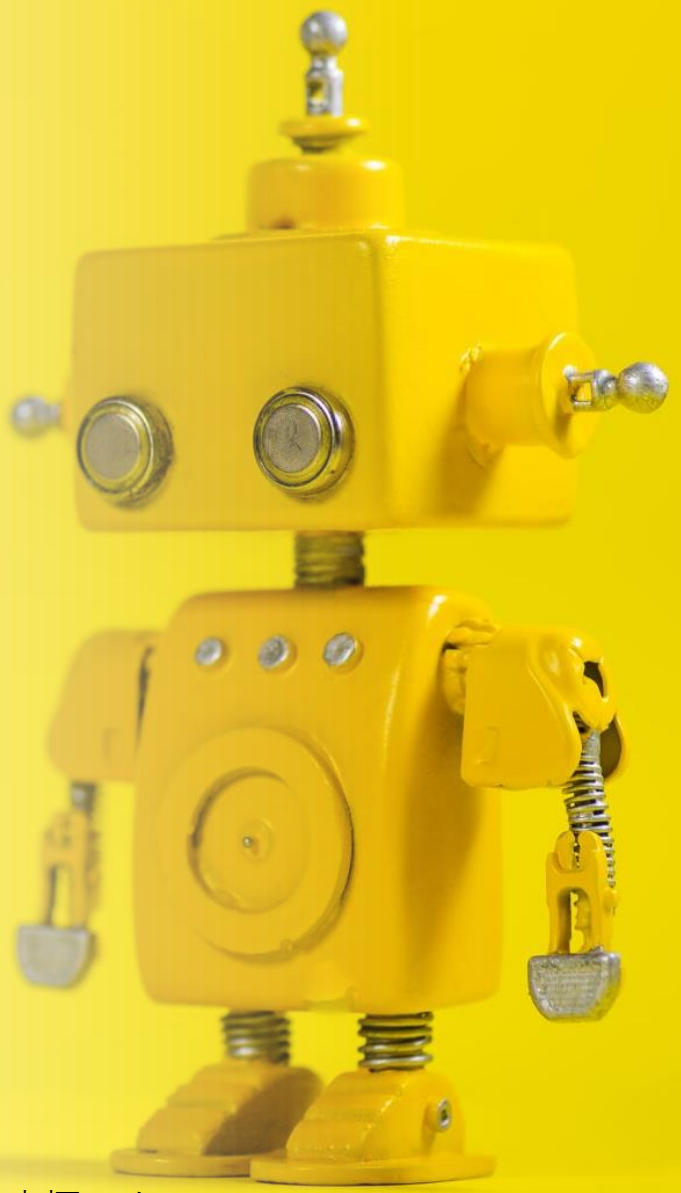
自作AIを
システムに
組み込んだ時の
苦労話

テクノロジーイノベーション部
橘高　新三郎



目次

フリーアドレスで出た課題

- 2018年1月にみなとみらいに本社移転

- フリーアドレスを導入
  ⇒コミュニケーションは活発になったが
  人を探すのに時間がかかるように

在席表示システムをご存知ですか？

- 自分のアイコンを移動させて現在地を周知
- 2018年8月に公開
- フリーアドレスの課題を解決

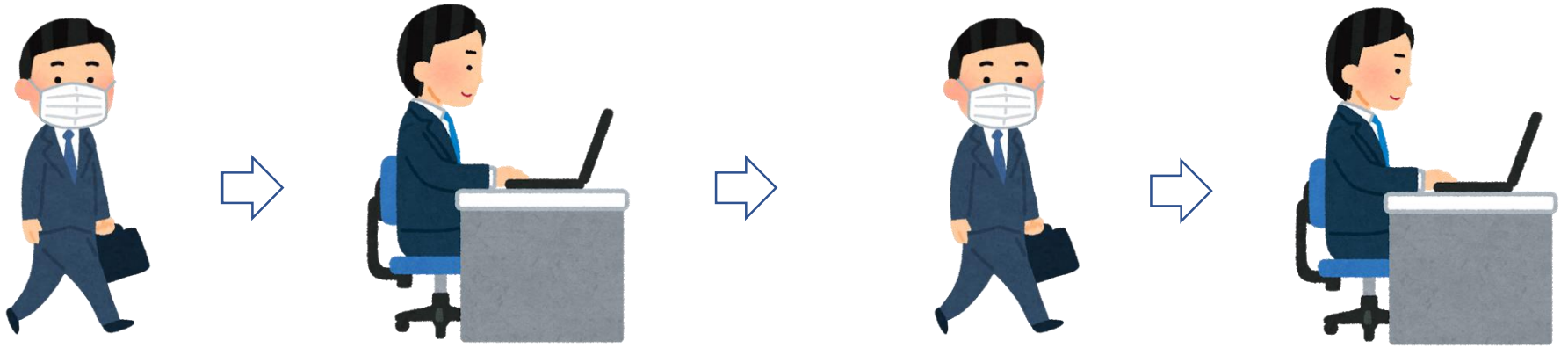
在席表示システムをご存知ですか？

- 自分のアイコンを移動させて現在地を周知
- 2018年8
- フリーアドレスの課題

流行らない

得をしない「面倒」は続かない

- 単純だが、毎日するとなると面倒



出社　システムに登録　移動　システムに登録

- このシステムで楽ができるのは自分以外の人

アイコンの移動を自動でしたい

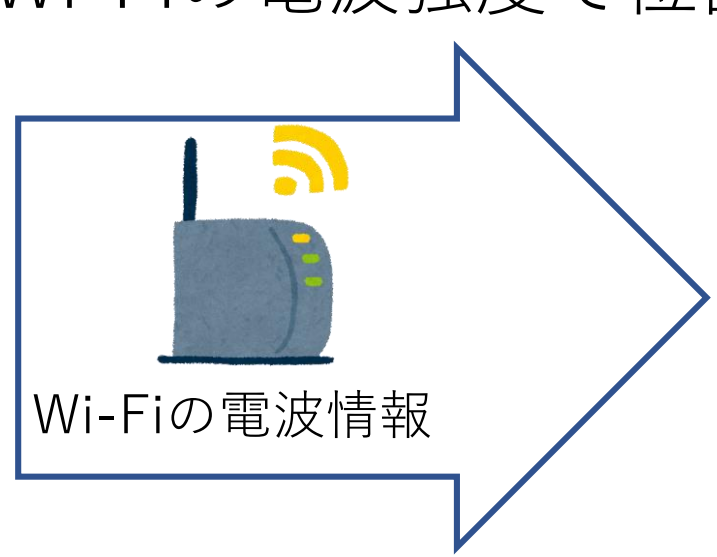
- 現在地を自動でシステムに反映させる

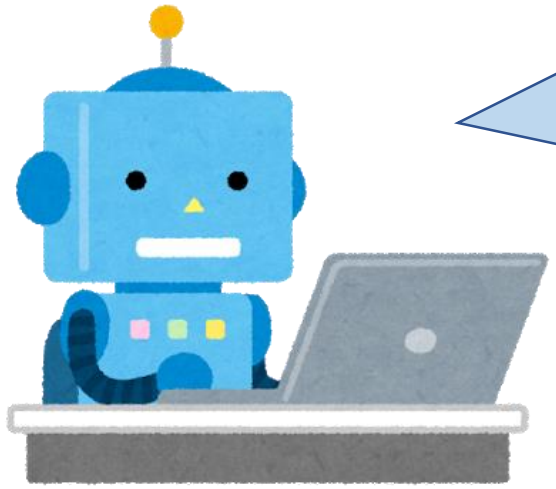
出社　システムに登録　移動　システムに登録

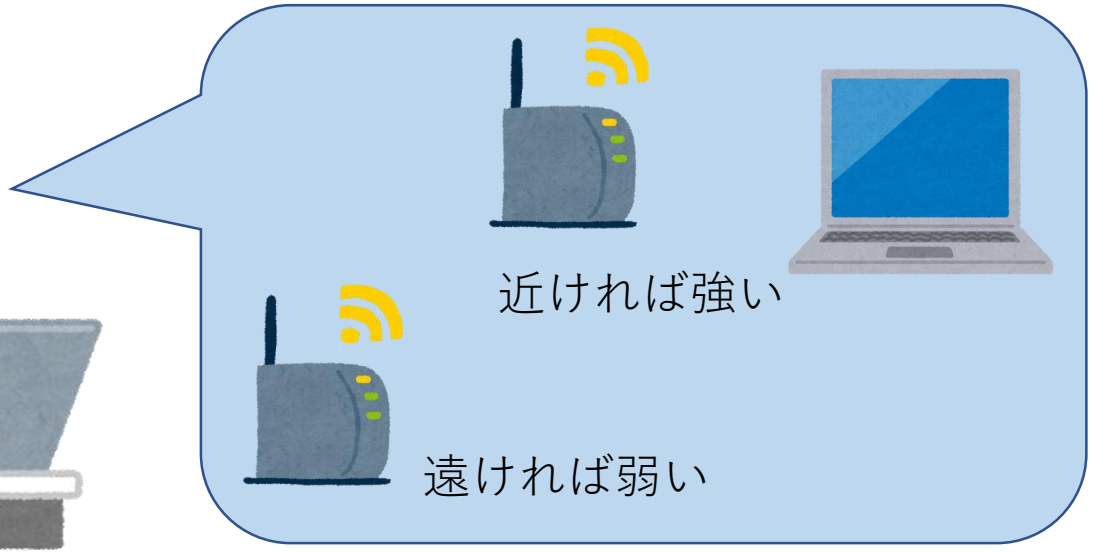
AI　開発編

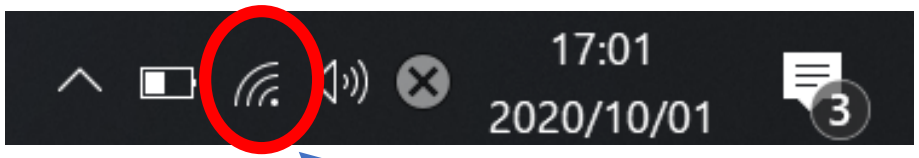
位置ってどうやったらわかるの？

- Wi-Fiの電波強度で位置を推定する







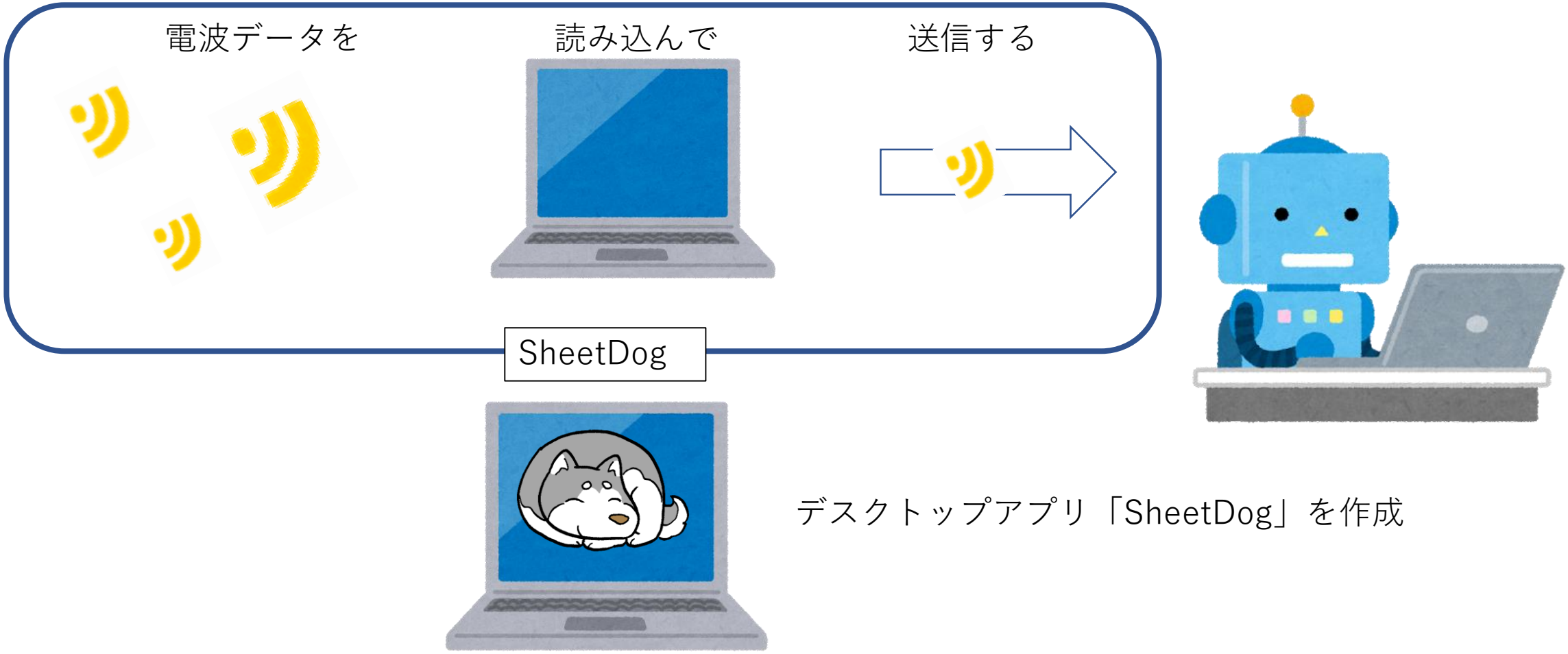


しかも
無料

アイコンで確認できるので、プログラムでも取得できるはず

デスクトップアプリを作らないと！

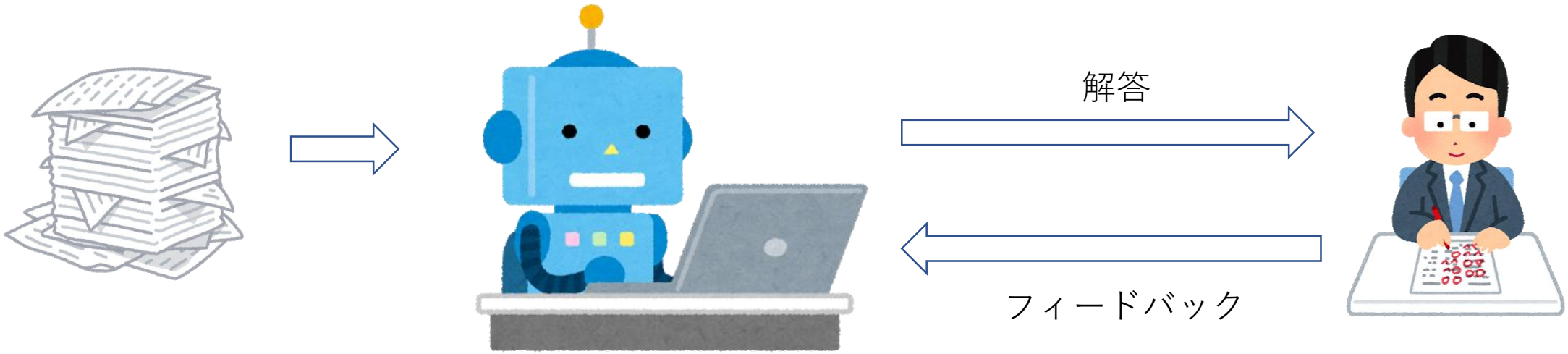
- それぞれのパソコンが受信している電波データが必要



デスクトップアプリ「SheetDog」を作成

位置推測の方法はAIで（AIの仕組み）

・事前に問題をいくつも解かせる
・問題が解けるようになるまで答え合わせをして
　フィードバックを繰り返す
・似たような問題も解けるようになる



問題と解答の作成

- 問題が解けるようになるには、たくさんの問題と解答が必要

電波データ

社員の位置

- 「電波データ」を送るアプリと「社員の位置」を位置する表示システムを連携させてデータを集める

問題と解答の作成

・データ収集のためにひたすら歩き回る

1データ

位置座標　X座標　Y座標

例）['1021','104']

＋

電波データ

例）['82','83'…,'0']

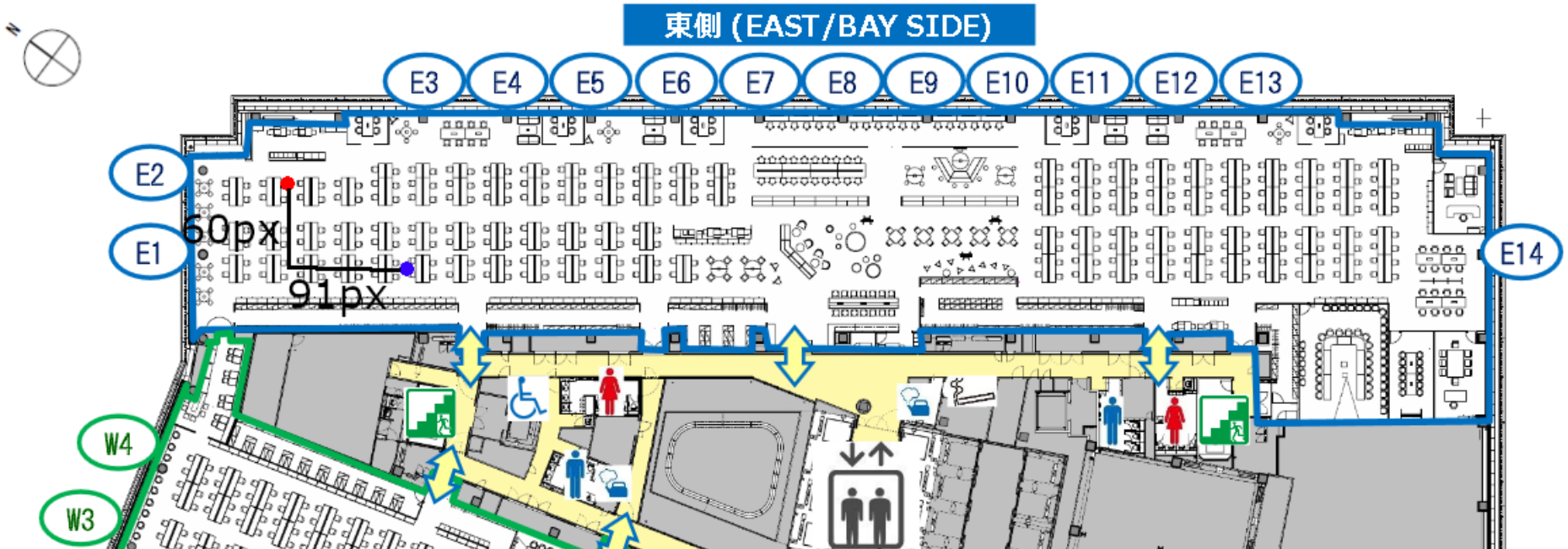
電波の強さ（%）

総データ数：330個

最初のAI

- ３３０個のデータをもとにAIを作成
  （使用ライブラリ：Tensorflow）

  学習に使っていないデータを入れると全体の平均で80px(15m)ずれる



データを見てみると

- 同じ場所で取得した電波データでも全然違う

データA['00','89','40','50',…'00', '00’]
データB['00','89','40','50’,…'00','00’]
データC['00','00','40','00',… '00','00’]

縦列は同じアクセスポイントからの電波強度

さっきまで強い電波を取得していたのに、
急に取れなくなってる・・・

精度を上げるために

・単純にデータ不足？　データを増やしてみた
　　⇒平均のずれはあまり変わらず

・ずれが大きい場所と小さい場所がある。精度が悪い場所のデータを増やしてみた
　　⇒ずれの変化なし

・電波の強いアクセスポイントのデータだけを利用する
　　⇒60px(11m)のずれに減少

・座標を算出するのではなくマップをクラス分けして分類問題にする
　　⇒85％の正解率で分類が成功

マップの分類

・座標を算出するのではなく空間をクラス分けして分類問題にする
⇒85％の正解率で分類が成功

	座標を算出	エリアに分類
平均のずれ	15m	9m※

※分類されたエリアの中心に
算出されたとして計算

開発で大変だったこと（やったこと）

- AIの仕組みを考案
  ⇒Wi-Fiの電波強度から位置を推測する
- 最初のデータの収集
  ⇒社内を歩き回って変な目で見られても集め続ける
- 精度を上げる方法の模索
  ⇒いろんな原因を考えてとにかく試し続ける

開発で大変だったこと（残っている問題）

- 電波がたまに取得できなくなってずれが大きくなる
  ⇒何度も電波を取得して、各要素の高い数値を採用する

AI　運用編

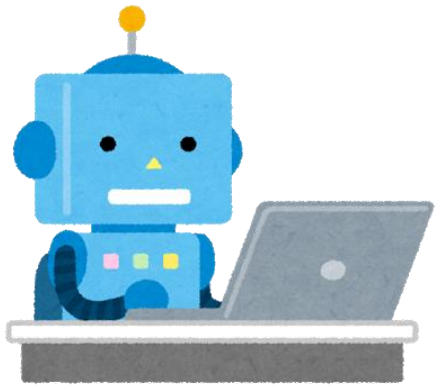
これまで出てきたもの

- WEBシステム「在席表示システム」（アイコンで位置の表示）
- デスクトップアプリ「SheetDog」
  （電波データの取得、送信、在席表示システムの表示）
- 位置推定AI（電波データから位置の推定）

それぞれを明記

SheetDog

位置推定AI



在席表示システム

システム構成図

SheetDog

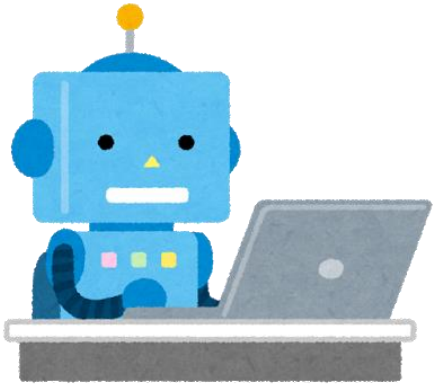
電波データ

在席表示システム

電波データ

座標

位置推定AI



アイコンの位置更新

ユーザーDB

運用でやりたいこと

- 使ってるうちに賢くしたい
  ⇒データをユーザーから集める
    ⇒集めたデータを使って精度を上げる

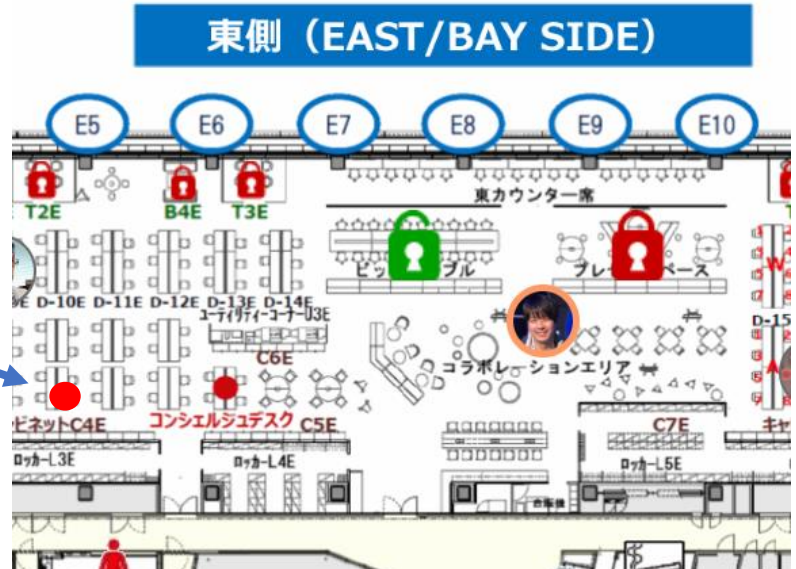
- AIライブラリを変えたい
  ⇒精度を上げるためにAIライブラリを変えられるように

使ってるうちに賢くしたい

- SheetDog（デスクトップアプリ）でユーザーからデータを収集
  ①アイコンが正しい位置になければ、ユーザーが自身の位置に
  　アイコンを手動で動かす

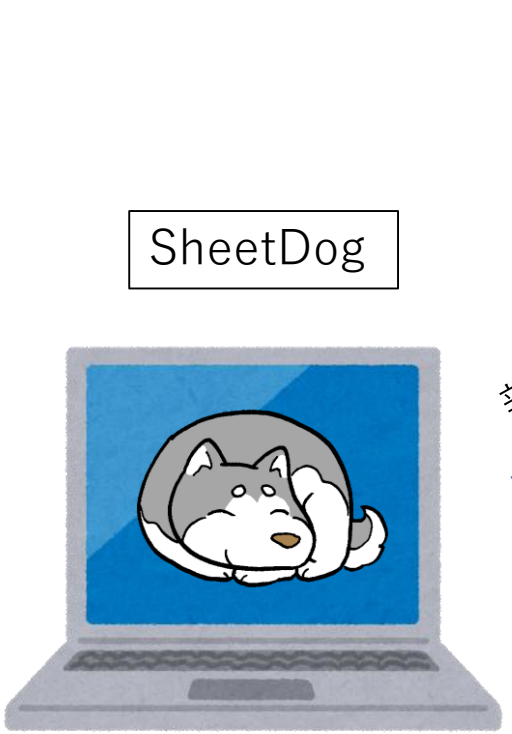
  ②登録ボタンをクリックして、アイコンの位置と電波データを送信

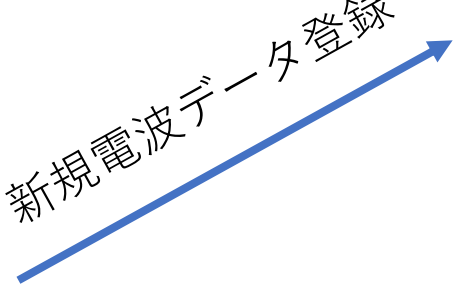
本当はここにいるので
アイコンを移動させる



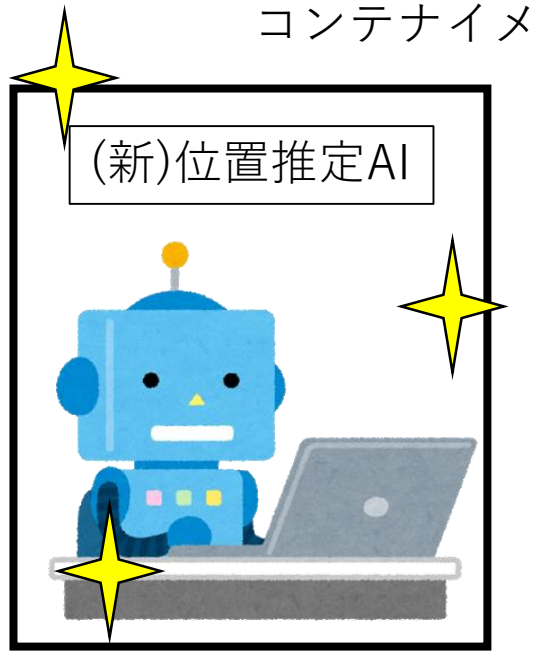
使ってるうちに賢くしたい

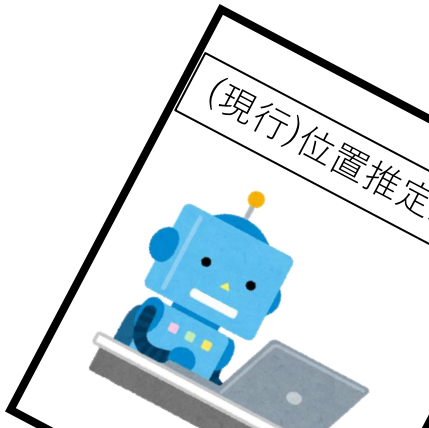
- ユーザーからデータを収集
  ⇒新しく集めたデータも使ってAIを定期的に作り直す





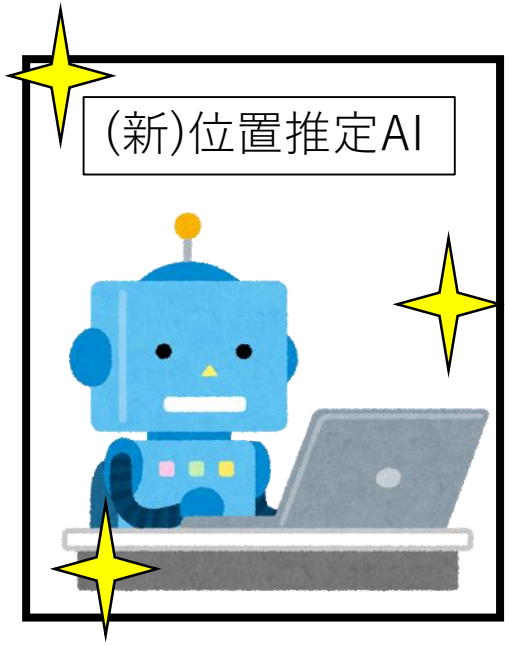
何回も作り直しをするので
コンテナイメージ化する



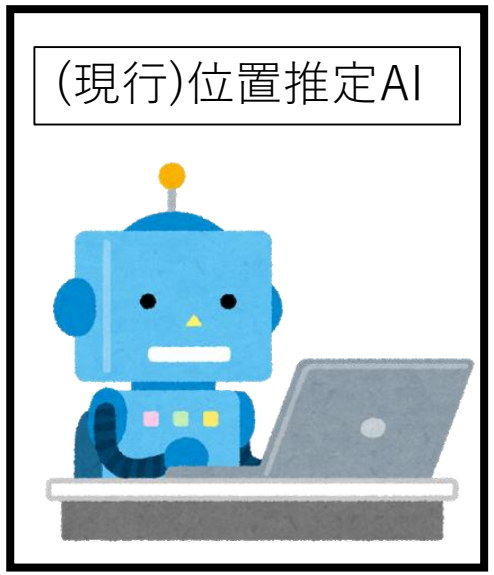


使ってるうちに賢くしたい

- 新しく作ったほうがいいものとは限らない
  ⇒同じ問題を解かせて、精度の高いほうを採用



VS



使ってるうちに賢くするプロセス

データ収集

ユーザー

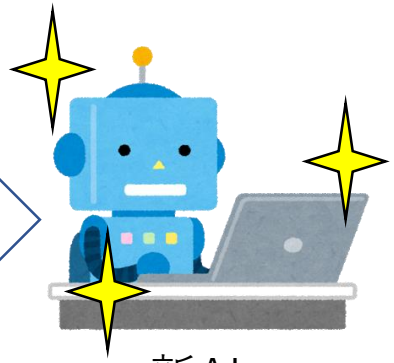
電波データ登録

電波データDB

学習

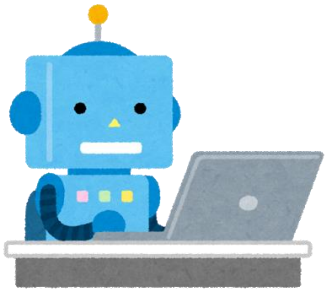
定期的に自動で
新AIを作成

評価



新AI

VS



現行AI

新AIのほうが
精度が高い

AI入れ替え

テスト、コンテナイメージ化、デプロイ

現行AIのほうが
精度が高い

更新作業終了
※データのクレンジングをしたい

使ってるうちに賢くする　※注意点その１

- 「過学習」を回避する必要がある

  ⇒学習を過度にしても、不足してもいけない

  ⇒「評価値」の向上が「一定期間」見られなければ学習を終了（EarlyStopping）

例）評価値を「精度」期間「3回」と設定したとき

学習回数	精度（%）	精度向上なしの回数
・・・		
200	80	0
201	81	0
202	79	1
203	80	2
204	80	3

201回目のAIを採用

学習終了

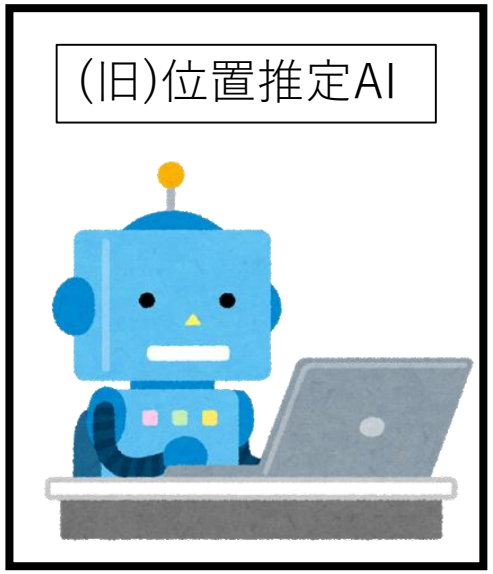
今回の条件

テストデータを入れた時の正解率が
100回
向上しなかったら学習を停止

使ってるうちに賢くする　※注意点その２

- 前のAIに戻せるように入れ替え前のコンテナイメージ、使用したコード、データを保管する

  ⇒実際に使ってみたら前のほうがよかった、という時に戻す



コンテナイメージ

・すぐに前の状態に戻せる

コード

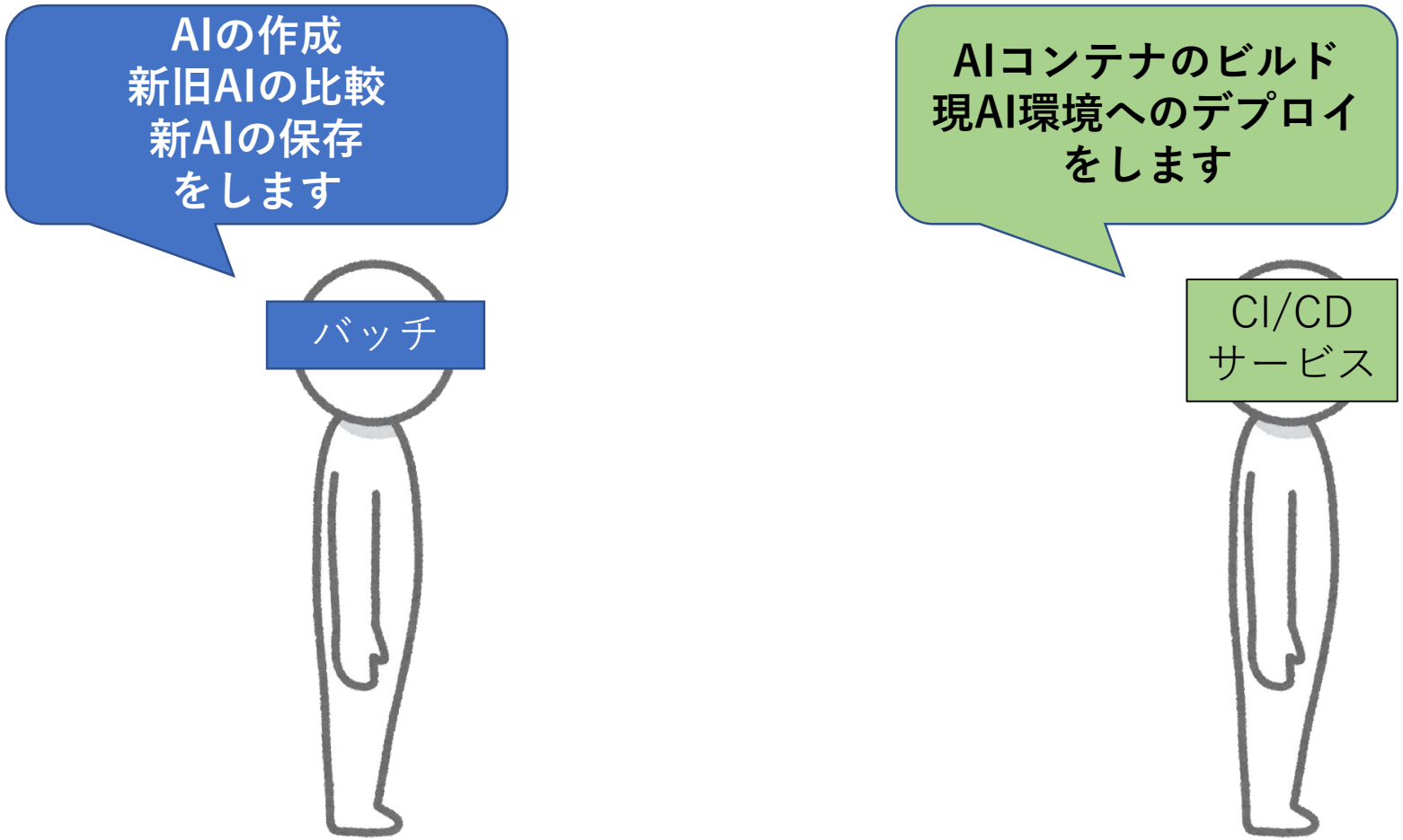
データ

・前のAIを再現できるようにしておく

・コードはタグで管理する

使ってるうちに賢くする　全体像

- バッチとサービスを使って実装



使ってるうちに賢くする　全体像

- その１　AIの作成

バッチは自動で走り出す

はじめは新しいAIを作る

AI作成環境

使ってるうちに賢くする　全体像

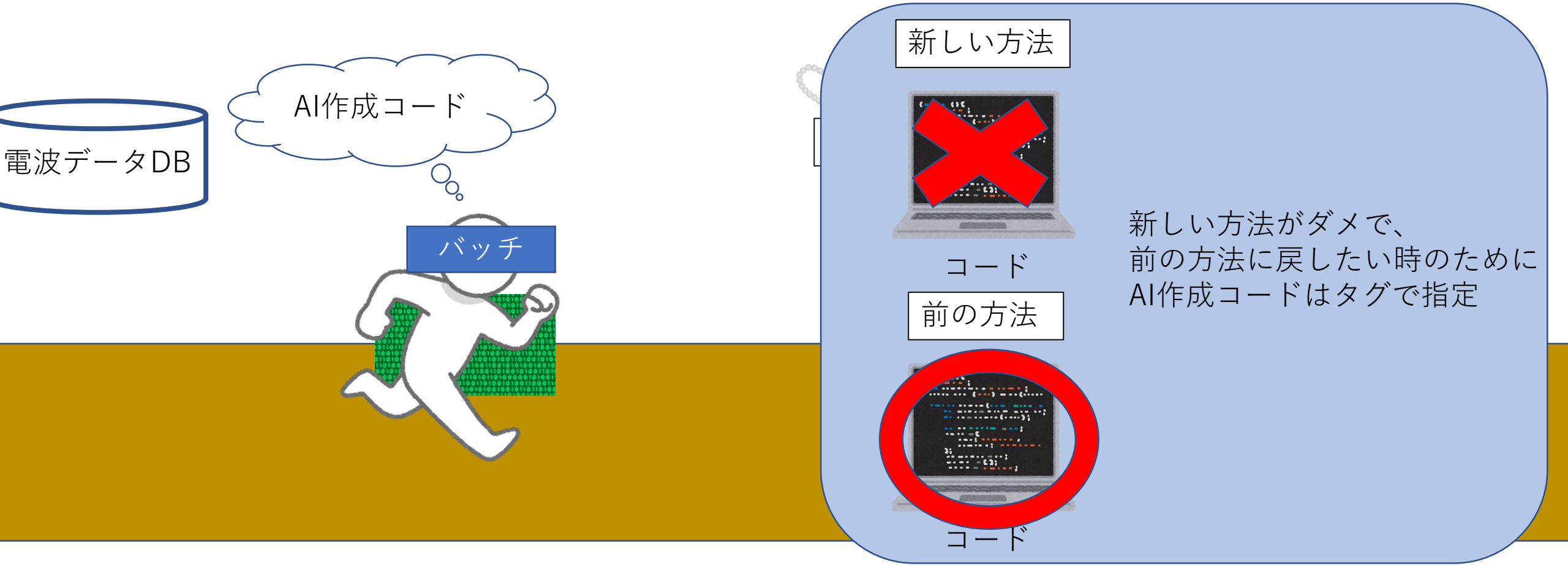
- その１　AIの作成（学習用データの取得）

学習用データをDBから取得

AI作成環境

使ってるうちに賢くする　全体像

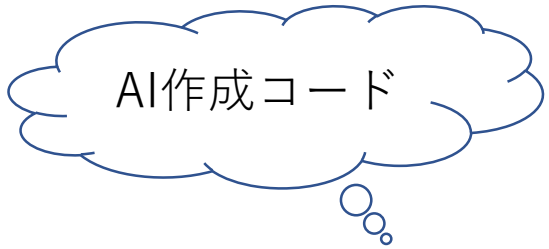
- その１　AIの作成（AI作成コードの取得）



使ってるうちに賢くする　全体像

- その１　AIの作成（AI作成コードの取得）

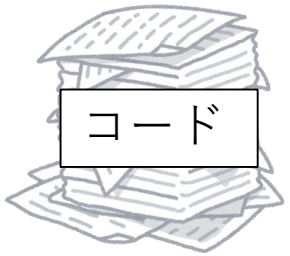
使いたいAI作成コードのタグを取得

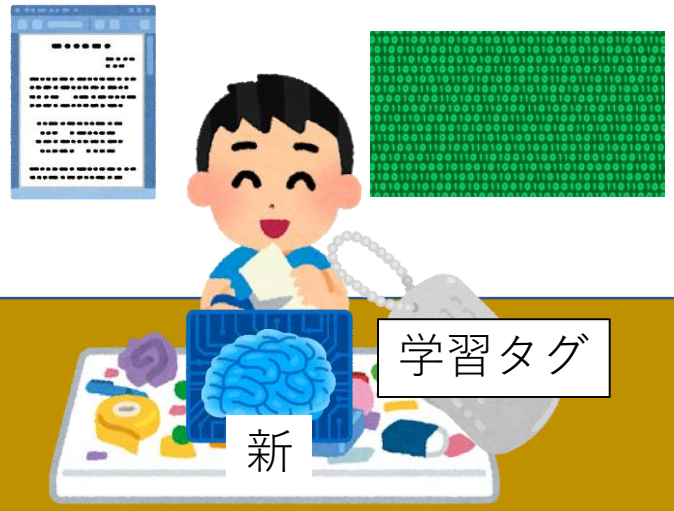


使ってるうちに賢くする　全体像

- その１　AIの作成（AI作成）

AIを作成する
学習終了タイミングは
EarlyStoppingで判定する

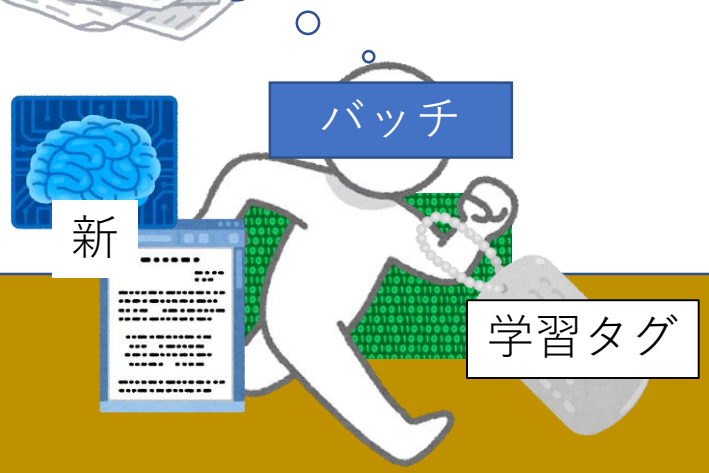




使ってるうちに賢くする　全体像

- その２　比較（現行AIの取得）

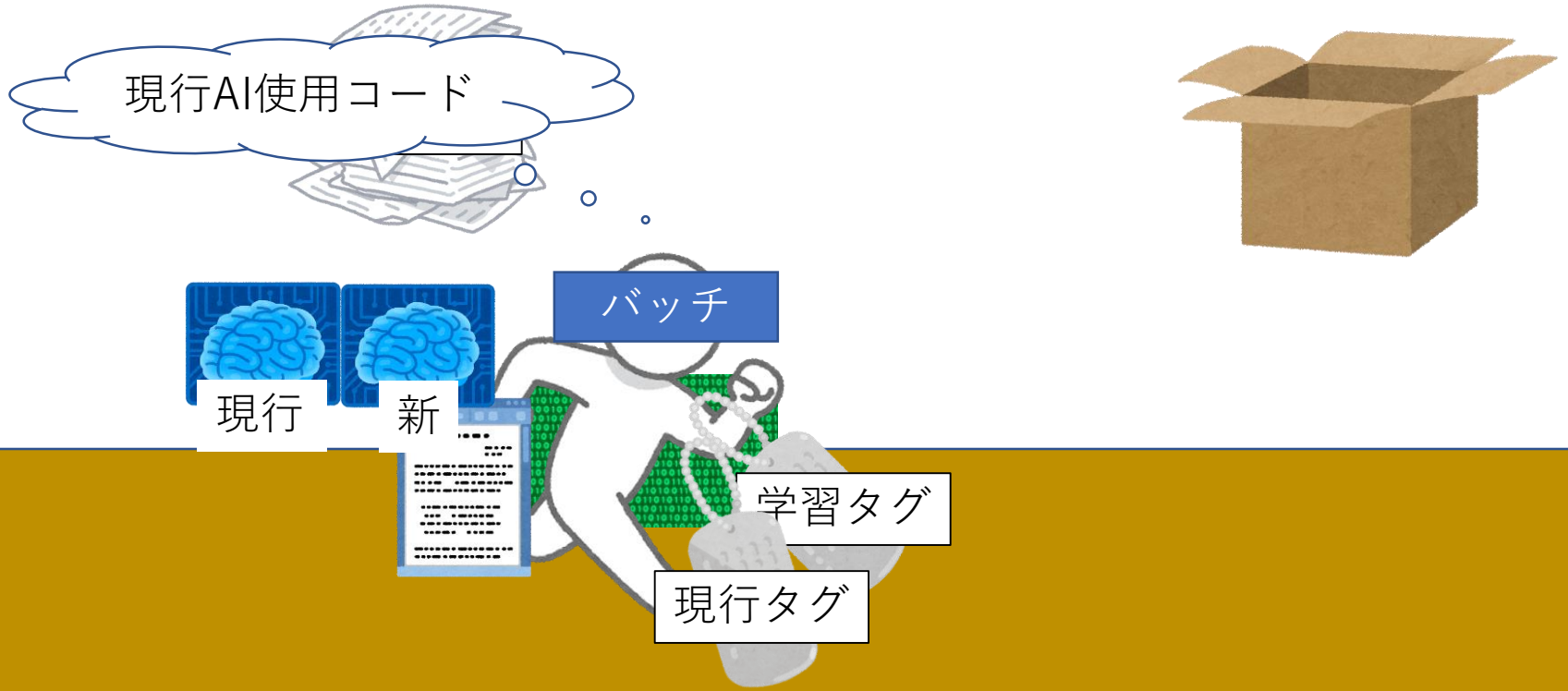
比較するために現行のAIを使う準備をする



使ってるうちに賢くする　全体像

- その２　比較（現行AIの取得）

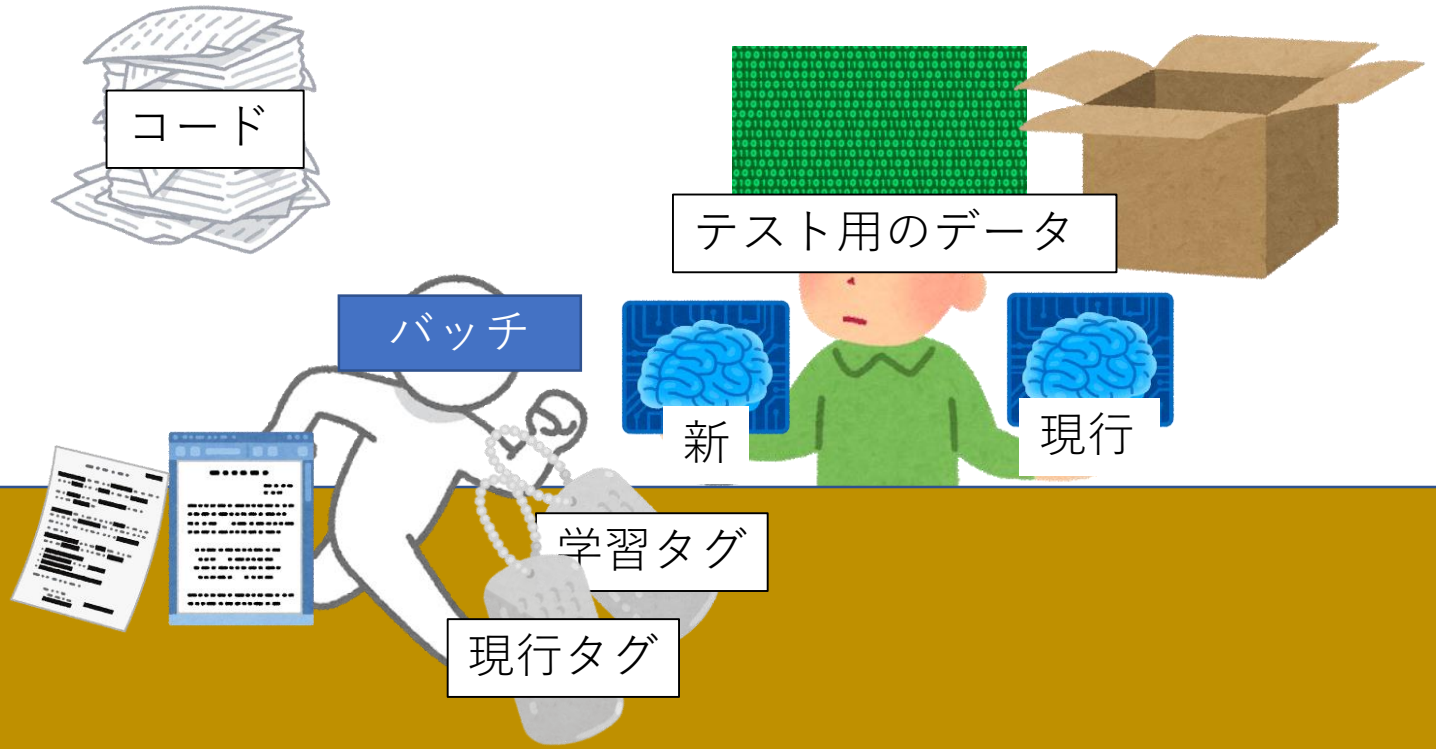
現行のAIとAIを作ったときの
コードタグを取得



使ってるうちに賢くする　全体像

- その２　比較

新しいAIと現行AIの精度を
比較する



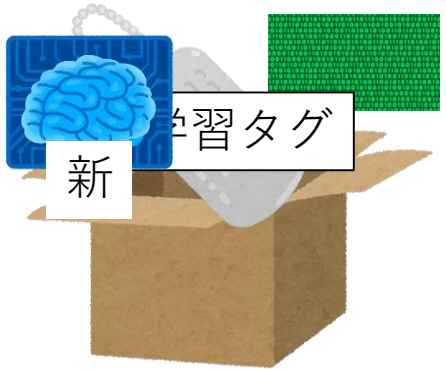
使ってるうちに賢くする　全体像

- その３　入れ替え

新しいAIの精度のほうが高ければ入れ替え作業をする

使ってるうちに賢くする　全体像

- その３　入れ替え（ストレージに保管）



新しいAIの[illegible]ださい

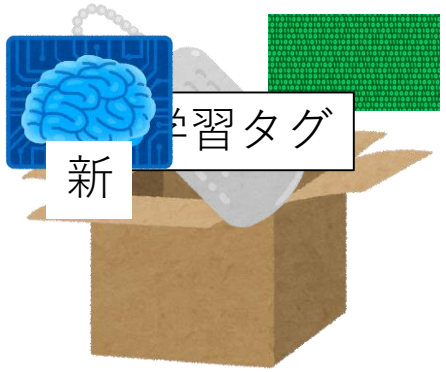
学習に使用したコードのタグと作ったAIとデータを保管する

ビルドとデプロイは別サービスを使用する

使ってるうちに賢くする　全体像

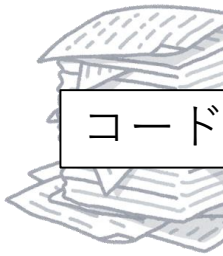
- その４　ビルド（サービスの自動開始）

AIの入れ替えがあった時に
走り出す



CI/CD環境

新しいAIの[illegible]ない



使ってるうちに賢くする　全体像

- その４　ビルド（コードの取得）

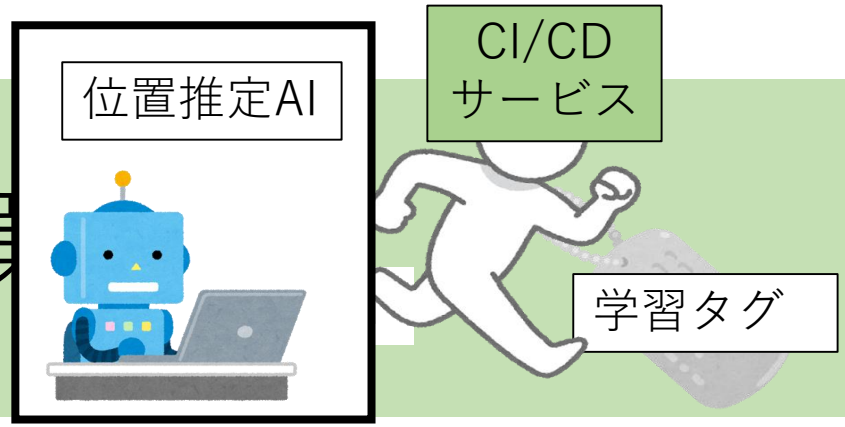
タグをもとに新しいAIを使うためのコードを取得

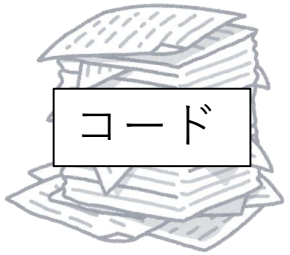
使ってるうちに賢くする　全体像

- その４　ビルド（コンテナイメージ化）

現行AIと入れ替えるために
コンテナイメージ化

CI/CD現

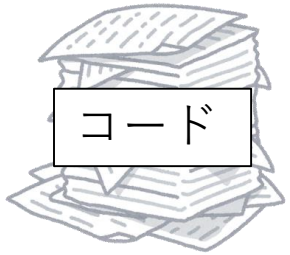


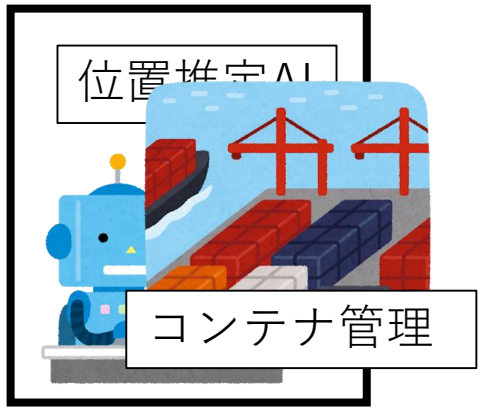


使ってるうちに賢くする　全体像

- その４　ビルド（コンテナイメージを保管）

コンテナイメージをコンテナストレージに保管

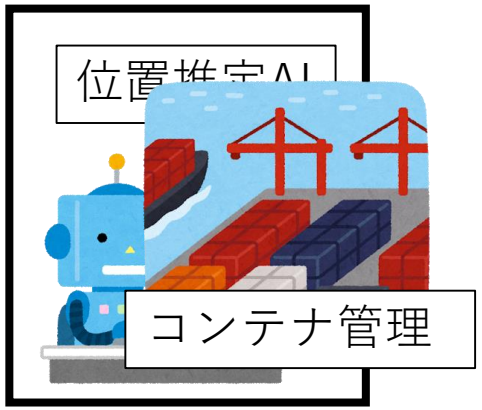




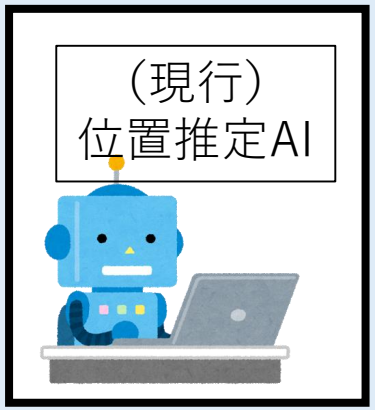
使ってるうちに賢くする　全体像

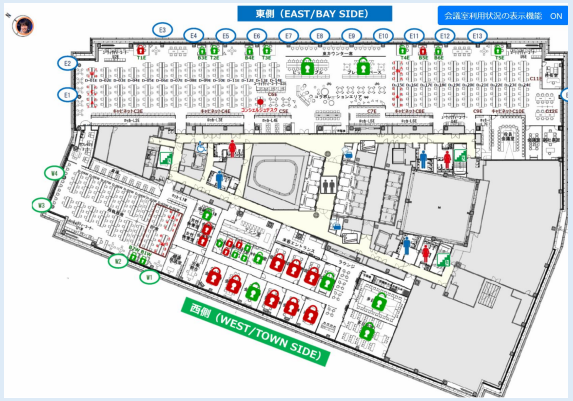
- その５　デプロイ（現行AIと入れ替え）

現行のAIと入れ替えて終了



在席表示システム

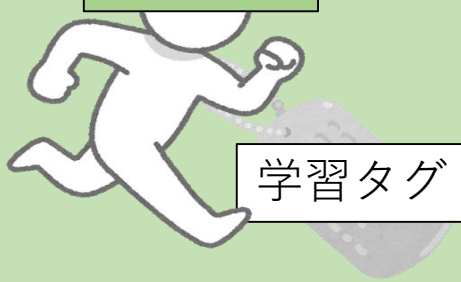




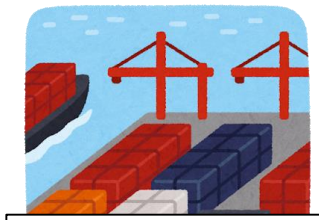
使ってるうちに賢くする　全体像

- その５　デプロイ（現行AIと入れ替え）

CI/CD
サービス



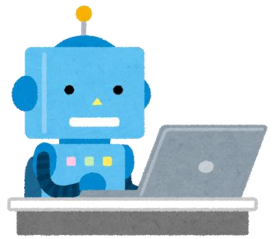
学習タグ

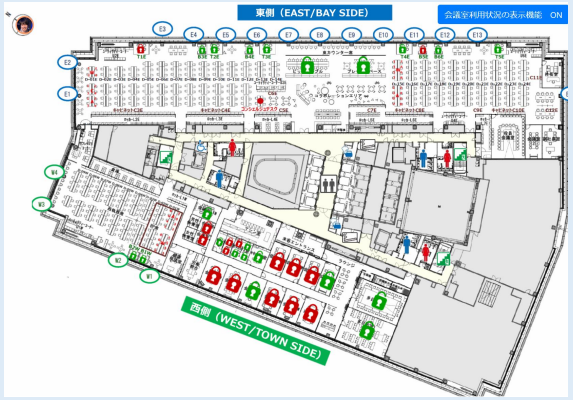


コンテナ管理

在席表示システム

位置推定AI





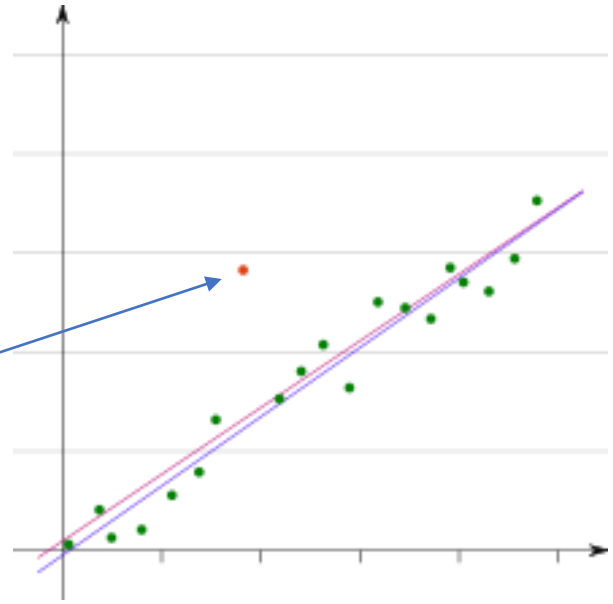
大変だったこと（やったこと）

- ユーザーからデータを集める仕組みを考える
- AIをコンテナイメージ化する
- 比較するプロセスを考える
- 自動で学習を終了させる仕組みを作る
- AIを再現できるように保管するものを考える
- AIライブラリが変わっても影響がない仕組みを考える
- 仕組みを実装する

大変だったこと（残っている問題）

- ユーザーから集めた情報が正しいものとは限らない
  ⇒誤りデータは間違いのもとなので除去したい
- 新しく作ったほうが悪かった時のプロセス
  ⇒データを増やしたら精度が下がったということなので、
  データのクレンジングをしたほうがいい

外れ値は削除したい



まとめ

- AI開発編

  Wi-Fiの電波強度から位置を推測するAIを作成

  精度の向上に効果があったこと

  – 入力データの特徴が取りやすいように変形

  – 回帰問題ではなく、分類問題として扱う

- AI運用編

  使っているうちに賢くするプロセスを考案

  ーデータが集まる仕組みを作っておく

  ー新しいAIがいいものとは限らないので、比較をしておく

  ー作ったAIを再現できるように作った時のコードとデータを残しておく

  ーAIは入れ替えが激しいのでコンテナイメージ化しておくのがおすすめ

さいごに

- お見せした通り、AIの精度はみなさんのデータに依存している部分が多くあります。ご協力よろしくお願いします。

- 今回はオーソドックスな形でつくってみたが、考慮する点が多くあるので、AWS SageMakerを使うとどれくらいこの機能が実装できるか試してみたい