令和７年度「地方やデジタル分野における専修学校理系転換等推進事業」

ビッグデータ技術応用教材資料

本ビッグデータ技術応用教材資料は、文部科学省の教育政策推進事業委託費による委託事業として、一般社団法人全国専門学校情報教育協会が実施した令和７年度「地方やデジタル分野における専修学校理系転換等推進事業」の成果物です。

情報成長分野の教育プログラム整備と教員育成による学科転換・新設推進事業

目　次

ビッグデータ技術応用

はじめに

ビッグデータ技術応用 実習ガイド

ビッグデータ技術応用

この教材は、『ビッグデータ技術基礎』で学んだ理論を、実際に手を動かして確認するための**実践教材**です。基礎教材では全7章で構成された内容で、ビッグデータ分析の理論や仕組みを体系的に学習しましたが、**『ビッグデータ技術応用』**では、それらの理論を３つの実習で、**「理解」を「実践」に変える**ことを目指します。

『ビッグデータ技術基礎』（理論・知識の学習）
- 第1章 ビッグデータとその活用
- 第2章 データ分析基礎
- 第3章 スクリプト言語による分析
- 第4章 ツールとモニタリング
- 第5章 列指向ストレージによる高速化
- 第6章 データマートの基本構造
- 第7章 大規模分散処理

→

『ビッグデータ技術応用』
（実践・体験による理解の深化）
- 実習１ 分析実践
- 実習２ 業務活用
- 実習３ 大規模処理

学習のゴール

ビッグデータ技術応用

- ✓ 『ビッグデータ技術基礎』で学んだ知識を**「使えるスキル」**に変える
- ✓ ビジネス課題をデータで解決する**「実践的な流れ」**を体験する
- ✓ 大規模データを扱うための**「基盤技術」**の基礎を習得する

実習構成

ビッグデータ技術応用

実習 1 分析実践
Pythonによる
顧客データ分析と
レポーティング

→

実習 2 業務活用
アクセスログETL処理
とBIツールでの可視化

→

実習 3 大規模処理
Sparkによる
大規模データ集計と
高速化

ビッグデータ技術基礎		ビッグデータ技術応用		
第1章 ビッグデータとその活用	→	実習 1		
第2章 データ分析基礎	→	実習 1		
第3章 スクリプト言語による分析	→	実習 1	実習 2	
第4章 ツールとモニタリング	→		実習 2	
第5章 列指向ストレージによる高速化	→			実習 3
第6章 データマートの基本構造	→		実習 2	
第7章 大規模分散処理	→			実習 3

実習で使用する主なツール・技術

ビッグデータ技術応用

Python (pandas, matplotlib)：データ分析のコア言語・ライブラリ

Google Colaboratory：環境構築不要のPython実行環境

Tableau / Power BI：データを可視化するBIツール

Apache Spark　：大規模データを高速処理する分散処理フレームワーク

Docker　：Spark環境をPC上で動かすための基盤技術

ポイント
すべて無料で利用できるツール・環境を選定しています

実習の進め方

ビッグデータ技術応用

導入ガイド　：実習の概要、環境準備、データの入手方法等

オンライン講義（動画）　：各Partの解説とデモ

ワークブック　：実際にコードや考察を記入する教材

発展課題　：学んだ知識を試す追加課題

実習ステップ

ビッグデータ技術応用

準備	**導入ガイド**を読み、環境と機材を準備する
視聴	**オンライン講義**（動画）を視聴する
実践	**ワークブック**のタスクに沿って、自分で配線やコード実行する
記録	**ワークブック**に実行結果や気づきを記録する
考察	**考察問題**に挑戦し、自分の言葉でまとめる
挑戦	**発展課題**でさらにスキルを磨く

ビッグデータ技術応用

「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」

実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理

NEXT

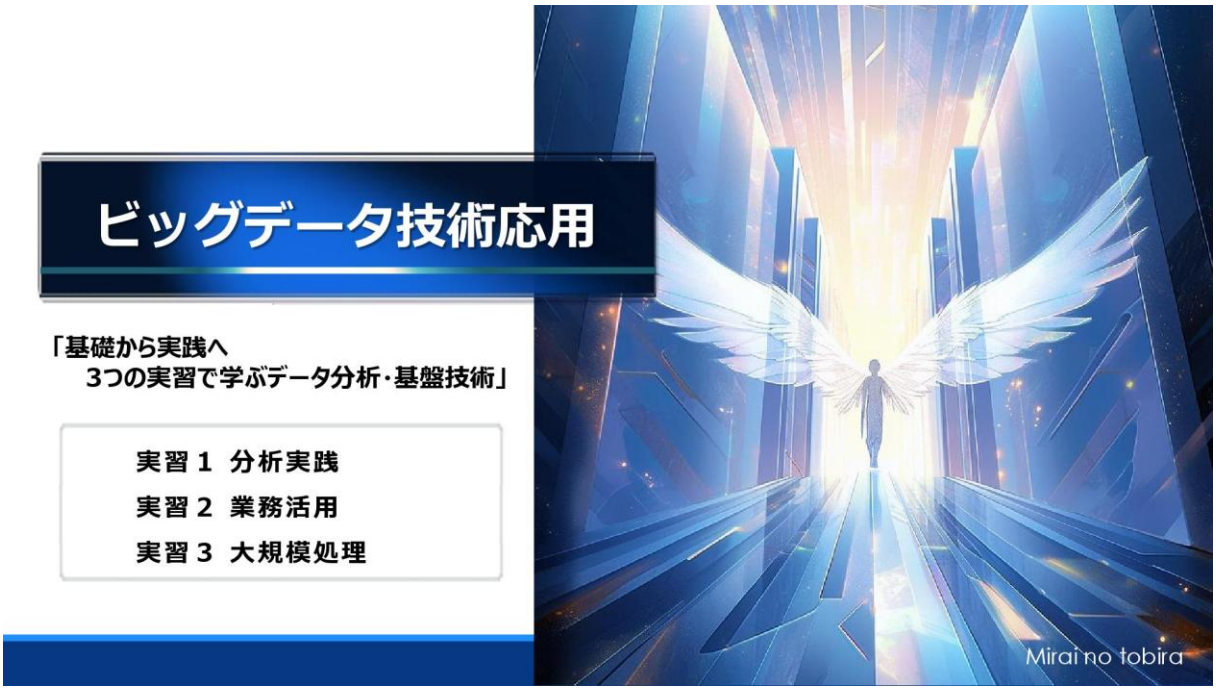
ビッグデータ技術応用

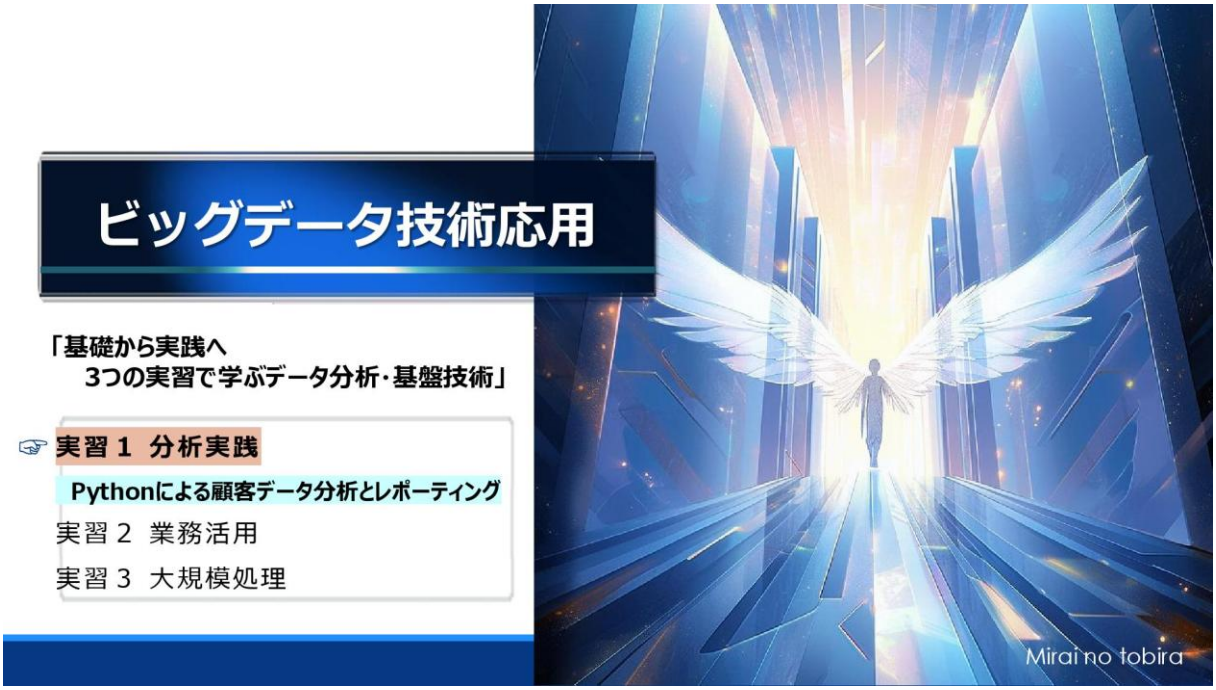
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」

- 実習１ 分析実践
- 実習２ 業務活用
- 実習３ 大規模処理

ビッグデータ技術応用__実習１

導入編


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
Pythonによる顧客データ分析とレポーティング
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira

実習構成

ビッグデータ技術応用

実習 1 分析実践
Pythonによる
顧客データ分析と
レポーティング

実習 2 業務活用
アクセスログETL処理と
BIツールによる可視化

実習 3 大規模処理
Spark による
大規模データ処理

ビッグデータ技術基礎		ビッグデータ技術応用		
第1章 ビッグデータとその活用	→	**実習 1**		
第2章 データ分析基礎	→	**実習 1**		
第3章 スクリプト言語による分析	→	**実習 1**	**実習 2**	
第4章 ツールとモニタリング	→		**実習 2**	
第5章 列指向ストレージによる高速化	→			**実習 3**
第6章 データマートの基本構造	→		**実習 2**	
第7章 大規模分散処理	→			**実習 3**

学習のゴール

ビッグデータ技術応用

「データ分析の一連のプロセスを、すべて自分の手で体験する」

- ✓ **データを読み込み** （Part 1）
- ✓ **データをきれいにし** （Part 2）
- ✓ **集計・可視化して特徴を探す** （Part 3）
- ✓ **レポートにまとめる** （Part 4）

… この**全工程を体験**します

利用するデータ

ビッグデータ技術応用

KAGGLE:データ分析コンペのサイトで公開されている Eコマースデータ

- オンラインショップの 約54万件に及ぶ実際の購買履歴データ
- 「誰が」「いつ」「何を」「いくつ」「いくらで」買ったのかが記録されている

☞ このデータから、「優良顧客は誰か？」というテーマで分析を進めます

利用するツール

ビッグデータ技術応用

分析には、**Python** と **pandasライブラリ** を使う

実行環境は、**Google Colaboratory** を推奨

- 環境構築なしに Python を実行できる無料のサービス
- 自分のPCに Python をインストール不要

実習１ 分析実践

ビッグデータ技術応用

実習１ 分析実践
Pythonによる顧客データ分析とレポーティング

実習２ 業務活用

実習３ 大規模処理

Part 1：環境構築とデータ理解
Part 2：データクレンジングと前処理
Part 3：探索的データ分析と可視化
Part 4：統合分析（RFM）とレポート作成

実習の進め方

ビッグデータ技術応用

導入ガイド	： 実習の概要、環境準備、データの入手方法等
オンライン講義（動画）	： 各Partの解説とデモ
ワークブック	： 実際にコードや考察を記入する教材
発展課題	： 学んだ知識を試す追加課題


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


NEXT
ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira

ビッグデータ技術応用__実習１

Part1


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
Pythonによる顧客データ分析とレポーティング
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


ビッグデータ技術応用
実習１ 分析実践
Pythonによる顧客データ分析とレポーティング
Part １：環境構築とデータ理解
Part ２：データクレンジングと前処理
Part ３：探索的データ分析と可視化
Part ４：統合分析（RFM）とレポート作成
Mirai no tobira

実習１ 分析実践

ビッグデータ技術応用

実習１ 分析実践
Pythonによる
顧客データ分析と
レポーティング

➡ 実習２ 業務活用 ➡ 実習３ 大規模処理

☞**Part 1：環境構築とデータ理解**

Part 2：データクレンジングと前処理

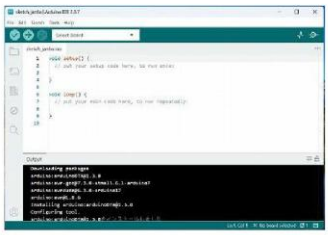
Part 3：探索的データ分析と可視化

Part 4：統合分析（RFM）とレポート作成

Part 1 ：環境構築とデータ理解

実習１ 分析実践

データ分析の第一歩：まずは敵を知る



Part１：環境構築とデータ理解 の学習目標

実習１ 分析実践

- ✓ Google Colab で Python環境を準備できる
- ✓ pandas で CSVデータを読み込める
- ✓ データの全体像（行数、列数、型）を把握できる
- ✓ 基本統計量（平均、最小値など）の意味を理解できる
- ✓ データに潜む「問題点」（欠損値、異常値）を発見できる

実習１ Part１の進め方

実習１ 分析実践

４つのタスク

- ➢ **タスク1：環境構築**　（分析の「作業場」準備）
- ➢ **タスク2：データ読み込み**　（分析対象をセット）
- ➢ **タスク3：データ構造の把握**　（「戸籍謄本」を見る）
- ➢ **タスク4：基本統計量の算出**　（「健康診断」をする）

環境の選択

実習 1 分析実践 Part 1

タスク1：Python環境の選択

	Google Colaboratory (推奨)	ローカル環境（Jupyter）
インストール	**不要（ブラウザのみ）**	必要 (Anaconda等)
必要なもの	Googleアカウント	Python, 各種ライブラリ
メリット	**環境構築でつまずかない**	オフライン作業可能

結論

この実習では Google Colaboratory を使います
環境構築の時間を節約し、分析に集中しましょう

Google Colaboratory を始めよう

実習 1 分析実践 Part 1

Google Colaboratory の始め方／4つのステップ

1. **https://colab.research.google.com** にアクセス
2. **Googleアカウントでログイン**
3. **「ファイル」→「ノートブックを新規作成」**
4. **ノートブック名を変更**（例：bigdata_ex1_part1.ipynb）

分析の『道具』を準備しよう

実習 1 分析実践 Part 1

ライブラリのインポート

```
# データ操作・分析
import pandas as pd
# 数値計算
import numpy as np
# グラフ描画
import matplotlib.pyplot as plt
import seaborn as sns
```


データの読み込み

実習 1 分析実践 Part 1

タスク2：Kaggle の CSVデータを読み込む

ステップ1：ファイルアップロード

```
from google.colab import files
uploaded = files.upload()
```

ステップ2：CSV を pandas で読み込む

```
# 読み込み（文字化け対策の encoding を指定）
df = pd.read_csv('data.csv', encoding='ISO-8859-1')
```

ポイント

encoding='ISO-8859-1' は、海外のデータ（特にヨーロッパ系）を扱う際の文字化けを防ぐ「おまじない」です

データの「チラ見」

実習 1 分析実践 Part 1

まずは『チラ見』する： .head()

✓ 54万行のデータのまずは先頭5行を見てどんな列があるか確認する

```
df.head()
```


データの「健康診断」

実習 1 分析実践 Part 1

タスク3：データの『健康診断』

① 形状確認 (.shape)：何行、何列？

② 基本情報 (.info())：データ型は？ 欠損値は？

数値データの統計量

実習 1 分析実践 Part 1

タスク4：数値データの『健康診断』

.describe() で、数値列の統計量（平均(mean)、最小(min)、最大(max)）を一覧表示します

実習 1 Part 1 のまとめ

実習 1 分析実践 Part 1

- ✓ **分析用の「生データ」は、そのままでは使えない**
- ✓ **問題①：欠損値**
  CustomerID が大量に抜けている
- ✓ **問題②：異常値（返品）**
  Quantity にマイナスの値がある
- ✓ **問題③：異常値（無料）**
  UnitPrice に 0.0 の値がある

Part 1：環境構築とデータ理解　ビッグデータ技術応用 実習 1

NEXT

実習１ Part２：データクレンジングと前処理

実習１ 分析実践

実習１ Part２では、
これらの「問題点」をすべて解決する
「データクレンジング（お掃除）」を行います

NEXT

ビッグデータ技術応用

実習１ 分析実践
Pythonによる顧客データ分析とレポーティング

Part １：環境構築とデータ理解
☞ **Part ２：データクレンジングと前処理**
Part ３：探索的データ分析と可視化
Part ４：統合分析（RFM）とレポート作成

ビッグデータ技術応用__実習１

Part2


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
Pythonによる顧客データ分析とレポーティング
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


ビッグデータ技術応用
実習１ 分析実践
Pythonによる顧客データ分析とレポーティング
Part １：環境構築とデータ理解
Part ２：データクレンジングと前処理
Part ３：探索的データ分析と可視化
Part ４：統合分析（RFM）とレポート作成
Mirai no tobira

実習 1 分析実践

ビッグデータ技術応用

実習 1 分析実践
Pythonによる
顧客データ分析と
レポーティング

実習 2 業務活用

実習 3 大規模処理

Part 1：環境構築とデータ理解

☞**Part 2：データクレンジングと前処理**

Part 3：探索的データ分析と可視化

Part 4：統合分析（RFM）とレポート作成

Part 2：データクレンジングと前処理

実習 1 分析実践

分析のための『お掃除』と『下ごしらえ』

Part 2：データクレンジングと前処理 の達成目標

実習 1 分析実践

- ✓ Part 1で見つけたデータの問題点を特定できる
- ✓ 欠損値（空白）を適切に処理（削除）できる
- ✓ 異常値（マイナス値など）をルールに基づいて処理できる
- ✓ データ型を分析しやすいように変換できる
- ✓ 分析に役立つ新しい列（特徴量）を作成できる

クレンジングの重要性

実習 1 分析実践

なぜデータクレンジングが必要？

➢ **GIGO** (Garbage In, Garbage Out)

「ゴミを入れたら、ゴミしか出てこない」

汚れたデータ (Garbage In)

- ▶ 欠損値 (CustomerID = ?)
- ▶ 異常値 (Quantity = -100)
- ▶ 型がバラバラ (ID = 123.0)

間違った分析結果 (Garbage Out)

- ▶ 顧客数が正しく数えられない
- ▶ 売上合計がマイナスになる
- ▶ エラーで分析が止まる

Part 1 のおさらい

実習 1 分析実践 Part 2

Part 1で発見した問題点

- **問題①：欠損値**
  - `CustomerID` が約25%も空白 (NaN)
- **問題②：異常値** (Quantity)
  - `Quantity` の最小値がマイナス（返品データ）
- **問題③：異常値** (UnitPrice)
  - `UnitPrice` の最小値が 0.0（無料？）
- **問題④：データ型**
  - `CustomerID` が整数なのに小数 (float) になっている

今回のミッション
これら4つの問題をすべて解決し、分析できある「きれいなデータ」を作ります

データの準備

実習 1 分析実践 Part 2

タスク1：データの準備

Part 1 と同様に、ライブラリをインポートし、データを読み込みます

今回の重要ポイント：

元のデータを df_raw にコピーして保護する

```
# 元のデータをコピーして作業用DataFrameを作成
df_raw = df.copy()
```

問題①：欠損値の処理

実習 1 分析実践 Part 2

タスク2：欠損値の処理 (CustomerID)

分析目的：「優良顧客を見つける」

→ **顧客ID** (CustomerID) **が不明なデータは？**

→ 分析に使えない！ → **削除する**

```
# CustomerID が欠損している行を削除
df_cleaned = df.dropna(subset=['CustomerID'])
```


処理結果の確認

実習 1 分析実践 Part 2

必ず確認！処理前後の比較

➢ **処理を実行したら、必ず結果を確認する癖をつけましょう**

処理前 (df)
- ▶ 行数: 541,909 行
- ▶ CustomerID欠損: 135,080 件

処理後 (df_cleaned)
- ▶ 行数: 406,829 行
- ▶ CustomerID欠損: 0 件

ポイント
約13.5万行のデータが、今回の分析対象から除外されたことを確認します

問題②③：異常値の処理

実習 1 分析実践 Part 2

タスク3：異常値の処理 (Quantity, UnitPrice)

分析目的：「購入データ」の分析

→ **数量**(Quantity)**が 0 以下**（マイナス）のデータは？
→ **「返品」**データであり、**「購入」**ではない → **削除する**
→ **単価**(UnitPrice)**が 0 以下**のデータは？
→ **「売上」**に貢献しない（無料サンプル等）→ **削除する**

異常値処理のコード

実習 1 分析実践 Part 2

Pandasの『条件抽出』

条件に合う行だけを残す、という書き方を使います

```
# Quantity が 0 より大きいデータのみを抽出
df_cleaned = df_cleaned[df_cleaned['Quantity'] > 0]

# UnitPrice が 0 より大きいデータのみを抽出
df_cleaned = df_cleaned[df_cleaned['UnitPrice'] > 0]
```

Part 2：データクレンジングと前処理 ビッグデータ技術応用 実習 1

問題④：データ型の変換

実習 1 分析実践 Part 2

タスク4：データ型の変換

欠損値がなくなったので、
CustomerID を「小数(float)」から「整数(int)」に変換できます

```
# .astype() を使って型を変換
df_cleaned['CustomerID'] =
df_cleaned['CustomerID'].astype('int64')
```

変換前		変換後
12345.0	➡	12345

特徴量の作成

実習 1 分析実践 Part 2

特徴量作成①：TotalPrice

分析に必要な**「合計金額」列**を、**既存の列から作成**します
これを **特徴量エンジニアリング** と呼びます

TotalPrice = Quantity × UnitPrice

```
df_cleaned['TotalPrice'] = df_cleaned['Quantity'] * df_cleaned['UnitPrice']
```

日付データの処理

実習１ 分析実践 Part２

特徴量作成②：日付データ

InvoiceDate 列（文字列）を**「日付型」に変換**し
便利な情報を抽出します

```
# 1. 文字列を日付型に変換
df_cleaned['InvoiceDate'] =
pd.to_datetime(df_cleaned['InvoiceDate'])

# 2. 日付型から「月」「曜日」「時間」などを抽出
df_cleaned['Month'] =
df_cleaned['InvoiceDate'].dt.month
df_cleaned['DayOfWeek'] =
df_cleaned['InvoiceDate'].dt.day_name()
df_cleaned['Hour'] = df_cleaned['InvoiceDate'].dt.hour
```


実習１ Part２のまとめ

実習１ 分析実践 Part２

「汚れた」生データから「きれいな」分析用データを作成しました

- ✅ **欠損値処理：.dropna() で不要な行を削除した**
- ✅ **異常値処理：条件指定 df[**（条件）**] で不要な行を削除した**
- ✅ **型変換　　：.astype() でデータ型を正しく直した**
- ✅ **特徴量作成：四則演算や .dt で新しい列を追加した**

🎯 **最も重要なポイント**
データクレンジングは「なんとなく」ではなく**「分析の目的に合わせて」**ルールを決めることが重要

NEXT

実習１ Part３：探索的データ分析と可視化

実習１ 分析実践

実習１ Part３では、
きれいになったデータを使って
分析とグラフ作成を行います

NEXT

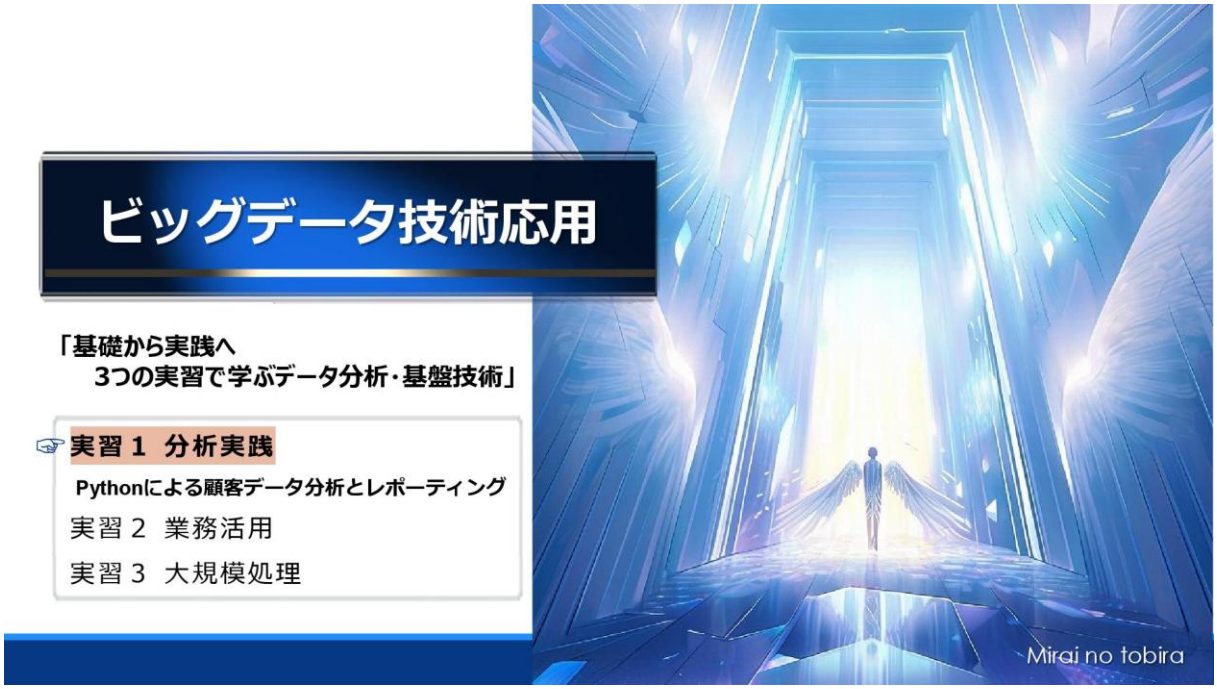
ビッグデータ技術応用

実習１ 分析実践
Pythonによる顧客データ分析とレポーティング

Part １：環境構築とデータ理解
Part ２：データクレンジングと前処理
☞ **Part ３：探索的データ分析と可視化**
Part ４：統合分析（RFM）とレポート作成

ビッグデータ技術応用__実習１

Part3


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
Pythonによる顧客データ分析とレポーティング
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


ビッグデータ技術応用
実習１ 分析実践
Pythonによる顧客データ分析とレポーティング
Part １：環境構築とデータ理解
Part ２：データクレンジングと前処理
Part ３：探索的データ分析と可視化
Part ４：統合分析（RFM）とレポート作成
Mirai no tobira

実習１ 分析実践

ビッグデータ技術応用

実習１ 分析実践
Pythonによる
顧客データ分析と
レポーティング

実習２ 業務活用

実習３ 大規模処理

Part １：環境構築とデータ理解
Part ２：データクレンジングと前処理
☞ **Part ３：探索的データ分析と可視化**
Part ４：統合分析（RFM）とレポート作成

Part３：探索的データ分析と可視化

実習１ 分析実践

データと対話し、インサイト（発見）を引き出す

Part３：探索的データ分析と可視化 の達成目標

実習１ 分析実践

- ✓ データ分析の「問い」を立てる
- ✓ .groupby() でデータを自在に集計する
- ✓ matplotlib で折れ線グラフを作成し、トレンドを読む
- ✓ seaborn で棒グラフを作成し、ランキングを比較する
- ✓ グラフからビジネス的な「考察」を導き出す

分析の「問い」

実習１ 分析実践

データに尋ねる『問い』

Part２でデータはきれいになりました。しかし、ただ眺めていても答えは出ません
分析とは、データに**「具体的な問い」**を投げかけることです

Part３で答える「問い」：

- ▶ **問い①：全体の売上はどのように推移しているか？（トレンド分析）**
- ▶ **問い②：どの商品が一番売れているか？（売れ筋分析）**
- ▶ **問い③：どの国からの注文が多いのか？（顧客分布）**

最強の武器: .groupby()

実習 1 分析実践 Part 3

最強の集計武器： .groupby()

これらの「問い」に答えるため、**pandas の .groupby() を使います**

「OO ごとに、△△ を □□ する」

- ▶ 例：「**月ごとに、売上** を **合計** する」
- ▶ 例：「**商品 ごとに、数量** を **合計** する」
- ▶ 例：「**国 ごとに、売上** を **合計** する」

問い①：売上トレンド

実習 1 分析実践 Part 3

タスク2：売上トレンドの分析

ステップ1：集計

```
# 「年月(YearMonth) ごとに、売上
(TotalPrice) を 合計(sum) する」
monthly_sales =
df_cleaned.groupby('YearMonth')
['TotalPrice'].sum()
```

ステップ2：可視化（折れ線グラフ）

```
# グラフのサイズ指定
plt.figure(figsize=(12, 6))
# 折れ線グラフを描画
plt.plot(monthly_sales.index.to_timestamp(),
monthly_sales.values, marker='o')
# タイトルやラベル
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.show()
```

考察①：グラフから読む

実習 1 分析実践 Part 3

グラフから何を読み取るか？

考察のポイント

- 傾向　：全体として売上は伸びているか？（→ 右肩上がりに見える）
- 特異点：特に目立つ月は？（→ 11月が突出して高い）
- 仮説　：なぜ11月が高い？（→ クリスマスの年末商戦が理由かも？）

分析とは、グラフを作ることではなく、グラフから「仮説」を立てることです

Seabornで可視化

実習 1 分析実践 Part 3

Seabornで棒グラフをきれいに描画

ランキングの可視化には**「棒グラフ」**が最適です

seaborn を使うと簡単に描画できます

```
plt.figure(figsize=(12, 7))
# 棒グラフ（barplot）を作成
sns.barplot(x=top_10_products.values, y=top_10_products.index)
plt.title('Top 10 Best-Selling Products')
plt.show()
```

問い③：国別売上

実習１ 分析実践 Part３

タスク4：国別売上の分析

集計コード：

```
# 「国(Country) ごとに、売上(TotalPrice) を 合計(sum) する」
country_sales =
df_cleaned.groupby('Country')['TotalPrice'].sum().sort_values(ascending=False)
```

集計結果（抜粋）：

United Kingdom	6,747,155.95
Netherlands	283,373.40
EIRE	250,001.38
Germany	208,091.21

UKを除外して可視化

実習１ 分析実践 Part３

UKを除外して比較する

```
# 'United Kingdom' を除外(.drop)して、上位10件を取得
top_10_excluding_uk = country_sales.drop('United Kingdom').head(10)

# 棒グラフで可視化
plt.figure(figsize=(12, 7))
sns.barplot(x=top_10_excluding_uk.values,
y=top_10_excluding_uk.index)
plt.title('Top 10 Countries by Sales (Excluding UK)')
plt.show()
```

実習１ Part３のまとめ

実習１ 分析実践 Part３

✓ きれいなデータに「問い」を投げかけ、
集計・可視化を行いました

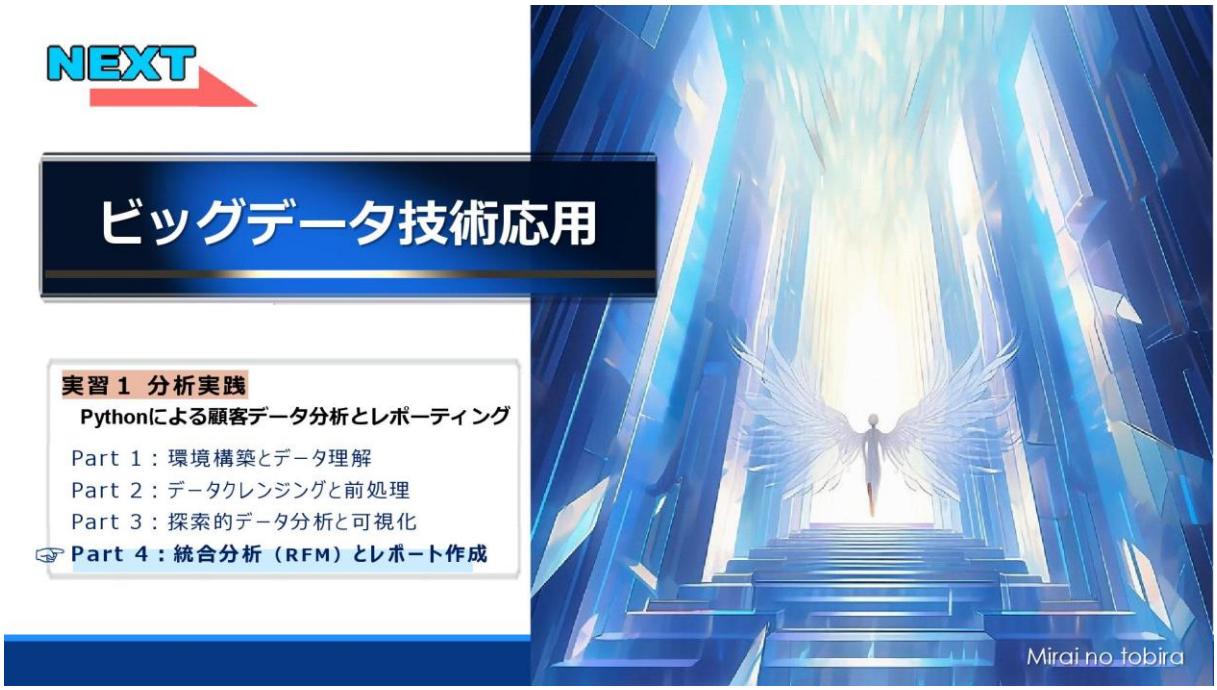
- ✅ **集計** ：`.groupby()` が最強の武器
- ✅ **可視化**（トレンド） ： 折れ線グラフ (`plt.plot`)
- ✅ **可視化**（ランキング） ： 棒グラフ (`sns.barplot`)
- ✅ **考察：** グラフから傾向や仮説を読み取ることがゴール

NEXT

Part４：統合分析（RFM）とレポート作成

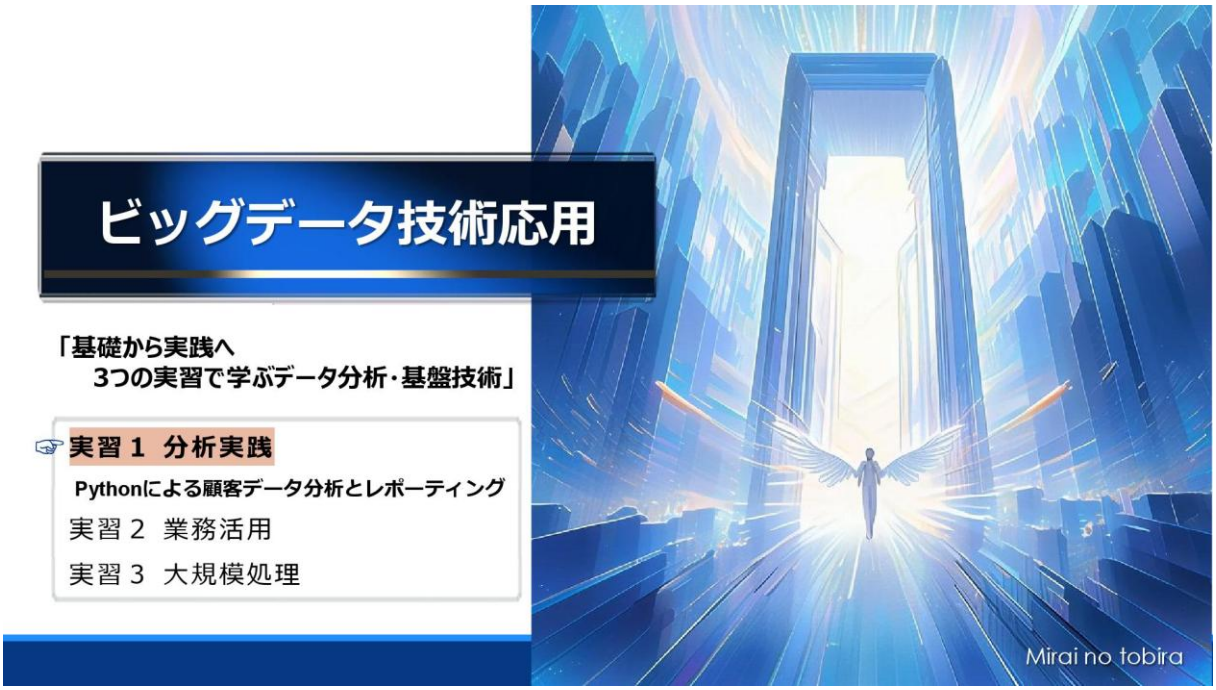
実習１ 分析実践

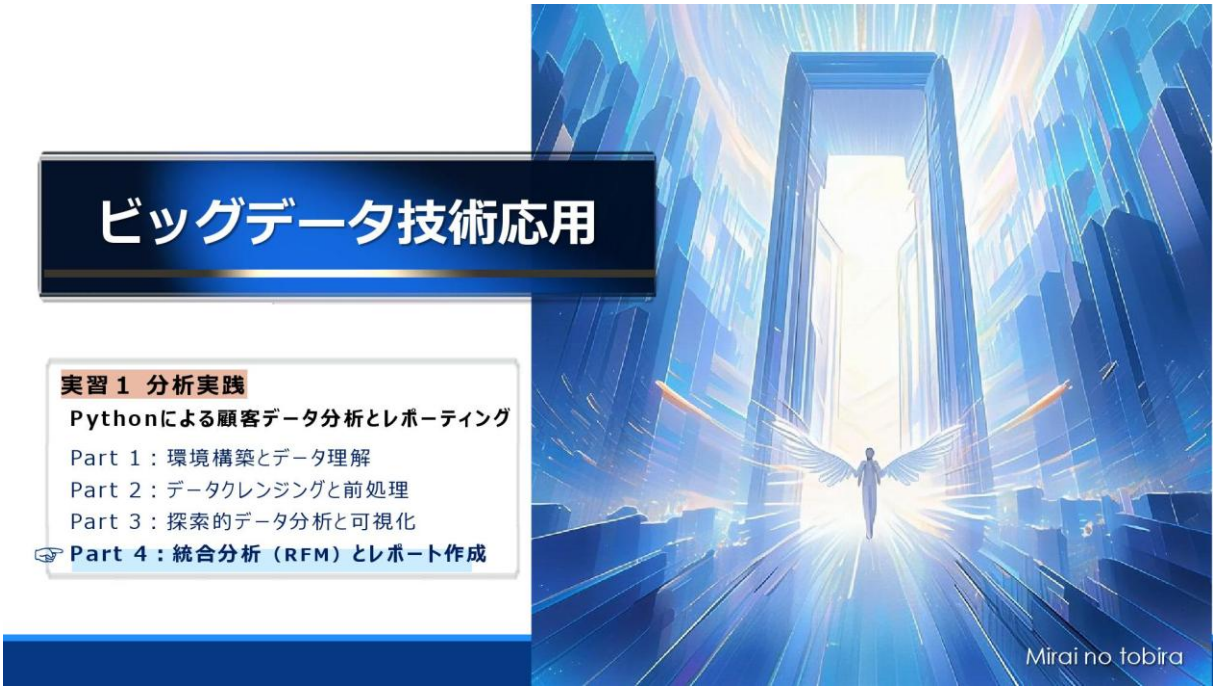
実習１ Part４では、
統合分析（RFM）とレポート作成
分析の「問い」を「顧客」に絞ります


NEXT
ビッグデータ技術応用
実習１ 分析実践
Pythonによる顧客データ分析とレポーティング
Part 1：環境構築とデータ理解
Part 2：データクレンジングと前処理
Part 3：探索的データ分析と可視化
Part 4：統合分析（RFM）とレポート作成
Mirai no tobira

ビッグデータ技術応用__実習１

Part4


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
Pythonによる顧客データ分析とレポーティング
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


ビッグデータ技術応用
実習１ 分析実践
Pythonによる顧客データ分析とレポーティング
Part １：環境構築とデータ理解
Part ２：データクレンジングと前処理
Part ３：探索的データ分析と可視化
Part ４：統合分析（RFM）とレポート作成
Mirai no tobira

実習 1　分析実践

ビッグデータ技術応用

実習 1 分析実践
Pythonによる
顧客データ分析と
レポーティング

実習 2　業務活用

実習 3　大規模処理

Part 1：環境構築とデータ理解
Part 2：データクレンジングと前処理
Part 3：探索的データ分析と可視化
Part 4：統合分析（RFM）とレポート作成

Part 4：統合分析（RFM）とレポート作成

実習 1　分析実践

分析結果を価値に変える：顧客理解と提案

Part 4：統合分析（RFM）とレポート作成の達成目標

実習 1 分析実践

- ✓ 顧客分析のフレームワーク「RFM分析」を理解し、実装できる
- ✓ 顧客をスコアリングし「優良顧客」や「離反顧客」を特定できる
- ✓ セグメントごとの特徴を分析し、施策のヒントを得る
- ✓ 分析プロセス全体の結果を統合し、レポートとしてまとめる

RFM分析とは？

実習 1 分析実践

顧客を3つの視点で評価：RFM分析

顧客の価値を測るための、シンプルで強力なフレームワークです

R	F	M
Recency	Frequency	Monetary
最新購買日	購買頻度	購買金額
(最近来たか？)	(何回買ったか？)	(いくら使ったか？)

🎯 なぜRFMが重要？

「Rが高く、Fが高く、Mが高い顧客」＝ 最も重要な優良顧客

この顧客層を特定し、維持・育成することがビジネス成長の鍵です

RFM指標の計算

実習 1 分析実践 Part 4

タスク2-1：RFM指標の計算

顧客ごとにR, F, Mを計算します

```
# 基準日を設定（最終取引日の翌日）
snapshot_date = df_cleaned['InvoiceDate'].max() + dt.timedelta(days=1)

# 顧客ごとに集計
rfm_data = df_cleaned.groupby('CustomerID').agg({
'InvoiceDate': lambda date: (snapshot_date - date.max()).days, # Recency
'InvoiceNo': 'nunique', # Frequency
'TotalPrice': 'sum' # Monetary
})

# 列名変更
rfm_data.rename(columns={...}, inplace=True)
```


RFMスコアリング

実習 1 分析実践 Part 4

タスク2-2：RFMスコアの計算

R, F, Mの値を、それぞれ1～5点のスコアに変換します

pd.qcut() でデータを5等分します

Recency: 短いほど高スコア (5点)／**Frequency:** 多いほど高スコア (5点)／**Monetary:** 高いほど高スコア (5点)

```
# Recency（値が小さいほど良い）
r_labels = range(5, 0, -1)
rfm_data['R_Score'] = pd.qcut(rfm_data['Recency'], q=5, labels=r_labels, ...)
# Frequency, Monetary（値が大きいほど良い）
fm_labels = range(1, 6)
rfm_data['F_Score'] = pd.qcut(rfm_data['Frequency'].rank(...), q=5,
labels=fm_labels, ...)
rfm_data['M_Score'] = pd.qcut(rfm_data['Monetary'], q=5, labels=fm_labels, ...)
```

顧客セグメンテーション

実習 1 分析実践 Part 4

タスク2-3：RFMセグメントの作成

RとFのスコアを使って、顧客をグループ分け（セグメンテーション）します

セグメント例：

- ▶ Champions (R≥4, F≥4)：最優良顧客
- ▶ Loyal Customers (R≥3, F≥3)：優良顧客
- ▶ Potential Loyalists (R≥4, F<3)：育成候補
- ▶ At Risk Customers (R<3, F≥4)：離反危険
- ▶ Lost Customers (R<3, F<3)：離反顧客

セグメント分析

実習 1 分析実践 Part 4

タスク2-4：セグメント分析と可視化

作成したセグメントごとに、顧客数や平均購入額などを比較します

```
# セグメントごとに集計
segment_summary = rfm_data.groupby('Segment_Name').agg(...)

# 顧客数を棒グラフで表示
sns.countplot(y='Segment_Name', data=rfm_data, ...)
plt.show()
```

レポート作成のポイント

実習１ 分析実践　Part４

タスク3：分析レポートを作成する

✓ 分析は「伝える」までがセット

レポート作成のポイント：

① **目的**　：なぜこの分析をしたのか？
② **データ**：何を使い、どう処理したか？
③ **発見**　：何がわかったか？ (グラフとともに)
④ **提案**　：だから、どうすべきか？ (ネクストステップ)

実習１ のまとめ

実習１ 分析実践

4つのPartを通して、データ分析の基本プロセスを完走しました

- ✅ **生データの理解と問題発見** (Part 1)
- ✅ **データのクレンジングと前処理** (Part 2)
- ✅ **集計と可視化によるインサイト発見** (Part 3)
- ✅ **RFM分析による顧客理解とレポート作成** (Part 4)

NEXT

実習２ 業務活用

ビッグデータ技術応用

実習１の経験を活かし

実習２ 業務活用（ETL処理とBIツール）

実習３ 大規模処理 （ Spark ）

へと進みましょう

NEXT

ビッグデータ技術応用

「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」

実習１ 分析実践

☞ **実習２ 業務活用**

アクセスログETL処理とBIツールによる可視化

実習３ 大規模処理

ビッグデータ技術応用_実習２

導入編


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
アクセスログETL処理とBIツールによる可視化
実習３ 大規模処理
Mirai no tobira

実習構成

ビッグデータ技術応用

実習１ 分析実践
Python による
顧客データ分析と
レポーティング

→

実習２ 業務活用
アクセスログETL処理と
BIツールによる可視化

→

実習３ 大規模処理
Spark による
大規模データ処理

ビッグデータ技術基礎		ビッグデータ技術応用		
第1章 ビッグデータとその活用	→	実習１		
第2章 データ分析基礎	→	実習１		
第3章 スクリプト言語による分析	→	実習１	**実習２**	
第4章 ツールとモニタリング	→		**実習２**	
第5章 列指向ストレージによる高速化	→			実習３
第6章 データマートの基本構造	→		**実習２**	
第7章 大規模分散処理	→			実習３

学習のステップ

ビッグデータ技術応用

実習１では、比較的整理されたCSVデータを扱ったが、実世界のデータは「生の」扱いにくい形をしていることが多く、実習２で扱う「アクセスログ」は、その代表例で、
Webサーバーが自動で記録したテキストデータ
ここから分析に必要な情報を取り出し(Extract)、使いやすい形に変換し(Transform)、最終的に分析ツールで読み込める形にする(Load)
この一連の流れを**「ETL処理」**と呼ぶ

✓ **実習２の前半では、Pythonを使ってこのETL処理を実践します**

実習で使用するツール

ビッグデータ技術応用

実習２の後半では、ETL処理で作ったデータを**「BIツール」**で可視化

Tableau (タブロー) と **Power BI** (パワービーアイ) という2つのツールを使う

これらのツールを使うと、プログラミングなしに、ドラッグ＆ドロップなどの

簡単な操作で、インタラクティブなグラフやダッシュボードを素早く作成できる

実習２ 業務活用

ビッグデータ技術応用

実習１ 分析実践 → **実習２ 業務活用**
アクセスログETL処理と
BIツールによる可視化 → 実習３ 大規模処理

Part１：アクセスログとETLの基礎

Part２：データ型の整備とデータマートの基礎

Part３：Tableau Publicによるダッシュボード作成

Part４：Power BI Desktopによるダッシュボード作成

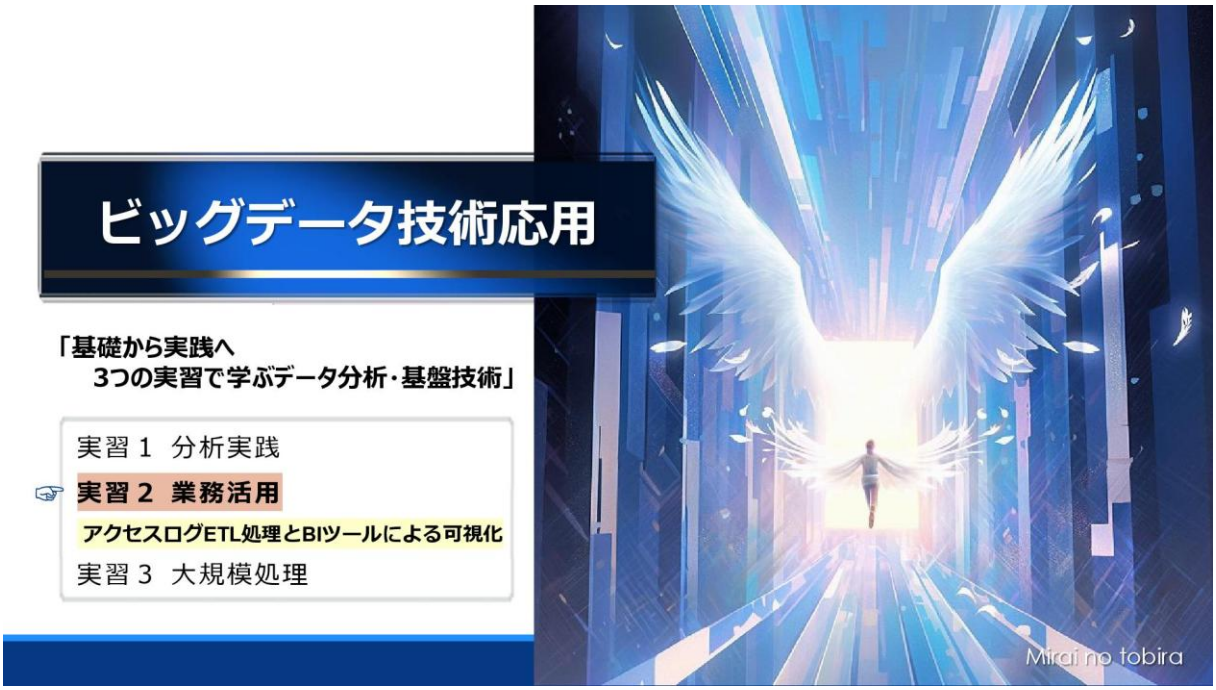
▶ データ加工（Python）から可視化（BIツール）までを
一気通貫で体験できる

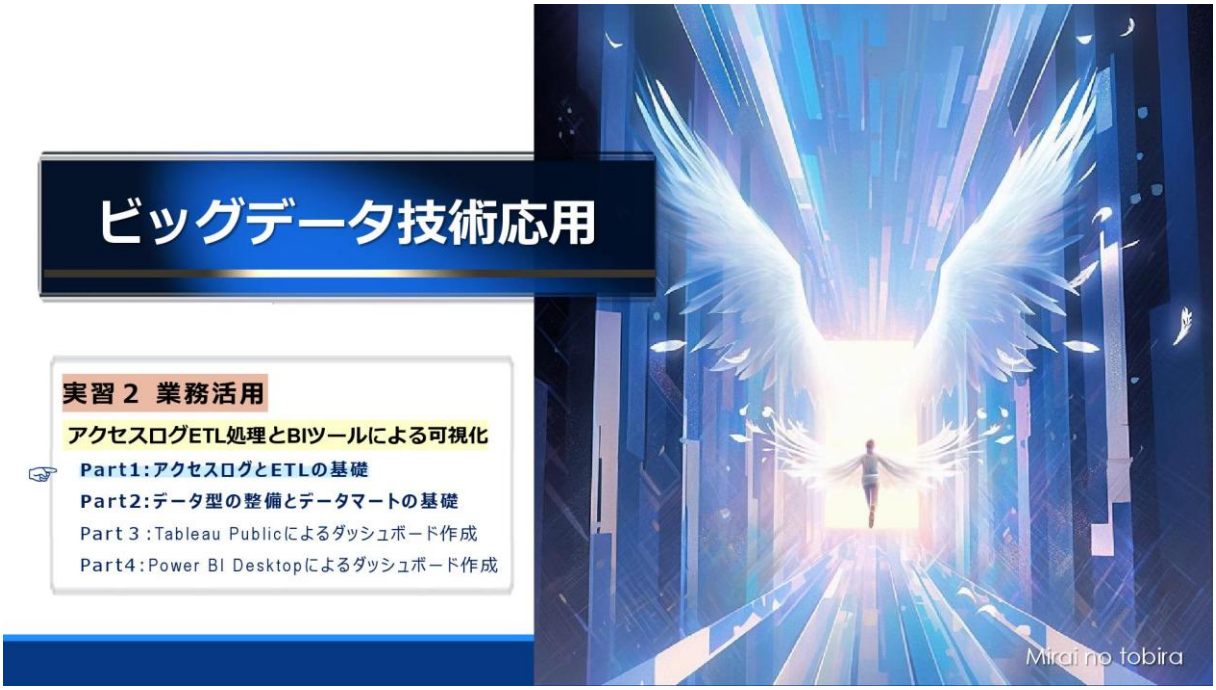

ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


NEXT
ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
アクセスログETL処理とBIツールによる可視化
実習３ 大規模処理
Mirai no tobira

ビッグデータ技術応用_実習２

Part1


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１　分析実践
実習２　業務活用
アクセスログETL処理とBIツールによる可視化
実習３　大規模処理
Mirai no tobira


ビッグデータ技術応用
実習２　業務活用
アクセスログETL処理とBIツールによる可視化
Part1:アクセスログとETLの基礎
Part2:データ型の整備とデータマートの基礎
Part３:Tableau Publicによるダッシュボード作成
Part4:Power BI Desktopによるダッシュボード作成
Mirai no tobira

実習２ 業務活用

ビッグデータ技術応用

実習１ 分析実践 → **実習２ 業務活用**（アクセスログETL処理とBIツールによる可視化） → 実習３ 大規模処理

Part１：アクセスログとETLの基礎

Part２：データ型の整備とデータマートの基礎

Part３：Tableau Publicによるダッシュボード作成

Part４：Power BI Desktopによるダッシュボード作成

実習２ Part １：アクセスログとETLの基礎

実習２ 業務活用

Webサイトの足跡を宝に変える技術

Part 1：アクセスログとETLの基礎 の達成目標

実習 2 業務活用

- ✓ **Webアクセスログがどのような情報か理解する**
- ✓ **ETL**（Extract, Transform, Load）**の概念を理解する**
- ✓ **Pythonでテキストファイルを読み込み、行ごとに処理する方法を学ぶ**
- ✓ **正規表現を使ってログから必要な情報**（IP, 日時, URL等）**を抽出する**
- ✓ **抽出したデータを構造化**（DataFrame）**する方法を学ぶ**

Webアクセスログとは？

実習 2 業務活用

Webサーバーの『日記』：アクセスログ

Webサーバーに「誰が」「いつ」「どのページに」アクセスしたかの記録です

一般的な形式 (Common Log Format):

```
127.0.0.1 - - [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

含まれる主な情報：

- ▶ IPアドレス　：アクセス元のコンピュータ
- ▶ タイムスタンプ　：アクセス日時
- ▶ リクエスト　：どのページ(URL)を見たか
- ▶ ステータスコード: 結果 (200=成功, 404=Not Found)
- ▶ 転送量　：送信したデータサイズタスク1：環境構築

ログデータの課題

実習２ 業務活用 Part１

生のログデータは『扱いにくい』

アクセスログは宝の山ですが、そのままでは分析できません

- ▶ **非構造化データ**　：ただのテキスト。Excelで開いても意味不明
- ▶ **大量**　：人気サイトなら1日数十GB～TBになることも
- ▶ **形式が微妙に違うことも**：サーバーの種類や設定で形式が変わる

→ 分析できる「綺麗な形」に加工する必要がある

ETLとは？

実習２ 業務活用 Part１

データを『変身』させる魔法：ETL

扱いにくい生データを、分析しやすい綺麗なデータに加工するプロセスです

E: Extract 抽出	→	T: Transform 変換・加工	→	L: Load 書き出し
ログファイルから データを読み込む		必要な情報を取り出し 形式を整える		分析ツールで読み込める 形式で保存

🎯 実習2のゴール

このETLプロセスをPythonで実装し、最終的にBIツールで可視化できるデータ（データマート）を作成します

データの準備

実習２ 業務活用 Part１

タスク1：データの準備

実習では、サンプルアクセスログファイル (access_log.txt) を使います
（実際のログがない場合は、提供されるPythonスクリプトで生成します）

ステップ1：ファイル読み込み

```
# ファイルを開いて一行ずつ読み込む準備
file_path = 'access_log.txt'
with open(file_path, 'r') as f:
# 最初の数行を表示して確認
for i, line in enumerate(f):
if i < 5:
print(line.strip()) # strip()で改行文字を削除
else:
break
```


情報抽出の武器：正規表現

実習２ 業務活用 Part１

テキストから宝を探す：正規表現

ログのような「決まったパターン」を持つテキストから、
特定の情報（IPアドレス、日時など）を抜き出すための強力なツールです

例：IPアドレスを抜き出すパターン

```
r"(¥d{1,3}¥.¥d{1,3}¥.¥d{1,3}¥.¥d{1,3})"
```

- ¥d{1,3}：1～3桁の数字
- ¥.：ピリオド（特殊文字なので ¥ が必要）
- (...)：カッコで囲んだ部分を抽出

ポイント
正規表現は最初は難解に見えますが、ログ解析やテキスト処理には必須のスキルです。少しずつ慣れていきましょう

正規表現の実装 (Python)

実習２ 業務活用 Part１

タスク2：Pythonと正規表現で情報抽出

Pythonの re モジュールを使って、ログ1行から情報を抽出します

```
import re # 正規表現モジュール
# ログのパターンを定義 (Common Log Format用)
log_pattern = re.compile(r'(¥S+) (¥S+) (¥S+) ¥[([¥w:/]+¥s[+¥-]¥d{4})¥] "(¥S+)¥s?(¥S+)?¥s?(¥S+)?" (¥d{3}) (¥S+)')
# サンプルログ1行
line = '127.0.0.1 - - [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326'
# パターンにマッチさせる
match = log_pattern.match(line)
if match:
ip_address = match.group(1) # 1番目のカッコの中身
timestamp = match.group(4) # 4番目のカッコの中身
request_url = match.group(6) # 6番目のカッコの中身
status_code = match.group(8) # 8番目のカッコの中身
print(f"IP: {ip_address}, Time: {timestamp}, URL: {request_url}, Status: {status_code}")
```


正規表現の実装 (Python)

実習２ 業務活用 Part１

```
import re # 正規表現モジュール
# ログのパターンを定義 (Common Log Format用)
log_pattern = re.compile(r'(¥S+) (¥S+) (¥S+) ¥[([¥w:/]+¥s[+¥-]¥d{4})¥]
"(¥S+)¥s?(¥S+)?¥s?(¥S+)?" (¥d{3}) (¥S+)')
# サンプルログ1行
line = '127.0.0.1 - - [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326'
# パターンにマッチさせる
match = log_pattern.match(line)
if match:
ip_address = match.group(1) # 1番目のカッコの中身
timestamp = match.group(4) # 4番目のカッコの中身
request_url = match.group(6) # 6番目のカッコの中身
status_code = match.group(8) # 8番目のカッコの中身
print(f"IP: {ip_address}, Time: {timestamp}, URL: {request_url}, Status: {status_code}")
```

Part 1：アクセスログとETLの基礎　ビッグデータ技術応用 実習２

全ログデータの処理

実習２ 業務活用 Part１

タスク3：全ログデータをループ処理

ログファイル全体を1行ずつ読み込み、正規表現で情報を抽出し、リストに格納していきます

```
parsed_data = [] # 結果を格納するリスト
with open(file_path, 'r') as f:
for line in f:
match = log_pattern.match(line)
if match:
# 抽出した情報を辞書形式でリストに追加
data = { 'ip': match.group(1),
'time': match.group(4),
'url': match.group(6),
'status': match.group(8)
# ... 他の情報も必要なら追加 ...}
parsed_data.append(data)
print(f"処理完了！ {len(parsed_data):,} 件のログを解析しました。")
```


全ログデータの処理

実習２ 業務活用 Part１

```
parsed_data = [] # 結果を格納するリスト
with open(file_path, 'r') as f:
for line in f:
match = log_pattern.match(line)

if match:
# 抽出した情報を辞書形式でリストに追加
data = {
'ip': match.group(1),
'time': match.group(4),
'url': match.group(6),
'status': match.group(8)

# ... 他の情報も必要なら追加 ...
}
parsed_data.append(data)
print(f"処理完了！ {len(parsed_data):,} 件のログを解析しました。")
```

Part 1：アクセスログとETLの基礎 ビッグデータ技術応用 実習２

DataFrame への変換

実習２ 業務活用 Part１

タスク4：構造化データ (DataFrame) へ

解析結果（辞書のリスト）を、pandas DataFrame に変換します
これでやっと分析できる形になります

```
import pandas as pd

# リストからDataFrameを作成
log_df = pd.DataFrame(parsed_data)

# 結果を確認
log_df.info() # データ型や欠損値を確認
log_df.head() # 最初の5行を表示
```


実習２ Part１のまとめ

実習２ 業務活用 Part１

生のアクセスログを分析可能な形にする第一歩を踏み出しました

- ✅ アクセスログの内容とETLの概念を理解した
- ✅ Pythonでテキストファイルを読み込む方法を学んだ
- ✅ 正規表現を使って必要な情報を抽出するスキルを身につけた
- ✅ 抽出結果を DataFrame に変換し、構造化できた

NEXT

Part２：データ型の整備とデータマートの基礎

実習２ 業務活用

実習２ Part２では、
Part１で作成した DataFrame を
さらに使いやすくします

NEXT

ビッグデータ技術応用

実習２ 業務活用

アクセスログETL処理とBIツールによる可視化

Part1：アクセスログとETLの基礎

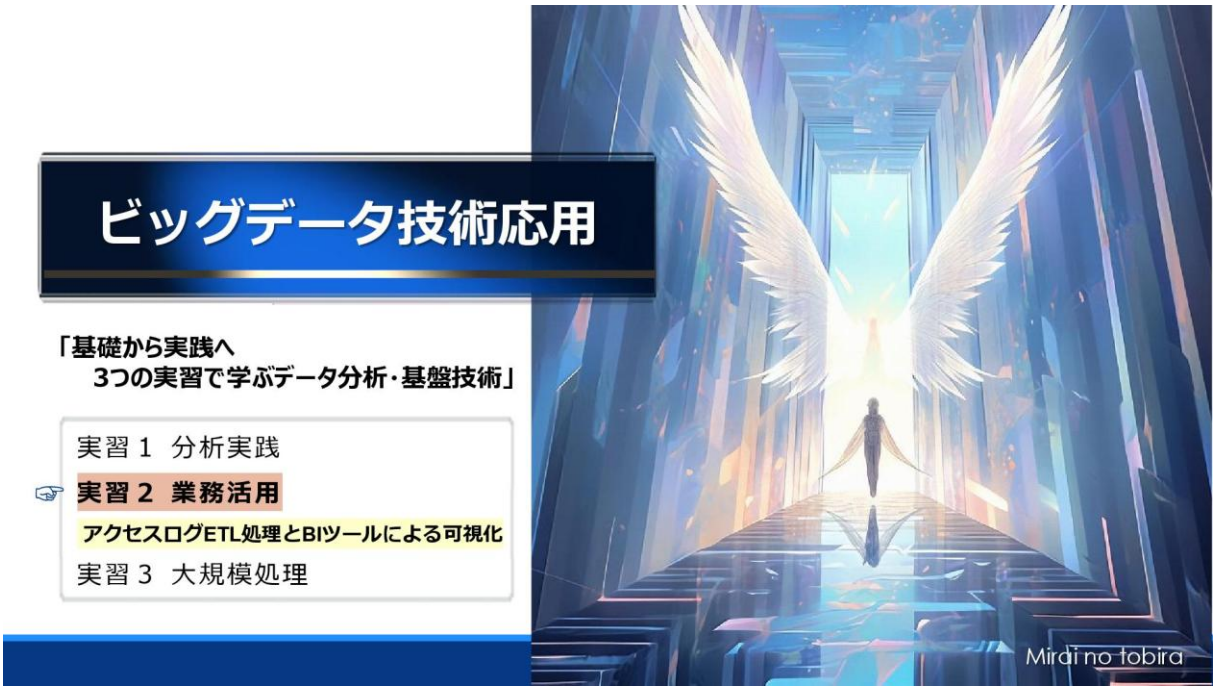
Part2:データ型の整備とデータマートの基礎

Part３：Tableau Publicによるダッシュボード作成

Part4：Power BI Desktopによるダッシュボード作成

ビッグデータ技術応用_実習２

Part2


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
アクセスログETL処理とBIツールによる可視化
実習３ 大規模処理
Mirai no tobira


ビッグデータ技術応用
実習２ 業務活用
アクセスログETL処理とBIツールによる可視化
Part1：アクセスログとETLの基礎
Part2:データ型の整備とデータマートの基礎
Part３:Tableau Publicによるダッシュボード作成
Part4:Power BI Desktopによるダッシュボード作成
Mirai no tobira

実習２ 業務活用

ビッグデータ技術応用

実習１ 分析実践 → **実習２ 業務活用**（アクセスログETL処理とBIツールによる可視化） → 実習３ 大規模処理

Part１：アクセスログとETLの基礎

Part２：データ型の整備とデータマートの基礎

Part３：Tableau Publicによるダッシュボード作成

Part４：Power BI Desktopによるダッシュボード作成

Part２：データ型の整備とデータマート基礎

実習２ 業務活用

分析の下準備：データを『使える』形にする

Part 2 : データ型の整備とデータマート基礎 の達成目標

実習 2 業務活用

- ✓ **ログのタイムスタンプ**（文字列）**を日時型** (datetime) **に変換できる**
- ✓ **日時データから年・月・曜日・時間などを抽出できる**
- ✓ **データマートの考え方を理解し、設計できる**
- ✓ **分析に必要な列を選択し、CSVファイルとして保存できる**

Part 1 のおさらい

実習 2 業務活用 Part 2

Part 1 の成果：構造化データへ

Part 1 では、生のログテキストを解析し、
log_df というDataFrameを作成しました

▶ **Part 2 の課題**

- ➢ この `log_df` はまだ「下ごしらえ」が必要です
- ➢ 特に `time` 列が文字列のままでは、時間軸での分析ができません

日時データ処理の重要性

実習２ 業務活用 Part２

タスク2：なぜ日時データ処理が重要？

文字列のまま (object)

- ▶ 月ごとの集計ができない
- ▶ 時間帯別の分析ができない
- ▶ 曜日による違いが見れない
- ▶ 並び替えが文字順になる

➡

日時型へ変換 (datetime)

- ▶ 月・曜日・時間などを抽出可能
- ▶ 時間軸での集計・分析が可能
- ▶ 時系列グラフの作成が可能
- ▶ 日付順での並び替えが可能

🎯 結論

時間に関する分析（いつアクセスが多いかなど）を行うには、**必ず日時型への変換が必要**

pd.to_datetime

実習２ 業務活用 Part２

文字列を日時に変身：pd.to_datetime

pandasの強力な関数です

ただし、元の文字列の「形式」を正しく指定する必要があります

ログの形式: `'10/Oct/2000:13:55:36 -0700'`

対応する書式指定子 (format): `'%d/%b/%Y:%H:%M:%S %z'`

```
log_df['timestamp'] = pd.to_datetime(log_df['time'],
format='%d/%b/%Y:%H:%M:%S %z', errors='coerce')
```

Part２：データ型の整備とデータマート基礎　ビッグデータ技術応用 実習２

日時からの特徴量抽出

実習２ 業務活用 Part２

日時データの『解剖』：.dt アクセサ

日時型に変換すると、.dt という便利な道具箱が使えます

```
# 年月日時などを抽出
log_df['year'] = log_df['timestamp'].dt.year
log_df['month'] = log_df['timestamp'].dt.month
log_df['day'] = log_df['timestamp'].dt.day
log_df['hour'] = log_df['timestamp'].dt.hour
log_df['dayofweek'] = log_df['timestamp'].dt.dayofweek # 月=0, 日=6
log_df['dayname'] = log_df['timestamp'].dt.day_name() # Monday, Tuesday...
```


URLクリーニング (補足)

実習２ 業務活用 Part２

タスク3：URLを綺麗にする (オプション)

URL末尾のパラメータ (?id=1 など) が不要な場合
.str.split('?') で除去できます

```
log_df['url_cleaned'] = log_df['url'].str.split('?').str[0]
```

データマートとは？

実習２ 業務活用 Part２

タスク4：分析用の『売店』を作る：データマート

分析目的に合わせて、必要な情報だけを選び、使いやすく整理したデータセット

元のデータ (log_df)	データマート (access_mart_df)
▶ 全情報を含む	▶ 今回の分析に必要な列だけ
▶ データ量が大きい	▶ データ量が小さい
▶ 不要な列もある	▶ BIツールで高速に動作

🎯 **今回の設計**
「いつ」「誰が」「どのページを」見たかを分析するため、
`timestamp`, `ip`, `url`, `status` 及び抽出した日時関連列を選択します

データマート作成＆保存

実習２ 業務活用 Part２

タスク5：データマート作成と保存 (Load)

ステップ1：列選択

```
mart_columns = ['timestamp', 'year', ..., 'status'] # 必要な列リスト
access_mart_df = log_df_final[mart_columns].copy()
```

ステップ2：CSVファイルへ保存

```
output_file_name = 'access_log_mart.csv'
access_mart_df.to_csv(output_file_name, index=False) # index=False が重要
```

🎯 **ETL完了！**
これでETLプロセス (Extract, Transform, Load) が完了！`access_log_mart.csv` がPart 3&4の分析データです

実習２ Part２のまとめ

実習２ 業務活用 Part２

扱いにくいログデータを、BIツールで使える綺麗なデータマートに加工しました

- ✅ `pd.to_datetime` で日時型に変換する重要性と方法
- ✅ `.dt` アクセサで日時から特徴量を抽出する技術
- ✅ データマートの考え方と設計
- ✅ `.to_csv` で処理結果を保存するETLのLoadステップ

NEXT

Part３：Tableau Public によるダッシュボード作成

実習２ 業務活用

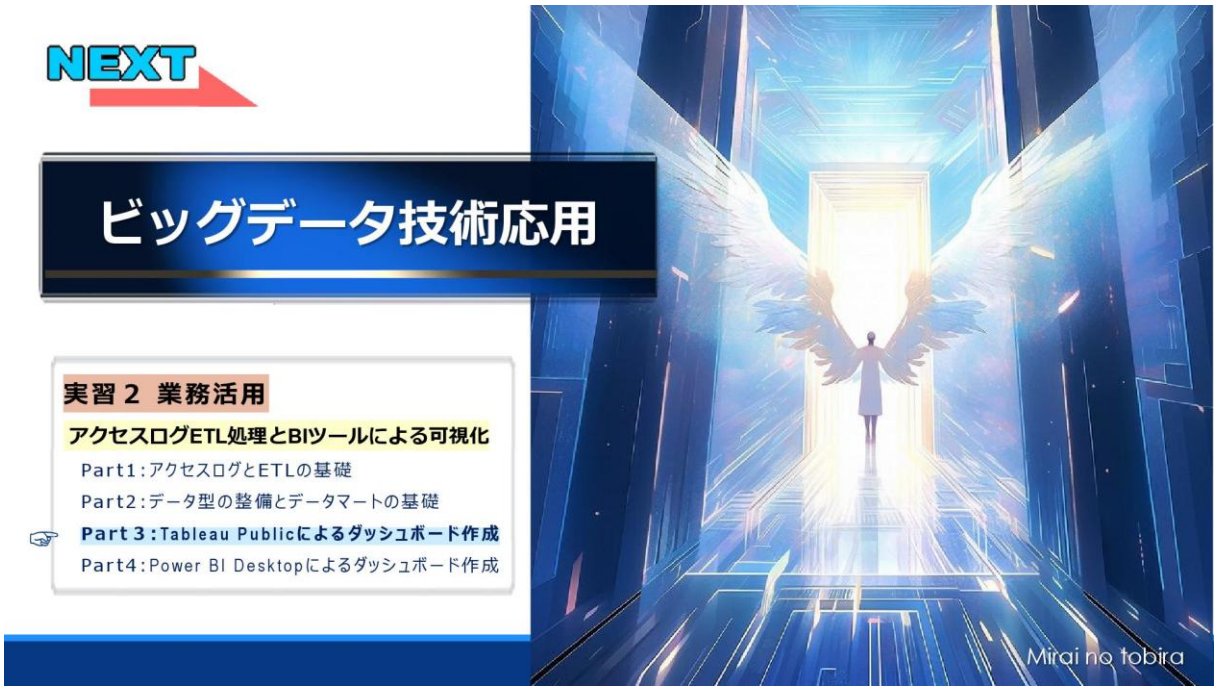
このあと実習２では、

Part２で作成した access_log_mart.csv を使って

Part３：Tableau Public でインタラクティブなダッシュボードの作成

Part４：Power BI Desktop で同様のダッシュボードの作成

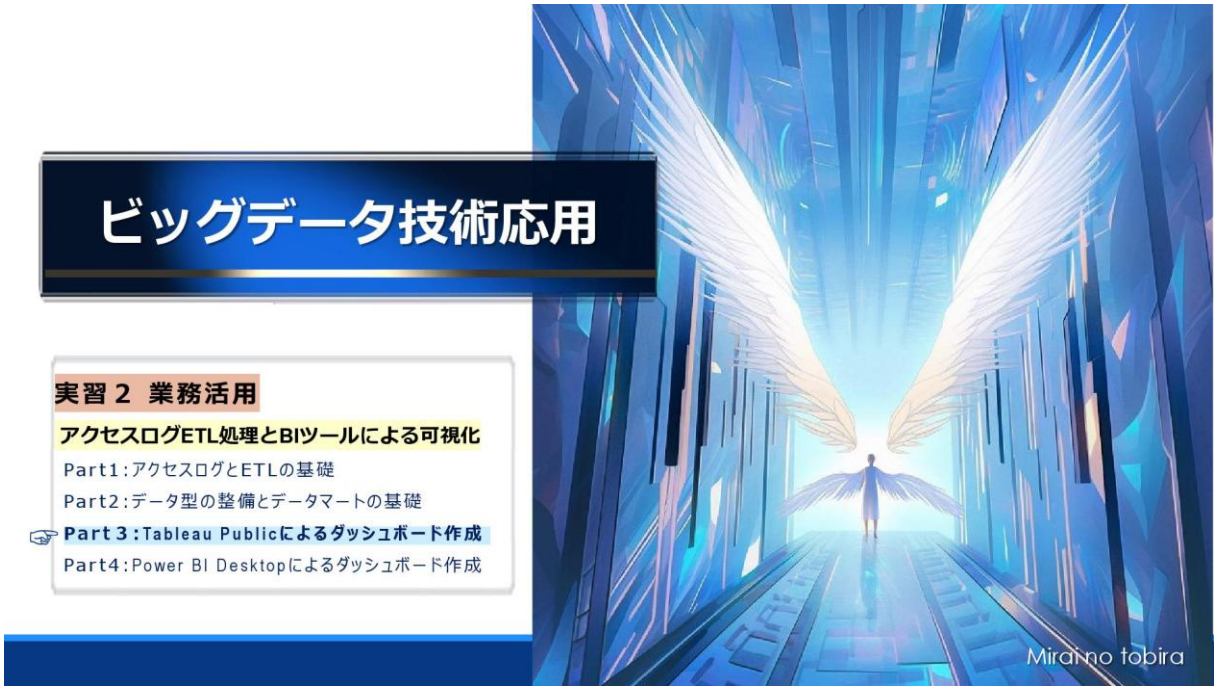
をします


NEXT
ビッグデータ技術応用
実習２ 業務活用
アクセスログETL処理とBIツールによる可視化
Part1：アクセスログとETLの基礎
Part2：データ型の整備とデータマートの基礎
Part３：Tableau Publicによるダッシュボード作成
Part4：Power BI Desktopによるダッシュボード作成
Mirai no tobira

ビッグデータ技術応用_実習２

Part3


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１　分析実践
実習２　業務活用
アクセスログETL処理とBIツールによる可視化
実習３　大規模処理
Mirai no tobira


ビッグデータ技術応用
実習２　業務活用
アクセスログETL処理とBIツールによる可視化
Part1：アクセスログとETLの基礎
Part2：データ型の整備とデータマートの基礎
Part３：Tableau Publicによるダッシュボード作成
Part4：Power BI Desktopによるダッシュボード作成
Mirai no tobira

実習２ 業務活用

ビッグデータ技術応用

実習１ 分析実践 → **実習２ 業務活用**（アクセスログETL処理とBIツールによる可視化） → 実習３ 大規模処理

Part１：アクセスログとETLの基礎
Part２：データ型の整備とデータマートの基礎
☞ **Part３：Tableau Publicによるダッシュボード作成**
Part４：Power BI Desktop によるダッシュボード作成

Part３：Tableau Public によるダッシュボード作成

実習２ 業務活用

データを『見える化』する力を身につける

Part 3 :Tableau Public によるダッシュボード作成 の達成目標

実習 2 業務活用

- ✓ Tableau Publicの基本操作（データ接続、画面構成）を理解する
- ✓ CSVデータをTableauに接続し、データ型を設定できる
- ✓ ディメンションとメジャーを使ってグラフを作成できる
- ✓ 複数のグラフを組み合わせてダッシュボードを構築できる
- ✓ ダッシュボードにインタラクション（フィルター）を設定できる
- ✓ 作成した成果をWebに公開できる

Tableau Publicとは？

実習 2 業務活用 Part 3

BIツール：Tableau Public

- ▶ 世界中で広く使われているBI (Business Intelligence) ツールの一つ
- ▶ ドラッグ＆ドロップで簡単に高機能なグラフやダッシュボードが作成可能
- ▶ **Public版は無料**で利用できるが、作成物はWebに公開される
- ▶ データからインサイトを得るための強力な武器

なぜBIツール？
Python (matplotlib/seaborn) でもグラフは作れますが、BIツールはより
手軽に（コード不要）／**インタラクティブに**（操作できる）／**美しく** データを可視化できます

準備：ツールとデータ

実習２ 業務活用 Part３

タスク1：準備するもの

① **Tableau Public Desktop:**

▶未インストールの場合は公式サイトからダウンロード＆インストール

▶Tableau Publicアカウントでログイン

② **データファイル:**

▶Part 2で作成した access_log_mart.csv

データ接続と型設定

実習２ 業務活用 Part３

タスク2：データをTableauに接続

手順：

1. Tableau Public 起動 →「接続」→「テキストファイル」
2. access_log_mart.csv を選択
3. データソース画面で列とデータ型を確認
4. **重要：**
   ip 列のデータ型アイコンをクリック →「地理的役割」→「IPアドレス」を設定

ワークシート画面

実習２ 業務活用 Part３

グラフ作成の舞台：ワークシート

データ接続後、「シート1」をクリックしてワークシート画面へ

グラフ作成① 折れ線

実習２ 業務活用 Part３

タスク3-2：時間帯別アクセス（折れ線グラフ）

手順：

1. **メジャー「レコード数」**を**「行」**シェルフへドラッグ
2. **ディメンション「Hour」**を**「列」**シェルフへドラッグ
3. → 自動で折れ線グラフが完成

グラフ作成② 棒グラフ

実習２ 業務活用 Part３

タスク3-3：曜日別アクセス（棒グラフ）

手順：

1. 新しいシートを作成
2. **メジャー「レコード数」**を**「行」**シェルフへ
3. **ディメンション「Dayname」**を**「列」**シェルフへ
4. → 自動で棒グラフが完成！
5. （オプション）「Dayname」の並び順を調整

グラフ作成③ 地図

実習２ 業務活用 Part３

タスク3-4：アクセス元マップ

手順：

1. 新しいシートを作成
2. **ディメンション「Ip」（地球儀アイコン）**をダブルクリック
3. → 自動で地図が表示
4. **メジャー「レコード数」**をマークカードの**「サイズ」**へドラッグ
5. → アクセス数に応じて点のサイズが変わる

ダッシュボード作成

実習２ 業務活用 Part３

タスク4：ダッシュボードで統合

作成した３つのグラフ（シート）を1画面にまとめます

手順：

1. 新しいダッシュボードを作成（田アイコン）
2. 左側のシート一覧から、各シートを右側の領域へドラッグ＆ドロップして配置
3. サイズやレイアウトを調整

インタラクション設定

実習２ 業務活用 Part３

タスク4-3：ダッシュボードを『動かす』

グラフ同士を連動させます

手順：

1. ダッシュボード上の「アクセス元マップ」シートを選択
2. 右上に表示される「フィルターとして使用」（漏斗マーク）をクリック
3. → これで地図をクリックすると他のグラフが連動する

🎯 **インタラクションの価値**
見る人が自由にデータを深掘りできるようになり、分析の幅が広がります

保存と公開

実習２ 業務活用 Part３

タスク5：成果を世界に公開

Tableau Public では、作成したワークブックは
サーバーに保存・公開します

手順：

1. 「ファイル」→「Tableau Public に保存」
2. ログイン → 名前を付けて「保存」
3. → 自動的にWebブラウザで公開ページが開く

⚠ 注意
公開されたURLは誰でもアクセス可能です
個人情報や機密情報が含まれていないことを確認してください

実習２ Part３のまとめ

実習２ 業務活用 Part３

BIツールの代表格、Tableau Public の基本操作をマスターしました

- ✅ データ接続とデータ型の重要性を理解した
- ✅ ドラッグ＆ドロップで基本的なグラフを作成できた
- ✅ 複数のグラフを組み合わせてダッシュボードを構築できた
- ✅ フィルター機能でインタラティブな分析を体験した
- ✅ 作成した成果をWebで共有する方法を学んだ

NEXT

Part４：Power BI Desktop によるダッシュボード作成

実習２ 業務活用

実習２Part４では、
Part 2で作成した access_log_mart.csv を使って
BIPower BI Desktop でダッシュボードを作成します

NEXT

ビッグデータ技術応用

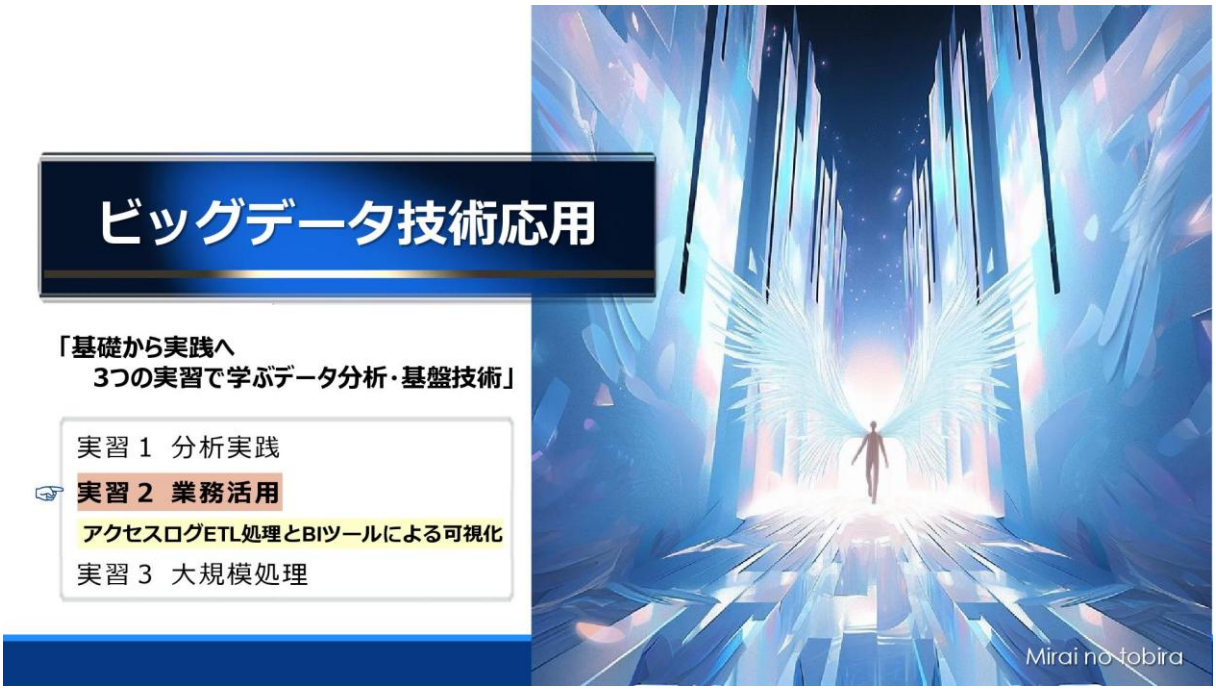
実習２ 業務活用

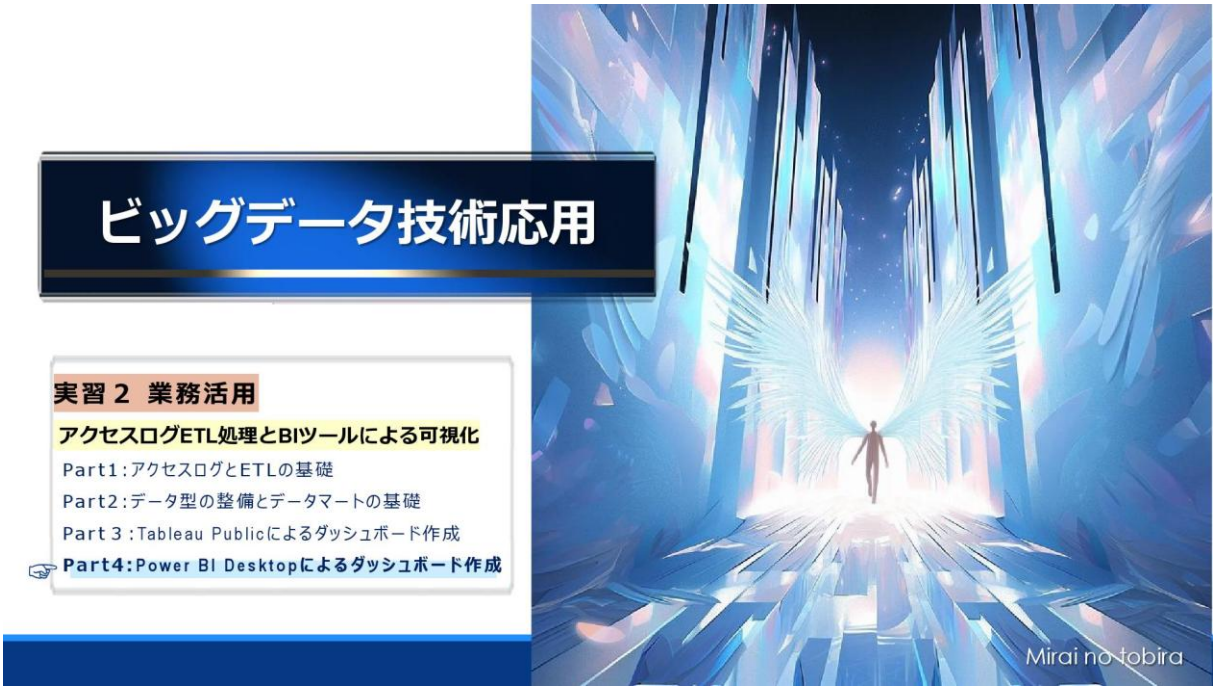
アクセスログETL処理とBIツールによる可視化

Part1：アクセスログとETLの基礎
Part2：データ型の整備とデータマートの基礎
Part３：Tableau Publicによるダッシュボード作成
Part4：Power BI Desktopによるダッシュボード作成

ビッグデータ技術応用_実習２

Part4


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１　分析実践
実習２　業務活用
アクセスログETL処理とBIツールによる可視化
実習３　大規模処理
Mirai no tobira


ビッグデータ技術応用
実習２　業務活用
アクセスログETL処理とBIツールによる可視化
Part1：アクセスログとETLの基礎
Part2：データ型の整備とデータマートの基礎
Part３：Tableau Publicによるダッシュボード作成
Part4：Power BI Desktopによるダッシュボード作成
Mirai no tobira

実習２ 業務活用

ビッグデータ技術応用

実習１ 分析実践 → **実習２ 業務活用**（アクセスログETL処理とBIツールによる可視化） → 実習３ 大規模処理

Part１：アクセスログとETLの基礎
Part２：データ型の整備とデータマートの基礎
Part３：Tableau Public によるダッシュボード作成
☞ **Part４：Power BI Desktop によるダッシュボード作成**

Part４：Power BI Desktop によるダッシュボード作成

実習２ 業務活用

もう一つのBIツール：Tableauとの違いは？

Part4:Power BI Desktopによるダッシュボード作成 の達成目標

実習２ 業務活用

- ✓ Power BI Desktopの基本操作（データ取得、画面構成）を理解する
- ✓ CSVデータをPower BIに取得し、Power Queryでデータ型を設定できる
- ✓ レポートビューで基本的な視覚化（グラフ）を作成できる
- ✓ 複数のグラフを配置してレポート（ダッシュボード）を構築できる
- ✓ グラフ間のインタラクションを確認し、レポートを保存できる

Power BI Desktopとは？

実習２ 業務活用 Part４

Microsoft の BIツール : Power BI Desktop

- ▶ Microsoftが提供する無料のBIツール（Windows専用）
- ▶ Excelライクな操作感＋高機能なデータ加工（Power Query）
- ▶ 豊富なグラフ種類と、柔軟なレポートデザインが可能
- ▶ Office 365 との連携が強力
- ▶ Web公開には Power BI Service（有償の場合あり）が必要

Tableau Publicとの主な違い

- ▶ **OS** ：Power BI DesktopはWindowsのみ、Tableau PublicはWin/Mac対応
- ▶ **公開** ：Power BIはローカル保存可、Tableau PublicはWeb公開必須
- ▶ **操作感** ：Power BIはExcel寄り、Tableauは独自のデザイン思想

準備：ツールとデータ

実習２ 業務活用 Part４

タスク1：準備するもの

① **Power BI Desktop:**

▶未インストールの場合は Microsoft Store等からインストール

▶Windows PCが必要

② **データファイル:**

▶Part 2で作成した access_log_mart.csv

データ取得と型設定

実習２ 業務活用 Part４

タスク2：データを Power BI に取得

手順：

1. Power BI起動 →「データを取得」→「テキスト/CSV」
2. access_log_mart.csv を選択 →「読み込み」
3. 右側「フィールド」ペインに列が表示される
4. 「ホーム」タブ →「データの変換」で Power Query エディターを開く
5. 各列のデータ型を確認・修正（例: timestampを日付/時刻へ）
6. 「閉じて適用」

レポートビュー画面

実習２ 業務活用 Part４

グラフ作成の舞台：レポートビュー

Power Query を閉じると、レポートビューに戻ります

視覚化の作成

実習２ 業務活用 Part４

タスク3：グラフ（視覚化）を作成する

基本手順：

1.「視覚化」ペインでグラフ種類を選択

2.「フィールド」ペインから項目を、視覚化ペインの「軸」「値」等へドラッグ

3.（必要なら）「値」の集計方法を「カウント」等に変更

作成するグラフ：

▶ **時間帯別アクセス：** 折れ線グラフ（X軸: hour, Y軸: timestampのカウント）

▶ **曜日別アクセス：** 集合縦棒グラフ（X軸: dayname, Y軸: timestampのカウント）

→ X軸の並び順を「dayofweek」基準で並べ替え

・アクセス元マップ： マップ（場所: ip, バブルサイズ: timestampのカウント）

→ データビューでip列のデータカテゴリを「IPアドレス」に設定

レポートの作成

実習２ 業務活用 Part４

タスク4：レポート（ダッシュボード）を構築

作成した３つのグラフをレポートキャンバス上に配置し
レイアウトを整えます

インタラクションの確認

実習２ 業務活用 Part４

グラフ同士の連携を確認

Power BI では、デフォルトで同じページ上のグラフが連動します

試してみよう：

- **曜日グラフの「Monday」をクリック**
  → 他のグラフが月曜データに絞り込まれる
- **地図上の大きな円をクリック**
  → 他のグラフがその地域のデータに絞り込まれる

Tableauとの比較
Tableauでは明示的に「フィルターとして使用」を設定しましたが
Power BIはデフォルトで連動するのが特徴です

ファイルの保存

実習２ 業務活用 Part４

タスク5：レポートを保存する

作成したレポートは .pbix ファイルとしてローカルに保存します

手順：

1. 「ファイル」→「名前を付けて保存」
2. ファイル名（例: AccessLogReport.pbix）を入力して「保存」

成果物
この `.pbix` ファイルが Part４の成果物となります

実習２のまとめ

実習２ 業務活用

BIツールの代表格2つのBIツールを体験しました
Part４では Power BI Desktop の基本操作を学び、
実習２全体を通して以下のスキルを習得しました

- ✅ **ETLプロセス（ログ解析→データマート作成）の実装**
- ✅ **BIツール（Tableau & Power BI）でのデータ接続・可視化**
- ✅ **インタラクティブなダッシュボード（レポート）の構築**

NEXT

実習３：Sparkによる大規模データ処理

ビッグデータ技術応用

実習３では
実習１,２では扱いきれなかった
PCのメモリを超えるような「巨大データ」
を扱う技術に挑戦します

NEXT

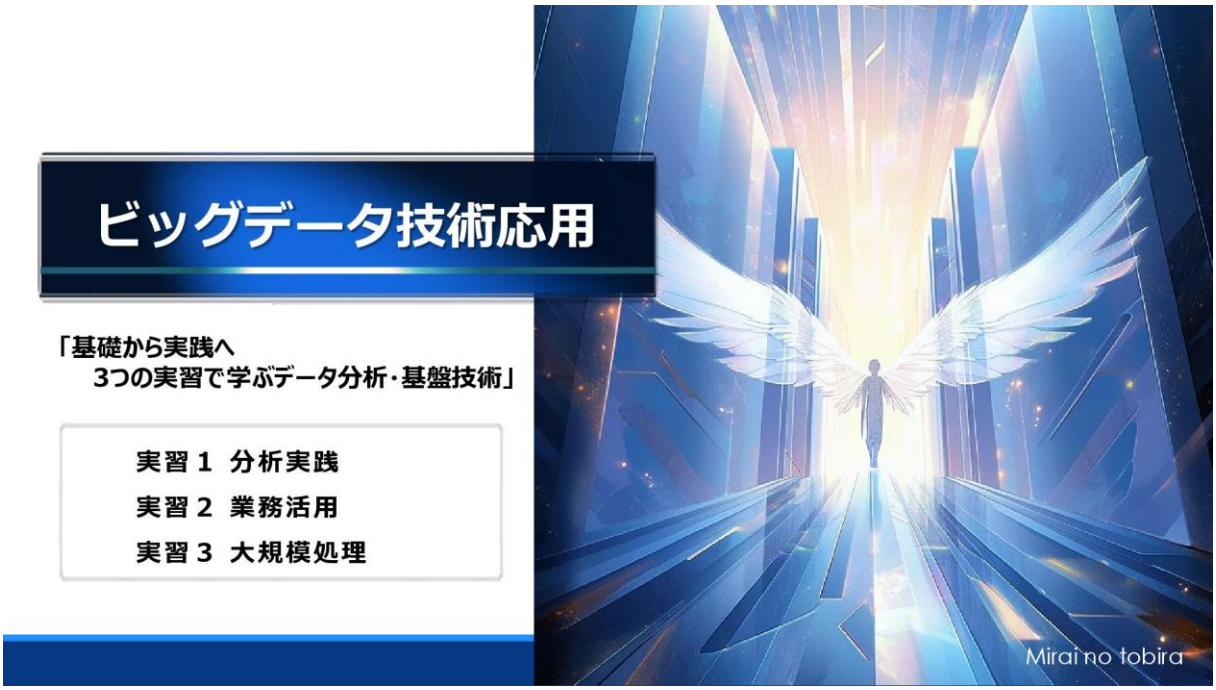
ビッグデータ技術応用

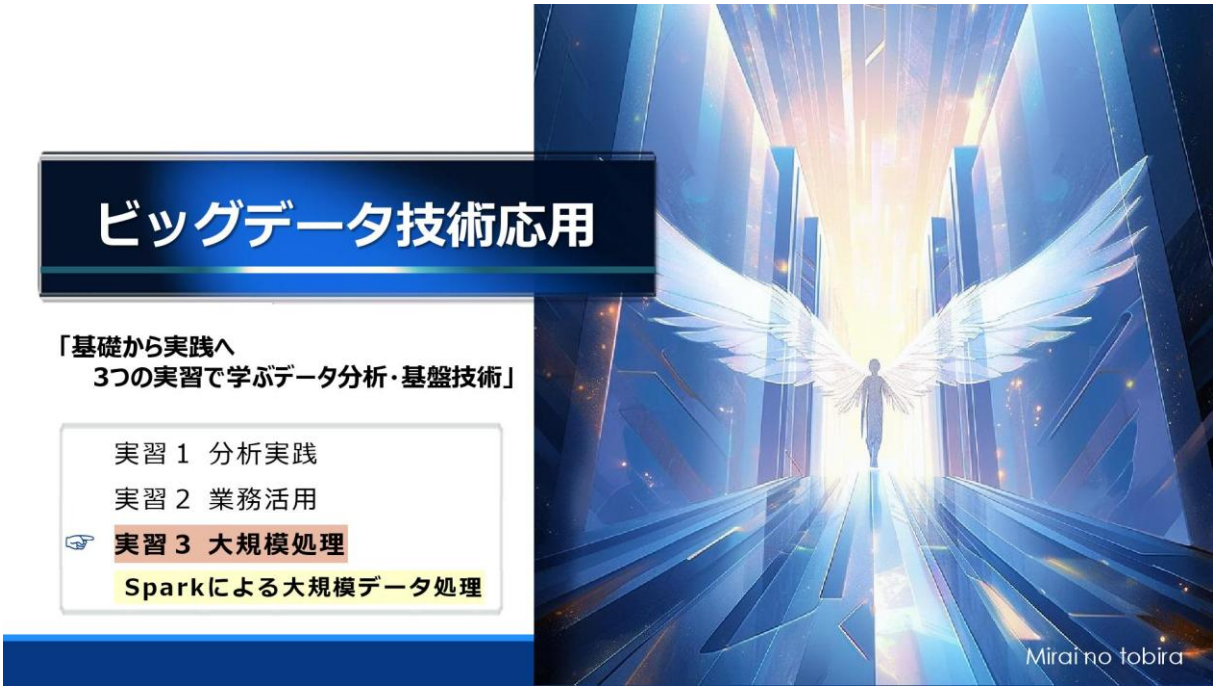
「基礎から実践へ
3つの実習で学ぶデータ分析･基盤技術」

実習１ 分析実践
実習２ 業務活用
☞ **実習３ 大規模処理**
Sparkによる大規模データ処理

ビッグデータ技術応用_実習 3

導入編


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira

実習構成

ビッグデータ技術応用

実習１ 分析実践
Python による
顧客データ分析と
レポーティング

実習２ 業務活用
アクセスログETL処理と
BIツールによる可視化

実習３ 大規模処理
Spark による
大規模データ処理

ビッグデータ技術基礎		ビッグデータ技術応用		
第1章 ビッグデータとその活用	→	実習１		
第2章 データ分析基礎	→	実習１		
第3章 スクリプト言語による分析	→	実習１	実習２	
第4章 ツールとモニタリング	→		実習２	
第5章 列指向ストレージによる高速化	→			**実習３**
第6章 データマートの基本構造	→		実習２	
第7章 大規模分散処理	→			**実習３**

実習の背景

ビッグデータ技術応用

pandas ：大きな弱点 → データをすべてPCのメモリに読み込んで処理する

→ メモリ不足でエラー、処理が異常に遅くなる

数GBを超えるデータ → pandasだけで扱うのは、現実的に非常に困難

Apache Spark： 分散処理のためのフレームワーク

→ たくさんのコンピュータ（ノード）で手分けして同時に処理

複数のマシン（or1台のマシンの複数のCPUコア）を使って高速に処理可能

実習３ 大規模処理

ビッグデータ技術応用

実習１ 分析実践 → 実習２ 業務活用 → **実習３ 大規模処理** Spark による 大規模データ処理

Part１：分散処理の必要性と Spark 環境
Part２：Spark DataFrame の基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）

実習ステップ

ビッグデータ技術応用

〔実習３の４つの Part〕

Part１: pandas の限界を体感し、Spark 環境（Google Colaboratory）を準備
Part２: Spark を使って大規模な CSV や Parquetファイルを読み込み、DataFrameの基本的な操作を学ぶ
Part３: 同じ集計処理を pandas と Spark で実行し、処理時間を比較する
Part４: Spark でより実践的な集計を行い、結果を pandas DataFrame に変換して可視化する

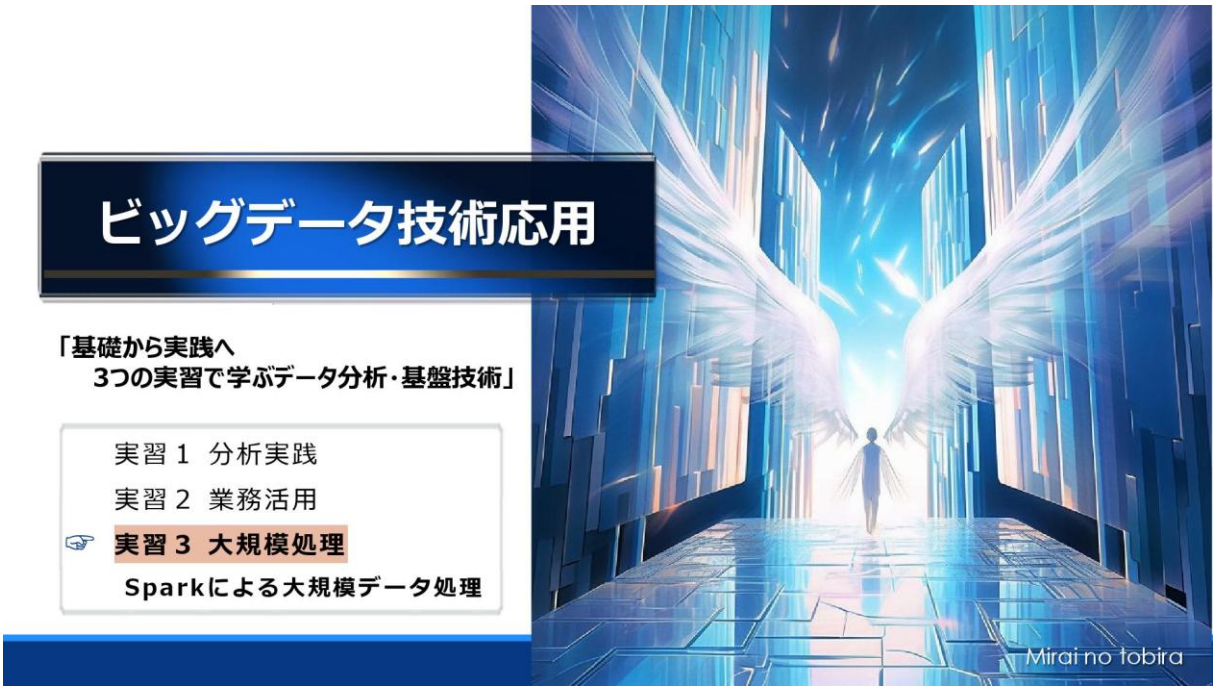
ゴール：Spark の基本的な使い方をマスターし、その効果を実感すること
実行環境：実習１,２と同様 Google Colaboratory を推奨
データ：Part１のワークブック内で生成する数GB規模の擬似センサーデータ

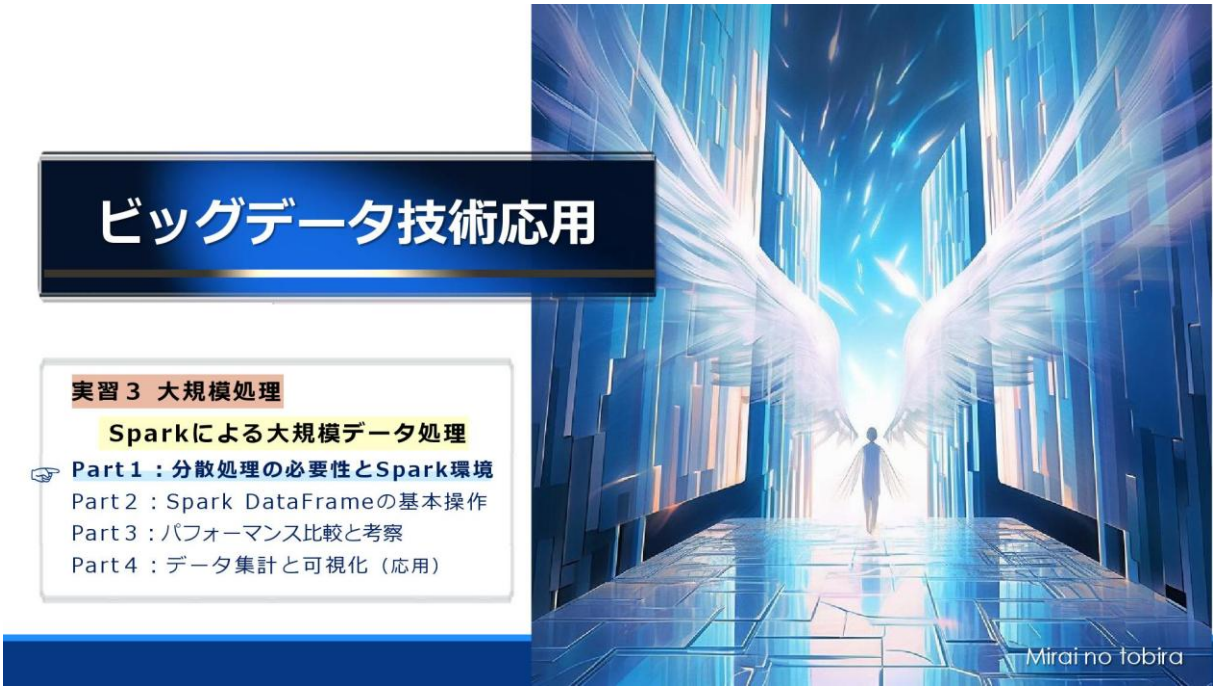

ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


NEXT
ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira

ビッグデータ技術応用_実習３

Part1


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira


ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

Part１：分散処理の必要性とSpark環境

実習３ 大規模処理

なぜ Spark が必要？ Pandas の限界を知る

Part１:分散処理の必要性とSpark環境 の達成目標

実習３ 大規模処理 Part１

- ✓ Pandas が大規模データ処理で限界に達する理由を体感する
- ✓ 分散処理の基本的な考え方を理解する
- ✓ Apache Spark の役割を知る
- ✓ Google Colab上で PySpark 環境をセットアップできる
- ✓ Spark アプリケーションの入口 (SparkSession) を作成できる
- ✓ 分析用の大規模データを生成できる

Pandas の限界

実習 3 大規模処理 Part 1

タスク1：Pandas の限界を体験

Pandas は便利ですが、データを「全量メモリに読み込む」のが基本です

もし データサイズ > PCのメモリ容量 なら？

`MemoryError!` または、処理が異常に遅くなります（スワップ発生）

解決策：分散処理

実習 3 大規模処理 Part 1

解決策：データを『分けて』処理する

メモリに乗り切らないなら、データを小さく分割し、
複数のCPUコアやマシンで手分けして処理すれば良い！

→ これが**分散処理**の考え方です

メリット
- メモリ制限を超えるデータを扱える
- 並列処理により高速化が期待できる

Apache Spark とは？

実習３ 大規模処理 Part１

分散処理の実行エンジン：Apache Spark

- 大規模データ向けの高速な**分散処理フレームワーク**
- メモリ上での処理が得意で、Hadoop MapReduceより高速
- Python (PySpark), Scala, Java, R など多様な言語に対応
- DataFrame API により、pandasに似た感覚で分散データを操作可能

Spark 環境セットアップ (Colab)

実習３ 大規模処理 Part１

タスク3：ColabでSparkを使う準備

Google Colab上で PySpark を使うための準備は簡単です

ステップ1：インストール

```
!pip install pyspark -q
```

ステップ2：SparkSession作成

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("MyApp").getOrCreate()
```

SparkSession
これがSparkを使うための「入口」 ノートブックの最初に必ず実行します

SparkSession の確認

実習３ 大規模処理 Part１

Sparkの準備完了

SparkSessionが作成されると、spark という変数が使えるようになります

```
spark # 変数の中身を表示
```

出力例：

```
<pyspark.sql.session.SparkSession object at 0x...>
SparkSession - in-memory
...
Spark UI
...
```

このような情報が表示されれば、Sparkを使う準備はOKです

大規模データの生成

実習３ 大規模処理 Part１

タスク4：『メモリ不足』を起こすデータを作る

Sparkの効果を実感するために、
pandasでは扱いにくいサイズのデータ（数GB）を生成します

```
def generate_sensor_data(num_rows, file_path):
# ... (データ生成ロジック) ...
df.to_csv(file_path, index=False)

# 例: 5000万行 (約2GB) のデータを生成
NUM_ROWS = 50_000_000
CSV_PATH = 'large_sensor_data.csv'
generate_sensor_data(NUM_ROWS, CSV_PATH)
```

⚠ 注意
- データ生成には時間がかかります（数分）
- Colabのディスク容量に注意してください

生成データの確認

実習３ 大規模処理 Part１

巨大ファイルの誕生

データ生成後、ファイルサイズを確認します

```
!ls -lh large_sensor_data.csv
```

出力例：

```
-rw-r--r-- 1 root root 2.1G Oct 20 19:30 large_sensor_data.csv
```

→ 数GBのCSVファイルができました。これを pandasで読み込むのは大変そうです

実習３ Part１のまとめ

実習３ 大規模処理 Part１

大規模データ処理への第一歩を踏み出しました

- ✅ Pandasのメモリ限界を体験した
- ✅ 分散処理と Spark の必要性を理解した
- ✅ Colab上 に PySpark 環境を準備できた
- ✅ 分析対象となる大規模CSVデータを生成した

Part１：分散処理の必要性とSpark環境　ビッグデータ技術応用 実習３

NEXT

Part２：Spark DataFrame の基本操作

実習３ 大規模処理

実習３ Part２では、
Spark を使って巨大CSVファイルを読み込み
DataFrame として操作する方法を学びます

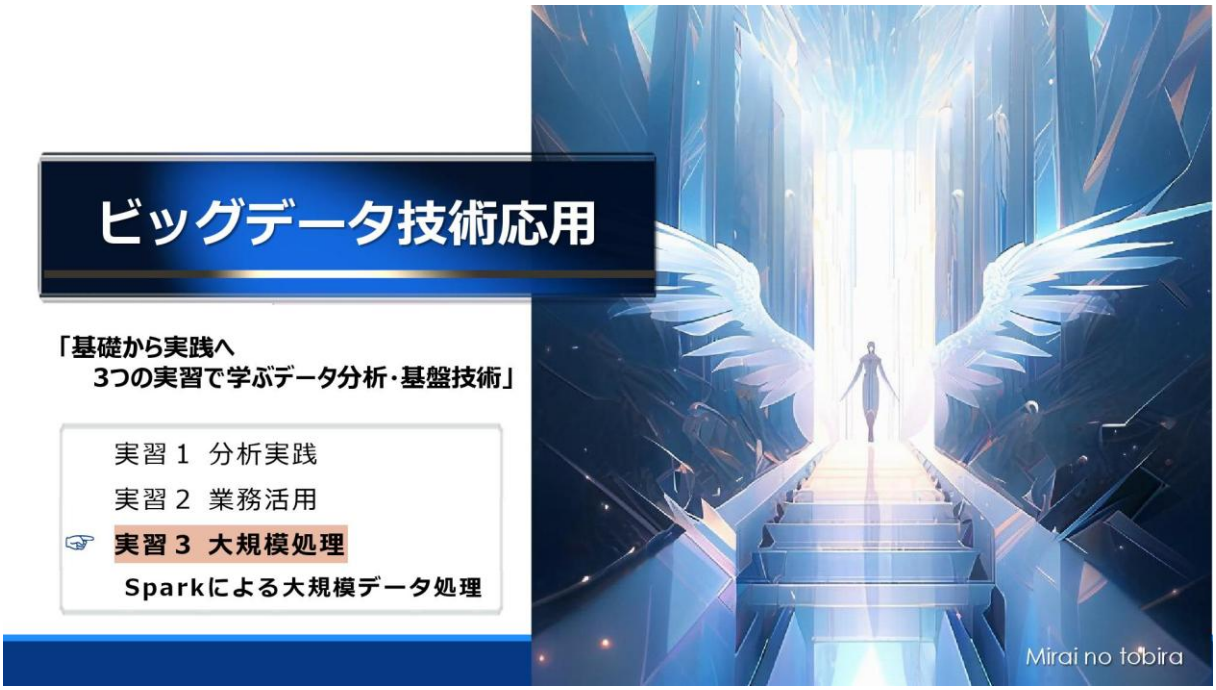
NEXT

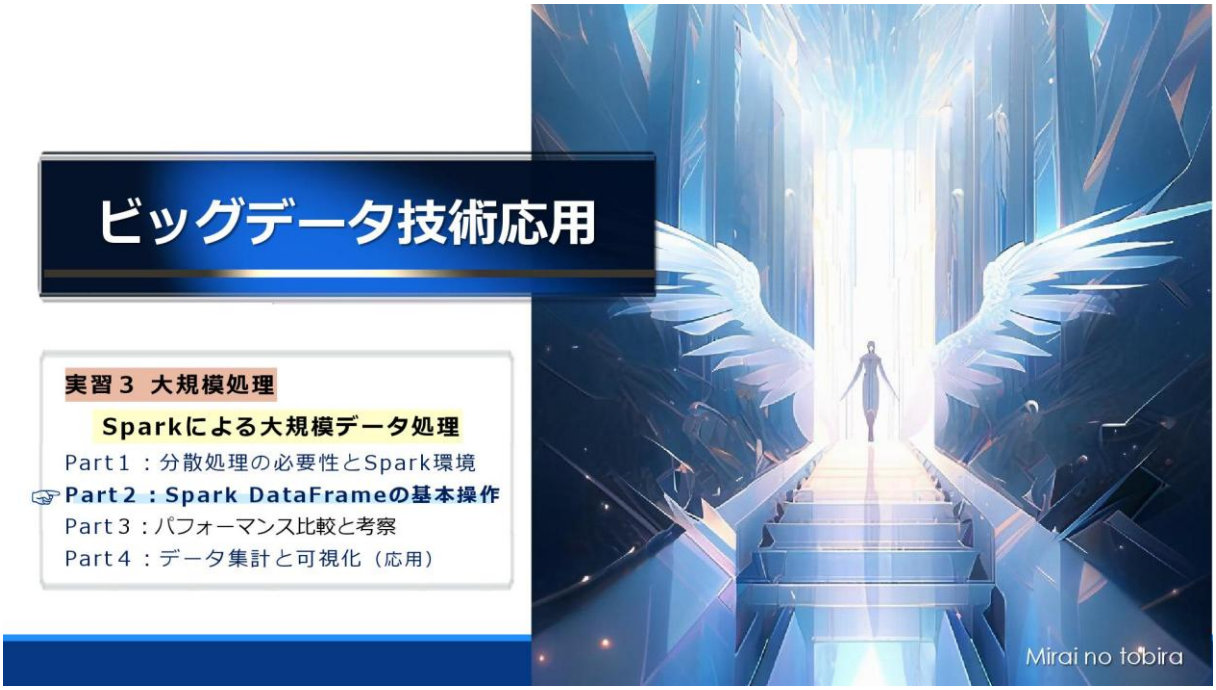
ビッグデータ技術応用

実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）

ビッグデータ技術応用_実習３

Part2


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira


ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

Part 2 ：Spark DataFrame の基本操作

実習 3 大規模処理

巨大データを自在に操る第一歩

Part 2 : Spark DataFrame の基本操作 の達成目標

実習 3 大規模処理 Part 2

- ✓ Spark で巨大CSVファイルを DataFrame として読み込む
- ✓ DataFrame のスキーマ（データ型）を確認する
- ✓ 基本的なAction (show, count) と Transformation (select, filter, groupBy) を使う
- ✓ Spark の「遅延評価」とは何かを理解する
- ✓ データを効率的な Parquet 形式で保存・読み込みする

巨大CSVの読み込み

実習３ 大規模処理 Part２

タスク2：Spark で巨大CSVを読み込む

spark.read.csv() を使います

```
sensor_df = spark.read.csv(
'large_sensor_data.csv',
header=True, # ヘッダーあり
inferSchema=True # スキーマ自動推論
)
sensor_df.show(5) # Action実行 （ここで初めて読み込みが走る）
```

⚠ 注意点
- `inferSchema=True` は便利だが、巨大ファイルでは推論に時間がかかる
- `.show()` のようなActionを実行するまで、実際の読み込みは開始されない（**遅延評価**）
- 読み込み（特に初回）には時間がかかる（数分）

スキーマ確認 & カウント

実習３ 大規模処理 Part２

データの中身を確認：スキーマと行数

スキーマ確認 (.printSchema()):

```
sensor_df.printSchema()
```

```
root
|-- sensor_id: string (nullable = true)
|-- timestamp: string (nullable = true)
|-- temperature: double (nullable = true)
|-- humidity: double (nullable = true)
|-- pressure: double (nullable = true)
```

行数カウント (.count() - Action):

```
total_rows = sensor_df.count() # 全データをスキャンするため時間がかかる
```

🎯 ポイント
- `timestamp` が `string` になっている → 自動推論の限界。後で修正が必要
- `.count()` はActionなので、実行するとSparkクラスタ全体で計算が走る

Transformation vs Action

実習３ 大規模処理 Part２

Sparkの操作：Transformation と Action

Transformation (変換)

- 新しいDataFrameを作る操作
- 例: select, filter, groupBy
- 指示しただけでは**実行されない**
- 「処理計画」を組み立てるイメージ

➡ (Action実行)

Action (実行)

- 結果を返す、または書き出す操作
- 例: show, count, collect, save
- これを呼び出すと初めて計算が走る
- 「実行！」の号令イメージ

遅延評価 (Lazy Evaluation)

Actionが呼ばれるまでTransformationを実行しない仕組み。これによりSparkは処理全体を最適化できます

基本操作① Select & Filter

実習３ 大規模処理 Part２

タスク3：列選択と行フィルタリング

Pandasと似た感覚で操作できます。

列選択 (.select() - Transformation):

```
selected_df = sensor_df.select("sensor_id", "temperature")
```

行フィルタリング (.filter() / .where() - Transformation):

```
# temperature が 30より大きい行を抽出
high_temp_df = sensor_df.filter(sensor_df.temperature > 30)
# または SQLライクな書き方
high_temp_df = sensor_df.where("temperature > 30")
```

結果表示 (.show() - Action):

```
high_temp_df.show(5) # ここで初めて filter が実行される
```

基本操作② GroupBy & Agg

実習３ 大規模処理 Part２

タスク3：グループ集計

groupBy() でグループ化し、agg() で集計します

```
from pyspark.sql.functions import count, avg # 集計関数をインポート
# sensor_id ごとにグループ化 (Transformation)
grouped_df = sensor_df.groupBy("sensor_id")
# 集計 (Transformation)
agg_df = grouped_df.agg(
count("*").alias("record_count"),
avg("temperature").alias("avg_temp")
)
# 結果表示 (Action)
agg_df.show(10)
```

ポイント
- 集計関数 (`count`, `avg` など) は `pyspark.sql.functions` からインポートして使う
- `.alias()` で結果の列名を指定できる

Parquet 形式の利点

実習３ 大規模処理 Part２

タスク4：なぜParquetが良いのか？

大規模データを扱うなら CSV より Parquet が断然有利

CSV
- 行指向
- 圧縮効率 低
- 全列読み込み
- スキーマ情報 なし

Parquet
- **列指向**
- **圧縮効率 高**
- **必要列のみ読込**
- **スキーマ情報 あり**

結果
ファイルサイズが小さくなり、読み込み速度が劇的に向上
データ型推論 (inferSchema) も不要に

Part２：Spark DataFrame の基本操作　ビッグデータ技術応用 実習３

Parquet への変換・保存

実習３ 大規模処理 Part２

CSV → Parquet へ変換・保存

Spark DataFrame を Parquet 形式で保存するのは簡単です

```
parquet_output_path = 'sensor_data.parquet'

# .write.parquet() を使う (Action)
sensor_df.write.mode("overwrite").parquet(parquet_output_path)
```

⚠ 書き込みには時間がかかります

Parquet の読み込み

実習３ 大規模処理 Part２

Parquet を高速読み込み

読み込みは spark.read.parquet() を使います

```
# Parquetディレクトリを指定して読み込む
sensor_df_parquet =
spark.read.parquet(parquet_output_path)

# スキーマ確認 (inferSchema不要！)
sensor_df_parquet.printSchema()

# 表示 (Action) - CSVより速いはず！
sensor_df_parquet.show(5)
```

🎯 比較ポイント
CSV読み込み(スライド3)の `.show(5)` と比べて、実行時間がどれくらい短縮されたか確認しましょう

実習３ Part２のまとめ

実習３ 大規模処理 Part２

Spark DataFrame の基本的な操作をマスターしました

- ✅ **Spark での CSV/Parquet 読み込み**
- ✅ **スキーマ確認、行数カウント**
- ✅ **Transformation** (select, filter, groupBy, agg)
- ✅ **Action** (show, count, write)
- ✅ **遅延評価の概念**
- ✅ **Parquet 形式の利点と使い方**

NEXT

Part３：パフォーマンス比較と考察

実習３ 大規模処理

実習３ Part３では、

同じ処理を **pandas と Spark** で実行し

処理速度を比較します

Spark の真価を体感しましょう

NEXT

ビッグデータ技術応用

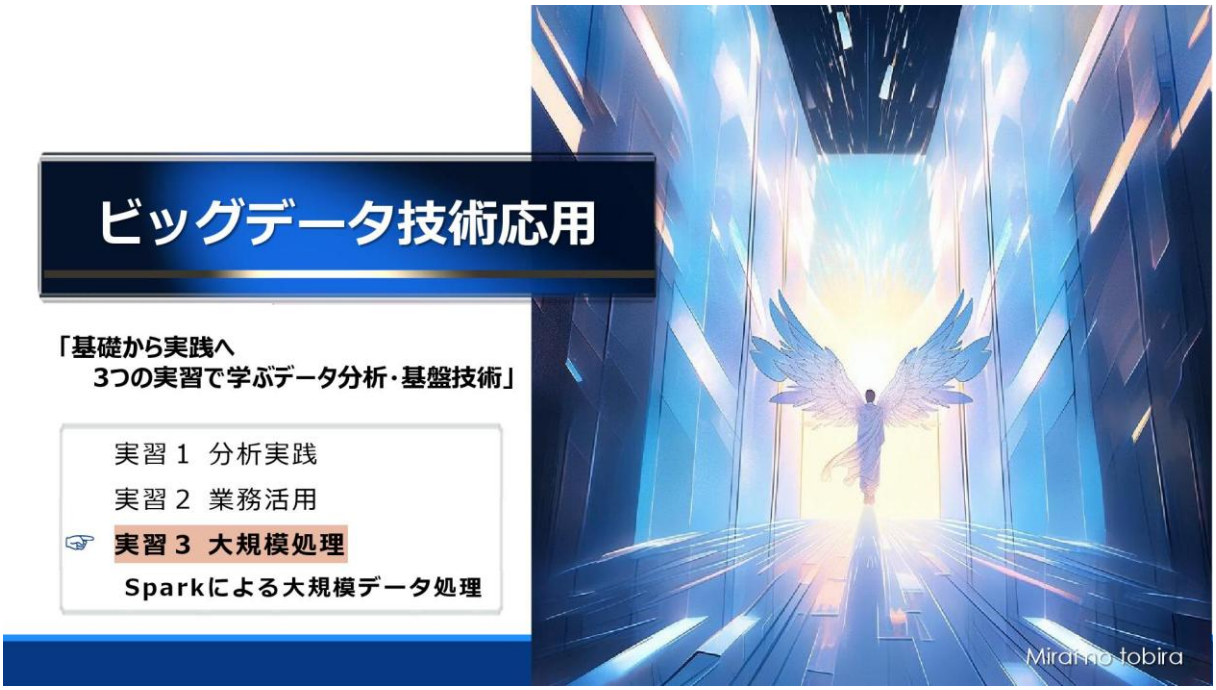
実習３ 大規模処理

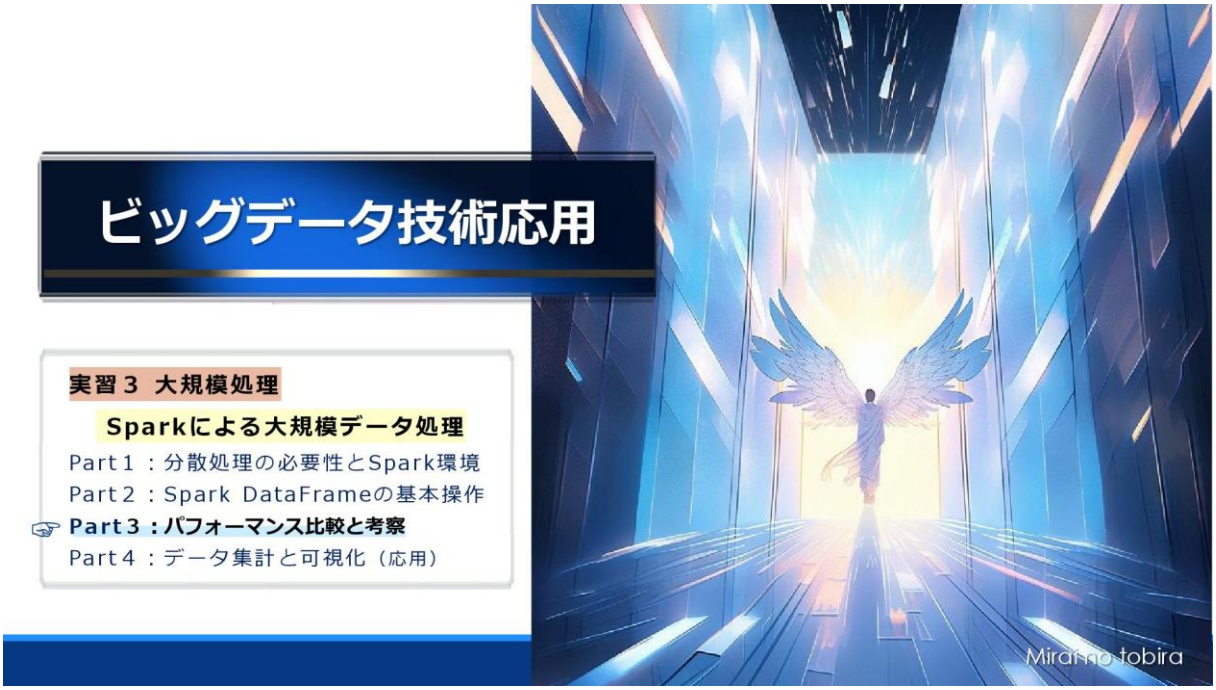
Sparkによる大規模データ処理

Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
☞ **Part３：パフォーマンス比較と考察**
Part４：データ集計と可視化（応用）

ビッグデータ技術応用_実習３

Part3


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira


ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

Part３：パフォーマンス比較と考察

実習３ 大規模処理

Pandas vs Spark：速いのはどちら？

Part３：パフォーマンス比較と考察 の達成目標

実習３ 大規模処理 Part３

- ✓ 同じ集計処理を pandas と Spark で実行し、時間を計測する
- ✓ 大規模データに対する両者のパフォーマンス（速度）の違いを比較する
- ✓ なぜ Spark が高速なのか、その理由（分散処理など）を考察する
- ✓ %%time マジックコマンドの使い方をマスターする

比較実験の準備

実習３ 大規模処理 Part３

タスク1：比較実験

比較対象：
- **Pandas:** シングルマシン、メモリ処理
- **Spark:** 分散処理エンジン

比較する処理：
- センサーIDごとの統計量（レコード数、温度の平均/最大/最小）の算出

使用データ：
- Part２で作成した `sensor_data.parquet`（数GB）

計測方法：
- `%%time` マジックコマンド

Pandasでの挑戦

実習３ 大規模処理 Part３

タスク2：Pandasで挑戦

ステップ1：データ読み込み (Parquet)

```
%%time
try:
  pandas_df = pd.read_parquet('sensor_data.parquet',
engine='pyarrow')
except MemoryError:
  print("💥 メモリ不足！")
  pandas_df = None
```

ステップ2：集計処理 (もし読み込めたら)

```
%%time
if pandas_df is not None:
  pandas_agg = pandas_df.groupby('sensor_id').agg(...)
  print(pandas_agg.head())
```

Sparkでの挑戦

実習３ 大規模処理 Part３

タスク3：Sparkで挑戦

ステップ1：データ読み込み定義 (これは一瞬)

```
sensor_df_spark =
spark.read.parquet('sensor_data.parquet')
```

ステップ2：集計処理 (Action実行時に計算開始)

```
%%time
from pyspark.sql.functions import count, avg, max, min
spark_agg = sensor_df_spark.groupBy("sensor_id").agg(
  count("*").alias("count"),
  avg("temperature").alias("avg_temp"),
  # ... max, minも同様 ...
)
spark_agg.show(5) # Action!
```


結果比較

実習３ 大規模処理 Part３

タスク4：結果比較

計測結果を比較してみましょう

処理	Pandas 時間	Spark 時間
センサー別集計	[エラー or XX 秒]	[YY 秒]

なぜ Spark は速い？①

実習３ 大規模処理 Part３

考察：Spark高速化の秘密① 分散処理

Spark が速い最大の理由は**「分散処理」**

- ▶ Spark はデータを複数のパーティションに分割
- ▶ それらを複数の Executor（CPUコアやマシン）で**並列に処理**
- ▶ Pandas は基本的に1コアで順番に処理

→ データが大きければ大きいほど並列処理の効果でSparkが有利

なぜ Spark は速い？②

実習３ 大規模処理 Part３

考察：Spark高速化の秘密② メモリ管理＆遅延評価

メモリ管理：

- ▶ Pandasは全データをメモリに乗せようとする
  → メモリ不足のリスク
- ▶ Sparkは必要なデータだけをExecutorのメモリに乗せて処理
  → メモリ効率が良い

遅延評価：

- ▶ Actionが呼ばれるまで処理を遅らせる
  → Sparkが処理全体を最適化（例：不要なデータの読み込みを省略）可能

実習３ Part３のまとめ

実習３ 大規模処理 Part３

Pandas と Spark の比較実験をして、Spark の威力を体感しました

- ✅ **大規模データでは Pandas は限界があり、Spark は処理可能**
- ✅ **Spark は分散処理により Pandas より高速に処理できる**
- ✅ **その理由は、並列処理、効率的なメモリ管理、遅延評価にある**
- ✅ **%%time で処理時間を計測・比較する方法を学んだ**

→ 結論として、ビッグデータを扱うなら Spark が必須

NEXT

実習３Part４：データ集計と可視化（応用）

実習３ 大規模処理

実習３ Part４では、

- ✓ もう少し複雑な集計処理に挑戦
- ✓ Spark の集計結果を **pandas DataFrame** に変換
- ✓ Matplotlib/Seaborn で見慣れたグラフを作成

を行います


NEXT
ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

ビッグデータ技術応用_実習３

Part4


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira


ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

Part４：データ集計と可視化（応用）

実習３ 大規模処理

Spark で処理し、Pandas で描く：実践的ワークフロー

Part４：データ集計と可視化（応用）の達成目標

実習３ 大規模処理 Part４

- ✓ Spark で応用的な集計（日時変換、ウィンドウ関数、SQL）を行う
- ✓ Spark DataFrame を Pandas DataFrame に変換する (.toPandas())
- ✓ .toPandas() の注意点（メモリ）を理解する
- ✓ 変換後の Pandas DataFrame を使い慣れたツールで可視化する
- ✓ Spark と Pandas/可視化ライブラリを連携させる流れを習得する

応用的な集計① 日時処理

実習３ 大規模処理 Part４

タスク2：応用的な集計① 日時処理

Part２で string のままであった timestamp 列を日時型に変換し
時間帯の平均値を計算します

ステップ1：日時型へ変換

```
from pyspark.sql import functions as F
df_with_ts = sensor_df.withColumn(
  "ts", F.to_timestamp(F.col("timestamp"), 'yyyy-MM-dd HH:mm:ss')
)
```

ステップ2：Hour抽出 & 集計

```
df_with_hour = df_with_ts.withColumn("hour", F.hour(F.col("ts")))
hourly_avg = df_with_hour.groupBy("hour").agg(
  F.avg("temperature").alias("avg_temp"), ...
).orderBy("hour")
hourly_avg.show()
```


応用的な集計② ウィンドウ関数

実習３ 大規模処理 Part４

タスク2：応用的な集計② ウィンドウ関数

センサーごとに時間順に並べ、過去N件の移動平均を計算する例

```
from pyspark.sql.window import Window

# ウィンドウ定義: sensor_idで分割し、tsで並び替え
windowSpec = Window.partitionBy("sensor_id").orderBy("ts")

# 移動平均を計算（現在行と過去2行）
df_with_ma = df_with_ts.withColumn(
  "temp_ma_3", F.avg("temperature").over(windowSpec.rowsBetween(-2, 0))
)
df_with_ma.select("sensor_id", "ts", "temperature",
"temp_ma_3").show(10)
```

応用的な集計③ Spark SQL

実習３ 大規模処理 Part４

タスク2：応用的な集計③ Spark SQL

DataFrame API の代わりに、SQLクエリで集計することも可能

ステップ1：テーブル登録

```
sensor_df.createOrReplaceTempView("sensor_data")
```

ステップ2：SQL実行

```
sql_result = spark.sql("""
  SELECT sensor_id, COUNT(*) as count, AVG(temperature) as avg_temp
  FROM sensor_data
  GROUP BY sensor_id
""")
sql_result.show(10)
```


Spark から Pandas へ

実習３ 大規模処理 Part４

タスク3：Spark → Pandas へデータを受け渡す

Sparkで集計した「結果」（通常はメモリに乗るサイズ）を
使い慣れた pandas DataFrame に変換します

```
# 例：時間帯別平均データをPandasに変換
hourly_avg_pd_df = hourly_avg_df.toPandas() # Action!

# 変換後の確認
print(type(hourly_avg_pd_df))
hourly_avg_pd_df.info()
```

⚠ 重要注意点
`.toPandas()` は、**分散していたデータをすべてDriverのメモリに集めます**
変換対象のデータが巨大な場合、ここで **MemoryError** が発生します
必ず、**集計後**の十分に小さくなったDataFrameに対してのみ使用してください

Pandas で可視化

実習３ 大規模処理 Part４

タスク3：Pandas DataFrameでいつも通り可視化

Pandas DataFrame を使えば、その後は実習１,２の実習と同様

```
import matplotlib.pyplot as plt
import seaborn as sns

# 例：時間帯別 平均温度の折れ線グラフ
plt.figure(figsize=(12, 6))
sns.lineplot(data=hourly_avg_pd_df, x='hour',
y='avg_temperature', marker='o')
plt.title('Average Temperature by Hour')
plt.show()
```


実践的なワークフロー

実習３ 大規模処理 Part４

Spark + Pandas/可視化ライブラリ連携

実務でもよく使われる Spark + Pandas 連携ワークフローワークフローです

1. 読み込み
Sparkで大規模データ(Parquet等)をロード

↓

2. 分散処理
Sparkで重い集計や加工を高速実行

↓

3. 軽量化
結果がメモリに乗るサイズに凝縮

↓

4. 変換
.toPandas()で Pandas DFへ

↓

5. 可視化
Matplotlib等でグラフ化・分析

ポイント
- **重い処理・集計：** Sparkの分散処理能力を活用
- **最終的な可視化・レポート：** 使い慣れたPandas + Matplotlib/Seaborn を活用
- **橋渡し役：** `.toPandas()` （ただしメモリに注意）

Part 4 ／ 実習3 まとめ

実習 3 大規模処理

Part 4 では応用的な集計と可視化連携を学び、
実習 3 全体を通して以下のスキルを習得しました

- ✅ Spark の必要性と基本操作
- ✅ Pandas vs Spark のパフォーマンス比較体験
- ✅ Parquet 形式の利点
- ✅ Spark での集計結果を Pandas で可視化する連携

ビッグデータ技術応用 まとめ

ビッグデータ技術応用

実習 1 , 2 , 3 を通して
データ分析～ETL～BI～大規模処理まで
データ活用の重要スキルを幅広く学びました

今回学んだ Spark は入口、ビッグデータの世界は広大
さらに深く次のような項目を学んでいくことが望まれます

- Spark Streaming (リアルタイム処理)
- MLlib (Sparkの機械学習ライブラリ)
- Hadoopエコシステム (HDFS, Hive, etc.)
- クラウドプラットフォーム (AWS, GCP, Azure) での Spark
- データパイプライン構築 (Airflowなど)


ビッグデータ技術応用
Mirai no tobira

ビッグデータ技術応用_実習まとめ編

ビッグデータ技術応用

「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」

実習１ 分析実践

Python による顧客データ分析とレポーティング

実習２ 業務活用

アクセスログETL処理とBIツールによる可視化

実習３ 大規模処理

Sparkによる大規模データ処理

ビッグデータ技術応用

実習３ 大規模処理

Sparkによる大規模データ処理

Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
☞ **Part４：データ集計と可視化（応用）**

実習構成

ビッグデータ技術応用

実習１ 分析実践

Python による
顧客データ分析と
レポーティング

Pythonでの
データ分析プロセス
Clean →
Analyze →
Visualize →
Report

実習２ 業務活用

アクセスログETL処理と
BIツールによる可視化

ETL処理 & BIツールでの
ダッシュボード作成
Python →
Tableau/Power BI

実習３ 大規模処理

Spark による
大規模データ処理

Sparkによる
分散処理の基礎
Pandas限界 →
Spark

ビッグデータ技術応用 習得済スキル

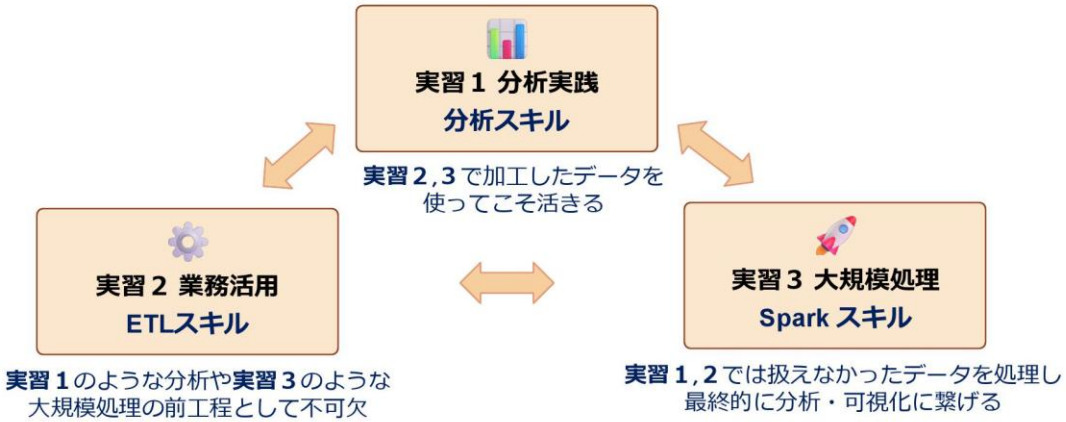
ビッグデータ技術応用

実習１,２,３を通してデータ分析～ETL～BI～大規模処理まで
データ活用の重要スキルを幅広く学びました

- **Python データ分析スキル**
  Pandas, Matplotlib, Seaborn を使った実践的な分析力
- **ETL基礎スキル**
  生データ (テキスト含む) を分析可能な形に加工する力 (正規表現含む)
- **BIツール活用スキル**
  Tableau, Power BI を使ってデータを分かりやすく見える化する力
- **大規模データ処理入門スキル**
  Sparkの基本を理解し、メモリを超えるデータを扱う考え方
- **問題解決 & 考察力**
  データから課題を発見し、解決策を考え、結果　を伝える力

スキルの繋がり

ビッグデータ技術応用



実習１のような分析や**実習３**のような
大規模処理の前工程として不可欠

実習１,２では扱えなかったデータを処理し
最終的に分析・可視化に繋げる

学びは続く：Next Step

ビッグデータ技術応用

応用教材で得た基礎を元にさらに専門性を深めましょう

- ✓ **データ分析深化**
  機械学習、統計モデリング、特定分野（マーケティング、金融など）の分析手法
- ✓ **データエンジニアリング深化**
  データパイプライン構築 (Airflow)、クラウド (AWS, GCP, Azure)、
  データベース (SQL, NoSQL)、ストリーミング処理 (Kafka, Spark Streaming)
- ✓ **BI深化**
  高度なダッシュボードデザイン、DAX/LOD 表現、データガバナンス
- ✓ **実践経験**
  Kaggleコンペへの挑戦、インターンシップ、卒業研究でのデータ活用

「ビッグデータ技術応用」のその先

今回学んだ Spark は入口、ビッグデータの世界は広大
さらに深く次のような項目を学んでいくことが望まれます

- Spark Streaming (リアルタイム処理)
- MLlib (Sparkの機械学習ライブラリ)
- Hadoopエコシステム (HDFS, Hive, etc.)
- クラウドプラットフォーム (AWS, GCP, Azure) での Spark
- データパイプライン構築 (Airflowなど)

ビッグデータ技術応用

「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」

実習 1 分析実践
実習 2 業務活用
実習 3 大規模処理


ビッグデータ技術応用
Mirai no tobira

令和7年度「地方やデジタル分野における専修学校理系転換等推進事業」

情報成長分野の教育プログラム整備と教員育成による学科転換・新設推進事業

コンテナ技術リカレント教育プログラム

令和８年２月

一般社団法人全国専門学校情報教育協会

〒164-0003 東京都中野区東中野 1-57-8 辻沢ビル３F

電話：03-5332-5081 FAX.03-5332-5083