体調予測アプリ：最終設計ドキュメント

> **バージョン**: v2.0（Claude・ChatGPT レビュー反映済） **作成日**: 2026-02-06 **ステータス**: MVP設計確定 → 実装フェーズへ移行

---

1. アプリの目的

日々の生活データ（睡眠・歩数・ストレス・体調）から、**「今日」の体調低下リスクを予測する**個人向けアプリを開発する。

ユーザーが不調の兆候を事前に把握し、以下の意思決定をサポートする：

- 無理な予定を避ける
- 生活リズムを調整する

> **変更点**: 元の設計では「今日」と「今後3日」を同時に提供する予定だったが、MVP段階では**今日のリスクのみ**を開放する。3日リスクはUIを用意するが、開放は段階的に行う（後述）。

---

2. ターゲットユーザー

- 体調の波に悩んでいる人
- コンディション管理をしたい人
- 日々の体調を記録したい人

---

3. 提供する価値

1. **個人専用の体調予測** — 自分のデータから学習した予測モデル
2. **不調の"予報"による行動調整** — 予測結果に基づく事前対策
3. **体調データの蓄積と振り返り** — 記録の可視化・傾向把握

---

4. 予測の基本設計

出力

予測種別	内容	MVP開放条件
今日のリスク	今日の不調確率（%）	体調入力14日分
3日リスク	今後72時間以内に1回でも不調になる確率（%）	体調入力60日以上 AND 不調10件以上

目的変数

2値分類：不調フラグ（0 or 1）

不調の定義

- 過去14日の体調平均と比較
- **「平均より1段階以上低い日」を不調とする**

14日未満の期間（初期ルール）

- `daysCollected < 14` の期間は `y_today` を生成しない（学習行を作らない）
- UIは「学習中」のみ表示
- 14日分のデータが溜まった時点で初めて不調ラベルの生成と予測を開始する

3日リスクのラベル設計

```
y_3d(t) = OR(y(t+1), y(t+2), y(t+3))
```

- 72時間以内に1回でも不調があれば正例
- ピンポイントの t+3 予測は行わない（精度が不安定なため）

> **変更理由**: 期間ラベル方式の方が正例数が増えて学習が安定し、ユーザー体験としても「この先3日間で体調を崩すリスクがあります」の方が自然で行動変容につながりやすい。

---

5. 予測対象の設計

今日のリスク

- 入力：当日朝までのデータ

- 出力：今日の不調確率

3日間の期間リスク

- 入力：当日朝までのデータ
- 出力：今後72時間以内に一度でも不調になる確率

「当日朝までのデータ」の定義

- `x(t)` は **date_key=t の睡眠（起床日の睡眠）＋ t-1 までの体調・歩数・ストレス**で構成する
- 歩数は t 当日はまだ増えるため、原則 `steps(t-1)` を使用する
- 睡眠のみ date_key=t（当日起床分）を使える（起床時点で確定しているため）

予測機能の段階的開放

フェーズ	開放条件	表示
データ収集中	14日未満	「学習中」「あと〇日で開放」
今日のリスク開放	14日以上	今日の不調確率を表示
3日リスク開放	60日以上 AND 不調10件以上	3日リスクを表示
3日リスク未開放（条件未達）	60日以上だが不調が少ない	「準備中」「あと〇日」表示

---

6. 特徴量設計

必須（入力ゼロで計算可能な派生特徴量を含む）

基本特徴量：

- 睡眠時間
- 歩数
- 曜日
- 休日フラグ（**MVPは `is_weekend`（土日）のみ。**祝日対応は将来、日本祝日カレンダー or API で追加）

体調の時系列特徴量（入力負担ゼロ・MVP必須）：

- `mood(t-1)` ：前日の体調スコア
- `ma3(t-1)` ：t-1時点の3日移動平均
- `ma7(t-1)` ：t-1時点の7日移動平均
- `delta1(t-1) = mood(t-1) - mood(t-2)` ：t-1時点の前日差分
- `dev14(t-1) = mood(t-1) - ma14(t-1)` ：t-1時点の14日平均との偏差

> ⚠️ **リーク防止ルール**: x(t) に使ってよい体調系は **t-1以前のみ**。特徴量は全部「t-1時点で確定しているもの」に揃える。 `mood(t)` はラベル生成・表示には使うが、予測入力には絶対に入れない。

睡眠・歩数の偏差特徴量（入力負担ゼロ）：

- `sleep_dev` ：直近平均睡眠時間との差
- `steps_dev` ：直近平均歩数との差

任意

- ストレス（ユーザー入力）

> **設計意図**: ユーザー入力負担を増やさず、既存データからの派生特徴量で予測力を高める。体調スコアの時系列特徴量は最も予測力が高いと想定される（体調悪化は前日から兆候が出やすい）。

---

7. 日付基準

- 起床日をその日の睡眠として扱う
- 例：2/5 23:40 〜 2/6 06:30 → 2/6 の睡眠として扱う

date_key の定義

- `date_key` は**起床時刻を端末ローカルTZに変換した `YYYY-MM-DD`**
- Firestore に保存する timestamp は**UTC**
- 併せて `tz_at_wake` （例: `Asia/Tokyo` ）を内部保存する（ユーザー非表示）

> **設計根拠**: 海外移動や端末TZ変更時にdate_keyがズレる事故を防止する。

---

8. 入力設計

毎日入力（必須）

- **体調（5段階）** — 顔アイコン等で1タップ入力

> **変更点**: 3段階 → 5段階に確定。3段階だと「普通」に集中し分散が小さくなるため、予測精度に悪影響。UIは「5段階だけど1タップ」で負担が増えないよう設計する。

任意入力

- ストレス

睡眠

- **Apple Health / Google Fit から自動取得**（無料機能として提供）
- 連携できない場合：就寝時刻・起床時刻を手入力

歩数

- **Apple Health / Google Fit から自動取得**（無料機能として提供）
- 連携できない場合：手入力

---

9. 欠損値の扱い

体調（目的変数）

- 未入力日は学習に使わない

特徴量

- 過去7日平均で補完
- `*_missing` フラグを付与（欠損自体が情報になるケースをカバー）

---

10. モデル設計

初期（MVP）

- **ロジスティック回帰**（確率出力・解釈可能・軽量）
- ベースラインモデルとして継続利用

ハイパーパラメータ・正則化の段階設計

データ量に応じてパラメータを切り替える。少量データでは正則化を強くしてモデルを単純に保ち、データが増えたら柔軟にする。

ロジスティック回帰：

データ量	正則化強度 (C)	正則化種類	根拠
14〜59日	0.1（強め）	L2	少量データで過学習を防ぐ
60〜149日	0.5（やや強め）	L2	データ増加に合わせ少し柔軟に
150日以上	1.0（標準）	L2	標準的なバランス。300件程度ではこれ以上緩めると過学習リスク

LightGBM（60日以上で使用）：

データ量	max_depth	num_leaves	n_estimators	min_child_samples	learning_rate
60〜149日	3	8	100	5	0.1
150〜299日	4	16	150	5	0.05
300日以上	5	31	200	3	0.05

> **設計根拠**: 個人データ60〜200件の規模でOptunaなどの自動チューニングを毎日回すのは過学習リスクが高く、バッチ時間も増加する。データ量ベースの段階切替は安定性とパフォーマンスのバランスが良い。各パラメータはKaggle経験則と少量データでの実験知見に基づく。

LightGBM の導入条件と切替ロジック

条件	値

データ日数	60日以上
不調フラグ(1)の件数	10件以上
判定	**AND条件**（両方を満たした場合のみ）

切替ロジック（重要）：

条件達成は「LGBMに切り替える」ではなく、**「LGBMを試せる条件が整った」という意味**。少量データ（60〜120日程度）ではロジスティック回帰の方が安定するケースが普通にあるため、以下の手順で判定する：

1. 初期（条件未達）：ロジスティック回帰のみ
2. 条件達成（60日 AND 不調10件）：ロジスティック回帰と LightGBM を**両方学習**
3. 検証スコア（時系列分割でのAUC等）が良い方を採用
4. 採用モデルを `model_status.modelType` に記録

検証の分割方法：

データ量に応じて検証サイズを段階的に調整する。直近データを検証に使うことでリーク防止を担保しつつ、少量データ時の学習データ不足と大量データ時の「直近を学習に使えない」問題を両方回避する。

データ量	検証サイズ	根拠
14〜29日	20%（最低3日）	データが少ないので割合ベースで学習データを確保
30〜99日	7日	1週間分で曜日パターンをカバー
100日以上	14日	2週間分で安定した評価

- 指標：AUC（＋可能なら PR-AUC もログ出力）
- rolling window 検証は将来の改善として検討

> **設計根拠**: 固定14日だと少量データ時に学習データが不足する（例: 20日で学習6件）。割合固定（例: 20%）だと大量データ時に直近が大量に学習から外れる（例: 300日で直近60日が検証）。体調データは最近のパターンが最も重要なので、直近を必要以上に学習から外すのは予測精度に悪影響。段階設計がバランスが良い。

- 検証は毎日のバッチで自動判定し、より良い方を `predictions` に反映
- **条件未達の場合**（例：60日で不調3件）はロジスティック回帰のまま維持し、UXは「体調が安定しています」というポジティブなメッセージに変換

> **設計根拠**: 「LGBMの方が高性能だから常に強い」は半分正しく半分間違い。少量データでは単純で頑健なモデルの方がUX的に強いケースが多い。LGBMが安定してロジ回帰を上回り始める目安は総サンプル150〜300件以上・不調クラス20件以上（経験則）。条件達成時点（60日）では拮抗〜ロジ有利のことが多いため、自動比較方式を採用する。

不均衡データへの対処方針

このアプリでは出力が確率値であり、閾値（0.5）で二値判定する場面がない。そのため不均衡の悪影響は相対的に小さい。対処は「問題が出てから入れる」順序で行う。

運用方針：

- MVP（ロジスティック回帰）はまず `class_weight=None` で学習
- 出力確率が極端に低くレンジが潰れる（例：ほぼ 0.05〜0.15 に集中）など「分離が弱い」兆候が出たら `class_weight='balanced'` を試す
- LightGBM も同様に、まずデフォルトで学習し、必要なら `scale_pos_weight` 等を導入
- 不均衡対策の評価は**偽陽性増加（元気なのに高リスク）を特に重く見る**（離脱リスクが高いため）

手法の優先度：

手法	評価
何もしない（デフォルト）	◎ まず試す
クラス重み付け	○ 確率分布が潰れたら導入
アンダーサンプリング＋バギング	△ データ180日以上・不調30件以上で検討
SMOTE	× 非推奨（個人×時系列データでは人工サンプルが有害）

> **設計根拠**: このアプリでは「不調です」と言われて元気だった（偽陽性）方が、「大丈夫です」と言われて体調を崩した（偽陰性）よりもユーザーの信頼を損ねやすく離脱に直結する。重み付けは偽陽性を増やすリスクがあるため、必要性が確認されてから導入する。

学習方式

- 個人ごとのモデル
- 毎日1回再学習（深夜バッチ）

スケール時の移行パス

- 初期：個人モデル（少数ユーザー想定）
- 成長期：共通モデル＋個人平均との差分特徴量でスケール
- 将来：個人モデルは選択式（高精度モード）として有料化検討
- **設計上の注意**: Cloud Runの実装を「ユーザー単位で処理」ではなく「パイプラインを差し替え可能」にしておく

---

11. 信頼度（Confidence）設計

設計方針

予測が外れたときのユーザー離脱を防ぐため、予測確率と同時に**信頼度**を表示する。

MVP段階：ルールベースのみ（3段階）

信頼度	条件
低	データ30日未満 OR 不調5件未満
中	データ30〜59日 AND 不調5件以上
高	データ60日以上 AND 不調10件以上

- 直近7日の入力欠損率が30%以上の場合 → 1段階ダウン
- `model_status` に `unhealthyCount` と `recentMissingRate` を追加して管理

表示方針（統合UI）

信頼度	表示
低	「参考値です」表示に固定（確率の尖り具合は無視）
中以上	確率の尖り具合でニュアンスを追加（将来）

将来の拡張：確率の尖り具合

- モデル出力確率が 0.4〜0.6 の範囲 → 「判断が微妙なゾーン」のニュアンスを追加
- **ルールベース信頼度** = モデル全体の成熟度
- **確率の尖り具合** = 今日の予測の確信度

- 実装はユーザーフィードバックで「予測が曖昧でわからない」という声が出てから（YAGNI）

表示例

```
信頼度：高 × 確率が尖っている
→ 「不調リスク：高め」

信頼度：高 × 確率が0.5付近
→ 「不調リスク：やや注意（微妙なラインです）」

信頼度：低 × どんな確率でも
→ 「不調リスク：○%（まだ学習中の参考値です）」
```

---

12. 改善アドバイスの自動生成

目的

リスク表示だけでは「じゃあどうすればいい？」が分からない。ユーザーが**今日〜明日に変えられる行動**に限定して、**具体的な数値付きの改善アドバイス**を提示する。

対象パラメータ（可変なもののみ）

パラメータ	アドバイス例
睡眠時間	「あなたの場合、7時間以上の睡眠でリスクが下がる傾向があります。今夜は23:00までに就寝がおすすめです」
歩数	「8,000歩以上の日は体調が安定する傾向があります」
ストレス	「ストレスLv3以下の日は不調率が15%低くなっています」

> **対象外**: 曜日・過去の体調スコアなどユーザーが変えられないものはアドバイスに含めない。

生成ロジック

バッチ処理時に、各可変特徴量について以下を計算する：

1. **個人の統計量を算出**: 好調日(y=0)と不調日(y=1)の各特徴量の平均・分布を比較
2. **最適閾値の推定**: 好調日のデータから「リスクが低い行動水準」を算出（好調日の25パーセンタ

イル値）

3. **ギャップ検出**: 今日の値が「最適水準」から離れている特徴量を抽出
4. **具体的アドバイス生成**: 最適水準の数値と、推奨就寝時刻を含むテンプレートメッセージを生成

例：
- 好調日の睡眠：平均`7.2h` / `25`パーセンタイル`6.5h`
- 不調日の睡眠：平均`5.8h`
- 今日の睡眠：`5.5h` → ギャップ大
- 好調日の就寝時刻：平均`23:10`

→ 「あなたの好調日は平均`7.2`時間の睡眠を取っています。
今夜は`23:00`頃までに就寝すると、明日のリスクが下がる傾向があります」

睡眠の具体的アドバイス設計

睡眠は最も行動変容に繋がりやすいため、特に詳細に設計する：

分析項目	計算方法	アドバイスへの反映
推奨睡眠時間	好調日の平均睡眠時間	「〇時間以上の睡眠がおすすめ」
推奨就寝時刻	好調日の平均就寝時刻	「〇時頃までに就寝を」
リスク低下率	推奨水準以上の日の不調率 vs 未満の日の不調率	「不調率が〇%低くなっています」

Firestore 保存

`predictions/{date_key}` に `advice` フィールドを追加：

```
{
  "advice": [
    {
      "feature": "sleep_hours",
      "message": "あなたの好調日は平均7.2時間の睡眠です。今夜は23:00頃までに就寝がおすすめで
      "detail": {
        "targetValue": 7.2,
        "currentValue": 5.5,
        "recommendedBedtime": "23:00",
        "riskReduction": "不調率が30%低下"
      },
      "priority": 1
```

```
      }
    ]
  }
```

表示ルール

- 予測機能が開放済み（14日以上）の場合のみ表示
- アドバイスは**最大2件**（情報過多を防ぐ）
- リスクが低い日（例: 15%以下）はアドバイスなし（「良い調子です」のみ）
- 好調日のデータが10日未満の場合、具体的な数値は出さずテンプレートメッセージのみ

---

13. 特徴量の寄与度表示

目的

「なぜこの予測になったのか」をユーザーに説明することで、予測への納得感と信頼を高める。

実装方式

モデル	寄与度の算出方法
ロジスティック回帰	標準化済み係数 × 今日の特徴量値
LightGBM	SHAP値（`shap` ライブラリ）

バッチ処理での計算

ロジスティック回帰の場合：

```
# 標準化済み係数 × 標準化済み特徴量 = 各特徴量の寄与度
contributions = scaler.transform(X_today) * model.coef_[0]
```

LightGBMの場合：

```
import shap
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_today)
```

Firestore 保存

`predictions/{date_key}` に `contributions` フィールドを追加：

```
{
  "contributions": [
    {"feature": "sleep_hours", "label": "睡眠時間", "value": -0.35, "direction": "r
    {"feature": "mood_t1", "label": "前日の体調", "value": 0.28, "direction": "posi
    {"feature": "steps_t1", "label": "昨日の歩数", "value": -0.15, "direction": "ne
  ]
}
```

- `value`：寄与度（正=リスク増加方向、負=リスク低下方向）
- 上位3件のみ保存（UIの見やすさ優先）

表示方針

- 「今日の予測に影響した要因 TOP3」として表示
- 正方向（リスク増加）は赤系、負方向（リスク低下）は緑系で色分け
- ユーザー向けラベル（`sleep_hours` → 「睡眠時間」）に変換して表示

特徴量のユーザー向けラベル対応表

特徴量	ユーザー向けラベル
sleep_hours	睡眠時間
steps_t1	昨日の歩数
mood_t1	前日の体調
ma3_t1	直近3日の体調傾向
ma7_t1	直近7日の体調傾向
delta1_t1	体調の変化
dev14_t1	普段との体調差
sleep_dev	普段との睡眠差
steps_dev	普段との歩数差
day_of_week	曜日
is_weekend	休日かどうか

stress_val	ストレス

---

14. データ可視化（グラフ設計）

14.1 体調 × 特徴量 比較グラフ

ユーザーが体調と任意の特徴量を重ねて時系列で確認できるグラフ。モデルが何を学習しているかの透明性を確保する。

仕様：

- 横軸：日付（直近7日 / 30日 / 全期間を切替可能）
- 左軸：体調スコア（1〜5、固定）
- 右軸：ユーザーが選択した特徴量（自動スケール）
- 特徴量は**全学習特徴量を選択肢として表示**（透明性重視）

選択可能な特徴量とスケール：

特徴量	表示名	取りうる範囲	備考
sleep_hours	睡眠時間	0〜12h	
steps_t1	昨日の歩数	0〜30,000	個人の最大値で上限調整
stress_val	ストレス	1〜5	体調と同スケール
mood_t1	前日の体調	1〜5	
ma3_t1	直近3日の体調傾向	1〜5	
ma7_t1	直近7日の体調傾向	1〜5	
delta1_t1	体調の変化	-4〜+4	0を中心に表示
dev14_t1	普段との体調差	-4〜+4	0を中心に表示
sleep_dev	普段との睡眠差	-6〜+6h	0を中心に表示
steps_dev	普段との歩数差	動的	0を中心に表示
day_of_week	曜日	0(月)〜6(日)	

is_weekend	休日	0 or 1	

スケール自動調整ルール：

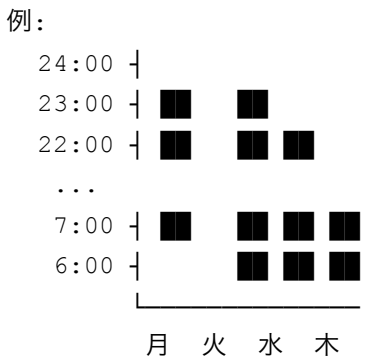
- 固定範囲がある特徴量（体調、ストレス等）はその範囲を使用
- 歩数のように範囲が個人差大きい特徴量は、個人データの5パーセンタイル〜95パーセンタイルで上下限を設定
- 偏差系（差分・dev系）は0を中心に±最大絶対値で対称にスケーリング

14.2 睡眠パターングラフ（専用）

就寝・起床時刻は数値特徴量とスケールが異なるため、別グラフで可視化する。

仕様：

- 横軸：日付
- 縦軸：時刻（0:00〜24:00、上下反転で深夜が上に来る表示も検討）
- 就寝時刻を棒の下端、起床時刻を棒の上端として、睡眠時間をバー表示
- 色分け：睡眠時間が推奨値（好調日の平均）以上なら緑系、未満ならオレンジ系



---

15. 外れた日のフォロー設計

目的

- 予測外れによる離脱を防ぐ
- フィードバックデータで予測改善を加速

実装方針

- **毎日の通知で聞くのは避ける**（入力負担で離脱リスク）
- **週次まとめ入力の導線に1タップフィードバックを組み込む**

「先週の予報、実際はどうでした？」
［当たった］［外れた］［わからない］

---

13. システム構成

フロントエンド

- Flutter（iOS / Android）

バックエンド

- Firebase Firestore（データ保存）

学習・推論

- Cloud Run（Python）
- **毎日 03:30 JST にバッチ実行**
- その日の `predictions/{today}` を生成（当日分）
- アプリ側は起床後にFirestoreから当日の予測結果を読み取って表示

バッチ対象ユーザーの抽出：

- 対象： `daily` が直近2日で更新されたユーザー（ `updatedAt` で判定）
- それ以外はスキップ（非アクティブユーザーの無駄な処理を防止）

---

14. データ構造（概要）

日次ログ

```
users/{uid}/daily/{date_key}
```

フィールド	内容

sleep	就寝 / 起床 / 時間 / 取得方法
steps	歩数
stress	ストレス（任意）
moodScore	体調（5段階）
tz_at_wake	起床時の端末タイムゾーン（例: `Asia/Tokyo` ）※内部保存・ユーザー非表示

モデル状態

`users/{uid}/model_status/current`

フィールド	内容	カウント定義
daysCollected	収集日数	`moodScore` が存在する日のみカウント
daysRequired	必要日数	—
ready	予測開放済みか	—
unhealthyCount	不調フラグ(1)の累計件数	不調ラベル `y=1` が生成された日のみカウント
recentMissingRate	直近7日の入力欠損率	—
modelType	現在のモデル種別（logistic / lightgbm）	—
confidenceLevel	信頼度（low / medium / high）	—

予測結果

`users/{uid}/predictions/{date_key}`

フィールド	内容
pToday	今日の不調確率
p3d	3日リスク確率（開放後のみ）

confidence	信頼度（low / medium / high）
generatedAt	予測生成タイムスタンプ（UTC）
modelVersion	使用モデル（例: `logistic_v1` , `lgbm_v1` ）

---

15. 技術スタック

レイヤー	**技術**
フロントエンド	Flutter
データベース	Firebase Firestore
学習・推論	Cloud Run（Python）
初期モデル	ロジスティック回帰（scikit-learn）
段階モデル	LightGBM（条件達成後）

---

16. リリース方針

- 初期は完全無料
- 学習後に予測機能を段階的に開放
- ユーザーの継続率を最優先

---

17. マネタイズ候補（未実装・設計メモ）

以下は将来の有料化候補として設計段階で記録しておくもの。MVP段階では実装しない。

- 週次 / 月次レポート（PDF出力等）
- 詳細分析ダッシュボード
- 予測期間の延長（7日予測）
- 個人モデル（高精度モード）の選択式提供

---

18. 将来的な拡張（未確定）

- 7日予測
- 月次分析
- 有料機能

実装優先度（MVP）

優先度	タスク	備考
1	体調5段階入力のUI実装	顔アイコン1タップ
2	体調時系列特徴量の追加	前日体調・移動平均・差分・偏差
3	今日のリスクのみ開放	3日リスクは60日以降
4	信頼度はルールベース3段階	model_statusに項目追加
5	週次フィードバック導線	1タップフィードバック

変更履歴

日付	変更内容	根拠
2026-02-06	体調スコア: 3段階 → 5段階	分散確保・予測精度向上
2026-02-06	3日リスク: MVP非開放（60日以降に段階開放）	初期精度の信頼性担保
2026-02-06	3日ラベル: t+3 → OR(t+1..t+3)	正例数増加・学習安定
2026-02-06	特徴量追加: 体調時系列・偏差系	入力負担ゼロで予測力向上
2026-02-06	LightGBM切替条件: 60日 AND 不調10件	サンプル数担保
2026-		

02-06	信頼度設計を新設	予測外れ時の離脱防止
2026-02-06	外れフォロー設計を新設	エンゲージメント・予測改善
2026-02-06	model_statusにフィールド追加	unhealthyCount, recentMissingRate, modelType, confidenceLevel
2026-02-06	date_key定義を明確化（ローカルTZ + UTC保存 + tz_at_wake）	TZ変更・海外移動時の事故防止
2026-02-06	休日フラグ: MVPは is_weekend のみ	祝日APIは後回しで実装軽量化
2026-02-06	14日未満の不調定義ルール追加	エッジケースでの実装事故防止
2026-02-06	時系列特徴量のリーク防止ルール明記	t-1以前のみ使用可、mood(t)は入力に入れない
2026-02-06	model_statusのカウント定義を明確化	daysCollected=moodScore存在日、unhealthyCount=y=1の日
2026-02-06	predictions に generatedAt, modelVersion 追加	デバッグ・原因追跡用
2026-02-06	バッチ実行時刻を 03:30 JST に固定	アプリ側の表示タイミング設計明確化
2026-02-06	不均衡データ対処方針を追加	デフォルトは何もしない→必要時のみ重み付け。偽陽性を特に重視
2026-02-06	LightGBM切替: 単純切替 → 両方学習＋検証比較方式に変更	60日時点ではロジ回帰が安定するケースが多いため自動比較を採用
2026-02-07	体調時系列特徴量の表記をt-1時点で統一（delta1, dev14等）	リーク防止ルールとの矛盾解消
2026-02-07	「当日朝までのデータ」の範囲を厳密に定義	睡眠=当日起床分、体調/歩数/ストレス=t-1以前
2026-02-07	検証スコアの分割方法を明記（直近14日検証、指標AUC）	実装時の迷い防止
2026-02-07	バッチ対象ユーザー抽出ルール追加（直近2日更新のみ）	スケール時のコスト・速度対策

2026-02-12	改善アドバイス自動生成機能を追加（セクション12）	リスク表示だけでは行動変容に繋がらないため
2026-02-12	特徴量寄与度表示機能を追加（セクション13）	予測の説明性向上・ユーザーの納得感確保
2026-02-12	データ量ベースのハイパーパラメータ段階設計を追加	過学習防止とデータ規模に応じた柔軟性の両立
2026-02-12	アドバイスを具体的数値付き（推奨睡眠時間・就寝時刻等）に強化	行動変容の促進
2026-02-12	ヘルスケア連携（Apple Health / Google Fit）を無料MVP機能に変更	ユーザー入力負担の軽減
2026-02-12	特徴量比較グラフを追加（体調×任意特徴量の2軸表示）	学習データの透明性確保
2026-02-12	睡眠パターン専用グラフを追加（就寝・起床時刻のバー表示）	時刻データは数値グラフと性質が異なるため分離
2026-02-12	検証分割を固定14日からデータ量ベースの段階設計に変更	少量データ時の学習データ不足と大量データ時の直近学習除外を両方回避