卒業論文

機械学習によるホタルイカ身投げ量予測およびWeb情報公開システムの開発

Machine learning prediction of firefly squid suicide numbers and development of a web information disclosure system

富山県立大学 工学部 情報システム工学科
2120006 海野幸也
指導教員 António Oliveira Nzinga René 准教授
提出年月: 令和8年(2026年)2月

目 次

図一覧

図一覧

表一覧

表一覧

記号一覧

以下に本論文において用いられる用語と記号の対応表を示す.

用語	記号
GBDT における時点 t での予測値	$\hat{y}_i^{(t)}$
GBDT における時点 t で追加される新たな決定木	$f_t(x_i)$
GBDT における目的関数	$\mathrm{Obj}^{(t)}$
GBDT における損失関数	L
GBDT における正則化項	Ω
GBDT における損失関数の 1 次微分（勾配）	g_i
GBDT における損失関数の 2 次微分（ヘシアン）	h_i
GBDT における正則化パラメータ	λ
GBDT における葉の数に対するペナルティ	γ
周期的エンコーディングにおける周期を持つ変数	t
周期的エンコーディングにおける周期	T
周期的エンコーディングにおける sin 変換後の値	$x_{\sin}$
周期的エンコーディングにおける cos 変換後の値	$x_{\cos}$
評価指標におけるデータ数	n
評価指標における i 番目の実測値	y_i
評価指標における i 番目の予測値	$\hat{y}_i$
評価指標における実測値の平均	$\bar{y}$
評価指標における決定係数	R^2
特徴量 x_j の Split ベースの重要度	$I_{split}(x_j)$
ノード分割における利得（Gain）	G
学習された決定木の総数	T
k 番目の木の非葉ノード集合	$\mathcal{N}_k$
予測モデルの出力する予測スコア	y
予測モデルの入力となる説明変数（特徴量）	X

第1章

はじめに

§ 1.1　本研究の背景

現代社会は，Internet of Things（IoT），ビッグデータ，人工知能（AI）といった第 4 次産業革命の中核をなす技術群の急速な進展により，劇的な変化を遂げている．これらの技術は，現実空間のあらゆる事象をデータ化し，サイバー空間で分析・解析することで新たな価値を生み出す「Society 5.0」の実現に向けた原動力となっている．特に，膨大なデータから人間には認識困難な法則性を見出し，未来を予測する機械学習技術の応用は，製造業における予知保全から金融市場の変動予測に至るまで，多岐にわたる分野で意思決定の高度化に寄与している．こうした中，我が国の重要施策である地方創生や観光立国の実現においても，デジタル技術の活用による産業構造の変革，いわゆる「観光 DX（デジタルトランスフォーメーション）」が喫緊の課題となっている．従来の観光産業は，長年の経験や勘といった属人的な知見に依存する側面が強く，データに基づいた客観的なマーケティングやサービスの提供が遅れている現状がある．不確実性の高い自然現象を観光資源とする場合，その傾向はさらに顕著となり，機会損失や顧客満足度の低下を招く要因となっている [1]．

富山県においては，春季に富山湾沿岸でのみ観測されるホタルイカの「身投げ」という現象が，地域固有の極めて重要な観光資源として知られている．産卵のために深海から海岸線へ押し寄せたホタルイカが，青白い光を放ちながら波打ち際を埋め尽くす光景は幻想的であり，シーズン中には県内外から多くの愛好家や写真家が海岸へ足を運ぶ．しかしながら，この身投げ現象は，新月の前後であることや，特定の風向・風速，波高，潮位，海水温といった多様かつ複雑な自然条件が合致した際にのみ発生する稀有な現象であり，その発生メカニズムには未解明な部分も多い．現状，身投げの発生予測に関する情報は，公的な予報システムが存在せず，主にインターネット上の掲示板や SNS を通じた個人間の情報交換に依存している．しかし，これらの情報は個人の主観に基づく定性的なものが大半であり，情報の断片化や信憑性の欠如，リアルタイム性の不足といった課題を抱えている．特に遠方からの来訪者にとっては，確度の低い情報をもとに深夜の海岸へ移動せざるを得ず，結果として身投げに遭遇できず徒労に終わるリスクが高い．これは観光資源としての価値を十分に活かしきれていないことを意味し，データサイエンスによる客観的な予測情報の提供が強く求められている状況にある [2]．

また，Web アプリケーション開発の領域に目を転じると，クラウドネイティブな技術の普及や，コンテナ仮想化技術の成熟により，開発・運用のパラダイムシフトが起きている．バックエンドとフロントエンドを分離し，各々を独立して開発・デプロイするマイクロサービスアーキテクチャや，サーバーレスコンピューティングの活用は，システムの柔軟性と

スケーラビリティを飛躍的に向上させた．このような現代的なソフトウェア工学の知見を取り入れ，ユーザー体験（UX）を重視したインターフェースを通じて高度な予測情報を提供することは，単なる技術的挑戦にとどまらず，地域社会における情報の非対称性を解消し，持続可能な観光モデルを構築する上でも大きな意義を持つと考えられる [3].

§ 1.2　本研究の目的

本研究の目的は，複雑な自然現象であるホタルイカの身投げを対象とし，過去の気象データ等を用いた機械学習による予測モデルの構築と，その情報を一般ユーザーが直感的に利用可能な Web アプリケーションとして社会実装することである．第一の目的は，データサイエンスのアプローチによる予測精度の検証である．具体的には，過去 10 年間にわたるホタルイカの身投げ発生実績データ（湧き量）と，それに対応する気象庁等の公的機関が提供する気象データ（気温，降水量，風向，風速），潮汐データ，月齢データを収集・統合し，包括的なデータセットを構築する．自然現象の時系列データには特有の周期性が存在するため，月齢や季節性を sin/cos 変換を用いて連続値として表現する特徴量エンジニアリングや，過去の気象条件が遅れて影響するラグ特徴量の生成といった前処理を施す．これらのデータを用いて，勾配ブースティング決定木の一種である LightGBM により回帰モデルを学習させる．従来の線形モデルでは捉えきれない非線形な関係性をどの程度学習可能か，また，未知のデータに対してどの程度の精度（RMSE，MAE 等の指標）で予測が可能かを定量的に評価し，その有効性と限界を明らかにする．

第二の目的は，構築した予測モデルを実用的な Web サービスとして具現化し，その有用性を検証することである．既存の掲示板システム等は，スマートフォンでの閲覧に最適化されておらず，情報の検索性や視認性に課題があった．本研究では，現代の Web 標準技術に準拠したシステム開発を行う．バックエンドには，静的型付け言語であり並行処理性能に優れた Go 言語と，高速な Web フレームワークである FastAPI を採用し，機械学習モデルの推論 API およびアプリケーションロジックを実装する．フロントエンドには，React ベースのフレームワークである Next.js を採用し，サーバーサイドレンダリング（SSR）やインクリメンタル静的再生成（ISR）を活用することで，高速なページ表示と SEO（検索エンジン最適化）に配慮した Single Page Application（SPA）を構築する． さらに，インフラストラクチャには Docker によるコンテナ技術と Google Cloud Platform（Cloud Run）を採用し，オートスケーリングや CI/CD（継続的インテグレーション/継続的デリバリー）パイプラインを整備することで，保守性と可用性の高い運用環境を実現する．

本研究では，単に予測アルゴリズムの研究にとどまらず，データの収集・加工からモデルの学習，API の実装，そしてエンドユーザーへのインターフェース提供に至るまでの一連のシステム開発工程をフルスクラッチで実践する．これにより，アカデミックな知見とエンジニアリング技術を融合させ，地域固有の課題解決に資するアプリケーションを構築し，その社会的有用性を示すことを最終的な到達点とする．また，本システムの運用を通じて得られるユーザーからのフィードバックやアクセスログを分析することで，今後の観光 DX におけるデータ活用の在り方についても考察を加える．

§ 1.3　本論文の概要

本論文は次のように構成される.

第 1 章 本研究の背景と目的について説明する．ホタルイカの身投げ現象が持つ社会的・観光的価値と現状の課題を整理し、機械学習と Web 技術によってそれらを解決する本研究の目的を明確にする。

第 2 章 ホタルイカの身投げ現象に関する生態学的背景と既存サービスの課題を文献に基づいて整理し、それらの課題を解決するために本研究で採用した Web アプリケーション技術の妥当性と新規性を理論的に明らかにする.

第 3 章 本研究で用いる時系列データ分析および LightGBM を中心とした機械学習手法について、数式と理論に基づいて体系的に解説する.

第 4 章 提案手法について説明する．ホタルイカ身投げ予測のためのデータセット構築、特徴量設計、予測モデルの学習手法、および Web アプリケーションとしての実装方法を詳細に説明する.

第 5 章 構築した予測モデルおよび Web アプリケーションの性能について、評価指標を用いた定量的実験結果とその妥当性を示す.

第 6 章 実験結果を踏まえた考察を行うとともに、本研究の限界点を整理し、今後の改良点や発展可能性について述べる.

第2章

関連技術と予備知識

本章では，本研究の主題であるホタルイカの身投げ現象に関する生物学的・環境的背景と，本システムが解決しようとする既存の情報収集手段の課題について述べる．さらに，本研究で開発する Web アプリケーションにおいて採用した技術スタック（Go, Next.js, Docker, Cloud Run 等）について，その選定理由と技術的な優位性を詳述する．

§ 2.1　ホタルイカの生態と身投げ発生のメカニズム

ホタルイカの生活史と富山湾への回帰

毎年産卵期の 2 月から 6 月にかけて，富山湾ではホタルイカ（*Watasenia scintillans*）が大挙して海岸に押し寄せ，波に巻かれて浜辺に打ち上げられる現象が観測される．(図 2.1) 真っ暗な海岸で多数のホタルイカがブルーの発光器を明滅させ，海岸を青く染めて打ち寄せるさまはたいへん幻想的な光景である．富山周辺ではこの現象は「ホタルイカの身投げ」とよばれ，古く江戸時代から知られており,「蜃気楼」とともに春先の富山湾の風物詩となっている．

富山湾は、陸地からわずかな距離で急激に深海へと落ち込む、通称「あいがめ」と呼ばれる特異な海底地形を有していることで知られる [4]（図 2.2）．岸に近い水深 30m から 60m の沿岸域に設置された定置網には、夜間に深海から浮上した深海生物が、一般の魚類とともに生きたまま混獲される．この同じ定置網に、毎年産卵期を迎える 2 月から 6 月にかけて、成熟して外套長 4cm から 6cm に達したホタルイカが大量に入網することが古くから知られている．

ホタルイカの寿命は短く、およそ 1 年と考えられている．山陰沖や富山湾周辺で孵化した稚イカは、成長に伴って日本海全域を大きく回遊し、翌年の春には再び自身が孵化した海域に回帰して産卵するという生活史を持つ．富山湾に回帰した産卵期のホタルイカは、昼間は外敵を避けるため水温の低い湾内の水深 200m から 300m 付近に群をなして留まっている．その後、夕刻になると一斉に浮上を開始し、午前 0 時頃には産卵のため海岸近くの浅所（水深 300m 付近から海面表層）に出現する [6].

こうした規則的な産卵行動は、日周期活動や月周期活動、概年リズムといった複数の生物時計の精緻な組み合わせによって制御されていると考えられている．実際、この行動は月の満ち欠けと強く連動しており、満月の日に活発化し、特に 4 月から 5 月の満月の大潮付近に当該年度の出現のピークを迎える．これら一連の行動は環境要因と生理的要因の密接

図 2.1: ホタルイカの身投げの様子

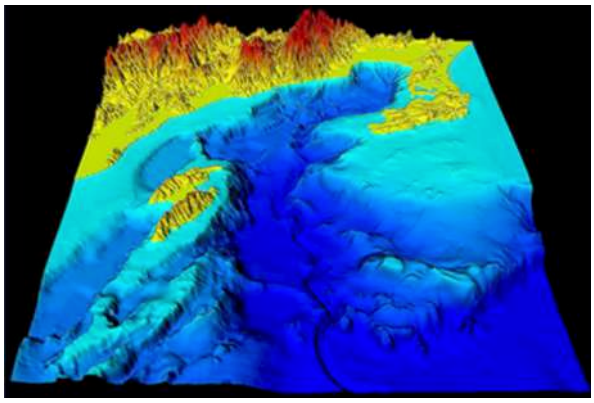

図 2.2: 富山湾の特異な海底地形「あいがめ」

な連鎖によって成り立っており、その要素の一つでも欠けると、正常な産卵行動は完遂されないことが示唆されている [7].

視覚特性と環境光の利用

ホタルイカは中深海に生活の基盤をおくが，海水中の環境よりはむしろ，海水外の環境要因，とくに「環境光」を積極的に生活に利用していることが報告されている [8]．ホタルイカは特異的に分化した層状の網膜内に 3 種の視物質を個別に配置しており，これによって脊椎動物とはまったく異なる高度な波長識別能を有する．

また，ホタルイカは自ら発する光の色調を黄系から青色系まで変化させることができる能力を持つ．彼らの発光の極大波長は，自身の持つ視物質の極大波長によく一致しており，まわりの光の強さに応じて発光波長と発光強度を調節することが可能である．この高度な光受容・発光能力が，次節で述べる定位行動や鉛直移動において重要な役割を果たしている．

先行研究における身投げ発生要因の知見

本研究において AI による予測モデルを構築するためには，先行研究によって明らかにされている身投げ発生のメカニズムと，関与する環境因子を正しく理解する必要がある．既存の海洋研究や漁獲統計の分析 [7] によれば，身投げ現象は単なる光への誘引ではなく,「月光による定位行動の喪失」と「気象条件による物理的輸送」が複合的に重なった際に発生すると結論付けられている．本研究では，以下の 3 点を予測における重要な環境因子として位置づける．

1. **月光と定位行動の喪失（対象：月齢・照度）**
   産卵期のホタルイカは，本来は月光を基準にした「定位行動」に従って陸地と沖の間を移動しているとされる．しかし，新月（月齢 0）前後においては，定位の基準となる月光が存在しない．このため，ホタルイカは方位決定を誤り，制御を失って海岸に異常接近してしまう．すなわち，新月という条件は，身投げが発生するための「前提条

件（個体が沿岸に溜まる状態）」を作り出す因子であると解釈できる．

2. **南風による物理的漂着（対象：風向・風速）**
   方位決定を誤って海岸に接近した個体群が，最終的に浜辺に打ち上げられるかどうかは「風」に依存する．先行研究では，新月前後で定位を失っている状況下において,「南寄りの風」が吹くことで発生する潮の流れや波が，ホタルイカを陸地へと打ち上げる物理的な駆動力になると報告されている．逆に，北寄りの風は身投げの発生を抑制する要因となり得る．
3. **降雨による垂直浮上の阻害（対象：降水量）**
   ホタルイカが深海から表層へ移動する「垂直日周活動」は，日没に伴う光強度の減少がトリガーとなっている．しかし，垂直浮上が開始される夕刻の時間帯に「降雨」が存在すると，この行動自体が誘発されないとの報告がある [7]．雨天時は海岸近くの浅所に来遊する個体数が激減することが定置網の漁獲データからも確認されており，降雨は身投げ発生に対する強い「阻害要因（負のバイアス）」として機能すると考えられる．

以上の既往研究の知見に基づくと，身投げは「新月」かつ「南風」かつ「非降雨」の条件下で発生確率が最大化すると考えられる．本研究では，これらの気象・海象条件を特徴量として機械学習モデルに組み込むことで，身投げ発生の有無および発生量の予測を試みる．

§ 2.2　既存サービスと本研究の立ち位置

既存の情報収集手段の限界

現在，身投げの予兆を知るために利用されている手段は，主に「SNS によるリアルタイム検索」「汎用気象予報サイトの閲覧」「旧来の掲示板サイト」の 3 つに大別される．しかし，これらは以下の課題を抱えている．

1. **情報の即時性と予測性の欠如**
   SNS（X や Instagram 等）は,「今，湧いている」という事実を知るには有効であるが，現地に行く前の計画段階で必要な「予報」を得ることはできない．また，情報のノイズが多く，過去のデータとして蓄積・分析することも困難である．
2. **多変量データの統合における認知負荷**
   気象予報サイトでは，天気・風速・波高・潮位などが個別の数値として提供される．利用者はこれらの変数を脳内で統合し,「今日は条件が良いか」を判断しなければならない．これには高度な経験と知識が必要であり，初心者にとっては「行ってみたが何もいなかった（空振り）」という徒労を招く原因となっている．
3. **ユーザビリティ (UX) の課題**
   既存のホタルイカ情報掲示板は，2000 年代初頭の設計のまま運用されているものが多く，スマートフォンでの閲覧に最適化されていない．また，画像投稿のハードルが高い，過去のログが検索できないなど，現代の Web 標準と比較してユーザビリティが著しく低い．

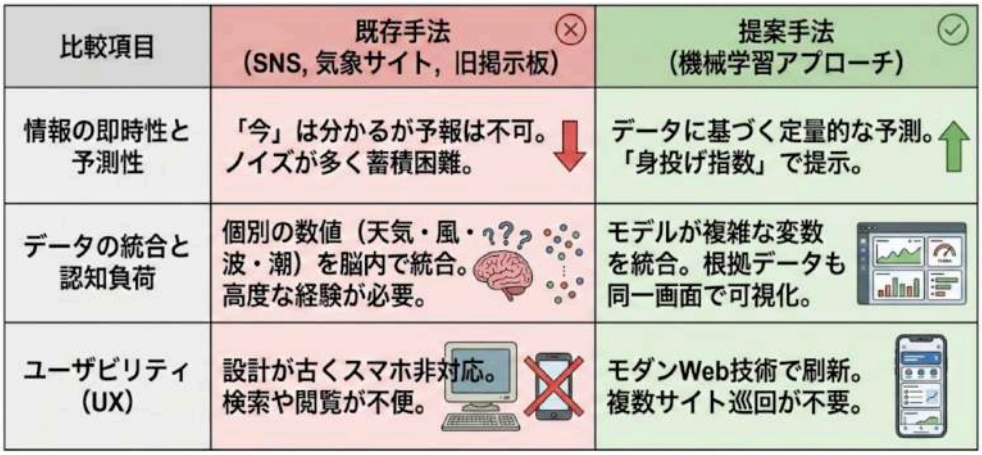

図 2.3: 既存手法と提案手法（機械学習アプローチ）の比較

本研究のアプローチと新規性

本研究では，これらの課題を「機械学習による定量的な予測」と「モダン Web 技術による UX の刷新」の両面から解決することを目的とする（図 2.3）.

- **データ駆動型アプローチによる予測の客観化**
  経験則に依存していた予測プロセスを，過去 10 年間の気象・海洋データと身投げ実績データを用いた機械学習モデル（LightGBM）に置き換える．これにより，気象条件の複雑な組み合わせを「身投げ指数」という単一の指標に落とし込み，誰にでも分かりやすい形で提供する点が本研究の最大の新規性である.

- **情報の集約と可視化**
  単なる数値予測だけでなく，判断の根拠となる詳細データ（時間ごとの風向き，潮位グラフ等）を同一インターフェース上で可視化する．これにより，利用者は複数のサイトを巡回する必要がなくなる.

§ 2.3　Web アプリケーション開発技術の選定理由

バックエンド：静的型付け言語 Go の優位性

本システムのサーバーサイド言語には，Google が開発した Go 言語（Golang）を採用した．近年，Web サービスのバックエンド開発において，Python や Node.js（JavaScript）などの動的型付け言語から，Go や Rust などの静的型付け言語への移行が進んでいる [9].

- **型安全性による堅牢性の確保**
  Go はコンパイル時に厳密な型チェックを行うため，実行時エラー（Runtime Error）の多くを未然に防ぐことができる（図??）．これは，予期せぬシステムダウンが許されない Web サービスの運用において大きな利点となる.

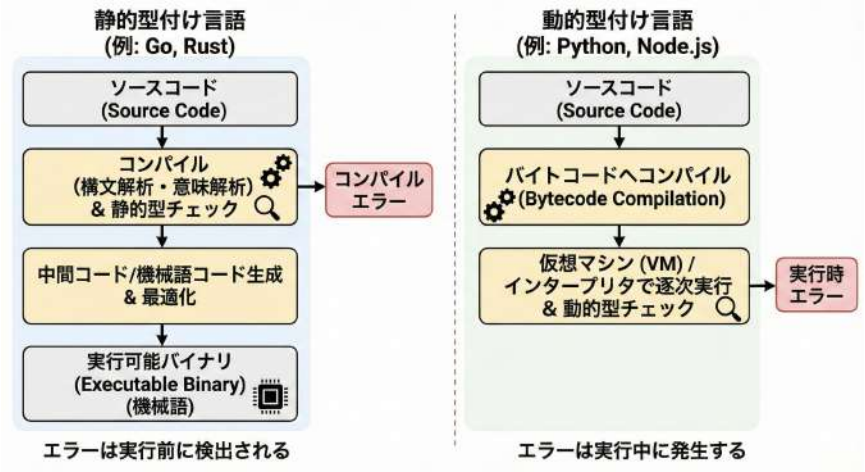

図 2.4: 静的型付け言語と動的型付け言語のコンパイル・実行プロセスの比較

- **並行処理性能 (Goroutines)**
  Go は「Goroutine」と呼ばれる軽量スレッドを言語レベルでサポートしている．これにより，多数のユーザーからのリクエストを少ないメモリ消費で同時に処理することが可能である．ホタルイカのシーズン中，突発的なアクセス増加が予想される本システムにおいて，この高い並行処理性能は不可欠である．

- **API 開発における FastAPI (Python) の役割**
  一方で，機械学習モデルの推論部分には Python の FastAPI を採用した．これは，機械学習エコシステム（scikit-learn, LightGBM 等）が Python に集中しているためである．Go をメインの Web サーバーとし，推論が必要な場合のみ Python のマイクロサービスを呼び出す構成とすることで，Web サーバーの高速性と機械学習の利便性を両立させている．

アーキテクチャ設計：標準ライブラリによる関心事の分離

Go 言語は，標準ライブラリの `net/http` パッケージが非常に強力であり，外部フレームワークに依存せずとも高機能な Web サーバーを構築可能である．本システムでは，過度な抽象化によるオーバーヘッドを避けるため，Web フレームワークを使用せず，標準ライブラリのみを用いたシンプルな構成を採用した．

一方で，コードの保守性と可読性を維持するため,「クリーンアーキテクチャ」の思想を取り入れた 3 層構造（レイヤードアーキテクチャ）による責務分離を行った（図 2.5）[11].

1. **プレゼンテーション層 (Handler)**
   Go の標準パッケージである `net/http` を用いて HTTP リクエストを受け付け，レスポンスを返却する「入り口」のレイヤーである．ここではリクエストボディの JSON パースやステータスコードの制御のみを行い，具体的な計算処理は行わない．
2. **ユースケース層 (Service)**
   本システムの核心であるビジネスロジックを実装するレイヤーである.「気象データか

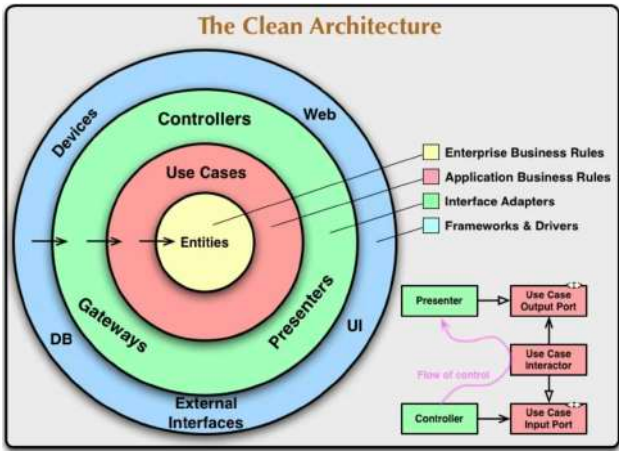

図 2.5: レイヤードアーキテクチャの構成

ら身投げ指数を判定する」「過去データを集計する」といった処理を行う．この層は HTTP 通信の詳細（ヘッダーやクエリパラメータ等）を知る必要がなく，純粋な Go の構造体と関数のみで記述される．

3. **インフラストラクチャ層 (Repository)**
   データベース（Supabase）への SQL 発行や，データ変換を行う「出口」のレイヤーである．

このように，標準ライブラリをベースにしつつ適切に階層化を行うことで,「外部ライブラリへの依存リスク」を最小限に抑えながら，長期的な運用に耐えうる堅牢なシステム構造を実現した．

フロントエンド：コンポーネント指向と ISR

フロントエンド開発には，React ベースのフレームワークである Next.js を採用した．従来のサーバーサイドレンダリング（JSP や PHP 等）と比較し，以下の技術的優位性がある．

- **コンポーネント指向による再利用性**
  UI を独立した部品（コンポーネント）として管理することで，コードの再利用性が高まり，保守性が向上する．例えば,「身投げ指数グラフ」や「天気カード」などの部品を定義すれば，それらを組み合わせるだけで効率的にページを構築できる．

- **ISR (Incremental Static Regeneration) の活用**
  本システムでは，Next.js の ISR 機能を活用する．ISR とは，事前に静的な HTML を生成（ビルド）しておきつつ，一定時間ごとにバックグラウンドでデータを再取得し，ページを更新する技術である [12]．気象予報や身投げ予測は，秒単位で変化するものではなく，1 時間に 1 回程度の更新で十分である．ISR を採用することで，ユーザーには静的ファイル（HTML/CSS）を高速に配信しつつ，サーバーへの負荷を大幅に軽減することが可能となる．

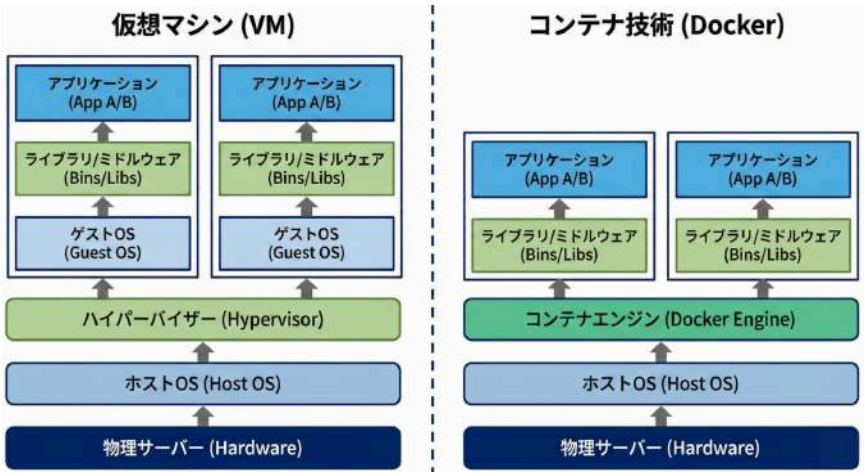

図 2.6: 仮想マシン (VM) とコンテナ技術 (Docker) のアーキテクチャ比較

インフラストラクチャ：コンテナ技術とサーバレス

開発環境および本番環境の構築には，Docker と Google Cloud Run を採用した．

- **コンテナ技術 (Docker) による環境の抽象化**
  Docker は，アプリケーションとその依存関係（ライブラリ，OS 設定など）を「コンテナ」という単位にパッケージ化する技術である（図 2.6）．これにより，「開発者の PC では動いたが，本番サーバーでは動かない」といった環境差異に起因する問題を排除できる．本研究では，Frontend，Backend，Database をそれぞれコンテナ化し，`docker-compose` を用いてオーケストレーションを行うことで，開発環境の再現性を担保している．

- **サーバレスアーキテクチャ (Cloud Run)**
  Google Cloud Run は，コンテナをサーバーレスで実行できる環境である．従来の仮想マシン（VM）方式では，アクセスがない夜間やオフシーズンであってもサーバーを常時稼働させる必要があり，維持コストがかさんでいた．対して Cloud Run は，リクエストが発生した瞬間のみコンテナを起動し，処理が終了すれば停止する「ステートレス」な動作を基本とする．これにより，リクエスト数に応じた従量課金が適用され，オフシーズンのランニングコストをほぼゼロに抑えることが可能となる．この特性は，季節性が極めて強い「ホタルイカ予測」というアプリケーションの性質に合致している．

開発プロセスの自動化：GitHub Actions による CI/CD

本研究のような個人開発あるいは小規模開発において，開発サイクルを高速化し，かつ品質を担保するためには，手作業によるオペレーションを排除する必要がある．本システムでは，GitHub Actions [13] を用いた CI/CD（Continuous Integration / Continuous Delivery）パイプラインを構築した．

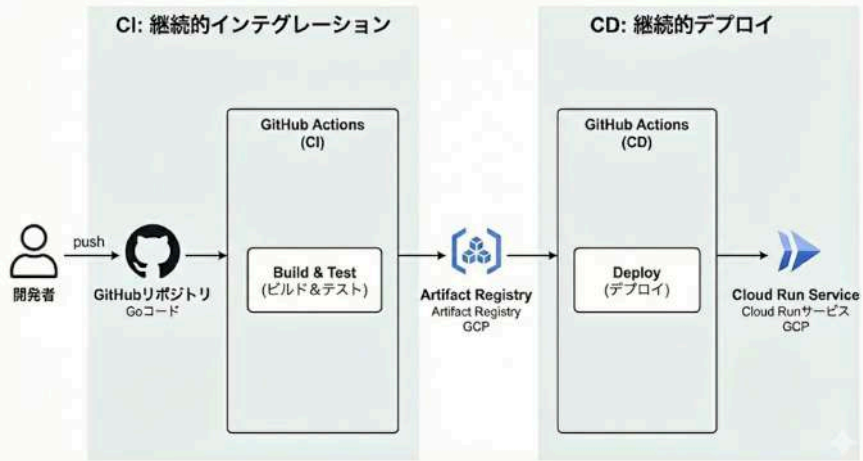

図 2.7: GitHub Actions による CI/CD パイプライン

- **継続的インテグレーション (CI)**
  ソースコードが GitHub のリポジトリにプッシュされたタイミングで，Go の静的解析ツール（Linter）および単体テストが自動的に実行される．クリーンアーキテクチャによってロジックが分離されているため，データベース接続をモック（模倣）した高速なテストが可能であり，バグの早期発見に寄与している．

- **継続的デリバリー (CD)**
  テストを通過し，main ブランチへマージされたコードは，自動的に Docker イメージとしてビルドされる．生成されたイメージは Google Artifact Registry へとプッシュされ，最終的に Cloud Run 上の本番環境へデプロイされる（図 2.7）．

この自動化により，ホタルイカのシーズン中のような一刻を争う状況下でも，機能追加やバグ修正を安全かつ迅速にユーザーへ届けることが可能となった．

データベース：BaaS の活用

データの永続化には，Supabase（PostgreSQL）[14] を採用した．これは，近年注目されている BaaS（Backend as a Service）の一種である．

掲示板機能において，投稿（Post）とそれに対する返信（Reply），あるいはリアクション（Reaction）といったデータは，相互に強い関連性を持つ．このような構造化されたデータを矛盾なく管理するためには，NoSQL ではなく，厳密なスキーマを持つリレーショナルデータベース（RDB）が適している．Supabase のようなマネージドサービスを利用することで，セキュリティパッチの適用やバックアップの管理といったインフラ管理コストを削減し，アプリケーションのロジック開発や精度向上といった本質的な研究活動に注力することが可能となった．

第3章

機械学習と時系列データ分析

本章では，ホタルイカの身投げ量を予測するために採用した機械学習手法と，時系列データを分析するための理論的背景について述べる．本研究では，気象条件や月齢といった多変量データから数値を予測する回帰問題として定式化し，勾配ブースティング決定木の一種である LightGBM を採用した．また，時間的な順序を持つデータの特性を考慮した前処理と評価手法についても詳述する．

§ 3.1　勾配ブースティング決定木と LightGBM

決定木分析とアンサンブル学習

本研究で扱うデータは，気温，風速，月齢といった数値データが表形式で整理された構造化データである．このようなデータに対して高い予測精度を示す手法として，決定木に基づいたアンサンブル学習が知られている．

決定木（Decision Tree）は，データの特徴量に対して「ある閾値以上か未満か」という条件分岐を繰り返し，データを分類あるいは回帰する手法である．単一の決定木は，結果の解釈が容易であるという利点を持つ一方で，学習データに対して過剰に適合する「過学習（Overfitting）」を起こしやすいという課題がある．

この課題を解決するために用いられるのがアンサンブル学習（Ensemble Learning）である．アンサンブル学習とは，複数のモデル（弱学習器）を組み合わせることで，単一のモデルよりも高い汎用性と予測精度を実現する手法である．代表的なアプローチには，複数の決定木を並列に学習させ多数決や平均を取る「バギング（Bagging）」と，モデルを逐次的に学習させ，前のモデルの弱点を次のモデルが補う「ブースティング（Boosting）」がある [15].

勾配ブースティング決定木 (GBDT)

勾配ブースティング決定木（Gradient Boosting Decision Tree: GBDT）は，ブースティングの手法を用いた強力な機械学習アルゴリズムである．GBDT では，最初に作成した決定木の予測誤差（残差）を計算し，その誤差を予測するように 2 つ目の決定木を作成する．さらに，その予測結果の誤差を 3 つ目の決定木が予測する，というプロセスを繰り返す．

数学的には，損失関数（Loss Function）を定義し，その勾配（Gradient）に沿って誤差

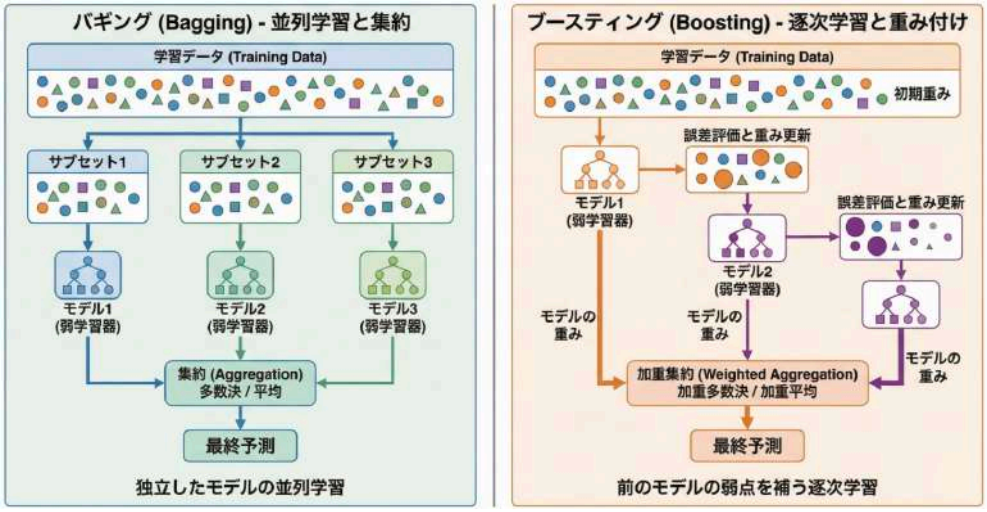

図 3.1: アンサンブル学習におけるバギングとブースティングの違い

を最小化するように新たな木を追加していくことから「勾配ブースティング」と呼ばれる．最終的な予測値は，作成されたすべての決定木の予測値に学習率（Learning Rate）を掛け合わせたものの総和となる．これにより，個々の決定木は単純なモデルであっても，全体として非常に複雑な非線形関係を表現することが可能となる [16].

GBDT の数理的最適化プロセス

GBDT は，目的関数を最小化するために加法モデル（Additive Model）の形式で学習を行う．時点 t における予測値を $\hat{y}_i^{(t)}$ とすると，これは前段までの予測値 $\hat{y}_i^{(t-1)}$ に新たな決定木 $f_t(x_i)$ を加えたものとして以下のように表される．

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{3.1}$$

このとき，最小化すべき目的関数 $\mathrm{Obj}^{(t)}$ は，損失関数 L とモデルの複雑さを抑える正則化項 Ω を用いて以下のように定義される．

$$\mathrm{Obj}^{(t)} = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{3.2}$$

ここで，LightGBM を含む近代的な GBDT アルゴリズムでは，損失関数を現在の予測値 $\hat{y}_i^{(t-1)}$ の周りで 2 次のテイラー展開（Taylor Expansion）によって近似し，最適化を効率化している．

$$\mathrm{Obj}^{(t)} \simeq \sum_{i=1}^{n} [L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2] + \Omega(f_t) \tag{3.3}$$

ただし，g_i および h_i はそれぞれ損失関数の 1 次微分（勾配：Gradient）および 2 次微分（ヘシアン：Hessian）であり，以下のように定義される．

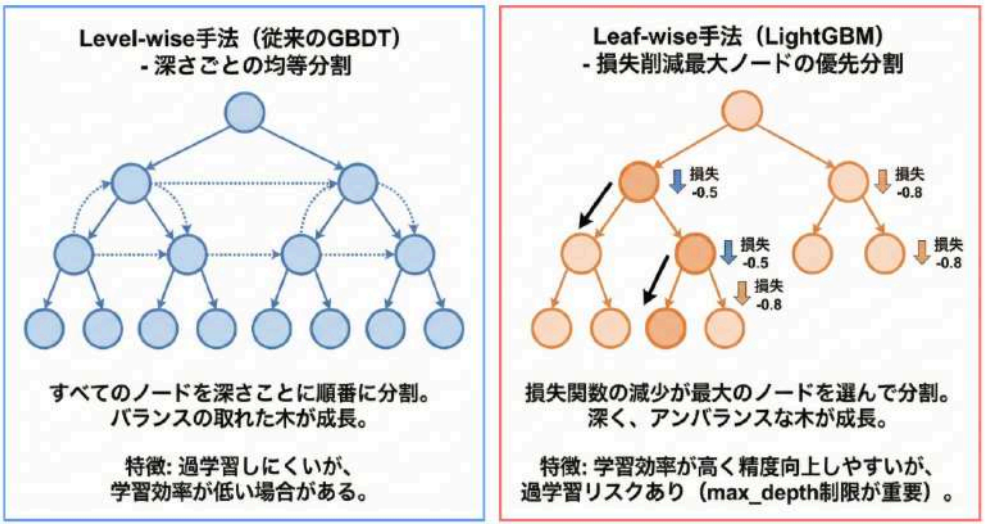

図 3.2: Level-wise 手法と Leaf-wise 手法の比較

$$g_i = \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}, \quad h_i = \frac{\partial^2 L(y_i, \hat{y}_i^{(t-1)})}{\partial (\hat{y}_i^{(t-1)})^2} \tag{3.4}$$

この 2 次近似により，任意の損失関数に対して統一的な最適化手法を適用することが可能となり，本研究のような複雑な要因が絡む回帰問題においても高い柔軟性を発揮する．

LightGBM の特徴と優位性

本研究では，GBDT の実装として，Microsoft 社によって開発された LightGBM を採用した．LightGBM は，従来の XGBoost などの GBDT 実装と比較して，以下の特徴により，大規模なデータセットに対しても高速かつ高精度な学習が可能である．

Leaf-wise (Best-first) 分割

従来の GBDT は，決定木の深さごとにすべてのノードを分割する「Level-wise」手法を採用していた．これに対し LightGBM は，損失関数の減少が最も大きくなるノードを選んで分割する「Leaf-wise」手法を採用している（図 3.2 参照）．これにより，同じノード数の場合，Level-wise よりも学習データの損失を小さく抑えることができ，予測精度が向上する傾向がある．ただし，木が深くなりすぎることで過学習を起こすリスクがあるため，木の深さ（max_depth）に制限を設けることが重要となる [17].

ヒストグラムベースのアルゴリズム

LightGBM は，連続値の特徴量をバケット（ビン）に分割し，ヒストグラム化して扱う．これにより，分割点の探索計算量が大幅に削減され，メモリ使用量も効率化される．気象データのような連続値が多い本研究のデータセットにおいて，この特性は学習時間の短縮に大きく寄与する．

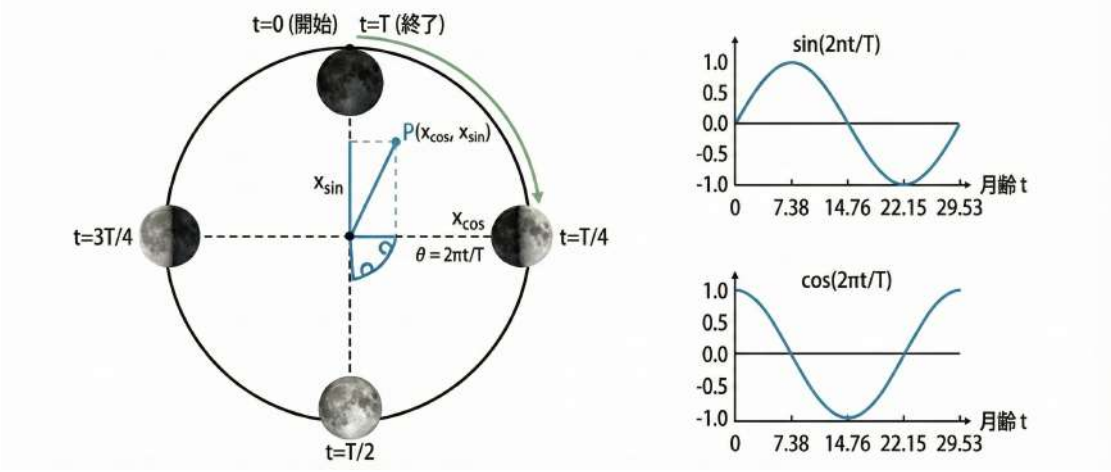

図 3.3: 三角関数を用いた周期的データの表現

§ 3.2　時系列データの特性と処理手法

時系列データの特性と課題

本研究で扱うデータは，日ごとの気象条件や身投げ量であり，時間の経過とともに観測された「時系列データ」である．一般的な機械学習のデータセットとは異なり，時系列データには以下の特性がある．

1. **時間的な順序性**: データの順序に意味があり，シャッフルすることができない．
2. **自己相関（Autocorrelation）**: 昨日の値が今日の値に影響を与えるといった，時間的な依存関係が存在する．
3. **周期性（Seasonality）**: ホタルイカの身投げは春に集中するといった季節性や，潮汐のような周期的な変動を含む．

これらの特性をモデルに正しく学習させるためには，適切な特徴量エンジニアリング（変数の加工）が必要不可欠である．特に本研究では,「周期的な情報の数値化」と「過去の情報の活用」に重点を置いた．

周期的特徴量のエンコーディング

時間や日付に関するデータ（月，日，時間，月齢など）は，そのまま数値として扱うと問題が生じる場合がある．例えば，時間は「0 時」から「23 時」まで存在するが，数値としての「0」と「23」は大きく離れている．しかし，現実の時間感覚では，23 時の 1 時間後は 0 時であり，両者は連続している．

このように，数値上の大小関係と実態の連続性が乖離している問題を解決するために，本研究では三角関数（サイン･コサイン）を用いた周期的エンコーディング（Cyclical Encoding）を導入した．周期 T を持つ変数 t に対して，以下の変換を行うことで，データを単位円上の座標 $(x_{\sin}, x_{\cos})$ として表現する [18].

$$x_{\sin} = \sin\left(\frac{2\pi t}{T}\right) \tag{3.5}$$

$$x_{\cos} = \cos\left(\frac{2\pi t}{T}\right) \tag{3.6}$$

例えば，月齢（約 29.5 日周期）の場合，$T = 29.53$ として変換を行う．これにより，周期の始まりと終わりが円環上で接続され，機械学習モデルが周期性を正しく認識できるようになる．

ラグ特徴量の導入

時系列予測において，過去の観測値は将来を予測するための最も強力な情報源の一つである．そこで,「ラグ特徴量（Lag Features）」を作成し，モデルの入力データに追加した．ラグ特徴量とは，時点 t の予測を行う際に，時点 $t-1$（1 日前）や $t-2$（2 日前）のデータを説明変数として利用するものである．

本研究では，当日の気象予報データだけでなく，過去数日間の気温，風速，および過去の身投げ発生状況をラグ特徴量として組み込んだ．これにより，モデルは「前日に波が高かった場合，翌日に身投げが発生しやすい」といった，時間的な因果関係や傾向を学習することが可能となる [19].

時系列交差検証 (Time Series Cross-Validation)

機械学習モデルの性能を評価する際，通常はデータをランダムに分割する「K 分割交差検証（K-Fold Cross-Validation）」が用いられる．しかし，時系列データに対してランダムな分割を行うと,「未来のデータを使って過去を予測する」という状況（リーケージ：Leakage）が発生してしまう．これでは，実際の運用環境（未来の予測）とは異なる状況で評価することになり，モデルの性能を過大評価する危険性がある．

この問題を避けるため，本研究では「時系列交差検証（Time Series Split）」を採用した．これは，訓練データとテストデータの時間的な順序を守りながら検証を行う手法である．具体的には，ある時点までのデータを訓練用とし，その直後のデータをテスト用とする．検証が進むごとに訓練データの期間を拡張し（Walk-Forward Validation），常に「過去のデータで学習し，未来のデータを予測する」形を維持する（図 3.4 参照）.

この手法を用いることで，実運用時と同様の条件下でモデルの汎化性能を正しく評価することが可能となる [20].

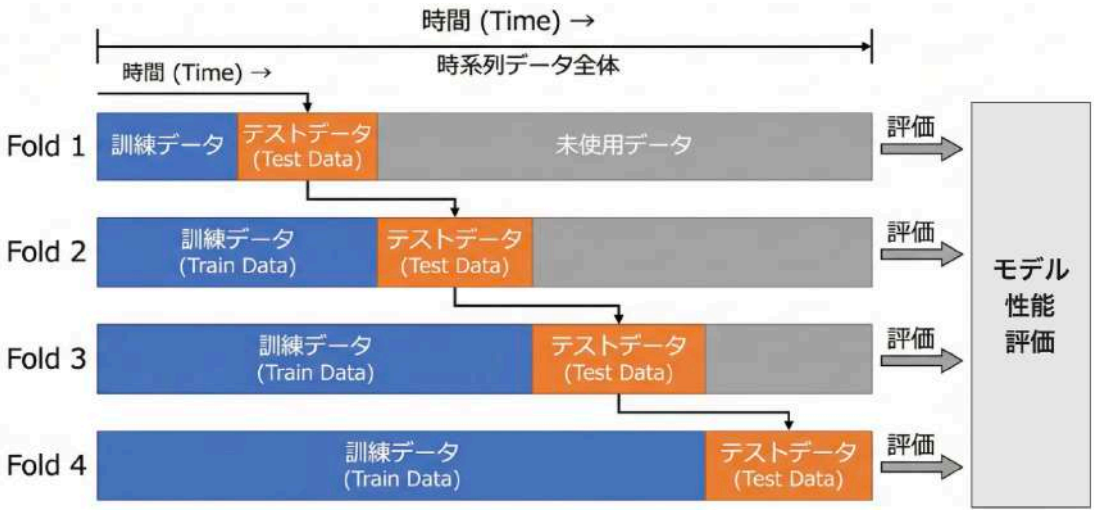

図 3.4: 時系列データを考慮した交差検証の仕組み

§ 3.3　モデルの評価指標と解析手法

評価指標の定義

構築した予測モデルの性能を定量的に評価するために，本研究では以下の 3 つの指標を採用した [21]．ここで，n はデータ数，y_i は i 番目のデータの実測値，$\hat{y}_i$ はモデルによる予測値，$\bar{y}$ は実測値の平均を表す．

決定係数 (R^2 Score)

決定係数は，モデルがデータの変動をどれだけ説明できているかを表す指標であり，1 に近いほど当てはまりが良いことを示す．回帰モデルの全体的な性能を把握するために用いる．

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.7}$$

平均絶対誤差 (MAE: Mean Absolute Error)

MAE は，予測値と実測値の差（誤差）の絶対値の平均である．外れ値の影響を受けにくく，誤差の大きさを直感的に解釈しやすいという特徴がある．

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.8}$$

二乗平均平方根誤差 (RMSE: Root Mean Squared Error)

RMSE は，誤差を二乗して平均し，その平方根をとったものである．二乗することによって，MAE よりも大きな誤差に対してペナルティを大きく与える特性がある．ホタルイカの身投げ予測においては，突発的な大量発生（爆湧き）や皆無の日を大きく外さないことが重要であるため，この指標も併せて評価に用いる．

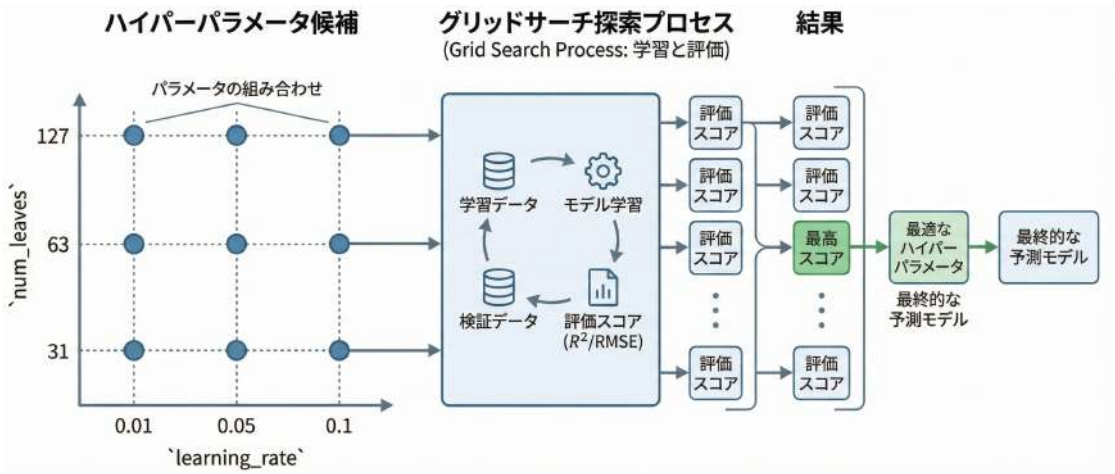

図 3.5: ハイパーパラメータ最適化の概念

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.9}$$

ハイパーパラメータの最適化

LightGBM を含む機械学習モデルには，学習によってデータから自動的に決定される「パラメータ」とは別に，人間があらかじめ設定する必要がある「ハイパーパラメータ」が存在する．主なハイパーパラメータには以下のようなものがある．

- **num_leaves**: 決定木の葉の最大数．モデルの複雑さを制御し，大きくすると表現力が上がるが過学習しやすくなる．
- **max_depth**: 決定木の深さの制限．
- **learning_rate**: 学習率．小さいほど学習が慎重に進み精度が上がりやすいが，計算時間がかかる．
- **n_estimators**: 作成する決定木の総数．

これらの値はデータの性質によって最適な組み合わせが異なる．不適切なハイパーパラメータを設定すると，訓練データには適合しても未知のデータには適合しない「過学習（Overfitting）」や，学習不足による「未学習（Underfitting）」を引き起こす原因となる．

本研究では，最適なハイパーパラメータの組み合わせを探索するために「グリッドサーチ（Grid Search）」を用いた．グリッドサーチとは，調整したいパラメータの候補値をあらかじめいくつか設定し，そのすべての組み合わせについてモデルの学習と評価（前述の時系列交差検証）を行う手法である．探索の結果，最も検証スコア（本研究では R^2 または RMSE）が良かったパラメータの組み合わせを採用し，最終的な予測モデルを構築する [22].

特徴量重要度によるモデルの解釈

構築された機械学習モデルが，どのような根拠に基づいてホタルイカの身投げ量を予測しているかを明らかにすることは，予測精度の確認と同等に重要である．特に本研究で用いる LightGBM のようなアンサンブル学習モデルは，変数間の複雑な交互作用を捉えられる一方で，その判断プロセスが不透明（ブラックボックス）になりやすい．そこで本研究では，モデル内部での各特徴量の寄与度を定量化する「特徴量重要度（Feature Importance）」を解析手法として導入する．

LightGBM における重要度の算出には，主に「Split（分割回数）」と「Gain（利得）」の 2 つの指標が用いられる．

1. **Split ベースの重要度**
   ある特徴量 x_j が，全決定木の中でノードの分割に使用された総回数である．これは，その変数がどれだけ頻繁に判断基準として選ばれたかを示す指標であり，モデルの構造的な安定性や寄与の普遍性を評価するのに適している．

$$I_{split}(x_j) = \sum_{k=1}^{T} \sum_{v \in \mathcal{N}_k} \mathbb{I}(v_{feat} = x_j) \tag{3.10}$$

   ここで，T は学習された決定木の総数，$\mathcal{N}_k$ は k 番目の木の非葉ノード集合，$\mathbb{I}(\cdot)$ は条件が真のときに 1 を返す指示関数である．

2. **Gain ベースの重要度**
   ある特徴量 x_j による分割が，ターゲット（身投げ量）の予測誤差をどれだけ減少させたかの合計値である．これは予測精度への直接的な寄与度を表す．ノード v において利得 G は，勾配 g_i とヘシアン h_i を用いて以下の数式で定義される．

$$G = \frac{1}{2}\left[\frac{(\sum_{i\in L} g_i)^2}{\sum_{i\in L} h_i + \lambda} + \frac{(\sum_{i\in R} g_i)^2}{\sum_{i\in R} h_i + \lambda} - \frac{(\sum_{i\in \text{all}} g_i)^2}{\sum_{i\in \text{all}} h_i + \lambda}\right] - \gamma \tag{3.11}$$

本研究では，多種多様な気象条件（風向，風速，月齢等）の中から，モデルが予測の際に「どの変数を最も普遍的な判断基準として多用したか」を明らかにするため，Split ベースの重要度を主として採用する．これにより，特定の条件下でのみ効く因子ではなく，シーズンを通じて身投げ予測の根幹をなす支配的要因を特定することが可能となる．

第4章

提案手法

本章では，第3章で述べた時系列解析および機械学習の理論的背景に基づき，本研究において構築した「ホタルイカ身投げ予測システム」の具体的な実装手法について詳述する．本システムは，非構造化データである掲示板の口コミを定量化するデータマイニングフェーズ，LightGBMを用いた予測モデル構築フェーズ，および予測結果をエンドユーザーに提供するWebアプリケーション実装フェーズの3段階で構成される．以下，各フェーズにおける技術的選定の根拠と実装の詳細について論じる．

§ 4.1　データセット構築と特徴量エンジニアリング

機械学習モデルの予測精度は，入力データの質と量に強く依存する．特に本研究の課題であるホタルイカの身投げ現象は，公的な観測データが存在しないため，信頼性の高い正解ラベル（教師データ）を独自に生成することが最大の課題となる．本章では，機械学習モデルの入力となる特徴量の生成について述べる．具体的には，多角的な環境データの統合プロセス（図4.1参照）と，定性的な口コミ情報から定量的な特徴量を抽出する手法について述べる．

口コミデータの収集と生成AIによる定量化

本研究では，富山県のホタルイカ愛好家の間でデファクトスタンダードとなっている地域掲示板サイト「ホタルイカ掲示板」[23]を情報源として採用した．この掲示板には，現地に赴いたユーザーからリアルタイムな出現状況が数多く投稿されている．

スクレイピングによるデータ収集とその配慮

Pythonのスクレイピングライブラリを用いて同サイトの過去ログを取得した．対象期間は2015年から2025年までの10年間とし，ホタルイカの接岸シーズンである各年2月から5月のデータに限定した．なお，スクレイピングの実施にあたっては，対象サイトのサーバー負荷を最小限に抑えるため，リクエスト間に十分な待機時間を設ける（Polite Scraping）とともに，取得したHTMLデータから不要なタグ情報を除去し，純粋なテキストデータのみを抽出するクレンジング処理を実装した．この結果，1220日分に相当する投稿データをデータベース化することに成功した．

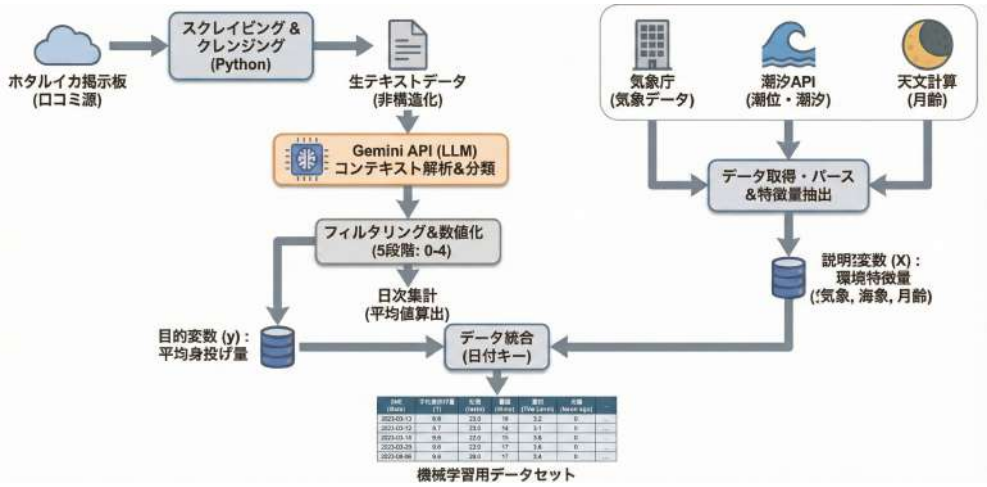

図 4.1: データ収集からデータセット構築までの処理フロー

Gemini API を用いたコンテキスト解析と数値化

収集した口コミデータは自然言語（非構造化データ）であり，そのままでは機械学習の学習データとして利用できない．従来の手法として「特定の単語（例：『たくさん』『湧いた』）の出現頻度を数える」といったキーワードマッチングが存在するが，文脈を考慮できないため,「昨日は湧いた」といった過去形の報告や,「人は多いがイカはいない」といった報告を誤検知するリスクがあった．

この課題を解決するため，本研究では Google の大規模言語モデル（LLM）である Gemini API [24] を導入し，各投稿を解析・分類させた．具体的には，以下のルールと定義を含むプロンプトを構築し，各テキストを 7 つのカテゴリに分類した．

1. 分類ルールとフィルタリング

プロンプトには，以下の判断基準を明記し，ノイズの除去を徹底した．

- **リアルタイム性の確認**: 「昨日」「先週」「身投げ跡」といった記述が含まれる場合，その時点の状況ではないため「時間不明」として除外する．
- **直接的な目撃情報の優先**: 「人の多さ」や「車の混雑」に関する記述など，ホタルイカ自体の記述がない場合は「不明」として除外する．

2. 湧き具合の 5 段階分類

有効なリアルタイム情報と判断された投稿については，その記述内容に基づき，湧き量を以下の 5 段階のカテゴリに分類した．

なし ホタルイカが全く観察されていない状態（キーワード：全くいない，皆無，成果なし，0 匹）．

少ない わずかな数が観察された状態（キーワード：ちらほら，ポツポツ，数匹〜10 匹程度）．

普通 平均的な数が観察された状態（キーワード：そこそこ，まあまあ，10〜50 匹程度）．

多い かなりの数が観察された状態（キーワード：たくさん，50〜100 匹程度，多く見られる）．

非常に多い 異例に大量の数が観察された状態（キーワード：爆湧き，足の踏み場がない，100 匹以上，キロ単位）．

日次集計による正解ラベルの定義

AI によって出力された分類カテゴリ（なし〜非常に多い）を，解析のためにそれぞれ 0 から 4 までの整数値にマッピングした（「時間不明」「不明」はデータから除外）．その後，同一日付の有効なデータを集約し，そのスコアの平均値を算出することで，その日の「平均身投げ量（avg_amount）」を定義した．例えば，ある日に「非常に多い（4）」と「普通（2）」の報告があった場合，その日の指標は 3.0 となる．このように連続値として扱うことで，0 か 1 かの二値分類よりも細かな粒度での回帰分析が可能となった．

環境データの収集と統合処理

目的変数である身投げ量に対し，予測の手掛かりとなる説明変数（特徴量 X）は，気象学的要因と海洋学的要因の双方から収集を行った．これらは日付をキーとして単一のデータフレームに統合される．

- **気象データ**: 気象庁の「過去の気象データ・ダウンロード」サイト [25] より，富山県内の沿岸観測地点における過去 10 年分のデータを取得した．CSV 形式で提供されるデータをパースし，天気概況，気温，降水量，風向，風速の各項目を抽出した．
- **潮位・潮汐データ**: 潮汐 API [26] を利用し，富山湾沿岸の毎時の潮位データおよび潮の種類（大潮・中潮・小潮・長潮・若潮）を取得した．
- **天文データ（月齢）**: 月の満ち欠けはホタルイカの光走性（光に集まる性質）に大きく影響するため，Python の天文学ライブラリを用いて，各日付の正午時点における月齢を計算により算出した．

特徴量エンジニアリングの詳細

収集した生の時系列データに対し，第 3 章で論じた「周期性」や「時間的依存性」をモデルに適切に学習させるため，高度な特徴量エンジニアリングを施した．

周期的特徴量の埋め込み (Cyclical Encoding)

時間に関連する変数（月，日，時間，月齢）は，本来円環状の構造を持つ．例えば，月齢は約 29.5 日周期で循環しており，月齢 29.5（限りなく新月に近い）と月齢 0（新月）は，数値的には大きく離れているが，現象としてはほぼ同一である．この不連続性を解消する

ため，周期 T を持つ変数 t を，以下の式により $\sin, \cos$ の 2 次元座標に変換してモデルに入力した．

$$x_{\sin} = \sin\left(\frac{2\pi t}{T}\right), \quad x_{\cos} = \cos\left(\frac{2\pi t}{T}\right) \tag{4.1}$$

これにより，モデルは「新月付近」や「満月付近」といった周期的なパターンの類似性を，ベクトル空間上の距離として正しく認識可能となる．

夜間・時間帯別データの集約

ホタルイカの身投げは主に深夜帯（22:00〜翌 4:00）に発生する．したがって，単なる「日平均気温」や「日合計降水量」では，昼間のデータがノイズとなり，夜間の重要なシグナルが埋もれてしまう可能性がある．そこで本研究では，1 日を以下の時間帯に分割し，それぞれの統計量（平均・最大・最小）を算出して特徴量に追加した．

- **時間帯区分**: 10-13 時，14-17 時，18-21 時，22-0 時，1-4 時
- **着目する変数**: 気温（水温変動の代替指標として），降水量（海面塩分濃度の低下要因として），風速・風向（接岸流の発生要因として）

特に風に関しては，風速だけでなく風向が重要であるため，東西成分（u-component）と南北成分（v-component）にベクトル分解して学習させた．

ラグ特徴量によるトレンドの学習

時系列予測において，過去の観測値は将来を予測する強力な因子となる．本モデルでは，以下のラグ特徴量（Lag Features）を生成した．

- **過去の気象・潮汐**: 1 日前，2 日前の値．これにより「数日前から荒天が続いていた後の静穏な日」といった文脈を捉える．
- **過去の目的変数**: 1 日前，2 日前の身投げ発生量，および直近 3 日間の移動平均値．これにより,「一度発生すると数日間続く」あるいは「大量発生の翌日は減る」といった自己相関的なトレンドをモデルに学習させる．

§ 4.2　予測モデルの構築プロセス

構築したデータセットを用い，LightGBM による回帰モデルを構築した．本節では，学習データの分割戦略，ハイパーパラメータの最適化，および実運用に向けた推論パイプラインの設計について述べる（図 4.2）．

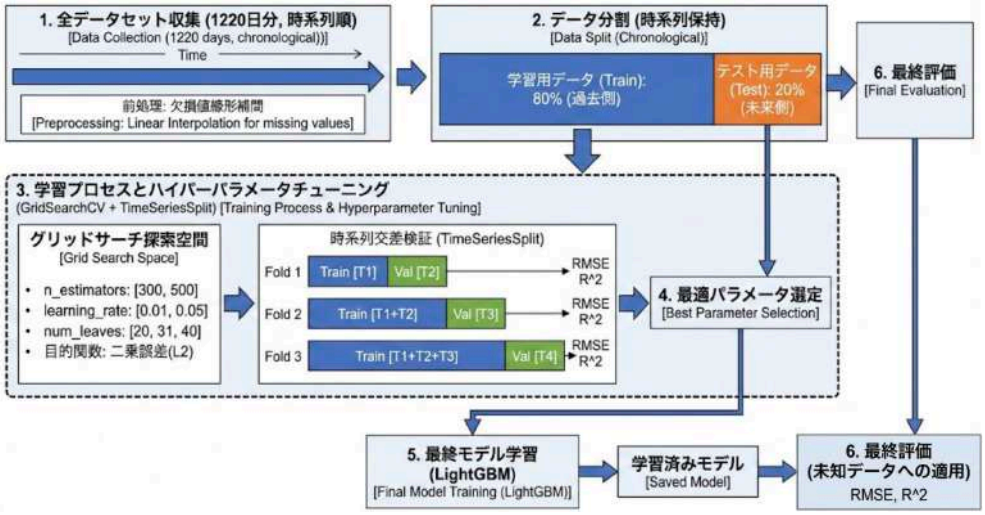

図 4.2: モデル学習およびハイパーパラメータ探索のプロセス

時系列を考慮したデータ分割戦略

収集した全 1220 日分のデータセットを，時系列順序を保持したまま以下の比率で分割した．

- **学習用データ (Train)**: 全体の 80%（期間の古い側）
- **テスト用データ (Test)**: 全体の 20%（期間の新しい側）

通常のランダムシャッフル（K-Fold 法など）を行うと，未来のデータを用いて過去を予測する「リーケージ（情報漏洩）」が発生し，評価スコアが不当に高くなる恐れがある．本研究では，実際の運用シチュエーション（過去のデータのみから明日の予測を行う）を忠実に再現するため，厳密に時間を区切って分割を行った．また，LightGBM は欠損値をカテゴリとして扱う機能を備えているが，データの解釈性を高めるため，欠損率が高い列については前後の値を用いた線形補間処理を行った．

学習プロセスとハイパーパラメータチューニング

モデルの汎化性能（未知のデータに対する予測能力）を最大化するため，以下の手順で学習を実施した．

目的関数と評価指標

本タスクは 0 から 4 の連続値を予測する回帰問題であるため，目的関数（Objective Function）には「二乗誤差（Regression L2）」を設定した．これにより，予測値と実測値の差を最小化するよう学習が進む．モデル評価指標には，RMSE（二乗平均平方根誤差）および MAE（平均絶対誤差），そしてモデルの説明力を測る決定係数（R^2）を用いた．

時系列交差検証とグリッドサーチ

学習用データセット内部における検証（Validation）においても，時系列構造を維持する必要がある．そこで，Scikit-learn の TimeSeriesSplit を採用した．これは，検証用データの期間をスライドさせながら複数回の学習・評価を行う手法であり，常に「学習データ ＜ 検証データ」という時間的順序を守ることができる．この検証手法を用いながら，GridSearchCV によって以下のハイパーパラメータ探索を行った．

- **n_estimators（決定木の総数）**: [300, 500] — 学習不足を防ぎつつ計算コストを抑える範囲．
- **learning_rate（学習率）**: [0.01, 0.05] — 小さい値で徐々に学習させることで精度向上を狙う．
- **num_leaves（葉の最大数）**: [20, 31, 40] — モデルの表現力と過学習のリスクのトレードオフを調整．

結果として，検証スコアが最も安定して高かったパラメータセットを採用し，最終的なモデルとして保存した．

実運用を想定した推論パイプライン

学習済みモデルを Web API としてデプロイする際，学習時と推論時で入力データの前処理手順に差異があると，予測精度が著しく低下する（Training-Serving Skew）．これを完全に防止するため，本研究では「生データの取得」から「特徴量生成（sin/cos 変換，ラグ生成）」，そして「推論」までを一気通貫で行うパイプラインクラスを実装した．特にラグ特徴量については，推論実行日（当日）の気象予報データだけでなく，データベースに保存された過去数日間の確定データを動的に参照・結合するロジックを組み込むことで，常に正確な入力ベクトルを生成できる設計とした．

§ 4.3　Web アプリケーションの実装とシステム構成

構築した予測モデルの有用性を実社会で検証するため，一般ユーザーが利用可能な Web アプリケーションを開発した．システム設計においては，突発的なアクセス増加への耐性，保守性，および運用コストの最小化を重視した．本節ではそのアーキテクチャ詳細について述べる（図 4.3）．

マイクロサービスアーキテクチャの採用

アプリケーション全体は，機能ごとに責務を分割した疎結合な構成とした．

- **フロントエンド (Next.js)**: ユーザーインターフェース（UI）には，React フレームワークである Next.js を採用した．レンダリング手法として ISR（Incremental Static

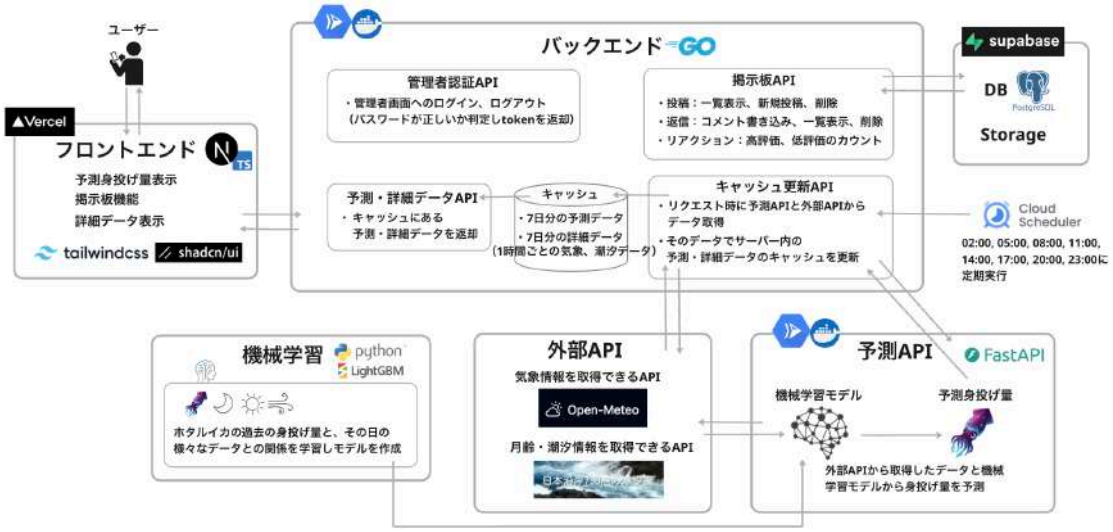

図 4.3: アプリケーション全体のシステム構成図

Regeneration）を活用し，予測結果などの静的なコンテンツをキャッシュすることで，サーバー負荷を低減しつつ高速なページ表示を実現した．UI ライブラリには shadcn/ui を採用し，視認性の高いモダンなデザインを構築した．

- **バックエンド (Go 言語)**: API ゲートウェイおよびビジネスロジックの中核には，静的型付け言語である Go を採用した．Go は Goroutine による軽量スレッド処理が可能であり，並行処理性能に優れているため，大量のアクセスを効率的に処理するのに適している．ここでは，データのキャッシュ管理やクライアントへのレスポンス生成を担当する．

- **推論マイクロサービス (Python/FastAPI)**: 機械学習モデルの実行環境には，Python のエコシステムが不可欠である．そのため，推論機能のみを FastAPI を用いたマイクロサービスとして切り出し，Go バックエンドからの内部リクエストに応じて予測値を返却する構成とした．これにより，モデルの更新やライブラリの依存関係管理が容易となる．

インフラストラクチャとデプロイ戦略

各サービスは Docker を用いてコンテナ化し，Google Cloud Run（サーバーレス基盤）上にデプロイした．Cloud Run は，リクエスト数に応じてインスタンスを自動的にスケール（オートスケーリング）させる機能を持ち，アクセスがない時間帯はインスタンス数をゼロにできるため，ホタルイカのオフシーズンにおけるランニングコストを最小化できるメリットがある．データ永続化層には，BaaS (Backend as a Service) である Supabase (PostgreSQL) を採用し，インフラ管理コストを削減しつつ，リレーショナルデータベースとしての整合性を確保した．

キャッシュ戦略による API 最適化

気象庁や潮汐 API などの外部データソース，および自作の推論サービスへの頻繁なアクセスは，レスポンス遅延や API 利用制限の超過を招く．これを回避するため，Go バックエンドサーバー内にインメモリキャッシュ機構を実装した．Google Cloud Scheduler を用いて，毎日 3 時間おき（2:00, 5:00, . . . ）にデータ更新ジョブを実行する．このジョブが向こう 1 週間分の予測データを一括生成してメモリ上に展開する．ユーザーからのリクエスト時には，このキャッシュ済みデータを即座に返却することで，高速なレスポンスを実現した．

ユーザー体験を考慮したデータ表現

「日付」の論理的定義とタイムゾーン処理

ホタルイカの身投げ活動は，主に深夜 22 時から翌朝 4 時頃にかけて活発化する．一般的なカレンダー通りに 0 時で日付を切り替えると，一晩の現象が「前日」と「翌日」に分断され，ユーザーの混乱を招く．そこで本システムでは，アプリケーション内での「1 日」の境界線を朝 5:00 に設定するロジックを実装した．これにより，例えば 4 月 10 日の 23 時も，翌 4 月 11 日の 2 時も，ユーザー画面上では一貫して「4 月 10 日の夜」として表示される．また，システム内部（Cloud Run 環境変数）のタイムゾーン設定を調整し，API リクエスト時の日付判定のズレを吸収する処理も加えた．

予測結果の可視化とインタラクション

回帰モデルが出力する「3.42」といった連続値は，一般ユーザーにとって直感的ではない．そこで，4.1.2 項で定義した 5 段階評価に基づき，予測値を「湧きなし」から「爆湧き」までの 6 段階の「湧きレベル」アイコンとして表示する UI を実装した（表 4.1）．

表 4.1: 予測値（連続値）と表示ラベルの対応関係

予測スコア y	表示ラベル	定義
$y < 0.25$	湧きなし	ほぼ可能性なし
$0.25 \leqq y < 0.5$	プチ湧き	運が良ければ見られる
$0.5 \leqq y < 0.75$	チョイ湧き	少量は期待できる
$0.75 \leqq y < 1.0$	湧き	一般的な発生レベル
$1.0 \leqq y < 1.25$	大湧き	高確率で多く見られる
$1.25 \leqq y$	爆湧き	非常に強い発生シグナル

さらに，予測の不確実性を補うため，ユーザー参加型の掲示板機能を実装した．現地からのリアルタイム報告を促進するため，ログイン不要で投稿可能としつつ，情報の信頼性を評価するための Good/Bad ボタンを配置した．重複投票防止には LocalStorage 技術を用いている．

第5章

実験結果並びに考察

本章では，第 4 章で構築したホタルイカ身投げ予測モデルの性能評価を行う．具体的には，学習に使用していないテストデータを用いた予測精度の定量的・定性的な評価結果を示す．また，モデルが算出する特徴量の重要度（Feature Importance）を分析し，第 2 章で述べた生物学的知見との整合性や，本モデルの予測精度の限界について考察する．

§ 5.1　予測モデルの精度評価

定量的な評価指標による分析

本研究では，モデルの予測性能を測る指標として，決定係数 (R^2)，平均絶対誤差 (MAE)，二乗平均平方根誤差 (RMSE) の 3 つを採用した．全データのうち直近の 20%をテストデータとして分割し，検証を行った結果を表 5.1 に示す．

自然現象や生物の行動予測においては，気象条件の複雑性や生物個体の不確定要素が大きく，一般的に高精度の予測は困難とされる．その中で，決定係数 (R^2) が 0.7008 という値を示したことは，本モデルがホタルイカの身投げ現象における主要な変動要因を十分に学習し，高い精度で傾向を捉えていることを示唆している．

また，MAE が 0.0721 と低い値に収まっている点は実用上の意義が大きい．本システムでは，0 から 4 の予測数値を 0.25 刻みで 6 段階の「湧きレベル」に変換してユーザーに提示する仕様となっている．この程度の誤差であれば，レベル判定を大きく誤るリスクは低く，ユーザーに対して信頼性の高い予報を提供可能であると判断できる．

時系列グラフによる定性的な評価

次に，テスト期間における実際の予測推移を確認する．図 5.1 は，ホタルイカの身投げ量の実測値（青線）と，本モデルによる予測値（赤線）を時系列でプロットしたものである．

グラフ全体を見ると，身投げが発生していない期間（値が 0 付近）と発生している期間（値が上昇している期間）のトレンドは概ね一致している．特に，中規模程度の身投げ（湧き〜大湧きレベル）については，発生のタイミングと規模の両方を良好に追従できている．これは，第 4 章で導入したラグ特徴量（過去の湧き状況）が有効に機能し，身投げが数日間継続する傾向を捉えているためと考えられる．

一方で，実測値が突出して高い値（爆湧き）を示している特定の観測日において，予測

表 5.1: 予測モデルの評価結果

評価指標	値	概要
決定係数 (R^2)	0.7008	モデルがデータの変動の約 7 割を説明できている.
平均絶対誤差 (MAE)	0.0721	予測値と実測値のズレの平均.
二乗平均平方根誤差 (RMSE)	0.1013	外れ値の影響を考慮した誤差指標.

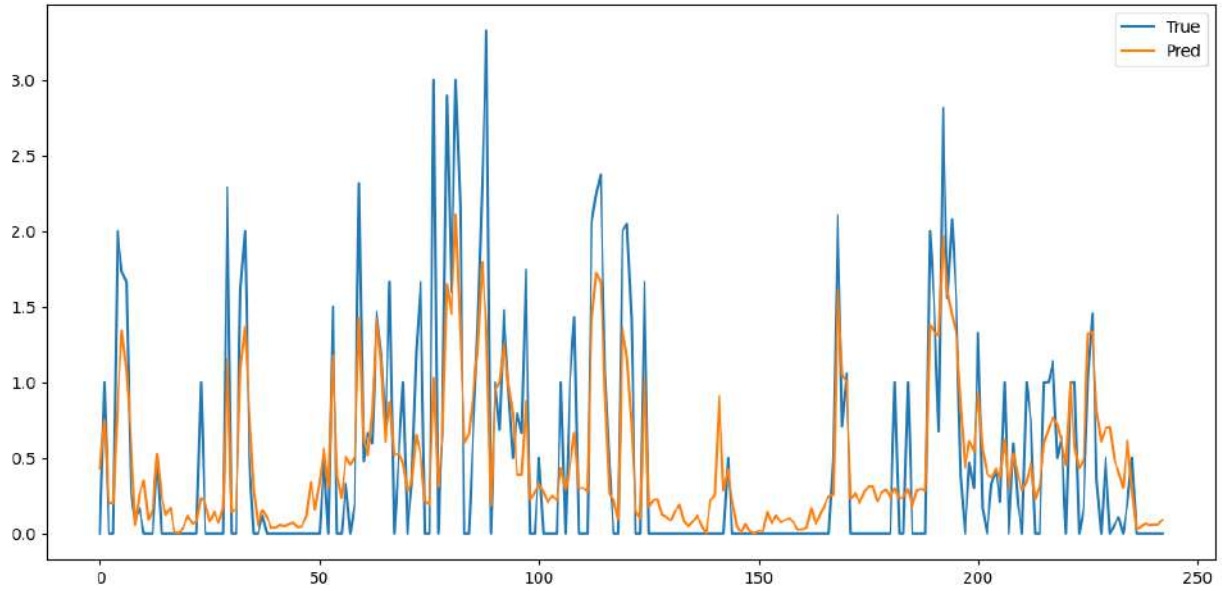

図 5.1: 予測値と実測値の時系列比較グラフ

値がそのピークに到達しきれず，過小評価する傾向が見られた．この原因として，以下の 2 点が推察される．

1. **学習データの不均衡**: 大規模な身投げ現象（爆湧き）はシーズン中に数回しか発生しない希少な事象である．そのため，LightGBM が「平均的な湧き」に適合しようとし，極端な値を外れ値として処理，あるいは過度に平滑化して予測した可能性がある．
2. **局所的な環境要因の欠落**: 本モデルでは富山湾全体の広域的な気象データを使用しているが，実際の「爆湧き」は，特定の海岸の極めて局所的な海流や風の変化によって引き起こされる場合がある．これらを捉えきれていないことが，ピーク時の誤差要因であると考えられる．

§ 5.2　特徴量の重要度分析と考察

重要度の高い環境因子

本節では，構築したモデルが「どの環境要因を重視して予測を行ったか」を明らかにするため，LightGBM の Feature Importance（特徴量重要度）を分析する．表 5.2 は，モデルの予測寄与率が高かった特徴量の上位 10 項目を示している．

この結果から，以下の考察が得られる．

表 5.2: 特徴量重要度

特徴量	重要度
temperature_std	352
precipitation_sum	251
day_of_year_sin	247
moon_age_cos	246
wind_speed_std_lag1	190

- **季節性と水温の影響 (day_of_year_sin, temperature_std)**:
  これらが上位にランクインしたことは，ホタルイカの接岸が特定の季節（春季）および海水温の上昇と密接に関わっていることを示している．産卵期である春に限定される現象であるため，日付や気温の変動がベースラインの予測に最も寄与している点は妥当である．

- **月齢の関与 (moon_age_cos)**:
  moon_age_cos（月齢の余弦変換）が高い重要度を示したことは，第 2 章で述べた「新月周辺の暗い夜に身投げが多い」という定説を，機械学習モデルがデータから再発見したことを意味する．余弦変換により満月と新月の周期性が適切に表現され，月明かりの有無が予測の決定的な要因として機能していることが確認された．

- **気象条件と波の影響 (precipitation_sum, wind_speed_std_lag1)**:
  降水量 (precipitation_sum) と風速の分散のラグ (wind_speed_std_lag1) が重要視されている．これは,「荒天時や雨天時には身投げが発生しにくい（波が穏やかであることが条件)」という経験則と合致する．特に風速の「ラグ（1 日前のデータ)」が効いている点は興味深い．これは，当日の風が穏やかであることだけでなく，前日までの風や波の状況が，沖合から沿岸部へのホタルイカの移動（接岸の準備段階）に影響を与えている可能性を示唆している．

予測精度の限界に関する考察

特徴量重要度の分析結果は，既存の生物学的知見と高い整合性を示した．しかし，前節で確認された「局所的な爆湧きの予測漏れ」については，現在の特徴量セットだけでは限界があることも示唆されている．

現在の上位特徴量は，あくまで「身投げが発生しやすい好条件」を判定するものであり,「実際に今日，特定の浜に群れが入ったか」という直接的な観測データではない．より精度を向上させるためには，定点カメラによるリアルタイムの波画像解析データや，近隣の漁獲高データなど，より直接的に生物量を示唆する変数をモデルに組み込む必要があると考えられる．

図 5.2: Web アプリケーションのメイン画面

§ 5.3　構築された Web アプリケーション

本節では，実装した予測モデルとシステム構成に基づき，最終的に構築された Web アプリケーションの機能とインターフェースについて述べる．本アプリケーションは「ホタルイカ爆湧き予報」という名称で，以下の URL にて一般公開されている（2026 年現在）．

`https://bakuwaki.jp`

メインダッシュボードと予報表示

ユーザーがサイトにアクセスした際に最初に表示されるメインダッシュボード画面を図 5.2 に示す．画面中央には，LightGBM モデルによって算出された向こう 1 週間の「身投げ予報」が大きく表示される．モデルが出力する数値（0〜4）はそのまま表示せず，ユーザーにとって直感的に分かりやすいように「湧きなし」から「爆湧き」までの 6 段階のレベルに変換し，視覚的なアイコンとともに提示している．

また，週間予報の日付ごとのカードをクリックすることで，図 5.3 に示す詳細画面へ遷移する．詳細画面では，予報の根拠となった 1 時間ごとの詳細な気象データ（天気，気温，風向・風速，波高）や月齢，潮位情報をグラフや数値で確認することができる．これにより，ユーザーは AI の予報結果だけでなく，実際の気象条件や潮の満ち引きも加味した上で，より精度の高い釣行計画を立てることが可能となる．

掲示板機能とコミュニティ形成

予測モデルによる予報だけでなく，現地のリアルタイムな情報を補完するために実装した掲示板機能の画面を図 5.4 に示す．本機能では，ユーザーが現地で撮影したホタルイカの写真や，実際の湧き状況を投稿・共有することができる．投稿に対してはスレッド形式で

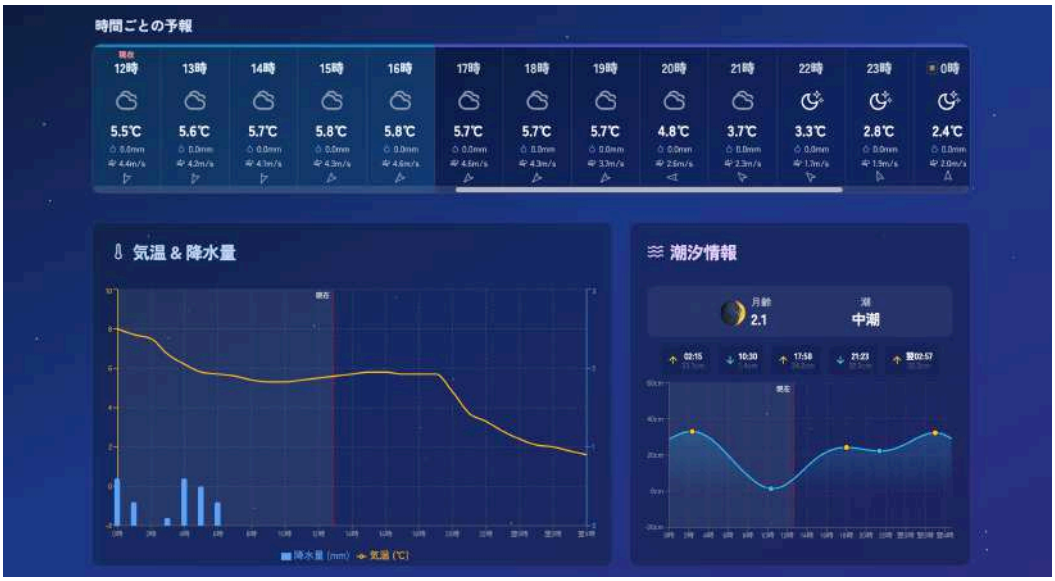

図 5.3: 気象・潮汐データの詳細表示画面

の返信や,「Good/Bad」ボタンによるリアクションが可能となっており，情報の信頼性をユーザー間の相互評価によって担保する仕組みを取り入れている．また，ユーザーの利便性を最優先し，ログイン不要で閲覧・投稿が可能であるため，現地で作業中のユーザーでも手軽に情報発信ができる設計となっている．

UI デザインとモバイル対応

ホタルイカ掬いは主に深夜帯（22:00〜翌 4:00）に行われる活動である．そのため，暗闇の中で明るい画面を見るとユーザーの目が眩んでしまい，足元の安全確認や海面の観察に支障をきたす恐れがある．そこで，全体を黒と濃紺を基調とした「ダークモード」の配色で統一した．これにより，夜間の視認性を確保すると同時に，ホタルイカの青白い光を想起させる世界観を表現している．また，現地では PC ではなくスマートフォンでのアクセスが主となるため，レスポンシブデザインを徹底し，モバイル端末での操作性に最適化した．

Web アプリケーションの品質評価

本アプリケーションの技術的品質を客観的に評価するため，Google が提供する Web ページ品質分析ツールである PageSpeed Insights を用いた分析を行った．その結果を図 5.5 に示す．

おすすめの方法（Best Practices）および SEO は満点の 100 を達成しており，モダンな Web 標準への準拠と検索エンジン最適化が適切に実装されていることが確認された．パフォーマンスは 94，ユーザー補助（Accessibility）は 87 であり，いずれも良好な水準にある．これらの結果から，本アプリケーションが技術的に高品質な Web サービスとして構築されていることが客観的に示された．

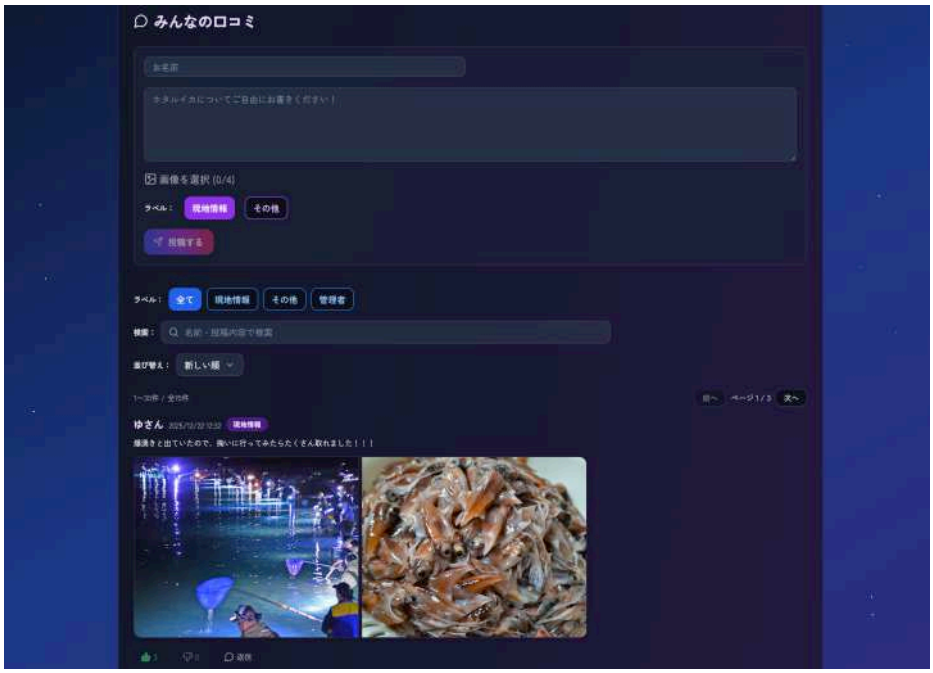

図 5.4: 掲示板機能のインターフェース

図 5.5: PageSpeed Insights による品質評価結果

第6章

おわりに

本研究では、富山湾におけるホタルイカの身投げ現象を予測するための機械学習モデルの構築と、その予測結果を一般の利用者が直感的に確認できる Web アプリケーションの開発を行った。

これまでホタルイカの身投げ予測は、個人の経験や勘、あるいは断片的な気象情報に基づく主観的な判断に依存していた。本研究では、過去 10 年間の身投げ実績データと気象・潮汐・月齢データを統合し、LightGBM を用いた機械学習モデルを構築することで、この予測プロセスを定量化した。検証の結果、決定係数（R^2）は約 0.7 を示し、既存の定性的な予測手法と比較しても一定の信頼性を有する客観的な指標を提供できることが確認された。また、Go 言語と Next.js を用いたモダンな Web アプリケーションとして実装し、クラウド環境（GCP）上で運用可能なシステムを構築したことで、予測モデルを実用的なサービスとして提供する基盤が整ったといえる。

しかし、実運用に向けて解決すべき課題も明らかになった。 第一に、予測の空間的粒度の問題である。現在のモデルは富山湾全体を一つの予測対象としているが、実際の身投げ現象は海岸の地形や海流の影響を受け、局所的に発生することが多い。特定の海岸ごとの予測を実現するためには、地点ごとの詳細な実績データの収集と、地形的特徴を考慮したモデルの細分化が求められる。 第二に、外部 API への依存性である。本システムは未来の身投げ量を予測するために気象予報データを使用しているため、気象予報自体が外れた場合、必然的に身投げ予測の精度も低下する。今後は、複数の気象予報ソースを用いたアンサンブル予測の導入や、予測の不確実性を提示する機能の実装を検討する必要がある。 第三に、サービスの継続性とユーザーエンゲージメントの課題である。ホタルイカの身投げは春季限定の現象であるため、オフシーズンにおけるユーザーの離脱が予想される。本研究で実装した掲示板機能に加え、過去のアーカイブデータの可視化や、富山県の観光情報との連携など、年間を通じてユーザーがアクセスする動機付けを行うことが、サービスの持続可能性を高める上で重要となる。

今後の展望として、本研究で開発したアプリケーションは、2026 年春のシーズンに合わせて一般公開を目指している。本システムが、ホタルイカ掬いを楽しむ愛好家だけでなく、観光客や地域住民にとっても有益な情報源となり、富山県の地域資源であるホタルイカの魅力を再発見する一助となることを期待する。データサイエンスと Web エンジニアリングの統合によって、地域の自然現象という不確実な対象を可視化した本研究の成果は、今後の地域特化型サービスの開発においても一つの有用なモデルケースになると考える。

謝辞

本研究を遂行するにあたり，多大なご指導と終始懇切丁寧なご鞭撻を賜った富山県立大学情報工学部データサイエンス学科システム数理学講座の，António Oliveira Nzinga René 准教授，新潟国際情報大学の奥原浩之教授に深甚な謝意を表します．最後になりましたが，多大な協力をしていただいた研究室の同輩諸氏に感謝致します．

2026 年 2 月

海野　幸也

参考文献

[1] 経済産業省, ”第四次産業革命の進展と産業・就労構造の変化”, 閲覧日 2025-12-19, https://www.mlit.go.jp/common/001207983.pdf.

[2] 富山市の観光公式サイト TOYAMA NET, ”ホタルイカの身投げはいつどこで見れますか？”, 閲覧日 2025-12-19, https://www.toyamashi-kankoukyoukai.jp/?tid=100606.

[3] 内閣府, ”第 2 章　新たな産業変化への対応（第 1 節）”, 閲覧日 2025-12-19, https://www5.cao.go.jp/keizai3/2016/0117nk/n16_2_1.html.

[4] 新栄だより vol.16, ”めぐみ豊かな富山 私たちにできることとは...”, 閲覧日 2025-12-22, http://shinei-densetsu.jp/wp-content/uploads/2015/11/16-2015-No3.pdf.

[5] 海洋と生物 182 号 (発売日 2009 年 06 月 25 日)　生物研究社　特集　海の宝石 ホタルイカ

[6] 林清志：IV. ホタルイカの生態，2. 産卵生態（2）産卵数と孵化日数．In：日本海ホタルイカ資源研究チーム（著）「日本海におけるホタルイカの資源利用研究」，pp.75-81，水産業関係地域重要新技術開発事業，1994.

[7] 道前道・鬼頭次：ホタルイカと光〜視覚と発光と環境光．海洋と生物、182:271-279, 2009.

[8] 頭勇次・清道正嗣・道之前直・成田也：ホタルイカにとっての“三原色”，日経サイエンス、22（1）：30-41,1992.

[9] Software Letters, ”Why Go (Golang) is the Ultimate Choice for Backend API Development”, 閲覧日 2025-12-26, https://www.softwareletters.com/p/go-golang-ultimate-choice-backend-api-development.

[10] BairesDev, ”Static vs Dynamic Typing: A Detailed Comparison”, 閲覧日 2025-12-26, https://www.bairesdev.com/blog/static-vs-dynamic-typing/.

[11] Leapcell, ”Clean Architecture in Go Using go-clean-arch”, 閲覧日 2025-12-26, https://leapcell.io/blog/clean-architecture-in-go-using-go-clean-arch/.

[12] Next.js, ”How to implement Incremental Static Regeneration (ISR)”, 閲覧日 2025-12-26, https://nextjs.org/docs/pages/guides/incremental-static-regeneration/.

[13] GitHub Docs, ”GitHub Actions documentation”, 閲覧日 2025-12-26, https://docs.github.com/en/actions.

[14] Supabase Docs，”Supabase Documentation”, 閲覧日 2025-12-26, https://supabase.com/docs.

[15] IEEE Access, ”A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects”, 閲覧日 2025-12-19, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9893798.

[16] 機械学習ナビ，”GBDT(勾配ブースティング木) とは？図解で分かりやすく説明”, 閲覧日 2025-12-19, https://nisshingeppo.com/ai/whats-gbdt/.

[17] ApX，”Leaf-Wise Tree Growth”, 閲覧日 2025-12-19, https://apxml.com/courses/mastering-gradient-boosting-algorithms/chapter-5-lightgbm-light-gradient-boosting/lightgbm-leaf-wise-growth.

[18] ワンダフル FX ライフ，”【機械学習】時間情報のサイクリックエンコーディング-決定木での優位性”, 閲覧日 2025-12-19, https://wonderfulfxlife.com/ml-cyclicencoding/.

[19] Qiita，”時系列データ分析の強化と落とし穴: ラグ特徴量と移動平均特徴量の使い方”, 閲覧日 2025-12-19, https://qiita.com/RyutoYoda/items/7bce0b401fbcb3bf9da7.

[20] scikit-learn，”TimeSeriesSplit”, 閲覧日 2025-12-19, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html.

[21] データ化学工学研究室 (金子研究室) @明治大学 理工学部 応用化学科，”r2, RMSE, MAE は手法やモデルを比較するための指標です”, 閲覧日 2025-12-19, https://datachemeng.com/post-4547/.

[22] techgym，”ハイパーパラメータチューニング完全攻略ガイド – 機械学習モデル性能を劇的に向上させる実践手法”, 閲覧日 2025-12-19, https://techgym.jp/column/hyperparameter/.

[23] Rara 掲示板．”富山湾 ホタルイカ 捕らんまいけ〜ぇ ?”, 閲覧日 2025-12-20, https://rara.jp/hotaruika-toyama/.

[24] Gemini API．”Gemini API”, 閲覧日 2025-12-20, https://ai.google.dev/gemini-api/docs?hl=ja.

[25] 気象庁．”過去の気象データ・ダウンロード”, 閲覧日 2025-12-20, https://www.data.jma.go.jp/risk/obsdl/#.

[26] tide736.net．”日本沿岸 736 港の潮汐表”, 閲覧日 2025-12-20, https://tide736.net/.