IT資格道場

AWS MLA-C01 学習用テキスト

予備知識ゼロから合格ラインへ —— 読んで理解する1冊目

無料サンプル(第1章まで)

AWS 公式 Exam Guide(MLA-C01・2026年9月6日取得)準拠
2026年7月版

本書のご利用にあたって

- 本書は「IT資格道場」(www.shikaku-doujou.com)の購入者限定テキストのうち、学習用テキストの第1章までを収録した**無料サンプル**です。どなたでもご覧いただけます。
- 無料サンプルであっても著作物です。本書の転載・改変しての再配布・二次販売を禁じます。
- 本書を AI モデルの学習データとして利用すること(学習目的の入力・データセット化を含む)を禁じます。
- 製品版(学習用テキスト・頻出度別ノート・暗記用ノートの3点)は、ご購入者を識別できる透かし入りのPDFとして提供されます。
- 本PDFはスマートフォン・タブレットでの閲覧に合わせたA5版面・文字サイズで組版しています。

不正な配布を見つけた場合は info@shikaku-doujou.com までご連絡ください。

目次

この教材群の使い方(3点セット)

種類	役割
① 学習用テキスト(本書)	これだけで問題が解けるようになる本編
② 頻出度別ノート	テキストを読んだら次はこれ。頻出順に絞った問題と解説で「解ける」に橋渡し。試験直前の最終チェックにも
③ 暗記用ノート	覚えるべき要点だけを一問一答に凝縮。空き時間の反復と用語の再確認用

使い方: ① 学習用を読む → ② 頻出度別で解き方を掴む → ③ 問題演習で腕試し → ④ 頻出度別に戻って再確認(試験直前もここ)。暗記用は空き時間や用語の再確認に。

このテキストの位置づけ: 本書は①の学習用テキストです。まずはここを通読し、これだけで問題が解けるようになることを目標にしてください。

はじめに ―― この教材の使い方

このテキストは、Amazon Web Services（AWS）が実施する **AWS Certified Machine Learning Engineer – Associate（試験コード MLA-C01）** の合格を目的に作られています。

この試験が問うのは、モデルの理論でも数式の導出でもありません。問われるのは、**データを取り込んで整え、モデルを学習させて評価し、要件に合う形でデプロイし、動き続ける仕組みとして監視・保守・保護できること**です。公式の試験ガイドも、対象外の職務として「エンドツーエンドの ML ソリューションの

設計」「ML 戦略の策定」「2つ以上の ML 領域の深い作り込み」「モデルの量子化と精度影響の分析」を明記しています。

本書は AWS の公式教材ではなく、IT資格道場が独自に書き下ろしたオリジナルの教材です。実際に出題された問題を再現したものではありません。

試験の基本情報

項目	内容
試験名	AWS Certified Machine Learning Engineer – Associate（MLA-C01）
実施	Amazon Web Services（Pearson VUE のテストセンター、またはオンライン監督試験）
問題数	**65問**（うち採点対象 50問・非採点 15問。どれが非採点かは受験者には分かりません）
試験時間	**130分**
出題形式	**択一（4肢1正解）／複数選択（5肢以上から2つ以上）／並べ替え（3〜5項目）／対応付け（3〜7組）**
合格ライン	**スケールスコア 100〜1,000 のうち 720**
採点方式	全体で合否を決める補償型。分野ごとの足切りはありません

受験料・試験時間・出題数・試験ガイドは改定されることがあります。お申し込みの前に、AWS 公式サイトで最新の情報をご確認ください。

想定される受験者

公式ガイドが想定するのは、**Amazon SageMaker と AWS の各サービスを使った ML エンジニアリングの経験が1年以上**、加えてバックエンド開発・DevOps・データエンジニア・データサイエンティストのいずれかの経験が1年以上ある人です。とはいえ本書は、**ML の用語と AWS の基礎から順に積み上げる構成**にしてあります。SageMaker を触ったことがなくても、章の順に読めば必要な知識がそろいます。

出題範囲の全体像 —— 4つのドメインと本書の章

ドメイン	分野	採点対象に占める割合（公式）	本書
1	機械学習のためのデータ準備	**28%**	第1章
2	ML モデル開発	**26%**	第2章
3	ML ワークフローのデプロイとオーケストレーション	**22%**	第3章
4	ML ソリューションの監視・保守・セキュリティ	**24%**	第4章

本書の章の並びは、試験ガイドの並びとまったく同じです。節見出しの末尾にある `[1.1]` のような番号は、試験ガイドの Task statement の番号です。公式ガイドを手元に置いて対照しながら読めます。AWS はドメインごとの割合だけを公表しており、Task 単位の出題数は公表していません。本書もそれ以上の数字は書きません。

サービス名・機能名は英語のまま覚える

本文では、**SageMaker の機能名・API 名・設定項目名を英語表記のまま**書いています（Data Wrangler、Feature Store、Clarify、Model Monitor、automatic model tuning、Inference Recommender、Model Registry …）。本番の設問文と選択肢に出るのはこの表記であり、日本語訳だけを覚えても画面では出会いません。英語の名前を見た瞬間に「どのドメインの、どの場面の機能か」が反射で出る状態を目指してください。

全設問に効く判断軸 ―― 「3つの問い」

問い	何を切り分けるか	分かれ方
問い1: どの工程の話か	データ準備／学習・評価／デプロイ／監視・保守	同じサービス名でも工程が違えば答えが変わります（例: Clarify は「学習前のバイアス」と「本番のドリフト」の両方に出ます）
問い2: 要件は何を最優先しているか	レイテンシ・スループット・コスト・運用負荷	エンドポイント種別（リアルタイム／サーバーレス／非同期／バッチ変換）は、この軸だけで一意に決まります
問い3: 最小権限・最小露出で満たせているか	IAM のスコープ・ネットワーク経路・ログの残し方	要件を満たす案が複数あるとき、正解はより狭い権限・より閉じた経路のものです

4ステップ学習法

購入者限定テキストは3冊で1セットです。

- **STEP 1（理解する）**: 本書「学習用テキスト」を通読し、4つのドメインの地図と3つの問いを頭に入れる

- **STEP 2（解き方を掴む）**：「頻出度別ノート」で頻出度の高い論点を解いて、解き方を掴む
- **STEP 3（腕試し）**：本サイトの問題演習で、本番と同じ4形式に慣れる
- **STEP 4（再確認）**：「頻出度別ノート」に戻って総ざらい。試験直前もここへ戻る

「暗記用テキスト」は本線には入れません。通勤時間や休憩時間など、**空き時間の反復と用語の再確認**に並走させてください。

それでは、第1章から始めます。

第1章 機械学習(ML)のためのデータ準備(ドメイン1・採点対象の28%)

この章は MLA-C01 で最大の配点を持つ分野です。問われるのは「S3 とは何か」ではなく、**与えられた制約(データ量・更新頻度・レイテンシ要件・コスト・コンプライアンス・運用負荷)のもとで、どの取り込み経路・どの保存先・どの変換ツールを選ぶと要件を満たすか**という実装判断です。

Practitioner 系の試験との違いははっきりしています。Practitioner は「SageMaker Feature Store とは何かを説明せよ」を問い、Associate の MLA-C01 は「**オンラインストアとオフラインストアのどちらを有効にし、どちらに何を書き、推論時にどちらから読むか**」を問います。したがって本章では、道具ごとに次の3つをそろえて覚えます。

1. **できること**(機能と、公式ドキュメントに書かれた具体的な数値・既定値)
2. **できないこと・制約**(誤答肢はここを突いてきます)
3. **他の候補との分かれ目**(どの条件で選択が切り替わるか)

試験は65問・130分・720点(スケールドスコア)で、択一(multiple choice)・複数選択(multiple response)に加え、**並べ替え(ordering)と対応付け(matching)**が出ます。並べ替えは「取り込み→変換→検証→特徴量登録→学習」のような**工程の順序**を、対応付けは「要件↔サービス」「指標↔用途」のような**対応関係**を問います。本章では順序と対応関係を意識して表で明示します。

シナリオ問題の読み方も先に固めておきます。**問題文の最後の1〜2文に「運用負荷を最小にして」「最も低コストで」「コードを書かずに」「ミリ秒のレイテンシで」といった評価軸が置かれ、そこが正解を一意に決めます。**技術的に成

立する選択肢が複数並ぶのが普通で、評価軸に照らして最良のものだけが正解です。

1-1 データを取り込み保存する [1.1]

この Task statement は、**生データがどこにあり、どの形式で、どうやって ML の作業場所まで運ぶか**を扱います。ML 固有の話に見えて、実際にはストレージとデータ移動の設計判断がそのまま問われます。

1-1-1 データ形式と取り込みの仕組み [1.1]

データ形式(data formats) は、試験で最も繰り返し問われる基礎です。まず**検証済み形式(validated formats)と非検証形式(non-validated formats)**の区別から入ります。

- **検証済み形式**……ファイル自身がスキーマ(列名・データ型)を持ち、読み込み時に型が保証される形式です。**Apache Parquet・Apache ORC・Apache Avro** が該当します。列の型が壊れたデータは書き込み時点で弾かれるため、下流のジョブが型エラーで落ちにくくなります。
- **非検証形式**……スキーマを持たず、読む側が型を推測する形式です。**CSV・JSON** が該当します。柔軟ですが、「数値のはずの列に空文字が混ざっていて学習が落ちる」といった事故が起きます。

各形式の性質を表で押さえます。**この表は「アクセスパターンから形式を選ぶ」設問(1-1-6)の根拠**になります。

形式	構造	圧縮	スキーマ	得意な読み方	典型的な用途
CSV	行指向・テキスト	なし(外部圧縮は可)	持たない	全列を頭から読む	小規模データ、他システムとの受け渡し、SageMaker組み込みアルゴリズムの入力
JSON	階層構造・テキスト	なし	持たない	1レコードずつ	API のレスポンス、ネストした半構造化データ
JSON Lines	1行1 JSON レコード	なし	持たない	ストリーミング的に1行ずつ	ログ、SageMaker の BlazingText・DeepAR の入力(`application/jsonlines`)
Apache Parquet	列指向・バイナリ	高い(列ごとに最適な符号化)	持つ	**必要な列だけ読む**	データレイクの分析、Athena・Redshift Spectrum・Spark の入力

形式	構造	圧縮	スキーマ	得意な読み方	典型的な用途
Apache ORC	列指向・バイナリ	高い	持つ	必要な列だけ読む、述語プッシュダウン	Hive 由来のエコシステム、EMR 上の分析
Apache Avro	**行指向**・バイナリ	中程度	持つ(スキーマをファイルに同梱)	1レコード全体を読む	**スキーマ進化**が必要なストリーミングのシリアライズ
RecordIO	レコード連結のバイナリコンテナ	—	—	順次読み出し	SageMaker AI の組み込みアルゴリズム、画像データ

> **よく出る取り違え**:「Avro は列指向」は誤りです。**Avro は行指向**で、列指向は **Parquet と ORC** です。したがって「1億行のうち200列から3列だけを毎日集計したい」は Parquet か ORC が正解で、Avro は不正解になります。逆に「レコード全体を丸ごと書き出し、スキーマが頻繁に変わるストリームを扱いたい」は Avro が正解です。

RecordIO(正確には protobuf recordIO)は SageMaker AI 固有なので、独立して覚える価値があります。公式ドキュメントは次のように定めています。

- protobuf recordIO 形式では、SageMaker AI は**データセットの各観測値を4バイト浮動小数点数(float)の集合としてバイナリ表現に変換**し、protobuf の values フィールドに格納します。

- 対応するコンテンツタイプは `application/x-recordio-protobuf` で、**Factorization Machines・K-Means・k-NN・Latent Dirichlet Allocation・Linear Learner・NTM・PCA・Random Cut Forest・Sequence-to-Sequence** が受け付けます。
- `application/x-recordio`(protobuf でない方)は **Object Detection アルゴリズム**が受け付けます。
- `text/csv` は **IP Insights・K-Means・k-NN・LDA・Linear Learner・NTM・PCA・RCF・XGBoost** が受け付けます。`text/libsvm` は XGBoost です。
- 画像系は `application/x-image` ・ `image/jpeg` ・ `image/png` (Object Detection、Semantic Segmentation)です。

CSV を SageMaker AI の組み込みアルゴリズムに渡すときの規約は、そのまま設問になります。

- **CSV ファイルにヘッダー行を含めてはいけません。**
- **目的変数(ターゲット)は先頭の列**に置きます。
- ターゲットを持たない教師なし学習では、コンテンツタイプでラベル列数を指定します(例: `content_type=text/csv;label_size=0`)。

> **よく出る取り違え**:「学習用の CSV は1行目に列名を書く」は SageMaker AI の組み込みアルゴリズムでは**誤り**です。「学習が始まらない/精度が異常に低い」という症状の原因としてヘッダー行とターゲット列の位置は頻出です。

取り込みの仕組み(ingestion mechanisms) は、大きく3系統に分かれます。この分類は 1-1-3 と 1-2-5 につながります。

仕組み	何をするか	代表的なサービス	遅延の目安
バッチ取り込み	一定量・一定時刻でまとめて運ぶ	AWS Glue の ETL ジョブ、AWS DataSync、Amazon EMR	分〜時間
ストリーミング取り込み	発生した順に流し続ける	Amazon Kinesis Data Streams、Amazon Data Firehose、Amazon MSK(Apache Kafka)	秒
イベント駆動取り込み	何かが起きた時だけ動く	Amazon S3 イベント通知 → Amazon EventBridge → AWS Lambda	秒

図 1-1　データ形式の2分類と、取り込みの仕組みの3系統

1-1-2 中核となる AWS のデータソース [1.1]

ML のデータ置き場として押さえるべき中核は **Amazon S3・Amazon EFS・Amazon FSx for NetApp ONTAP**(および FSx for Lustre)です。3つの性格の違いが、そのまま設問の分かれ目になります。

データソース	種別	アクセス方法	ML での主な役割	分かれ目
Amazon S3	オブジェクトストレージ	HTTPS の API(GetObject 等)	データレイクの本体。学習データ・モデル成果物・Feature Store のオフラインストア	既定の選択肢。POSIX のファイル操作が要らないなら S3
Amazon EFS	共有ファイルストレージ(NFS)	ファイルシステムとしてマウント	複数のノートブック・学習インスタンスから**同じディレクトリを同時に読み書き**したい時	「POSIX の共有ファイルシステムが要る」「学習前にすでに EFS にデータがある」
Amazon FSx for Lustre	高性能ファイルシステム	マウント	大規模データの反復学習で**スループットと IOPS を稼ぐ**時	「学習を何度も回し、S3 からの読み出しがボトルネック」
Amazon FSx for NetApp ONTAP	フル機能のファイルストレージ	**NFS・SMB・iSCSI・NVMe** のマルチプロトコル	オンプレの NetApp ONTAP 資産をそのまま AWS へ持ち込む時	「オンプレの NetApp から、アプリのコードを変えずに移行/バックアップ/バースト」

Amazon S3 の要点は次のとおりです。

- オブジェクトはバケットに置き、**キー(プレフィックス付きの名前)** で識別します。プレフィックス設計が後の読み出し性能とコストを左右します。
- **バージョニング**を有効にすると、上書き・削除しても以前のバージョンが残ります。学習データの再現性(どのバージョンで学習したか)を担保する手段として出ます。
- **ライフサイクル設定**でストレージクラスを自動遷移させ、保管コストを下げます(1-1-4)。
- **S3 Express One Zone** は単一アベイラビリティゾーンの高性能クラスで、**一桁ミリ秒のアクセス**を提供します。SageMaker AI のモデル学習は、S3 Express One Zone の**ディレクトリバケット**を **File mode・Fast file mode・Pipe mode の入力元として利用できます**。ただし、ディレクトリバケットへの SageMaker AI の出力データの暗号化は **SSE-S3 のみ**で、**SSE-KMS は現時点で未対応**です。

Amazon FSx for NetApp ONTAP の押さえどころは、公式ドキュメントに沿って次の点です。

- **NFS・SMB・iSCSI・NVMe** のマルチプロトコルアクセスに対応し、Linux・Windows・macOS のいずれからも使えます。
- **単一の名前空間でペタバイト規模**のデータセットに対応し、ファイルシステムあたり数十 GB/秒のスループットが出せます。
- **自動データ階層化**により、アクセス頻度の低いデータを低コストの階層(キャパシティプール)へ自動的に移し、SSD ストレージの料金は一部のデータにだけ払う形にできます。
- **圧縮・重複排除・コンパクション**でストレージ消費を減らせます。**SnapMirror レプリケーション**、**FlexCache**、**SnapLock の WORM**(Compliance モードと Enterprise モード)にも対応します。

- **Multi-AZ と Single-AZ** の配置を選べます。保存時の暗号化は **AWS KMS キー**、転送時の暗号化は SMB Kerberos・IPSEC・Nitro ベースの暗号化です。

> **よく出る取り違え**:「FSx for NetApp ONTAP は Linux 専用」は誤りです。**マルチプロトコル**が最大の特徴で、Windows の SMB クライアントからも同じデータを読めます。「Windows と Linux の両方から同じ学習データを触りたい」「オンプレの ONTAP をそのまま移したい」が出たらこれです。逆に「学習ジョブのスループットを最大化したい」だけなら FSx for Lustre です。

1-1-3 ストリーミングデータソースからの取り込み [1.1]

ストリーミング(streaming) の取り込みでは、**Amazon Kinesis・Apache Flink・Apache Kafka** の3つの名前が必ず出ます。役割が違うので、混ぜないことが第一です。

名前	正体	AWS 上での姿	何をするもの
Amazon Kinesis Data Streams	ストリームの**保管と配送**	マネージドサービス	生成されたレコードを順序付きで保持し、複数のコンシューマーに配る
Amazon Data Firehose	ストリームの**配送(サーバーレス)**	マネージドサービス	バッファしてから宛先へ届ける。コードを書かない配送路
Apache Flink	ストリームの**処理エンジン**	**Amazon Managed Service for Apache Flink**	ウィンドウ集計・結合・異常検知などをストリーム上で行う

名前	正体	AWS 上での姿	何をするもの
Apache Kafka	ストリームの**保管と配送**(OSS)	**Amazon MSK**(Managed Streaming for Apache Kafka)	Kafka の API 互換が要る、既存の Kafka 資産を移す
Amazon Kinesis Video Streams	映像ストリームの保管	マネージドサービス	カメラ映像を取り込み、画像系 ML の入力にする

Kinesis Data Streams の数値は、容量とスケーラビリティのトラブルシューティング(1-1-9)の根拠になるので、公式ドキュメントの値をそのまま覚えます。

項目	プロビジョンドモード	オンデマンドモード
シャードあたりの書き込み	最大 **1 MB/秒 または 1,000 レコード/秒**	シャードは自動でスケール
シャードあたりの読み出し	最大 **2 MB/秒 または 2,000 レコード/秒**（`GetRecords`）	同左
ストリームの既定スループット	プロビジョンしたシャード数に比例	新規は**書き込み 4 MB/秒・読み出し 8 MB/秒**から開始
オンデマンドの上限	—	バージニア北部・オレゴン・アイルランドは**書き込み 10 GB/秒・読み出し 20 GB/秒**まで、他リージョンは**書き込み 200 MB/秒・読み出し 400 MB/秒**まで自動スケール
レコードのペイロード上限	**base64 エンコード前で最大 10 MiB**	同左

項目	プロビジョンドモード	オンデマンドモード
`GetRecords` の1回の上限	**10 MB または 10,000 レコード**、シャードあたり**毎秒5回**の読み取りトランザクション	同左
保持期間	**最小24時間、最大 8,760 時間(365日)**	同左
登録コンシューマー数(拡張ファンアウト)	20	On-demand Advantage モードで最大50、標準は20
モードの切り替え	各データストリームにつき**24時間に2回**まで	同左

> **よく出る取り違え**:「シャードを増やせば読み出しも自動的に速くなる」は半分だけ正しい表現です。**シャードあたりの読み出しは 2 MB/秒で共有**されるため、コンシューマーが増えるほど1つあたりの取り分が減ります。「複数のアプリケーションが同じストリームを同時に低遅延で読みたい」の正解は**拡張ファンアウト(enhanced fan-out)の登録コンシューマー**で、コンシューマーごとに専用の 2 MB/秒が割り当てられます。シャード追加ではありません。

Amazon Data Firehose は、ML のデータ取り込みで最も出番の多い配送路です。

- 宛先は **Amazon S3・Amazon Redshift・Amazon OpenSearch Service・OpenSearch Serverless・Splunk・Apache Iceberg テーブル**、および任意の HTTP エンドポイント(Datadog、Dynatrace、MongoDB、New Relic、Elastic 等)です。
- **レコードは最大 1,000 KB** です。

- **バッファサイズ(MB)とバッファ間隔(秒)** のどちらかに達したら宛先へ配送します。**「遅延を短くしたい」ならバッファ間隔を小さく、「小さいファイルが大量にできるのを防ぎたい」ならバッファサイズを大きく**、という対応関係が問われます。
- **既存の Kinesis データストリームを自動で読み込む**設定ができます。つまり Kinesis Data Streams → Firehose → S3 が定番の並びです。
- 配送前に **AWS Lambda による変換**をかけられます。また Amazon Redshift 宛ての場合は、いったん S3 に届けてから **COPY コマンド**でロードします。
- 変換を有効にすると、**変換前のソースデータを別の S3 バケットへバックアップ**できます。「変換の不具合で元データを失いたくない」の正解です。

> **よく出る取り違え**: 「リアルタイムに1件ずつ処理したいので Firehose を使う」は誤りです。Firehose は**バッファしてから届ける**ので、必ず秒単位の遅延が入ります。**1件ずつ即座にカスタム処理したいなら Kinesis Data Streams + AWS Lambda、集計やウィンドウ処理が要るなら Amazon Managed Service for Apache Flink、コードを書かずに S3 へ貯めたいだけなら Amazon Data Firehose** という切り分けになります。

Apache Kafka(Amazon MSK)を選ぶ条件もはっきりしています。「すでにオンプレで Apache Kafka を運用しており、プロデューサー/コンシューマーのコードを変えずに移行したい」「Kafka Connect や Kafka Streams のエコシステムを使いたい」が出たら MSK です。新規構築で AWS ネイティブに寄せたいなら Kinesis です。

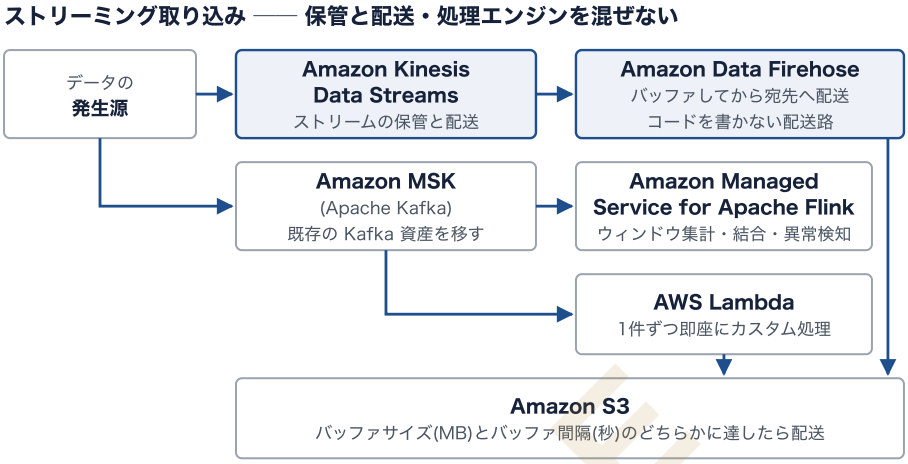


図 1-2　ストリーミング取り込み —— 保管と配送・処理エンジンを混ぜない

1-1-4 AWS のストレージ選択肢とトレードオフ [1.1]

AWS storage options の設問は、ほぼ **S3 のストレージクラス**と**ブロック/ファイル/オブジェクトの使い分け**の2本です。

まず**ストレージの3類型**を対応付けで押さえます。

類型	サービス	接続の形	ML での使いどころ	苦手なこと
オブジェクト	Amazon S3	HTTPS の API	学習データ、モデル成果物、オフラインストア	部分更新(オブジェクトは丸ごと置き換え)

類型	サービス	接続の形	ML での使いどころ	苦手なこと
ブロック	Amazon EBS	1つの EC2 インスタンスにアタッチ	ノートブックインスタンスの作業領域、DB のデータ領域	複数インスタンスからの同時共有(io2/io1 のマルチアタッチを除く)
ファイル	Amazon EFS、Amazon FSx	NFS/SMB でマウント	複数の学習インスタンスからの共有読み書き	単体の低コスト大量保管

S3 のストレージクラスは、公式ドキュメントの比較表がそのまま出題根拠です。

ストレージクラス	想定用途	耐久性	可用性	AZ 数	最小保存期間	最小課金オブジェクトサイズ	備考
S3 Standard	月に何度もアクセスし、ミリ秒で読みたいデータ	99.999999999%	99.99%	3以上	なし	なし	既定のクラス

ストレージクラス	想定用途	耐久性	可用性	AZ 数	最小保存期間	最小課金オブジェクトサイズ	備考
S3 Intelligent-Tiering	アクセスパターンが**不明・変動する**データ	99.999999999%	99.9%	3以上	なし	なし	**取り出し料金なし**。オブジェクトごとの監視・自動化料金がかかる。**128 KB 未満は監視対象外**で常に高頻度アクセス階層
S3 Standard-IA	長期保管・月1回程度のアクセス、ミリ秒で読みたい	99.999999999%	99.9%	3以上	**30日**	**128 KB**	GB あたりの取り出し料金がかかる

ストレージクラス	想定用途	耐久性	可用性	AZ 数	最小保存期間	最小課金オブジェクトサイズ	備考
S3 One Zone-IA	再作成可能で、たまにしか読まないデータ	99.999999999%	99.5%	**1**	**30日**	**128 KB**	AZ の消失に耐えられない
S3 Express One Zone	レイテンシに最も敏感な用途	99.999999999%	99.95%	**1**	なし	なし	一桁ミリ秒。ディレクトリバケットに格納
S3 Glacier Instant Retrieval	四半期に1回程度・ミリ秒で読みたいアーカイブ	99.999999999%	99.9%	3以上	**90日**	**128 KB**	取り出し料金がかかる
S3 Glacier Flexible Retrieval	年1回程度・**数分〜数時間**での取り出しでよいアーカイブ	99.999999999%	99.99%(復元後)	3以上	**90日**	—	**復元(restore)してからでないと読めない**

ストレージクラス	想定用途	耐久性	可用性	AZ 数	最小保存期間	最小課金オブジェクトサイズ	備考
S3 Glacier Deep Archive	年1回未満・**数時間**での取り出しでよいアーカイブ	99.999999999%	99.99%(復元後)	3以上	**180日**	—	最も低コスト。復元が必須

S3 Intelligent-Tiering の階層遷移は、日数がそのまま問われます。

階層	移動条件	取り出し
Frequent Access	アップロード時・遷移直後の既定	ミリ秒
Infrequent Access	**30日連続**でアクセスがない	ミリ秒
Archive Instant Access	**90日連続**でアクセスがない	ミリ秒
Archive Access(任意で有効化)	有効化後、**最低90日連続**でアクセスがない	分〜時間。`RestoreObject` が必要
Deep Archive Access(任意で有効化)	有効化後、**最低180日連続**でアクセスがない	時間単位。`RestoreObject` が必要

> **よく出る取り違え**:「アクセスパターンが読めないので S3 Standard-IA にしてコストを下げる」は誤りになりがちです。**Standard-IA は取り出しのたびに GB あたりの料金がかかる**ため、思ったよりアクセスされると Standard より高くつきます。**アクセスパターンが不明・変動するなら S3**

> **Intelligent-Tiering**(取り出し料金なし)が正解です。「**明確に月1回程度と分かっている**」なら Standard-IA です。

> **よく出る取り違え**:「学習データのアーカイブを S3 Glacier Deep Archive に置いておき、必要になったらすぐ学習に使う」は誤りです。**Glacier Flexible Retrieval と Glacier Deep Archive のオブジェクトは復元しないと読めません。**「アーカイブしつつ、必要な時はミリ秒で読みたい」の正解は **S3 Glacier Instant Retrieval** です。

AZ 単位の耐障害性も対応付けで問われます。**S3 One Zone-IA と S3 Express One Zone 以外のすべてのクラスは、AZ の物理的消失に耐えるよう設計されています。**

図 1-3　ストレージの3類型 —— 保存の単位と接続の形で分かれる

1-1-5 ストレージからのデータ抽出 [1.1]

Extracting data from storage は、**Amazon S3・Amazon EBS・Amazon EFS・Amazon RDS・Amazon DynamoDB** からどう取り出すか、そして取り出しが遅い/失敗する時にどのオプションで直すかを問います。

保管元	標準的な取り出し方	遅い/詰まる時の打ち手
Amazon S3	SDK/CLI の `GetObject`、Athena でのクエリ、Glue/Spark からの読み込み	**Amazon S3 Transfer Acceleration**(遠距離のアップロード)、マルチパートアップロード、プレフィックスの分散、S3 Express One Zone
Amazon EBS	インスタンスにアタッチしてファイルとして読む	**Amazon EBS Provisioned IOPS**(io2 Block Express / io1)へ変更、gp3 の IOPS/スループットを個別に引き上げ
Amazon EFS	マウントして読む	スループットモードの見直し、複数クライアントからの並列読み出し
Amazon RDS	SQL で抽出し、S3 へエクスポート	**リードレプリカ**から抽出して本番の負荷を避ける、Glue の JDBC 接続でパーティション並列読み
Amazon DynamoDB	`Scan` / `Query`、**S3 へのエクスポート**機能	`Scan` **は全項目を読むため重い**。抽出用途では S3 エクスポートか、Glue のコネクタを使う

Amazon S3 Transfer Acceleration の仕様は、公式ドキュメントに沿って次のとおりです。

- **バケット単位の機能**で、クライアントと S3 汎用バケットの間の**長距離転送**を高速化します。**Amazon CloudFront のエッジロケーション**を経由し、エッジに届いたデータを最適化されたネットワーク経路で S3 へ運びます。
- 使うべき場面は「**世界中の顧客が1つのバケットへアップロードする**」「大陸間で数 GB〜数 TB を定常的に転送する」「インターネット経由のアップロードで手持ちの帯域を使い切れない」です。
- 要件: **仮想ホスト形式のリクエストのみ対応、バケット名は DNS 準拠でピリオド(.)を含んではいけない**、バケットで有効化が必要、有効化から**最大20分**で速度が上がります。
- 専用エンドポイント `{bucket-name}.s3-accelerate.amazonaws.com`（IPv6 は `.s3-accelerate.dualstack.amazonaws.com`）を使います。通常のエンドポイントも引き続き使えます。
- 追加のデータ転送料金がかかる場合があります。

Amazon EBS のボリュームタイプも、性能要件との対応付けで問われます。

タイプ	種別	サイズ	最大 IOPS	最大スループット	典型的な用途
gp3	汎用 SSD	1 GiB〜64 TiB	**80,000**	**2,000 MiB/秒**	一般的なトランザクション、ブートボリューム、開発/テスト。**IOPS とスループットをサイズと独立に設定できる**
gp2	汎用 SSD	1 GiB〜16 TiB	16,000	250 MiB/秒 (サイズ依存で 128〜250)	旧世代の汎用。IOPS がサイズに連動する
io2 Block Express	プロビジョンド IOPS SSD	4 GiB〜64 TiB	**256,000**	**4,000 MiB/秒**	**一貫したサブミリ秒のレイテンシ**、80,000 IOPS 超、耐久性 99.999%
io1	プロビジョンド IOPS SSD	4 GiB〜16 TiB	64,000	1,000 MiB/秒	16,000 IOPS 超が要る I/O 集約型データベース
st1	スループット最適化 HDD	125 GiB〜16 TiB	500	500 MiB/秒	**ビッグデータ、データウェアハウス、ログ処理**

タイプ	種別	サイズ	最大 IOPS	最大スループット	典型的な用途
sc1	Cold HDD	125 GiB〜16 TiB	250	250 MiB/秒	アクセス頻度が低く、**最も低コスト**にしたいスループット志向の保管

> **よく出る取り違え**:「大量の連続読み出しを速くしたいので io2 にする」は評価軸を外しています。**大きなブロックを連続で読むワークロード(ログ処理・ビッグデータ)は st1** が費用対効果で勝ちます。**io2/io1 を選ぶのは「小さな I/O を高い IOPS で、一貫した低レイテンシで」**という条件が付いた時です。なお **st1 と sc1 はブートボリュームに使えません**。

1-1-6 アクセスパターンに応じたデータ形式の選択 [1.1]

「どの形式を選ぶか」は、**読み方(アクセスパターン)から逆算**します。この対応表がそのまま設問の答えになります。

アクセスパターン	選ぶ形式	理由
200列のうち3列だけを毎日集計する	**Parquet** または **ORC**	列指向なので**必要な列のブロックだけ読む**。スキャン量が減り、Athena の課金(スキャンした GB)も下がる
1レコード全体を都度読み書きし、スキーマが頻繁に変わる	**Avro**	行指向でスキーマをファイルに同梱するため、**スキーマ進化**に強い
外部システムとの受け渡し・人が目視する	**CSV**	どこでも読める。ただしスキーマを持たない

アクセスパターン	選ぶ形式	理由
ネストした半構造化データをそのまま保持する	**JSON**(または JSON Lines)	階層構造を保てる
SageMaker AI の組み込みアルゴリズムで大量の数値データを学習する	**RecordIO-protobuf**	4バイト float のバイナリで、CSV よりも読み込みが速く小さい
画像を大量に扱う	**RecordIO**(または JPEG/PNG のまま)	小さいファイルを大量に置くと読み出しのオーバーヘッドが大きくなるため、まとめる

よく出る取り違え:「CSV を gzip 圧縮すれば Parquet と同じ」は誤りです。**gzip 圧縮した CSV は分割できない(splittable でない)** ため、Spark や Athena が並列に読めず、1ファイルを1タスクが読む形になります。**Parquet と ORC は列単位で読め、かつ分割可能**です。「Athena のクエリが遅く、スキャン量が多い」の正解は**列指向形式への変換＋パーティション化**です。

パーティション化(partitioning) も同じ文脈で問われます。S3 のキーを `year=2026/month=09/day=06/` のように区切ると、クエリが不要なプレフィックスを読み飛ばせます。**AWS Glue のジョブや Amazon Data Firehose の動的パーティショニング**でこの構造を作ります。

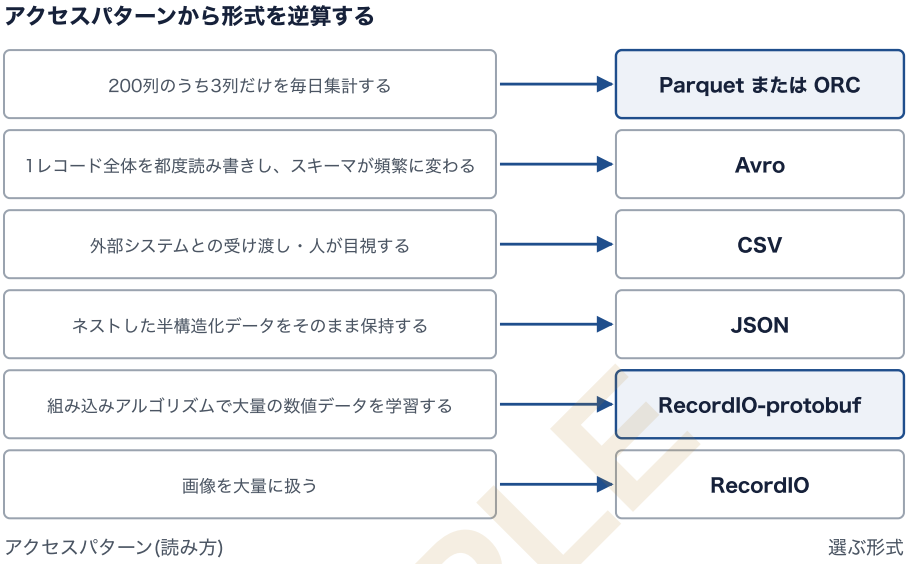


図 1-4　アクセスパターンから形式を逆算する

1-1-7 SageMaker Data Wrangler と Feature Store への取り込み [1.1]

SageMaker Data Wrangler は、公式ドキュメントによれば **SageMaker Studio Classic の機能**で、データの**インポート・準備・変換・特徴量化・分析**を一気通貫で行う道具です(現在は SageMaker Canvas に統合された新しい体験も提供されています)。

Data Wrangler が接続できるインポート元は次のとおりです。**この一覧はそのまま設問になります。**

- **Amazon S3**
- **Amazon Athena**

- **Amazon Redshift**
- **Snowflake**
- **Databricks**

Data Wrangler の主要機能は5つです。

機能	内容
Import	上記のデータソースに接続して取り込む
Data Flow	一連のデータ準備ステップを**データフロー**として定義する。複数ソースの結合もここで行う
Transform	文字列・ベクトル・数値の標準的な変換、テキストや日時の埋め込み、カテゴリのエンコーディング
Generate Data Insights	**Data Insights and Quality Report** でデータ品質を自動検証し、異常を検出する
Analyze	散布図・ヒストグラムなどの可視化、**ターゲットリーケージ分析**、**クイックモデル**による特徴量の重要度確認

Data Wrangler のエクスポート先も対応付けで頻出です。

エクスポート先	何のために
Amazon S3	変換済みデータセットとして保存し、学習ジョブの入力にする
Amazon SageMaker Pipelines	変換をパイプラインの一工程として自動実行する
Amazon SageMaker Feature Store	作った特徴量を中央のストアに登録し、再利用する

エクスポート先	何のために
Python スクリプト	変換処理をコードとして持ち出し、独自のワークフローに組み込む

SageMaker Feature Store は、特徴量の**作成・保存・共有・管理**を担います。仕組みの中心は次の3語です。

- **フィーチャーグループ(FeatureGroup)**……特徴量の集まり。表に見立てると、各列が特徴量で、各行が一意の識別子を持ちます。**作成後もスキーマを進化させられます(mutable)**。名前は**リージョンとアカウント内で一意**です。
- **レコード(Record)**……1つの **RecordIdentifier** に対応する特徴量の値の集まりです。
- **イベント時刻(event time)** ……そのレコードがいつの状態かを表す時刻です。オフラインストアはこれに基づくプレフィックスで保存されます。

オンラインストアとオフラインストアの違いは、この Task statement の最重要点です。

項目	オンラインストア	オフラインストア
保持するもの	**最新のレコードだけ**	**すべてのレコード**(履歴データベース)
想定用途	**リアルタイム推論**。低ミリ秒のレイテンシの読み出しと高スループットの書き込み	**データ探索・モデル学習・バッチ推論**
実体	マネージドの低レイテンシストア	**自分の S3 バケット**。**Parquet 形式**で保存
書き込みの性質	上書き(最新値)	**追記のみ(append-only)**

項目	オンラインストア	オフラインストア
クエリ	API での読み出し、フィーチャーグループをまたいだ結合	**Amazon Athena** でクエリ可能

取り込み(ingestion)の2つの経路も、そのまま並べ替え問題になります。

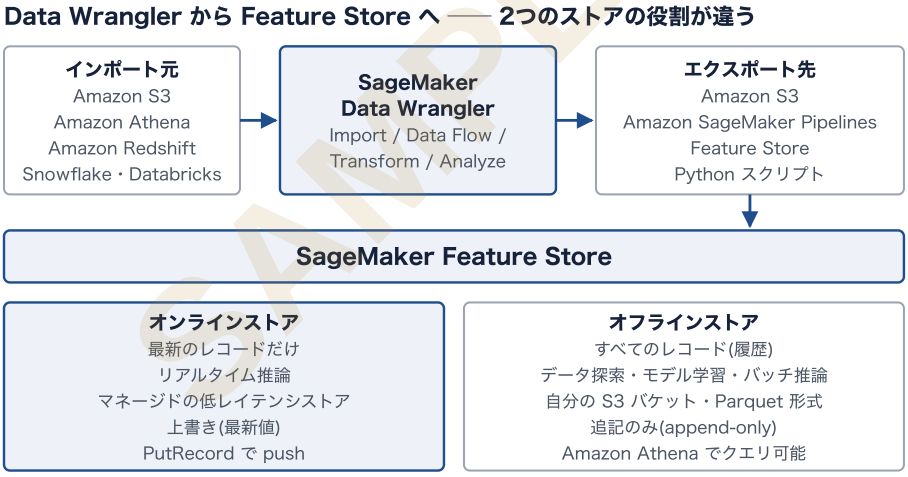
1. **ストリーミング取り込み**……同期 API の `PutRecord` を呼んでレコードを push します。同期呼び出しなので、1回の呼び出しで小さなバッチの更新を送れます。**更新を検知した瞬間に最新の特徴量値を反映**でき、これを**ストリーミング特徴量(streaming features)** と呼びます。**Amazon MSK(Apache Kafka)や Amazon Kinesis** をソースにして、抽出した特徴量を直接オンラインストアへ流し込めます。
2. **バッチ取り込み**……Data Wrangler からエクスポートしたノートブックを **SageMaker Processing ジョブ**として実行し、まとめて取り込みます。**バッチ取り込みはオフラインストアへの取り込みを可能にし**、フィーチャーグループがオンラインとオフラインの両方に設定されていればオンラインストアへの取り込みもできます。**100万行以上の大きなバッチ**にも対応します。

レコードの削除は `DeleteRecord` API で行います。**これはオンラインストアから消すと同時に、削除されたレコードをオフラインストアへ追加します**(オフラインは追記のみだからです)。

Feature Store が解く問題も覚えておきます。**学習時とサービング時で特徴量の作り方がずれる「トレーニング/サービングスキュー(training-serving skew)」を減らす**ことです。同じ定義の特徴量を、学習でも推論でも同じストアから読むことで防ぎます。

よく出る取り違え:「オフラインストアから低レイテンシでリアルタイム推論用の特徴量を読む」は誤りです。オフラインストアは S3 上の Parquet であり、**リアルタイム推論に要るのはオンラインストア**です。逆に「過去のある時点の特徴量でモデルを学習し直したい」はオフラインストア(履歴を全部持っている)です。

重要: フィーチャーグループの名前や、説明・タグなどのメタデータに、**個人を特定できる情報(PII)や機密情報を入れてはいけません**と公式ドキュメントが明示しています。1-3-4 のマスキングの話と対で問われます。



ストリーミング取り込みは同期 API の PutRecord、バッチ取り込みは Processing ジョブ。
DeleteRecord はオンラインストアから消すと同時に、オフラインストアへ追加する。
解く問題はトレーニング/サービングスキュー(training-serving skew)を減らすこと。

図 1-5　Data Wrangler から Feature Store へ —— 2つのストアの役割が違う

1-1-8 複数ソースのデータを結合する [1.1]

Merging data from multiple sources は、**プログラミング技法・AWS Glue・Apache Spark** の3つの手段の使い分けです。

手段	何をするか	向く条件
プログラミング技法 (programming techniques)(pandas の merge/join、SQL の JOIN 等)	ノートブックや Lambda 上で結合する	データが小さく、探索段階。単発の作業
AWS Glue	サーバーレスの ETL。**Data Catalog** にスキーマを持ち、**クローラ**で自動的にテーブル定義を作る	定常的な ETL、複数の異種データソース、インフラを管理したくない
Apache Spark	分散処理エンジン。**Amazon EMR** 上、または Glue の実行基盤として動く	大規模データ、既存の Spark コード資産、細かいチューニングが要る
SageMaker Data Wrangler	データフロー上で GUI で結合する	ML の前処理として、少ないコードで試行錯誤したい

結合で必ず考えることを3つ挙げます。

- **キーの整合**……結合キーの型・大文字小文字・前後の空白がずれていると、行が落ちます。事前に正規化します。
- **粒度の一致**……「1顧客1行」のテーブルと「1顧客N行」のテーブルを内部結合すると行が膨らみます。集約してから結合するか、結合後に重複排除(1-2-1)します。
- **時点の一致**……特徴量の時刻とラベルの時刻がずれると、**未来の情報が特徴量に混ざる(ターゲットリーケージ)** 事故が起きます(1-3-8)。

1-1-9 取り込み・保存のトラブルシューティング(容量とスケーラビリティ) [1.1]

Troubleshooting and debugging は、**症状 → 原因 → 打ち手**の対応表で覚えます。ここは実装・運用の粒度で問われる、MLA-C01 らしい領域です。

症状	見るべき指標・ログ	ありがちな原因	打ち手
Kinesis への書き込みが `ProvisionedThroughputExceededException` で失敗する	CloudWatch の `WriteProvisionedThroughputExceeded`、`IncomingBytes`、`IncomingRecords`	**シャードあたり 1 MB/秒・1,000 レコード/秒**を超えている。またはパーティションキーが偏っている(ホットシャード)	シャードを追加(`UpdateShardCount`)、**オンデマンドモードへ切り替え**(24時間に2回まで)、パーティションキーを分散させる
Kinesis の読み出しが5秒待たされる/例外になる	`GetRecords.IteratorAgeMilliseconds`	`GetRecords` が **10 MB を返した後、5秒以内の後続呼び出しは例外**。またはコンシューマー数が多く 2 MB/秒を食い合っている	**拡張ファンアウト**の登録コンシューマーにする、消費側を並列化する
S3 へのアップロードが遅い(海外拠点から)	転送時間、帯域の利用率	長距離のインターネット経路	**S3 Transfer Acceleration** を有効化、**マルチパートアップロード**で並列化
S3 のリクエストが `503 Slow Down` になる	S3 のリクエストメトリクス	プレフィックスあたりのリクエスト集中	**プレフィックスを分散**させる、リトライにエクスポネンシャルバックオフを入れる

症状	見るべき指標・ログ	ありがちな原因	打ち手
学習ジョブが「ディスク容量不足」で落ちる	学習ジョブのログ	**File mode は全データセットをインスタンスのボリュームへダウンロードする**ため、収まらない	**Fast file mode か Pipe mode** へ変更する、インスタンスのボリュームを大きくする、`ShardedByS3Key` で分散する
学習の開始までが極端に長い	学習ジョブの開始時刻	File mode で巨大データセットを全部ダウンロードしている	**Fast file mode**(ダウンロードを待たずに開始できる)へ変更する
Glue のジョブが OOM やタイムアウトで落ちる	Glue のジョブメトリクス、Spark UI	データの偏り(スキュー)、小さいファイルが大量、ワーカー不足	ワーカー数/タイプを上げる、**小さいファイルをまとめる**、パーティション数を調整する
DynamoDB からの抽出が本番に影響する	消費キャパシティ	`Scan` **が全項目を読んでいる**	**S3 へのエクスポート**機能を使う、オンデマンドキャパシティにする
EBS のディスク I/O が頭打ち	CloudWatch の `VolumeQueueLength`、`BurstBalance`	gp2 のバーストクレジット枯渇、IOPS 上限	**gp3 に変更して IOPS/スループットを個別に上げる**、**Provisioned IOPS(io2/io1)** にする

> **誤答肢の切り方**:「Kinesis の書き込みエラーの対処」で「Amazon Data Firehose に切り替える」という選択肢は、**書き込み側のスループット不足の解決になっていません**。「レコードのサイズを 10 MiB 以下にする」も、既

> に守っているなら無意味です。**症状の指標(どのメトリクスが上限に張り付いているか)を先に確定してから打ち手を選ぶ**、が読み方です。

1-1-10 コスト・性能・データ構造に基づく初期ストレージ判断 [1.1]

最後に、**Making initial storage decisions** を1枚の判断表にまとめます。並べ替え・対応付けの根拠になります。

条件	選ぶもの	理由
構造化・半構造化データを大量に貯め、後から分析も学習もする	**Amazon S3**(Parquet) + Glue Data Catalog	最も安く、あらゆるサービスから読める
アクセス頻度が読めない	**S3 Intelligent-Tiering**	取り出し料金なしで自動最適化
複数の学習インスタンスから POSIX で同時に読む	**Amazon EFS**	共有ファイルシステム
学習を何度も回し、読み出しスループットが律速	**Amazon FSx for Lustre**	数百 GB/秒のスループットと数百万 IOPS
Windows と Linux の双方から同じデータを触る/オンプレ NetApp から移す	**Amazon FSx for NetApp ONTAP**	NFS・SMB・iSCSI・NVMe のマルチプロトコル
キーで1件ずつ超低レイテンシに引く(推論時のルックアップ)	**Amazon DynamoDB**	一桁ミリ秒のキーバリューアクセス
SQL の結合・トランザクションが要る業務データ	**Amazon RDS**	リレーショナル
大規模な分析クエリを SQL で回す	**Amazon Redshift**	列指向の MPP データウェアハウス

条件	選ぶもの	理由
全文検索・ログ分析	**Amazon OpenSearch Service**	検索インデックス
監査のため長期保管するが、ほぼ読まない	**S3 Glacier Deep Archive**	最安。ただし復元が必要・最小180日

コストの見方も一行で押さえます。ストレージの費用は「**保管料金 + リクエスト料金 + 取り出し料金 + データ転送料金**」の合計です。**AWS Cost Explorer** で実績を分解し、**AWS Budgets** で閾値超過を通知し、**AWS Trusted Advisor** で使われていないリソースを洗い出し、**AWS Compute Optimizer** でインスタンスの過剰プロビジョニングを見つけます。この4つの役割分担は対応付けで出ます。

1-2 データを変換し特徴量エンジニアリングを行う [1.2]

ここからは、集めたデータを**モデルが学習できる形に整える**工程です。統計的な技法の名前と、それを AWS のどの道具で実行するか、の両方が問われます。

1-2-1 データクレンジングと変換の技法 [1.2]

Data cleaning の中心は4つです。**外れ値(outliers)の検出と処理・欠損値の補完(imputing missing data)・結合(combining)・重複排除(deduplication)**。

外れ値(outliers) の検出方法を対応表で押さえます。

手法	判定の仕方	向く条件
標準偏差法(zスコア)	平均から標準偏差の3倍を超える値を外れ値とする	分布が正規分布に近い
四分位範囲(IQR)法	第1四分位数 − 1.5×IQR より小さい、第3四分位数 + 1.5×IQR より大きい値	分布が歪んでいてもよい。箱ひげ図と対応する
Random Cut Forest	木構造でデータ点の「切り離しやすさ」を測り異常スコアを出す	教師なしで多次元の異常を検出したい。SageMaker AI の組み込みアルゴリズムにある

外れ値の処理は3択で、シナリオの目的で切り替わります。

処理	内容	選ぶ条件
除去(削除)	該当行を捨てる	明らかな入力ミス・センサー故障で、件数が少ない
クリッピング(ウィンザライズ)	上限・下限の値に丸める	値は異常だが、その行の他の情報は使いたい
変換	**対数変換(log transformation)** などで裾を圧縮する	分布そのものが右に長い裾を持つ(所得・価格など)

欠損値の補完(imputation) も、選択の理由まで問われます。

手法	内容	注意点
平均値補完	数値列を平均で埋める	外れ値に引きずられる。分散が小さくなる
中央値補完	数値列を中央値で埋める	歪んだ分布に強い

手法	内容	注意点
最頻値補完	カテゴリ列を最頻値で埋める	カテゴリの偏りが強まる
k近傍法(k-NN)補完	似た行の値から推定して埋める	精度は高いが計算コストが高い
多重代入	複数の補完値を作り、不確実性ごと扱う	実装が重い
行を落とす	欠損のある行を削除する	欠損がランダムでないと**選択バイアス**を生む(1-3-7)
「欠損」という値にする	カテゴリとして `missing` を割り当てる	欠損していること自体に意味がある時に有効

> **よく出る取り違え**:「欠損が多い列は全部平均で埋める」は誤りになりがちです。**欠損の割合が非常に高い列は、埋めるより列ごと落とす**方が良いことが多く、また**欠損していること自体が予測に効く**なら、欠損フラグ列を作るのが正解です。「補完の方法を決める前に、なぜ欠損したのかを見る」が読み方です。

重複排除(deduplication) は、**完全一致の重複**と**ほぼ一致の重複(表記ゆれ)** を分けて考えます。完全一致は `DISTINCT` や `drop_duplicates` で足りますが、表記ゆれは正規化(大文字小文字・全角半角・空白の統一)を先に行います。**AWS Glue の FindMatches**(機械学習トランスフォーム)は、完全一致しないレコードの名寄せに使えます。

結合(combining) は 1-1-8 のとおりで、キー・粒度・時点の3点を確認します。

1-2-2 特徴量エンジニアリングの技法 [1.2]

Feature engineering techniques は名前と効果の対応がそのまま出ます。

技法	何をするか	いつ使うか
データスケーリング(data scaling)	値の範囲をそろえる	距離を使うアルゴリズム(k-NN、K-Means、SVM)や勾配降下法を使うモデルで必須
正規化(normalization / Min-Max スケーリング)	最小0・最大1の範囲に収める	値の上下限が決まっている時。外れ値に弱い
標準化(standardization / zスコア)	平均0・標準偏差1に変換する	分布が正規分布に近い時。**外れ値に比較的強い**
特徴量の分割(feature splitting)	1つの列を複数に割る(住所→都道府県/市区町村、日時→年/月/日/曜日/時間帯)	元の列のままでは意味が取れない時
ビニング(binning / 離散化)	連続値を区間に区切ってカテゴリにする(年齢→10代/20代…)	非線形な関係を線形モデルに取り込みたい時、外れ値の影響を抑えたい時
対数変換(log transformation)	値の対数を取る	右に長い裾を持つ分布(価格・所得・アクセス数)を正規分布に近づける
多項式特徴量・交互作用項	列同士を掛け合わせる	特徴量の組み合わせが効く時

正規化と標準化の分かれ目は頻出なので言い切ります。

	正規化(Min-Max)	標準化(zスコア)
変換後の範囲	0〜1 に収まる	範囲は固定されない(平均0・標準偏差1)

	正規化(Min-Max)	標準化(zスコア)
外れ値の影響	**強く受ける**(最大値が外れ値だと他が潰れる)	比較的受けにくい
向く場面	ニューラルネットワークの入力、画像の画素値	分布が正規分布に近い、外れ値がある

> **よく出る取り違え**:「木ベースのモデル(決定木・ランダムフォレスト・XGBoost)でもスケーリングが必要」は誤りです。**木ベースのモデルは分割の閾値で判断するため、スケーリングの影響を受けません。**「XGBoost の精度が出ないのでデータを正規化する」という選択肢は、ほぼ誤答肢です。**スケーリングが要るのは、距離や勾配を使うアルゴリズム**です。

スケーリングの適用順序も並べ替えで問われます。**必ず学習データと検証/テストデータを分割してから、学習データだけでスケーラーを学習(fit)し、その同じスケーラーで検証/テストデータを変換(transform)します。**全データでスケーラーを学習すると、テストデータの情報が学習に漏れます(**データリーケージ**)。

1-2-3 エンコーディングの技法 [1.2]

Encoding techniques は、カテゴリ変数やテキストを数値に直す作業です。

技法	何をするか	次元数	向くもの	落とし穴
ワンホットエンコーディング(one-hot encoding)	カテゴリごとに0/1の列を作る	カテゴリ数と同じだけ増える	**順序のない名義尺度**(都道府県、色)	**カーディナリティが高い列で次元が爆発**する

技法	何をするか	次元数	向くもの	落とし穴
ラベルエンコーディング(label encoding)	各カテゴリに整数を割り当てる	1列のまま	**順序のある順序尺度**(小/中/大)、木ベースのモデル	**順序のないカテゴリに使うと、モデルが大小関係を学んでしまう**
バイナリエンコーディング(binary encoding)	カテゴリを整数にしてから2進数の各桁を列にする	log2(カテゴリ数) 程度	**カーディナリティが高い名義尺度**	解釈しにくい
トークン化(tokenization)	テキストを単語やサブワードの単位に分ける	語彙サイズに依存	自然言語のテキスト	言語や分かち書きの方針に依存する

高カーディナリティの扱いは対応付けで頻出です。

カテゴリ数	推奨	理由
少ない(数個〜十数個)	**ワンホットエンコーディング**	素直で、線形モデルでも扱える
多い(数百〜)	**バイナリエンコーディング**、ハッシュ化、または埋め込み(embedding)	次元の爆発を避ける
順序がある	**ラベルエンコーディング**(順序どおりに番号を振る)	大小関係が意味を持つ

> **よく出る取り違え**:「郵便番号をラベルエンコーディングして線形回帰に入れる」は典型的な誤りです。郵便番号に大小関係は無いのに、モデルは「番号が大きいほど値が大きい」と学習してしまいます。**順序のないカテゴ**

> **リにラベルエンコーディングを使ってよいのは、木ベースのモデルなど、数値の順序を大小として解釈しないモデルに限られます。**

トークン化(tokenization) の先には、テキストを数値ベクトルにする手法が続きます。試験では名前と性質の対応で出ます。

手法	内容
Bag of Words	単語の出現回数を数える。語順を捨てる
TF-IDF	頻出しすぎる語の重みを下げ、その文書に特徴的な語を強調する
N-gram	連続するN語をひとまとまりとして扱い、語順の情報を一部残す
単語埋め込み(word embedding)	意味の近い語が近いベクトルになるよう学習する。SageMaker AI の BlazingText が生成できる

1-2-4 データを探索・可視化・変換するツール [1.2]

SageMaker Data Wrangler・AWS Glue・AWS Glue DataBrew の3つが並びます。**この3つの分かれ目は、MLA-C01 で最頻出の対応付け**です。

ツール	誰が使う想定か	コードの要否	得意なこと	ML との距離
SageMaker Data Wrangler	データサイエンティスト・ML エンジニア	ほぼ不要(Python の追加も可)	**ML 前処理に特化**。データ品質と分析のレポート、**ターゲットリーケージ分析**、**クイックモデル**、**Feature Store へのエクスポート**	**最も近い**。SageMaker Studio 内で完結する
AWS Glue DataBrew	ビジネスアナリスト・データエンジニア	**不要(ノーコード)**	**250種類以上**の既成の変換、プロファイリング、**PII の検出とマスキング**	中間。汎用のデータ準備
AWS Glue(ETL ジョブ)	データエンジニア	**必要**(PySpark / Scala、ビジュアル ETL もあり)	大規模な定常 ETL、Data Catalog、クローラ、**AWS Glue Data Quality**	遠い。汎用のデータ統合基盤

AWS Glue DataBrew の仕様は、公式ドキュメントに沿って次のとおりです。

- **コードを書かずに**データをクレンジング・正規化できる**ビジュアルなデータ準備ツール**です。カスタム開発の前処理に比べ、準備にかかる時間を**最大80パーセント**削減できるとされています。
- **250種類を超える既成の変換**から選べます。異常値のフィルタリング、標準的な形式への変換、無効値の修正などです。

- **サーバーレス**で、クラスターの作成やインフラの管理なしにテラバイト規模の生データを扱えます。
- 使い方は「**プロジェクト**を作ってデータに接続 → グリッド状の画面で値の分布とチャートを見る → 変換を選ぶ → 変換の前後を即座にプレビュー → **レシピ**として保存」という流れです。レシピは**更新・再利用**でき、他のデータセットにも適用できます。
- **自然言語処理(NLP)の技法**を使って文を句に分割する変換もあります。
- レシピをデータセット全体に実行した結果は **Amazon S3 に保存**されます。

> **よく出る取り違え**:「コードを書けないアナリストにデータのクレンジングを任せたい」で **AWS Glue の ETL ジョブ**を選ぶのは誤りです。**ノーコードなら DataBrew、ML の前処理で Feature Store まで繋ぐなら Data Wrangler、大規模で定常的な ETL のパイプラインなら Glue** という切り分けになります。

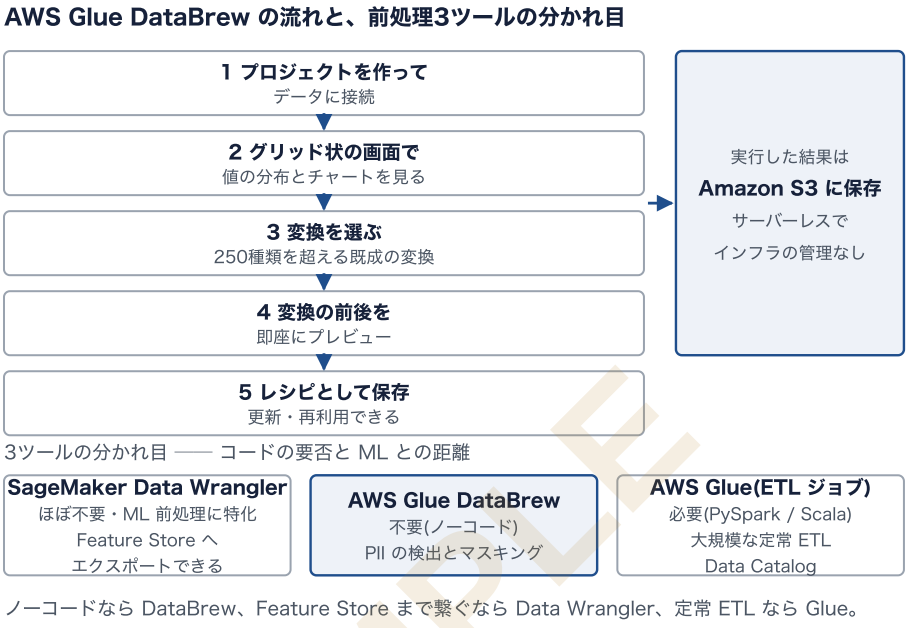


図 1-6　AWS Glue DataBrew の流れと、前処理3ツールの分かれ目

1-2-5 ストリーミングデータを変換するサービス [1.2]

Services that transform streaming data の中核は **AWS Lambda** と **Spark** です。ここに Flink を加えた3択で、条件によって答えが変わります。

変換の仕方	サービス	向く条件	苦手なこと
レコード単位の軽い変換	**AWS Lambda**	1件ずつのフィルタ・整形・項目追加。Amazon Data Firehose の変換としても呼べる	状態を持った集計、ウィンドウ処理

変換の仕方	サービス	向く条件	苦手なこと
ウィンドウ集計・結合・状態を持つ処理	**Amazon Managed Service for Apache Flink**	「直近5分間の平均」「セッション単位の集計」「複数ストリームの結合」	単純な整形にはやや重い
大規模なバッチ/マイクロバッチ処理	**Spark**(Amazon EMR 上、または Glue)	大量データの変換、既存の Spark コード、Structured Streaming	秒未満の低遅延

AWS Lambda を選ぶ判断材料は、「**サーバーレスで運用負荷が最小**」「イベント駆動」「短時間で終わる処理」です。**Amazon Data Firehose のデータ変換として Lambda を指定する**構成は定番で、Firehose が S3 へ届ける前に整形・フィルタ・JSON から Parquet への前処理などを差し込めます。

Spark を選ぶ判断材料は、「**分散処理が要る規模**」「既存の PySpark/Scala の資産がある」「複雑な結合や集計」です。実行基盤は **Amazon EMR**(クラスターを自分で持つ。細かいチューニング可)か **AWS Glue**(サーバーレス。運用負荷が小さい)を選びます。

> **誤答肢の切り方**:「ストリーム上で直近1時間の移動平均を計算して異常を検知したい」で **AWS Lambda** を選ぶのは、**Lambda が状態を持たない**ため不適切です(自前で DynamoDB 等に状態を持てば可能ですが、運用負荷が増えます)。この条件の正解は **Amazon Managed Service for Apache Flink** です。逆に「1件ずつ個人情報のフィールドを落としたいだけ」で Flink を選ぶのは過剰で、**Lambda** が正解です。

1-2-6 高品質なラベル付きデータセットを作るアノテーション/ラベリングのサービス [1.2]

Data annotation and labeling services は3つを役割で分けます。

サービス	何をするか	使いどころ
Amazon SageMaker Ground Truth	ラベリングジョブを作り、人間のワーカーにラベル付けをさせる。**自動データラベリング**も併用できる	学習用データセットをゼロから作る
Amazon Mechanical Turk	世界中の独立した作業者に細かいタスクを委託する仕組み	機密性が低く、大量に安く回したいラベリング
Amazon Augmented AI(Amazon A2I)	**推論結果**のうち低信頼度のものを人間にレビューさせる	運用中のモデルの出力を人が確認する

Ground Truth と A2I の分かれ目を言い切ります。**Ground Truth は「学習前のデータにラベルを付ける」、A2I は「学習後のモデルが出した予測を人がレビューする」**です。工程の位置が違います。

Amazon A2I の要点は次のとおりです。

- **低信頼度の予測**、または**ランダムに抽出した予測サンプル**を人間のレビューに回します。
- **フローディフィニション(人間によるレビューのワークフロー定義)** を作り、条件に合致したら**ヒューマンループ**が起動します。
- 連携先は **Amazon Textract**(単一ページ文書の重要なキーと値のペア)、**Amazon Rekognition**(不適切画像の判定)、**SageMaker AI のエンドポイントにデプロイした自前のモデル**、**Amazon Comprehend**(感情

分析・構文・エンティティ検出)、**Amazon Transcribe**、**Amazon Translate**、および表形式データです。

- レビュー結果を使ってモデルを**インクリメンタルに再学習**できます。

1-2-7 AWS のツールでデータを変換する [1.2]

Transforming data by using AWS tools は、**AWS Glue・DataBrew・Amazon EMR 上の Spark・SageMaker Data Wrangler** の4択を、要件で選び分ける設問になります。1-2-4 の表に「規模」と「運用負荷」の軸を足して判断します。

要件の言い回し	正解	なぜ他が違うか
「コードを書かずに」「アナリストが自分で」	**AWS Glue DataBrew**	Glue と EMR はコードが要る。Data Wrangler は ML 前処理寄りで、Studio の利用が前提
「運用負荷を最小に」「サーバーレスで」定常 ETL	**AWS Glue**	EMR はクラスターの管理が要る
「既存の Spark ジョブをそのまま」「Hadoop エコシステム(Hive・HBase・Presto)も使う」	**Amazon EMR**	Glue は Spark のマネージド実行だが、Hadoop エコシステム全体は EMR
「ML の前処理をして、そのまま Feature Store と学習パイプラインへ繋ぐ」	**SageMaker Data Wrangler**	他はエクスポート先の統合が弱い
「クラスターのコストを細かく最適化したい」「スポットインスタンスを使いたい」	**Amazon EMR**	インスタンスの購入オプションを自分で選べる

Amazon EMR を選んだ場合のコスト最適化も、対応付けで問われます。

購入オプション	向くワークロード
オンデマンド	短期・不定期で、中断が許されない
リザーブドインスタンス / Savings Plans	常時稼働のマスター/コアノード。1年または3年の稼働が読める
スポットインスタンス	**中断されても再実行できる**タスクノード。大幅に安い

> **よく出る取り違え**:「EMR のコストを下げるため、すべてのノードをスポットインスタンスにする」は誤りです。**マスターノードやコアノード(HDFS のデータを持つ)がスポットで中断されるとクラスターやデータが壊れます**。スポットにしてよいのは**タスクノード**です。

Amazon SageMaker Processing も、この文脈で覚えます。SageMaker AI が管理するインスタンス上で前処理・後処理・評価のスクリプトを実行する仕組みで、**SageMaker Pipelines の一工程**として組み込めます。Data Wrangler からエクスポートしたバッチ取り込みの処理も Processing ジョブとして走ります(1-1-7)。

1-2-8 AWS のツールで特徴量を作成・管理する [1.2]

Creating and managing features by using AWS tools の主役は **SageMaker Feature Store** です。仕様は 1-1-7 に書いたとおりなので、ここでは**運用の観点**を足します。

特徴量を再利用可能にする仕組みは3つです。

1. **フィーチャーグループの発見**……作成後、権限のある他の利用者が**フィーチャーグループ名・説明・レコード識別子名・作成日・タグ**で検索して見つ

けられます。「チーム間で特徴量を共有し、同じものを何度も作り直すのをやめたい」の正解です。

2. **メタデータ**……短い説明、ストレージ設定、レコードを識別する特徴量、イベント時刻に加え、**作者・データソース・バージョン**などをタグとして持てます。
3. **スキーマの進化**……フィーチャーグループは作成後も**変更可能(mutable)** で、スキーマを進化させられます。

特徴量の履歴と時点指定の取り出しも押さえます。オフラインストアは**追記のみ**で全レコードの履歴を保持し、**Feature Store は異なる時点のデータを抽出**して、学習・検証・テスト用のデータセットを作ることを支援します。これが「**過去のある時点で分かっていた情報だけを使って学習し直す**」という要求への答えです。

耐障害性も一行で出ます。**Feature Store は複数のアベイラビリティゾーンに分散**しており、一部の AZ に障害があっても他の AZ を使えます。

> **誤答肢の切り方**: 「特徴量を再利用したいので、変換済みの CSV を S3 の共有バケットに置いて周知する」という選択肢は、**技術的には可能でも、定義の一元管理・発見可能性・学習とサービングの一致という3点を満たしません。**「運用負荷を最小にして特徴量を組織で共有する」の正解は **SageMaker Feature Store** です。

1-2-9 AWS のサービスでデータを検証しラベル付けする [1.2]

Validating and labeling data by using AWS services は、**SageMaker Ground Truth と Amazon Mechanical Turk** の関係が中心です。

Amazon SageMaker Ground Truth の仕様は、公式ドキュメントに沿って次のとおりです。

- **Amazon Mechanical Turk のワーカー、自分で選んだベンダー企業、社内のプライベートな作業者**のいずれか(または組み合わせ)と機械学習を使って、ラベル付きデータセットを作ります。
- ワークフォースの3択は次のとおりです。
  - **Amazon Mechanical Turk のワークフォース**……**世界中の50万人を超える独立した作業者**。
  - **プライベートワークフォース**……**組織内でデータを扱うため**に、自社の従業員や契約者から作る。
  - **ベンダー企業**……**AWS Marketplace** で見つけられる、データラベリングを専門とする企業。
- ML のアプリケーションに応じて、**組み込みタスクタイプ(built-in task types)** を選べます。独自の UI とツールを提供する**カスタムラベリングワークフロー**も作れます。
- **自動データラベリング(automated data labeling)** は、**どのデータを人間がラベル付けする必要があるかを機械学習で判断する**仕組みで、ラベリングの時間と手作業を減らせます。
- **ラベリング UI テンプレート**は、タスクと指示をワーカーに提示する Web ページです。組み込みテンプレートから始めるか、**HTML 2.0 のコンポーネント**で自作します。
- データセットは S3 バケットに置き、バケットには**ラベル付け対象のデータ・入力マニフェストファイル・出力マニフェストファイル**の3つが入ります。**出力マニフェストファイルにラベリングジョブの結果が入ります**。

- ラベリングジョブのイベントは **Amazon CloudWatch の `/aws/sagemaker/LabelingJobs` グループ**に出ます。ログストリーム名はラベリングジョブ名です。
- **ストリーミングラベリングジョブ**を作ると、**常時稼働するラベリングジョブ**に新しいデータオブジェクトを送り続け、ワーカーがリアルタイムに受け取れます。

ワークフォースの選び方は、機密性で決まります。

条件	選ぶワークフォース	理由
機密データ・PII を含む、社外に出せない	**プライベートワークフォース**	組織内で扱える
大量・低コスト・公開しても問題ないデータ	**Amazon Mechanical Turk**	50万人超の規模
専門知識が要る(医療画像など)が、社内に人手がない	**ベンダーワークフォース**	AWS Marketplace の専門企業

> **よく出る取り違え**:「医療画像に匿名化せずラベルを付けたいので Amazon Mechanical Turk を使う」は、コンプライアンス上の誤りです(1-3-5)。**PHI を含むデータのラベリングは、プライベートワークフォースか、要件を満たすベンダーワークフォース**が正解です。

アノテーションの品質を上げる仕組みも押さえます。同じデータを複数のワーカーにラベル付けさせ、その結果を統合する**アノテーション統合 (annotation consolidation)** により、個々のワーカーの誤りを打ち消します。「ラベルの品質がばらつく」の打ち手です。

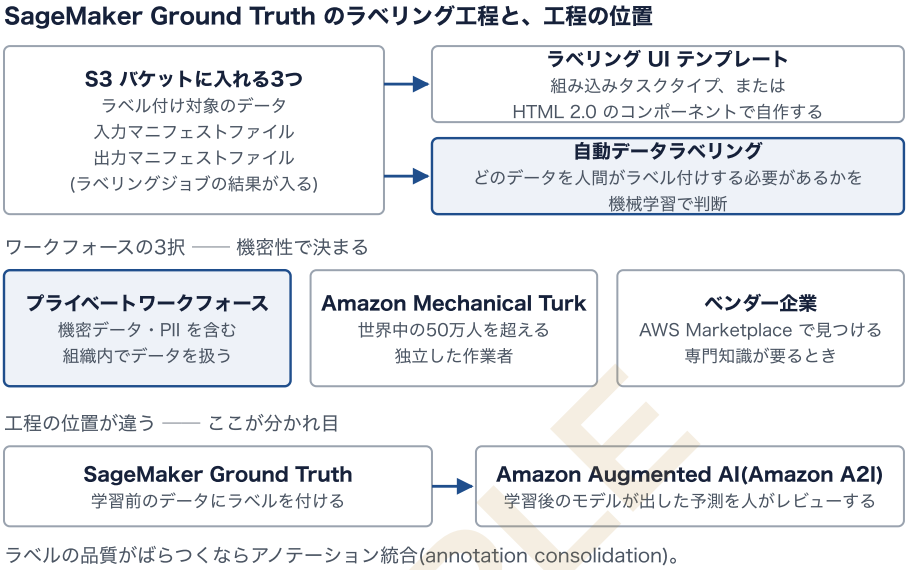


図 1-7　SageMaker Ground Truth のラベリング工程と、工程の位置

1-3 データの完全性を確保しモデリング用にデータを準備する [1.3]

この Task statement は、**バイアス・セキュリティ・コンプライアンス・データ品質**という「壊れていないことを保証する」側の話です。ドメイン1の中で最も落としやすく、かつ用語を知っていれば確実に取れる領域です。

1-3-1 数値・テキスト・画像データの事前学習バイアス指標 [1.3]

事前学習バイアス指標(pre-training bias metrics) は、**モデルを学習させる前に、生データの段階で計算できる**バイアスの指標です。**Amazon SageMaker Clarify** が計算します。

公式ドキュメントの前提を先に押さえます。

- 想定するのは**二値分類**で、結果は**ポジティブ(値1)** と**ネガティブ(値0)** の2つです。
- 比較する集団を**ファセット(facet)** と呼びます。**ファセット a** はバイアスが有利に働く側、**ファセット d** は不利に働く側です。
- **観測ラベル y**(データに記録された結果)と、**予測ラベル y'**(モデルが付けた結果)を区別します。**事前学習バイアス指標はすべてモデルに依存しない(model-agnostic)** ため、どんなモデルにも使えます。

8つの事前学習バイアス指標を表にします。**とくに CI と DPL は名指しで問われます**。

指標	測るもの	値の範囲	値の読み方
クラス不均衡 (Class Imbalance / CI)	**異なるファセット値の間での、メンバー数の不均衡**	正規化後 **[-1, +1]**	正の値は**ファセット a の学習サンプルが多い**。ゼロ付近はサンプル数が均衡。負の値は**ファセット d の方が多い**
ラベルの比率の差 (Difference in Proportions of Labels / DPL)	**異なるファセット値の間での、ポジティブな結果の不均衡**	二値・多カテゴリのラベルは **[-1, +1]**、連続値のラベルは (-∞, +∞)	正の値はファセット a の方がポジティブな結果の割合が高い。ゼロ付近は均等。負の値はファセット d の方が高い
カルバック・ライブラー情報量(KL)	ファセットごとの結果の分布が、エントロピー的にどれだけ乖離しているか	**[0, +∞)**	ゼロ付近は似た分布。正の値が大きいほど乖離が大きい
ジェンセン・シャノン情報量(JS)	同上	**[0, +∞)**	同上

指標	測るもの	値の範囲	値の読み方
Lpノルム(LP)	異なるファセットの結果分布の p ノルム差	**[0, +∞)**	同上
全変動距離(TVD)	結果分布の **L1 ノルム差の半分**	**[0, +∞)**	同上
コルモゴロフ・スミルノフ(KS)	ファセット間の結果分布の**最大の乖離**	**[0, +1]**	ゼロ付近はすべての結果カテゴリで均等。1に近いと、あるカテゴリのラベルが片方のファセットに集中している
条件付き人口統計的格差(CDD / CDDL)	結果の格差を**全体だけでなくサブグループごとにも**測る	**[-1, +1]**	正の値はファセット d が承認より拒否されている状況。ゼロ付近は平均的に格差なし。負の値はファセット a が拒否されている状況

CI と DPL の違いは、言葉で言い切れるようにします。

- **CI はサンプル数の話**……「そもそも中年層以外のデータが足りないのではないか」を測ります。ラベル(結果)は見ません。
- **DPL は結果の話**……「ある集団の方がローンの承認率が高くなるようなラベルの付き方をしていないか」を測ります。

> **よく出る取り違え**: 「CI が大きいのでリサンプリングした。これでバイアスは解消した」は誤りです。**CI を直してもラベルの偏り(DPL)は残ります**。逆に

> 「DPL がゼロなのでバイアスは無い」も誤りで、**サンプル数の偏り(CI)は残っている**ことがあります。**CI と DPL は別の軸で、両方見る**が正解です。

> **よく出る取り違え**:「事前学習バイアス指標でモデルの予測の公平性を測る」は誤りです。事前学習バイアス指標は**予測ラベル y' を使いません**。モデルの予測に対するバイアスは**学習後(post-training)のバイアス指標**の領域です。

数値・テキスト・画像それぞれでの当てはめ方も一行ずつ押さえます。

データ種別	ファセットの取り方の例	CI の見え方
数値(表形式)	年齢層・地域・性別などの列の値	ある層の行数が極端に少ない
テキスト	文書に付いた属性(方言・言語・書き手の属性)	ある属性の文書数が少ない
画像	画像に付いたメタデータ(撮影条件・被写体の属性)	ある条件の画像枚数が少ない

1-3-2 数値・テキスト・画像データセットの CI に対処する戦略 [1.3]

Strategies to address CI の中心は、**合成データ生成(synthetic data generation)** と**リサンプリング(resampling)** です。

戦略	内容	向く条件	落とし穴
オーバーサンプリング(over-sampling)	少数側のレコードを複製して増やす	全体のデータ量が少なく、行を捨てたくない	単純な複製は**過学習**を招く

戦略	内容	向く条件	落とし穴
アンダーサンプリング(under-sampling)	多数側のレコードを間引く	データ量が十分あり、学習時間を短くしたい	**情報を捨てる**ため精度が落ちうる
合成データ生成(synthetic data generation)	少数側に似た新しいデータを人工的に作る。表形式なら**SMOTE**、画像なら生成モデル	少数側のパターンを増やしたいが複製では足りない	現実に存在しないパターンを作ると、かえって歪む
データ拡張(data augmentation)	既存データに変換を加えて水増しする。画像なら回転・反転・切り抜き・明るさ変更、テキストなら同義語置換・逆翻訳	**画像・テキスト**で最も効く	変換がラベルの意味を壊さないことが前提(数字の「6」を180度回すと「9」になる、など)
クラスの重み付け	学習時に少数クラスの誤りへのペナルティを重くする	データそのものは変えたくない	アルゴリズムが重み付けに対応している必要がある

データ種別ごとの当てはめは対応付けで出ます。

データ種別	主な打ち手
数値(表形式)	**SMOTE などの合成データ生成**、オーバー/アンダーサンプリング、クラスの重み付け
テキスト	同義語置換、逆翻訳(別言語に訳して戻す)、少数クラスの文書収集
画像	**データ拡張**(回転・反転・切り抜き・色調変更)、生成モデルによる合成画像

> **よく出る取り違え**:「クラス不均衡があるので精度(accuracy)で評価する」は誤りです。**99パーセントが陰性のデータでは、全部を陰性と答えるだけで精度99パーセントになります**。不均衡データの評価には**適合率(precision)・再現率(recall)・F1スコア・AUC-PR**を使います。データ準備の段階で不均衡に気づいたら、評価指標もセットで見直すのが正しい流れです。

> **よく出る取り違え**:「データ拡張は学習データにも検証データにも同じように適用する」は誤りです。**データ拡張は学習データにだけ適用**します。検証/テストデータは、本番で来るデータの分布をそのまま代表していなければ評価になりません。

1-3-3 データを暗号化する技法 [1.3]

Techniques to encrypt data は、**保存時(at rest)** と**転送時(in transit)** に分けて、鍵を誰が持つかで整理します。

Amazon S3 の保存時暗号化の4択は、そのまま対応付けの設問になります。

方式	鍵を管理するのは	使う場面	注意点
SSE-S3	**AWS(S3 が管理する鍵)**	既定でよい。運用負荷が最小	鍵のアクセス制御を細かく分けられない
SSE-KMS	**AWS KMS のキー**(AWS 管理キーまたはカスタマー管理キー)	**誰が復号できるかを IAM と鍵ポリシーで制御したい**、**CloudTrail で鍵の使用を監査したい**、鍵のローテーションを制御したい	KMS の API 呼び出しにコストとスロットリングがある。**S3 バケットキー**で呼び出しを減らせる

方式	鍵を管理するのは	使う場面	注意点
DSSE-KMS	AWS KMS のキー	**二重の暗号化**が規定で求められる	コストが高い
SSE-C	**利用者(リクエストごとに鍵を渡す)**	鍵を AWS に一切預けたくない	**鍵を失うとデータを復号できない**。鍵管理の責任がすべて利用者側
クライアントサイド暗号化	利用者(アップロード前に暗号化)	AWS に平文を渡したくない	鍵管理と復号処理を自分で持つ

AWS KMS の押さえどころは3つです。

- **AWS 管理キー**と**カスタマー管理キー(CMK)** があり、**キーポリシーで細かい権限制御・自動ローテーション・削除のスケジュールを設定できるのはカスタマー管理キー**です。
- **キーポリシーと IAM ポリシーの両方**で許可が要ります。「クロスアカウントで暗号化データを読めない」の原因はほぼ**鍵ポリシーに** `kms:Decrypt` **が無い**ことです。
- **AWS CloudTrail** に鍵の使用が記録されるため、「誰がいつ復号したか」の監査要件に応えられます。

Amazon SageMaker AI の暗号化は、公式ドキュメントに沿って次の観点で押さえます。

- **保存時の暗号化**……学習ジョブ・処理ジョブ・ノートブックの**ストレージボリュームを KMS キーで暗号化**し、S3 上の入出力データも暗号化します。
- **転送時の暗号化**……**SSL/TLS で通信**します。AWS は **TLS 1.2 を必須**とし、**TLS 1.3 を推奨**しています。

- **コンテナ間通信の暗号化(inter-container traffic encryption)** …… 分散学習で複数のインスタンス間を流れるデータを暗号化します。**学習時間が延びる**トレードオフがあります。
- **ネットワークの分離**……ジョブを **VPC 内で実行**し、インターネットへ出さない構成にします。S3 へは **VPC エンドポイント**経由でアクセスします。
- **認証情報の保護**……**AWS Secrets Manager** にデータベースの認証情報を置き、**AWS IAM Identity Center または IAM** で最小権限を割り当てます。**多要素認証(MFA)** を各アカウントで使います。
- **FIPS 140-3 準拠の暗号モジュール**が必要な場合は、**FIPS エンドポイント**を使います。

> **重要**: **タグや Name のような自由記述のフィールドに、顧客のメールアドレスなどの機密情報を入れてはいけません**と公式ドキュメントが明示しています。これらは請求や診断ログに使われる可能性があるためです。

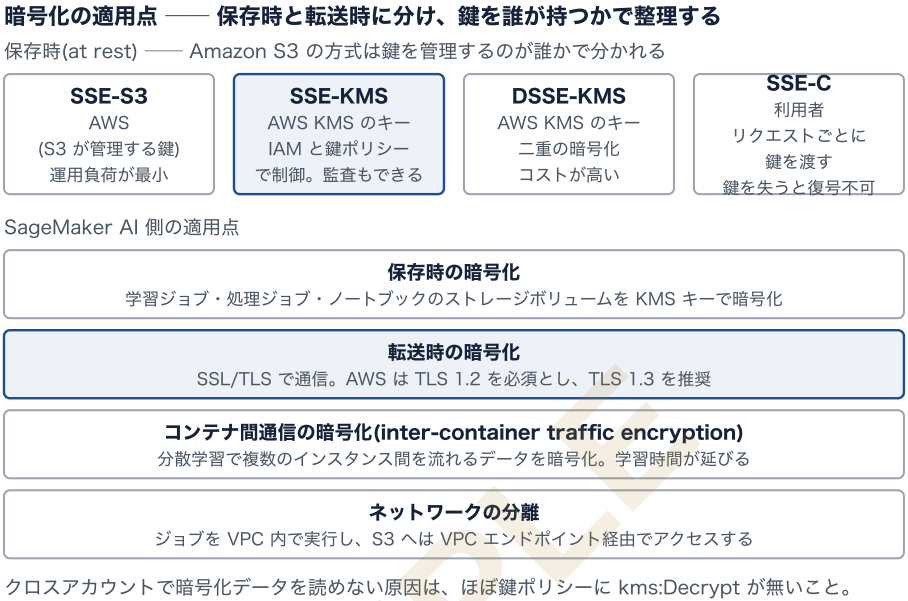


図 1-8　暗号化の適用点 —— 保存時と転送時に分け、鍵を誰が持つかで整理する

1-3-4 データの分類・匿名化・マスキング [1.3]

Data classification, anonymization, and masking の主役は **Amazon Macie** です。

Amazon Macie の仕様は、公式ドキュメントに沿って次のとおりです。

- **機械学習とパターンマッチングで機密データを検出**するデータセキュリティサービスです。対象は **Amazon S3 の汎用バケット**です。
- 有効にすると **S3 汎用バケットのインベントリを自動生成**し、**セキュリティとアクセス制御の観点で評価と監視**を始めます。バケットが公開状態になるなどの問題を検出すると、**ポリシー検出結果(policy finding)** を生成します。

- 機密データの検出は2つの方法があります。
  - **自動機密データ検出(automated sensitive data discovery)** ……S3 バケットのインベントリを継続的に評価し、**サンプリング**で代表的なオブジェクトを選んで分析します。**広く浅く、どこに機密データがありそうかの見取り図**を得る用途です。
  - **機密データ検出ジョブ(sensitive data discovery jobs)** ……分析するバケット・サンプリングの深さ・カスタム条件を自分で決めます。**一度だけ**のオンデマンド分析にも、**定期実行**にもできます。**深く狙って調べる**用途です。

- 検出の基準は2種類です。
  - **マネージドデータ識別子(managed data identifiers)** ……多くの国と地域の**個人を特定できる情報(PII)・金融情報・認証情報**など、増え続ける種類の機密データを検出します。
  - **カスタムデータ識別子(custom data identifiers)** ……自分で定義する**正規表現(regex)** に、必要なら文字列と近接ルールを加えたものです。自社固有の知的財産や独自データを検出できます。

- **許可リスト(allow lists)** で、無視してよいテキストやパターン(自社の公開電話番号、テスト用のサンプルデータなど)を指定できます。
- 検出結果は **Amazon EventBridge にイベントとして発行**され、Lambda 関数や Amazon SNS トピックへルーティングできます。**AWS Security Hub CSPM** への発行も設定できます。
- 複数アカウントは **AWS Organizations との統合**か、招待の送受信で集中管理できます。

匿名化とマスキングの技法を対応表で押さえます。

技法	内容	復元
削除(suppression)	該当の列や値を消す	不可
マスキング(masking)	一部を伏せ字にする(電話番号の下4桁だけ残す等)	不可
仮名化(pseudonymization)/トークン化	実データを別の識別子に置き換え、対応表を別に安全に保管する	**対応表があれば可能**
ハッシュ化	一方向関数で変換する	不可(ただしソルトが無いと総当たりで逆引きされうる)
一般化(generalization)	粒度を粗くする(生年月日→生年、住所→都道府県)	不可
差分プライバシー	統計的なノイズを加え、個人の寄与を隠す	不可

どのサービスで実行するかも対応付けで出ます。

やりたいこと	サービス
S3 の中に PII があるかを**発見**する	**Amazon Macie**
データ準備の中で **PII を検出してマスキング**する	**AWS Glue DataBrew**(PII 検出・マスキングの変換)、**AWS Glue** の PII 検出変換
テキストの中の個人情報エンティティを検出する	**Amazon Comprehend**(PII エンティティ検出)
医療テキストから PHI を検出する	**Amazon Comprehend Medical**
データレイクの**列・行・セル単位のアクセス制御**を掛ける	**AWS Lake Formation**
バケット単位・オブジェクト単位のアクセス制御	**IAM** ポリシー、S3 バケットポリシー

> **よく出る取り違え**:「Amazon Macie でデータをマスキングする」は誤りです。**Macie は発見と可視化のサービスで、データを書き換えません。**マスキングを実行するのは **DataBrew・Glue・自前の処理**です。「発見は Macie、加工は DataBrew/Glue」と対で覚えます。

1-3-5 コンプライアンス要件がもたらす影響 [1.3]

Implications of compliance requirements は、**PII・PHI・データレジデンシー(data residency)** の3語が中心です。

用語	意味	ML の設計への影響
PII(個人を特定できる情報 / personally identifiable information)	氏名・住所・メールアドレス・電話番号・各種 ID など、個人を特定できる情報	収集の最小化、暗号化、匿名化/マスキング、アクセス記録。**特徴量として本当に必要かを問う**
PHI(保護対象保健情報 / protected health information)	医療・健康に関する個人情報	上記に加え、扱えるサービス・扱える人の制限が厳しくなる。**ラベリングはプライベートワークフォース**へ
データレジデンシー(data residency)	データを**特定の国・地域の中に留める**要件	**リージョンの選択**が設計制約になる。クロスリージョンのレプリケーションやバックアップを禁止する場合がある

データレジデンシーが設計に効く具体点を並べます。

- **学習データは学習ジョブと同じリージョンに置く必要があります。**公式ドキュメントは「**データセットは学習ジョブと同じ AWS リージョンに存在しなければならない**」と明示しています。

- S3 のクロスリージョンレプリケーション(CRR)、バックアップの保管先、CloudWatch Logs の集約先まで、**データが越境しないか**を確認します。
- マネージドサービスであっても、**どのリージョンで処理されるか**を確認します。
- **ローカルゾーンのディレクトリバケット**を使う、といった選択肢も、データレジデンシーと分離の要件に対する答えになります。

運用に落とす道具も対応付けで押さえます。

目的	サービス
誰が何の API を呼んだかを記録し、監査に応える	**AWS CloudTrail**
リソースの設定が規定に沿っているかを継続評価する	**AWS Config**
機密データの所在を継続的に把握する	**Amazon Macie**
権限を最小にする	**IAM**(最小権限、ロールベース)、**SageMaker Role Manager**
認証情報を安全に保管する	**AWS Secrets Manager**
ネットワークを閉じる	**Amazon VPC**、VPC エンドポイント、SageMaker のネットワーク分離

> **誤答肢の切り方**:「PII を含むデータで学習したい」というシナリオでは、**「PII 列をそのまま特徴量に入れる」選択肢は常に誤り**です。正しい流れは**「Macie で所在を把握 → 不要な PII 列を落とす → 必要なものは仮名化/マスキング → 暗号化して保存 → 最小権限でアクセス → CloudTrail で監査」**です。この順序は並べ替え問題になります。

1-3-6 データ品質の検証 [1.3]

Validating data quality は、**AWS Glue DataBrew** と **AWS Glue Data Quality** の2つで問われます。

AWS Glue Data Quality の仕様は、公式ドキュメントに沿って次のとおりです。

- **オープンソースの DeeQu フレームワーク**の上に構築された、**マネージドかつサーバーレス**のデータ品質サービスです。インストール・パッチ適用・保守が不要です。
- ルールは **DQDL(Data Quality Definition Language)** という専用言語で書きます。**25種類以上の既成のデータ品質ルール**から始められます。
- **ルールの推奨(recommendation)** 機能があり、データを分析して**自動的にデータ品質ルールを作成**します。「Create Data Quality Rules → Recommend rules」の2クリックで始められます。
- **機械学習による異常検知**で、見つけにくいデータ品質の問題を検出します。**動的ルール**(例: `RowCount > avg(last(10))`)も使えます。
- 評価すると**データ品質スコア**が出ます。スコアは「**ルールセットを評価したときに合格(true)したルールの割合**」です。
- **どのレコードが品質チェックに失敗したかを特定**でき、隔離して直せます。
- 入口は2つあります。
  - **AWS Glue Data Catalog** ……すでにカタログ化済みのデータセットに対して、**コードを書かない担当者(データスチュワード、ビジネスアナリスト)** が品質ルールを設定できます。**保存されたデータ(at rest)** の品質管理です。

    - **AWS Glue ETL ジョブ** ……**プロアクティブ**な品質チェックで、**データレイクへ読み込む前に**不良データを特定して除外できます。**流れているデータ(in transit)** の品質管理です。

- 2つの入口の違いは、そのまま対応付けの設問になります。

機能	Data Catalog 側	ETL ジョブ側
ルールの推奨	**対応**	非対応
DQDL ルールの作成と実行	対応	対応
オートスケーリング	非対応	**対応**
失敗したレコードの特定	非対応	**対応**
Amazon EventBridge との統合	対応	対応
Amazon CloudWatch との統合	対応	対応
データ品質の結果を S3 に書き出す	対応	対応
増分的なデータ品質	プッシュダウン述語で対応	**AWS Glue ブックマーク**で対応
ML による異常検知	対応	対応

- **制限**も数値で問われます。**1つのルールセットに 2,000 ルール**まで、**ルールセットのサイズは 65KB** まで、**統計はアカウントあたり 100,000 件**まで、統計の保持は**最大2年**です。
- **考慮事項**として、**ネストしたデータやリスト型のデータソースは評価できません**(先にフラット化が必要です)。

AWS Glue DataBrew のプロファイリングは、統計を出す側の役割です。値の分布・欠損の数・重複・相関などを可視化し、**どこが壊れているかを目で見つける**用途です。**ルールとして固定し、パイプラインの中で自動的に合否を判定したい**なら AWS Glue Data Quality です。

SageMaker Data Wrangler の Data Insights and Quality Report(1-1-7)も同じ土俵にありますが、**ML の前処理フローの中で品質を確認する**位置づけです。

> **よく出る取り違え**:「データ品質が悪いので Amazon Macie で検証する」は誤りです。**Macie は機密性の話で、品質の話ではありません。**「欠損・重複・値の範囲外を継続的に検出したい」の正解は **AWS Glue Data Quality**、「まず目で見て把握したい」の正解は **DataBrew のプロファイリング**です。

1-3-7 データのバイアス源の特定と緩和 [1.3]

Identifying and mitigating sources of bias では、**選択バイアス(selection bias)** と**測定バイアス(measurement bias)** が名指しで問われます。

バイアスの種類	何が起きているか	具体例	緩和策
選択バイアス(selection bias)	**データの集め方**が母集団を代表していない	Web アンケートで集めたため、ネットを使わない層が入っていない	サンプリング設計を見直す、層化抽出、不足層のデータを追加収集、重み付け

バイアスの種類	何が起きているか	具体例	緩和策
測定バイアス(measurement bias)	**測り方・記録の仕方**が集団によって違う	一部の店舗だけ古い計測器を使っており、値が系統的にずれる	計測方法を統一する、機器ごとの校正、記録元をメタデータとして持ち検証する
ラベリングバイアス	ラベルを付ける人の主観が偏っている	審査担当者の判断が集団によって甘辛違う	**アノテーション統合**、複数ワーカー、ガイドラインの明確化
確認バイアス	期待に合うデータだけを採用する	仮説に合わないデータを外れ値として捨てる	除外の基準を事前に決め、記録に残す
時間的バイアス(データドリフト)	集めた時期の分布が現在と違う	コロナ禍のデータで平常時を予測する	直近データで学習し直す、時系列の分割で検証する

Amazon SageMaker Clarify の使い方は、この文脈で3つに整理します。

1. **事前学習バイアスの検出**……1-3-1 の8指標を生データに対して計算します。**モデルを学習させる前に、資源を投じる前に**気づけるのが利点です。
2. **学習後バイアスの検出**……モデルの予測に対するバイアスを測ります(ドメイン3の領域)。
3. **説明可能性(explainability)** ……**SHAP 値**による特徴量の寄与度を出し、「なぜこの予測になったか」を説明します。

バイアスへの対処の順序は、並べ替え問題になります。

1. **ファセット(比較する属性)と、望ましい結果(ポジティブラベル)を定義する**
2. **SageMaker Clarify で事前学習バイアス指標(CI・DPL 等)を計算する**

3. **どのバイアスがどこから来たか(選択バイアスか測定バイアスか)を特定する**
4. **リサンプリング・合成データ生成・データの追加収集で緩和する**
5. **再計測して、指標が許容範囲に入ったことを確認する**
6. **学習し、学習後バイアス指標と説明可能性で再確認する**

> **よく出る取り違え**:「バイアスがあるので、その属性の列(性別・年齢など)をデータから削除すれば公平になる」は誤りです。**相関する他の列(代理変数 / proxy)からモデルが同じ情報を学び直します**(郵便番号が地域や所得の代理になる、など)。列を消すだけでなく、**指標で測り、データの偏りそのものを直す**必要があります。しかも**属性を消すと、バイアスを測ることすらできなくなります**。

1-3-8 予測バイアスを減らすためのデータ準備 [1.3]

Preparing data to reduce prediction bias の中心は、**データセットの分割(dataset splitting)・シャッフル(shuffling)・拡張(augmentation)** の3つです。

データセットの分割は、目的ごとに3つに分けます。

分割	用途	典型的な割合
学習データ(training)	モデルのパラメータを学習する	60〜80パーセント
検証データ(validation)	ハイパーパラメータの調整、早期停止の判断	10〜20パーセント
テストデータ(test)	**最終的な性能の評価**。一度も学習に使わない	10〜20パーセント

分割の方法も条件で切り替わります。

方法	内容	使う条件
ランダム分割	無作為に分ける	行同士が独立している一般的な表形式データ
層化分割(stratified split)	**各分割でクラスの比率を保つ**ように分ける	**クラス不均衡があるとき**。少数クラスがテストに1件も入らない事故を防ぐ
時系列分割(time-based split)	**時刻で切り、過去で学習して未来で検証する**	**時系列データ**。ランダムに分けると未来の情報が学習に混ざる
グループ分割	同じ実体(同一顧客・同一患者)の行を同じ側にまとめる	1つの実体が複数行を持つとき
k分割交差検証(k-fold cross-validation)	データを k 個に分け、順に検証用にして k 回学習する	データが少なく、1回の分割では評価がぶれるとき

シャッフル(shuffling) は、**データが何らかの順序で並んでいることによる偏りを消す**ために行います。

- 元データが「クラスAが先頭、クラスBが後半」と並んでいると、分割した結果が偏ります。**分割の前にシャッフルする**のが基本です。
- 学習中も、**エポックごとにシャッフル**することで、ミニバッチごとの偏りを減らし、収束を安定させます。
- SageMaker AI の **Pipe mode はマネージドのシャーディングとシャッフル**に対応しています(1-3-9)。
- **例外**: **時系列データは順序に意味があるため、時刻をまたいでシャッフルしてはいけません**。

データ拡張(augmentation) は 1-3-2 のとおりですが、**予測バイアスを減らす**という文脈では「少数側のパターンを増やして、モデルが多数側だけを見て決め打ちするのを防ぐ」役割になります。

リーケージ(leakage)の防止も、この項目で必ず確認します。

リーケージの型	何が起きるか	防ぎ方
ターゲットリーケージ	予測時点では知り得ない情報が特徴量に入っている(例:「解約したか」を予測するのに「解約手続きの日付」を入れる)	各特徴量が**予測時点で入手可能か**を1つずつ確認する。**Data Wrangler のターゲットリーケージ分析**で検出する
前処理のリーケージ	全データでスケーラーや補完の統計量を計算してから分割している	**分割してから、学習データだけで fit し、同じ変換を検証/テストに適用する**
重複によるリーケージ	同じレコードが学習とテストの両方に入っている	分割前に**重複排除**する。グループ分割を使う

> **よく出る取り違え**:「検証データの精度が非常に高いのに、本番で全く当たらない」の原因として、**ハイパーパラメータの調整不足**を選ぶのは誤りです。この症状は**ほぼリーケージ**で、まず「予測時点で知り得ない列が入っていないか」「前処理を分割前にやっていないか」「重複行が両側に入っていないか」を疑います。

1-3-9 モデル学習リソースへデータを投入する設定 [1.3]

Configuring data to load into the model training resource は、**SageMaker AI の入力モードとデータソースの選択**そのものです。公式ドキュメントの記述が、そのまま設問の根拠になります。

前提: 学習ジョブを作るとき、**学習データセットの場所**と**データ入力モード**を指定します。SageMaker AI が対応するのは **Amazon S3・Amazon EFS・Amazon FSx for Lustre** です。**データセットは学習ジョブと同じ AWS リージョンに存在しなければなりません。**

3つの入力モードを表にします。

入力モード	動作	学習開始のタイミング	ディスク容量の要件	対応する入力指定
File mode(既定)	S3 からコンテナ内のローカルディレクトリへ**データセット全体をダウンロード**し、ファイルシステムとして見せる	**全データのダウンロード完了後**	**データセット全体が収まる容量が必要**	**S3 プレフィックス・マニフェストファイル・拡張マニフェストファイル**
Fast file mode	S3 のオブジェクトを **POSIX 準拠のファイルシステムのインタフェースで公開**し、学習スクリプトが消費するタイミングで**オンデマンドにストリーミング**する	**ダウンロードを待たずに開始できる**	**データセット全体が収まる必要がない**	**S3 プレフィックスのみ**(マニフェスト・拡張マニフェストは非対応)
Pipe mode	S3 から**名前付きパイプ (FIFO)** へ直接ストリーミングする	ダウンロード不要で速い	**最終的なモデル成果物が入る分だけでよい**	S3。**マネージドのシャーディングとシャッフル**に対応

モードごとの詳細も、そのまま誤答肢の切り方になります。

- **File mode** は、**明示的に他を指定しない場合の既定の入力モード**です。ダウンロード速度は**データセットのサイズ・ファイルの平均サイズ・ファイル数**に依存します。分散学習では `ShardedByS3Key` オプションでデータセットを複数インスタンスに分割できます。**SageMaker AI のローカルモード**と互換です。
- **Fast file mode** は、**Pipe mode の性能上の利点を活かしつつ、S3 のデータソースにファイルシステムとしてアクセス**します。学習開始時にデータファイルを**特定するだけでダウンロードはしません。**したがって**S3 プレフィックス内のファイル数が少ないほど、学習の起動が速くなります。ランダムアクセスに対応**しますが、**逐次読み出しのときに最も性能が出ます。**ローカルモードと互換です。
- **Fast file mode の注意点**として、公式ドキュメントは **CloudTrail のコストが増える可能性**を挙げています。S3 のデータイベント、KMS で暗号化された S3 オブジェクトへのアクセス時の KMS 復号イベント、KMS 操作に関する管理イベントが追加でログに出るためです。
- **Pipe mode** は「**より新しく、より簡単に使える Fast file mode に大きく置き換えられた**」と公式ドキュメントに書かれています。**各パイプは1つのプロセスからのみ読み取れます。**TensorFlow 向けには SageMaker AI 固有の拡張があり、テキスト・TFRecords・**RecordIO** 形式のストリーミングをネイティブのデータローダーに統合できます。

ファイルシステムをデータソースにする場合は、次の制約が付きます。

データソース	前提条件	押さえどころ
Amazon FSx for Lustre	**学習ジョブが VPC に接続**する必要がある(DevOps 側の設定が要る)	**数百 GB/秒のスループットと数百万 IOPS**まで拡張でき、低レイテンシでファイルを取得できる。**マウント自体は速く、FSx に保存されたデータセットのサイズに依存しない**。データ転送料金を避けるため**ファイルシステムは単一の AZ**を使い、**その AZ にマップされる VPC サブネットを指定**する
Amazon EFS	**学習の前にデータが既に EFS 上に存在**している必要がある。**学習ジョブが VPC に接続**する必要がある	SageMaker AI が指定の EFS ファイルシステムをマウントしてから学習スクリプトを開始する
S3 Express One Zone	ディレクトリバケットの場所を入力に指定し、必要な権限を持つ IAM ロールの ARN を渡す	**File mode・Fast file mode・Pipe mode のすべてで利用可能**。ただし**出力データの暗号化は SSE-S3 のみ**(SSE-KMS は非対応)

入力モードの選び方を判断表にします。**この対応は並べ替え・対応付けの両方で出ます。**

条件	選ぶもの	理由
データセットが小さく、インスタンスのボリュームに収まる	**File mode**	既定で、最も単純

条件	選ぶもの	理由
データセットが大きく、**インスタンスのストレージに収まらない**	**Fast file mode** または **Pipe mode**	ダウンロードせずストリーミングする
学習の起動時間を短くしたい	**Fast file mode**	ダウンロード完了を待たずに開始できる
拡張マニフェストファイルでラベルとデータを対応付けたい	**File mode**	Fast file mode は拡張マニフェストに非対応
マネージドのシャッフルとシャーディングが要る	**Pipe mode**	公式に対応と明記されている
同じデータで何度も学習を回す(ハイパーパラメータ探索など)	**Amazon FSx for Lustre**	マウントが速く、繰り返しの読み出しが速い
すでにデータが共有ファイルシステム上にある	**Amazon EFS**	データを S3 へ移す工程を省ける
分散学習で**インスタンスごとにデータを分けたい**	**File mode +** `ShardedByS3Key`	データセットを複数インスタンスへ分割できる

入力チャネルの規約も1つ覚えます。**すべての SageMaker AI のアルゴリズムで、`InputDataConfig` の `ChannelName` は `train` に設定する必要があります。**アルゴリズムによっては `validation` や `test` の入力チャネルにも対応しており、これらは学習に使わないホールドアウトデータでの性能評価に使います。

学習済みモデルの保存先も対で押さえます。SageMaker AI のモデルは、`create_training_job` の `OutputDataConfig` の `S3OutputPath` で指定した S3 バ

ケットに `model.tar.gz` として保存されます。**この S3 バケットはノートブックインスタンスと同じ AWS リージョンにある必要があります。**

> **よく出る取り違え**:「学習データが大きいので Pipe mode を使う」は、いまは最善ではないことが多いです。公式ドキュメントが「**Pipe mode は、より新しく簡単な Fast file mode に大きく置き換えられた**」と書いているとおり、**S3 プレフィックスで足りるなら Fast file mode** が既定の答えになります。**Pipe mode を選ぶ理由が残るのは、マネージドのシャーディングとシャッフルが要る場合**です。

1-4 章末: ドメイン1の判断表 [1.1/1.2/1.3]

最後に、ドメイン1の判断を1枚にまとめます。**評価軸(問題文の最後の1〜2文)から逆に引く**のがこの表の使い方です。

問題文の評価軸	ほぼ確定する答え
「コードを書かずに」データを整形	**AWS Glue DataBrew**
「運用負荷を最小に」定常 ETL	**AWS Glue**(サーバーレス)
「既存の Spark/Hadoop 資産を活かす」	**Amazon EMR**
「ML の前処理から Feature Store まで一気に」	**SageMaker Data Wrangler**
「リアルタイム推論で低ミリ秒の特徴量取得」	**Feature Store のオンラインストア**
「過去の時点の特徴量で学習し直す」	**Feature Store のオフラインストア**(S3・Parquet・追記のみ)

問題文の評価軸	ほぼ確定する答え
「ストリームで直近N分の集計」	**Amazon Managed Service for Apache Flink**
「ストリームを1件ずつ軽く整形」	**AWS Lambda**
「コードなしでストリームを S3 へ貯める」	**Amazon Data Firehose**
「既存の Kafka をそのまま」	**Amazon MSK**
「200列から3列だけ集計」	**Parquet / ORC**
「スキーマが頻繁に変わる行指向」	**Apache Avro**
「アクセスパターンが不明」	**S3 Intelligent-Tiering**
「ミリ秒で読めるアーカイブ」	**S3 Glacier Instant Retrieval**
「海外からのアップロードが遅い」	**S3 Transfer Acceleration**
「小さな I/O を高 IOPS・一貫した低レイテンシで」	**Amazon EBS Provisioned IOPS(io2 Block Express / io1)**
「大きな連続読み出しを安く」	**Amazon EBS st1**
「Windows と Linux の双方から同じデータ」	**Amazon FSx for NetApp ONTAP**
「学習の読み出しスループットが律速」	**Amazon FSx for Lustre**
「学習データがインスタンスに収まらない」	**Fast file mode**(次点で Pipe mode)
「学習の起動を速く」	**Fast file mode**
「S3 に PII があるかを発見」	**Amazon Macie**
「PII をマスキング」	**AWS Glue DataBrew / AWS Glue**
「テキスト中の PII エンティティ検出」	**Amazon Comprehend**(医療は **Amazon Comprehend Medical**)

問題文の評価軸	ほぼ確定する答え
「データ品質ルールを継続的に判定」	**AWS Glue Data Quality**(DQDL)
「学習前にデータの偏りを測る」	**SageMaker Clarify の事前学習バイアス指標(CI・DPL 等)**
「機密データのラベリング」	**SageMaker Ground Truth のプライベートワークフォース**
「大量・低コストのラベリング」	**Amazon Mechanical Turk**
「低信頼度の推論を人がレビュー」	**Amazon Augmented AI(A2I)**
「誰が復号したかを監査したい」	**SSE-KMS**(+ AWS CloudTrail)
「データを国外に出せない」	**リージョンの選択**、CRR の禁止、VPC エンドポイント

この章で扱った道具は、次章以降で「どのアルゴリズムに、どの形で渡すか」という話に接続します。**データ準備の判断を1つ間違えると、後段のモデル開発とデプロイの選択肢がまるごと変わる**という構造を意識して読み進めてください。

サンプルはここまでです

続き(第2章〜巻末)は、AWS MLA-C01 問題集のご購入で、頻出度別ノート・暗記用ノートと合わせて3点すべてダウンロードできます。

AWS MLA-C01 学習用テキスト(無料サンプル)

版

2026年7月版(AWS 公式 Exam Guide（MLA-C01・2026年9月6日取得）準拠)

発行

IT資格道場(https://www.shikaku-doujou.com)

お問い合わせ

info@shikaku-doujou.com