機械学習のテスト自動化コトハジメ

2014.6.6 Machine Learning Casual Talks #1
at COOKPAD

@komiya_atsushi


TGIF
Thank God, it's Friday.

「お前誰よ？」

略して

「おまだれ」


KOMIYA Atsushi
@komiya_atsushi
よろずやエンジニアリングしてます。 #TokyoWebmining 事務局
Shibuya, Tokyo · k11i.biz

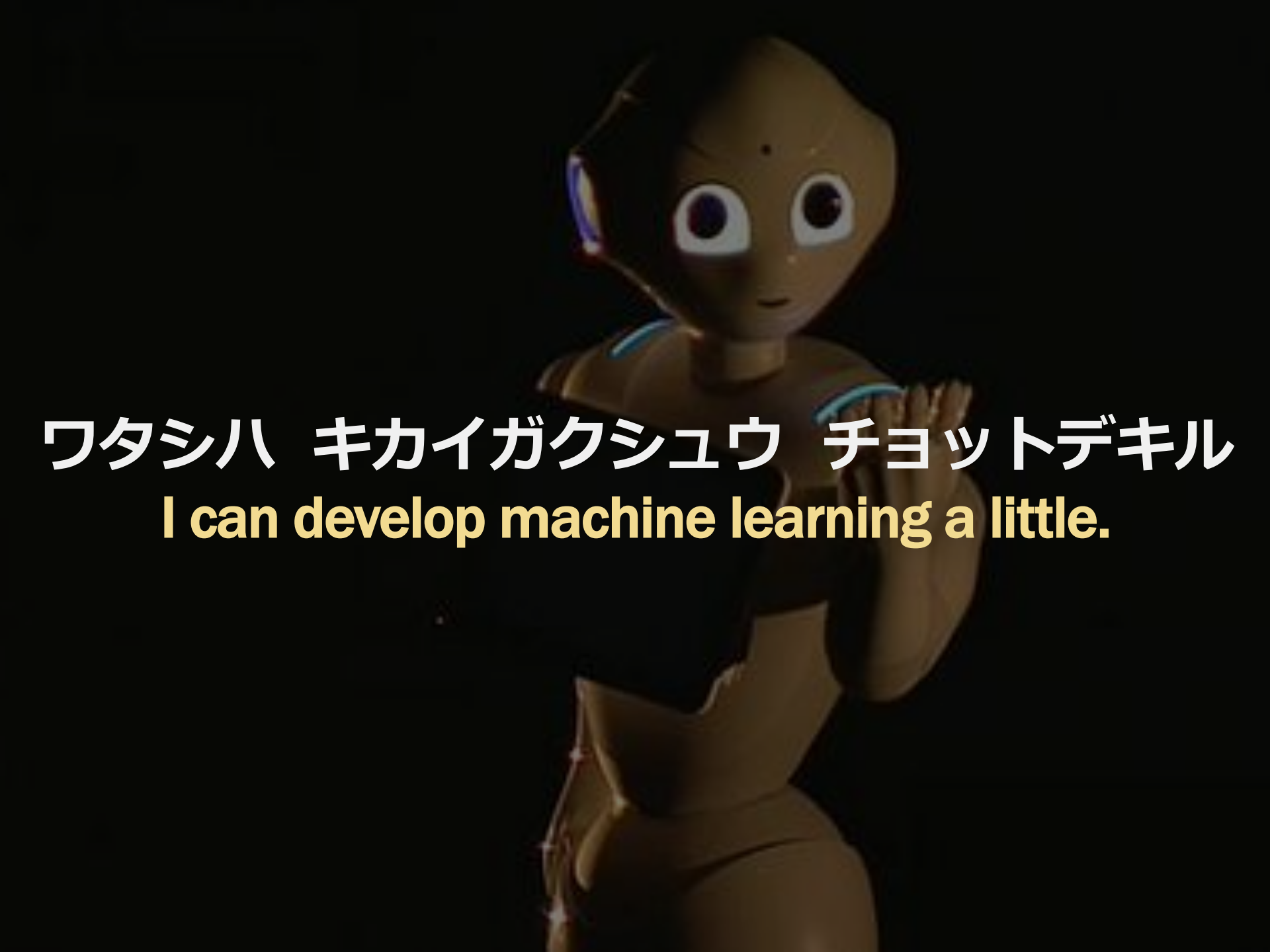
ALBERT Inc.

Analytical technology

Job:

Engineer

Machine Learning and me


ワタシハ キカイガクシュウ チョットデキル
I can develop machine learning a little.

Today’s topic

Test Automation

Code-driven testing (xUnit / xSpec)

+

Continuous integration

こちらに注目

Code-driven testing (xUnit / xSpec)

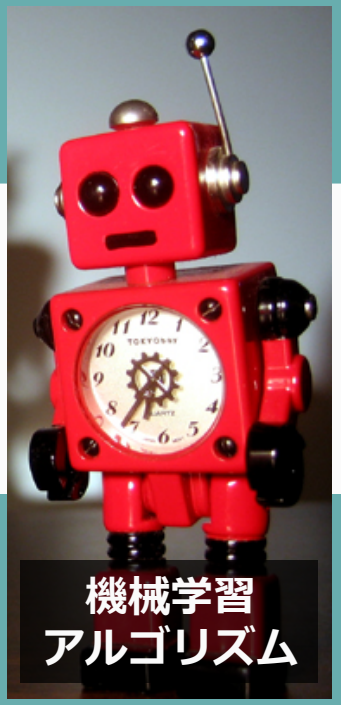
+

Continuous integration

Machine Learning

特徴量と正解情報の
組み合わせから
法則性を導き出す
（モデル化）

学習データ
（特徴量＆正解情報）



モデル

1. 学習フェーズ

正解が未知のデータ
（特徴量のみ）

モデルを元に
正解を推測する

分類・推定結果

2. 分類・推定フェーズ

Photo by littlelostrobot https://www.flickr.com/photos/littlelostrobot/215559356/

時間もあまりないし
説明はカジュアルに
割愛します

詳しくは PFI 比戸さんの資料がオススメ！

http://www.slideshare.net/shoheihido/cross-30115506/19

Why automated testing for machine learning ?

こんな経験
ありませんか？

機械学習アルゴリズムが
遅くて遅くて生きるのが辛い...

チューニングしてやったぜ！
何となく動作かくにん！ よかった♡

計算結果が全くおかしいことに
N ケ月後になってから発覚 ＼(^o^)／

※この話はフィクションです

機械学習アルゴリズムの精度を
上げるすんごいアイデア思いついた！

実装してみたら精度が向上した！
何となく動作かくにん！ よかった♡

実はコーナーケースなデータの
存在をまったく考慮できてなくて
本番環境で不慮の事故死...

※この話はフィクションです


本番環境で
事故を起こす奴は
!?
安全確認が
足りないんだよ

だからといって
Excel 方眼紙に書かれた
テスト項目を毎回消化するのも
バカらしい

機械学習の実装・利用に集中したい！

テストを
自動化しましょ！

悩みどころ

「期待する結果」の定義が難しい

機械学習の
精度は 100%
ではない

ランダムな
振る舞いをする
アルゴリズム

テストデータを作るのが辛い

どのような
入力データを
与えればよいか？

どのような
出力結果が
得られるのか？

テストケースが
NG となったときに
何がダメなのかが
分かりづらい

実装上の
不具合によって
NGとなったのか？

はたまた
入力データに
不手際があったのか？

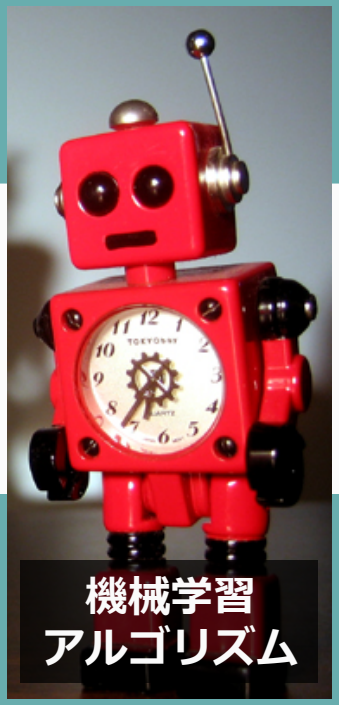
Software testing of Machine learning

機械学習への Code-driven testing 適用の基本

入力データと出力結果を意識する

特徴量と正解情報の
組み合わせから
法則性を導き出す
（モデル化）

学習データ
（特徴量＆正解情報）



モデル

1. 学習フェーズ

正解が未知のデータ
（特徴量のみ）

モデルを元に
正解を推測する

分類・推定結果

2. 分類・推定フェーズ

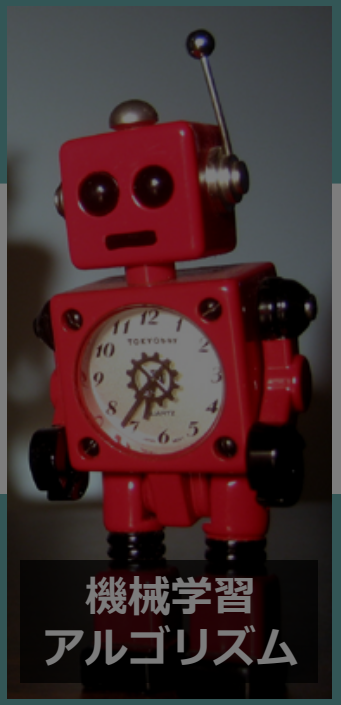
Photo by littlelostrobot https://www.flickr.com/photos/littlelostrobot/215559356/

特徴量と正解情報の
組み合わせから
法則性を導き出す
（モデル化）

学習データ
（特徴量＆正解情報）

1. 学習フェーズ

Input



モデル

Output

分類・推定結果

正解が未知のデータ
（特徴量のみ）

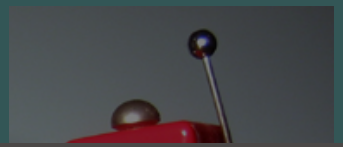
モデルを元に
正解を推測する

2. 分類・推定フェーズ

Photo by littlelostrobot https://www.flickr.com/photos/littlelostrobot/215559356/

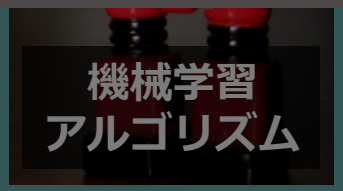
特徴量と正解情報の
組み合わせから
法則性を導き出す
（モデル化）

学習データ
（特徴量＆正解情報）



1. 学習フェーズ

入力データに対して、期待していた出力結果が得られているのか？



Output

分類・推定結果

正解が未知のデータ
（特徴量のみ）

モデルを元に
正解を推測する

2. 分類・推定フェーズ

Photo by littlelostrobot https://www.flickr.com/photos/littlelostrobot/215559356/

Black-box testing !

Patterns & Practices

※ おことわり

個人の経験より
得られた知見をもとに
お話をします
（これが正解、というわけ
ではありません）

テスト対象の分離・明確化

アプリケーション

ビジネスロジック

機械学習
アルゴリズム

アプリケーション

得られる精度
を検証する

ビジネスロジック

使い方の正しさ
を検証する

機械学習
アルゴリズム

実装の正しさ
を検証する

機械学習アルゴリズムに対するテスト

既存のライブラリを利用するならテストは不要

フルスクラッチ・独自実装する場合は必要

ビジネスロジックに対するテスト

機械学習に与える
入力データや出力結果の
取り扱いが複雑な場合に
実施すべき

アプリケーションに対するテスト

機械学習の結果の精度を定量評価できる仕組みが整っている場合に実現可能

テストデータの準備・作成

出力を人力計算できる小規模データを手で作る

Spark/MLlib: K-Means での例

```
@Test
public void runKMeansUsingStaticMethods() {
  List<Vector> points = Lists.newArrayList(
    Vectors.dense(1.0, 2.0, 6.0),
    Vectors.dense(1.0, 3.0, 0.0),
    Vectors.dense(1.0, 4.0, 6.0)
  );

  Vector expectedCenter = Vectors.dense(1.0, 3.0, 4.0);

  JavaRDD<Vector> data = sc.parallelize(points, 2);
  KMeansModel model = KMeans.train(data.rdd(), 1, 1, 1, KMeans.K_MEANS_PARALLEL());
  assertEquals(1, model.clusterCenters().length);
  assertEquals(expectedCenter, model.clusterCenters()[0]);

  model = KMeans.train(data.rdd(), 1, 1, 1, KMeans.RANDOM());
  assertEquals(expectedCenter, model.clusterCenters()[0]);
}
```

Spark/MLlib: K-Means での例

```
@Test
public void runKMeansUsingStaticMethods() {
  List<Vector> points = Lists.newArrayList(
    Vectors.dense(1.0, 2.0, 6.0),
    Vectors.dense(1.0, 3.0, 0.0),
    Vectors.dense(1.0, 4.0, 6.0)
  );

  Vector expectedCenter = Vectors.dense(1.0, 3.0, 4.0);

  JavaRDD<Vector> data = sc.p
  KMeansModel model = KMeans.                    K_MEANS_PARALLEL());
  assertEquals(1, model.clust
  assertEquals(expectedCenter

  model = KMeans.train(data.rdd(), 1, 1, 1, KMeans.RANDOM());
  assertEquals(expectedCenter, model.clusterCenters()[0]);
}
```

このテストデータに対して

クラスタの中心はこの値になる

擬似データを
自動生成する

MLlib: Logistic regression での例

```
// Generate input of the form Y = logistic(offset + scale*X)
def generateLogisticInput(
    offset: Double,
    scale: Double,
    nPoints: Int,
    seed: Int): Seq[LabeledPoint]  = {
  val rnd = new Random(seed)
  val x1 = Array.fill[Double](nPoints)(rnd.nextGaussian())

  val y = (0 until nPoints).map { i =>
    val p = 1.0 / (1.0 + math.exp(-(offset + scale * x1(i))))
    if (rnd.nextDouble() < p) 1.0 else 0.0
  }

  val testData = (0 until nPoints).map(i => LabeledPoint(y(i), Vectors.dense(Array(x1(i)))))
  testData
}
```

MLlib: Logistic regression での例

```
// Generate input of the form Y = logistic(offset + scale*X)
def generateLogisticInput(
    offset: Double,
    scale: Double,
    nPoints: Int,
    seed: Int): Seq[LabeledPoint]  =
  val rnd = new Random(seed)
  val x1 = Array.fill[Double](nPoints)(rnd.nextGaussian())

  val y = (0 until nPoints).map { i =>
    val p = 1.0 / (1.0 + math.exp(-(offset + sc
    if (rnd.nextDouble() < p) 1.0 else 0.0
  }

  val testData = (0 until nPoints).map(i => Lab                    ay(x1(i)))))
  testData
}
```

正規分布に従った乱数を生成し

条件に従ってラベル付け

これらを実践するためには、
各機械学習アルゴリズムに対する
本質的な理解が求められる

既存の枯れた実装を利用して生成する

入力として与える
データだけを準備
すればよい

ライブラリ:

libsvm, liblinear, SciPy, OpenCV

フレームワーク・ソフトウェア:

Mahout, Jubatus, R

テスト技法

モック

ビジネスロジックの
テストに集中したいが、

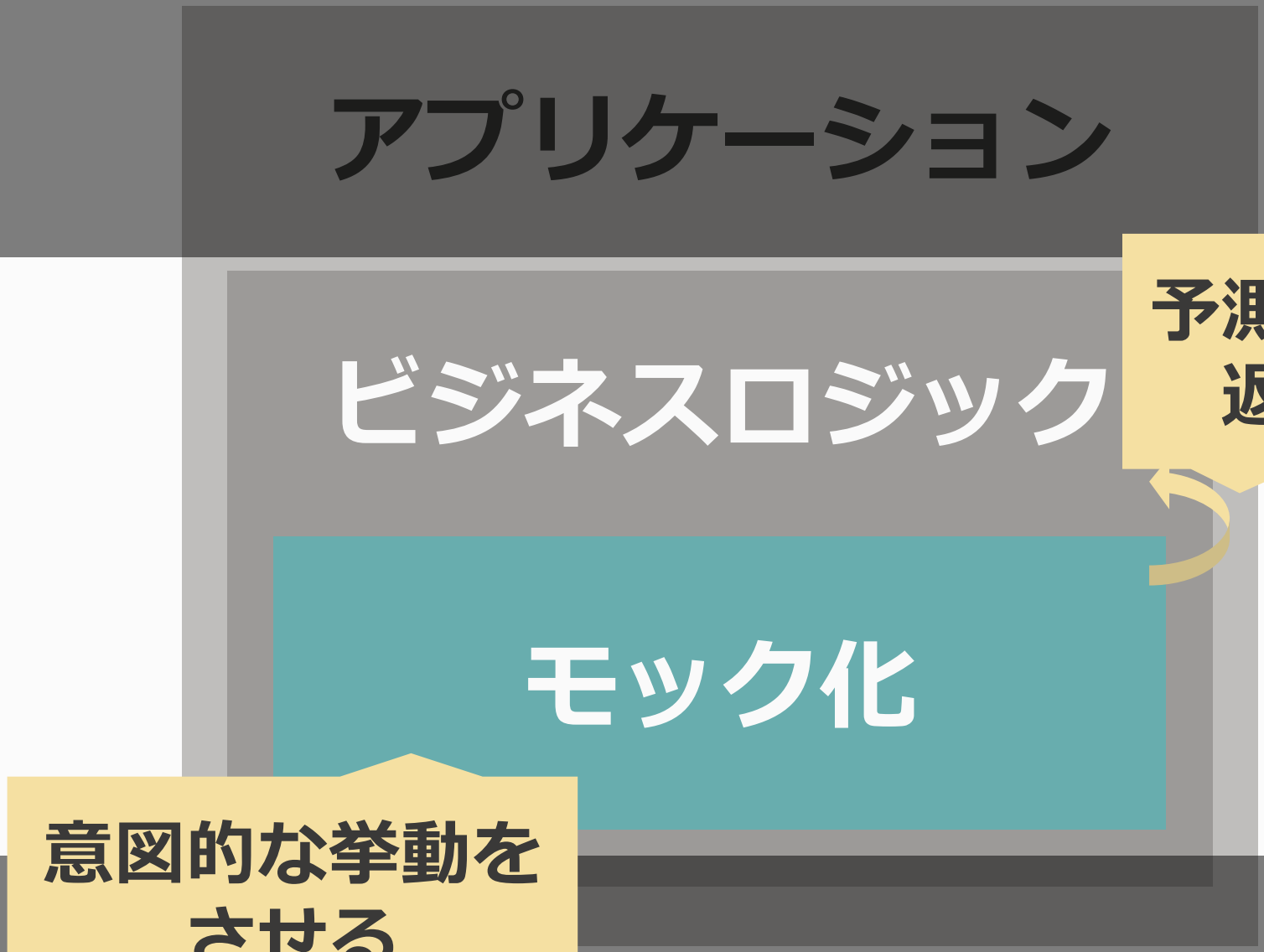
ビジネスロジック

機械学習
アルゴリズム

アプリケーション

機械学習アルゴリズムの
予測しづらい挙動が
悩ましい・・・

機械学習
アルゴリズム


アプリケーション
ビジネスロジック
モック化
予測可能な
返却値
意図的な挙動を
させる

フィクスチャ

アプリケーシ

ビジネスロジック

機械学習
アルゴリズム

テストケースごとに
モデルファイルを
用意・差し替える

```
svm_type c_svc
kernel_type rbf
gamma 0.001002
nr_class 3
total_sv 9
rho -0.000766337 0.003
label 0 1 2
nr_sv 3 3 3
SV
1 1 1:0.001 2:0.001 3:
1 1 1:0.001 2:0.014176
```

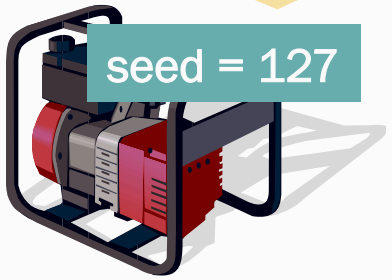
モデルファイル

アプリケーション

ビジネスロジック

機械学習
アルゴリズム

乱数シードを
固定する



乱数発生器

ホワイト
ボックス的観点

数値計算の結果が例外値になりうるケースを予測する

```
double v;

v = 1 / 0.0;
v = 0.0 / 0;
v = 1 + 0.000_000_000_000_000_09;

Math.exp(800);
Math.log(0);
Math.log(-1);
Math.sqrt(-1);
Math.pow(-1, 1.5);
```

```
double v;

v = 1 / 0.0;  // -> Infinity
v = 0.0 / 0;  // -> NaN
v = 1 + 0.000_000_000_000_000_09;  // 1.0

Math.exp(800);        // -> Infinity
Math.log(0);          // -> Neative infinity
Math.log(-1);         // -> NaN
Math.sqrt(-1);        // -> NaN
Math.pow(-1, 1.5);    // -> NaN
```

NaN

(negative) infinity

情報落ち

・・・と
そろそろいいお時間
ですのでこの辺で。

Conclusion


本番環境で
事故を起こす奴は
!?
安全確認が
足りないんだよ

そうならないようにするための自動テスト

ありがとう
ございました！