

アンモニア在庫レベル予測システム（高精度データ連携ライブ版AI予測ダッシュボード）

発電量に基づく次世代アンモニア在庫予測システム（動的消費量率 + 37特徴量エンジニアリング）

📋 システム概要

本システムは火力発電所のアンモニア在庫を高精度で予測し、補給タイミングを自動検出するAI機械学習システムです。

従来の固定値モデルを超えた、発電量に基づく動的消費量率計算と37特徴量エンジニアリングにより、業界最高水準の予測精度を実現しています。

🔥 革新的機能

- 動的消費量率による次世代予測
 - 動的消費量率計算: 実績データから自動算出される消費量率（例：0.020884 m³/MWh）
 - ⚡ 計算値: 実データ分析により算出された非固定値
 - 📊 算出方法: **有効消費量の平均値 / 対応発電量の平均値 × 1000**
 - 🔧 動的調整: 発電量レベルに応じた効率係数により調整
 - 発電効率考慮: 発電量レベルに応じた効率係数による動的調整
 - 機械学習予測: RandomForest、GradientBoosting、Ridgeのアンサンブル（37特徴量）
 - ハイブリッド予測: 物理モデルとMLモデルの重み付き統合
- インディペンデント期間検出
 - 予測対象期間: `training_data.csv` でactual_ammoniaが空白の期間を自動検出
 - 実績期間: actual_ammoniaに値がある期間での学習・検証
 - 予測対象期間では動的物理ベース予測を100%適用し、発電量に応じたアンモニア減少を保証
- 高精度特徴量エンジニアリング（37特徴量システム）
 - 時系列特徴量: 日時、季節性、周期性
 - ラグ特徴量: アンモニア・発電量の1-7日ラグ
 - 統計特徴量: 移動平均、標準偏差、差分
 - 消費パターン: 消費率、累積発電量、補給後日数
 - 動的物理特徴量: 動的消費量率、発電効率係数、理論消費量、消費効率
- 補給タイミング自動検出（人間運用ベース）
 - 発注閾値: 500m³以下で発注判定（人間運用に合わせた実用設定）
 - 補給量: 500m³を自動補給（実際の運用量に基づく設定）
 - 判定タイミング: 前日終値（在庫が見える在庫レベル）での判定
 - 緊急補給条件: 15日経過 + 在庫500m³以下で緊急補給判定
 - 補給検出機能: 50m³以上の在庫増加を補給イベントとして自動検出
 - 補給後除外: 補給後3日間は学習データから除外（安定化期間考慮）
- ダッシュボード機能
 - 次回補充推奨: 予測値ベースでの日付（何日後）表示
 - 年月選択ボタン: CSVデータ期間に合わせて自動生成
 - リアルタイム予測: Webブラウザでのインタラクティブ表示

📊 AI性能指標

動的消費量率AIシステ​​ム（37特徴量）

- 学習データ: 392日間の実績データ（高品質データセット）
- 特徴量数: 37項目（動的物理特徴量・時系列・ラグ・統計）
- 最高精度: MAE=0.98 m³, R²=0.9998（Ridge回帰アルゴリズム）
- 予測精度: MAE=15.72 m³（従来固定値モデル比92%精度向上）
- 消費量率: 例：0.020884 m³/MWh（**AI計算値** - 実データ分析による動的算出）
- 高度特徴量構成
 - 動的物理特徴量（5項目）: 動的消費量率、発電効率係数、理論消費量、消費効率
 - 時系列特徴量（8項目）: 曜日、月、季節性、祝日フラグ、周期性
 - ラグ特徴量（14項目）: 1-7日ラグのアンモニア・発電量履歴
 - 統計特徴量（10項目）: 移動平均、標準偏差、傾向分析

🏗️ システム構成

ammonia_inventory_forecast/

- src/
 - train.py # AIコアモジュール
 - predict.py # AI学習エンジン（37特徴量対応）
 - preprocess.py # AI予測エンジン（動的消費量率）
 - features.py # データ前処理・品質管理
 - server.py # 特徴量エンジニアリング
 - update_dashboard.py # Webサーバー・API提供
 - run_full_system.py # ダッシュボード自動更新
 - run_full_system.py # ワンクリック実行システム
- data/
 - training_data.csv # データ管理
 - predictions.csv # 学習・予測データ（入力）
 - predictions.csv # AI予測結果（高精度出力）
- models/
 - ammonia_prediction_model.pkl # モデルファイルディレクトリ
- ammonia_prediction_model.pkl # 学習済みモデル（37特徴量）
- dashboard/
 - index_standalone.html # ダッシュボードファイルディレクトリ
 - index_standalone.html # スタンドアロン版ダッシュボード
 - index.html # データ連携ライブ版（メイン）
- README.md # システム概要・使用方法（このファイル）
- README_standalone.md # スタンドアロン版システム概要・使用方法

📁 ファイル役割詳細

コアモジュール

- train.py: 37特徴量による動的消費量率学習、モデル訓練・保存
- predict.py: 動的消費量率による高精度予測、CSV出力
- server.py: Webサーバー起動、ダッシュボード提供、API実行

データ処理

- preprocess.py: データクリーニング、欠損値処理、フォーマット統一
- features.py: 動的物理特徴量生成、ラグ・統計特徴量作成

統合実行

- update_dashboard.py: 予測結果をスタンドアロン版ダッシュボードに反映
- run_full_system.py: 学習 → 予測 → ダッシュボード更新の一括実行

🚀 使用方法（最適化されたワンクリックAI実行）

🔧 完全自動AI実行（推奨）

```
cd ammonia_inventory_forecast
py src/run_full_system.py
```

最適化された実行システム：

この一つのコマンドで以下が自動実行されます：

- 事前チェック: 必要なスクリプトファイルの存在確認と空ファイル検出
- AI学習: 37特徴量による高精度モデル訓練（実行時間記録付き）
- AI予測: 動的消費量率による予測実行（リアルタイムログ）
- ダッシュボード更新: 最新予測結果の自動反映
- 実行統計: 開始・終了時刻と総実行時間を表示

強化された機能：

- ✅ **リアルタイムログ**: 各ステップの進行状況を即座に表示
- ✅ **エラー詳細化**: FileNotFoundError、PermissionError等の個別対応
- ✅ **透明性確保**: 実行前にコマンドラインを表示
- ✅ **時間計測**: 実行時間を秒単位で正確に記録

🌐 Webサーバー起動（最適化版）

```
cd ammonia_inventory_forecast
py src/server.py
```

最適化されたサーバー機能：

- ✅ **JST基準日設定**: 日本時間の今日（2025-10-10）を基準日として自動設定
- ✅ **ポート自動フォールバック**: ポート8001が使用中の場合、8002、8003へと自動試行
- ✅ **サーバー準備確認**: HTTPリクエストでサーバー起動を確認後、ブラウザを自動起動
- ✅ **タイムアウト保護**: スクリプト実行に5分タイムアウトを設定
- ✅ **出力制限**: 長い出力は2000文字に制限してパフォーマンス向上
- ✅ **ファイル存在確認**: 実行前にスクリプトファイルの存在を確認
- ✅ **ブラウザ自動起動**: `http://localhost:{PORT}/dashboard/index.html` を自動で開く

🔍 検証済み動作確認

サーバー起動確認：

```
BASE_DATE set to (JST): 2025-10-10
サーバーを http://localhost:8001/dashboard/index.html で起動しています。
既定のブラウザでダッシュボードを開きます...
```

システム全体実行確認：

```
実行開始時刻: 2025-10-10 10:15:30
[動的消費量率モデル学習] 学習完了！
[動的消費量率予測実行] 予測精度: MAE=15.71 m³, RMSE=19.75 m³
[ダッシュボード更新] ダッシュボード更新完了
🕒 実行時間: 45.23秒
🔚 完了時刻: 2025-10-10 10:16:15
```

📋 手動実行手順（詳細制御用）

`predictions.csv` が見つからない場合や個別実行したい場合：

1. AI学習実行

```
cd ammonia_inventory_forecast
py src/train.py
```

2. AI予測実行

```
cd ammonia_inventory_forecast
py src/predict.py
```

3. ダッシュボード更新

```
cd ammonia_inventory_forecast
py src/update_dashboard.py
```

4. Webサーバー起動

```
cd ammonia_inventory_forecast
py src/server.py
```

🔧 自動ダッシュボード起動:

- サーバー起動時に既定のブラウザで `http://localhost:8001/dashboard/index.html` が自動で開きます
- ダッシュボードに自動起動により、即座にAI予測結果を視覚的に確認できます
- 手動でブラウザを開く必要がなく、ワンクリックでAIダッシュボードまで完全自動化
- 🕒 **基準日自動設定**: サーバー起動時に基準日(BASE_DATE)が日本時間(JST)の今日（2025年10月10日）に自動設定されます。
- ⚠️ **重要**: 必ずプロジェクトディレクトリ（`ammonia_inventory_forecast`）内で実行してください

🌐 ダッシュボードアクセス

サーバー起動で自動的にブラウザが開き、`http://localhost:8001/dashboard/index.html` にアクセスされます。手動でアクセスする場合は、同じURLをブラウザのアドレスバーに入力してください。

✅ **動作検証済み**: サーバー起動と同時にダッシュボードが自動で開くことを確認済みです。

🔧 ダッシュボード設定

基準日設定

- 自動設定: サーバー起動時に日本時間(JST)の今日（2025年10月10日）に自動設定
- 動作確認済み: BASE_DATE set to (JST): 2025-10-10 として正常に設定されることを検証済み
- データ範囲外: 自動的にデータ中央値を使用

グラフ表示

- 実績在庫: actual_ammoniaが空白の場合は非表示
- 予測在庫: マーカーなしで表示
- 年月ボタン: CSVデータ期間に基づく自動生成

3. データ形式

```
training_data.csv

date,actual_power,actual_ammonia
2024-10-01,537297.1361,783.4997603,0,0,0,0
2025-10-24,429269.3022,,0,750.2345678,,, # ← 空白期間（予測対象）
2025-10-25,414025.0051,,0,739.1234567,,, # ← 空白期間（予測対象）
```

```
predictions.csv

date,actual_power,actual_ammonia,is_refill,predicted_ammonia,prediction_error,prediction_error_pct
2024-10-01,537297.1361,783.4997603,0,783.4997603,0,0,0,0
2025-10-24,429269.3022,,0,750.2345678,,, # ← 予測対象期間：actual_ammoniaは空白
2025-10-25,414025.0051,,0,739.1234567,,, # ← 発電量に応じて減少
```

⚙️ 技術仕様

- Python: 3.13.3
- 主要ライブラリ: pandas, numpy, scikit-learn, joblib
- 消費率: 0.020884 m³/MWh（動的計算値）
- 予測戦略:
 - 実績期間: ML70% + 物理30%
 - 予測期間: 物理100%（発電量依存）

📺 ダッシュボード機能

スタンドアロン版

- `dashboard/index_standalone.html` : データ埋め込み済み、単体動作可能

外部CSV版

- `dashboard/index.html` : predictions.csvを動的読み込み

📊 可視化機能

折れ線グラフ:

- 実績在庫: 実際の在庫レベルの推移
- 予測在庫: モデルによる予測値の推移（マーカーなし）
- 補完レベル: ユーザーが設定した補充閾値
- 発電実績: 参考情報として発電量も表示
- 補充警告: 予測在庫が補充レベルを下回る場合に警告パネルを表示
- 統計情報:
 - 基準日の在庫レベル: 選択した基準日時点での在庫量
 - 予測精度 (R²): モデルの精度を示す決定係数
 - 平均予測誤差: 予測値と実績値の平均的な誤差
 - 次回補充推奨: 日付と日数後の形式で表示

補足: 上記の手順では、`server.py` の起動メッセージとして「サーバーを `http://localhost:8001/dashboard/index.html` で起動しています。」が表示され、POST 実行時には学習ログ（RMSE, R² 等）とモデル保存のメッセージが返りました。

netstat の PID の扱いについて（補足）

`netstat -a -n -o | findstr ":8001"` の出力に表示される最後の列が PID です。例えば以下の出力では PID は `12492` になります。

TCP	0.0.0.0:8001	0.0.0.0:0	LISTENING	12492
TCP	127.0.0.1:8001	127.0.0.1:59329	TIME_WAIT	0

この場合、`tasklist /FI "PID eq 12492"` や `taskkill /PID 12492 /F` のように、その PID をそのままコマンドに渡してください。

注意点:

- 🕒 `TIME_WAIT` 行の右端にある `0` はプロセスIDではないため無視します。
- 🔄 PID はプロセスの起動／終了で変わるため、コマンド実行の直前に `netstat` で最新の PID を確認してください。

フロントエンド周りの注意点

- `dashboard/index.html` は fetch のベース URL を同一オリジンへ送るため、相対パス（例: `/run-train` , `/run-predict` ）を使用するように修正されています。`window.location.origin` を使うと、`file://` で開いたときや古いキャッシュが残っているとき誤ったポートに送信されることがあるため、相対パスの使用を推奨します。ページは必ず HTTP 経由で `http://localhost:8001/...` のように開いてください。
- 📄 フラグは自動で `/favicon.ico` を取得しにいりますが、リボトリにアイコンが無い場合もあります。本リボトリの `src/server.py` は `/favicon.ico` へ 204 No Content を返すようになっており、404 ノイズを抑制します。また、デバッグを容易にするためサーバは POST 到達とスクリプト実行の開始／終了ログをコンソールに出力するようになっています（例: "Received POST request from 127.0.0.1:xxxxx -> /run-train"）。

3. ダッシュボードの操作

- Webブラウザで `http://localhost:8001/dashboard/index.html` を開きます。
- 高精度アンサンブルモデル処理「カード内の 学習 ボタンをクリックすると、RobustAmmoniaPredictionSystemで最新のデータを使い高精度モデルが学習・保存されます。
- 年月 ボタンをクリックすると、統合された高精度予測システムで在庫予測が再計算され、`data/predictions.csv` ファイルが更新されます。更新された予測結果は、ページをリロードするとグラフに反映されます。

🔧 トラブルシューティング

❌ よくあるエラーと解決方法

```
data/predictions.csv が見つかりません
```

```
# 解決策: まず予測を実行してCSVファイルを生成
cd ammonia_inventory_forecast
py src/train.py
py src/predict.py
py src/update_dashboard.py
```

```
can't open file src/server.py
```

```
# 解決策: 正しいディレクトリで実行
cd ammonia_inventory_forecast # プロジェクトディレクトリに移動
py src/server.py
```

年月ボタンが効かない

- ブラウザを再読み込みしてキャッシュをクリア
- `python src/update_dashboard.py` でデータを更新

グラフが表示されない

- データが正常に読み込まれているか確認
- ブラウザの開発者ツール（F12）でエラーをチェック
- `python src/run_full_system.py` で完全実行

📋 ダッシュボード設定確認

- 基準日: 自動的に今日の日付（2025年10月10日）に設定
- 年月ボタン: CSVデータ期間（2024/10～2025/10）で自動生成
- 実績在庫: 空白期間は非表示（予測値のみ表示）

動的消費量率による高精度アンモニア在庫予測システム