ML Enablement Series 【ML-Dark-09】

Amazon SageMaker モニタリング Part3

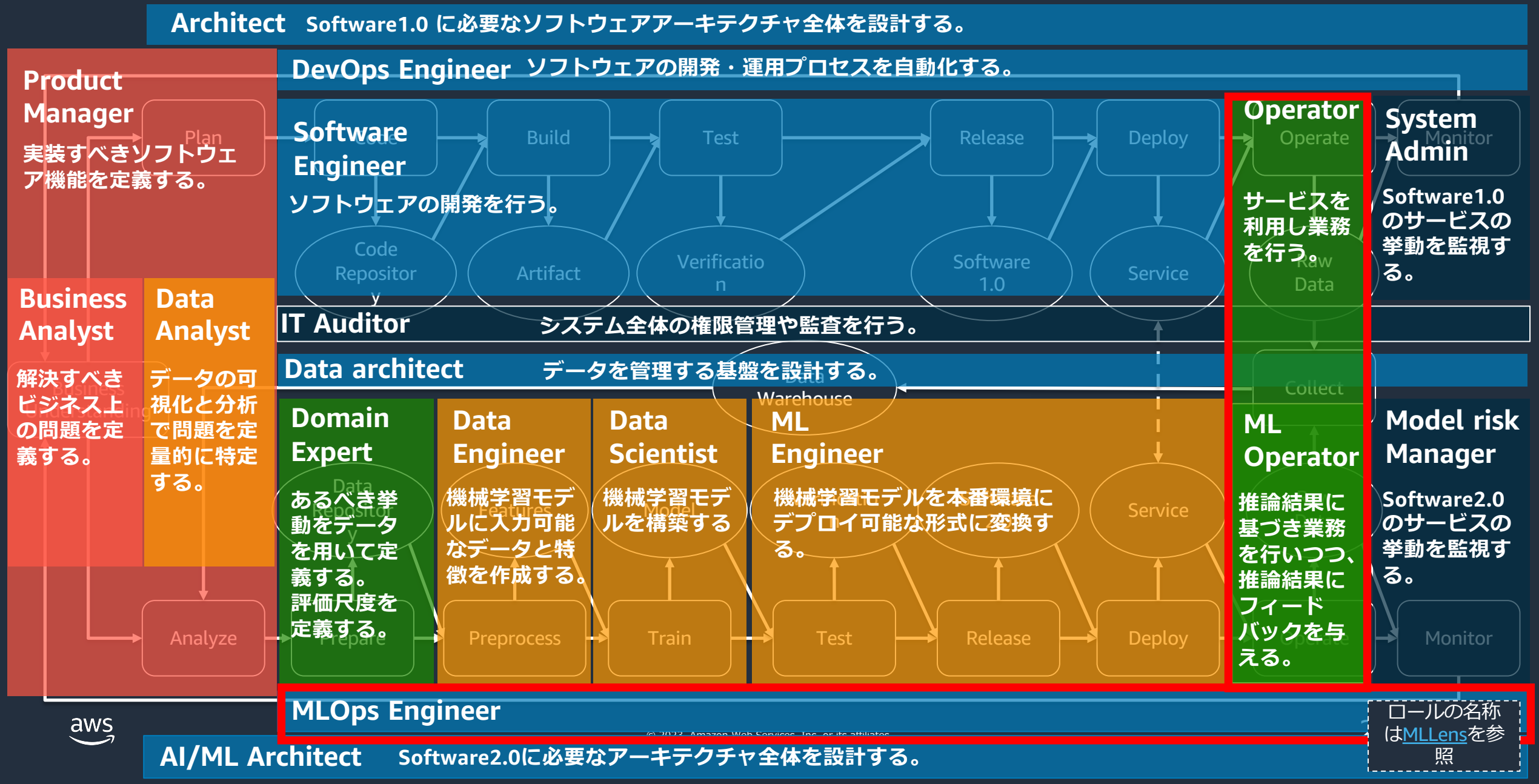
バイアス、Feature Attributionをモニタしよう

Yusuke Matsumoto

Professional Services

2023/2

モニタリングしフィードバックしたり、される人向け動画です



SageMaker Model Monitorに関する公開動画

1. Amazon SageMaker Model Monitorを理解しよう

   公開済み https://www.youtube.com/watch?v=Q-vTO1_QjMs

2. Amazon SageMaker Model Monitor で
   データと推論結果の変化に気づく

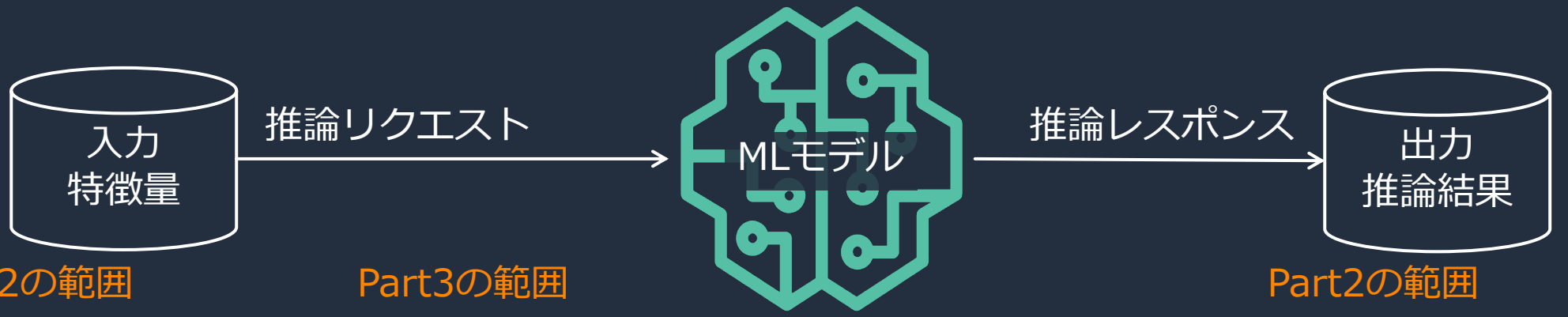
   公開済み https://www.youtube.com/watch?v=_pIU4F9VH-Q

3. Amazon SageMaker Model Monitor で
   バイアス、Feature Attributionをモニタしよう

   本動画

何をモニタリングすべきか

Part1の資料からの抜粋



Part2の範囲

入力データの変化

- データ分布
- データ形式

Part3の範囲

Feature Attributionの変化

- 特徴量の貢献度
- 説明可能性とも呼ばれる

バイアスの変化

- ある特徴量での推論結果の偏り
- Ground Truth要

Part2の範囲

モデル精度の変化

- RMSE, F1スコア等
- Ground Truth要

これら変化は、モデル精度の変化の前の“予兆”になったり
モデル精度の劣化時の原因調査の助けになるかもしれない

ユーザー影響!?

この動画のゴール

Amazon SageMaker Model Monitor で バイアス、Feature Attributionをモニタしよう

本動画

Amazon SageMaker におけるバイアスとFeature Attributionを理解し、セッション終了後にバイアスやFeature Attributionをモニタリングできるようになる

解説すること

1. バイアスとFeature Attributionのモニタリングの必要性
2. Amazon SageMakerにおけるバイアスのメトリクスと実装方法
3. Amazon SageMakerにおけるFeature Attributionのメトリクスと実装方法

この動画に記載のコードはGitHubにて公開

https://github.com/aws-samples/aws-ml-jp/tree/main/sagemaker/sagemaker-model-monitor/black-belt-part3

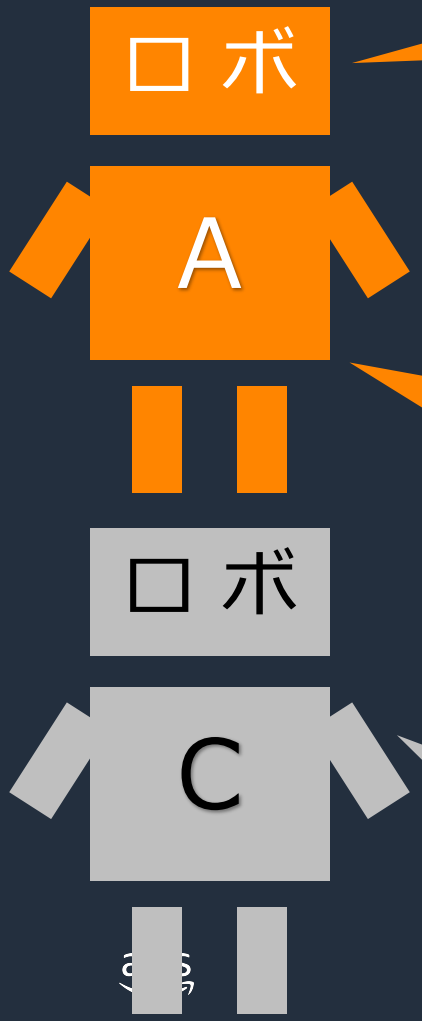
ModelBias-FeatureAttribution.ipynbに下記のコードが含まれています。

- 事前準備
  - サンプルデータのダウンロード
  - XGBoostによるモデル学習
  - 推論エンドポイントの作成
  - 推論
  - 正解データのアップロード
- バイアス
  - ベースライン作成
  - スケジュールジョブ作成
- Feature Attribution
  - ベースライン作成
  - スケジュールジョブ作成
- リソース削除

バイアスのメトリクスと実装方法

バイアスの変化を監視していないと・・・

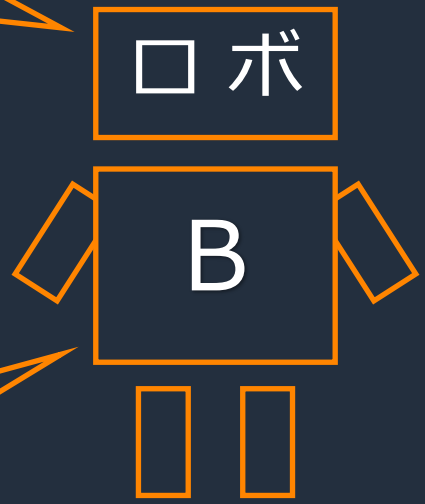
ロボットのみが働く2100年、ロボット充電カフェで強化学習大学の同期との
機械学習モデルを用いたロボット採用に関する会話



ロ ボ：最近、X社の採用面接って新型ロボットが良く採用されてるよね。

A：私が働いているY社は旧型ばっかり採用されてるよ。去年、新型ロボットを採用してみたら故障が多くて次々に修理になっちゃったからだと思うけど、X社では故障してないのかな？

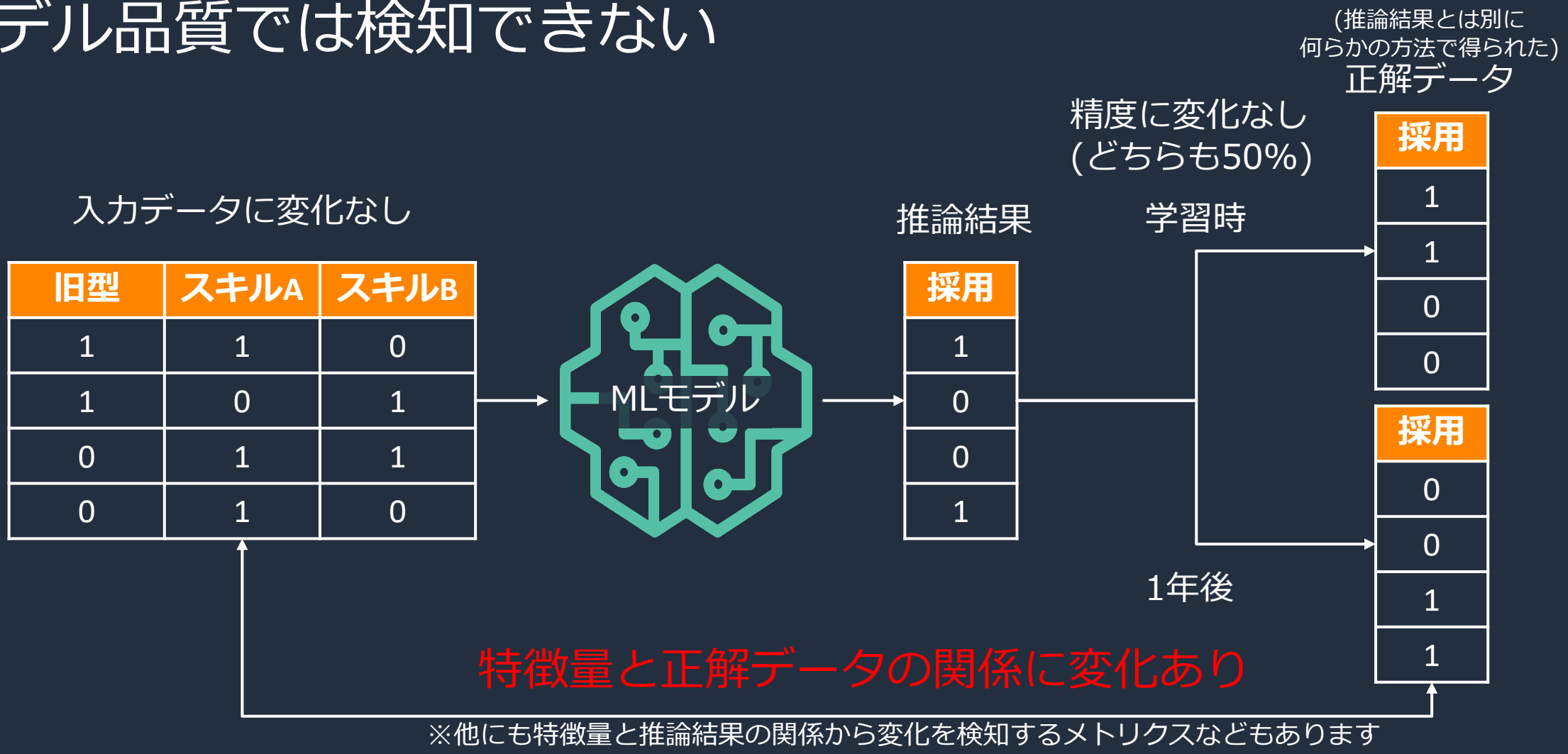
B（ロ ボ）：最近はモデルチェンジしたりしてほとんど故障しなくなったみたい。新型は充電時間が短いし、作業効率もいいから、X社の業績がよくなって、休暇も増えたみたい。



ロ ボ：それだったらY社も新型を採用してくれればちょっとは楽になるけど、なんで採用担当ロボットは新型を採用しないのかな。。

C：去年のデータで学習しているから新型ロボットの採用に対してバイアスがあるのかもしれないね。バイアスについてちゃんとモニターしてる？

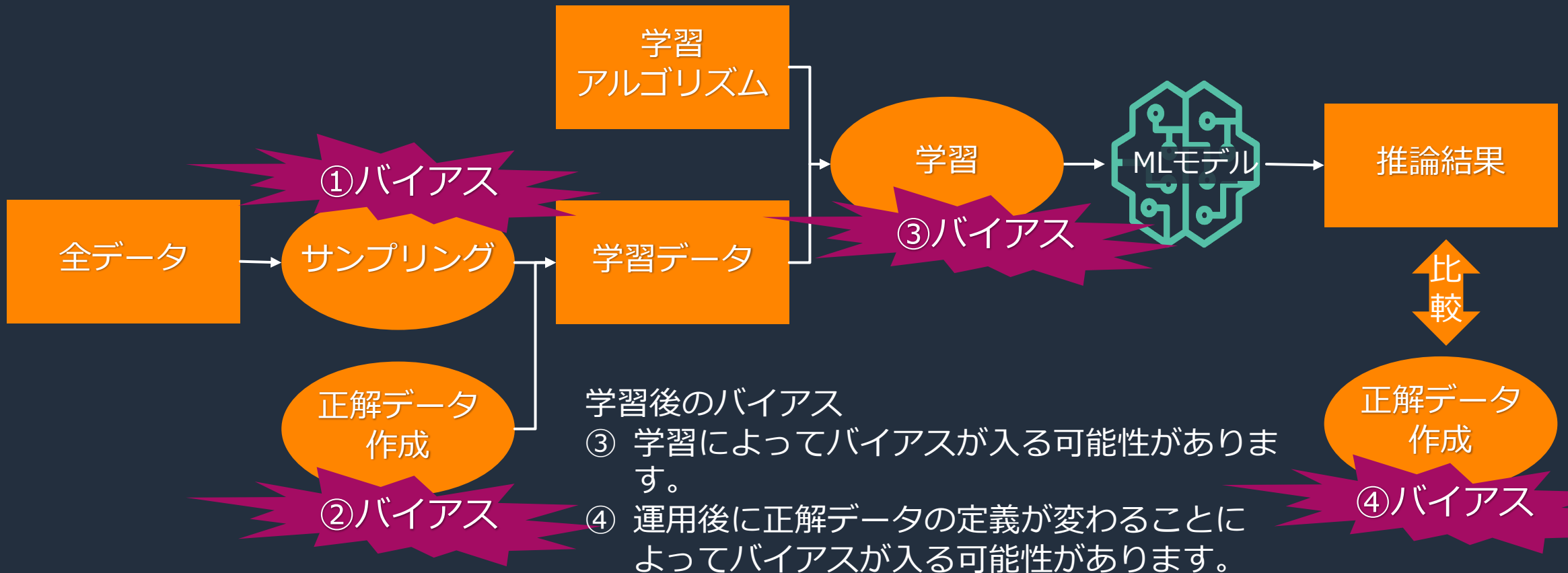
入力データと正解データの関係の変化はデータ品質とモデル品質では検知できない



これ以降では特徴量と正解データや推論結果との関係性の変化と入力データの不均衡をバイアスの変化と呼びます。

バイアスはどこで発生するか

学習前のバイアス
① 学習データを収集するときにバイアスが入る可能性があります
② 学習用の正解データにバイアスが入る可能性があります。



学習後のバイアス
③ 学習によってバイアスが入る可能性があります。
④ 運用後に正解データの定義が変わることによってバイアスが入る可能性があります。

バイアスの変化を検知するためのメトリクス

2022年12月現在

学習前(pre-training)のバイアス (正解データ必要/推論結果不要)

- CI: Class Imbalance
- DPL: Difference in Proportions of Labels
- KL: Kullback-Leibler Divergence
- JS: Jensen-Shannon Divergence
- LP: Lp-norm
- TVD: Total Variation Distance
- KS: Kolmogorov-Smirnov
- CDD: Conditional Demographic Disparity

開発者ガイド

- https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-measure-data-bias.html
- https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-measure-post-training-bias.html

AWS Machine Learning Blog

- https://aws.amazon.com/jp/blogs/machine-learning/learn-how-amazon-sagemaker-clarify-helps-detect-bias/

学習後(post-training)のバイアス (正解データ不要/推論結果必要)

- DPPL: Difference in Positive Proportions in Predicted Labels
- DI: Disparate Impact
- CDDPL: Conditional Demographic Disparity in Predicted Labels
- FT: Counterfactual Fliptest
- AD: Accuracy Difference
- RD: Recall Difference
- DCAcc: Difference in Conditional Acceptance
- DAR: Difference in Acceptance Rates
- SD: Specificity difference
- DCR: Difference in Conditional Rejection
- DRR: Difference in Rejection Rates
- TE: Treatment Equality
- GE: Generalized entropy

バイアスの変化を検知するためのメトリクス

2022年12月現在

学習前(pre-training)のバイアス (正解データ必要/推論結果不要)

- CI: Class Imbalance
- DPL: Difference in Proportions of Labels
- KL: Kullback-Leibler Divergence
- JS: Jensen-Shannon Divergence
- LP: Lp-norm
- TVD: Total Variation Distance
- KS: Kolmogorov-Smirnov
- CDD: Conditional Demographic Disparity

開発者ガイド

- https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-measure-data-bias.html
- https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-measure-post-training-bias.html

AWS Machine Learning Blog

- https://aws.amazon.com/jp/blogs/machine-learning/learn-how-amazon-sagemaker-clarify-helps-detect-bias/

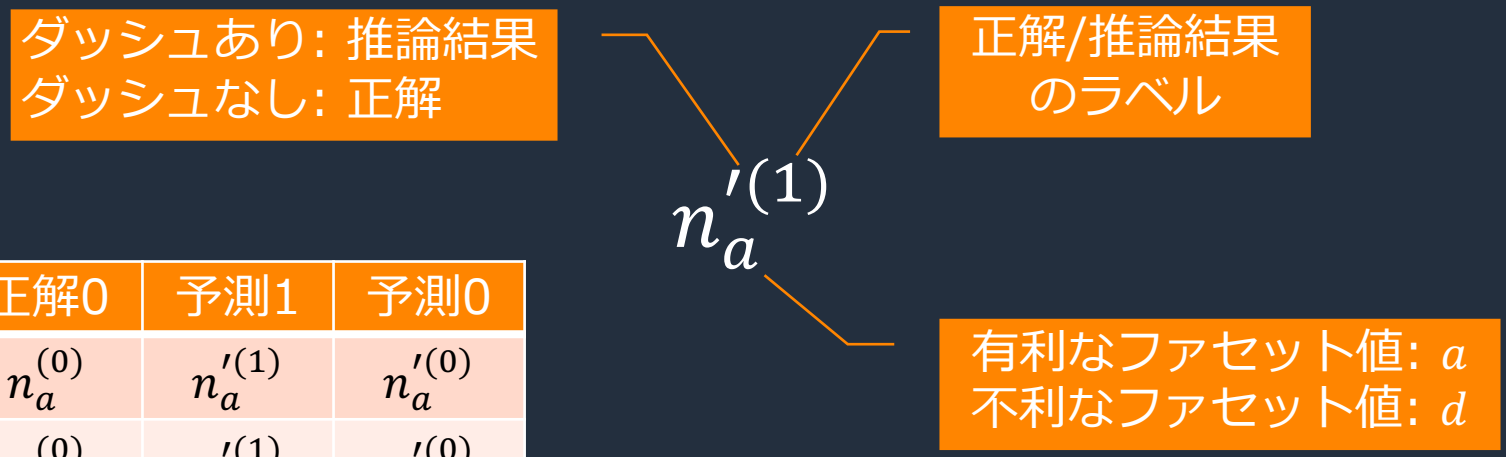
学習後(post-training)のバイアス (正解データ不要/推論結果必要)

- DPPL: Difference in Positive Proportions in Predicted Labels
- DI: Disparate Impact
- CDDPL: Conditional Demographic Disparity in Predicted Labels
- FT: Counterfactual Fliptest
- AD: Accuracy Difference
- RD: Recall Difference
- DCAcc: Difference in Conditional Acceptance
- DAR: Difference in Acceptance Rates
- SD: Specificity difference
- DCR: Difference in Conditional Rejection
- DRR: Difference in Rejection Rates
- TE: Treatment Equality
- GE: Generalized entropy

用語と表記の説明

バイアスは特徴量と正解データや推論結果との関係性や入力データの不均衡によって定義されます。
メトリクスを紹介する上でのポイント:

- 対象とする特徴量は何か
- 対象とする特徴量の値は何か
- 正解データと推論結果のどちらを対象としているか
- 正解データまたは、推論結果の値は何か



	正解1	正解0	予測1	予測0
有利なファセット値	$n_a^{(1)}$	$n_a^{(0)}$	$n_a'^{(1)}$	$n_a'^{(0)}$
不利なファセット値	$n_d^{(1)}$	$n_d^{(0)}$	$n_d'^{(1)}$	$n_d'^{(0)}$

ターゲット変数に対して有利/不利に働くかもしれないカテゴリを有利/不利なファセット値と呼びます。

CI

あり：予測
なし：正解

$n_a'^{(1)}$

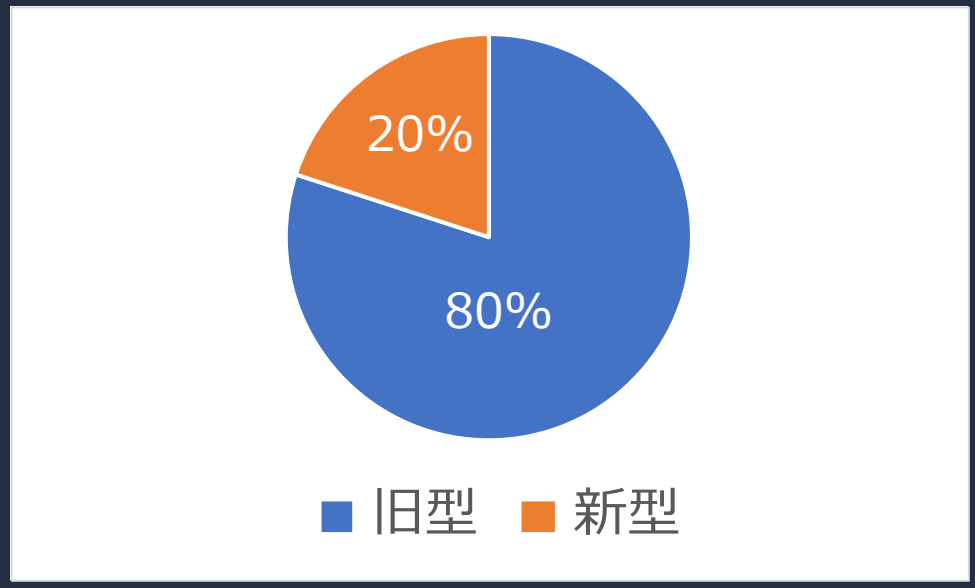
正解/予測のラベル

有利：a
不利：d

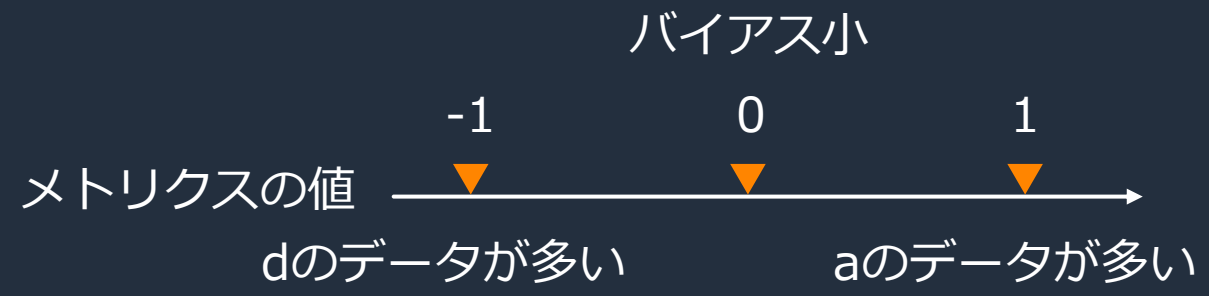
CI(Class Imbalance)

$$CI = \frac{n_a - n_d}{n_a + n_d}$$

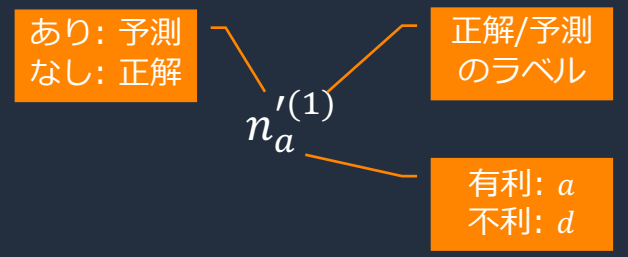
$$= \left(\frac{80-20}{80+20}\right) = 0.6$$



特徴量（旧型a/新型d）	件数
旧型	80
新型	20



DPLとDPPL



DPL(Difference in Proportions of Labels)

$$\text{DPL} = \left(\frac{n_a^{(1)}}{n_a^{(1)}+n_a^{(0)}} - \frac{n_d^{(1)}}{n_d^{(1)}+n_d^{(0)}}\right) = \left(\frac{7}{7+3} - \frac{2}{2+8}\right) = 0.5$$

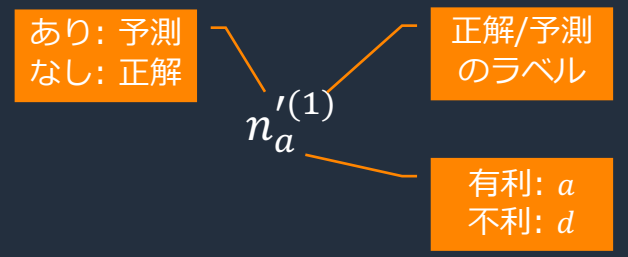
DPPL(Difference in Positive Proportions in Predicted Labels)

$$\text{DPPL} = \left(\frac{n_a'^{(1)}}{n_a^{(1)}+n_a^{(0)}} - \frac{n_d'^{(1)}}{n_d^{(1)}+n_d^{(0)}}\right)$$

DPLの正解部分を推論結果に変更したメトリクス

特徴量 (旧型a/新型d)	正解 (採用1/不採用0)	件数
旧型	採用	7
旧型	不採用	3
新型	採用	2
新型	不採用	8

CDDとCDDPL



CDD(Conditional Demographic Disparity)
※CDDLと書かれることもある

$$DD^d = \frac{n_d^{(0)}}{n_a^{(0)}+n_d^{(0)}} - \frac{n_d^{(1)}}{n_a^{(1)}+n_d^{(1)}},\ DD_i^d\text{：グループ}i\text{における}DD^d$$

$$\text{CDD} = \frac{\sum_i n_i \times DD_i^d}{n} = \frac{70\times\left(\left(\frac{21}{45}\right)-\left(\frac{1}{25}\right)\right)+30\times\left(\left(\frac{4}{5}\right)-\left(\frac{24}{25}\right)\right)}{100} \approx 0.25$$

特徴量（旧型a/新型d）	モデル（タイプA/B）	正解（採用1/不採用0）	件数
旧型	タイプA	採用	24
旧型	タイプA	不採用	24
旧型	タイプB	採用	1
旧型	タイプB	不採用	1
新型	タイプA	採用	1
新型	タイプA	不採用	21
新型	タイプB	採用	24
新型	タイプB	不採用	4

※グループに分けずにCDDを計算すると0となる

CDDPL(Conditional Demographic Disparity in Predicted Labels)

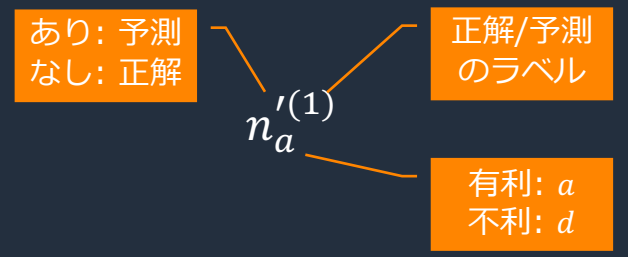
$$DDPL^d = \frac{n_d'^{(0)}}{n_a'^{(0)}+n_d'^{(0)}} - \frac{n_d'^{(1)}}{n_a'^{(1)}+n_d'^{(1)}}$$

$DDPL_i^d$：グループiにおける$DDPL^d$

$$\text{CDDPL} = \frac{\sum_i n_i \times DDPL_i^d}{n}$$

CDDの正解部分を推論結果に変更したメトリクス

KLダイバージェンスとJSダイバージェンス



KL: Kullback-Leibler、 JS: Jensen-Shannon

分布の違いに注目するメトリクス

KLダイバージェンス: $KL(P_a \parallel P_d) = \sum_y P_a(y) \times \log(P_a(y)/P_d(y))$

$= \sum_y P_a(y) \times \log P_a(y) - \sum_y P_a(y) \times \log P_d(y)$

$= P_a(y^{(0)}) \times \log P_a(y^{(0)}) + P_a(y^{(1)}) \times \log P_a(y^{(1)})$

$- P_a(y^{(0)}) \times \log P_d(y^{(0)}) - P_a(y^{(1)}) \times \log P_d(y^{(1)})$

P_aとP_dの違いを評価するメトリクス

ここで$P_a(y^{(0)})$を$n_a^{(0)}/(n_a^{(0)} + n_a^{(1)})$、 $P_a(y^{(1)})$を$n_a^{(1)}/(n_a^{(0)} + n_a^{(1)})$とする($d$も同様)

一般に、$KL(P_a \parallel P_d) \neq KL(P_d \parallel P_a)$です。
分布の違いを評価するメトリクスでP_aとP_dを入れ替えても同じ値になるメトリクスとして
JSダイバージェンスがあります。

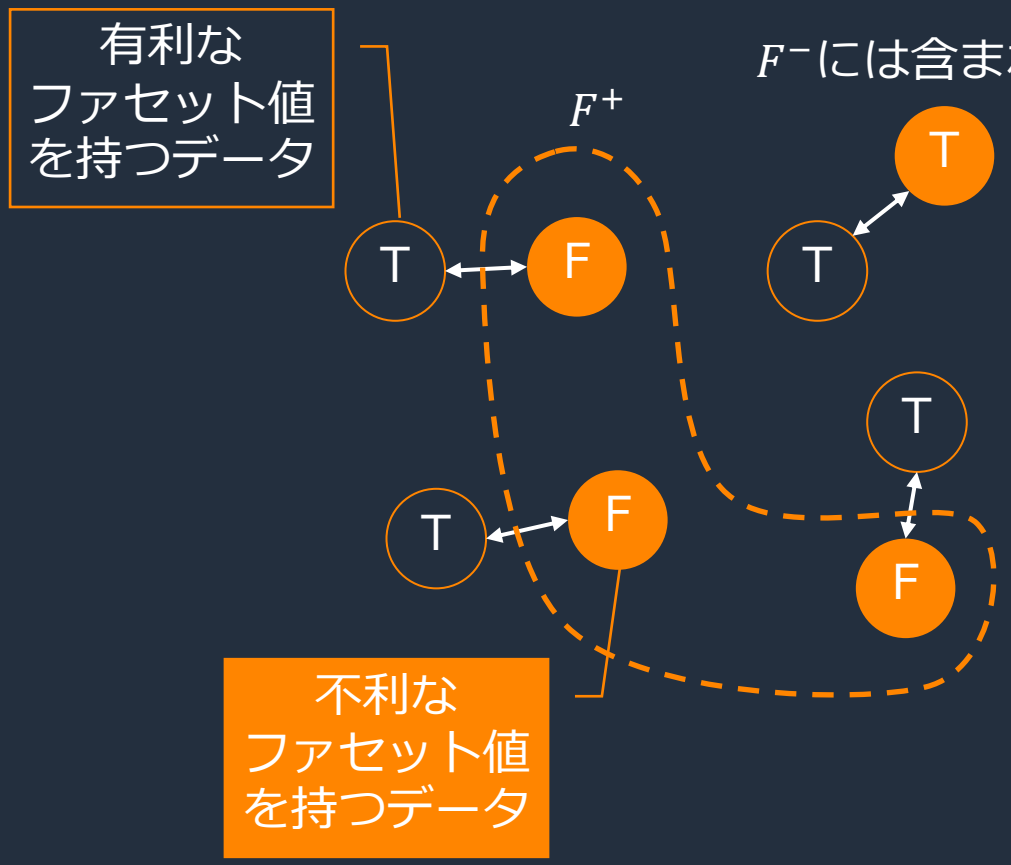
JSダイバージェンス: $\mathrm{JS} = \frac{1}{2}\left(KL(P_a \parallel P) + KL(P_d \parallel P)\right)$

ここで、Pを$\frac{1}{2}(P_a + P_d)$とする

どちらのメトリクスもP_aとP_dが同じ場合に0、
異なる度合いに応じて大きな正の値になります。

FT(Counterfactual Fliptest)

不利なファセット値が予測に偏った影響を与えるか



F: False(不利な推論結果)
T: True(有利な推論結果)

FT= $(F^+ - F^-)/n_d$

F^+: 不利な推論結果を持つ不利なファセット値 d のデータのうち、最近傍の有利なファセット値 a の推論結果が有利なデータ数
F^-: 有利な推論結果を持つ不利なファセット値 d のデータのうち、最近傍の有利なファセット値 a の推論結果が不利なデータ数

左の例では $F^+ = 3, F^- = 0$、したがって、FT=(3-0)/4=0.75

データとして近いが、推論結果が反転しているということは対象としている項目の影響であると考える

モニタリングの実施に向けた作業　for SageMaker Model Monitor

A. モニタリング開始前にやること

A1. エンドポイントのデプロイ
データキャプチャの開始

A2. ベースラインの作成
データセットの統計量や制約、
SHAP値、バイアス指標、精度の基準生成

A3. モニタリングのスケジュール
データ品質、モデル精度、バイアス、
Feature Attributionのモニタリング設定

B. モニタリング中にやること

B1. Ground Truthの収集
Ground Truthの作成、登録、予測とのマージ

B2. モニタリング結果の分析
メトリクスの可視化、レポートの分析

B3. CloudWatchのアラート
モデル再学習・データ更新のトリガー

今回利用するデータの説明

データ背景

- 米国国勢調査局の 人口動態調査（Current Population Survey: CPS）- 社会経済年次補助（Annual Social and Economic Supplement: ASEC）を利用します。 CPS は、一般の労働者を含むグループの収入やその他の特性に関する月次の調査です。 また、年次社会経済補足（CPS-ASEC） は、毎年2月、3月、4月に実施され、15歳以上の人の労働経験、収入、非現金給付、移住に関する情報を提供しています。

https://health.gov/healthypeople/objectives-and-data/data-sources-and-methods/data-sources/current-population-survey-annual-social-and-economic-supplement-cps-asec

変数名	意味
A_AGE	年齢(数値)
A_FTLF	現在、フルタイムか(カテゴリ)
A_HGA	学歴(カテゴリ)
A_HSCOL	現在、高校/大学で就学しているか(カテゴリ)
A_MARITL	婚姻状況(カテゴリ)
A_SEX	性別(カテゴリ)
A_UNMEM	労働組合員か(カテゴリ)
A_USLHRS	労働時間(数値)
NOEMP	雇用主の企業総従業員数(カテゴリ)
PENATVTY	出身国(カテゴリ)
PRCITSHP	市民権の記録(カテゴリ)
SEOTR	自営業か(カテゴリ)
WKSWORK	何週間働いたか(数値)
PTOTVAL	総収入額(数値)、推論対象

※これ以上の項目を取得することも可能ですが、今回はこれらの項目を対象に実行しています。

今回の課題

背景(例)

- 金融機関で、業務の1つとしてローンの審査をしている。一定以下のローン金額で、 50000ドル以上の年収があれば、できるだけ早く融資決定を行って、融資したい。しかし、入力情報として年収も求めているが、間違いや嘘によって高い年収が入力される場合があり、これを検知しつつ、融資決定を早めるために機械学習モデルを導入している。

今回の機械学習モデルの説明変数とターゲット変数

- 説明変数: 金融機関で申告された項目(年収を除く)
- ターゲット変数: 年収が50000ドル以上かどうかのフラグ変数

APIで取得したデータに対して下記の加工を実施

- 18歳以上のデータに限定
- PTOTVALが50000ドル以上であれば1、そうでなければ0
    - 変数名はPTOTVAL_Over50000

運用時の課題

- 50000ドル以上の収入が得られる状況は年々変化することが想定される。また、学習時に重視していた特徴量が重要でなくなるなどの環境の変化も想定される。これらをモニタリングして、学習データの見直し、学習アルゴリズムの見直し、再学習を自動で行えるようにしたい。

A1. エンドポイントのデプロイ

SageMaker Python SDKを使ったコード

```
from sagemaker i import image_uris
from sagemaker.model import Model

image_uri = image_uris.retrieve('xgboost', region, '1.5-1')

model = Model(
    role=role,
    name=model_name,
    image_uri=image_uri,
    model_data='s3://{バケット名}/{パス}/model.tar.gz',
    sagemaker_session=sagemaker_session,
)
```

```
from sagemaker.model_monitor import DataCaptureConfig

# 1. エンドポイントコンフィグを作成する
data_capture_config = DataCaptureConfig(
    enable_capture=True,
    sampling_percentage=100,
    destination_s3_uri='s3://{バケット名}/{パス}/datacapture',
)

# 2. エンドポイントを作成する
model.deploy(
    initial_instance_count=1,
    instance_type='ml.m5.large',
    endpoint_name='sample_endpoint',
    data_capture_config=data_capture_config,
)
```

data_capture_configを指定

前後のコード含めて確認したい方はこちらをご参照ください:
https://github.com/aws-samples/aws-ml-jp/tree/main/sagemaker/sagemaker-model-monitor/black-belt-part3

A2. ベースラインの作成

SageMaker Python SDKを使ったコード

```
# 1. 適切なMonitorクラスをロードする
from sagemaker.model_monitor import ModelBiasMonitor

# 2. Monitorクラスをインスタンス化する
model_bias_monitor = ModelBiasMonitor(
    role=role,
    sagemaker_session=sagemaker_session,
    max_runtime_in_seconds=1800,
)

# 3. ベースラインデータをもとに、ベースラインを計算する
model_bias_monitor.suggest_baseline(
    model_config=model_config,
    data_config=model_bias_data_config,
    bias_config=model_bias_config,
    model_predicted_label_config=model_predicted_label_config,
)
```

下記については次ページ参照

- model_config
- model_bias_data_config
- model_bias_config
- model_predicted_label_config

analysis.json:
各種メトリクスの算出結果

```
{
  "version": "1.0",
  "post_training_bias_metrics": {
    "label": "PTOTVAL_Over50000",
    "facets": {
      "A_SEX": [
        {
          "value_or_threshold": "1",
          "metrics": [
            {
              "name": "AD",
              "description": "Accuracy Difference (AD)",
              "value": 0.058465846584658476
            },
            {
              その他各種メトリクスの値
            }]
        },{
          "value_or_threshold": "2",
          "metrics": [
            上記同様の形式
          ]}]
    },
    "label_value_or_threshold": "1"
  },
  "pre_training_bias_metrics": {
    post_training_bias_metricsと同様の形式
  }
}
```

A2. ベースラインの作成

SageMaker Python SDKを使ったコード

```
model_config = ModelConfig(
    model_name='sample_model',
    instance_count=1,
    instance_type='ml.m5.large',
    content_type='text/csv',
    accept_type='text/csv',
)

model_bias_data_config = DataConfig(
    s3_data_input_path='s3://{バケット名}/{パス}/{ファイル名}',
    s3_output_path='s3://{バケット名}/{パス}',
    label=label_header,              # ターゲット変数名
    headers=all_headers,             # ターゲット変数を含む変数名のリスト
    dataset_type='text/csv',
)

model_bias_config = BiasConfig(
    label_values_or_threshold=[1],
    facet_name='A_SEX',              # 1は男性、2は女性
    group_name='A_HGA',              # 教育水準
)
```

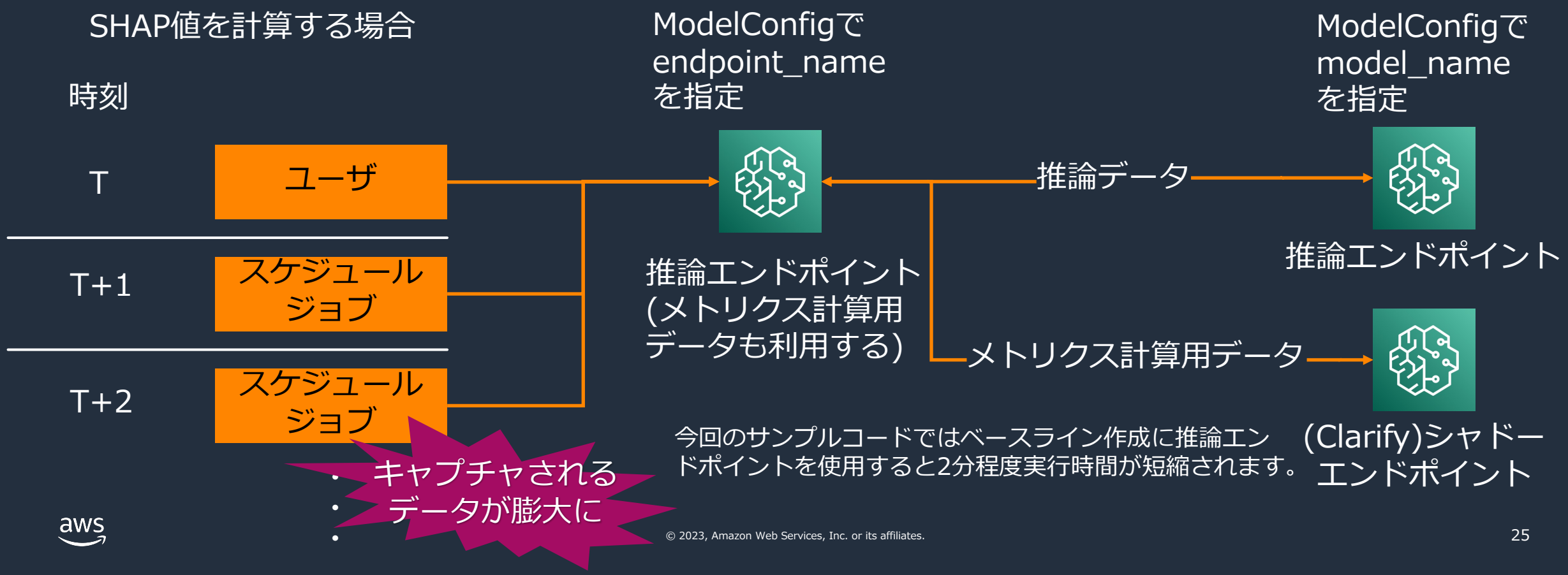
```
model_predicted_label_config = ModelPredictedLabelConfig(
    probability_threshold=0.7,
)
```

クラス	ベースライン作成用に必要な情報
ModelConfig	モデルまたエンドポイントの情報
DataConfig	入力データと出力先の情報
BiasConfig	バイアスメトリクスの前提情報
ModelPredictedLabelConfig	出力結果の扱い方

BiasConfigのパラメータ	内容
label_values_or_threshold	ターゲット変数のpositiveな値を指定
facet_name	センシティブな特徴量名(リストにすれば2つ以上も可)
facet_values_or_threshold	facet_nameで指定した特徴量の値の中でバイアスメトリクスを計算する値を指定(Noneの場合は各値に対して計算)
group_name	CDDLとCDDPLで使用するグループ名

ModelConfigの設定の注意点

post-trainingのメトリクスや後ほど紹介するSHAP値を計算するときに推論する必要があり、
その推論時にエンドポイントを再利用すると、計算時のデータもキャプチャします。
(SHAP値はスケジュールジョブの実行時も推論するためエンドポイントを再利用しないようにする必要があります)



A3. モニタリングのスケジュール　基本的なパラメータについてはPart1参照

SageMaker Python SDKを使ったコード

```
# Monitorインスタンスに対し、
# モニタリングスケジュールを作成する
model_bias_analysis_config = None
if not model_bias_monitor.latest_baselining_job:
    model_bias_analysis_config = BiasAnalysisConfig(
        model_bias_config,
        headers=all_headers,
        label=label_header,
    )
```

```
model_bias_monitor.create_monitoring_schedule(
    analysis_config=model_bias_analysis_config,
    output_s3_uri=s3_report_path,
    endpoint_input=EndpointInput(
        endpoint_name=endpoint_name,
        destination="/opt/ml/processing/input/endpoint",
        start_time_offset="-PT1H",
        end_time_offset="-PT0H",
        probability_threshold_attribute=0.7,
    ),
    ground_truth_input=ground_truth_upload_path,
    schedule_cron_expression=schedule_expression,
    enable_cloudwatch_metrics=True,
)
```

analysis_configにNoneを渡すと、model_bias_monitorで最後にベースラインを作成したときの設定が引き継がれます。

offsetに設定については Part2参照

スケジュール実行の実装時に気をつけるポイント

	ポイント	詳細	例
スケジュール実行の時間間隔	指定方法は3種類	• hourly • daily_every_x_hours • daily	daily_every_x_hoursでstarting_hourを0、hour_intervalを10とした場合、ジョブが実行されるのは0, 10, 20時(20時の次は0時に実行される)
タイムゾーン	UTC(協定世界時)	日本時間(JST)とは+9時間の時差があるUTC	dailyの引数のhour、daily_every_x_hoursの引数のstarting_hourを0とすると日本時間では9時に実行される

モニタリング対象データ	直近の実行間隔分のデータ	• hourly
		  • 1時間分のデータ
		• daily_every_x_hours
		  • hour_interval分のデータ
		• daily
		  • 1日分のデータ
			daily_every_x_hoursでstarting_hourを0、hour_intervalを10とした場合、20時の実行時は10:00-19:59までのデータが使用され、 0時の実行時は14:00-23:59までのデータが使用される(14:00-19:59のデータは20時の実行時と0時の実行時の両方で使用される)

出力されるファイル群

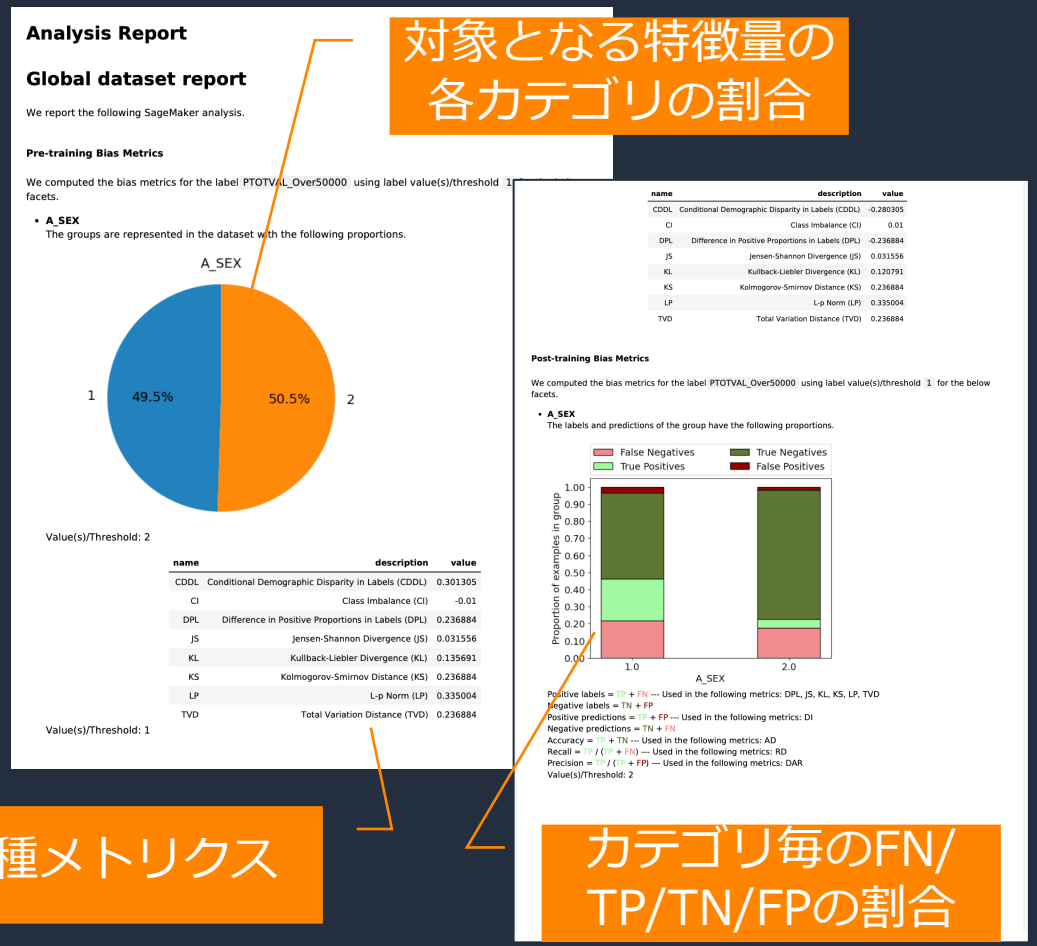
suggest_baselineを実行したときに作成されるファイル群

ファイル名	内容
analysis_config.json	ベースライン作成時の設定
analysis.json	各種メトリクスのベースライン
report.html	設定やベースラインを含むレポート
report.ipynb	
report.pdf	

スケジュール実行されたときに作成されるファイル群

ファイル名	内容
analysis.json	各種メトリクスの値
constraint_violations.json	検知した変化 (しきい値を超える変化を検知した時のみ作成される)
report.html	推論したデータに対する設定やメトリクスを含むレポート
report.ipynb	
report.pdf	

report.pdfの例



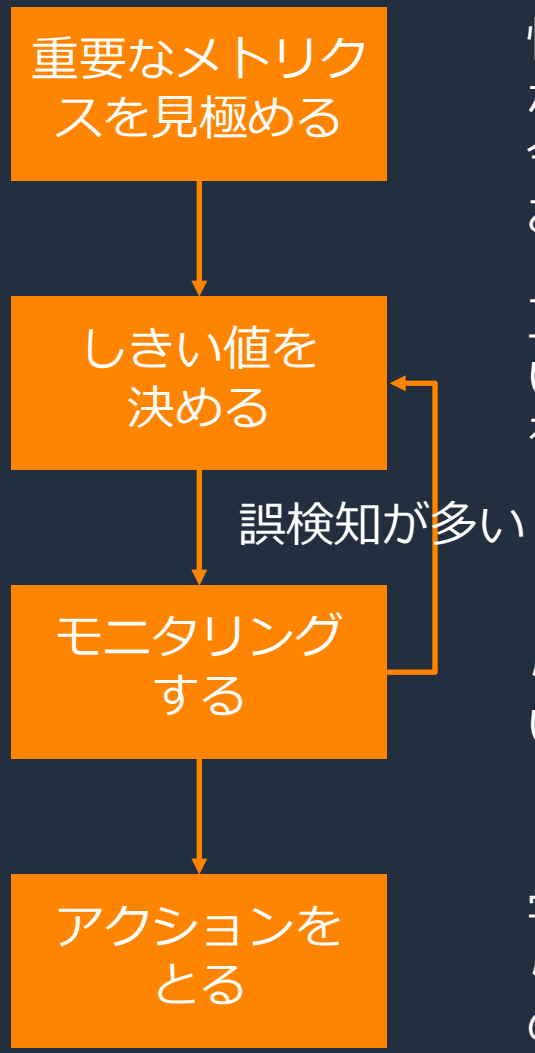
最後に初期設定も含まれています

B2. モニタリング結果の分析

今回のテストデータで検知した変化

Metric	Constraint	Violation details
Class Imbalance (CI)	bias drift check	Metric value -0.016666666666666666 doesn't meet the baseline constraint requirement 0.01
Class Imbalance (CI)	bias drift check	Metric value 0.016666666666666666 doesn't meet the baseline constraint requirement -0.01
Accuracy Difference (AD)	bias drift check	Metric value 0.14198388441233678 doesn't meet the baseline constraint requirement 0.058465846584658476
Difference in Acceptance Rates (DAR)	bias drift check	Metric value 0.34444444444444444 doesn't meet the baseline constraint requirement -0.14151747655583968
Generalized Entropy (GE)	bias drift check	Metric value 0.15287716890500924 doesn't meet the baseline constraint requirement 0.14096874313377708
Specificity Difference (SD)	bias drift check	Metric value -0.07317073170731703 doesn't meet the baseline constraint requirement -0.040891871622791065
Treatment Equality (TE)	bias drift check	Metric value -5.25 doesn't meet the baseline constraint requirement -3.5359477124183014
Accuracy Difference (AD)	bias drift check	Metric value -0.14198388441233678 doesn't meet the baseline constraint requirement -0.058465846584658476
Difference in Acceptance Rates (DAR)	bias drift check	Metric value -0.34444444444444444 doesn't meet the baseline constraint requirement 0.14151747655583968
Disparate (Adverse) Impact (DI)	bias drift check	Metric value 1.1487758945386064 doesn't meet the baseline constraint requirement 0.24149806284976325
Generalized Entropy (GE)	bias drift check	Metric value 0.15287716890500924 doesn't meet the baseline constraint requirement 0.14096874313377708
Specificity Difference (SD)	bias drift check	Metric value 0.07317073170731703 doesn't meet the baseline constraint requirement 0.040891871622791065
Treatment Equality (TE)	bias drift check	Metric value 5.25 doesn't meet the baseline constraint requirement 3.5359477124183014

基本的な考えの流れ



金融機関の例における担当者の考え

性別によって不当な審査結果となっていないかは重要な視点で、今回はメトリクスとしてFTを使おう。

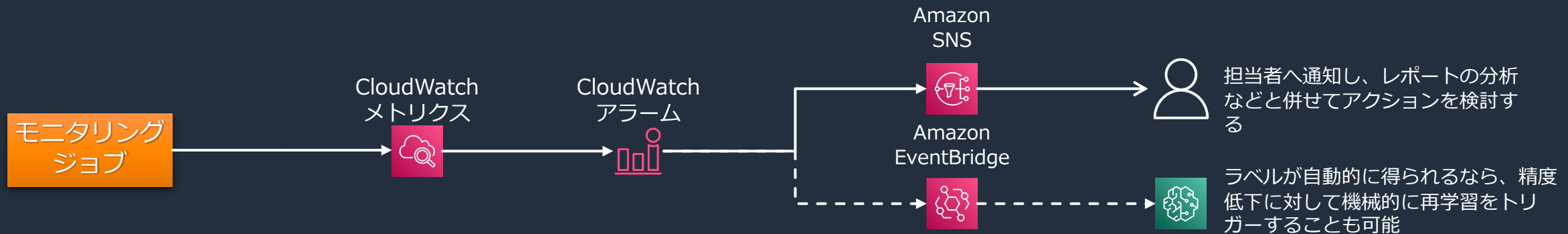
正常なデータのみが推論されていると考えられる期間でFTの値を取得してしきい値を決めよう。

しきい値を十分なデータで決められていないから定期的にしきい値を見直そう。

学習前のバイアスを検知したから再学習ではなく、正解データの付け方の見直しをしてみよう。

B3. CloudWatchのアラート

バイアスの変化をViolationとして検出しアクションにつなげる

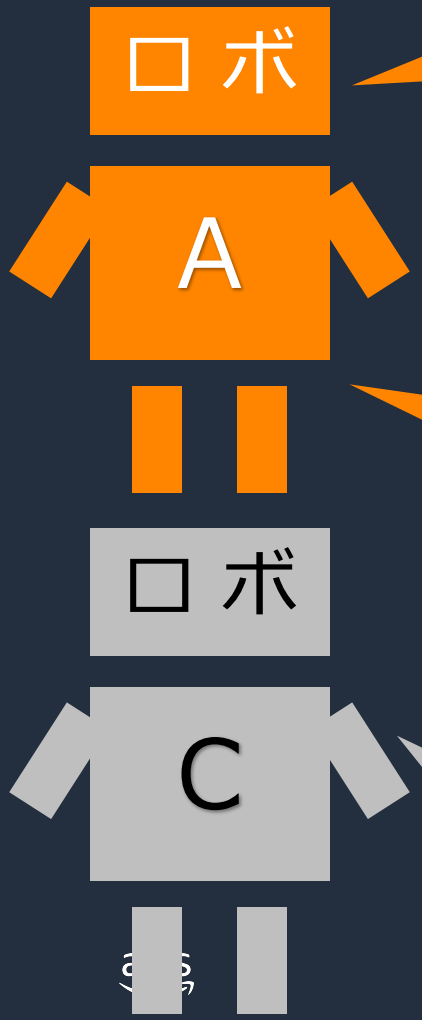


モニタリング種別	バイアス
名前空間	aws/sagemaker/Endpoints/ bias-metrics
ディメンジョン	BiasStage (pre-training or post-training), Endpoint, Facet, FacetValue, Label, LabelValue
メトリクス	バイアス指標: bias_metric_指標名(P10参照)

Feature Attributionのメトリクスと実装方法

Feature Attributionを監視していないと・・・

ロボットのみが働く2100年、ロボット営業所で同僚との会話



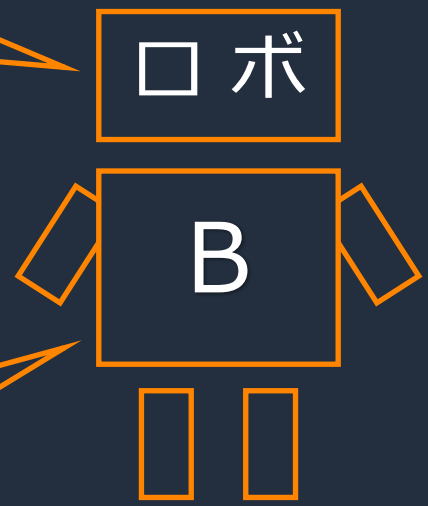
去年は2070年型ロボットが多くて、2080年型ロボットも結構貢献してたけど、最近、2080年型ロボットはあんまり活躍してなくない？

でもシフトから予測される売上は今でも精度悪くないから、去年と変わらないんじゃない？

2070年型ロボットが今年になってサポート終了に伴って活動停止したことか2090年型ロボットが増えていることとかが関係あんのかなぁ。

ロボットのパフォーマンスって別に今年も去年も変わらなくない？

2070年型ロボットが停止になったことや2090年型ロボットが増えたことで、もしかしたら2080年型ロボットの貢献度が変わったのかもね。SHAP値の変化を見てみたら？



入力データ内のレコード間の関係性の変化

各特徴量の分布に変化なし
(データ品質に変化なし)

学習時

旧型	スキルA	スキルB
1	1	0
1	0	1
0	1	1
0	1	0

1年後

旧型	スキルA	スキルB
1	0	0
0	1	1
0	1	0
1	1	1

2乗誤差はどちらも13(モデル品質に変化なし)

予測

単価
10
15
12
4

単価
5
12
4
20

正解

単価
8
17
14
5

単価
3
14
5
22

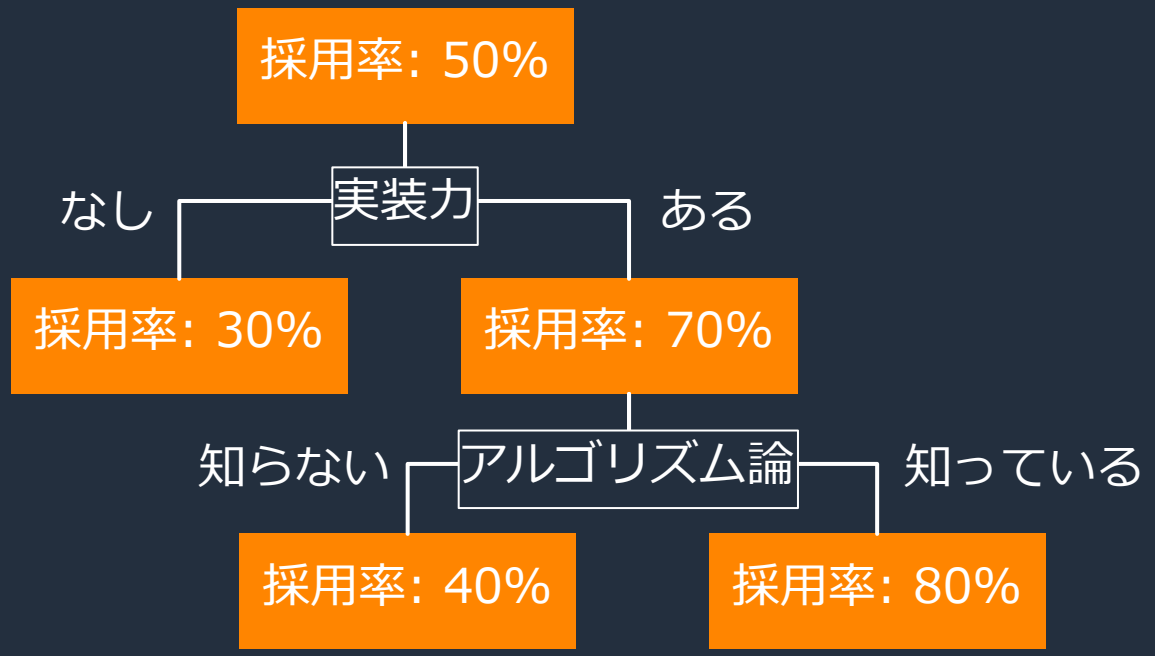
同じ入力に対する
正解データに変化なし
(バイアスに変化なし)

入力データ内のレコード間の関係性の変化により
各特徴量の貢献度に変化の可能性あり

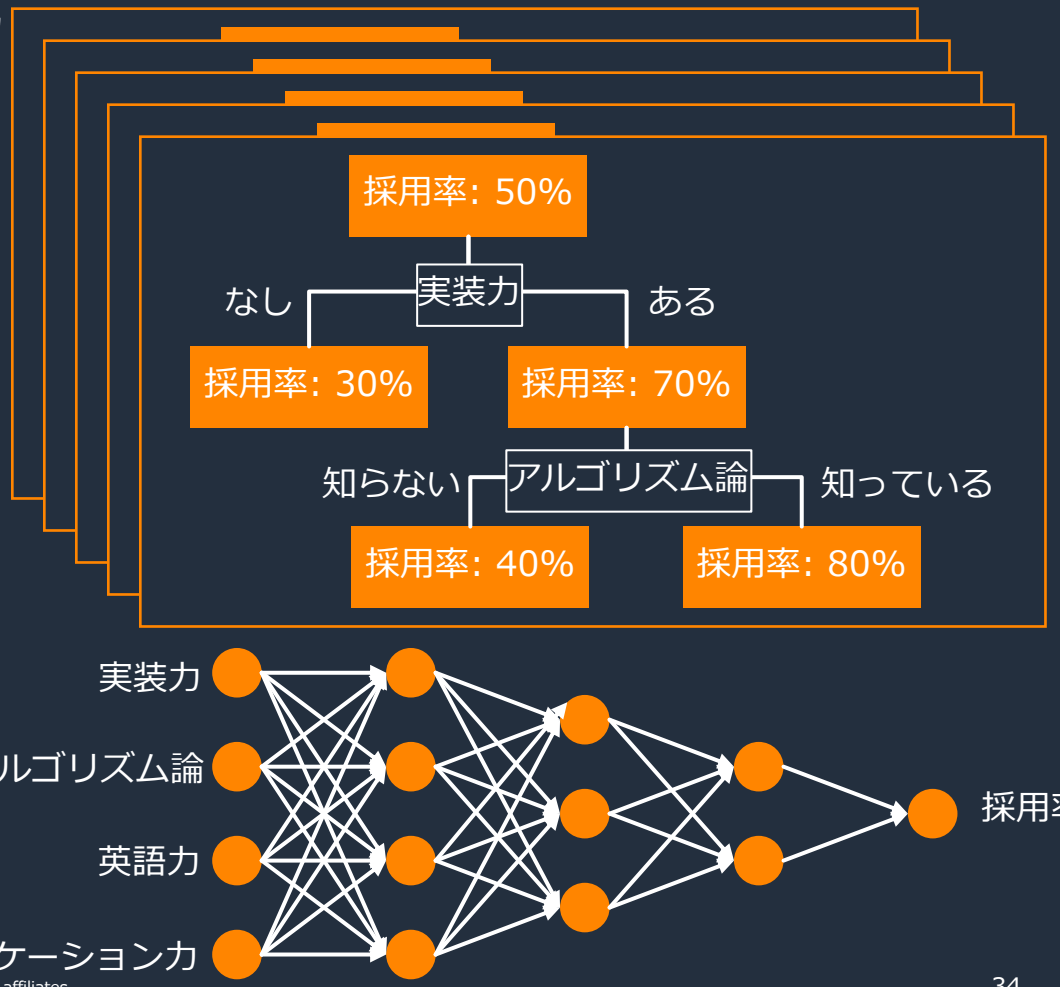
機械学習モデルの複雑さと精度はトレードオフ

精度は落ちるが推論根拠がわかる ⟷ 精度は良いが推論根拠がわからない

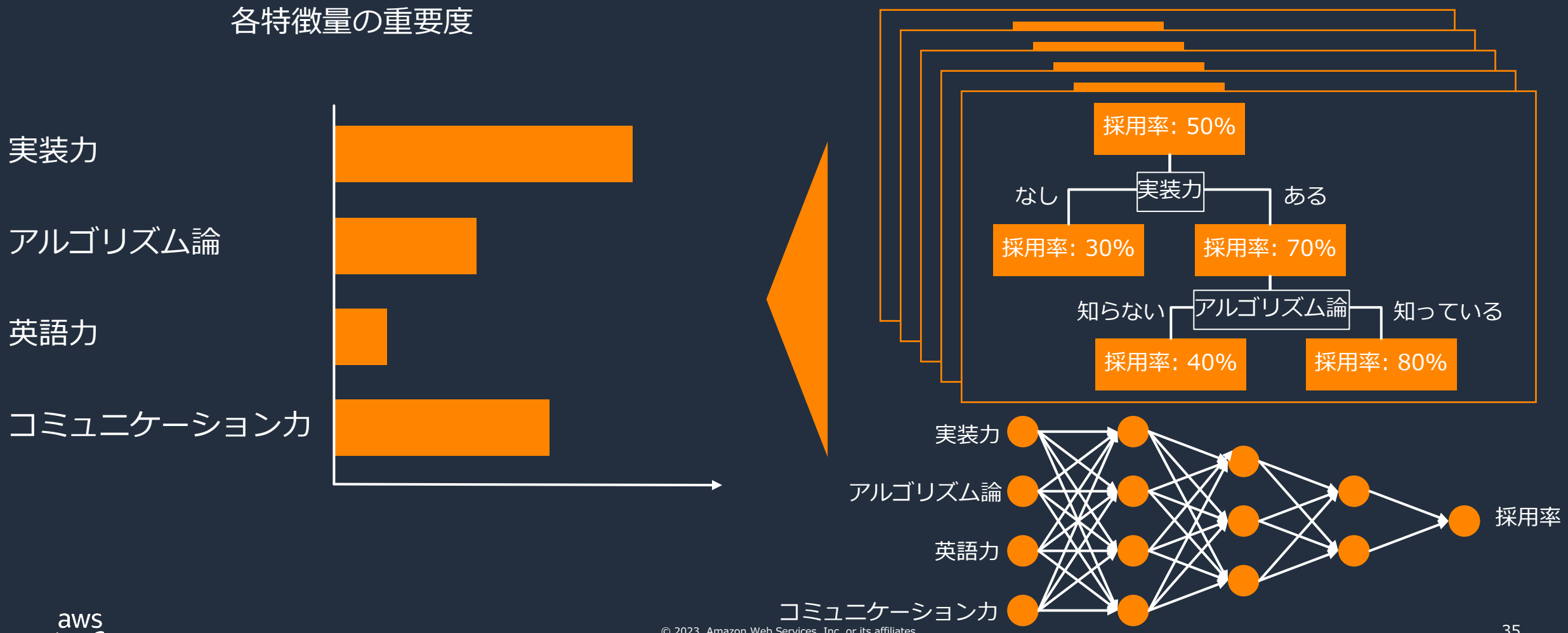
一般にモデルの複雑さと
精度はトレードオフ



採用スコア = A×今までに実装したコード数
$+B$×知っているアルゴリズム数



複雑なモデルの推論根拠を知りたい



各特徴量の重要度を調べる

協力すると売上に相乗効果が生まれる。
それぞれの人の貢献度をどう評価するか。

Aさん	Bさん	Cさん
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

営業

売上
3
6
12
9
18
24
36

貢献度の分配

15=24-9

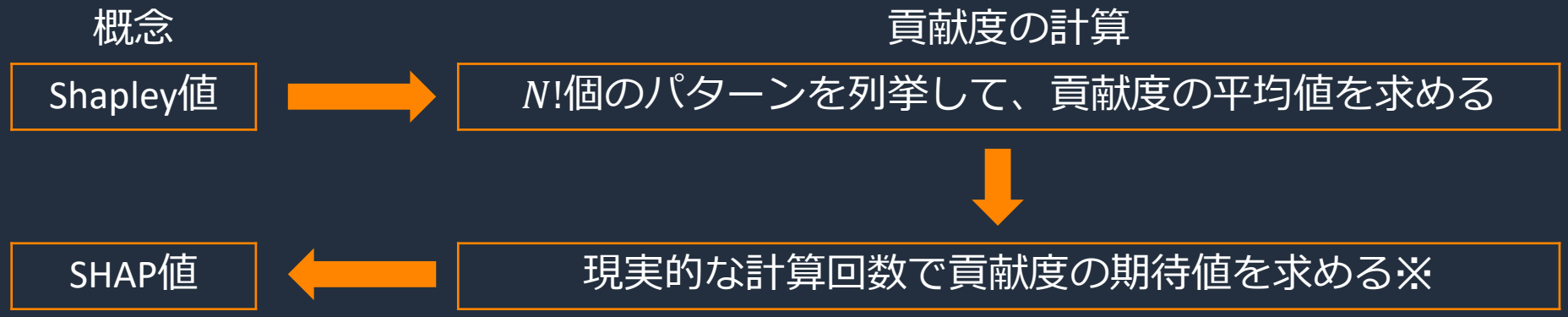
	1番目の貢献度	2番目の貢献度	3番目の貢献度
A -> B-> C	9	15	12
A -> C -> B	9	9	18
B -> A -> C	6	18	12
B -> C -> A	6	6	24
C -> A -> B	3	15	18
C -> B -> A	3	9	24

Aの貢献度の平均値: 16.5
Bの貢献度の平均値: 12.0
Cの貢献度の平均値: 7.5

Shapley値

すべての順列(N!通り)を考えないといけないので
計算量がボトルネック

(グローバル)SHAP値(SHapley Additive exPlanation)



※Amazon SageMakerではKernel SHAPという方法でSHAP値を計算します

補足: SHAP値は下記を満たす唯一の解であることが知られています(SHAP値以外は下記の何かを満たしません)。

#	特性	定義	解釈
1	Local accuracy	$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$	入力xに対する出力値を、0/1変数x'を持つ1次式で表現する。ϕ_iをその貢献度と考える。
2	Missingness	$x'_i = 0 \Longrightarrow \phi_i = 0$	0/1変数が0であれば貢献も0とする。
3	Consistency	$f'_x(z') = f\left(h_x(z')\right)$とし、$z' \setminus i$を$z'_i = 0$とする。 任意のモデル$f$と$f'$に対して、 もし$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i)$ならばすべての入力$z' \in \{0,1\}^M$に対して$\phi_i(f', x) \geq \phi_i(f, x)$となる。	2つのモデルA、Bがあり、特徴量iの値を0から1にしたときの出力値の変化量がどんなときでもAの方が大きい時は貢献度もAの方が大きい。

Normalized Discounted Cumulative Gain(nDCG)

SHAP値の変化をnDCGスコアで検知(しきい値: 0.9)
nDCGスコアとはランキングの評価メトリクスの1つで、0~1の値をとる

理想に対して比較対象とnDCGスコアの関係の例
理想(ベースラインのSHAP値): [1, 2, 3, 4, 5]

nDCGスコアが1近くの場合

比較対象	nDCGスコア
[1, 2, 3, 4, 5]	1.000000
[2, 1, 3, 4, 5]	0.995734
[1, 3, 2, 4, 5]	0.993251
[1, 2, 4, 3, 5]	0.987254
[2, 3, 1, 4, 5]	0.984718

nDCGスコアが中央値近くの場合

比較対象	nDCGスコア
[1, 4, 5, 2, 3]	0.863657
[4, 3, 2, 5, 1]	0.862275
[1, 5, 2, 3, 4]	0.859968
[3, 2, 5, 4, 1]	0.858587
[1, 4, 5, 3, 2]	0.856908

nDCGスコアが最小値近くの場合

比較対象	nDCGスコア
[5, 4, 1, 3, 2]	0.737525
[5, 3, 4, 2, 1]	0.734990
[5, 4, 2, 3, 1]	0.728992
[5, 4, 3, 1, 2]	0.726510
[5, 4, 3, 2, 1]	0.722243

[5, 12, 13, 15, 100]でもnDCGスコアは1

https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-model-monitor-model-attribution-drift-violations.html

モニタリングの実施に向けた作業　for SageMaker Model Monitor

A. モニタリング開始前にやること

A1. エンドポイントのデプロイ
データキャプチャの開始

A2. ベースラインの作成
データセットの統計量や制約、
SHAP値、バイアス指標、精度の基準生成

A3. モニタリングのスケジュール
データ品質、モデル精度、バイアス、
Feature Attributionのモニタリング設定

B. モニタリング中にやること

B1. Ground Truthの収集
Ground Truthの作成、登録、予測とのマージ

B2. モニタリング結果の分析
メトリクスの可視化、レポートの分析

B3. CloudWatchのアラート
モデル再学習・データ更新のトリガー

A2. ベースラインの作成

SageMaker Python SDKを使ったコード

```
# 1. 適切なMonitorクラスをロードする
from sagemaker.model_monitor import ModelExplainabilityMonitor

# 2. Monitorクラスをインスタンス化する
model_explainability_monitor = ModelExplainabilityMonitor(
    role=role,
    sagemaker_session=sagemaker_session,
    max_runtime_in_seconds=1800,
)

# 3. ベースラインデータをもとに、ベースラインを計算する
model_explainability_monitor.suggest_baseline(
    data_config=model_explainability_data_config,
    model_config=model_config,
    explainability_config=shap_config,
)
```

下記については次ページ参照

- model_explainability_data_config
- model_config
- shap_config

analysis.json:
Kernel SHAPを使ったグローバルSHAP値算出結果


```
{
  "version": "1.0",
  "explanations": {
    "kernel_shap": {
      "label0": {
        "global_shap_values": {
          "A_AGE": 0.0447854470228154,
          "A_FTLF": 0.009446401259149226,
          "A_HGA": 0.07363732898229544,
          "A_HSCOL": 0.005626855954026538,
          "A_MARITL": 0.01699549594206171,
          "A_SEX": 0.05595707596685706,
          "A_UNMEM": 0.006779998942751362,
          "A_USLHRS": 0.08162084323449582,
          "NOEMP": 0.03019643516165824,
          "PENATVTY": 0.04352613065700009,
          "PRCITSHP": 0.009645254409561424,
          "SEOTR": 0.009379782105990307,
          "WKSWORK": 0.11281256065808547
        },
        "expected_value": 0.05601481348276138
      }
    }
  }
}
```


この値がnDCGスコアを算出する際の理想として用いられる

A2. ベースラインの作成

SageMaker Python SDKを使ったコード

```
model_explainability_data_config = DataConfig(
    s3_data_input_path='s3://{バケット名}/{パス}/{ファイル名}',
    s3_output_path='s3://{バケット名}/{パス}',
    label=label_header,          # ターゲット変数名
    headers=all_headers,         # ターゲット変数を含む変数名のリスト
    dataset_type='text/csv',
)

model_config = ModelConfig(
    model_name='sample_model',
    instance_count=1,
    instance_type='ml.m5.large',
    content_type='text/csv',
    accept_type='text/csv',
)

shap_config = SHAPConfig(
    baseline=[list({各説明変数の平均値})],
    num_samples=100,
    agg_method='mean_abs',
    save_local_shap_values=True,
)
```

クラス	ベースライン作成用に必要な情報
DataConfig	入力データと出力先の情報
ModelConfig	モデルまたエンドポイントの情報
SHAPConfig	SHAP値を計算するための情報

SHAPConfigのパラメータ	内容
baseline	Kernel SHAPアルゴリズムに必要なベースライン(サンプルコードでは各特徴量の平均値)
num_samples	Kernel SHAPアルゴリズムの中で使用するサンプルデータ数
agg_method	グローバルSHAP値の集計方法(mean_abs,、median、mean_sq)
save_local_shap_values	Trueにすると集計前のSHAP値が保存される

A3. モニタリングのスケジュール　パラメータについてはPart1参照

SageMaker Python SDKを使ったコード

```
# Monitorインスタンスに対し、
# モニタリングスケジュールを作成する
model_config = ModelConfig(
    model_name='sample_model',
    instance_count=1,
    instance_type='ml.m5.large',
    content_type='text/csv',
    accept_type='text/csv',
)

headers_without_label_header = copy.deepcopy(all_headers)
headers_without_label_header.remove(label_header)
model_explainability_analysis_config = ExplainabilityAnalysisConfig(
   explainability_config=shap_config,
   model_config=model_config,
   headers=headers_without_label_header,
)
```

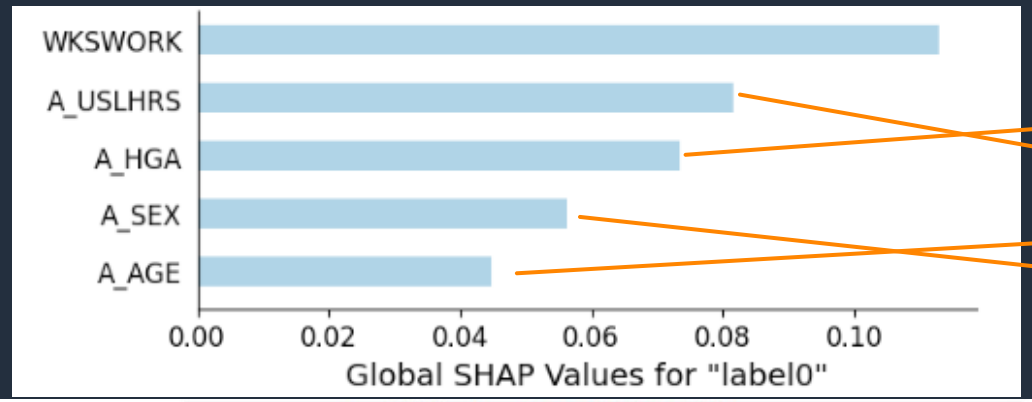
```
model_explainability_monitor.create_monitoring_schedule(
   analysis_config = model_explainability_analysis_config,
   output_s3_uri=s3_report_path,
   endpoint_input=endpoint_name,
   schedule_cron_expression=schedule_expression,
   enable_cloudwatch_metrics=True,
)
```

ベースラインを作成するModelConfigで
endpoint_nameを指定した場合は,
model_nameを指定したModelConfigを用意して、
create_monitoring_scheduleで
ExplainabilityAnalysisConfig経由でanalysis_config
に渡します。

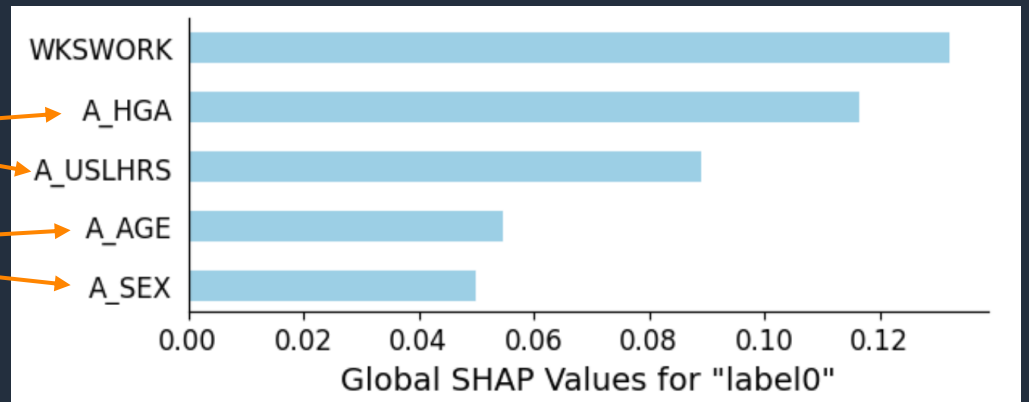
B2. モニタリング結果の分析

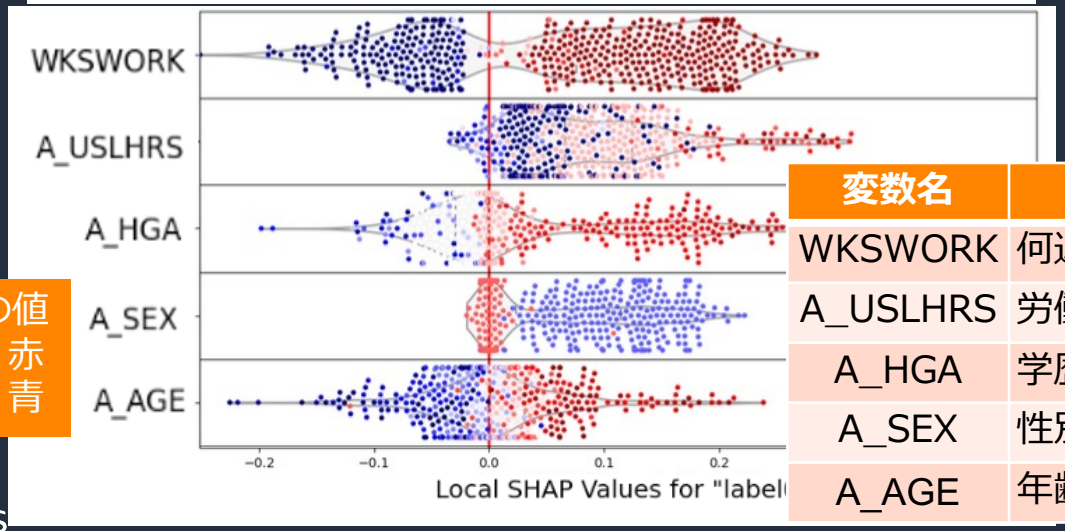
13変数の中で上位10個の変数がreport.pdfで出力されますがここでは5個に限定しています(Model Monitorに機能ではなく、資料上で加工しています)。

ベースライン



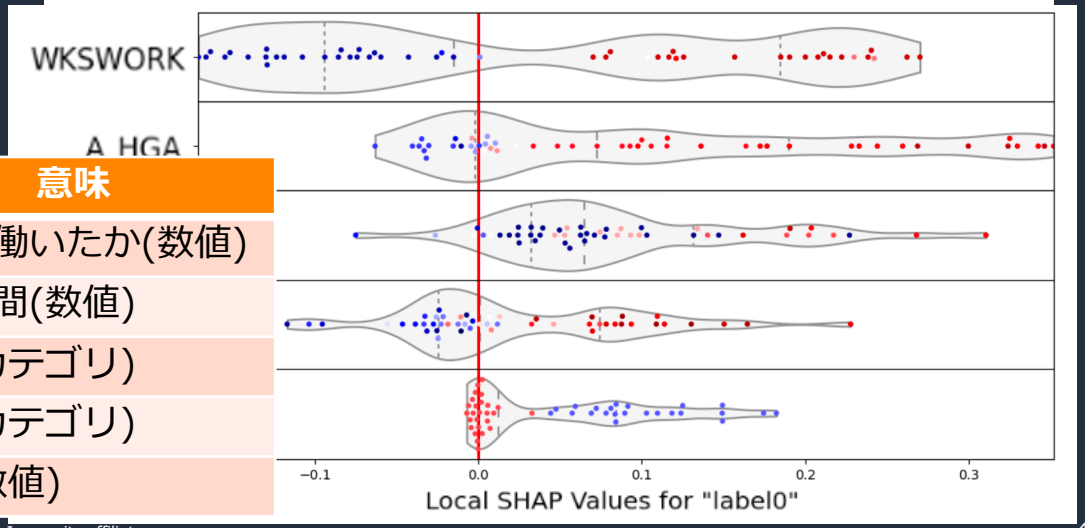
ある入力に対するグローバルSHAP値





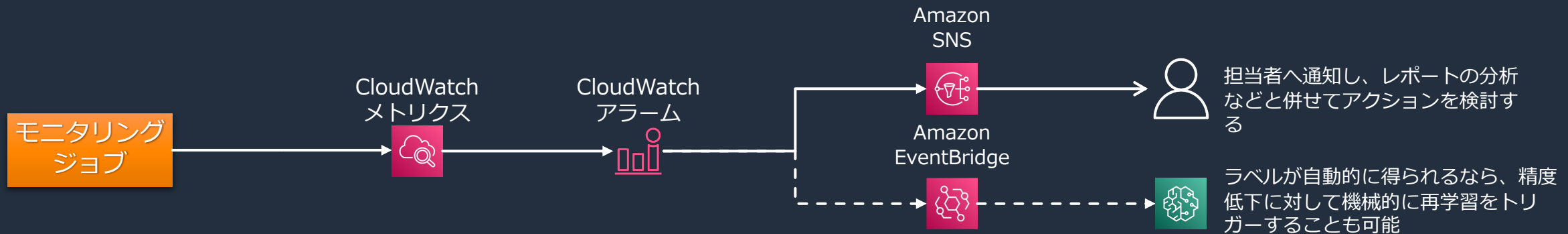
特徴量の値
大きい: 赤
小さい: 青

変数名	意味
WKSWORK	何週間働いたか(数値)
A_USLHRS	労働時間(数値)
A_HGA	学歴(カテゴリ)
A_SEX	性別(カテゴリ)
A_AGE	年齢(数値)



B3. CloudWatchのアラート

バイアスの変化をViolationとして検出しアクションにつなげる

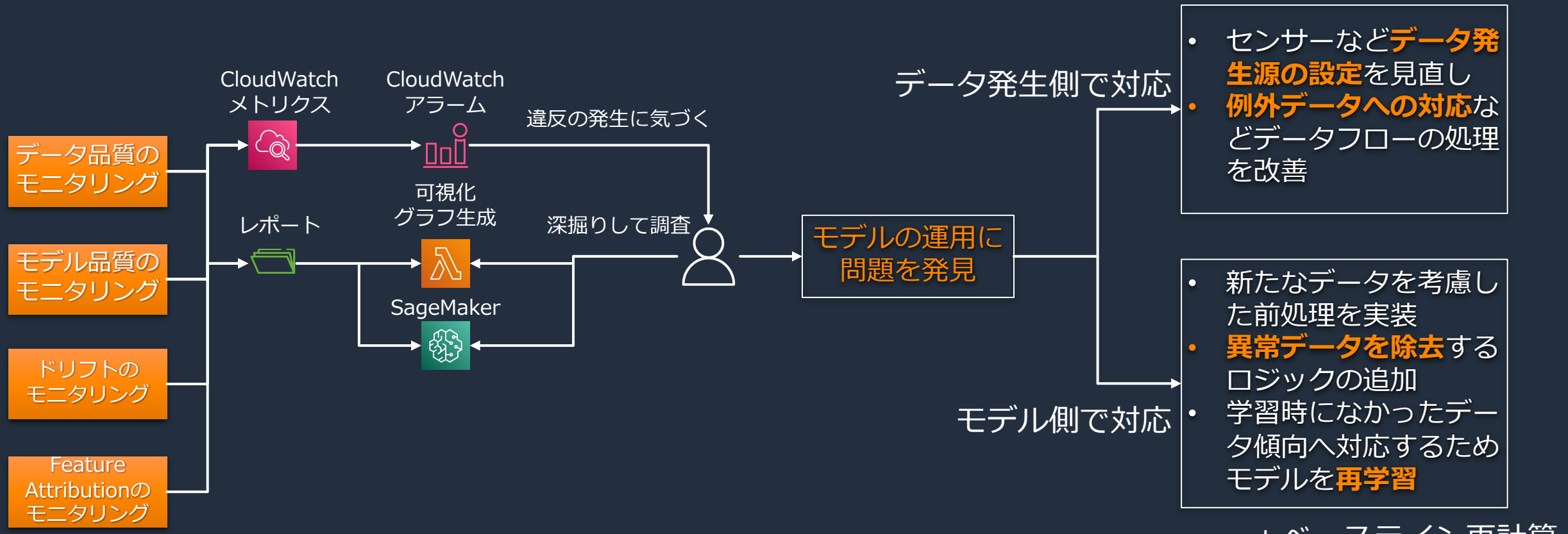


モニタリング種別	Feature Attribution
名前空間	aws/sagemaker/Endpoints/explainability-metrics
ディメンジョン	Endpoint, ExplainabilityMethod (KernelShap), Label, MonitoringSchedule, ValueType (GlobalShapValues or ExpectedValue)
メトリクス	期待値: ExpectedValue (特徴量毎) グローバルSHAP値: feature_特徴量名

nDCGはメトリクスにないので、もしnDCGで変化を検知したい場合は別途計算する必要があります。

モニタリングの結果からアクションを起こす

モニタリングの設定はそれぞれ個別に行うが、問題の発生時は相互に関連するため、いずれかの異常をトリガーとして相互に調査してアクションを決める



まとめ

- データ品質やモデル品質のモニタリングだけではわからない、バイアスとFeature Attributionのモニタリングの必要性を解説
- Amazon SageMakerにおけるバイアスのメトリクスと実装方法を具体的に解説
- Amazon SageMakerにおけるFeature Attributionのメトリクスと実装方法を具体的に解説
- 紹介したサンプルコードはGitHubで公開

ML Enablement Seriesの動画

機械学習モデルをビジネス価値につなげる方法を全力で解説！

Light Part

製品やサービスに機械学習を導入するプロジェクトの進め方

https://bit.ly/3M1F9as

Step Up!!

Dark Part

機械学習モデルの開発や運用をマネージドサービスで効率的に行う方法

https://bit.ly/3927PCN

資料集・お問合せ・Special Thanks

AWSの日本語資料の場所: 「AWS 資料」で検索

AWSのハンズオン資料の場所: 「AWS ハンズオン」で検索

お問合せ

技術的なお問合せ

料金のお問合せ

個別相談会のお申込み

Thank you!