

# 3. アプリケーション設計 & 構築

Azure Machine Learning – 構築・運用編

Keita Onabuta

FastTrack for Azure

Senior Customer Engineer for AI/ML

# Agenda

開発環境と言語・フレームワーク

モデル学習

デプロイメントと推論

アセット管理

MLOps

責任のある AI

# 開発環境と言語・フレームワーク

## 基礎知識

- 機械学習の主要な IDE
- Azure ML – Integrated Notebook
- Azure ML – AutoML
- Azure ML - Designer

## 検討事項

- 開発環境の選択
- Azure Machine Learning API の選択

## ガイドライン・実装手順

- API 選択ガイドライン

## Tips

- Designer チートシート
- コードテンプレート

# 機械学習の主要な IDE



Jupyter / JupyterLab は対話型で操作ノートブックベースの開発環境を提供。



Microsoft が提供する汎用的な開発環境。Python & R スクリプトの開発や Jupyter Notebook もサポート。  
※ GitHub Codespace を利用しているケースもあります。



機能豊富な Python 専用の開発環境。



機能豊富な R 専用の開発環境

# Azure ML – Integrated Notebook

Azure Machine Learning studio に統合されている Jupyter 互換のノートブック環境

## Jupyter との互換性

- Jupyter Notebook をインポートしそのまま実行可能

## 強化された Editor

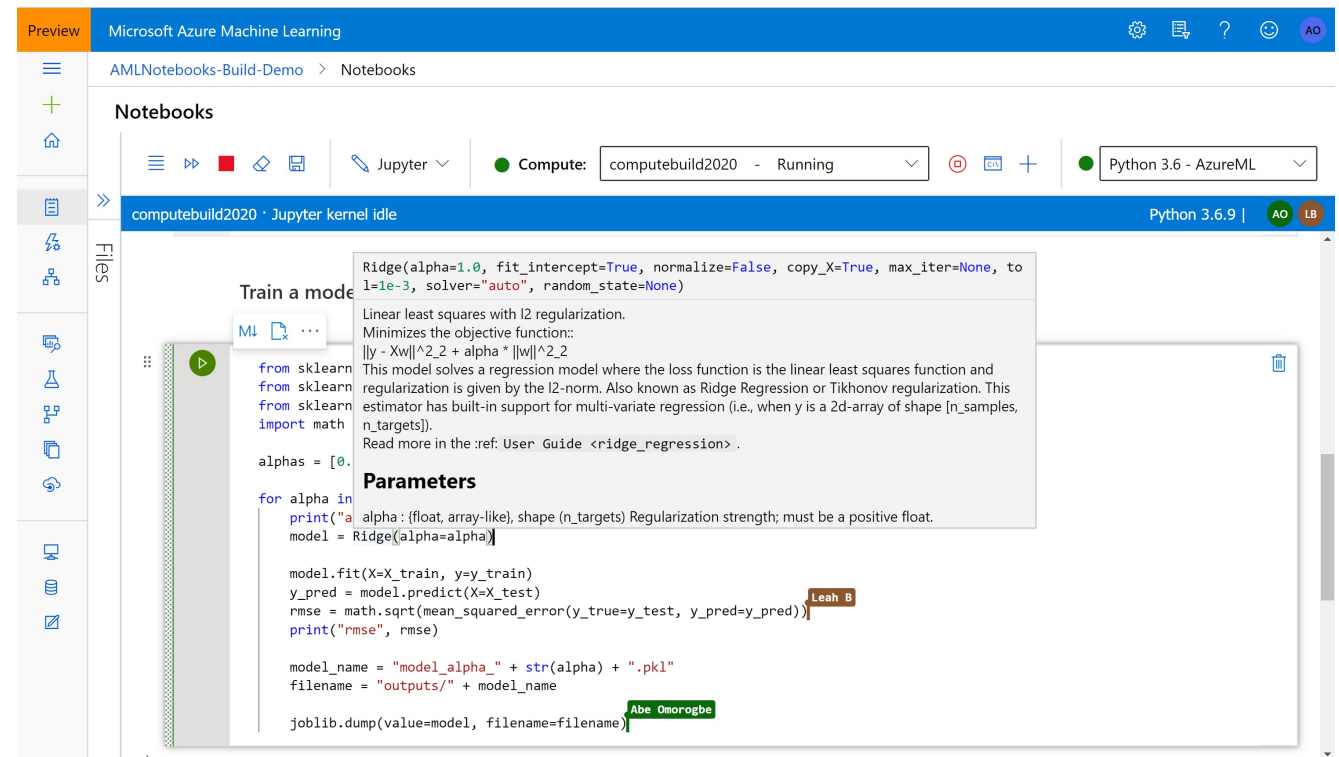
- Monaco editor, IntelliSense & Gather

## Git 統合とターミナル

- Git を利用したコード管理や CI/CD の実装
- ターミナルを用いた自由度の高い環境設定

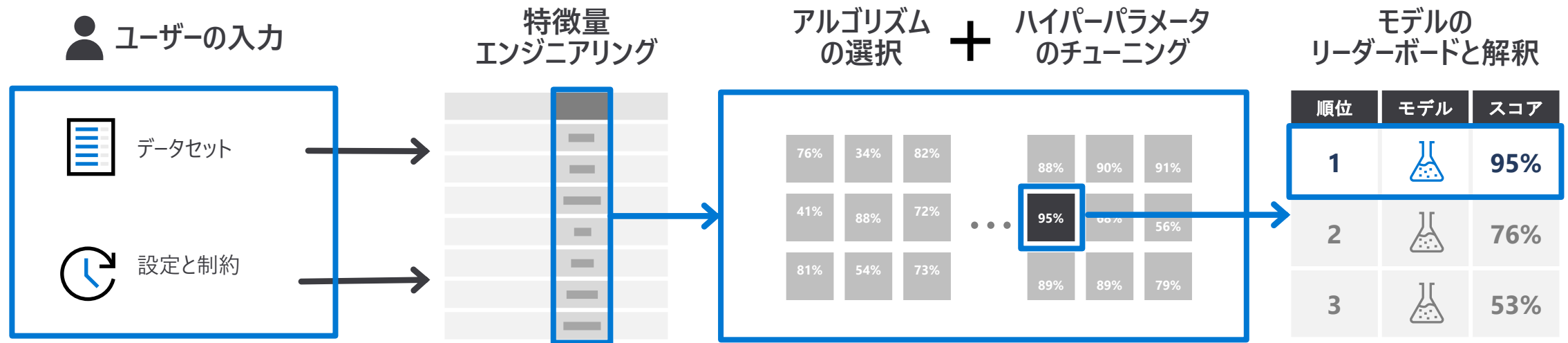
## VSCode 統合 (Preview)

- VS Code にそのまま移動可能



# Azure ML – AutoML (Automated Machine Learning)

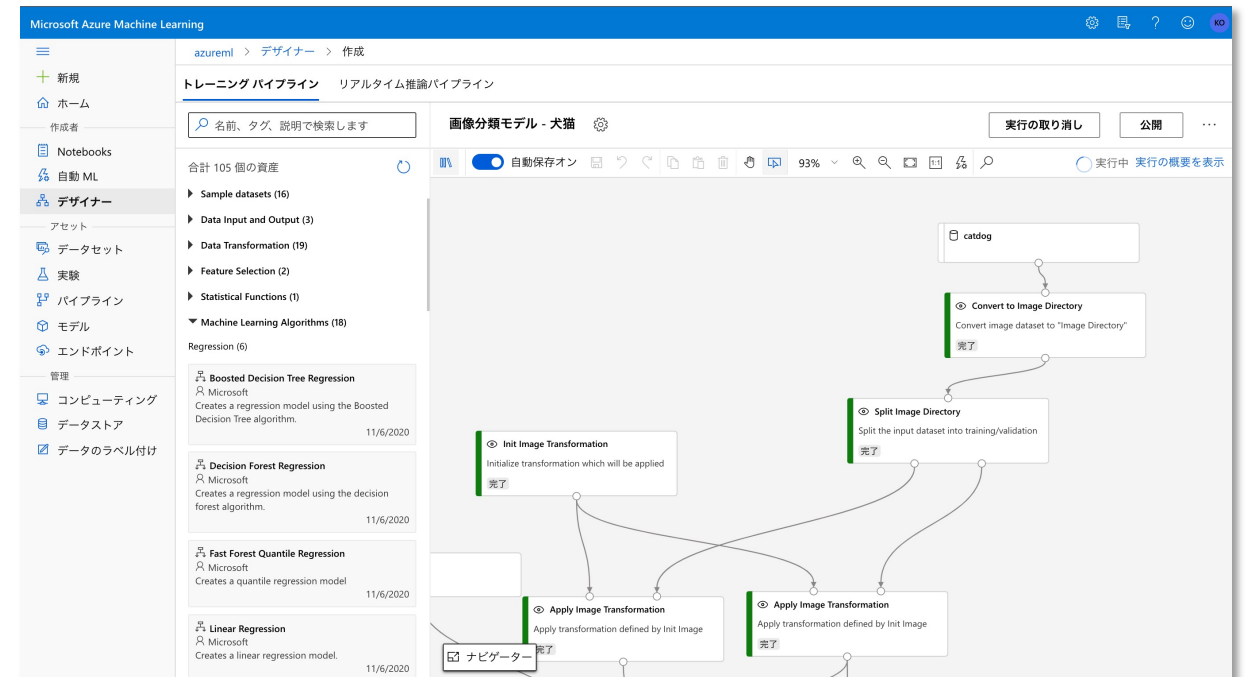
AutoML (自動機械学習) は、与えられたデータに対して「最高のモデル」を探索するために、特徴量エンジニアリング、アルゴリズムとハイパーパラメータの選択を自動実行します。



# Azure ML – Designer

## 機械学習のモデル構築、テスト、デプロイするためのビジュアルパイプライン

- 直感的なマウス操作でパイプライン構築
- 提供されるモジュール
  - 特徴量エンジニアリング
  - モデル学習 (回帰、分類、クラスタリング)
  - カスタムモデル・スクリプト (Python, R)
- 構築したパイプラインはシームレスに推論環境にデプロイ可能 (バッチ、リアルタイム)



# 開発環境の選択

Azure Machine Learning と連携して利用する開発環境 (IDE) を検討します。

## Azure Machine Learning のマネージド

- Compute Instance 側に自動的に構築され、運用管理される。
- 正式サポートされている基本的な API はインストール済みです。
  - Azure CLI v2 (Preview) などの API はデフォルトではインストールされていません。
- 対象 IDE : Jupyter Notebook, JupyterLab, Visual Studio Code, R Studio

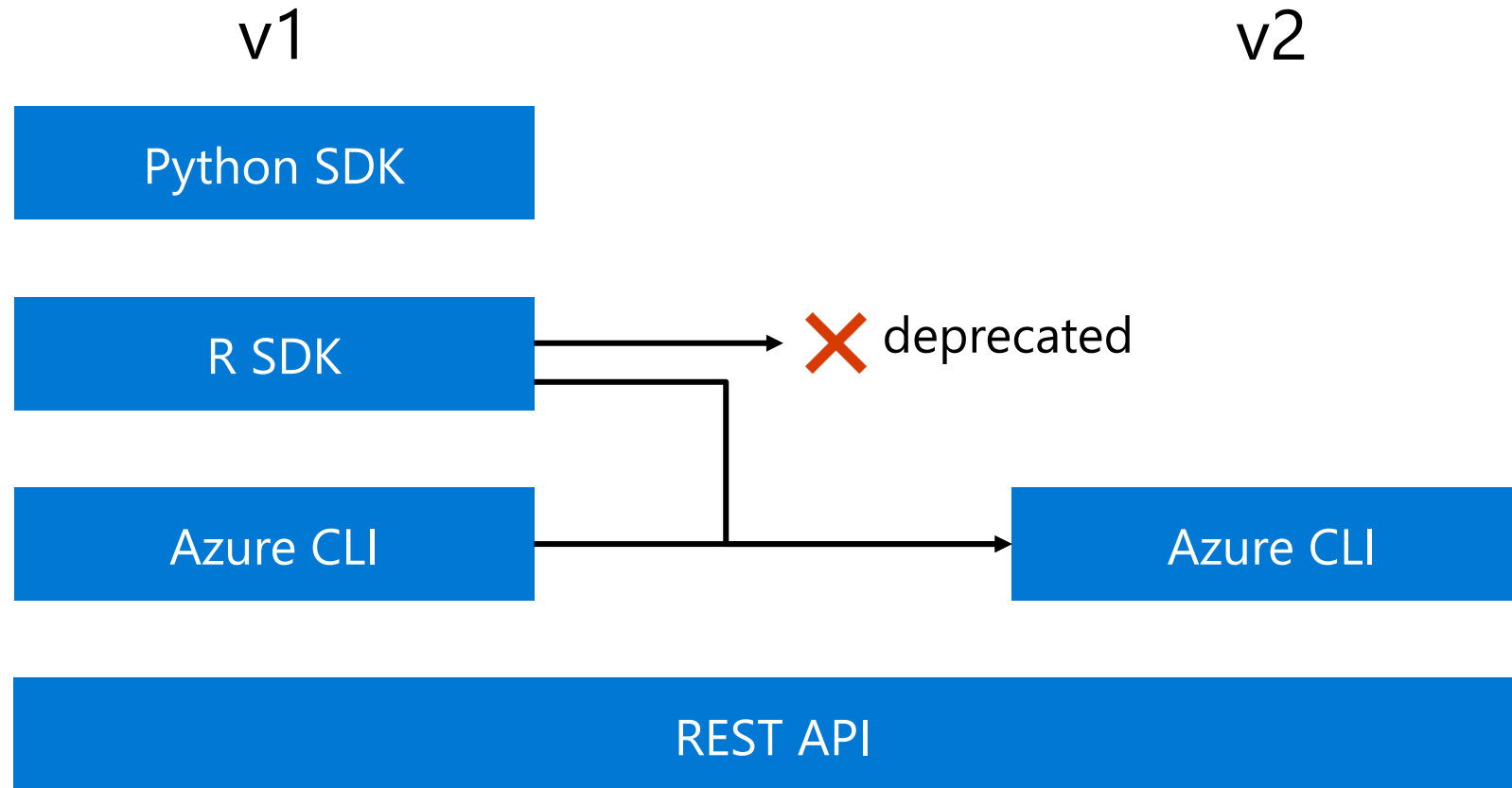
## ユーザ独自の開発環境

- ユーザ自身で環境を構築し運用管理する必要がある。
- 利用する API のクライアントをインストールすれば任意の開発環境が利用可能です。
- 例 : PyCharm, Visual Studio, Vim など



# Azure Machine Learning API の選択

Azure Machine Learning は Web インタフェース (Azure Machine Learning Studio) から操作することも可能ですが、外部環境から API を通じて様々な機能にアクセスできます。



# API の選択ガイドライン

プログラミング言語の観点から Azure Machine Learning で利用する API を選択する基本的なガイドラインです。

## Python

- Python SDK 、 Azure CLI v1 の利用が正式サポートされています。
- 将来的には Azure CLI は v2 に移行される予定なので利用を検討ください。

## R

- R SDK は deprecate されたので、 Azure CLI v2 への実装・移行を検討ください。

## その他 (Julia など)

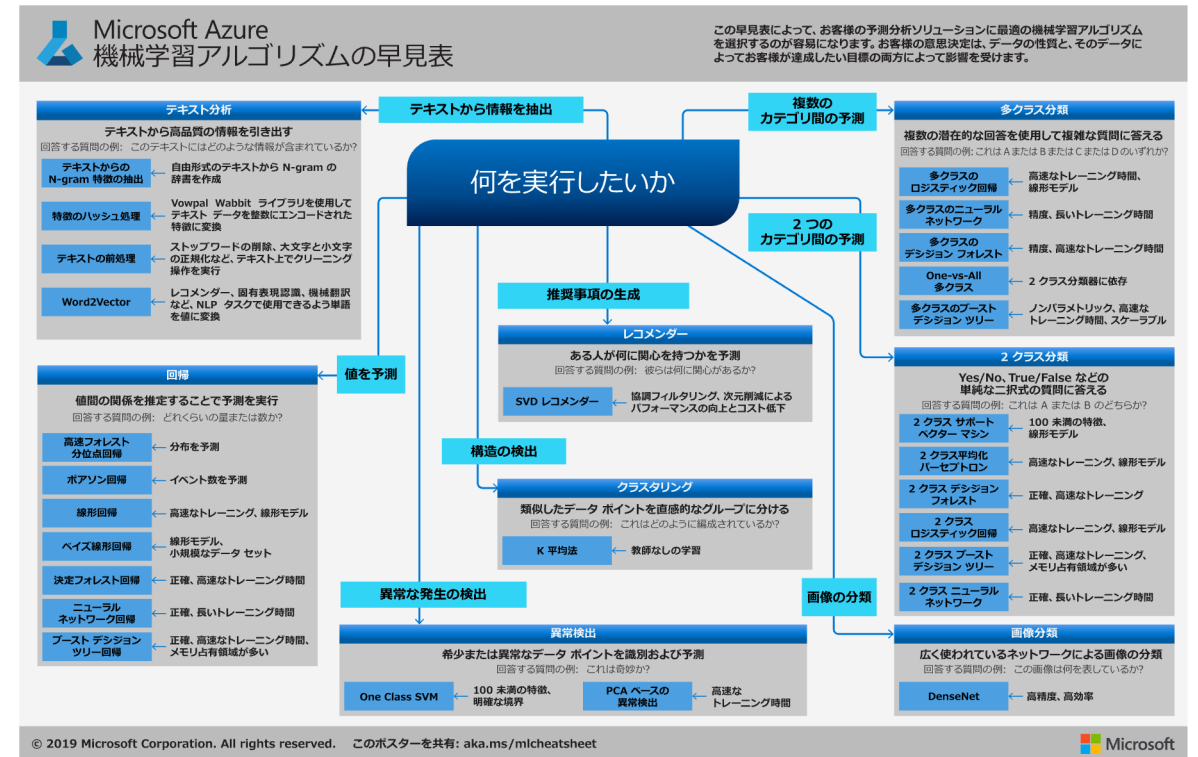
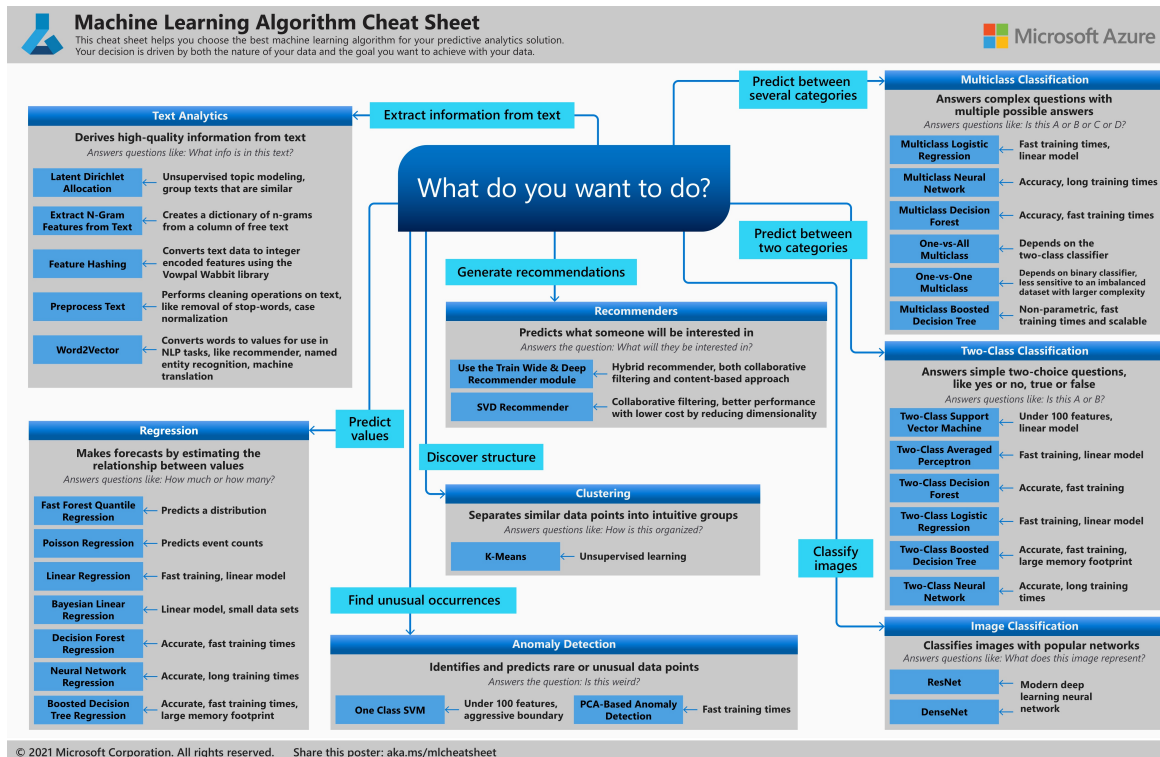
- v1 の API では工夫次第では実行できますが、基本的には Azure CLI v2 の利用を検討ください。

# Designer チートシート

Azure Machine Learning Designer を利用する際のガイドとなるチートシートです。

英語版

日本語翻訳版



# コードテンプレート

機械学習、データサイエンスでよく用いられるテンプレートを紹介します。

## Cookiecutter Data Science

- [Home - Cookiecutter Data Science \(drivendata.github.io\)](https://drivendata.github.io)

## DSLIP

- [dslp/dslp: The Data Science Lifecycle Process is a process for taking data science teams from Idea to Value repeatedly and sustainably. The process is documented in this repo. \(github.com\)](#)

# モデル学習

## 基礎知識

- Azure ML – Compute
- 分散学習
- ハイパーパラメータチューニング

## 検討事項

- 共有計算リソースの検討
- 分散学習の実装検討
- ハイパーパラメータチューニングの実装検討

## ガイドライン・実装手順

- コンピューティングクラスターでのジョブ実行フロー



# Computes

Computes は学習スクリプトを実行したりモデルを推論用途でホストするための計算リソースです。

Azure Machine Learning Python SDK, CLI, Studio から作成し運用管理することができます。

既存のリソースをアタッチすることもできます。

ローカル環境で実行したのちに、Compute Targets 上でスケールアップ・スケールアウトすることができます。

現在サポートされている Compute Target

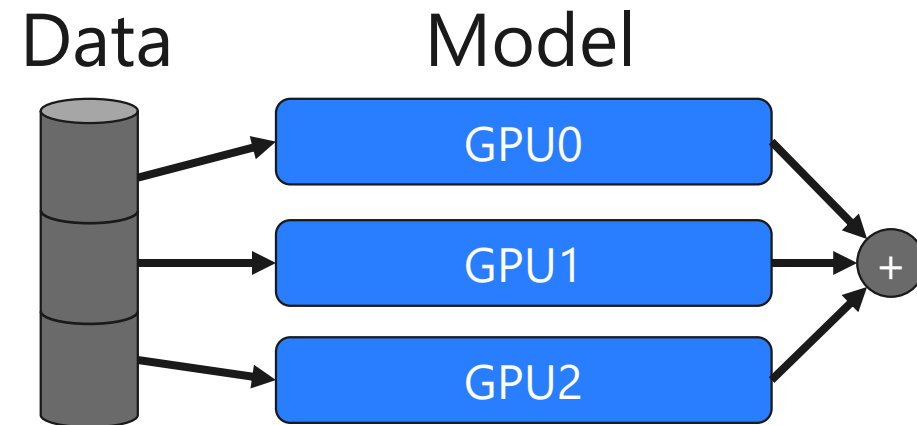
Compute Targets	学習	デプロイ
Local Computer	✓	
A Linux VM in Azure (such as the Data Science Virtual Machine)	✓	
Azure ML Compute Clusters	✓	✓
Azure ML Compute Instance		
Azure Databricks	✓	
Azure Data Lake Analytics	✓	
Azure HDInsight	✓	
Azure Container Instance		✓
Azure Kubernetes Service		✓
Azure IoT Edge		✓
Field-programmable gate array (FPGA)		✓
Azure Functions (preview)		✓
Azure App Service (preview)		✓
Azure Synapse Spark Pool (preview)	✓	
Azure Arc enable Kubernetes (preview)	✓	✓

# 分散学習 (Distributed Training)

複数の計算リソースを利用して大規模なモデルやデータを扱う機械学習を実行する

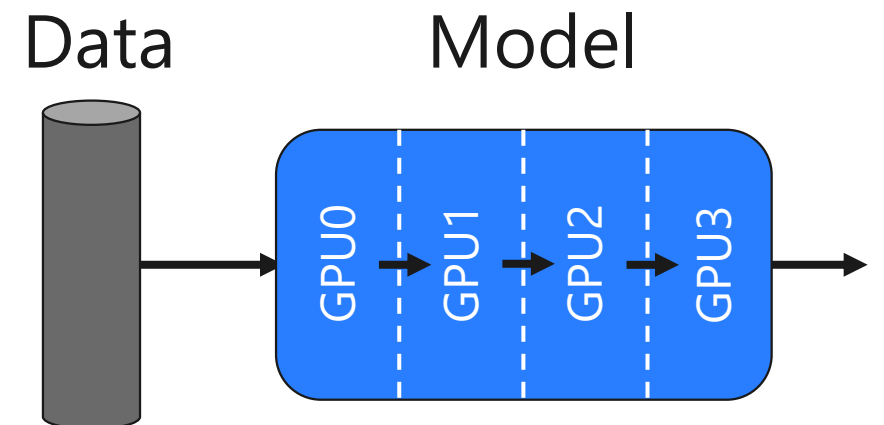
## Data Parallelism

- 同じモデルを各 GPU に配置して、データを分割したものを各 GPU のモデルに入力して出力を得て、最後に出力結果を集計する
- 必然的にミニバッチサイズが大きくなる点に注意が必要



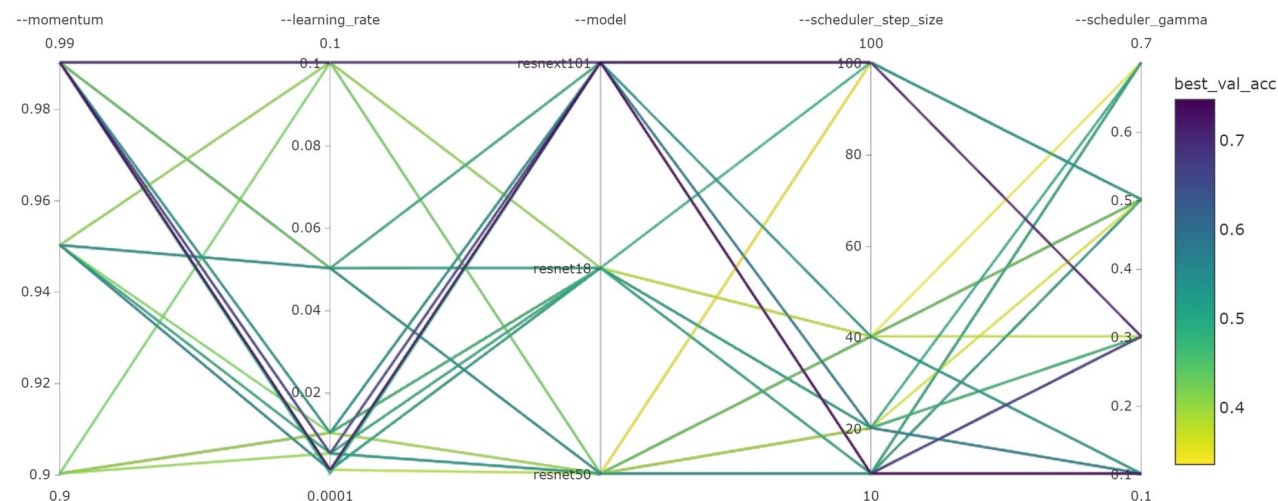
## Model Parallelism

- モデルの各パーツを GPU に分散配置してデータを全量入力し、GPU 間で担当パートの出力結果を転送しつつモデルの計算を行う
- 分散配置の仕方は層別に分離する場合もあれば層そのものを分割して配置する場合もある



# ハイパーパラメータチューニング

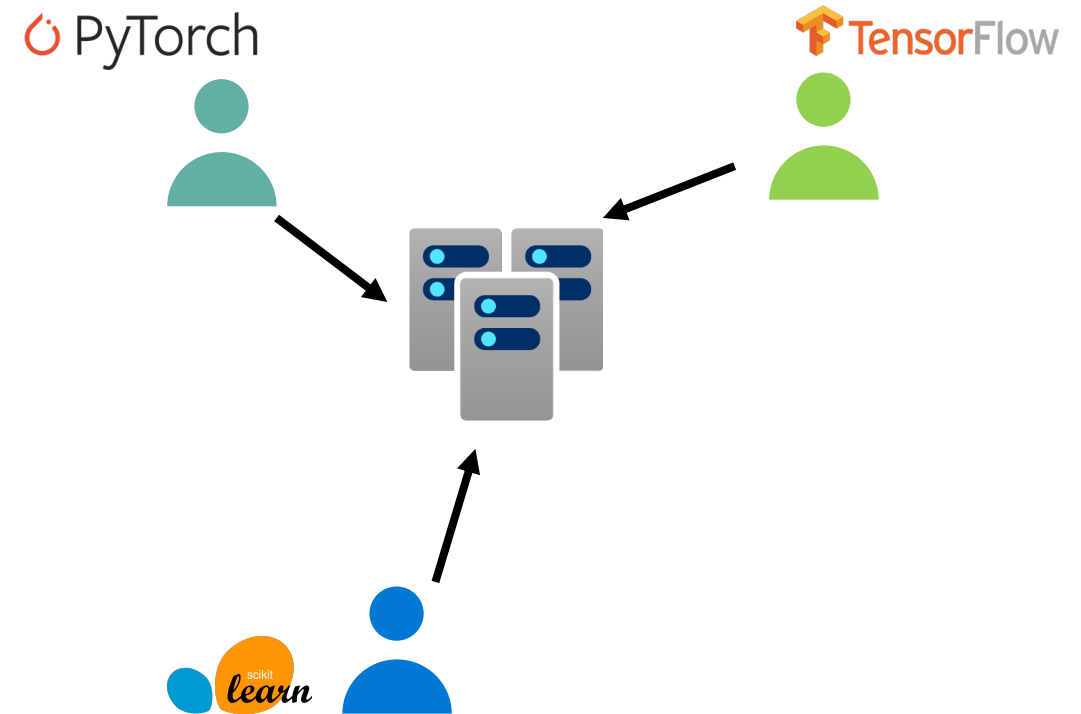
- ハイパーパラメータとは？
  - モデル学習開始前に設定するパラメータ
    - 例：学習率、Momentum、バッチサイズ、隠れ層数、ドロップアウト率、Estimator 数、木の深さ ...
- 最終的なモデル精度に影響する
  - 適切なハイパーパラメータを選択することで精度改善が見込まれる





# 共有計算リソースの検討

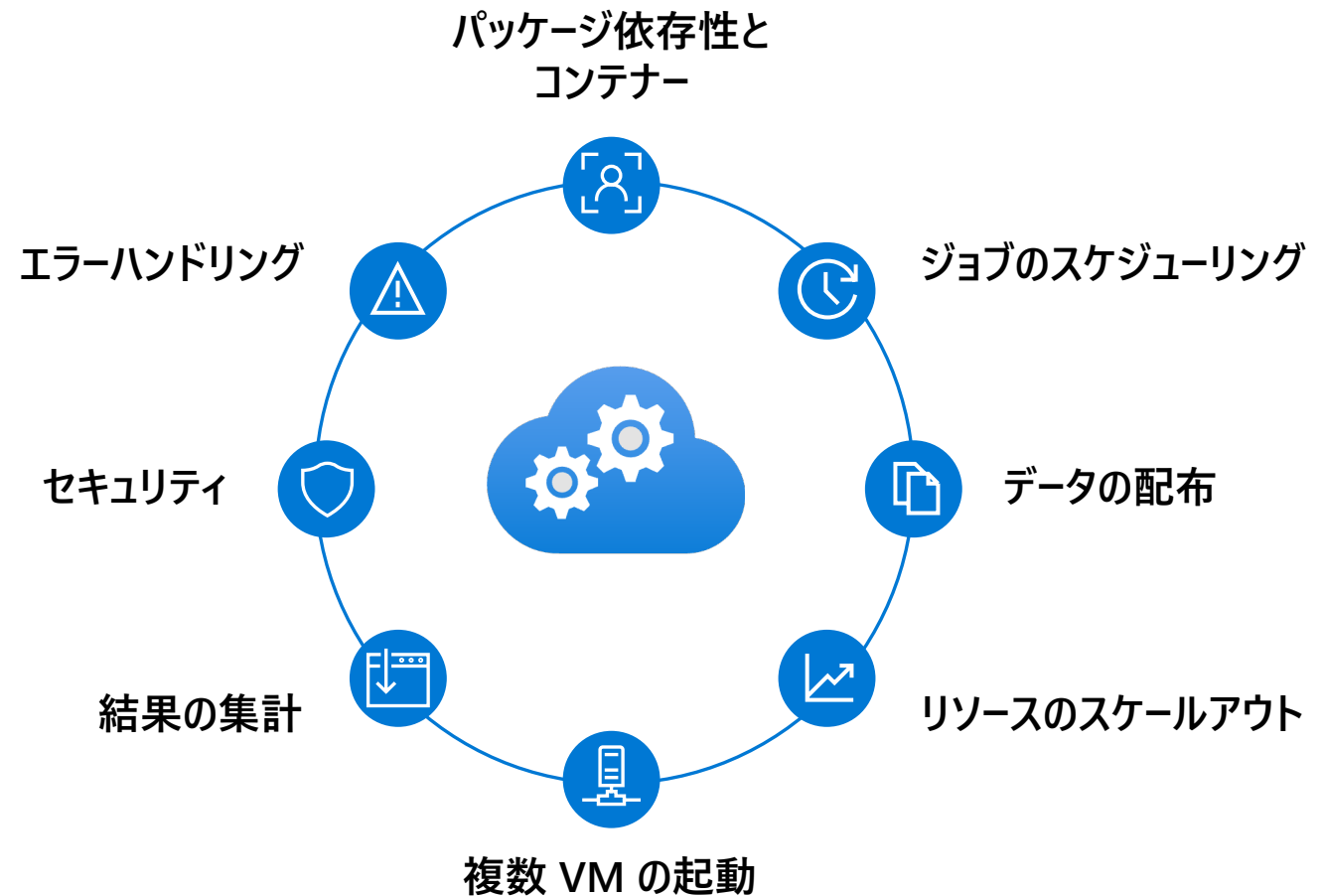
- スケジューリングとリソース管理の仕組み
  - 多種多様な計算環境 (CPU, GPU) をサポート
  - 複数ジョブを効率的に実行しクラスター無駄なく活用
- 生産性の向上
  - インフラ管理は最小限に
  - モデル構築にフォーカスできる環境
  - アセット (Code, Data, Model etc) を共有する仕組み



# 分散学習の実装検討

分散学習はインフラの構築や運用管理が煩雑です。

Azure Machine Learning では分散学習に必要なセットアップを自動で実行する機能が提供されています。



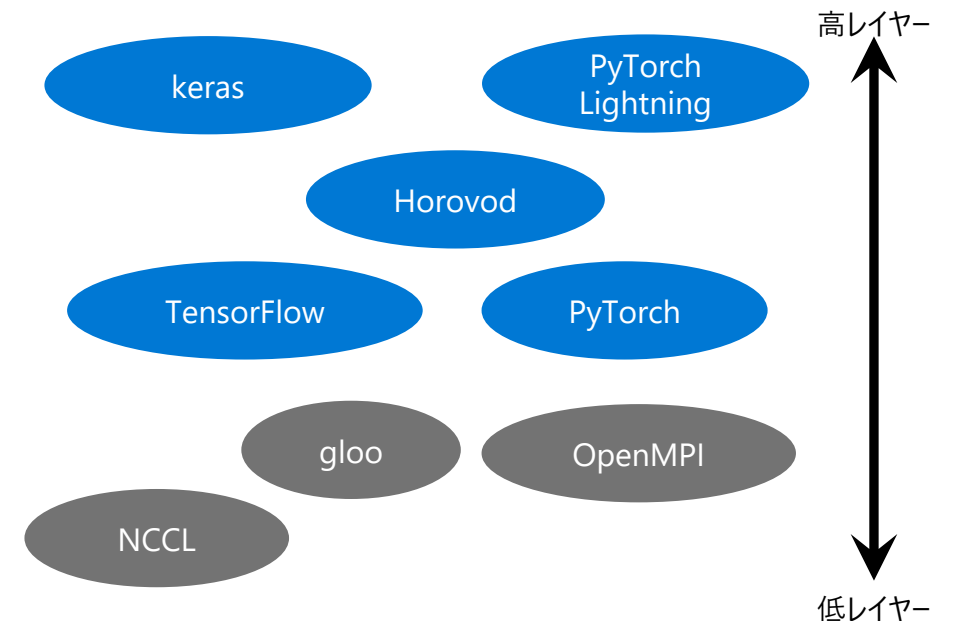
# 分散学習の実装方式の検討 (cont'd)

Azure Machine Learning 上で分散学習 (Distributed Training) を実装する方法を検討します。

## Azure Machine Learning ネイティブ

深層学習でよく使われる GPU の分散学習の技術が Azure ML の API に含まれています。コードはユーザ側で開発する必要がありますが、基本的なインフラ部分の構築・運用を Azure にお任せできます。

- MPI
- PyTorch
- TensorFlow
- Horovod



# 分散学習の実装方式の検討 (cont'd)

## Azure Machine Learning 動作実績あり

従来の機械学習アルゴリズムの分散学習の機能は直接 Azure ML の API からは提供していませんが、Community ベースでサンプルコードやライブラリが公開されています。コンピューティングクラスター上で利用できるため、自分で VM を構築して開発するよりも大幅に工数を削減できます。

- Spark
- DASK
- RAY



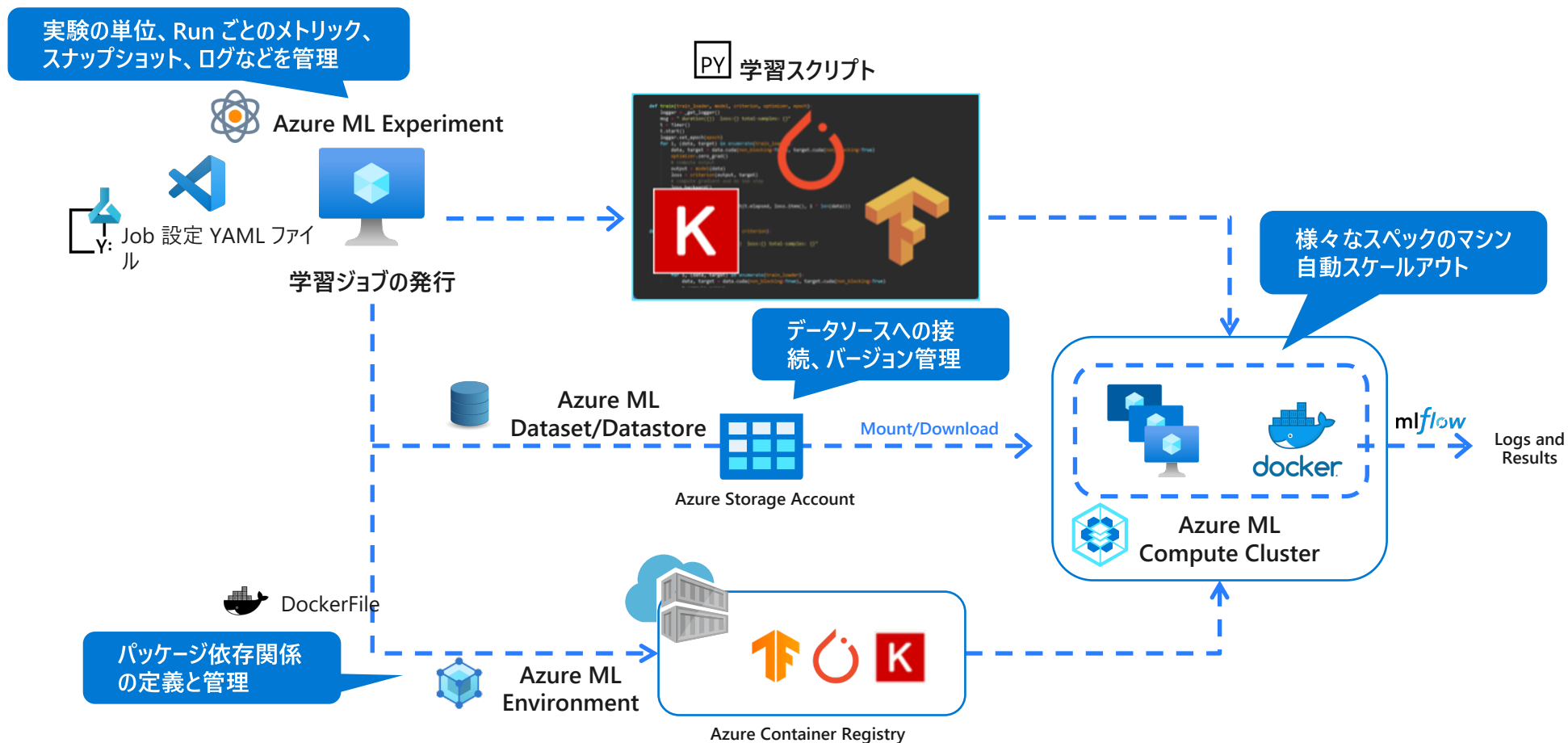
# ハイパーパラメータチューニングの実装検討

Azure Machine Learning 上でハイパーパラメータチューニング (Hyper Parameter Tuning) を実装する方法を検討します。

- Azure Machine Learning には **Hyperdrive** というハイパーパラメータチューニングの機能が内蔵されています。Hyperdrive は “Grid Search”, “Random Search”, “Beysian Optimization” の 3 つの手法が実装されています。
- オープンソーステクノロジーの Optuna, FLAML, NNI など Azure Machine Learning 上で利用することができます。

# コンピューティングクラスターでのジョブ実行フロー

コンピューティングクラスター環境でのジョブ実行のフローや関連するコンポーネントの概要です。



# デプロイメントと推論

## 基礎知識

- Azure ML - Pipelines

## 検討事項

- 推論形式の検討
- 推論環境の選択

## ガイドライン・実装手順

## Tips



# Pipelines

1つの Azure Machine Learning Pipeline は完成した機械学習タスクのワークフローを表し、機械学習の各サブタスクはパイプラインを構成する一連の手順として内包される

Azure Machine Learning Pipeline には、Python スクリプトを呼び出すだけのシンプルなものから、あらゆることを実行するものまで存在する。

## タスク

Pipelines は以下のような機械学習タスクにフォーカス:

- インポート、検証とクリーニング、変換、正規化、ステージングを含む「データの準備」
- パラメータ化された引数、ファイルのパス、ログとレポート出力の設定を含む「学習の設定」
- 効率的かつ繰り返し実行できる「学習と検証」。特定のデータサブセット、異なるハードウェアからなる計算リソース、分散処理、進捗状況の監視を指定することで、効率性を確保することが可能
- バージョン管理、スケーリング、プロビジョニング、アクセス制御等の要素を含む「デプロイ」



# 推論方式の選択

学習済みモデルをアプリケーションに連携・統合して業務で利用する推論の実行方式を検討します。大きく分けて 2 種類あります。

## □ リアルタイム推論

- 任意のタイミングで少量データに予測値を付与する推論の種類。通常 API 経由でデータの入出力を行う。

## □ バッチ推論

- バッチデータに対して予測値を付与する推論の種類。通常予測値はファイルやデータベースに格納され業務用アプリケーションと連携される。

# 推論環境の選択

要件に応じて適切な推論環境にモデルをデプロイします。Azure Machine Learning の機能としてサポートされている推論環境は下記です。

計算環境	推論形態	本番環境	GPU	分散・並列	オンプレミス	用途
Local Computer	Both		✓		✓	テスト・デバッグ
Azure Container Instances	Real-time					テスト・デバッグ
Azure Kubernetes Service	Real-time	✓	✓	✓		リアルタイム推論
Azure ML Compute Cluster	Batch	✓		✓		バッチ推論
Azure Synapse Analytics Spark Pool	Batch	✓		✓		バッチ推論
Azure IoT Edge	Realtime	✓			✓	IoT エッジ推論
Azure Arc	Both	✓	✓	✓	✓	ハイブリッド

# アセット管理

## 基礎知識

- MLflow

## 検討事項

- 実験追跡の実装方法の検討
- 環境構築方法の検討

## ガイドライン・実装手順

- 実験追跡の実装ガイドライン
- 環境 (Environments) の構築における注意点

# MLflow

機械学習のライフサイクルを管理するオープンソースのプラットフォーム

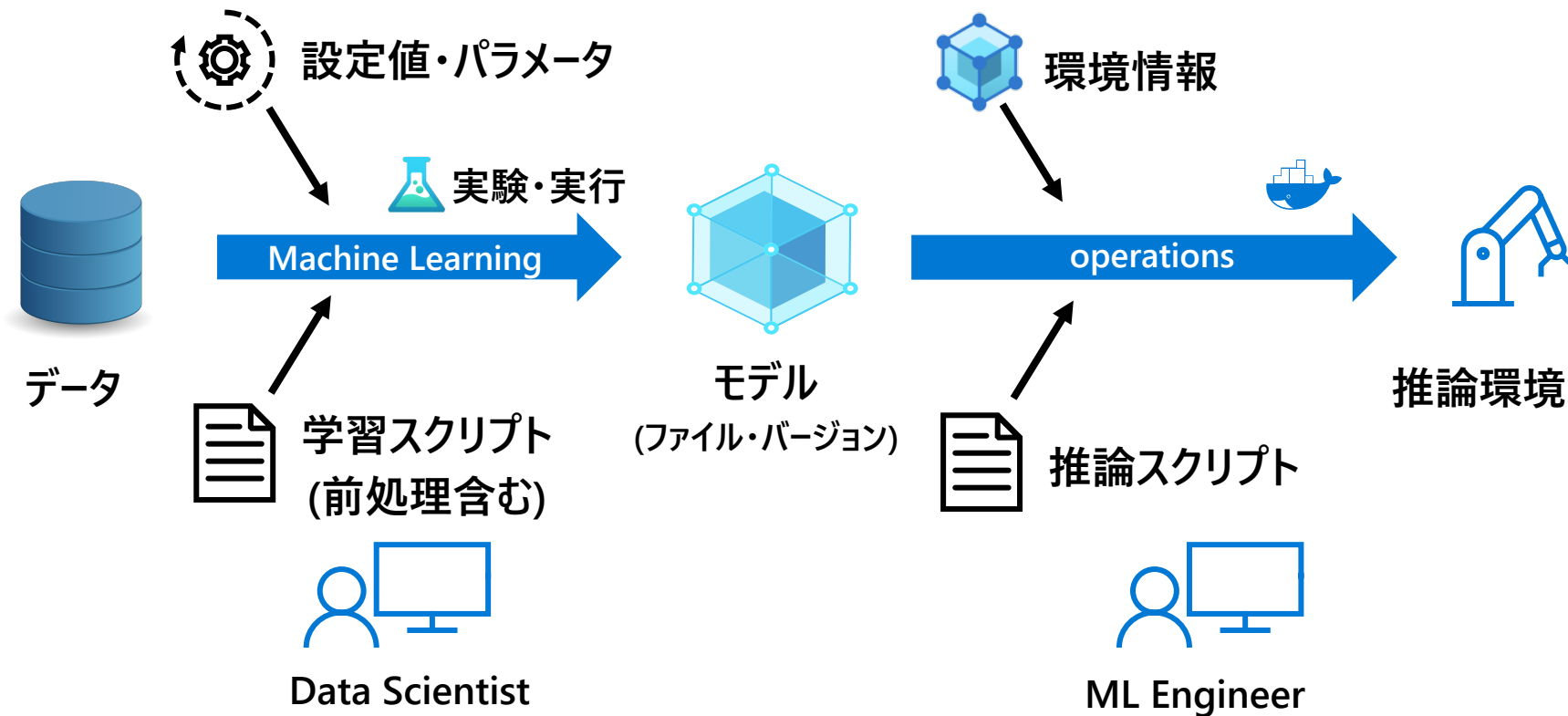
- ・ 実験 Experiment
- ・ 再現性 Reproducibility
- ・ デプロイ Deployment
- ・ モデル管理 Model Registry
  
- ・ API First
  - ・ あらゆる機械学習モデルに対応
  - ・ Python、R、Java、REST API
  
- ・ オンプレミス & クラウドで利用可能



> 9.9k stars, 319 Contributors on GitHub

# 実験追跡の実装方法の検討

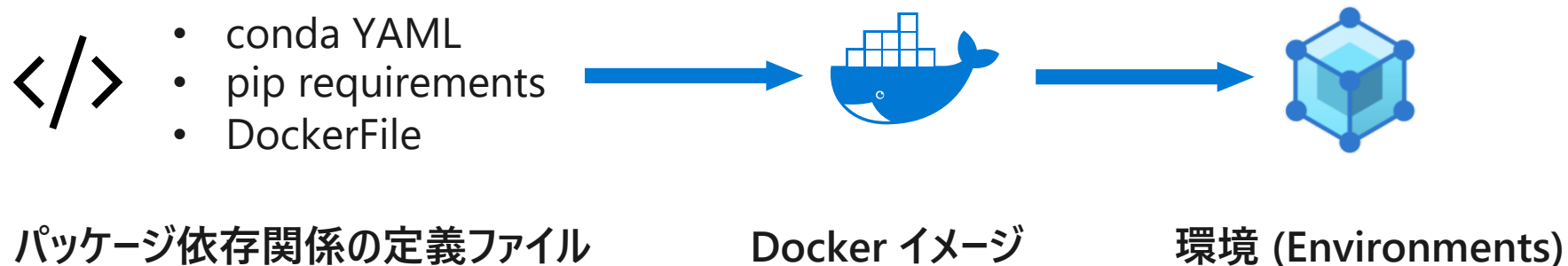
実験を追跡する仕組みを導入することで、モデル開発が効率的になったり、システム稼働しているモデルの透明性が高まります。



# 環境 (Environments) の構築方法の検討

環境 (Environments) の機能を活用することで再現性の高いモデル学習・モデル推論を行うことができるようになります。特に Python パッケージは非常に複雑な依存関係で構成されているものが多いため、環境の機能は重要な役割を果たします。

また、機械学習の実験では試行錯誤を行うため、固定のソフトウェア環境では利便性が落ちてしまうため、Data Scientist がある程度自由に操作可能な環境を検討する必要があります。



# 実験追跡の実装ガイドライン

Azure Machine Learning 上で実験を追跡方法の一覧を記載します。

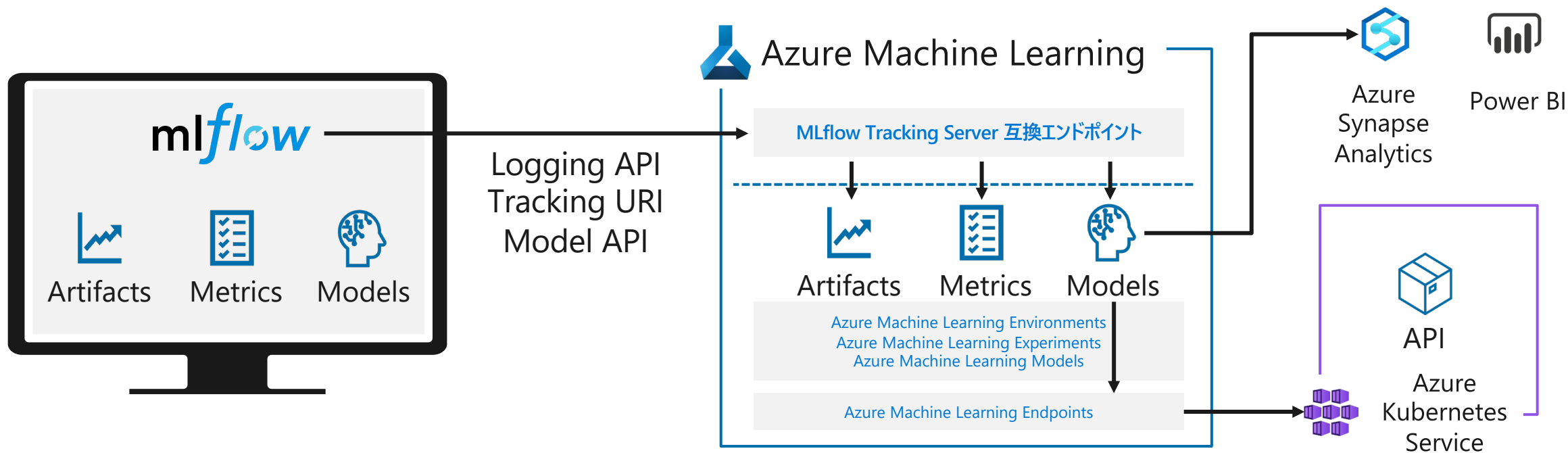
方法	対象	特徴
Azure ML Studio	実験サービス (AutoML, Designer)	モデルの精度など基本的なメトリックやログは自動で取得
Azure ML Python SDK	任意の Python コードで実装された実験	Azure ML の機能をフルサポート
MLflow (+Azure ML)	任意の Python コードで実装された実験	Auto Logging など高機能な追跡機能が利用可能
Third Party の技術	n/a	柔軟性はあるがユーザ自身で環境構築と運用が必要

初めて実験管理に取り組む場合は Azure ML Python SDK の利用を推奨します。また、MLflow を使い慣れていたたり、MLflow の便利な機能 (特に Auto Logging) を利用したい場合は MLflow の利用を検討ください。Azure CLI v2 + YAML でのジョブ実行 (Preview) のシナリオでは MLflow を用いることで Python SDK の記載が不要になる (コード変更が不要) というメリットもあります。

# 実験追跡の実装ガイドライン (cont'd)

## MLflow (+Azure ML)

Azure Machine Learning には MLflow Tracking Server 互換エンドポイントがあり、MLflow の API を使用して実験を追跡することができます。



実験

実験管理・記録

デプロイ



# 環境 (Environments) におけるセキュリティの考慮

多くの Python パッケージがインターネットへの接続を想定しており、セキュリティの観点での外部アクセスを制限するかどうかのポイントになります。生産性を高めるためにも**可能な限りインターネットへの送信通信を許可することを推奨**します。

直接のインターネット接続が許可されない場合のワークアラウンドとその長所・短所を挙げます。

ワークアラウンド	Pros	Cons
パッケージリポジトリに対する通信を許可	<ul style="list-style-type: none"><li>• pip や conda 等のコマンド、ライブラリをそのまま使用できる</li></ul>	<ul style="list-style-type: none"><li>• パッケージ単位での許認可は難しい</li></ul>
汎用の Proxy を用意して、許可したパッケージリポジトリのみ Proxy 経由での通信を許可	<ul style="list-style-type: none"><li>• pip や conda 等のコマンド、ライブラリをほぼそのまま使用できる</li><li>• ローカルキャッシュとして機能するため、サイズが大きいパッケージ・モデルのダウンロード時間の短縮が期待できる</li></ul>	<ul style="list-style-type: none"><li>• proxy の保守・運用をする必要がある</li><li>• パッケージ単位での許認可は難しい</li></ul>
ローカルのパッケージリポジトリとして機能する Proxy を用意し、承認されたパッケージのインストールを許可	<ul style="list-style-type: none"><li>• pip や conda 等のコマンドをほぼそのまま使用できる</li><li>• パッケージ単位でホワイトリスト方式で利用を許可することができる</li><li>• ローカルキャッシュとして機能するため、サイズが大きいパッケージのダウンロード時間の短縮が期待できる</li></ul>	<ul style="list-style-type: none"><li>• 自前で構築したりポジトリを保守・運用する必要がある</li><li>• huggingface 等、ライブラリが利用するリポジトリには適用できないため、別途 Proxy を用意する必要がある</li></ul>
オープン環境と閉域環境を用意し、検証済みパッケージを含むコンテナをオープン環境でビルドして閉域環境で使用	<ul style="list-style-type: none"><li>• パッケージ単位でホワイトリスト方式で利用を許可することができる</li><li>• 特定の学習・推論にあたって必要な全てのパッケージ・モデル等を含めることができ、制約が少ない</li></ul>	<ul style="list-style-type: none"><li>• パッケージ・モデルを追加する度にコンテナイメージのリビルドが必要になる</li><li>• コンピューティングインスタンスや JupyterLab のノートブック環境でパッケージを利用することが難しい</li></ul>

# MLOps

## 基礎知識

- MLOps 概要
- Azure ML - Pipelines

## 検討事項

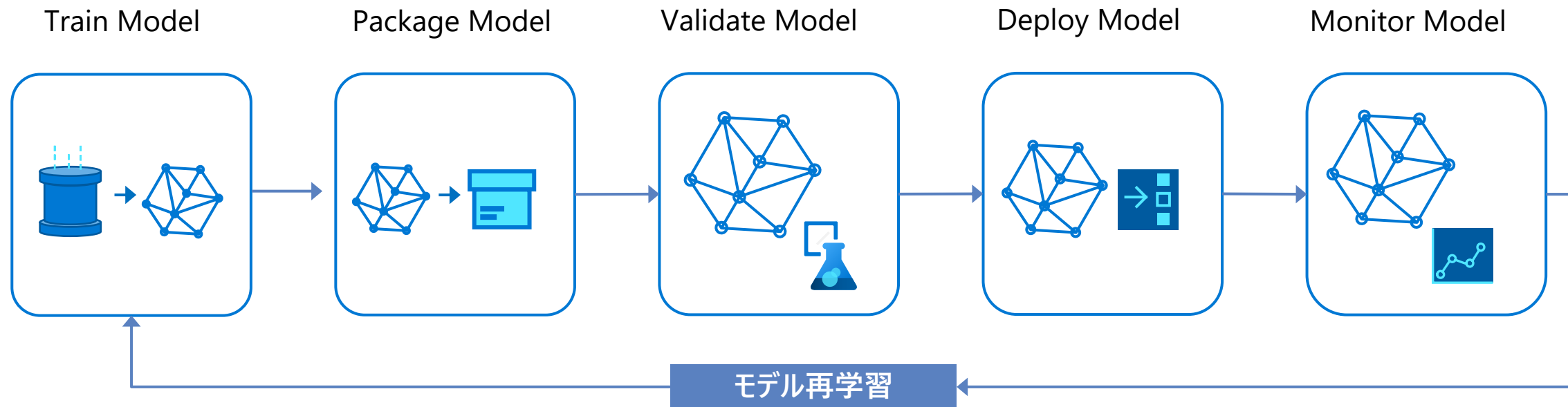
- パイプラインテクノロジーの選択
- 外部連携

## ガイドライン・実装手順

- MLOps 成熟度

# MLOps 概要

機械学習ライフサイクルを管理する手法・概念



## 開発と学習

ビジネス課題を解決する

## パッケージ化

モデルをどこでも使用可能にする

## 挙動の検証

応答性の観点と規制遵守の観点から

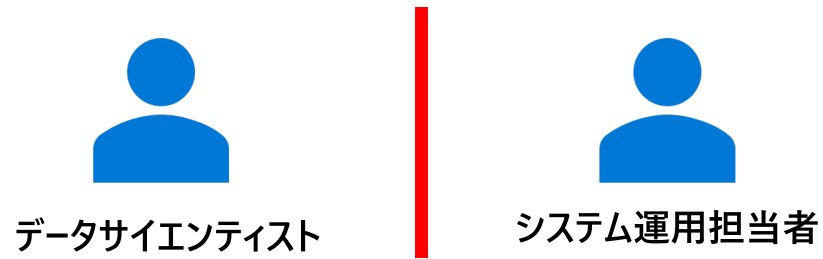
## デプロイ

予測値の生成にモデルを使用する

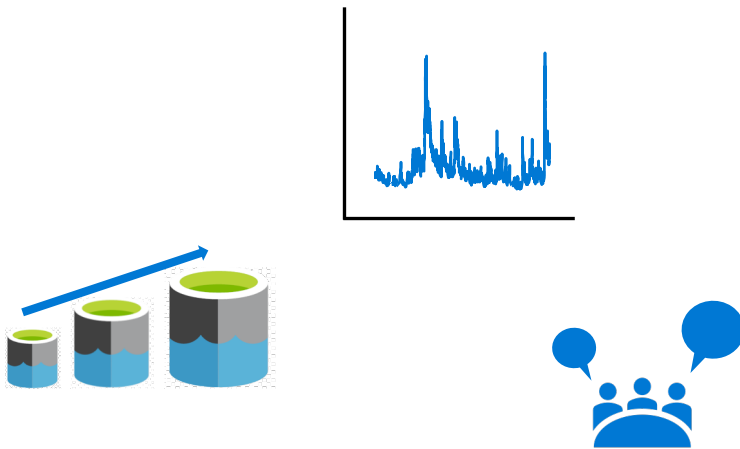
## モニタリング

挙動とビジネス価値を監視する  
陳腐化したモデルをいつ置換/廃止するか決める

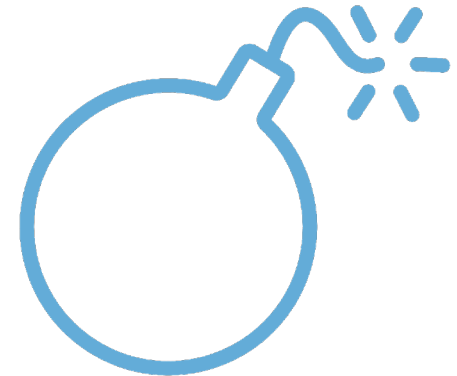
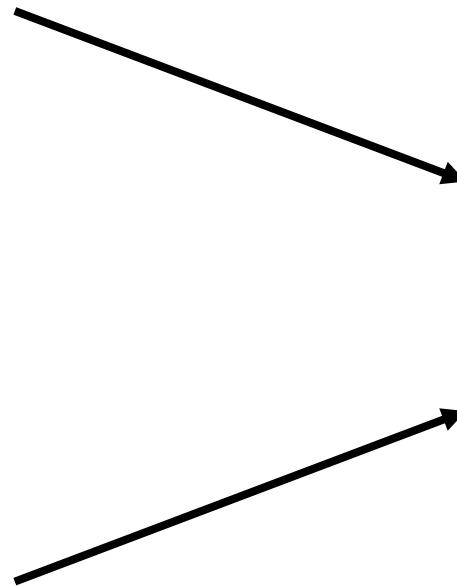
# MLOps 概要 (cont'd)



モデル作成者とシステム運用者の連携不足による機械学習モデル更新作業の停滞



データの増大、世の中の変化に伴うデータの変化、要件の変化



モデル劣化  
パフォーマンス劣化



# Pipelines

1つの Azure Machine Learning Pipeline は完成した機械学習タスクのワークフローを表し、機械学習の各サブタスクはパイプラインを構成する一連の手順として内包される

Azure Machine Learning Pipeline には、Python スクリプトを呼び出すだけのシンプルなものから、あらゆることを実行するものまで存在する。




## タスク

Pipelines は以下のような機械学習タスクにフォーカス:

- インポート、検証とクリーニング、変換、正規化、ステージングを含む「データの準備」
- パラメーター化された引数、ファイルのパス、ログとレポート出力の設定を含む「学習の設定」
- 効率的かつ繰り返し実行できる「学習と検証」。特定のデータサブセット、異なるハードウェアからなる計算リソース、分散処理、進捗状況の監視を指定することで、効率性を確保することが可能
- バージョン管理、スケーリング、プロビジョニング、アクセス制御等の要素を含む「デプロイ」

# パイプラインテクノロジーの選択

Azure や GitHub ではさまざまなパイプライン技術を提供しています。要件に応じて使い分ける必要があります。

シナリオ	ペルソナ	Azure/Microsoft	オープンソース	流れ	強み	
Model orchestration (Machine learning)	データサイエンティスト、機械学習エンジニア	Azure ML Pipelines	Kubeflow	Data → Model	分散、キャッシュ、コード優先、再利用	 Azure Machine Learning Pipeline
Data orchestration (Data prep)	データエンジニア	Azure Data Factory Pipelines, Azure Synapse Pipelines	Apache Airflow	Data → Data	厳密に型指定された移動、データ中心のアクティビティ	 Azure Data Factory Pipeline
Code & app orchestration (CI/CD)	アプリケーション開発者、DevOps エンジニア、機械学習エンジニア	Azure Pipelines, GitHub Actions	Jenkins	Code + Model → App/Service	ほとんどのオープンで柔軟なアクティビティのサポート、承認キュー、ゲートを使用したフェーズ	 GitHub Actions Azure Pipelines

# 外部連携

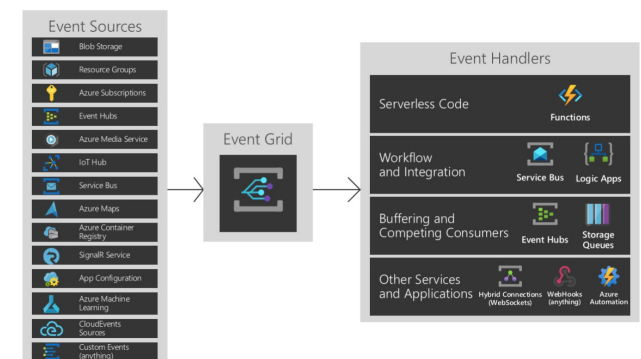
Azure Event Grid との連携機能によりイベントを検知し Azure Event Hubs, Azure Functions, Logic Apps などのイベントハンドラーに送信することができます。(Preview)

- Azure Machine Learning がサポートするイベント

イベントの種類	説明
Microsoft.MachineLearningServices.RunCompleted	機械学習実験の実行が完了したときに発生します。
Microsoft.MachineLearningServices.ModelRegistered	機械学習モデルがワークスペースに登録されたときに発生します。
Microsoft.MachineLearningServices.ModelDeployed	1 つ以上のモデルを持つ推論サービスのデプロイが完了したときに発生します。
Microsoft.MachineLearningServices.DatasetDriftDetected	2 つのデータセットのデータドリフト検出ジョブが完了したときに発生します。
Microsoft.MachineLearningServices.RunStatusChanged	実行状態が変更されたときに発生します。現時点では、実行状態が 'failed' の場合にのみ発生します。

- ユースシナリオ

- Job の成功をメールや Teams で通知する。
- データドリフトの検知を携帯電話の SMS で通知する
- CI/CD パイプラインをトリガーする。



# MLOps 成熟度

MLOps を実装する際の指針として、Microsoft が提供する MLOps 成熟度を参照ください。

	概要	技術
<b>Level 0</b> <b>No MLOps</b>	<ul style="list-style-type: none"><li>機械学習モデルのライフサイクル全体を管理することは困難</li><li>チームは別々で、リリースは困難</li><li>ほとんどのシステムは "ブラック ボックス" として存在し、デプロイ時およびデプロイ後のフィードバックはほとんどなし</li></ul>	<ul style="list-style-type: none"><li>手動によるビルドとデプロイ</li><li>モデルおよびアプリケーションの手動によるテスト</li><li>モデルのパフォーマンスの一元的追跡なし</li><li>モデル学習は手動</li></ul>
<b>Level 1</b> <b>DevOps no MLOps</b>	<ul style="list-style-type: none"><li>"No MLOps" よりもリリースの苦労は少ないが、新しいモデルごとにデータチームに依存</li><li>運用段階でのモデルのパフォーマンスに関するフィードバックは依然として限られる</li><li>結果の追跡および再現が困難</li></ul>	<ul style="list-style-type: none"><li>自動ビルド</li><li>アプリケーション コードの自動テスト</li></ul>
<b>Level 2</b> <b>Automated Training</b>	<ul style="list-style-type: none"><li>トレーニング環境は完全に管理され、追跡可能</li><li>モデルの再現が容易</li><li>リリースは手動であるが、摩擦は少ない</li></ul>	<ul style="list-style-type: none"><li>自動化されたモデルの学習</li><li>モデル学習のパフォーマンスを一元的に追跡</li><li>モデル管理</li></ul>
<b>Level 3</b> <b>Automated Model Deployment</b>	<ul style="list-style-type: none"><li>リリースは低摩擦で自動</li><li>デプロイから元のデータまで完全に追跡可能</li><li>環境全体 (学習 &gt; テスト &gt; 運用) を管理</li></ul>	<ul style="list-style-type: none"><li>デプロイするモデルのパフォーマンスに関する A/B テストを統合</li><li>すべてのコードのテストを自動化</li><li>モデルの学習性能を一元化</li></ul>
<b>Level 4</b> <b>Full MLOps Automated Retraining</b>	<ul style="list-style-type: none"><li>システムを完全自動化し、監視を容易化</li><li>運用システムは、改善方法に関する情報を提供。場合によっては、新しいモデルで自動的に改善</li><li>ゼロ ダウンタイム システムに近づく</li></ul>	<ul style="list-style-type: none"><li>モデル学習とテストを自動化</li><li>デプロイされたモデルからの詳細で一元化されたメトリック</li></ul>



# 責任のある AI

## 基礎知識

- 責任のある AI 概要
- Responsible AI Toolbox

## ガイドライン・実装手順

- Responsible AI Toolbox によるモデルのデバッグ

## Tips

- 責任のある AI アセスメント

# 責任のある AI

マイクロソフトは責任のある AI に関する 6 つの基本原則を実践しています。



Fairness



Reliability  
& Safety



Privacy &  
Security



Inclusiveness



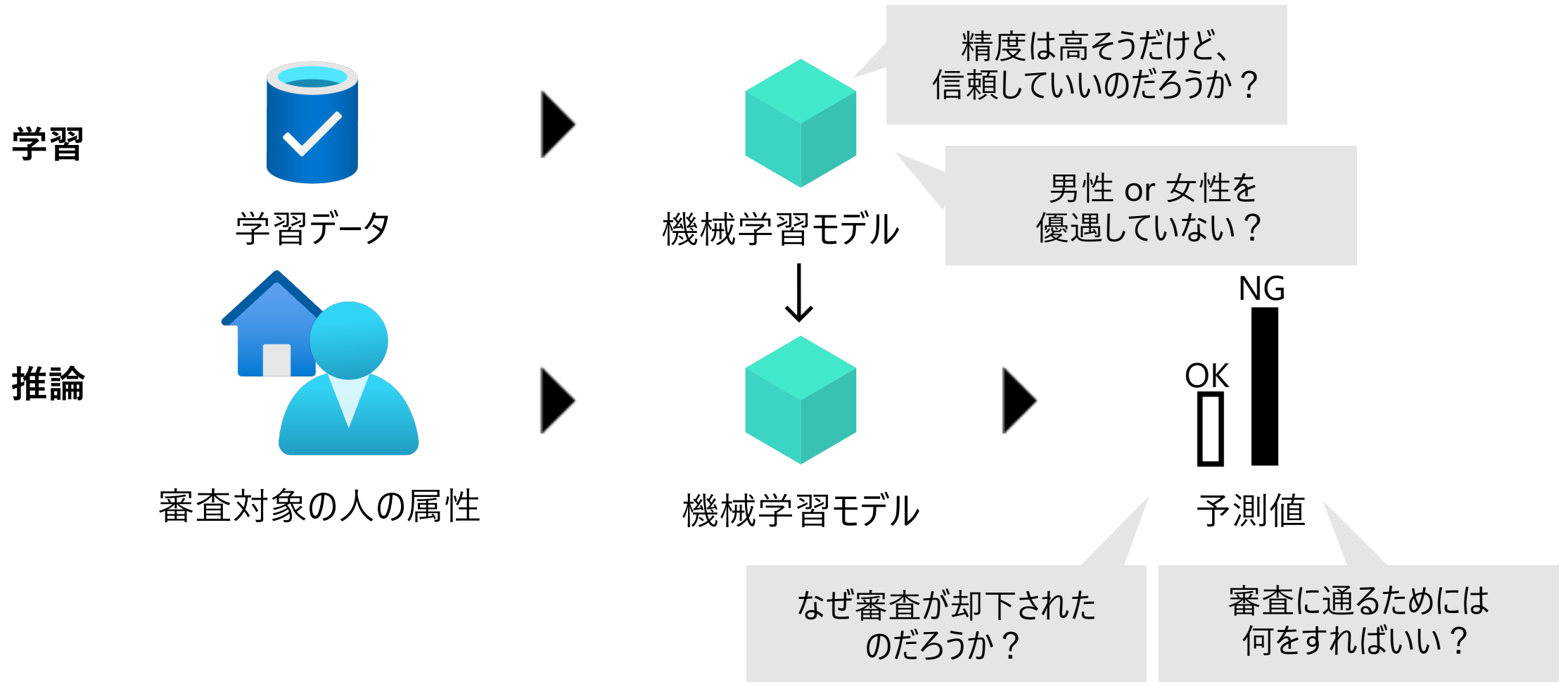
Transparency



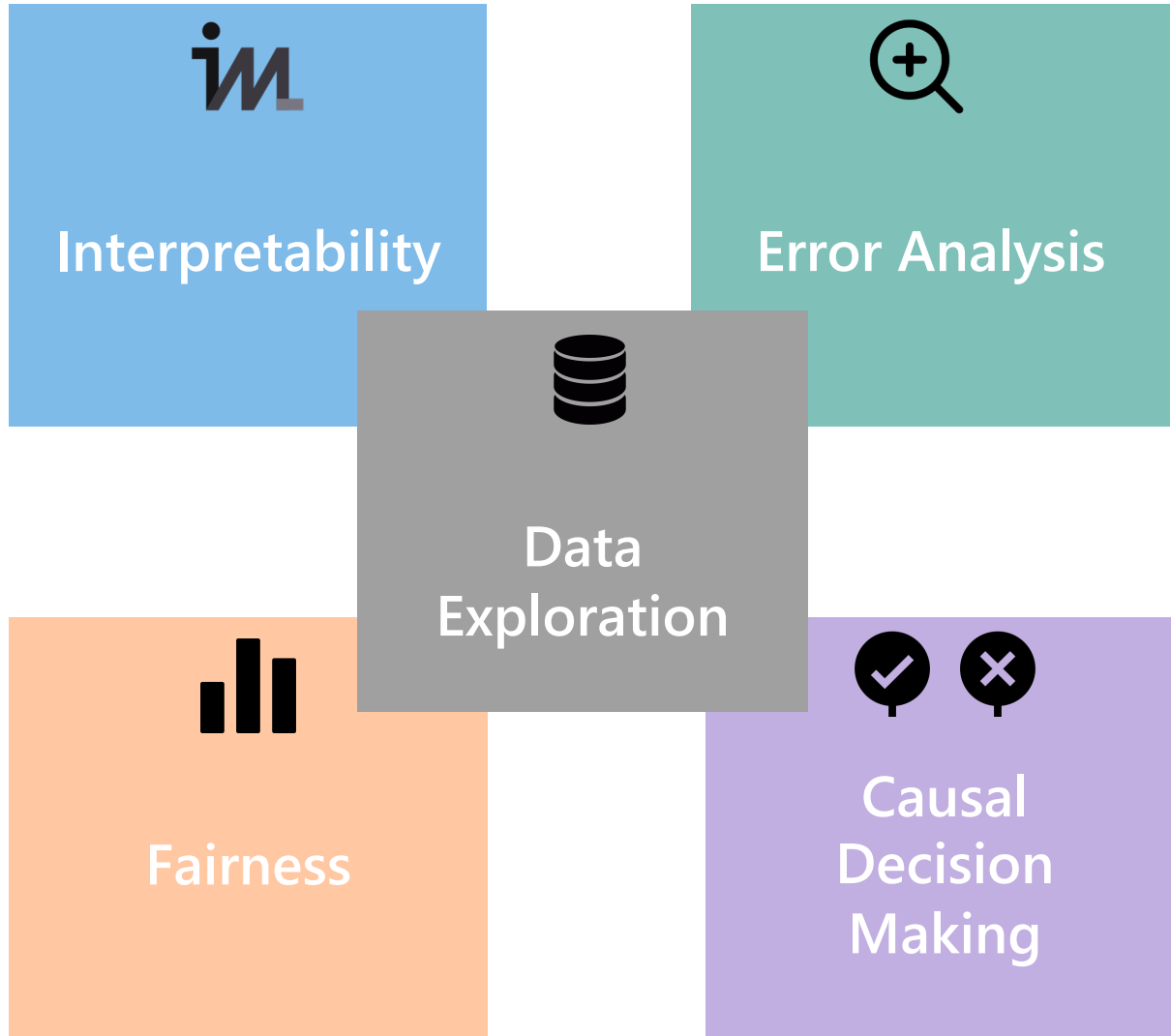
Accountability

# 責任のある AI (cont'd)

例：住宅ローンの審査におけるユースケース



# Responsible AI Toolbox



責任のある AI の開発を加速する  
統合的なオープンソースフレームワーク

- カスタマイズ性 + モジュール化
- エンドツーエンドのスムーズな実装
- インタラクティブ性
- 機械学習コードとの親和性

# Responsible AI Toolbox によるモデルのデバッグ

Responsible AI Toolbox を利用することで、機械学習モデルのデバッグを幾分か実施することができます。



機械学習モデルの問題 (バイアス、精度誤差) の存在を特定し、問題を深堀し原因を探り出します。その結果に基づいてモデルを修正したり、データを改善することで対処します。

# Responsible AI Toolbox によるモデルのデバッグ (cont'd)

フェーズ	コンポーネント	概要
特定	Error Analysis	機械学習の専門家にモデルエラー (誤差) に関する情報を提供し、誤った挙動をしたデータのコホートを特定することを支援します。
特定	Model Statistics	様々な観点でモデルの精度指標の算出と分析を支援します。
診断	Data Explorer	データを可視化する機能です。予測値や実測値、エラーグループ、特定の特徴量などの切り口で見ることができます。データの過不足や特徴を確認し理解するために使用します。
診断	Model Interpretability	機械学習モデルを解釈・説明する機能です。グローバルな説明とローカルな説明の2つの観点などのビューが提供されます。
診断	Counterfactual Analysis and What-If	モデルの予測値を変化させることができるような、最小限に変更されたデータセットを生成します。またインタラクティブに個々のデータポイントを変更し、予測値の変化を確認できます。このコンポーネントはDiCEをベースに開発しています。

# 責任のある AI のアセスメント

機械学習プロジェクトにおいて“責任のある AI”を遵守しているかを確認するための簡単なアセスメントを紹介します。

- Can the problem be addressed with a non-technical (e.g. social) solution?
- Can the problem be solved without AI? Would simpler technology suffice?
- Will the team have access to domain experts (e.g. doctors, refugees) in the field where the AI is applicable?
- Who are the stakeholders in this project? Who does the AI impact? Are there any vulnerable groups affected?
- What are the possible benefits and harms to each stakeholder?
- How can the technology be misused, and what can go wrong?
- Has the team analysed the input data properly to make sure that the training data is suitable for machine learning?
- Is the training data an accurate representation of data that will be used as input in production?
- Is there a good representation of all users?
- Is there a fall-back mechanism (a human in the loop, or a way to revert decisions based on the model)?
- Does data used by the model for training or scoring contain PII? What measures have been taken to remove sensitive data?
- Does the model impact consequential decisions, like blocking people from getting jobs, loans, health care etc. or in the cases where it may appropriate ethical considerations been discussed?
- Have measures for re-training been considered?
- How can we address any concerns that arise, and how can we mitigate risk?



Microsoft AI





© Copyright Microsoft Corporation. All rights reserved.