沖縄･由布島　2017

「GPSを実装したパーソナルIOTのご紹介と オープンソースの機械学習活用例」

話せばわかるコンピュータの会
千葉　忠悦
2017年3月11日

1. はじめに
2. 大量ログ時代
3. パーソナルIOTの勘どころ
4. 異常値検知　Jubatus実装
5. 機械学習(Randomforest分析）

自己紹介

千葉　忠悦（ちば　ちゅうえつ）

話せばわかるコンピュータの会　　副会長

１級陸上無線技術士

電気通信主任技術者


沖縄･竹富島　2017
1．はじめに
1．はじめに
2．大量ログ時代
3．パーソナルIOTの勘どころ
4．異常値検知　Jubatus実装
5．機械学習(Randomforest分析）

東日本大震災から6年

話せばわかるコンピュータの会
定点観測活動のご紹介

検索キーワード

三陸ルネサンス

http://www7b.biglobe.ne.jp/~chibacy/test/map.html

中国人観光客向けに「伊達政宗」をテーマにした
オプショナルツアーを企画（仙台市　観光振興部署）

中国人観光客

伊達政宗て
知らない？

サンドイッチマンの伊達
なら知っているよ！

ヒットせず

仙台城跡

伊達政宗像

伊達政宗生誕地

中国人になじみで、仙台ゆかりの「魯迅」をテーマにした
オプショナルツァーを企画

中国人観光客

魯迅だったら
中国人なら知っているよ！

魯迅像(東北大学)

階段教室(東北大学)

大人気となる

魯迅旧居(東北大学近く)

中国人観光客の仙台についての認知度、関心度は？

高い

低い

知ってる人もいる

- IOTの普及により、
  益々いろいろなデータが手に入れやすくなる

- 手に入れたデータからいかに新たな価値
  を見つけるのか？

- それには、ソースの違うデータを結合する
  ことが重要になる

タイ・クラビ　2006

IOT各国の取り組み

①アメリカの取り組み（インダストリー・インターネット）

・モノをソフトウエアで再定義する
・IOTから収集した膨大なデータの分析、再利用
・サービスまで含めたビジネスモデルの再構築

産業分野向け

インダストリアル
・インターネット
・コンソーシアム

GE、IBM、INTEL
シスコ、AT&T 他

参加数　約115社

コンシューマ向け

オールシーン
アライアンス

Qualcomm、マイクロソフト
ソニー　他

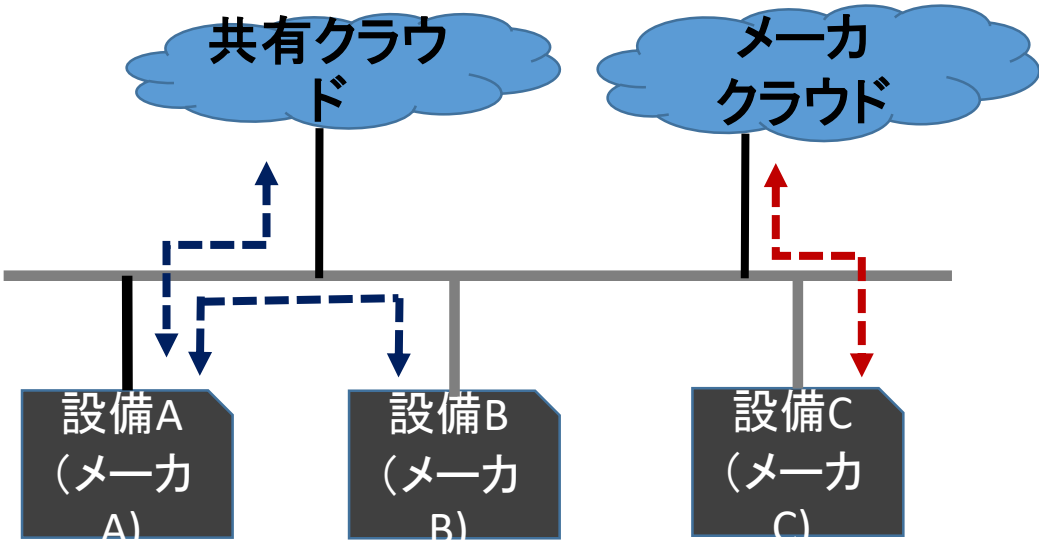
参加数　約100社

日経ビジネスより

②ドイツの取り組み（Industrie 4.0）

Industrie 4.0

ドイツ政府
SIMENS, BOSCH,SAP
BMW,VW
研究所、大学
中小企業、方政府、他



Industrie 4.0のねらい

1）IOT, ネットワーク、機械学習等を活用し
　生産工程のデジタル化、自動化、バーチャル化のレベルを大幅に高め
　生産効率の向上と競争力を高める
2）マスカスタマイゼーションの推進
3）標準化で先手をとる
　IOT双方の通信プロトコルなど

③インドの取り組み（IOT ポリシー）

・メーク・イン・インディア　工場誘致政策
・デジタル・インディア　デジタルインフラ整備
の橋渡しをする政策としてIOTポリシーを掲げる

インフラ整備の遅れを
IOT製品の開発で
取り戻す

500都市
スマートシティ化

大学教育にIOT関連
カリキュラムを導入

日経ビジネスより

④日本の取り組み

遅れ要因

もの作りの成功体験
から脱皮できない

ITシステムを
事業の裏方として
とらえる傾向が強い

先行要因

産業機器の自動化、
センサー技術高度化
で長い経験がある

IOTに必要な要素技術
がそろっている

IOTを新しいビジネスモデルを作る
キーテクノロジーととらえる

世界的なIOTの流れである
標準化、デファクトスタンダード化に参画していく

⑤IOTがもたらす具体的な変化とは

成熟期

公共サービスのIOT化

Industrial Internet
Industrie 4.0

マスカスタマイゼイション
生産

モノのソフトによる
再定義

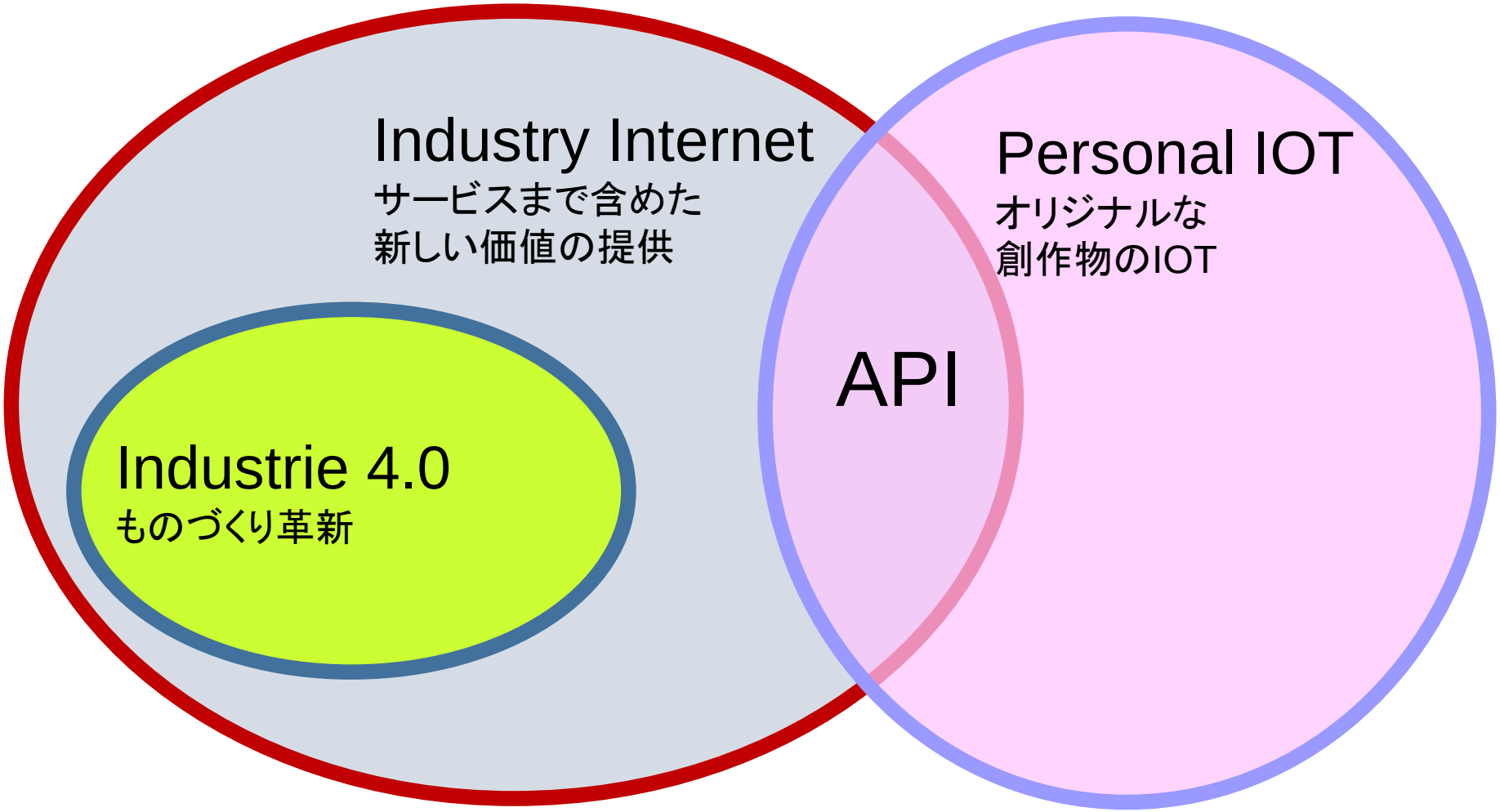
ソフトによりハードの潜在力
を引き出す

タイ・ピピ島　2004

2. 大量ログ時代

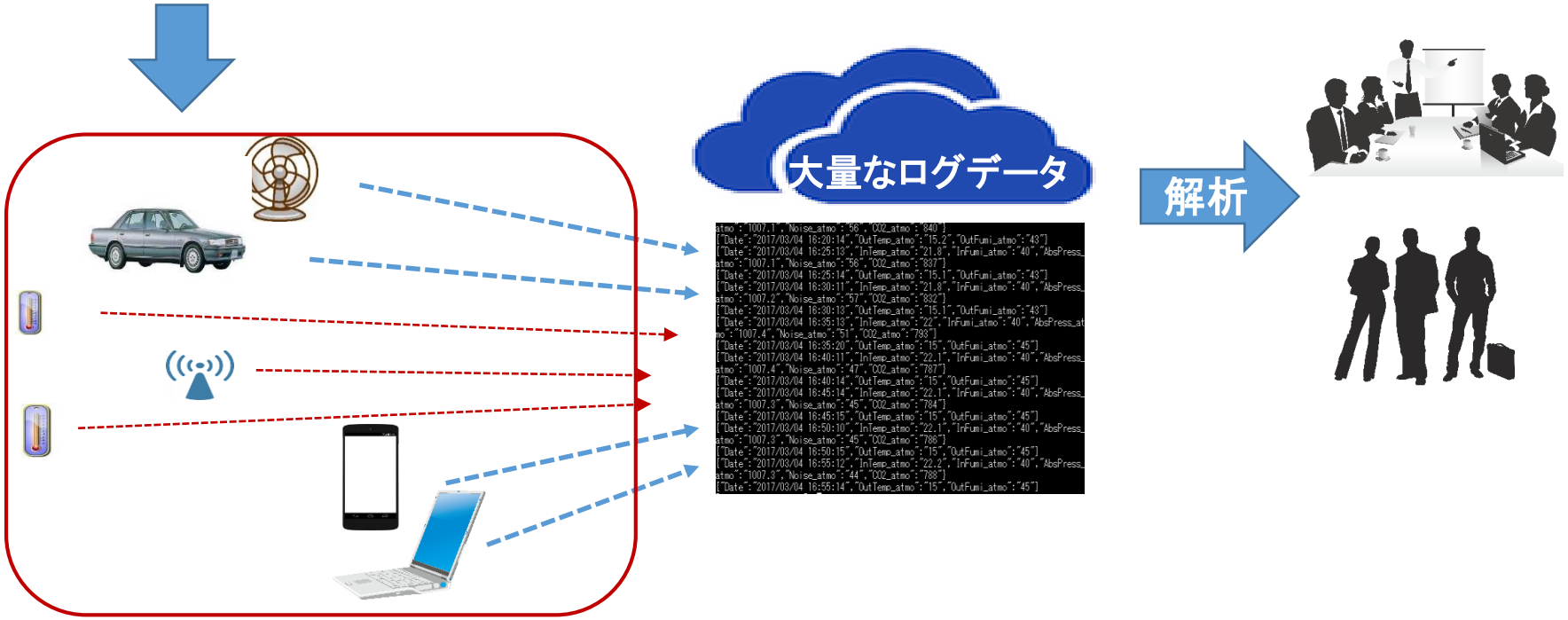
1. はじめに
2. 大量ログ時代
3. パーソナルIOTの勘どころ
4. 異常値検知　Jubatus実装
5. 機械学習(Randomforest分析）

①IOT Map


Industry Internet
サービスまで含めた
新しい価値の提供
Industrie 4.0
ものづくり革新
API
Personal IOT
オリジナルな
創作物のIOT

②IOTの普及=>大量ログ時代

IOT (Internet of Things) モノのインターネット



従来
人の操作によって、
インターネットと接続されていた。

IOT移行
モノがインターネットに接続され
データ送受信がモノどうしでも行われる。

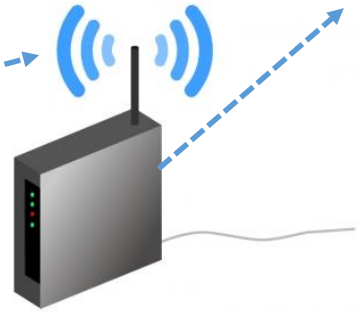
センサー　Network　ソフトウエア　クラウド

③市販IOTの例

Netatmo ウェザーステーション

http://www.focal.co.jp/products/detail.php?product_id=84

Wi-Fi

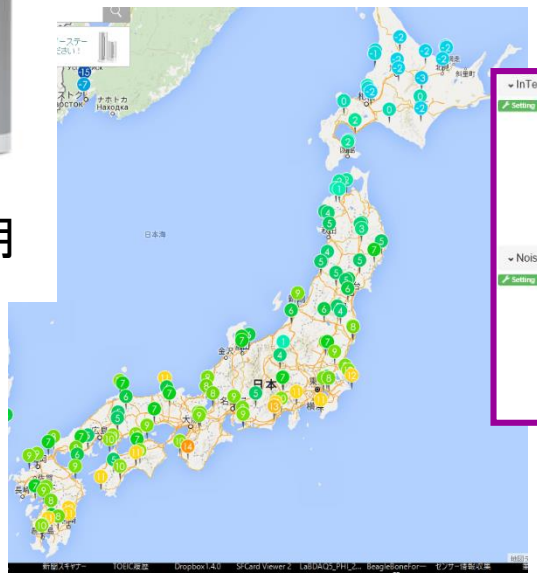


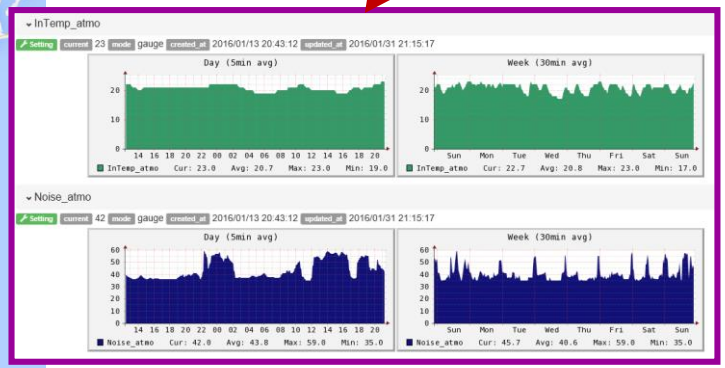
Netatmo クラウド

API

MaaS クラウド

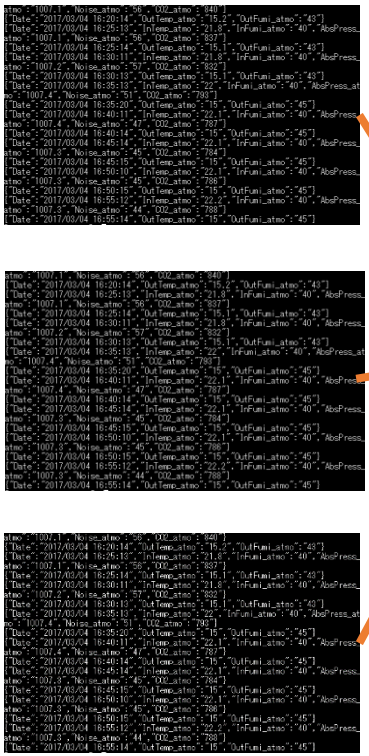
室外用　室内用





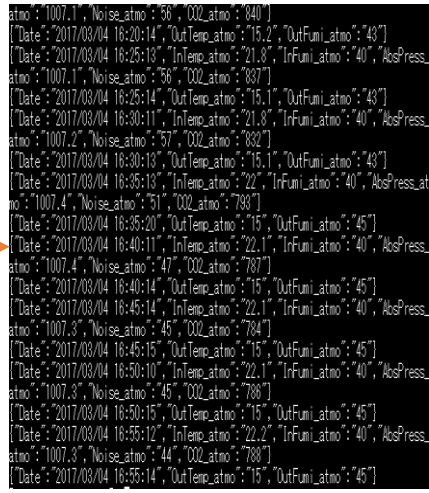
④大量ログデータの時代

**Sourceの違うログデータを
いかに結合(Merge)するか**



日付時刻での結合

結合

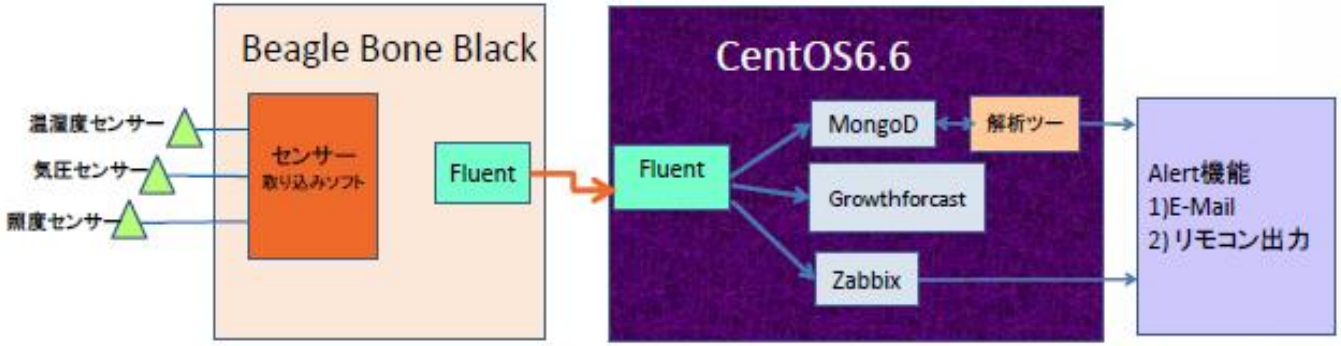



ロンドン・グリニッジ　2015
3. パーソナルIOTの勘どころ
1. はじめに
2. 大量ログ時代
3. パーソナルIOTの勘どころ
4. 異常値検知　Jubatus実装
5. 機械学習(Randomforest分析）

パーソナル IOTの実証実験

オープンソースをベースにしたスマートセンサーシステムの紹介
話せばわかるコンピュータの会
http://www7b.biglobe.ne.jp/~chibacy/maas/top.html

IOTセンサーデータの収集、解析、活用までを
オープンソースで一気通貫のプラットフォームで実現しました。



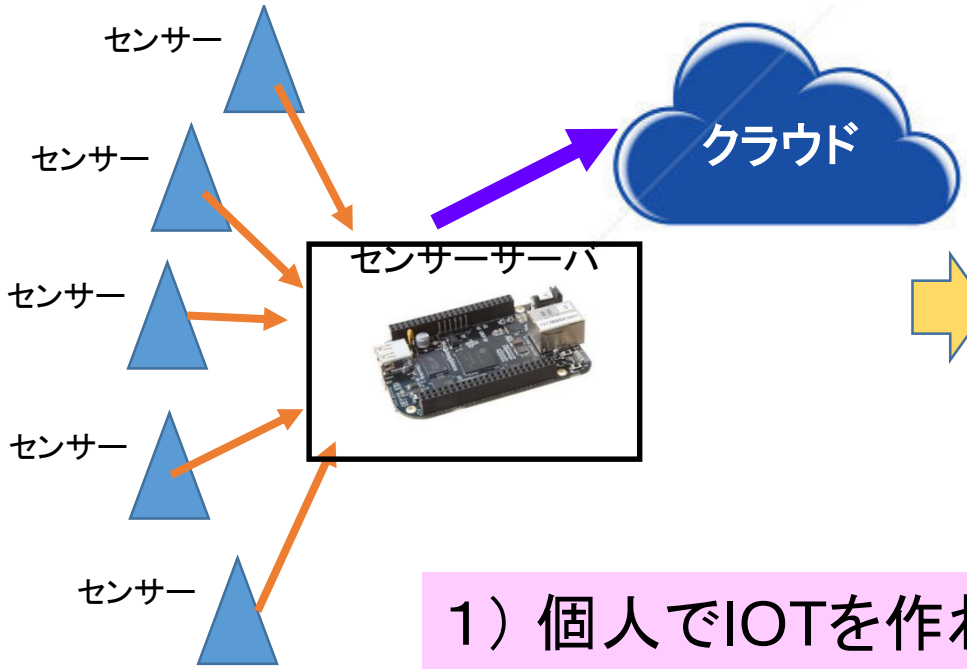
オープンソースを活用したIOT システム　ブロック図

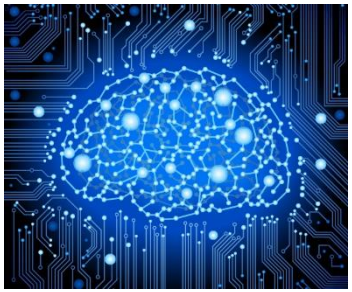
IOTのデータ収集ノード
- Beagle Bone Black(Debian)
- I2C Interface　センサー
- 3Dプリンタでの機構部分製

データ集配
- データ形式の標準化(JSON形式)
- Fluentでの収集、分配

データ解析・アクション
- NoSQL Mongodbの活用
- R言語での解析
- Zabbixでのアラート出力

①パーソナルとしてのIOT活用





Personal IOT

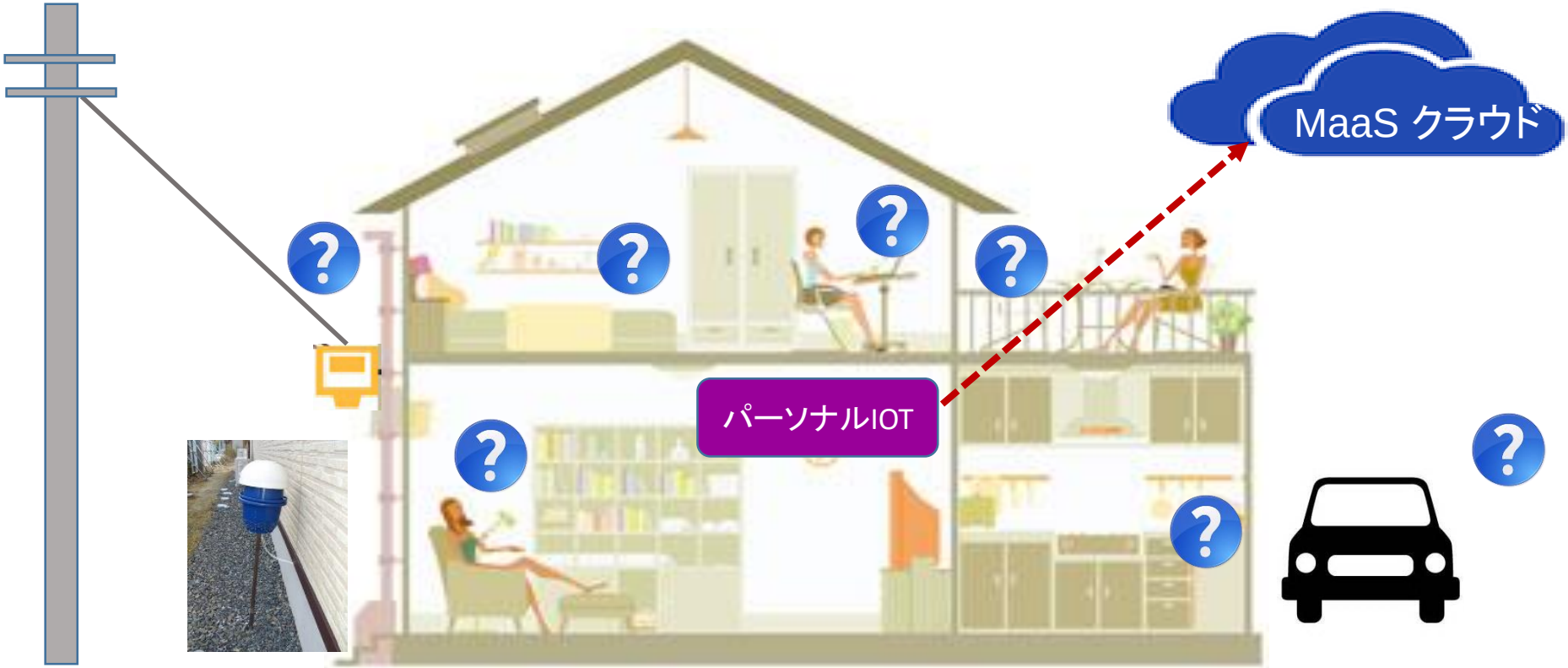
1）個人でIOTを作れる時代
2）知恵と創造力でIOTを活用する波を起こそう

センサー → クラウド → AI → IOT

②パーソナルIOTの勘どころ

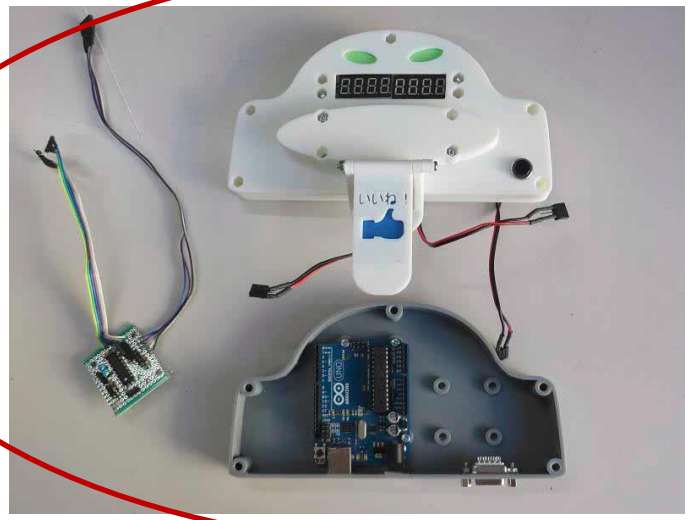
潜在している情報の顕在化

習慣の見える化 → 1）計量値 2）計数値 → さて何を、どのように見える化するか → 取得データの連携



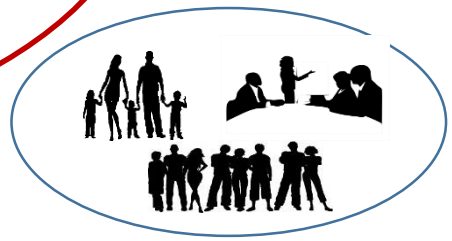
③パーソナルIOTを実現する要素技術

1）センサー、ハードウエア、3Dプリント
2）シングルボードコンピュータ
3）ネットワーク
4）OSSの活用（API,クラウド）



クラウド

連携する

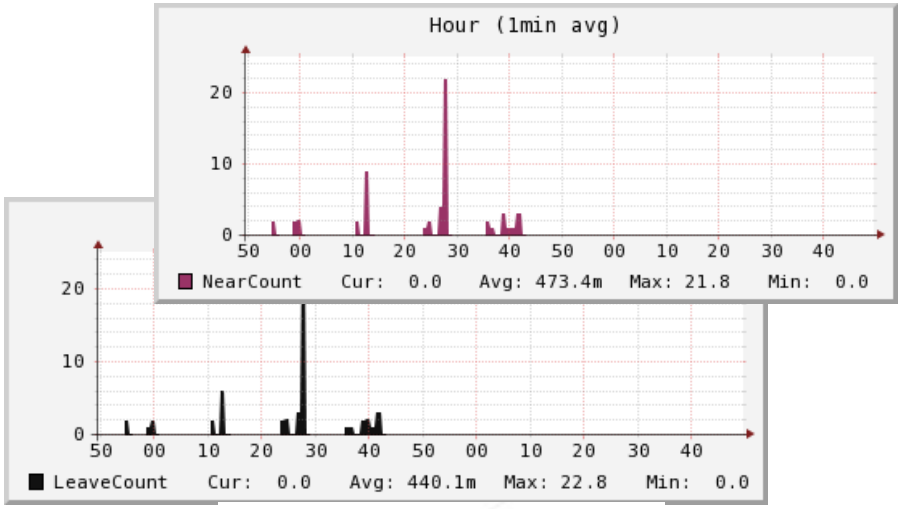


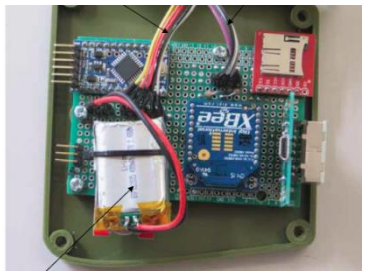
自分の持っていない
要素をどう**補完**するか？

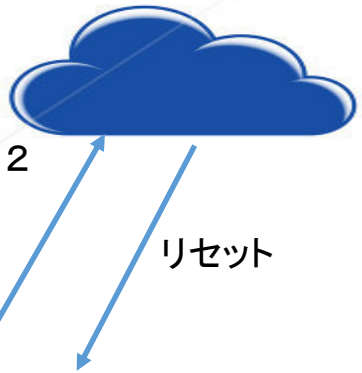
④パーソナルIOTの例

お近づきカウンター

1）ドップラセンサーで通行量をカウント
2）時系列での通行量の把握



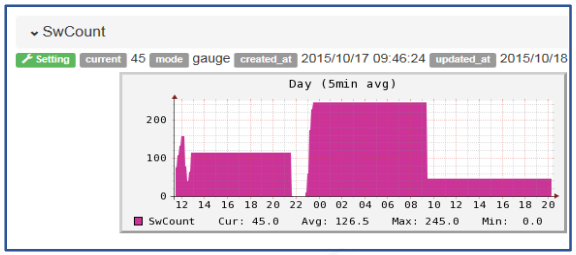




センサーサーバ

⑤パーソナルIOTの例：おもしろカウンタ

身の回りにある回数を数えるIOT
1）改めて、数えることにより見えてくること
2）関連情報と比較して、見えてくること



いいね；108

リセット

センサーサーバ

カウンタ値；108

リセット

汎用カウンタ

- Webからリアルの世界に飛び出たのが Personal IOT

Personal IOTを実現するには？

- 自分のアイデアをまずは描く
- 協力体制をいかに組むか
- OSSの活用がカギ

⑥パーソナルIOTへGPSを実装

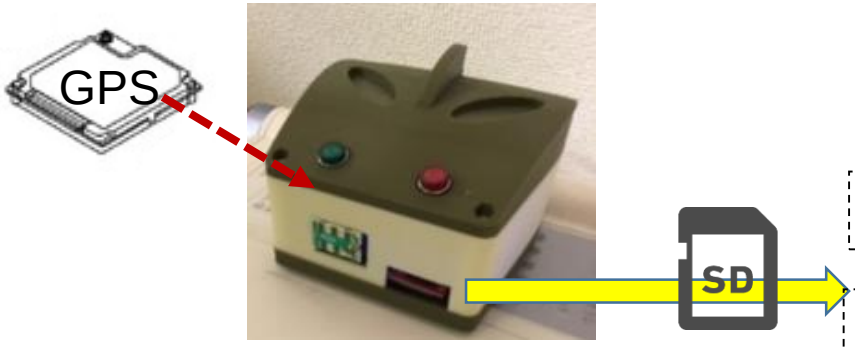
ログデータの結合のために、正確なDatetimeの記録が必要！

お近づきカウンターを単独起動の場合
時刻を刻む方法がないため、GPSを実装し
時刻情報とともに、位置情報のSDカードに記録するように改修

SDデータ記録例

GPSデータ（追加データ）

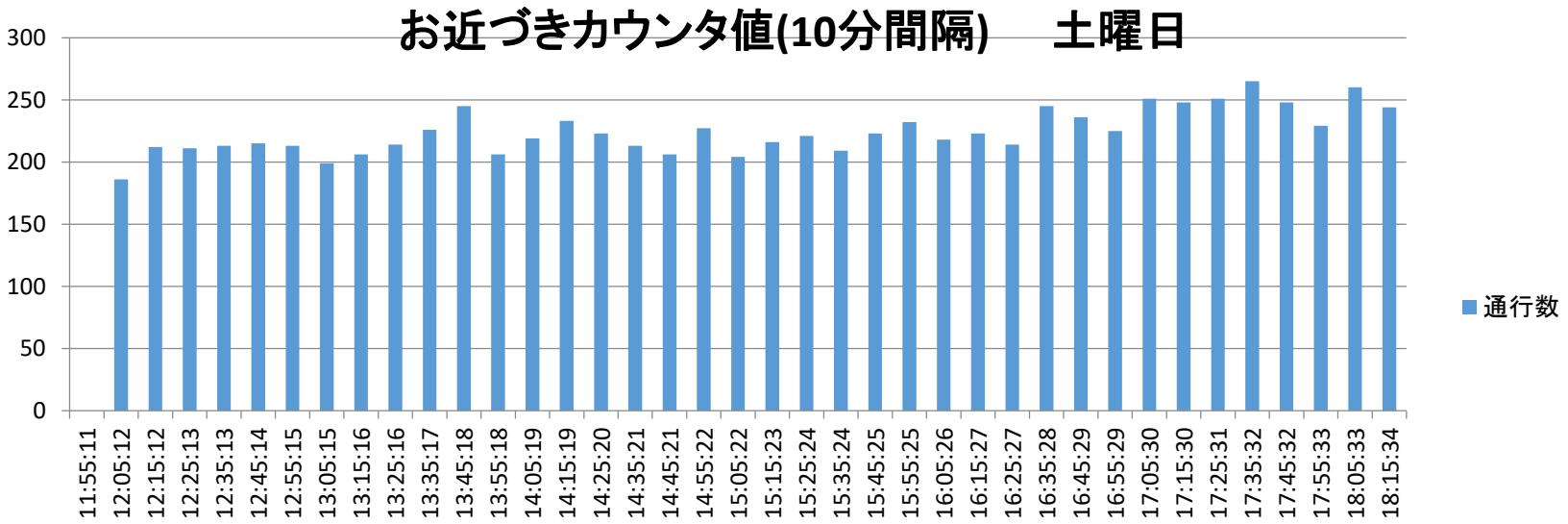
"lat";緯度データ,"lng";経度データ



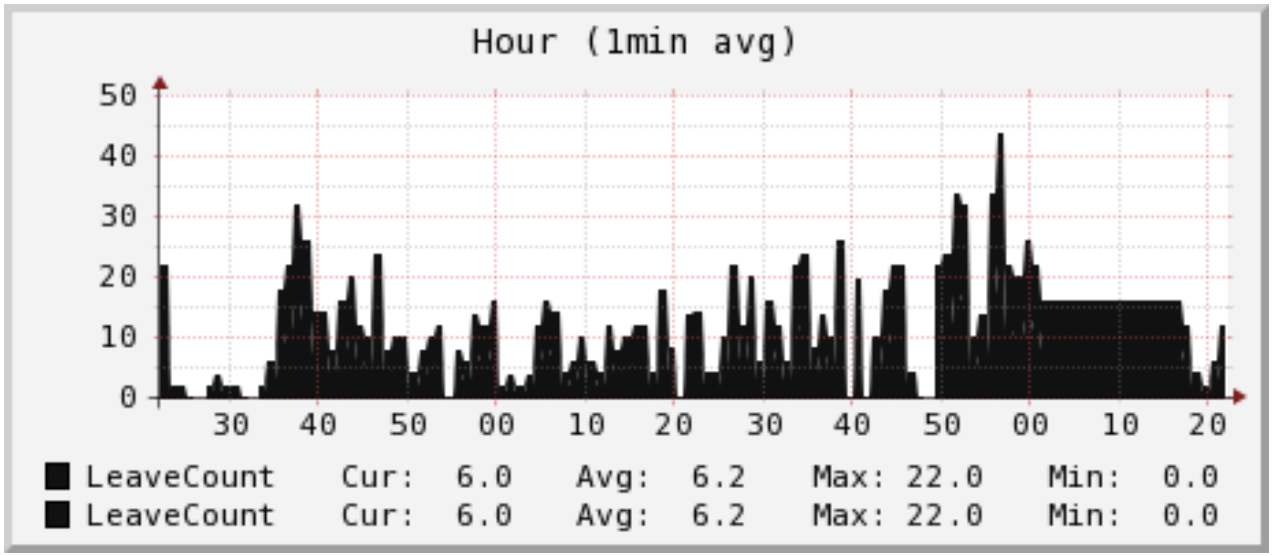
DateTime,APPR,AWAY

2017/03/11 11:00:00,77,78

⑦お近づきカウンタ　フィールド評価



昨日のお近づきカウンタ値



- IOTから出されるログデータをどう活用するか？
- 他のログデータとの連携がカギ
- ログデータどおしの結合にはDatetimeが重要

4. 異常値検知　Jubatus実装

1. はじめに
2. 大量ログ時代
3. パーソナルIOTの勘どころ
4. 異常値検知　Jubatus実装
5. 機械学習(Randomforest分析）

①異常値検知(機械学習;Jubatus)

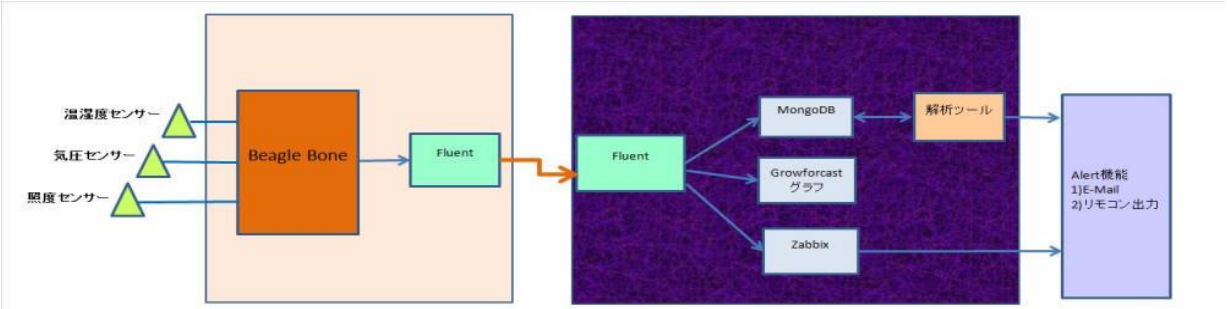


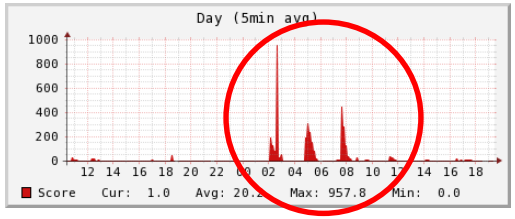
図1 オープンソースを活用したIOTシステム ダイアグラム

各項目の閾値判定ではない
多数項目による異常値検知とは？

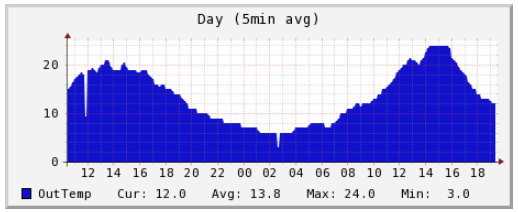
パーソナルIOT

MaaS クラウド

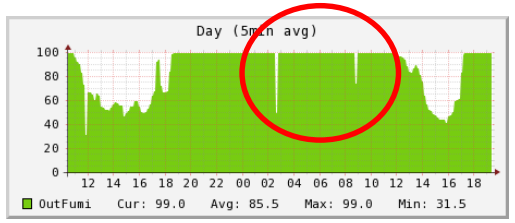
異常値スコア



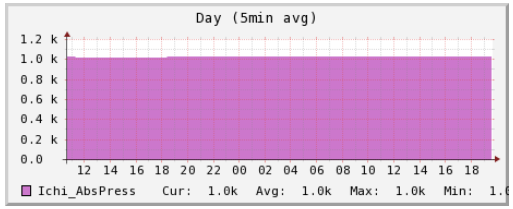
外気温度



外気湿度



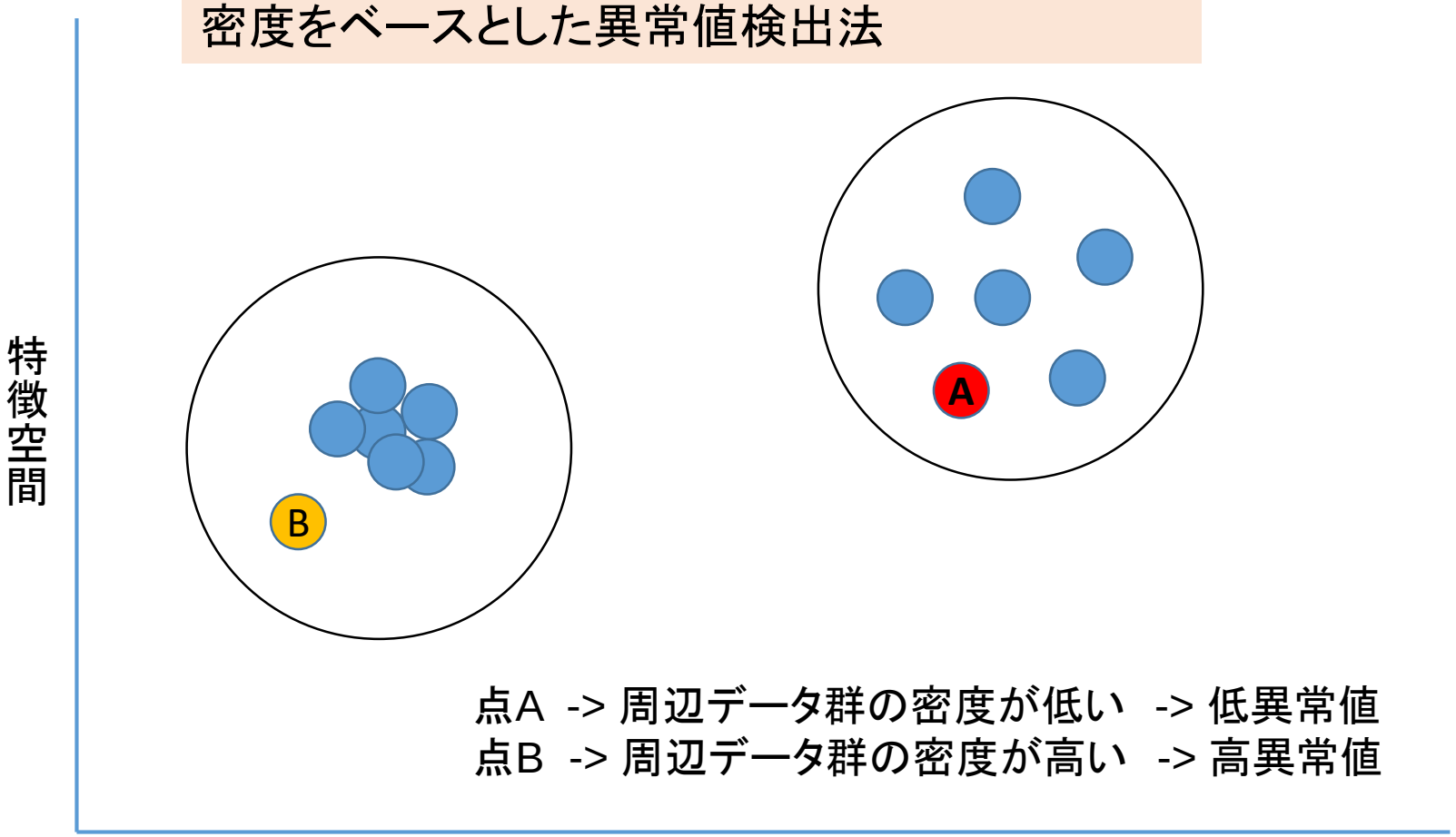
気圧



②Jubatusの異常検知

公式サイト　http://jubat.us/ja/

LOFは距離をベースとした異常検出法ではなく、
密度をベースとした異常値検出法



点A　-> 周辺データ群の密度が低い　-> 低異常値
点B　-> 周辺データ群の密度が高い　-> 高異常値

2次元の特徴ベクトルをもつデータの例

Jubatusの特徴

データ解析をリアルタイムで行うフレームワーク

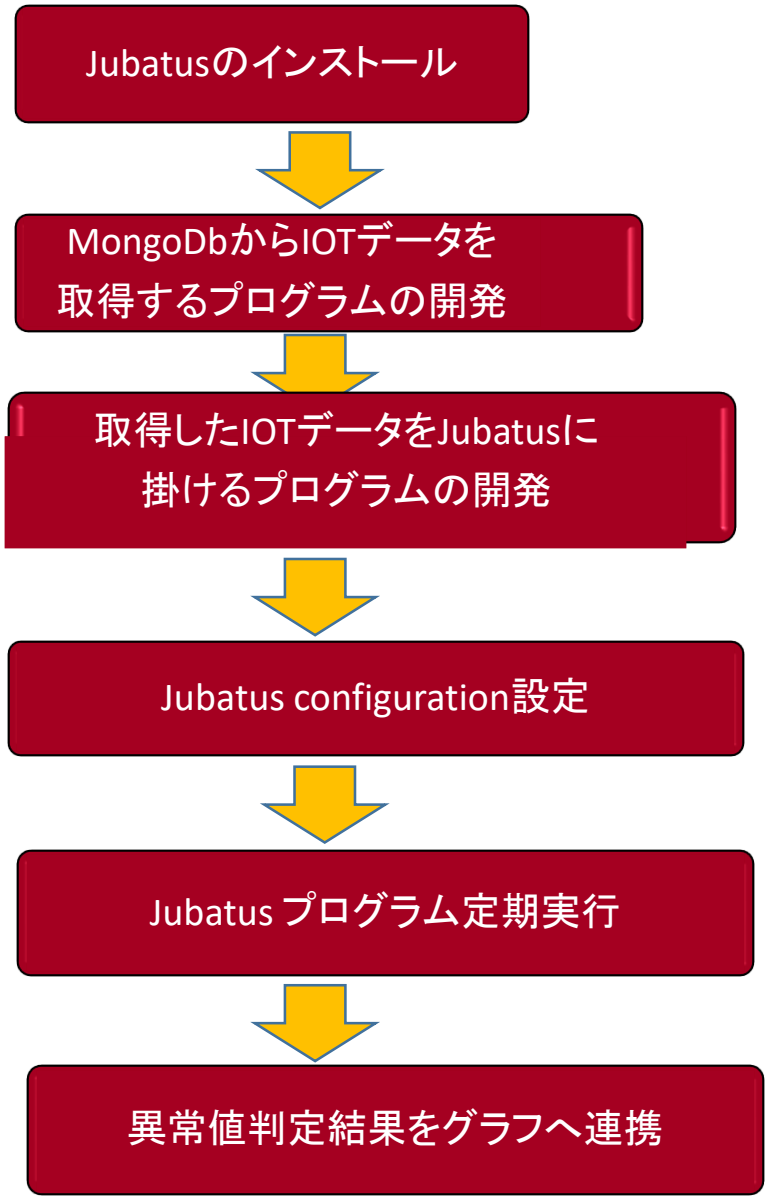
1)リアルタイムで大規模なデータ分析が可能

2)多値分類、異常検知、クラスタリング
　主な機械学習ライブラリを提供している

3)日本で開発されたOSS

4)LOFアルゴリズムを実装している

③異常検知(Jubatus)の導入手順



Jubatus実行プログラム　python

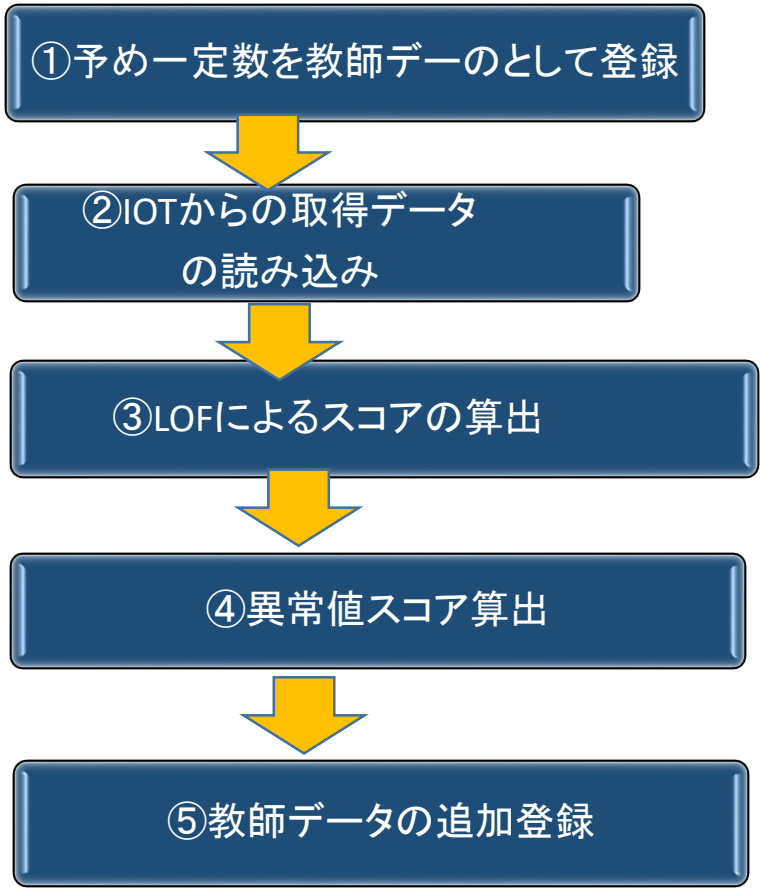
```
NAME = "lux";
#coding:UTF-8
import sys, json
import datetime
import os
from jubatus.anomaly import client
from jubatus.common import Datum
from get_10min import getmongo
Logfilename = "/var/log/juba.log"

def saveMeasDataLog():
        nowDate = datetime.datetime.today()
        CurMeasDatetime = '%04d/%02d/%02d %02d:%02d:%02d' %
(nowDate.year, nowDate.month, nowDate.day, nowDate.hour, nowDate.minute, nowDate.second)
        str1 = '{"Date":"'+ CurMeasDatetime + '",'
        str2 = '"OutTemp":"' + s_outtemp + '",'
        str3 = '"OutFumi":"' + s_outfumi + '",'
        str4 = '"Ichi_AbsPress":"' + s_ichiabs + '",'
        str5 = '"Score":"' + CurrScore + '"}¥r¥n'
        str = str1 + str2 + str3 + str4 + str5
        flg = os.path.exists(Logfilename)
        if not flg:
                fh = open(Logfilename, "w")
                fh.write(str)
                fh.close()

        fh = open(Logfilename, "a")
        fh.write(str)
        fh.close()
```

```
NAME = "lux";

if __name__ == '__main__':

    # 1.Jubatus Serverへの接続設定
    anom = client.Anomaly("127.0.0.1",9199,NAME)

    # 2.学習用データの準備
    getmongo =  getmongo()
    dic = getmongo.getDic()
    name = ''
    value = 0
    for line in dic:
      if (0 < line):
        outfumi = dic[line]['OutFumi']
        outtemp = dic[line]['OutTemp']
        ichiabspress = dic[line]['Ichi_AbsPress']
        s_outfumi = str(outfumi)
        s_outtemp = str(outtemp)
        s_ichiabs = str(ichiabspress)
        datum = Datum()

        for (k, v) in [
                ['name', name],
                ]:
            datum.add_string(k, v)

        for (k, v) in [
                ['outfumi', outfumi],
                ]:
            datum.add_number(k, v)
        for (k, v) in [
                ['outtemp', outtemp],
                ]:
            datum.add_number(k, v)
        for (k, v) in [
                ['ichiabspress', ichiabspress],
                ]:
            datum.add_number(k, v)
        # 3.データの学習（学習モデルの更新）
        ret = anom.add(datum)
        test1 = str(ret.score)
        print test1
        # 4.結果の出力
        if (ret.score == float('Inf')):
            ret.score = 0
            CurrScore = str(ret.score)
            saveMeasDataLog()
            print ret
        else:
            CurrScore = str(ret.score)
            saveMeasDataLog()
            print ret
```

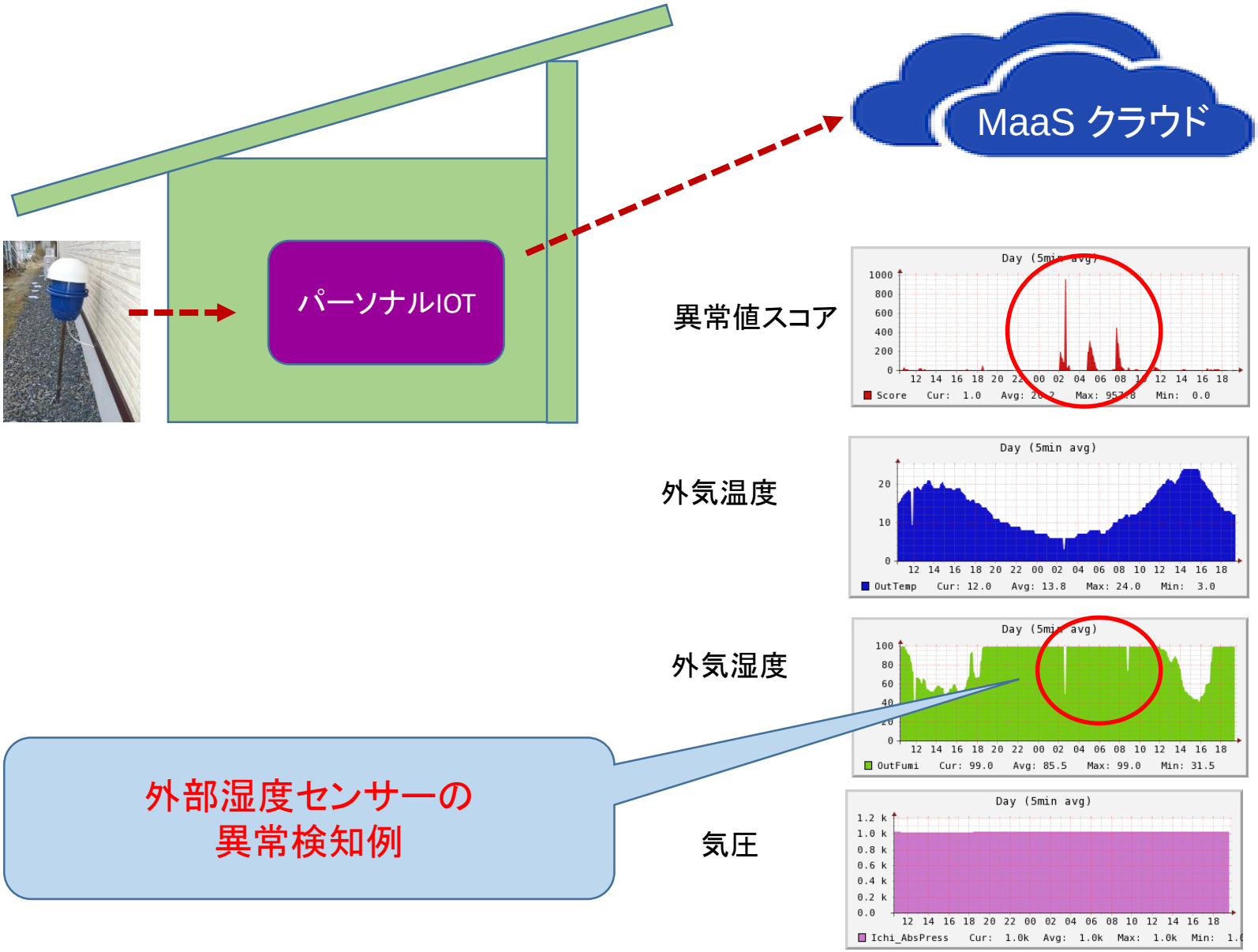
Jubatus パラメータ
lof.json

```
{
  "converter" : {
    "string_filter_types" : {},
    "string_filter_rules" : [],
    "num_filter_types" : {},
    "num_filter_rules" : [],
    "string_types" : {},
    "string_rules" : [
      { "key" : "*", "type" : "str", "sample_weight" : "bin", "global_weight" : "bin" }
    ],
    "num_types" : {},
    "num_rules" : [
      { "key" : "*", "type" : "num" }
    ]
  },
  "parameter" : {
    "nearest_neighbor_num" : 10,
    "reverse_nearest_neighbor_num" : 30,
    "method" : "euclid_lsh",
    "parameter" : {
      "hash_num" : 16,
      "table_num" : 16,
      "seed" : 1091,
      "probe_num" : 64,
      "bin_width" : 100000
    },
    "unlearner" : "lru",
    "unlearner_parameter" : {
        "max_size" : 1048576
    }
  },
  "method" : "lof"
}
```

④異常検知(Jubatus)の動作手順



⑤異常値検知（機械学習；Jubatus） 例1



湿度センサーのばらつき検出

結果　センサー基板交換前と今回の基板との違いは実装部品
R1の値が異なっている事が解った。

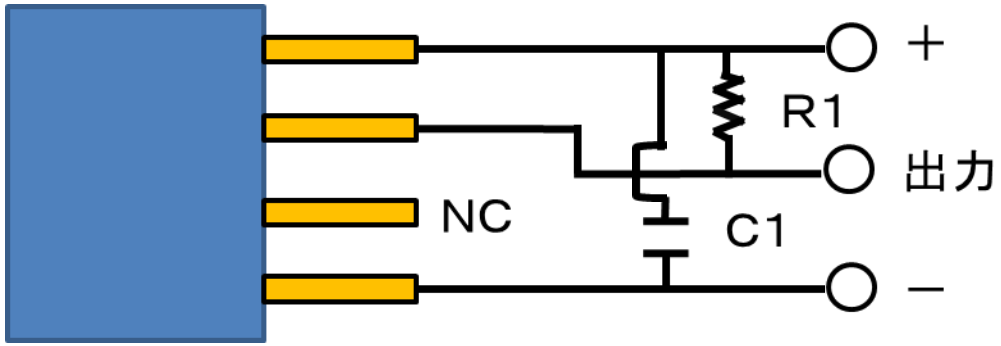


図1

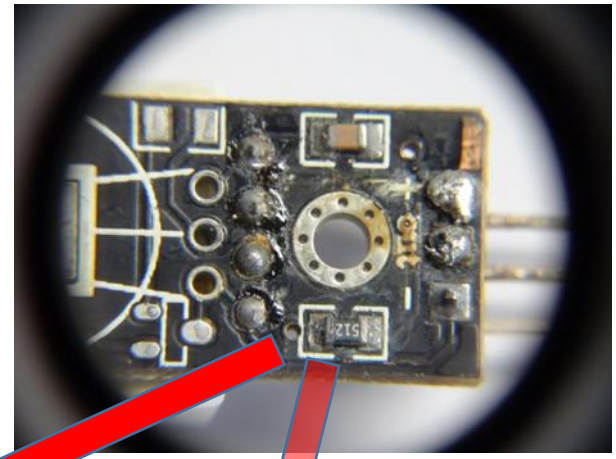


写真1

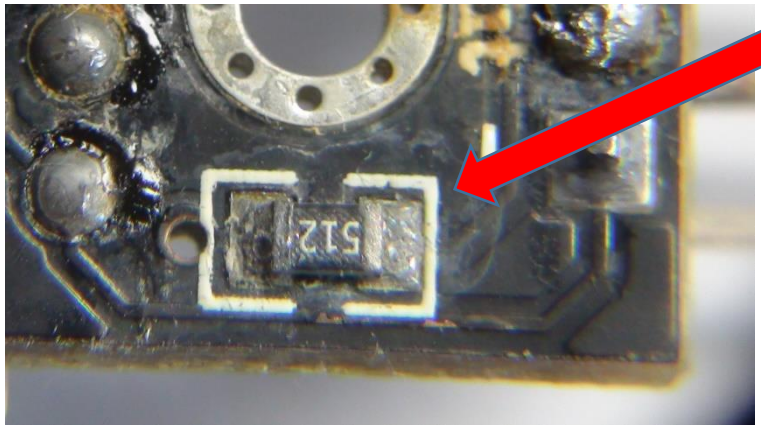


写真2　従来品　5.1KΩ

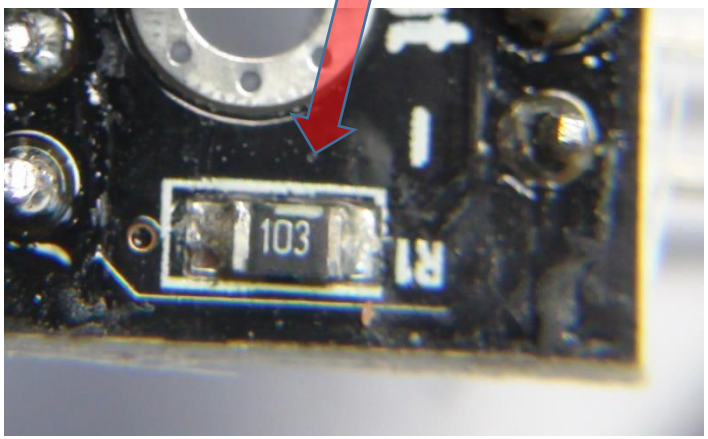
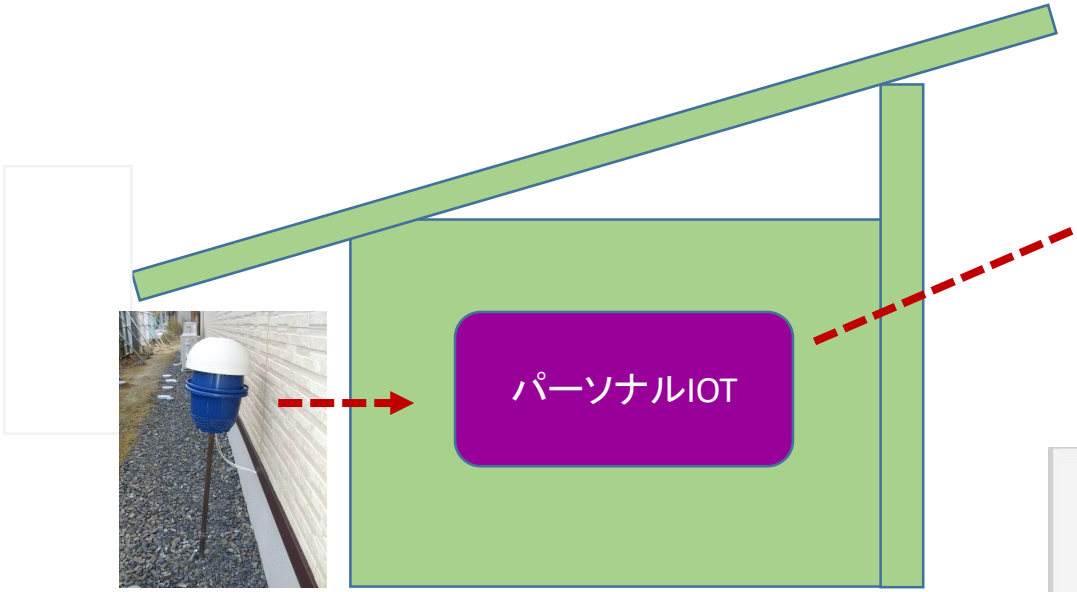
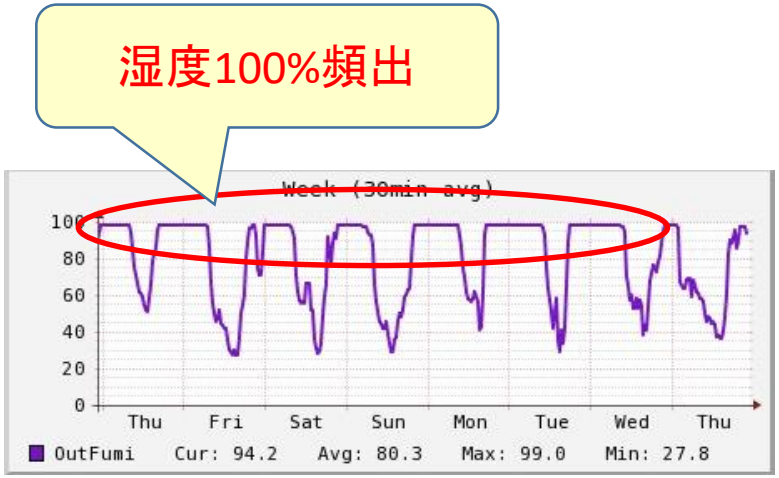


写真3　今回の交換品　10KΩ

⑥異常値検知(機械学習;Jubatus) 例2





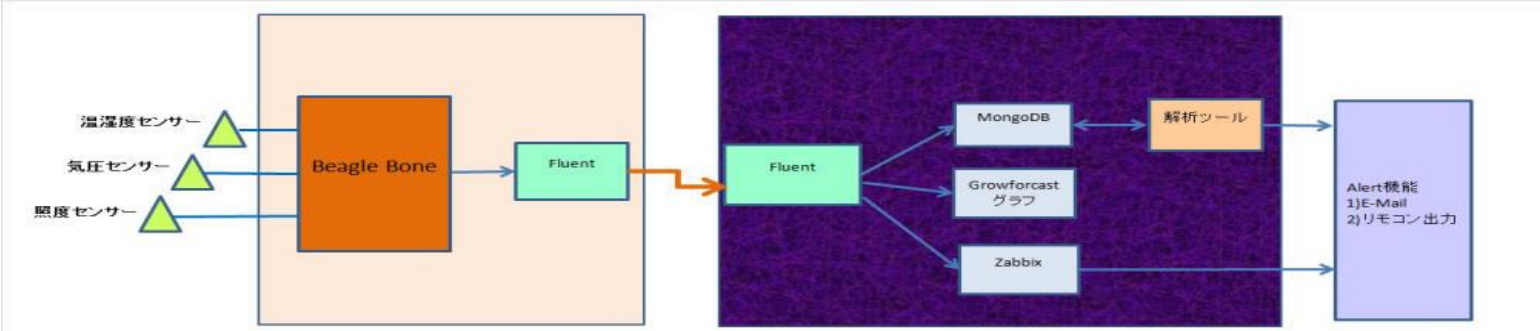
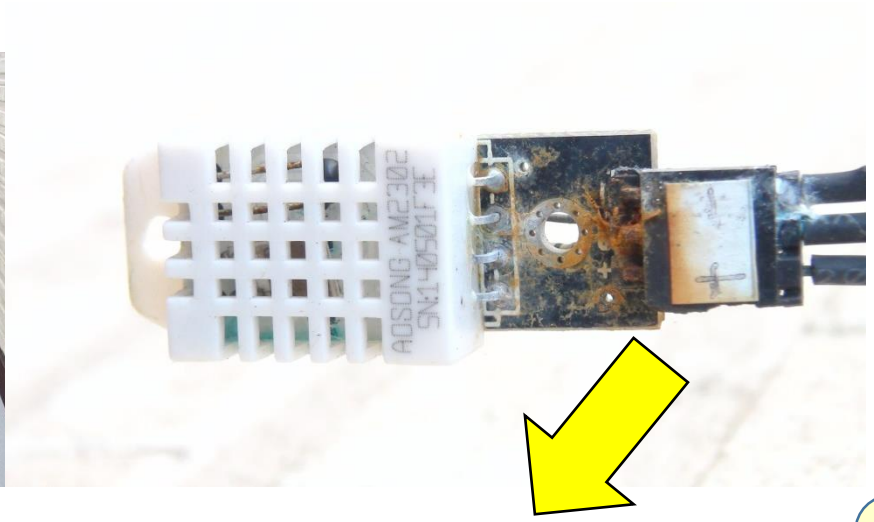
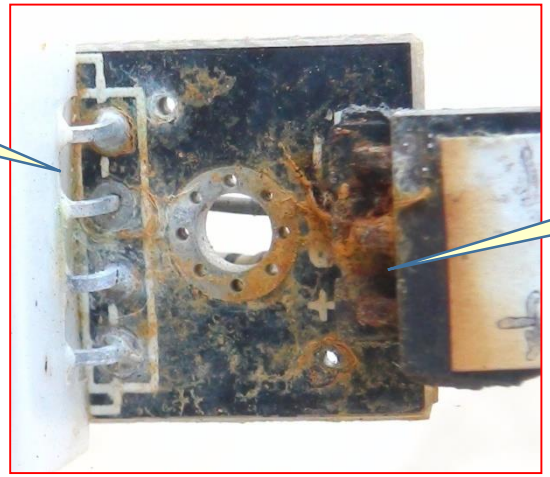


図1 オープンソースを活用したIOTシステム ダイアグラム

湿度データ異常の調査結果



センサー端子付近
の白化した腐食



ソケット端子付近は赤茶色の
ザビが発生

⑦JubatusをEdgeコンピュータで動かすには

センサー

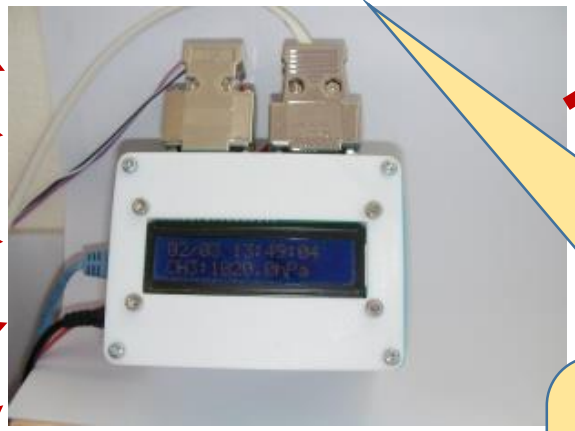
センサー

センサー

センサー

センサー

Edge コンピュータ
理想SPEC;64Bits



・異常検知判定
・統計情報まとめ
・生データー時保管

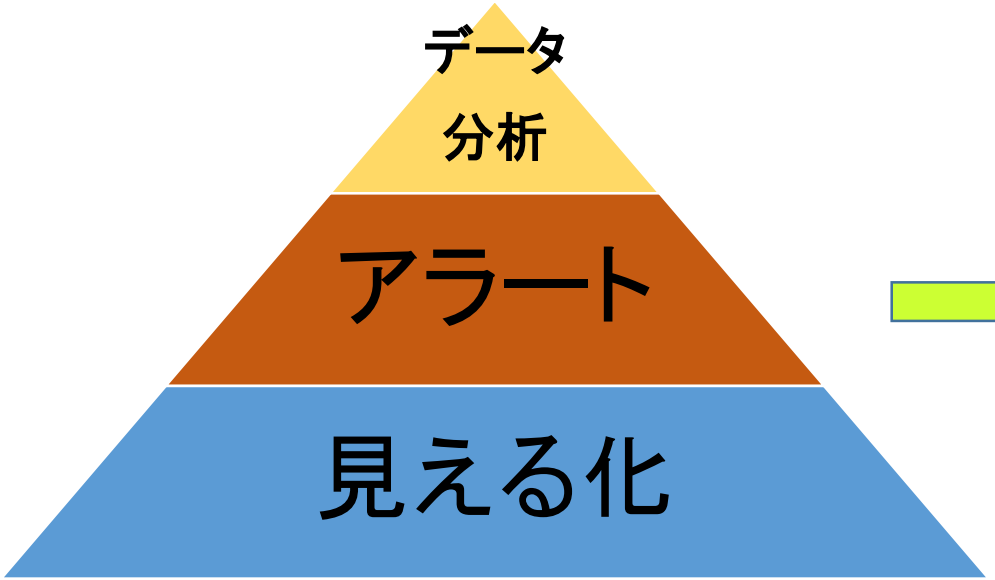
MaaS クラウド

・異常検知結果
・統計情報

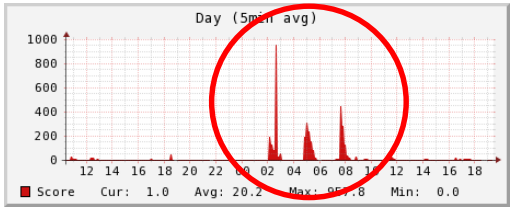
Jubatusは
センサーの近くで動かしたい

→ 32bitでは荷が重すぎる

⑧　実証実験で分かったこと



Jubatus(異常値検知)



1) アラートがあまりにも頻繁だと、慢性化する
2) データの組み合わせで複眼視が可能

- センサーは故障する
- 定期的にセンサーのセンシングをする
- IOTの環境を見てあげる

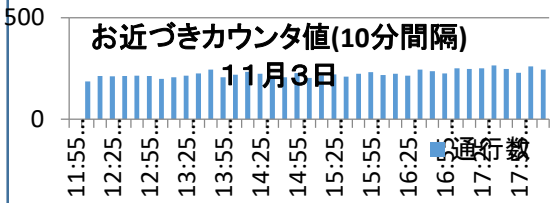
ロンドン・大英博物館　2015

5. 機械学習(Randomforest分析)

1. はじめに
2. 大量ログ時代
3. パーソナルIOTの勘どころ
4. 異常値検知　Jubatus実装
5. 機械学習(Randomforest分析）

①Randomforest（機械学習;)

通行量

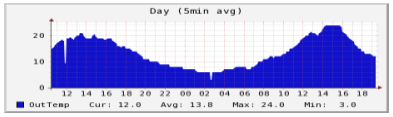


日付フォーマットの統一

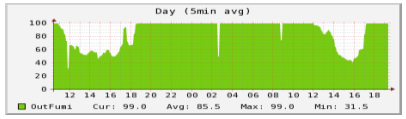
気象データ

Netatmo クラウド

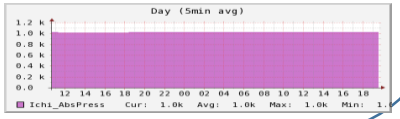
外気温度



外気湿度



気圧



日付フォーマットの統一

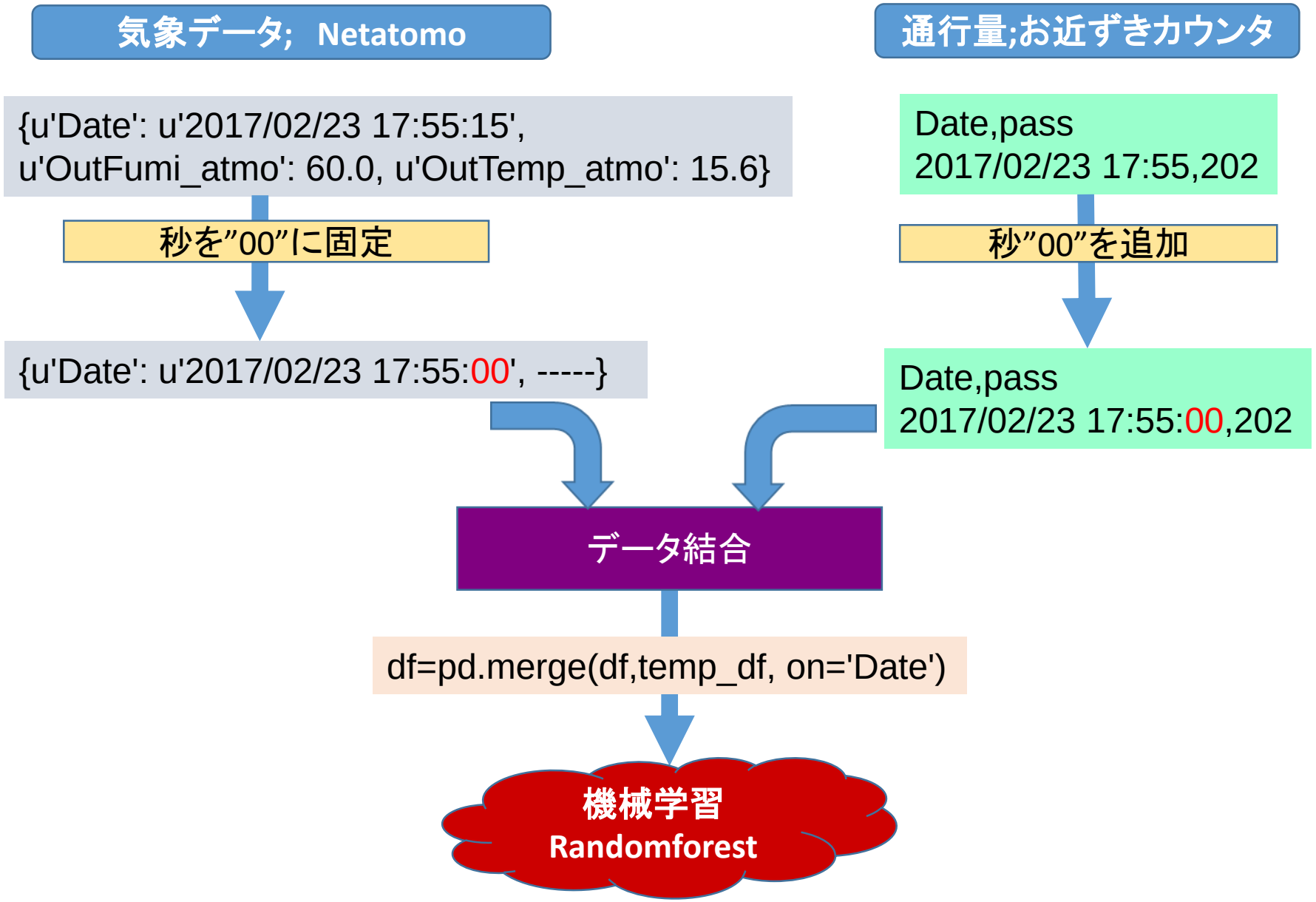
データ結合(Merge)

機械学習
Randomforest

通行量に
効いている
因子は？

気圧？、外気温度、湿度？

②データ結合はDatetimeの変換がポイント



Randomforestの特徴

学習対象データから多数の決定木を作成し、
それぞれの決定木の答えを多数決しクラスを決める

1)分類、回帰、予測が可能
　マイクロソフト「**Kinect**」に使われている

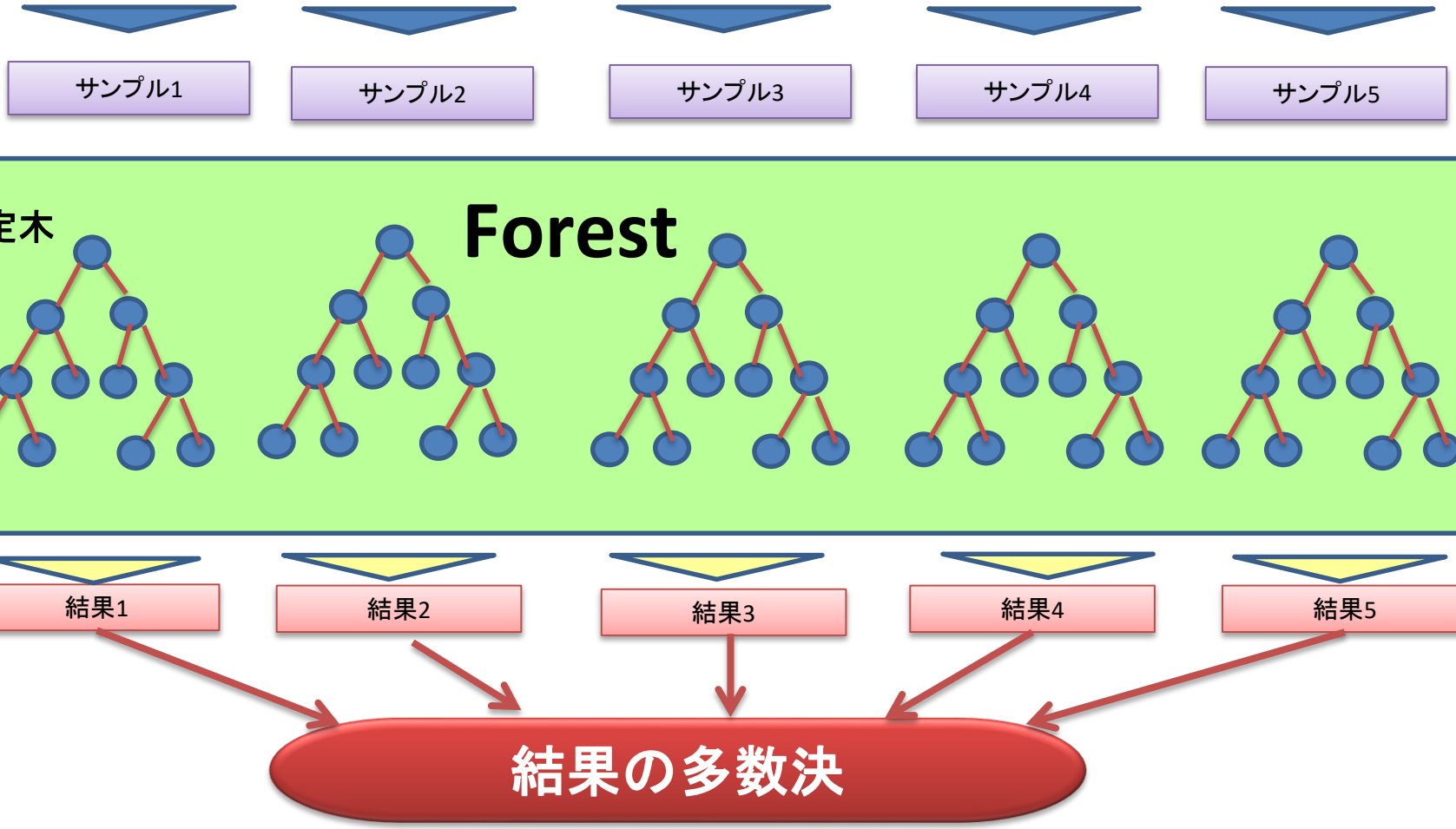
2)目的変数に対する、説明変数の重要度が出力可能

3)学習スピードが速い

4)過学習が起きにくい

③Randomforestの概要

NO	Date	Time	Ichi_AbsP	Ichi_Illum	OutFumi	OutTemp	Score
0	2016/10/19	11:10:01	1018	1273	99.9	17.1	3.98656
1	2016/10/19	11:20:02	1018	1281	99.9	17.4	3.46256
10	2016/10/19	12:25:02	1016	1432	92.8	20.4	1.19302
100	2016/10/19	21:30:01	1014	0	99.9	11.8	3.29488
101	2016/10/19	21:35:02	1014	0	99.9	11.9	3.11135
102	2016/10/19	21:40:02	1014	0	99.9	11.9	2.85117
103	2016/10/19	21:45:02	1014	0	99.9	12	2.63591



randomforest実行プログラム　python

```
#coding:UTF-8
import sys, json
import datetime
import os
from jubatus.anomaly import client
from jubatus.common import Datum
from get_netatmo import getmongo
import pandas as pd
import pandas.tseries.offsets as offsets
import numpy as np
from pandas.io.json import json_normalize
from sklearn.ensemble import RandomForestClassifier
from sklearn import preprocessing

NAME = "lux";

if __name__ == '__main__':

    # 1.Jubatus Serverへの接続設定
    anom = client.Anomaly("127.0.0.1",9199,NAME)

    # 2.学習用データの準備
    getmongo =  getmongo()
    dic = getmongo.getDic()
    elevations = json.dumps(dic)
    df=pd.read_json(elevations)
    temp_df=pd.read_csv("traffic.csv")
   df=df.T                    # 列、行の入れ替え
    pp = df["Date"]
    tmp = []
    tmp1 = []
 for i in range(len(pp)):
      tdatetime = datetime.datetime.strptime(pp[i], "%Y/%m/%d %H:%M:%S")#文字列から日付に変換
      tmp=tdatetime.replace(second=0)                    #秒を00に固定
      print tmp
      tstr = tmp.strftime('%Y/%m/%d %H:%M:%S')      #文字列への変換
      #df["Date"] = tmp    #曜日データの取得
      #df.loc[:, 'Date'] = tmp
      df['Date'].replace(pp[i],tstr,inplace=True)    #Dateを秒00に固定した値に入替する

print df.head(100)
df=pd.merge(df,temp_df, on='Date')
print df.head(100)
label_encoder = preprocessing.LabelEncoder()
pp = df["Date"]
tmp = []
tmp1 = []
for i in range(len(pp)):
    tdatetime = datetime.datetime.strptime(pp[i], "%Y/%m/%d %H:%M:%S")
    kdatetime = datetime.datetime.strptime(pp[i], "%Y/%m/%d %H:%M:%S")
    tdate = datetime.date(tdatetime.year, tdatetime.month, tdatetime.day)
    ttime = datetime.time(kdatetime.hour, kdatetime.minute, kdatetime.second)

    tmp.append(tdate.weekday())
    tmp1.append(ttime.hour)

df["weekday"] = tmp    #曜日データの取得
df["hour"] = tmp1      #時間データの取得
df = df.fillna(0)
forest = RandomForestClassifier(bootstrap=True, class_weight=None,criterion='gini',
        max_depth=10, max_features=3, max_leaf_nodes=None,
        min_samples_leaf=1, min_samples_split=3,
        min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=1,
        oob_score=False, random_state=0, verbose=0, warm_start=False)
features = df.columns[0:10]
Y=np.array(df["pass"], dtype="|S6")
del df['pass']
del df['Date']
features = df.columns[0:10]
X=df[features]
forest.fit(X,Y)
for feature, imp in zip(features, forest. feature_importances_):
        print(feature, imp)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_], axis=0)
indices = np.argsort(importances)[::-1]
print("Criterion variable:pass")
print("========================================")
print("Explanatory variable:Importance value")
for f in range(X.shape[1]):
        print("%d %s , %f" % (f + 1, df.columns[indices[f]],importances[indices[f]]))
print("========================================")
```

説明変数

目的変数

気圧

内部湿度

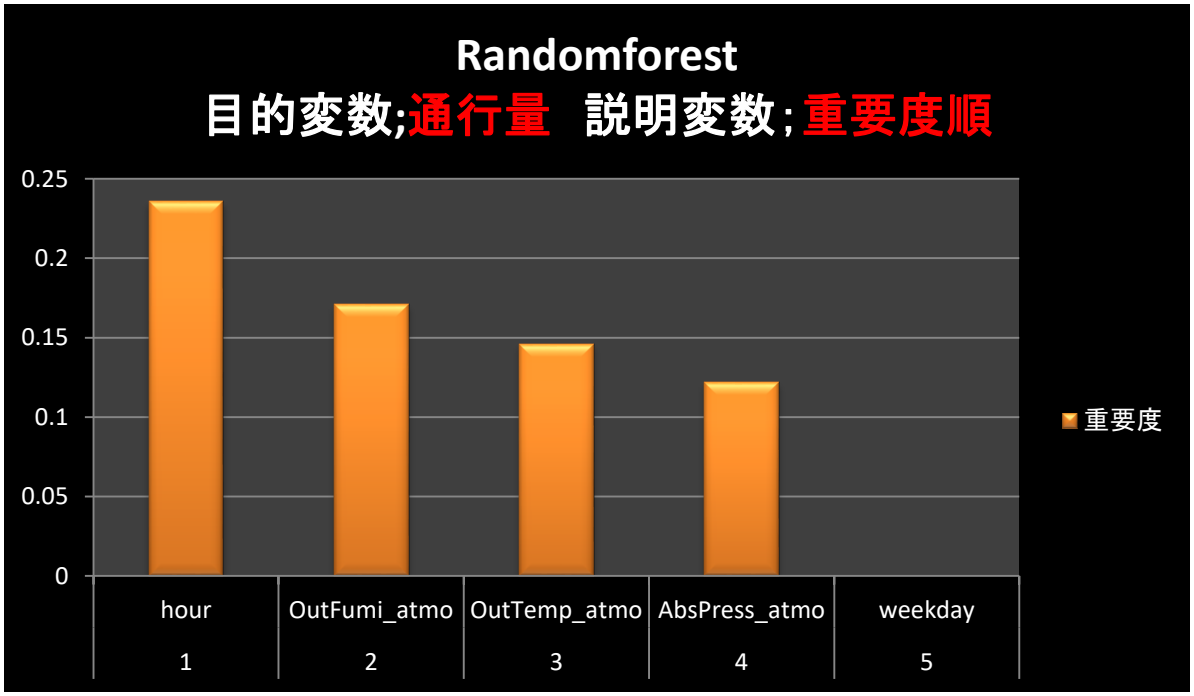
外部温度

外部湿度

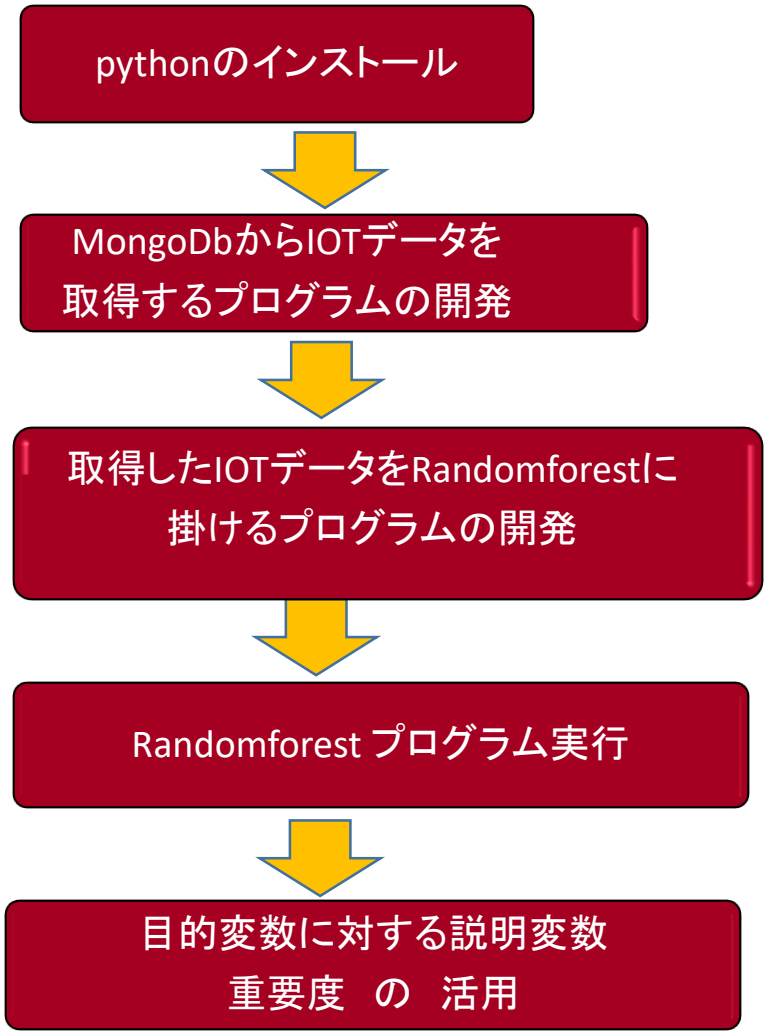
hour

weekday

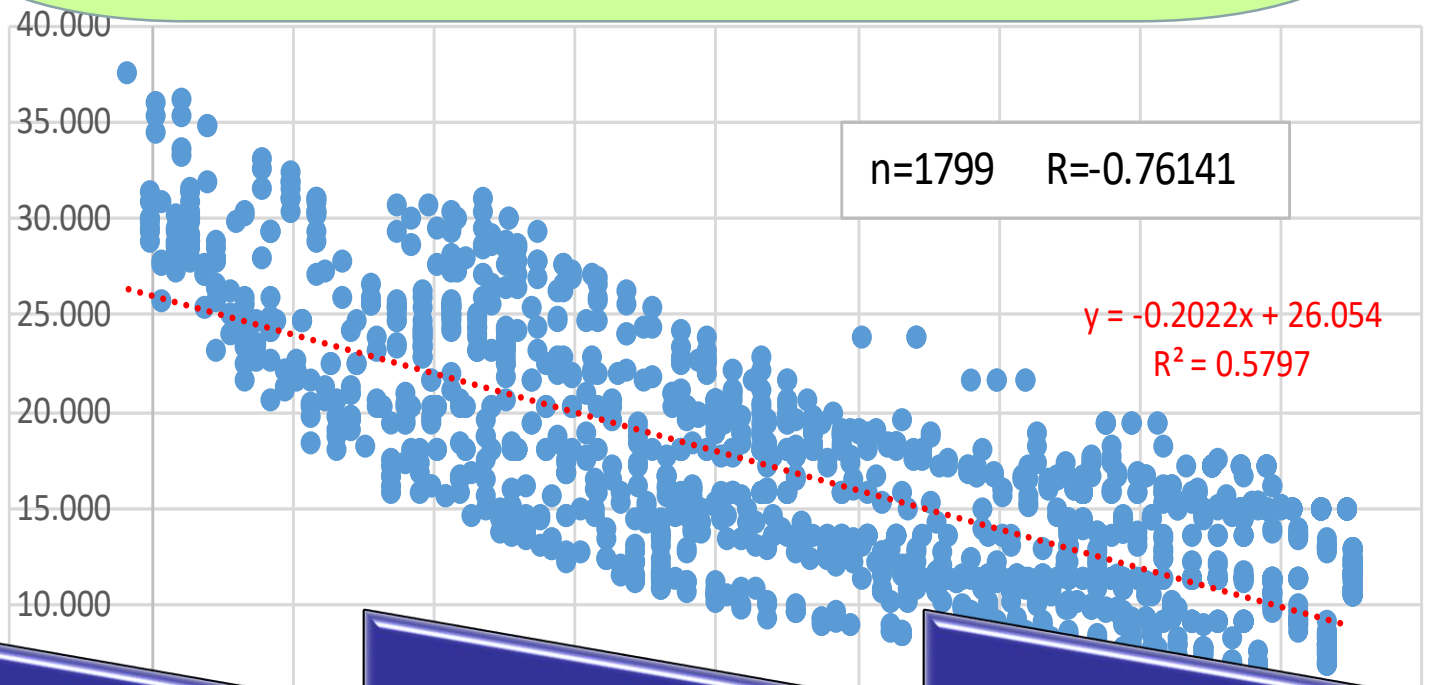
通行量



④Randomforestの導入手順



⑤従来型統計解析の問題



収集できる
データの膨大化

項目数が
高次元のデータ

データの膨大化
により
解析が困難

⑥データドリブン解析

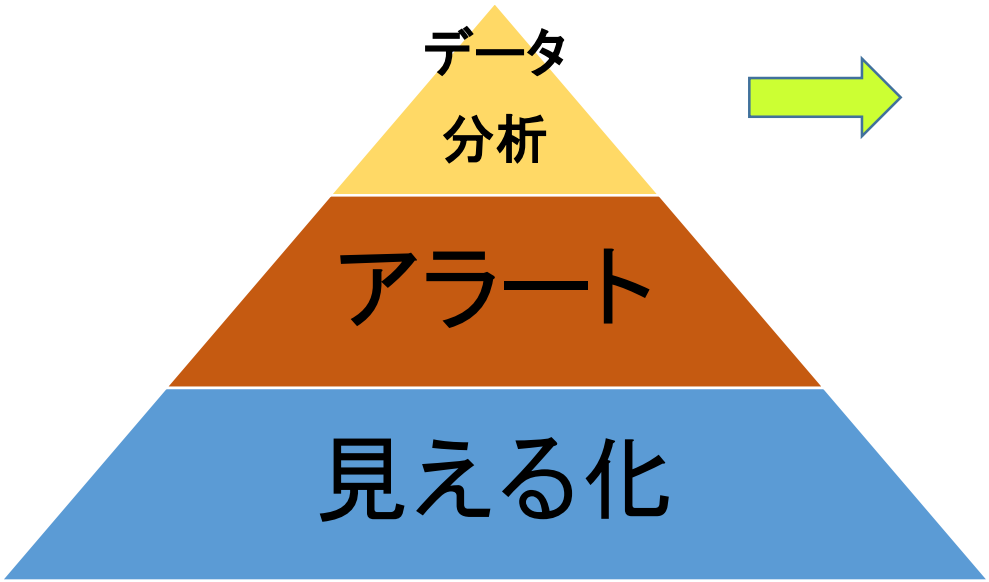
従来の統計解析

1. 仮説の設定
2. データ収集
3. 有意差検定、分散分析、実験計画
4. 目的変数に対する因子の確定
5. 具体的な対策の実施

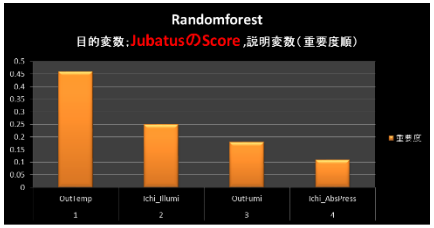
データドリブン解析

1. 仮説を最初に立てない
2. データ収集
3. 機械学習（決定木分析など）
4. 過学習の回避
5. 具体的な対策の実施

⑦実証実験で分かったこと



Randomforest



1) 目的の分析に対する適切な分析手法を見つけることがポイント
2) 分析結果を想定したデータのクレンジングが大事

- IOTから出されるログデータをどう活用するか？
- 他のログデータとの連携がカギ
- ログデータどおしの結合にはDatetimeが重要

- デジタル革命の大海原に
  Personal IOTの船を出し

- データという魚を取り

- OSSというレシピで
  オリジナル　シーフードを味わおう

沖縄･由布島　2017

ご静聴　ありがとうございました。