has joined the chat user10 has joined the chat user10: user10: hello~ </pre> | <pre> user7: q user7: 1 user4: hi~ ser2: good morning : nice to meet you user4: hello guest1 has left the chat lou has left the chat user5: a user5: sadsa user9 has joined the chat user10 has joined the chat 0: user10 user10: h ello~ </pre> | <pre> user7: c user7: q 1 user7: 1a user2: good morning user5: nice to meet you hello guest1 has left the chat lou has left the chat user5: a user5: sadsa user9 has joined the chat user10 has joined the chat 0: user10 user10: h ello~ </pre> | <pre> : c user7: q user7: 1 user4: hi~ ser2: good morning nice to meet you user4: hello guest1 has left the chat lou has left the chat sadsa user9 has joined the chat user10 has joined the chat 0: user10 user10: h ello~ </pre> | <pre> Please enter your username: user9 user10 has joined the chat user10: us er10 user10: hello~ </pre> | <pre> Please enter your username: user10 user10 hello~ </pre> | <pre> : b user7: c user7: q 7: user7: 1 user4: a use r4: hi~ user2: good morning user5: nice to meet you user4: hello guest1 has left the chat lou has left the chat user5: a user5: sadsa user9 has joined the chat user10 has joined the chat user10: user10: hello~ </pre> | <pre> ----- ----- ----- user7: a user7: b r7: c user7: q user7: 1 u ser7: 1 user4: a user4: hi~ user2: good morning user5: nice to meet you user4: hello guest1 has left the chat lou has left the chat a user5: sadsa user9 has joined the chat user10 has joined the chat er10 has joined the chat er10 user10: hello </pre> |
|--|--|--|--|--|--|---|---|--|

Linfa

머신러닝 라이브러리

SVM 예제

서포트 벡터 머신(support vector machine, SVM.)은 기계 학습의 분야 중 하나로 패턴 인식, 자료 분석을 위한 지도 학습 모델이며, 주로 분류와 회귀 분석을 위해 사용한다

```
use linfa::composing::MultiClassModel;
use linfa::prelude::*;
use linfa_svm::{error::Result, Svm};

fn main() -> Result<()> {
    let (train, valid) = linfa_datasets::winequality()
        .split_with_ratio(0.9);

    println!(
        "Fit SVM classifier with #{ } training points",
        train.nsamples()
    );

    let params = Svm::<_, Pr>::params()
        // .pos_neg_weights(5000., 500.)
        .gaussian_kernel(30.0);

    let model = train
        .one_vs_all()?
        .into_iter()
        .map(|(l, x)| (l, params.fit(&x).unwrap()))
        .collect::<MultiClassModel<_, _>>();

    let pred = model.predict(&valid);
```

```

// create a confusion matrix
let cm = pred.confusion_matrix(&valid)?;

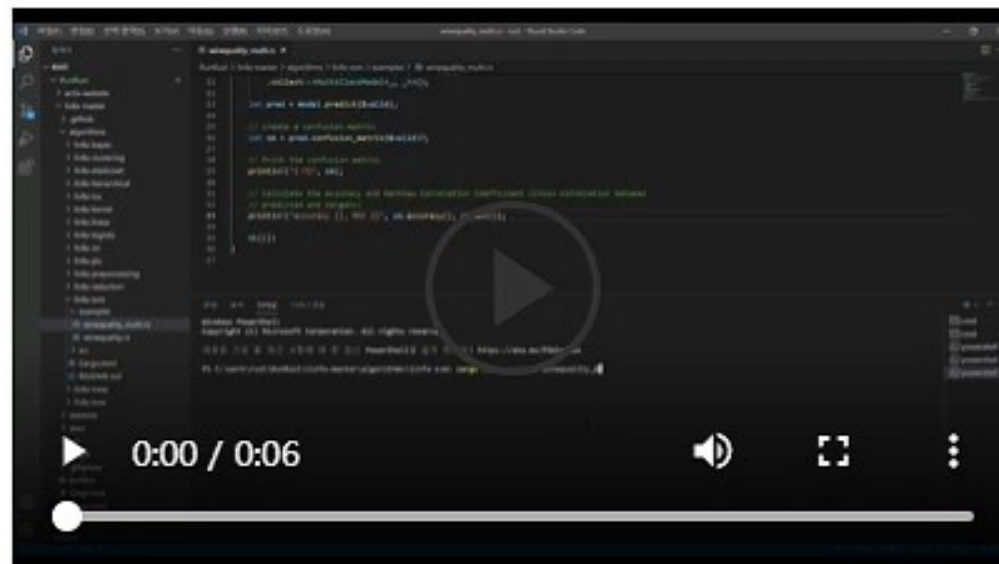
// Print the confusion matrix
println!("{:?}", cm);

// Calculate the accuracy and Matthew Correlation Coefficient (cross-correlation between
// predicted and targets)
println!("accuracy {}, MCC {}", cm.accuracy(), cm.mcc());

Ok(())
}

```

1440개 트레이닝 포인트, 러닝타임 1분, CPU, GPU 각각 사용률 15% 정도
(i7-10875H, 32GM, Win11 Home, RTX 3080 Laptop)



- 탐색기
- ▼ RUST
 - ▼ RunRust
 - > actix-web
 - ▼ linfa-master
 - > .github
 - ▼ algorithms
 - > linfa-bayes
 - > linfa-clustering
 - > linfa-elasticnet
 - > linfa-hierarchical
 - > linfa-ica
 - > linfa-kernel
 - > linfa-linear
 - > linfa-logistic
 - > linfa-nn
 - > linfa-pls
 - > linfa-preprocessing
 - > linfa-reduction
 - ▼ linfa-svm
 - ▼ examples
 - ⊗ winequality_multirs
 - ⊗ winequality.rs
 - > src
 - ⊗ Cargo.toml
 - ① README.md
 - > linfa-trees
 - > linfa-tsne
 - > datasets
 - > docs
 - > src
 - > target
 - ⊗ .gitignore
 - ⊗ build.rs
 - ⊗ Cargo.lock
 - ⊗ Cargo.toml
 - ⊗ CHANGELOG.md
 - > 개요
 - > 다임러

⊗ winequality_multirs X

RunRust > linfa-master > algorithms > linfa-svm > examples > ⊗ winequality_multirs

```

21         .collect::<MultiClassModel<_, _>>();
22
23     let pred = model.predict(&valid);
24
25     // create a confusion matrix
26     let cm = pred.confusion_matrix(&valid)?;
27
28     // Print the confusion matrix
29     println!("{:?}", cm);
30
31     // Calculate the accuracy and Matthew Correlation Coefficient (cross-correlation between
32     // predicted and targets)
33     println!("accuracy {}, MCC {}", cm.accuracy(), cm.mcc());
34
35     Ok(())
36 }
37

```

문제 출력 터미널 디버그 콘솔

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

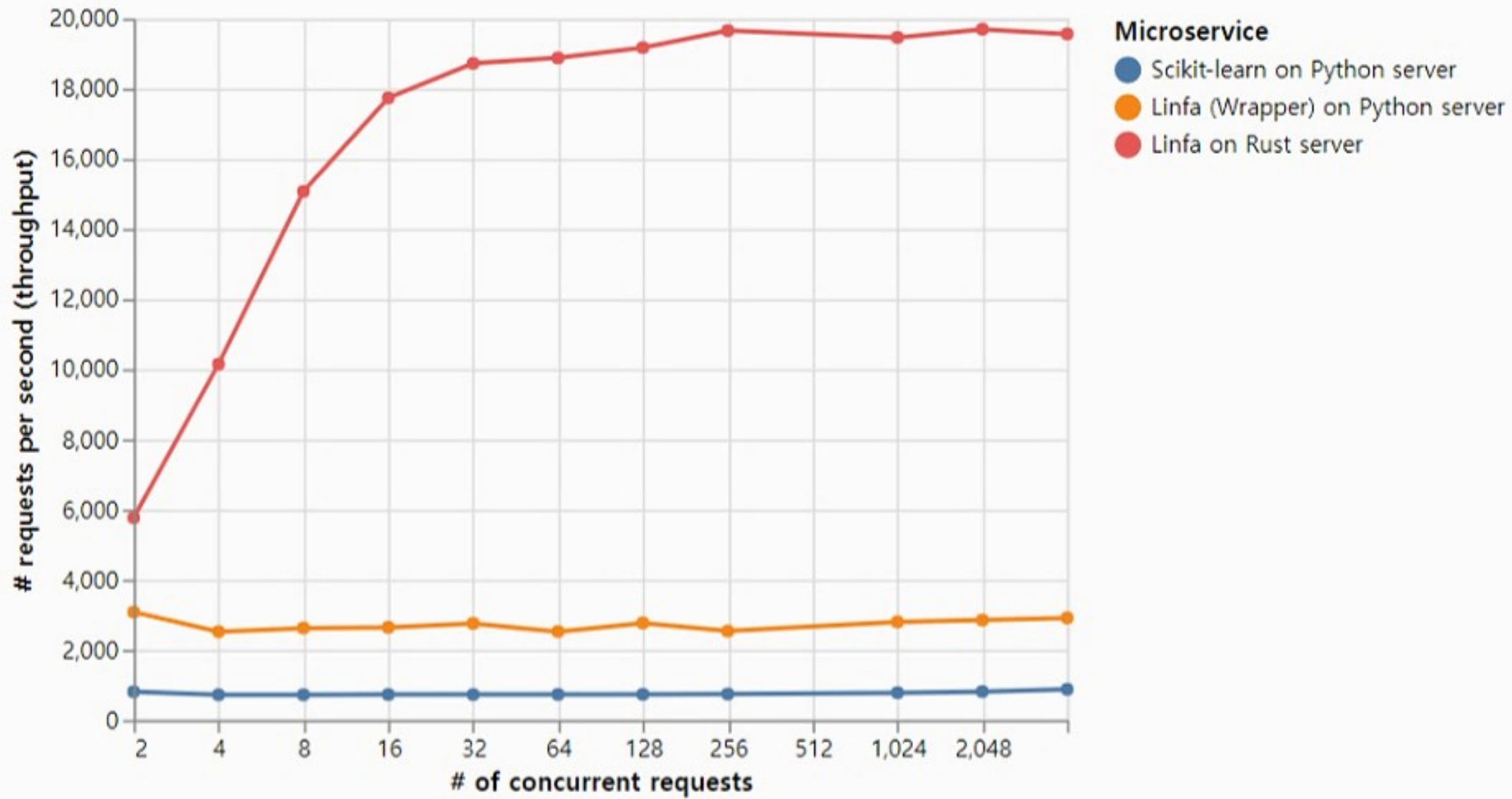
새로운 기능 및 개선 사항에 더 한 최신 PowerShell을 설치 하세요! <https://aka.ms/PSWindows>

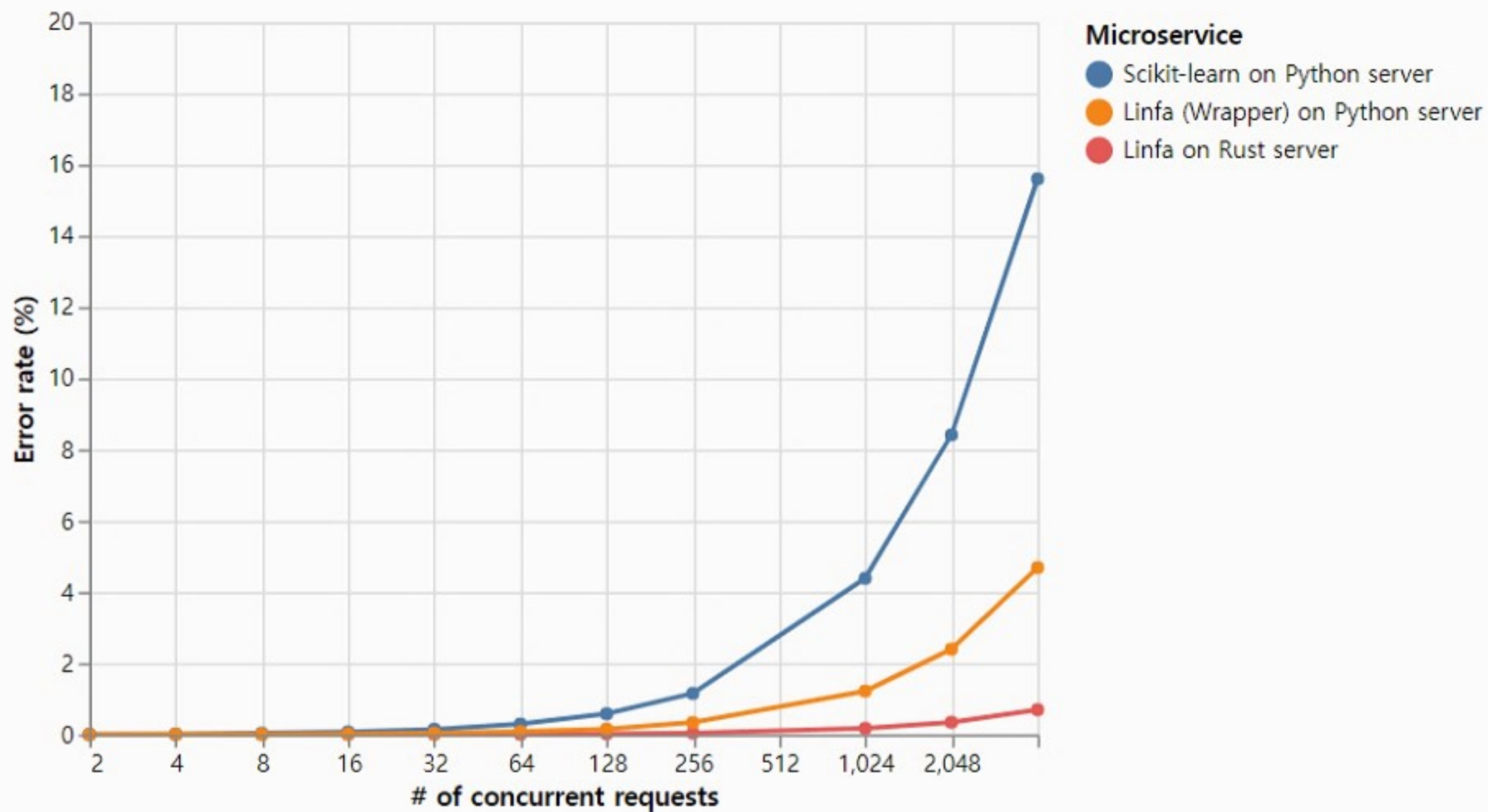
PS C:\work\rust\RunRust\linfa-master\algorithms\linfa-svm> cargo run --example winequality_multi
 Finished dev [unoptimized + debuginfo] target(s) in 0.12s
 Running 'C:\work\rust\RunRust\linfa-master\target\debug\examples\winequality_multi.exe'
 Fit SVM classifier with #1448 training points

| classes | 7 | 6 | 4 | 3 | 8 | 5 |
|---------|---|----|---|---|---|----|
| 7 | 2 | 11 | 0 | 0 | 0 | 2 |
| 6 | 5 | 43 | 5 | 0 | 1 | 14 |
| 4 | 0 | 1 | 0 | 0 | 0 | 5 |
| 3 | 0 | 0 | 0 | 0 | 0 | 3 |
| 8 | 0 | 2 | 0 | 0 | 0 | 0 |
| 5 | 2 | 33 | 0 | 0 | 0 | 30 |

accuracy 0.4716981, MCC 0.14869441

PS C:\work\rust\RunRust\linfa-master\algorithms\linfa-svm> █





<https://www.lpalmieri.com/posts/2019-12-01-taking-ml-to-production-with-rust-a-25x-speedup/>



결론

개발자들이 가장 좋아하는 언어 5년 연속 1위!

정말 써도 될까?

- 패키지 관리 도구 (Cargo) 가 조금 더 개선되었으면
- 메모리 관리가 더 어려워진 것 같음
- 패키지가 많지 않음
- 2019년에 유행하고 그 뒤로 조용함

rust Public

Empowering everyone to build reliable and efficient software.

 Rust  60,732  8,521  5,000+  468 Updated 5 minutes ago



Jan 9, 2021 – Nov 24, 2021

Contributions: Commits ▾

Contributions to master, excluding merge commits and bot accounts



게더타운을 Rust 로 만들어보면 재미있을 것 같다!

