ChatGPT

データセット構築パイプラインの調査報告

パイプライン概要と調査ポイント

`run_full_dataset.py` は株価・テクニカル指標・財務データ・市場指数(TOPIX)・投資フローデータ等を結合し、機械学習モデル向けの日次パネルデータセットを構築するスクリプトです [1] [2]。本調査では、このパイプライン（特に `run_full_dataset.py` から呼ばれる各処理）について以下の観点で確認しました：

- **特徴量計算処理の妥当性**：株価リターンや移動平均、テクニカル指標、Zスコア等の数式・ウィンドウ処理が正しく実装されているか
- **日次データとの結合処理**：週次データ（信用残・投資部門別売買など）を日次に拡張する際の「as-of結合」や有効期間の扱いが適切か
- **欠損処理・正規化・リーク防止**：欠損値や非営業日の処理、特徴量の正規化が適切に行われているか。また、将来データの混入を防ぐ措置（リーク防止）が実装されているか
- **MLデータセットの品質**：未来情報の混入がないか、クロスセクショナルな正規化に注意が払われているか、データの一貫性・整合性チェックが行われているか

以下、各ポイントについて詳細に検証し、問題点や改善提案を述べます。

1. 特徴量計算処理の実装と妥当性

(a) 株価リターン・対数リターン – 株価の前日比リターンや複数営業日リターンは、Polarsの `pct_change` を用いて計算されています [3]。例えば、1日リターン(`returns_1d`)は `Close` の前日比変化率、5日リターン(`returns_5d`)は5営業日前との比率として算出されます [3]。コード上でもデータフレームを銘柄コード単位(`over("Code")`)で並べ替えた上でリターン計算しており [4]、銘柄ごとの時系列方向に過去データから計算していることが確認できます。**この計算は正しく、将来日の価格を参照しない安全な実装**です。なお、10日・20日といった長期のリターンについては、パイプライン終盤で不足していれば追加算出する処理もあります [5] [6]（例えば120日リターンが無ければ追加）。

(b) ボラティリティ – `volatility_5d/10d/20d/60d` は、1日リターンの過去ローリング標準偏差を年率換算(√252)したものです [7]。コードでは、リターンに対し窓幅5日・10日等の `rolling_std` を計算し、それに `sqrt(252)` を掛けています [8]。例えば5日ボラティリティは以下のように計算されています（欠損値回避のため `min_periods=5`）：

[9]

上記の通り、**標準偏差算出期間や年率換算係数も一般的な定義通りで妥当**です。なお、開始直後のデータでは十分な過去日数がないため、最初の4日間は `volatility_5d` がNullとなりますが、これは自然な挙動です（後述の有効フラグで対応）。

(c) 移動平均・テクニカル指標 – 株価の単純移動平均(SMA)や指数平滑移動平均(EMA)も適切に算出されています。例えば、5日・10日・20日・60日・120日のSMAはPolarsの `rolling_mean` で計算され [10]、5日・10日・20日・60日・200日のEMAは `ewm_mean(span=...)` で算出されます [11]。EMA計算では `adjust=False`（通常の指数平滑）かつ `ignore_nulls=True`（欠損は無視）としており正確です [12]。また、**移動平均類の有効性**については、例えば**60日分のデータ蓄積が必要な指標**を扱うため、後述する `is_valid_ma` フラグで「60日未満の期間は不完全」とマークしています [13]。

テクニカル指標も代表的なものが計算されています。例えば:

- RSI (Relative Strength Index)：14日RSIは `pandas_ta` 利用で計算されます [14]（2日RSIも計算対象）。2日RSIについては、**データ蓄積5日以上で有効**とする基準がフラグで実装されています [13]。
- MACD系列：おそらく `pandas_ta` でMACD・シグナル・ヒストグラムを計算しています。コードではMACDが欠けていた場合、`macd = macd_signal + macd_histogram` で再構築しています [15]。これより**MACD, シグナル, ヒストグラムの関係は一貫しており**、欠損時も整合性が保たれます。
- ATR(14)：高値-安値や前日終値との差分からTrue Rangeを計算し、14期間の指数平滑平均でATRを算出しています [16]。この実装はWilderの方法（14日間の平滑移動平均）に近似した形で、**妥当な計算法**です。
- ストキャスティクス%K(14)：過去14日間の高値・安値レンジに対する当日終値の位置を%で表しています [17]。計算式も一般的な定義に沿っています。

- ADX(14)：+DI,　-DIの計算からADXを算出しています [18] [19]。具体的には、当日高値と前日高値差などから方向性指数(DM)を求め、14日EWMAで+DI/-DIを計算、そこからDXを得てさらに14日EWMAでADXとしています。**計算手順は正しく**、トレンドの強弱を捉える指標として機能するでしょう。

- ボリンジャーバンド系：市場指標計算内でBB幅(`bb_width`)、%b(`bb_position`)が算出され [20] [21]、銘柄個別のBBも同様に計算されていると思われます（コード上ではエイリアス統一で `bb_bandwidth` → `bb_width` 等のマッピングがあります [22]）。

(d) 市場 (TOPIX) 特徴量 – TOPIX指数データから**26種類の市場レベル特徴**が生成されます [23]。内容はリターン系列、トレンド系列、ボラティリティ・レンジ系列、リスク・ドローダウン系列、Zスコア正規化、および市場レジームを示すフラグです [23] [24]。実装上は `MarketFeaturesGenerator` クラスで以下のように計算されています:

- リターン: 1日、5日、10日、20日のTOPIXリターン (`mkt_ret_*`) [25]
- トレンド: 5日〜200日EMA (`mkt_ema_*`)、20日EMAとの差分 (`mkt_dev_20`)、短期/中期EMA間ギャップ(`mkt_gap_5_20`)、20日EMAの3日変化率(`mkt_ema20_slope_3`) [25] [26]
- ボラティリティ: 20日ローリングの標準偏差(年率換算) `mkt_vol_20d`、ATR14・NATR14 [27] [28]、20日ボリンジャーバンド%指標(`mkt_bb_pct_b`)と幅(`mkt_bb_bw`) [29]
- リスク指標: 累積高値からのドローダウン率(`mkt_dd_from_peak`)と大変動フラグ(`mkt_big_move_flag`：60日ボラの2倍を超える1日変動) [30] [31]
- **Zスコア系列**: 上記の中でも代表的な4指標（1日リターン、20日ボラ、BB幅、ドローダウン）について、過去252営業日（約1年）の移動平均・標準偏差からの偏差を計算し、`mkt_*_z` として出力します [32]。実装では `rolling_mean(252)` と `rolling_std(252)` を用い、**直近1年の中での相対的な水準**を示すようになっており妥当です [32]。なお、計算には `min_periods=252` が指定されているため少なくとも1年分データがなければZスコアはNullになります [32]。初期1年のZスコア欠損は後述のフラグで扱います。
- 市場レジームフラグ: 例えば mkt_bull_200（終値が200日EMAより上か）や、mkt_trend_up（短期トレンドが上向きか）、mkt_high_vol（ボラZが+1超か）、mkt_squeeze（BB幅Zが-1未満か）等が算出されます [33]。**これらは市場環境を示す直感的なブール特徴量で、定義も標準的**です。

以上の市場特徴量は銘柄データフレームに `Date` で左結合され、さらに後述の「クロス特徴量」算出に用いられます [34] [35]。

(e) クロス特徴量（銘柄×市場） – 8種類のクロス特徴量が定義されています [36]。これは各銘柄の値動きやボラティリティを、市場(TOPIX)の動きと組み合わせて得られる特徴です。`CrossMarketFeaturesGenerator.attach_market_and_cross()` 内で以下を計算しています [37] [38]：

- **β (60日ベータ)**: 各銘柄の過去60営業日におけるリターンと市場リターンの共分散/分散で計算されます [39] [40]。特徴的なのは、市場リターンに1日のラグを入れられる設計になっている点です [37]（デフォルト `beta_lag=1`）。**これにより、当日株価リターンと前日市場リターンの相関でβを求める形となり、同日値動きの同時性による情報リークを避けています** [37]。計算上は60日移動平均で共分散・分散を算出しβを得ています。また、データが少ない初期には60日βが計算できないため20日版を併用し、`beta_60d` として補完している点も堅実です [41] [42]。
- **α (アルファ)**: 上記βを用い、銘柄リターンから市場の影響分を差し引いた残差リターンです。1日アルファ `alpha_1d` は returns_1d - β×mkt_ret_1d で算出されます [43]。5日アルファ `alpha_5d` も同様です [44]。**これにより市場全体要因を除いた銘柄固有のリターンが特徴量化されています。**
- **相対強度**: `rel_strength_5d` は returns_5d - mkt_ret_5d として計算され、市場に対してその銘柄が直近強かったか弱かったかを表します [44]。
- **トレンド整合性**: `trend_align_mkt` は銘柄の短期トレンド指標(`ma_gap_5_20` の符号)と市場トレンド(`mkt_gap_5_20` の符号)が一致しているかを示すフラグです [45]。両者の符号が同じ（同方向のトレンド）なら1になります。
- **レジーム依存アルファ**: `alpha_vs_regime` は alpha_1d × mkt_bull_200 として算出されています [46]。つまり、市場が強気（200日線上）かどうかで銘柄アルファをゲーティングする特徴量です。強気相場下でプラスのアルファを出せている銘柄かどうか等を捉えます。
- **特異ボラ比**: `idio_vol_ratio` は volatility_20d / mkt_vol_20d で計算されます [47]。**銘柄の20日ボラが市場全体のボラと比べどれほど高いか**を示し、1より大きければ市場よりボラティリティが高いことを意味します。
- **β安定性**: `beta_stability_60d` は 1 / (std(β_60d, 過去20日) + ε) で、過去20日間のβ値の変動が小さい（安定している）ほど値が大きくなる特徴量です [48]。β値の計測誤差の大きさを逆指標化したもので、**安定したβを持つ銘柄かを示す指標**といえます。

以上のクロス特徴量の数式定義・計算手順に大きな問題は見られず、**市場全体との関連で銘柄特性を捉える有用な特徴を漏れなく計算している**と判断できます。

(f) 投資フロー特徴量 – 投資部門別売買（週次フロー）データからは13種類の特徴量が作られます [49]。週次データとして提供される「外国人」「個人」などの売買バランスを、区間データとして処理した後、銘柄の市場区分（セクション）単位で日次に展開しています。主な計算内容は以下です [50] [51]：

- **ネット比率**: 外国人と個人の売買バランスのネット額を総額で割った比率（例：`flow_foreign_net_ratio` は ForeignersBalance / ForeignersTotal） [50] [52]、個人も同様。0より大きければ買い越し、負なら売り越しを示す。
- **活動度**: 外国人の売買総額シェア（全市場に対する割合、`foreign_share_activity`）や、複数部門の買い越し割合を表すブレッドス指標(`breadth_pos`：6部門中何部門が買い越しか)が計算されています [53] [54]。後者はTrue/FalseをIntに変換して平均を取ることで実現しています（6部門中の正の割合） [54]。
- **Zスコア化**: 各セクション内で52週（1年）の移動平均・標準偏差を用い、外国人・個人のネットバランスや総売買高の異常度をZスコアにしています（例：`flow_foreign_net_z`, `flow_individual_net_z`, `flow_activity_z`） [51]。計算は （値 − 52週平均）/ 52週標準偏差 で行われており、各値がそのセクションの過去1年と比べどの程度偏っているかを示します [51]。
- **スマートマネー指数**: `flow_smart_idx` は (foreign_net_z − individual_net_z) として計算され、外国人と個人の資金フロー傾向の差を示す指数です [55]。さらに4週モメンタム `flow_smart_mom4` はこの指数の4週移動平均乖離、`flow_shock_flag` はスマートマネー指数の絶対値が2以上（統計的に大きな偏差）かを示すフラグです [55] [56]。

- **エイリアス統一**: 計算後、列名を仕様に合わせ `flow_` プレフィックスに揃える処理が行われています [57]。例えば `foreigners_net_ratio` 列を `flow_foreign_net_ratio` 別名で保持し、以降は後者を使うようにしています [58]（元列も保持はされていますが、最終的には不要な列はドロップされます [59]）。

以上のフロー特徴は**週次指標を適切に正規化・圧縮した上で日次データに反映するための計算**であり、特にZスコア化やモメンタム指標の導入は、フローの異常検知やトレンド把握に有用です。数式定義にも問題はありません。強いて言えば、52週のローリング計算で `min_periods` 指定がなく初期数週にも値が入る可能性がありますが、フラグ等で無効化される（後述）ため大きな誤用にはなりません。

(g) 信用残（週次信用取引残高）特徴量 – こちらも**週次データを日次に展開するブロック**で、信用買残・売残の水準や変化を特徴量化しています。計算内容と根拠をまとめると:

- **信用取引残高の合計**: `margin_long_tot`（信用買残高）, `margin_short_tot`（信用売残高）, そして両者の合計 `margin_total_gross` を計算 [60] [61]。これらは元データの数値をそのままFloat型で格納しています。
- **ネット・比率**: ネット残高 = 買残－売残(`margin_net`)、信用取組比率 = 買残/売残(`margin_credit_ratio`)、および**需給バランス**を示す `margin_imbalance` = (買残－売残)/(買残＋売残) を計算しています [62] [63]。`margin_credit_ratio` は買残が売残の何倍あるか、`margin_imbalance` は -1〜+1で需給の偏りを示す指標です。
- **標準化シェア**: 株式分割等での調整後残高比率でしょうか、`LongStandardizedMarginTradeVolume` を買残で割った比率(`margin_std_share_long`)などを算出しています [64]（この辺りは利用有無次第ですが、データの内訳情報と思われます）。
- **週次増減**: 各残高の前週差分である Week-over-Week 変化を算出し、`margin_d_long_wow`, `margin_d_short_wow`, `margin_d_net_wow` としています [65]。比率についても `margin_d_ratio_wow` = 今週の信用取組比率 − 前週比率 で変化を捉えます [65]。
- **モメンタム**: 信用残高合計の**4週モメンタム**を `margin_gross_mom4` として計算しています [66]。具体的には 今週の合計残高 − 過去4週平均 としており、増加傾向なら正、減少傾向なら負になります。
- **52週Zスコア**: 信用買残・売残・総残高・取組比率の各系列について、過去52週の平均・標準偏差からの偏差を計算し、それぞれ `long_z52`, `short_z52`, `margin_gross_z52`, `ratio_z52` としています [67] [68]。例えば `long_z52` の計算式は以下引用の通りです:

[67]

(上記コード: margin_long_tot の52週平均・標準偏差からの偏差を long_z52 として追加)

このように**1年程度の期間での相対的な高低を示す指数**となっており、極端な値であれば需給が例外的な水準にあることを捉えられます。Zスコア計算において適切に微小値EPSでゼロ割りも避けています [69]。

- **出来高基準の正規化**: 信用残高がその銘柄の流動性規模に対して大きいか小さいかを見るため、**20日平均出来高(ADV20)** で正規化した特徴量が追加されます [70]。例えば `margin_long_to_adv20` は買残高をADV20で割ったもの、`margin_d_long_to_adv20` は週次増加分を（ADV20×5日分）で割ったものです [71] [72]。後者では1週＝5営業日とみなし、5倍の出来高と比較しています。**これにより、出来高に比して極端に大きな信用残の動きも検知しやすくなっています**。ADV20は日次株価データから欠かさず計算され（調整済み出来高があればそれを使う設計） [73]、将来データを見ない形で期間平均されています [74]。

- **タイミング指標**: 日次データに結合後、各銘柄・日付行について「その日が週次データの発効日か」「発効日から何日経過したか」などの指標が付与されます。具体的には `margin_impulse`（発効日に1、それ以外0）、`margin_days_since`（その日の時点で週次データ発効から経過した営業日数）です [75]。**これらは週次→日次展開時に情報の鮮度をモデルに伝える重要な特徴です**。

以上の信用残特徴量計算は、**必要な指標を網羅し、適切なウィンドウ長で過熱感や変化を捉えられるよう工夫されています**。特にADV20によるスケーリングは株式ごとの出来高差をならす有効な正規化であり、妥当と言えます [76]。

(h) セクター特徴量 – コード上、`--sector-onehot33` 等のオプションがあり33業種のワンホットやセクター系列平均、ターゲットエンコーディングなどを追加できる予定ですが、現状の実装では `MLDatasetBuilder.add_sector_features/series/encodings()` はスタブ（中身が `return df`）になっています [77] [78]。そのため**現在のパイプラインではセクター関連特徴量は実質追加されていません**（listed_info_parquetを指定しても警告ログが出るだけです [79]）。今後の拡張予定と思われますが、**モデル性能向上の観点ではセクター平均や業種別トレンドを取り入れる余地**があります。実装する際はデータリークに注意してK-foldターゲットエンコーディングを行う等、計画されている通りのアプローチが望ましいでしょう [80]。

2. 週次データの日次結合処理 (as-of結合)

パイプラインでは、**日次株価データに対し週次データ（投資フローや信用残、四半期財務など）を「そのデータが有効となった日以降」に適用**する結合処理を行っています。これは**将来情報の混入を防ぐための重要な設計**であり、「発効日」の計算と **as-of 結合** によって実現されています。

(a) 財務諸表データの結合 (四半期決算等) – SafeJoinerV2にて `join_statements_with_dedup` 関数が実装されており、開示日時に応じて**effective_date（有効日）**を計算し、その日以降に財務データを適用しています [81] [82]。ルールは「**当日15:00より前に開示されたものは当日中に織り込み、15:00以降なら翌営業日から有効**」というものです [83]。コード上も、開示時刻カラム `DisclosedTime` と市場の通常取引終了時刻(通常15:00、半日なら11:30)を比較し、条件に応じてeffective_dateを当日または次営業日に設定しています [82]。このロジックにより、決算短信など**引け後に出た情報が当日中のデータに影響を与えない**ようになっています [84]。さらに、同一銘柄・同日で複数開示があった場合は最新のみ残す処理や、銘柄コードの正規化等も施されています [85]。最後に日次株価データと `effective_date` で結合し（銘柄コードでマッチ、日付はas-ofの後ろ向き結合）、`stmt_imp_statement`（開示インパルス）と `stmt_days_since_statement`（開示経過日数）が付与されます [86]。例えば、決算開示日当日は `stmt_imp_statement=1` かつ `stmt_days_since_statement=0` となり、以降日を追うごとに経過日数が増えていきます。**この一連の処理は財務データの非公開期間中の値をモデルに見せないための適切な措置**であり、リーク防止が徹底されています。

(b) 投資部門別売買（週次フロー）データの結合 – 週次フローは市場区分（セクション）単位で毎週公表されるため、**セクション×週次区間**データを作り、それを各銘柄の日次に紐付けています [87] [88]。具体的には:

- 公表日PublishedDateの翌営業日をその区間の `effective_start`（有効開始日）とし、次回公表分の前日を `effective_end` とします [89] [90]。例えば毎週水曜に先週分が発表される場合、水曜（T+1）が有効開始、翌週火曜が有効終了となります。
- こうして得られた各セクションごとの区間データに対し、前述のフロー特徴量（net比率やZスコア等）を計算します [50] [51]。
- 次に日次営業日カレンダーと各セクションの区間データを **後ろ向きのas-of結合** します [91] [92]。Polarsの `.join_asof(strategy="backward")` を利用し、各日のDateに対してその日以前で最新の `effective_start` を持つ区間レコードを紐付けます [93]。この時点では各日付行に、その日まで有効な最新フロー指標が一旦付与されます。
- しかし注意すべきは、**結合後の各日が本当にその区間内か**のチェックです。join_asofはデフォルトで「その日以前の最新区間」を持ってきますが、もし既に区間が終了(`effective_end`)していても次がない場合は直前の区間が引き続きマッチしてしまう可能性があります。そこで、コードでは各日行について `Date` が `effective_start` ～ `effective_end` の範囲外なら、そのフロー値をNullにリセット

する処理を行っています [94] 。これにより**区間が終了した後のデータには古い値が残らない**よう保証しています。
- `flow_impulse` と `days_since_flow` 列もここで追加されます。`flow_impulse` はその日がちょうど `effective_start` と同じなら1、そうでなければ0になるよう条件分岐で設定されます [95] 。`days_since_flow` は、有効区間内では（日付 − effective_start の日数）を計算し、区間外やデータ無しの場合は999を割り当てています [96] 。999は「データ未経験」を意味するダミー値です [97] （後工程の有効フラグ判定で>14なら無効扱いするための大きな数値）。

以上の手順により、例えば「先週末までの累計で外国人が買い越しか売り越しか」といったフロー情報は、その週の翌営業日（月曜か水曜など公表日）から次回更新までの日に適用されます。**これにより将来のフロー情報が過去日に混入することはなく**、また `flow_impulse` や `flow_days_since` で情報の鮮度をモデルに伝達できています [98] 。

さらに、セクション情報が不明な銘柄への対処もなされています。`attach_flow_with_fallback` では、銘柄の所属市場区分が取得できない場合に `AllMarket` として全市場統合データにフォールバックする戦略が実装されています [99] [100] 。フォールバック時は該当銘柄行に `is_section_fallback=1` を立てて識別し、フロー指標自体はAllMarketの値を割り当てるか、または0で埋める処理をしています [101] [102] 。このおかげで「区分不明のためフローデータ無し」という漏れが無い一方、フォールバック使用銘柄はモデル側で無視するなど対応も可能です。**いずれにせよ、週次フローデータが日次データに結合される処理はリーク防止と欠損対策が両立されており、堅牢な実装**と言えます。

(c) 信用残データの結合 – 信用残も週次ベースですが、こちらは銘柄単位で提供されます。実装は `margin_weekly.add_margin_weekly_block()` にまとまっており、基本的な流れはフローと類似しています [103] [104] :

- `PublishedDate` がある場合はその翌営業日を `effective_start` 、無い場合は週次データの日付 `Date` に対してあらかじめ指定されたラグ（日数）を加えた日を `effective_start` とします [105] [106] 。デフォルトでは `lag_bdays_weekly=3` 営業日（約半週後）となっており、例えば発行日が明示されないデータについて「週末データなら翌水曜から有効」程度の保守的ラグを置いています [107] 。このルールは設定で変更可能です。
- あとは各銘柄の週次行ごとに計算済み特徴量を持つ `w_feat` を用意し、日次株価の各行に対して **join_asof(`left_on=Date` , `right_on=effective_start` , `by=Code` , backward)** を行います [108] [109] 。これにより、その銘柄についてその日までに有効化された最新の信用残情報が付加されます。
- そして、先述した通りの `margin_impulse` （有効日フラグ）、`margin_days_since` （経過日数）、`is_margin_valid` （有効な信用残データがあるか）等を算出します [75] [110] 。信用残ではデータ無しの場合 `effective_start` 自体がNullになるため、`is_margin_valid = 0` になる設計です [111] 。一方、`margin_days_since` はNullのままとなりますが、Nullは「未適用」を意味すると解釈できます（flowでは-1や999埋めをしましたが、marginではFlagsで十分区別できるためか特に埋めずにNullを保持しています）。

以上、信用残の結合も**PublishedDateや指定ラグによるタイミングずれを考慮しており、将来週の残高データが過去日に表れないようになっています** [112] [113] 。もしPublishedDateが提供されるデータならより正確にT+1日を算出できますし、無い場合でも3営業日という保守的ラグで安全側に倒しています。**結合ロジック・リーク防止の観点で特に問題は見当たりません**。

3. 欠損処理・正規化・リーク防止策の評価

上述した計算・結合処理には、適切な欠損値処理やデータリーク防止の工夫が随所に盛り込まれています。ここではそれらを整理し、データセット全体の品質への影響を考察します。

(a) 欠損値と有効データのフラグ管理 – 特徴量計算上、生じる欠損（Null）には２種類あります。(1) **初期期間などデータ不足による計算不能**な欠損、(2) **そもそもその銘柄・日時には存在しない情報**による欠損です。これらに対し、本パイプラインでは**有効性フラグ (`is_*_valid`) を併用して欠損を管理**しています。

- **テクニカル指標の初期欠損**: 例えば60日移動平均は上場直後は計算できません。この場合パイプラインでは、`row_idx`（各銘柄の経過営業日数）を用いて、**60日未満の行では `is_valid_ma=0`** を付与しています [13]。実装上は最終段階で `row_idx>=60` なら1、それ未満は0とするInt8フラグを追加しています [13]。同様に、2日RSIなど期間依存指標にも `is_rsi2_valid` を付けています（5日以上経過で有効化） [13]。EMA200のような長期指標については、設計上はフラグを付与すべきですが現コードでは `is_ema200_valid` 生成が抜けている可能性があります（スペックにはあるが算出箇所不明 [114] [115]）。この点は改善余地です。

- **週次/非日次データの欠損**: 週次フロー・信用残・財務は常にあるとは限りません。未上場や非対象銘柄では丸ごと情報無しです。これについて各結合処理では、Nullのままとする代わりに**有効フラグで制御**しています。例としてフローでは、結合後 `activity_z` など主要指標がNullかどうかで `is_flow_valid` を付与しています [116]。実装では `is_flow_valid = (activity_zが非Nullか)` と単純に判断しています [116]。財務についても `stmt_days_since_statement` が存在すればそれが>=0かどうかで `is_stmt_valid` を付与（デフォルトでは全件Nullの場合0になる）しています [117]。信用残も `is_margin_valid` で有効/無効を示し、値が無いときは0になります [111]。これにより、モデル学習時に**無効データ行をフィルタ除外したり、フラグを入力に使って欠損を区別することが可能**になっています。実際ValidityFlagManagerでは、例えばフローなら「直近2週間以内にデータがあるか」「必須列がNullでないか」など複数条件でis_flow_validを再計算し直す機能もあります [118] [119]。

- **Null値の具体的処理**: 基本的にはNullを残す実装ですが、一部で**ダミー値埋め**も使われます。例えばフローの `days_since_flow` はNullだと計算に不便なため -1や999で埋めています [97]（前者はPolars前処理段階、後者はSafeJoinerV2で区間外を999日扱いに）。また財務の `stmt_days_since_statement` はNullを999で埋めています [86]。埋めた後でフラグ付けしているため、モデル入力上はフラグさえ使えば -1/999 自体に大きな問題はないでしょう（それらの値があっても is_valid=0なら学習時無視される想定）。とはいえ、**フラグを使わずにモデル投入する場合は -1/999 などが外れ値になりかねない**ため注意が必要です。本パイプラインでは幸い各フラグが仕様として定義されており [120] [121]、Nullやダミー値との組み合わせでデータ状態を正確に示す設計になっています。

(b) データの正規化 – 特徴量のスケーリングや正規化も適切に対処されています。

- **時系列方向の正規化**: Zスコア化は前述の通り、過去一定期間内での偏差として市場・フロー・信用残それぞれ導入済みです（期間長も適切）。また、出来高で割るスケーリングも信用残で導入され、規模のばらつきを吸収しています。これらは**同一銘柄（または同一セクション）の歴史的基準での正規化**であり、将来情報を参照しない安全な方法です。

- **クロスセクション（横断的）な正規化**: 本データセットでは、生データに対する**日ごとの横断的な標準化は敢えて行っていません** [122]。理由は、横断的Zスコアなどは全銘柄の値をその日中で集計するため、将来的にモデルを適用する際にはデータリークにつながるリスクがあるからです（訓練集合全体の平均・分散で標準化すると、その統計量にテスト期間の情報が含まれ得る）。実際、ドキュメントでも「**クロスセクショナルなZ変換やセクター内正規化は学習時の訓練ウィンドウ内で行う**」と明記されています [123]。従って、本パイプラインの出力データ自体には横断的標準化済みの列は含まれません（セクター相対項目など未実装のため余計に）。モデル構築者が必要に応じて訓練データ内で正規化する余地を残している設計と言えます。これは**データリーク防止の観点から極めて重要であり妥当な判断**です。

- **対数変換**:　価格変化には対数リターンも計算済みです [3] 。`log_returns_*d`列はClose価格のログ差で計算され、超過的な値動きに対してロバストな特徴となります。これも正規化の一種で、特に**非対称な分布を対数で安定化する効果**が期待できます。

(c) 未来情報混入の防止 – 前述の結合処理で詳細を述べた通り、本パイプラインでは**各データがその時点までに入手できた情報のみに基づくよう時系列をシフト**しています [98] 。具体例を挙げます:

- 決算情報は引け後なら翌日からしか反映しません [82] 。
- 投資部門別フローは、次の公表があるまで前回の値を保持しますが、それ以降の日には適用せずNull化します [94] 。
- 信用残も発表日翌営業日から一週間データを保持し、次回更新後は新データに切り替わります [107] [124] 。

株価から計算される特徴量においても、**未来日の株価を参照する処理はありません**。例えばリターンや移動平均計算はいずれも過去方向（`pct_change`や`rolling_mean`）で行われ、未来方向の`shift(-n)`等は使われていません [10] [125] 。クロス特徴量についてβ計算で市場リターンに1日ラグを入れるなど、むしろ**保守的に未来を見ない工夫**が見られます [37] 。

以上の点から、本データセットに**将来のターゲット情報が直接・間接に漏洩する箇所は見当たりません**。すべての結合は「as-of　(〜時点で利用可能な最新データ)」で行われており、時間順序の整合性が保たれています [126] 。

(d) データ整合性チェック – パイプライン終盤では、出力データフレームの整形と検証が行われています。主な処理は以下です:

- **カラム名・スキーマ統一**: 別名の揺れを全て公式ドキュメント準拠にリネームした後 [127] [115] 、想定すべき全カラムが揃っているかチェックしています。欠けている必須列があればNullで追加し、逆に余分な列はドロップして、**カラム順もドキュメント（DATASET.md）通りにソート**しています [128] [129] 。これにより、例えばoptionalな信用残を除くと常に決まった列数・列順のパーケットが得られます。
- **重複キー排除**: 念のため`(Code, Date)`の組が重複していた場合は後勝ちで重複を取り除く処理も入っています [130] （通常は発生しないはずですが、コード正規化後などで重複が起きた際の安全策）。
- **統計出力**: SafeJoinerやValidityFlagManagerの各所で結合カバレッジ（何%の行にデータが付いたか）や有効フラグの割合等をログ出力しています [131] [132] 。これはデータ品質を把握しリファインするのに有用です。実際、レポートによればフローのカバレッジや有効行数、ステートメントの有効割合などが計算されています [131] [133] 。

これらの措置により、**出力データセットはドキュメント定義と齟齬のない構造**となっており、またキーの一意性やNullフラグの扱いも明確です。**総じてデータの整合性は高く保たれている**と言えるでしょう。

4. 問題点と改善提案

調査の結果、本パイプラインは**日本株の包括的な特徴量エンジニアリングをリーク無く実現しており、現状大きな不備は見当たりませんでした**。計算式や結合ロジックも概ね妥当で、モデル学習に供するデータセットとして品質は高いです。その上で、わずかながら以下の改善・追加の提案を挙げます。

- **セクター特徴量の実装**: 現状スタブになっている業種別特徴を実装すれば、更なる性能向上が期待できます。特に**33業種のワンホット**や**業種指数との相対リターン**、**ターゲットエンコーディング**はモデルにセクター情報を織り込む上で有用でしょう。実装に当たっては、例えば**業種平均リターン**を当日値

で結合すると横断的リークになるため、**一定遅行させる**か**学習内処理**に留めるなど工夫が必要です（ドキュメントで言及の「cross-fit + lag」の方針 [80] を踏襲）。

- **テクニカル指標の有効フラグ充実**: `is_ema200_valid` の算出ロジックが見当たらないため、追加を検討すべきです。例えば `row_idx >= 200` でフラグを立てれば、200日未満のEMA200は無効と扱えます。現状EMA200自体は常に計算されますが、初期199日間は急峻に変動するため学習時に無効化できると望ましいです。同様に、EMA60についても60日未満期間では `is_valid_ma` で包括的にカバーされていますが、厳密には `is_ema60_valid` があっても良いかもしれません（もっとも60日は `is_valid_ma` に包含されているため実害は少ないでしょう）。また、RSI2の有効化条件5日は少し保守的にも感じます。RSIの計算自体は2日あればできますが、初期値安定のために5日見ているようです [13]。この閾値設定もチューニング余地があります。

- **フロー/信用残の欠損値扱い統一**: フローでは `days_since_flow` を-1/999で埋め、信用残では `margin_days_since` をNullのままとしています。この統一感の無さはモデル入力処理で戸惑う可能性があります。両者Null扱いで統一するか、あるいは信用残も `days_since` を例えば999で埋めておくなど、一貫させると分かりやすいです（ただしNullならフラグ見れば済むので大きな問題ではありません）。一方で、**フローの-1埋め**はValidityFlagManager内の条件とも整合しておらず（0以上が有効条件 [134]、-1は未経験）、-1は結局is_flow_valid=0にしかならないため、999埋めと機能上差がありません。ここは-1か999かに統一するか、いっそNullのままにして `is_flow_valid` だけ見れば良いようにも思われます。モデル入力時の扱いやすさを考え、**可能ならNull統一＋フラグで対応**が直感的でしょう。

- **ADX計算の微調整**: ADXのWilder平滑をEWMAで近似していますが、より厳密にやるなら初期14日間は単純平均、その後は前回値との組み合わせで計算する実装も考えられます。ただ、大勢に影響はないため優先度は低い改善点です。

- **ターゲット値の計算・確認**: 本パイプラインは特徴量生成が中心ですが、データセットには将来リターンをラベルとして含めています [135]。通常、例えば `target_5d` は (未来5日後のClose / 当日Close - 1) を意味するはずですが、その計算部分がコード上では見当たりませんでした。おそらくパイプラインV4（JQuantsPipelineV4Optimized）内で生成しているか、あるいは出力後に別スクリプトで付与している可能性があります。**未来リターン算出時はデータリークにならないよう、最終日分のターゲットをNullにする**等の配慮が必要です。もし未実装であれば、`pl.col("Close").pct_change(n)` 等でシフト(-n)して計算し、未来情報が入らないよう慎重に追加してください。

- **ログと統計の活用**: パイプライン中で出力しているカバレッジ率や有効行割合のログは、定期的にチェックしてデータの健全性をモニタすると良いでしょう。例えば**財務データのカバレッジが低下していれば最新期間でデータ取得漏れが起きていないか確認**する、**is_flow_validが極端に0ばかりならセクションマッピングに問題がないか検証**するといった形です。これはコード改善というより運用上の提案です。

以上の改善提案はいずれも致命的な不具合ではなく、**データセットのさらなる品質向上や将来の拡張を見据えたもの**です。現在の実装は総じて堅牢であり、各処理がドキュメント化された仕様 [112] [136] に忠実に沿っている点は特筆に値します。今後はこの基盤を維持しつつ、さらなる特徴量（セクター情報など）の追加やフラグ管理の洗練によってモデル性能最大化を図っていけるでしょう。

参考資料:

- Gogooku3 データセット仕様書 (DATASET.md) [137] [138]
- フロー・信用残の計算ロジック実装コード [51] [67]
- パイプライン全体処理コード (full_dataset.py) [139] [140]

• 安全結合処理 (SafeJoinerV2) 86 93 （リーク防止のas-of結合）

---

1 2 run_full_dataset.py
https://github.com/wer-inc/gogooku3/blob/11e828b7d27488a6c6e5c4af04da0d37d7fd6438/scripts/pipelines/run_full_dataset.py

3 5 6 8 9 10 11 12 13 15 16 17 18 19 22 49 79 80 114 115 117 125 127 128 129 130 135 139 140 full_dataset.py
https://github.com/wer-inc/gogooku3/blob/11e828b7d27488a6c6e5c4af04da0d37d7fd6438/src/pipeline/full_dataset.py

4 14 34 35 77 78 ml_dataset_builder.py
https://github.com/wer-inc/gogooku3/blob/11e828b7d27488a6c6e5c4af04da0d37d7fd6438/scripts/data/ml_dataset_builder.py

7 24 70 76 84 98 112 113 120 121 122 123 126 136 137 138 DATASET.md
https://github.com/wer-inc/gogooku3/blob/11e828b7d27488a6c6e5c4af04da0d37d7fd6438/docs/DATASET.md

20 21 23 25 26 27 28 29 30 31 32 33 36 37 38 39 40 41 42 43 44 45 46 47 48 market_features.py
https://github.com/wer-inc/gogooku3/blob/11e828b7d27488a6c6e5c4af04da0d37d7fd6438/src/features/market_features.py

50 51 52 53 54 55 56 57 58 59 87 88 89 90 91 92 97 99 100 101 116 flow_joiner.py
https://github.com/wer-inc/gogooku3/blob/11e828b7d27488a6c6e5c4af04da0d37d7fd6438/src/features/flow_joiner.py

60 61 62 63 64 65 66 67 68 69 71 72 73 74 75 103 104 105 106 107 108 109 110 111 124 margin_weekly.py
https://github.com/wer-inc/gogooku3/blob/11e828b7d27488a6c6e5c4af04da0d37d7fd6438/src/gogooku3/features/margin_weekly.py

81 82 83 85 86 93 94 95 96 102 131 132 133 safe_joiner_v2.py
https://github.com/wer-inc/gogooku3/blob/11e828b7d27488a6c6e5c4af04da0d37d7fd6438/src/features/safe_joiner_v2.py

118 119 134 validity_flags.py
https://github.com/wer-inc/gogooku3/blob/11e828b7d27488a6c6e5c4af04da0d37d7fd6438/src/features/validity_flags.py