ChatGPT

Gogooku3 多ホライズン株価予測パイプライン設計

1. 使用するAPIデータとMLにおける役割

J-Quantsが提供する**株式関連APIデータ**を総合的に利用し、多ホライズン（1日後・5日後・10日後・20日後）の株価リターン予測モデルを構築します。それぞれのデータ種別と機械学習上の役割は以下の通りです。

- **株価データ（時系列価格・出来高）** – **日次株価OHLCV**（Open, High, Low, Close, Volume）データは、価格変動そのものを捉える基本情報です [1]。価格データから**モメンタム**（過去リターン）や**ボラティリティ**、移動平均などの**テクニカル指標**を算出し、短期的なトレンドやリバーサル傾向を特徴量化します [2] [3]。

- **市場指標データ（指数系）** – **TOPIXなどの株価指数**データは、マーケット全体の動向やレジーム（強気/弱気相場、高ボラティリティ局面など）を捉える特徴量を提供します [4] [5]。指数データから**市場リターン**や**市場ボラティリティ**、**指数の移動平均トレンド**、**市場レジームフラグ**などを計算し、個別銘柄の値動きを市場全体と比較したり [6] [7]、**インデックス裁定効果**（市場との乖離やα成分）を特徴量化します。

- **投資部門別売買フロー** – **投資主体別売買動向（週次集計）**データ [8] は、国内外の個人・機関投資家の資金フローや需給バランスを示します。これを日次に展開して**資金流入出の割合**や**“スマートマネー”指数**を算出し、需給面から株価へのプレッシャーや市場センチメントを特徴量として取り込みます [9] [10]。特に外国人 vs 個人の売買バランスや、総売買代金の異常値検知（フローショック）は需給面のサプライズ指標となります。

- **財務業績データ（企業ファンダメンタルズ）** – **四半期財務諸表**データは企業の業績や財務状況を表し、中長期の価値評価に関わります。最新の決算発表データを**遡及的に各営業日に紐付け**ることで、株価に織り込まれたファンダメンタル情報（売上・利益の成長率、利益率、進捗率、ガイダンス修正等）を特徴量化します [11] [12]。これにより、企業価値やサプライズ（予想との差異）が株価リターンに与える影響をモデルが捉えられるようになります。

- **銘柄属性・イベントデータ** – **上場銘柄情報**（市場区分や業種セクター等）や**企業イベント**（上場・廃止、市場変更、決算発表タイミングなど）は、モデルの入力データ管理と特徴量生成の両面で重要です。例えば市場区分や業種は各銘柄の**静的特徴（セクター特性）**として扱い、必要に応じてダミー変数化や同業種比較に利用します。またIPOや上場廃止といったイベントは銘柄ごとの生存期間を管理し、予測時の銘柄ユニバースを正確に保つために使用します（モデル入力自体には直接使わない場合もありますが、**イベントフラグ**として特殊な相場影響を捉えることも検討できます）。決算発表日は **stmt_imp_statement** フラグで特徴量化し、その日を境にファンダメンタル指標が更新されることをモデルに伝えます [13]。

- **先物・オプションなどのデリバティブ**（必要に応じて） – J-Quants APIが提供する**指数先物**などのデータがある場合、例えば**日経先物の夜間変動**を翌日の先行シグナルとして使うなど、追加の市場先行指標を特徴量に組み込むことも可能です。こうしたデータは短期予測の精度向上や**オーバーナイトの情報反映**（先物市場での先行価格発見）をモデルに織り込むのに役立ちます。

以上のように、価格・市場・フロー・ファンダメンタル・イベントなど**多様なデータソース**を組み合わせることで、それぞれが表す「異なる角度の情報」（価格動態、需給、企業価値、市場全体動向など）を網羅し、モデルの予測性能を最大化します。

2. データソース別の特徴量一覧

各データソースから生成される主な特徴量と計算方法、適用タイミングを以下にまとめます。特徴量は**計算式とともにカラム名**を示し、必要な場合はその有効期間や算出に必要なデータ履歴（日数）を併記します。

2.1 株価系（価格・テクニカル）特徴量

データソース: 日次株価OHLCV（J-Quantsの `/prices/daily_quotes` 等）

- **基本価格情報 (OHLCV)** – `Open, High, Low, Close, Volume` は調整後株価と出来高 [1] 。株式分割などの調整考慮済み価格を使用します。`TurnoverValue`（出来高×株価, 売買代金）は取得可能なら利用し、なければ欠損とします [14] 。これらは当日の基礎情報で、**当日中に既知の値**として扱います（リークの心配はありません）。

- **リターン** – 過去の価格から計算するリターン系列。例: `returns_1d = Close/Close[-1] - 1`（1営業日リターン）、`returns_5d, returns_10d, returns_20d, returns_60d, returns_120d` はそれぞれ過去5日、10日、20日、60日、120日の終値リターン [15] 。対数リターンも併用し、`log_returns_1d = ln(Close/Close[-1])`（5日・10日等も同様）として計算します [16] 。**使用タイミング**: 過去履歴がない初期数日間は計算不可のためNaNになります（最低計算に必要な履歴日数: 例えば5日リターンには5営業日分の価格）。

- **ボラティリティ** – リターンの標準偏差から算出します。`volatility_5d = std(returns_1d, 5) * sqrt(252)` のように過去5日間の1日リターンの標準偏差を年率換算 [3] 。10日、20日、60日についても同様に計算します。`realized_volatility`（実現ボラティリティ）は例えば日中高値と安値から近似計算（Parkinsonの式など）も検討可能 [17] 。**使用タイミング**: リターンと同様、計算に窓長（日数）が必要なので、各銘柄について初期のうちはNaNとなり、十分な履歴が溜まった時点で値が有効になります（有効フラグで管理）。

- **移動平均（MA）・指数平滑移動平均（EMA）** – `sma_{5,10,20,60,120}` は単純移動平均（過去w日間のClose平均）、`ema_{5,10,20,60,200}` は指数移動平均 [18] 。長期の200日EMAも算出します。**使用タイミング**: 各MA計算には対応する履歴（例えば200日EMAには少なくとも200営業日の価格履歴）が必要で、不足する期間はNaNです。

- **価格位置・ギャップ系** – 移動平均や日中値幅に対する現在価格の位置関係を表す特徴量です [19] 。例: `price_to_sma5 = Close / sma_5`（短期平均比の位置）, `ma_gap_5_20 = (ema_5 - ema_20) / (ema_20 + 1e-12)`（短期EMAと中期EMAの乖離率） [20] , `high_low_ratio = High / (Low + 1e-12)`（当日高値と安値の比）, `close_to_high = (High - Close) / (High - Low + 1e-12)`（当日レンジ内での終値の位置）など。**使用タイミング**: これらは当日までのデータで算出可能ですが、移動平均に依存するものは該当窓長の履歴が必要。

- **出来高・回転率系** – 出来高のトレンドや規模感をとらえます [21] 。`volume_ma_5, volume_ma_20` は出来高の5日・20日平均、`volume_ratio_5 = Volume / volume_ma_5`（直近出来高の短期平均比）, `volume_ratio_20 = Volume / volume_ma_20`（中期平均比） [22] 。さらに `turnover_rate = Volume / (shares_outstanding + 1e-12)` で株式の回転率（発行株数に対する出来高割合）を算出（発行株数データが得られる場合）。`dollar_volume = Close * Volume`（売買代

金）も参考指標として含めます。**使用タイミング**: これらは当日データから算出（回転率は当日株数を用い当日算出）しますが、移動平均系は履歴依存。

- **テクニカル指標**（pandas-ta等ライブラリ活用） – 外部ライブラリで計算する高度なテクニカル指標を追加します [23]。例:

- **相対力指数 (RSI)**: `rsi_14`（14日）, `rsi_2`（2日）とし、さらに`rsi_delta = rsi_14.diff()`でRSIの変化量も特徴にします [23]。短期RSIは**短期リバーサル**を捉える指標です。
- **MACD**: `macd`, シグナル線 `macd_signal`, ヒストグラム `macd_histogram`（例: MACD 12-26-9）[24]。トレンドの勢いと転換点を示す指標。
- **ボリンジャーバンド**: `bb_upper, bb_lower, bb_middle`を内部的に計算し、バンド幅や位置として`bb_width = (bb_upper - bb_lower) / (Close + 1e-12)`, `bb_position = (Close - bb_lower) / (bb_upper - bb_lower + 1e-12)`を特徴量化 [25]。バンド幅は**ボラティリティの相対的な広がり**（スクイーズ検知）、位置は価格水準の極端さを示します。

- **ATR/ADX/ストキャスティクス**: `atr_14`（14日平均的なレンジ幅）, `adx_14`（14日ADXトレンド強度指標）, `%K (stoch_k)`（ストキャスティクス）等 [26]。**使用タイミング**: いずれも必要な履歴長があるため、その間はNaNとなり、値が計算可能になってから使用します。

- **価格系有効フラグ** – 上記の価格系特徴量には、十分な履歴が無い期間にNaNが生じます。**データリークを防ぎモデルで適切に無効化**するために、有効フラグを付与します [27]。例えば `is_rsi2_valid = (row_idx >= 5)` は5日以上経過してRSI2が計算可能かを示すフラグ、`is_ema200_valid = (row_idx >= 200)` は200日以上経過した銘柄かどうか（200日EMA利用可否）を示します [28]。`is_valid_ma` など総合的な有効フラグも用いて、不十分なウォームアップ期間のデータをモデル学習時にマスクします。

2.2 市場指数特徴量（マーケット全体指標）

データソース: 株価指数（例: TOPIXの日次OHLCVデータ、J-Quantsの `/indices/topix` 等）

- **市場リターン** – TOPIX終値から市場全体のリターンを計算します。`mkt_ret_1d = IndexClose / IndexClose[-1] - 1`（市場1日リターン）、`mkt_ret_5d, 10d, 20d` も同様に算出 [6]。モデルには各銘柄に対し同日の市場リターンを特徴量として付与します（全銘柄同じ値を持つ特徴量） [29]。**使用タイミング**: 市場リターンは当日までの指数で算出でき、各日共通の値として**Dateキーで結合**します [30]。

- **市場トレンド（移動平均）** – 市場指数のEMAを複数計算し、市場のトレンドを定量化します。`mkt_ema_5, mkt_ema_20, mkt_ema_60, mkt_ema_200` はそれぞれTOPIX終値の5日・20日・60日・200日指数移動平均 [31]。**使用タイミング**: 長期EMA（200日など）は初期に値が安定しないため、指数データ取得時に**長めのウォームアップ**（過去252日以上）を確保し、結合時に初期NaNが発生しにくいようにしています [32]。

- **市場乖離・ギャップ** – 市場指数における短期トレンドと中期トレンドの差異や傾きを表します。例: `mkt_dev_20 = (IndexClose - mkt_ema_20) / (mkt_ema_20 + 1e-12)`（指数の20日トレンドからの乖離）や、`mkt_gap_5_20 = (mkt_ema_5 - mkt_ema_20) / (mkt_ema_20 + 1e-12)`（5日と20日EMAの差の割合） [33]。`mkt_ema20_slope_3 = pct_change(mkt_ema_20, 3)` は20日EMAの3日間の変化率で、市場トレンドの加速/減速を示します [33]。

- **市場ボラティリティ・レンジ** – 市場全体の変動率や価格帯の幅を示す特徴です。 [34] 参照。`mkt_vol_20d = std(mkt_ret_1d, 20) * sqrt(252)`（市場リターンの20日年率ボラ） [34]。

`mkt_atr_14`は14日間の平均的な指数レンジ幅(ATR)、`mkt_natr_14 = mkt_atr_14 / (IndexClose + 1e-12)`で価格水準に規模正規化したATR、`mkt_bb_pct_b`（ボリンジャーバンド%BB）は市場指数の位置、`mkt_bb_bw`（バンド幅）は市場ボラティリティの相対指標[34]。

- **市場リスク指標** – 極端な市場状況を表すフラグや指標です。[35]参照。`mkt_dd_from_peak = (IndexClose - cummax(IndexClose)) / cummax(IndexClose)`は直近ピークからのドローダウン率で、市場が何%下落しているかを示します[35]。`mkt_big_move_flag = (abs(mkt_ret_1d) >= 2 * std(mkt_ret_1d, 60)).int8()`は直近60日間のボラの2倍を超える“異常値動き”が起きた日のフラグです[35]。

- **市場Zスコア** – 市場指標をさらに長期過去で標準化したものです[36]。例えば`mkt_ret_1d_z = z(mkt_ret_1d, 252)`は過去1年の1日リターン分布から見た当日リターンの偏差、`mkt_vol_20d_z`は20日ボラの偏差といった具合です[36]。`mkt_bb_bw_z`（バンド幅z）、`mkt_dd_from_peak_z`（ドローダウンz）も算出しています。**使用タイミング**: 252日以上の指数履歴がないとNaNになるため、指数データ取得時には>252日遡って取得し日付結合時に不足を補います[32]。

- **市場レジームフラグ** – 市場環境を離散的に表すバイナリ特徴量です[37]。例: `mkt_bull_200 = (IndexClose > mkt_ema_200).int8()`は200日線より指数が上→強気相場フラグ、`mkt_trend_up = (mkt_gap_5_20 > 0).int8()`は短期>中期トレンド→上昇トレンドフラグ、`mkt_high_vol = (mkt_vol_20d_z > 1.0).int8()`はボラ急騰局面、`mkt_squeeze = (mkt_bb_bw_z < -1.0).int8()`はボラ極小局面（バンド幅縮小）を示します[37]。これらは市場全体の状態を表し、個別銘柄がどの環境下にいるかモデルに認識させます。

2.3 クロス特徴量（個別銘柄 × 市場）

データソース: 個別銘柄株価データ＋市場指数データの組み合わせ

個別銘柄の動きを市場ベンチマークと比較することで**銘柄固有の要因（アルファ）**を抽出します。**キー**は`(銘柄Code, Date)`で、当日の銘柄と市場データから計算します。主なクロス特徴量は8種類です:

- **ベータ & アルファ**:
- `beta_60d = Cov(returns_1d, mkt_ret_1d, 60) / Var(mkt_ret_1d, 60)` – 過去60日間の銘柄リターンと市場リターンの共分散を市場分散で割った**60日ベータ値**です[38]。市場感応度（ベータ）がどれくらいか示します。
- `alpha_1d = returns_1d - beta_60d * mkt_ret_1d` – 当日リターンから市場の影響分を引いた**1日アルファ**[39]。`alpha_5d`も同様（5日リターンから市場ベータ分を控除）。アルファは市場に対する超過リターンで、銘柄固有の上昇/下落要因を表します。

- `beta_stability_60d = 1 / (std(beta_60d, 20) + 1e-12)` – ベータ値の直近20日間の標準偏差の逆数です[40]。ベータが安定していれば値が大きく、不安定なら小さくなる**ベータ安定度指標**で、銘柄のリスク特性変化を捉えます。

- **相対パフォーマンス**:

- `rel_strength_5d = returns_5d - mkt_ret_5d` – 銘柄5日リターンから市場5日リターンを差し引いた**相対リターン**[41]。プラスであれば直近で市場をアウトパフォームしていることを意味し、マイナスならアンダーパフォームです。

- `idio_vol_ratio = volatility_20d / (mkt_vol_20d + 1e-12)` – 個別銘柄の20日ボラティリティを市場20日ボラで割った**特有ボラ比率**[42]。1より大きければ市場よりボラが高く、低ければ安定的と判断できます。

- **市場との位置関係**:

- `trend_align_mkt = (sign(ma_gap_5_20) == sign(mkt_gap_5_20)).int8()` – 銘柄の短期トレンド方向 (5日 vs 20日EMAの差) が市場の短期トレンド方向と**同じか**を示すフラグ [42]。1であれば市場と足並みの揃った動きをしており、0なら逆行または独自の動きを示唆します。
- `alpha_vs_regime = alpha_1d * mkt_bull_200` – 市場が強気局面（mkt_bull_200=1）の日における銘柄1日アルファの値 [42]。強気相場下での銘柄独自要因の良し悪しを反映し、弱気局面では0になります（フラグとαの積）。これは**マーケットレジームと組み合わせたアルファ**で、市場環境による影響度合いを特徴づけます。

2.4 フロー特徴量（投資部門別資金流動）

データソース: JPX投資部門別売買状況（J-Quantsの `/markets/trades_spec` 等、通常は**週次集計**）

投資部門別（外国人、個人、投信、信託銀、自己など）の売買**金額・バランス**データから、需給フローの特徴量を日次ベースに生成します。週次データを各営業日にならすために下記手順で処理・結合します [8]:

1. **効果発現期間の設定**: 各週次データには集計期間（通常1週間）と公表日があります。公表日 (PublishedDate) の翌営業日をそのデータの `effective_start`（効果開始日）とし、次の公表日直前営業日をそのデータの `effective_end` と定義します [8]。こうすることで、週次フロー情報が**どの日からどの日まで有効か**を明確にします。各「投資部門セグメント」（例えば市場区分ごと）で、この効果期間が連続するよう管理します。

2. **日次展開**: 全営業日の日付と各投資部門セグメントの組み合わせのグリッドを作り、上記の `effective_start` 〜 `effective_end` の範囲にその週のデータを適用します [8]。結果として各営業日・各セグメントごとに、その日に有効な最新フロー値が割り当てられます。

3. **結合**: 銘柄ごとに自社が属する**市場セグメント（section_norm）**を割り当て、(section_norm, Date) をキーに銘柄データとフロー日次データをLeft Joinします [43]。これにより各銘柄に対して、その銘柄市場区分における投資部門別フロー指標を付与できます。結合後の列名は学習用に統一し、`flow_` プレフィックスを付けます [43]。

週次データの**生データ項目**には、外国人/個人/自己/...の売り・買い・合計・バランス金額（Balanceは買い-売り）、および全体計 (Total) などがあります [44]。これらから計算する特徴量一覧（17項目）は以下の通りです [9] [10]:

- **ネット売買比率**: 各主体の売買バランスを総売買高で割ったもの。例: `flow_foreign_net_ratio = ForeignersBalance / (ForeignersTotal + 1e-12)`（外国人の週間買越額をその主体の総売買高で割った比率） [9]、`flow_individual_net_ratio`（個人の買越比率）も同様に算出。0より大きければ買い越し、負なら売り越し。

- **活動度指標**: 市場全体や特定主体の売買高が平常時と比べて多いか少ないか。`flow_activity_ratio = TotalTotal / (mean(TotalTotal, 52) + 1e-12)` は**市場全体売買代金**が過去52週平均と比べてどれだけ多いか（1以上で平常以上の活況）を示します [45]。※52週（約1年）平均で正規化することで年末年始など季節要因を平滑化しています。`foreign_share_activity = ForeignersTotal / (TotalTotal + 1e-12)` は当該週の総売買高に占める外国人売買の割合です [46]（こちらは補助的指標で、必要なら `flow_` 接頭辞で学習用に追加可能）。

- **ブレッドス指標**: 投資主体全体の買い越し割合を示します。`breadth_pos = mean([ForeignersBalance>0, IndividualsBalance>0, TrustBanksBalance>0,`

`InvestmentTrustsBalance>0, ProprietaryBalance>0, BrokerageBalance>0])` は主要6主体について買越かどうかを1/0で見て平均した値（0〜1） [47] 。例えば0.67なら6主体中4主体が買い越しであったことを示し、広範囲に資金流入かどうかを表します。

- **フローZスコア**: 各主体の買越額や総売買高を52週ベースで標準化したものです [48] 。`flow_foreign_net_z = z(ForeignersBalance, 52)`（外国人買越額の52週Zスコア）, `flow_individual_net_z`（個人買越額Z）, `flow_activity_z = z(TotalTotal, 52)`（市場総売買高のZ）など。平常値に対する異常度を数量化し、他の期間との比較可能にします。

- **スマートマネー指標**: 上記Zスコアを組み合わせ、いわゆる“スマートマネー”（賢明な投資家）と“ダンマリマネー”の動きを捉えます [49] 。`flow_smart_idx = flow_foreign_net_z - flow_individual_net_z` は**外国人と個人の買越動向差**（外国人が買い越し基調で個人が売り越し基調なら正の大きな値）を指数化したものです [49] 。`flow_smart_mom4 = flow_smart_idx - mean(flow_smart_idx, 4)` はそのスマートマネー指数の直近4週に対するモメンタム（上昇ならプラス）です [49] 。また、`flow_shock_flag = (abs(flow_smart_idx) >= 2.0).int8()` はスマートマネー指数が絶対値2以上と大きく偏っている週にフラグを立てます [49] 。これらは需給面での極端な動きを検知する特徴量です。

- **フロータイミング**: 週次フロー情報が更新されたタイミングを示す変数です [50] 。`flow_impulse = (Date == effective_start).int8()` はその日が新しい週次データの有効開始日であるか（≒先週からフローが変化したタイミング）を示すフラグ [50] 。`flow_days_since = (Date - effective_start).days()` は最新データが適用されてから何営業日経過したかを日数で与えます [50] 。これにより、「フロー情報が鮮度何日目か」をモデルに伝え、時間経過による効果減衰などを学習させることも可能です（例えばフロー発表直後はインパクト大など）。

※こうしたフロー特徴量は**銘柄固有ではなく市場セグメント固有**ですが、各銘柄に市場区分（Prime市場、Standard市場等）を割り当てて結合しているため、結果的に**その銘柄に対応する市場の需給状況**として利用できます。期待されるカバレッジは営業日×セグメントの70〜95%程度で値が埋まり、銘柄が属するセグメントさえ適切に対応付けられていれば広範な銘柄で値が利用可能です [51] 。

2.5 ファンダメンタル特徴量（財務諸表）

データソース: 上場企業の財務データ（J-Quantsの `/fins/statements` 等、四半期決算速報値）

各銘柄について四半期ごとの財務指標を、適切な **“as-of結合”**（後述）により日次データセットに組み込み、以下の17種類の財務特徴量を計算します [11] [12] 。基本的に**最新の開示情報に基づき計算され、開示翌営業日以降にその値が反映**されます [52] 。

- **YoY成長率 (前年同期比)** – 四半期業績の対前年比成長を表します。同じ四半期（前年同期）の値と比較するため**4期前との比**を計算します [53] 。例:
- `stmt_yoy_sales = (NetSales - NetSales[-4]) / (abs(NetSales[-4]) + 1e-12)` – 売上高の前年同期比増減率 [53] 。
- `stmt_yoy_op = (OperatingProfit - OperatingProfit[-4]) / (abs(OperatingProfit[-4]) + 1e-12)` – 営業利益の前年同期比。

- `stmt_yoy_np = (Profit - Profit[-4]) / (abs(Profit[-4]) + 1e-12)` – 四半期純利益の前年同期比。 前年からの成長（プラスかマイナスか、その率）がわかり、成長株かどうかの指標となります。

- **利益率（マージン）** – 利益の売上に対する割合です [54] 。

- `stmt_opm = OperatingProfit / (NetSales + 1e-12)` – 営業利益率 [54]。
- `stmt_npm = Profit / (NetSales + 1e-12)` – 純利益率。企業の収益性を示す基本指標です。

- **進捗率** – 通期予想に対する四半期実績の進捗を示します [55]。

- `stmt_progress_op = OperatingProfit / (ForecastOperatingProfit + 1e-12)` – 営業利益の会社予想に対する進捗率 [55]。
- `stmt_progress_np = Profit / (ForecastProfit + 1e-12)` – 純利益の会社予想（当期純利益予想）に対する進捗率。進捗が高いほど業績が予想を上回っている傾向。

- **ガイダンス改定率** – 会社予想（ガイダンス）の修正率を表します [12]。直近の開示で新たに発表された予想値と、その前回開示時点の予想値との差分です。

- `stmt_rev_fore_op = (ForecastOperatingProfit - prev_ForecastOperatingProfit) / (abs(prev_ForecastOperatingProfit) + 1e-12)` – 営業利益予想の前回比改定率 [12]。
- `stmt_rev_fore_np = (ForecastProfit - prev_ForecastProfit) / (abs(prev_ForecastProfit) + 1e-12)` – 当期純利益予想の改定率。
- `stmt_rev_fore_eps = (ForecastEarningsPerShare - prev_ForecastEarningsPerShare) / (abs(prev_ForecastEarningsPerShare) + 1e-12)` – 1株当たり利益(EPS)予想の改定率。
- `stmt_rev_div_fore = (ForecastDividendPerShareAnnual - prev_ForecastDividendPerShareAnnual) / (abs(prev_ForecastDividendPerShareAnnual) + 1e-12)` – 年間配当予想の改定率 [12]。 これらはポジティブなら上方修正（サプライズ好材料）、ネガティブなら下方修正（悪材料）であり、株価へのインパクトが大きい指標です。

- **収益性指標** – 財務の健全性や効率性を表します [56]。

- `stmt_roe = Profit / (Equity + 1e-12)` – 自己資本利益率(ROE) [56]。純利益÷自己資本。
- `stmt_roa = Profit / (TotalAssets + 1e-12)` – 総資産利益率(ROA)。純利益÷総資産。 高いほど効率的に利益を上げていることを意味します。

- **会計上の特殊要因フラグ** – 財務数値の質や継続性に影響を与える会計変更を示すフラグです [57]。

- `stmt_change_in_est = (ChangesInAccountingEstimates ∈ {"true", "1"}).int8()` – 会計上の見積もり変更があった場合1 [57]。
- `stmt_nc_flag = ((ChangesBasedOnRevisionsOfAccountingStandard ∨ RetrospectiveRestatement) == true).int8()` – 会計基準変更や過去遡及修正があった場合1 [57]。 これらは特殊要因による数値の不連続や一過性を示唆し、モデルが過去との比較で誤解しないように役立ちます。

- **決算タイミング** – 決算発表直後かどうか、発表からの日数を示す特徴です [13]。

- `stmt_imp_statement = (Date == effective_date).int8()` – 当日のDateがその銘柄の決算発表日（または発表翌営業日）であれば1 [13]。決算発表タイミングのイベント効果を示す指標です。
- `stmt_days_since_statement = (Date - effective_date).days()` – 最終決算開示からの経過日数 [13]。日が経つほど決算情報の鮮度が落ち影響も薄れる可能性があります。

使用タイミングと結合: 財務データは決算発表（および会社予想更新）があった日に初めて数値が更新されます。**結合ルール**としては各銘柄について `effective_date` を定義し、「当日15:00までに開示があれば当日、

それ以降なら翌営業日をeffective_date」として、その日以降の営業日全てにその開示内容を遡及適用します [52]。日付ごとに各銘柄の最新開示情報をバックフィルし、当日には**最新の1件のみ**を紐付けます [52]。同一銘柄で同日中に複数開示があればタイムスタンプを比較し**最新のみを採用**します [58]。このようにすることで、発表前のデータには将来の決算情報が含まれず、発表直後からその情報が利用可能になるという形で**情報漏洩を防止**しています [52] [59]。

2.6 その他の特徴量・カテゴリ

上記主要データソース由来の特徴量以外にも、必要に応じて以下のような特徴量を検討します。

- **セクター・業種関連特徴**: 銘柄の属する業種（33業種区分など）や市場区分（プライム/スタンダード/グロース）を静的なカテゴリ特徴として扱います。例えばセクターをone-hotエンコードしたダミー変数や、**同業種内での相対指標**（セクター平均との売上成長率差、セクター指数との乖離リターンなど）を特徴量に加えることで、業種固有のトレンドやテーマの影響を織り込めます。将来的にはGraph Attention Networkでセクター内ピア関係をモデル化する構想もあります [60] [61]。

- **信用取引・融資・貸株**: J-Quantsに信用取引残高などのデータが含まれる場合、信用買残・売残やその倍率、回転日数等を特徴量とします。これらは個人投資家の投機的動きを反映し、**信用倍率が高すぎる銘柄の下落リスク**や、信用需給による価格歪みなどを捉えます。信用データは週次更新が多いためフローと同様に日次展開・遡及適用し、`credit_margin_ratio` や `short_balance_rate` 等の指標を計算します（データ未連携の場合は将来拡張項目）。

- **経済・金利・為替指標**: 必要に応じて日経平均、米国株指数、金利や為替などのマクロ指標データも外部データとして組み込みます。例えば長期金利の変化率、為替レートの動向は特定セクター（銀行や輸出関連）の株に影響を及ぼすため、モデルへの追加情報となり得ます。ただしJ-Quants標準API外のデータについては別途取得・結合が必要です。

以上、多岐にわたる特徴量群は**大きくカテゴリ分け**すると、「価格・テクニカル」「市場指標」「銘柄×市場クロス」「フロー（需給）」「財務（ファンダメンタル）」「イベント/タイミング」「セクター/属性」のようになります。それぞれのカテゴリ数はおおよそ以下のような規模感です [62]:

- 価格・テクニカル: 約80列 (価格5, リターン10, ボラ5, 移動平均10, 価格ギャップ8, 出来高6, テクニカル15 など)
- 市場（TOPIX）: 26列 [63]
- クロス: 8列 [64]
- フロー: 17列 [64]
- 財務: 17列 [64]
- フラグ類（成熟度/有効フラグ）: 8列
- ターゲット（目的変数）: 7列

総計で**約160〜170列規模**の特徴量となります [62]。

3. 安全なデータ結合ルール

複数ソースから取得したデータを一つの学習用データセットに統合する際には、**結合キーの統一**と**情報漏洩防止**のための時系列合わせが重要です。

- **結合の主キー**: 基本は**日付(Date)と銘柄コード(Code)**の組み合わせを一意キーとします [65]。価格・テクニカル特徴量やターゲットはこのキー単位で計算されます。市場指数データは銘柄に依存しないため**Dateで全銘柄に一様に結合**します [29]。投資部門フローは市場区分セグメントごとの値を**(Date,**

市場区分)キーで持ち、銘柄ごとに自分の市場区分を参照して結合します [8] [43] 。財務データは**(Code, 開示日)**単位で管理し、これを各Dateに遡及させて結合します（後述）。

- **取引カレンダーの統一**: 日付の結合は日本の営業日ベースで行います。J-QuantsのTrading Calendar APIを用いて全データで共通の営業日カレンダーを使用し [66] 、休日のズレ等を排除します。全てのDateは型を統一して（PolarsのDate型など）扱い、結合時の型不一致による不具合を防ぎます [30] [67] 。

- **Listed銘柄リストの管理**: 上場銘柄のUniverseは日々変化しうるため、**銘柄マスター (listed info)**を活用してその日存在する銘柄だけを対象とします。IPO銘柄は初登場日から、上場廃止銘柄は消滅日翌日以降はデータから除外されます [68] 。銘柄リスト取得API(`/listed/info`)を定期取得し、**営業日ごとの銘柄リスト**を管理します [69] 。結合ではまず価格データ（日次株価）にその日の銘柄Universeを反映させ、存在しない銘柄のレコードは持たないようにします（=株価がなければそのDate×Code行自体を生成しない）。これにより、後段の結合でも存在しない銘柄への不要な紐付けが発生せず、データ漏洩や不整合を防ぎます。

- **財務データのas-of結合**: 財務（決算）データは**後向き結合（backward as-of join）**を採用します [70] 。具体的には、各決算情報に`effective_date`を付与し、その銘柄に対して**その日以降のデータに適用**します [52] 。結合キーは概念的に`(Code, Date)`ですが、内部では「あるDateの時点で、そのCodeの最新開示情報を1件紐付ける」という処理です [52] 。Polars等では`join_asof`機能や、Dateでソートした上で各銘柄について直近の開示をlook-backする処理で実装します。さらに、**発表当日の午後発表分**は翌営業日から有効にするため、`DisclosedTime`を見て15時以降ならeffective_dateを翌日に繰り下げるといったルールを入れます [52] 。このようにして、例えば決算発表前日のデータにはその決算の値は未結合（NaN）で、発表日翌日から埋まる形となり、将来情報の混入を防ぎます [52] [59] 。同一銘柄で複数の開示が近接する場合も、必ずその時点最新のもの一つだけを残す（古い予想は置き換えられる）ようにします [58] 。

- **フローデータの期間展開と結合**: フロー（投資部門別売買）は週次集計値を日次にコピーして使うため、**日付での範囲結合**を行います。事前に各データ行に`effective_start`と`effective_end`を付与した上で、銘柄データとの結合時には「銘柄の属するsectionが同じで、Dateがその範囲内」という条件で結合します [8] [43] 。実装的には、全Date×sectionのデータフレームを用意してフロー値を期間フィルタで当てはめ、それを銘柄DataFrameにsectionキーで結合します。さらに、マーケット区分（section）自体も銘柄属性としてTime Machineに保存し、**各日付における銘柄の市場区分**を適切に扱います（IPO等で市場区分変化があればそれもイベント検知して反映）。こうした結合で、フロー値が存在しない組み合わせ（例: マザーズ銘柄にプライム市場フローを結合しない等）を防ぎます [71] 。また**“フロー0%問題”**として、フロー値が0でもNaNでも意味合いが異なるため、データ取得段階で0は0として保持しNaNとは区別します [72] 。結合後にNaNとなるのは「その日にそのセクションの値が存在しない場合」のみであり、これも有効フラグ`is_flow_valid`で管理します [73] 。

- **その他結合ルール**: 株価と指数は同日のDateで単純結合しますが、**Beta等のクロス特徴量**は計算時にラグが必要な場合があります（例えば当日リターンと当日市場リターンから算出するとき、厳密には当日終値同士なのでリークではないですが、Beta計算では過去60日間のデータのみ利用）。念のため、市場データ由来の特徴量で当日終値を使うものは**同日中に確定しうる情報**のみを使用します。またモデル入力前に**ターゲットとの時間ずれ**を再確認し、ターゲット期間にかかる情報（例えば5日先リターンを予測するのにその5日間の平均を特徴量に含めてしまう等）が含まれないことを検証します。

- **重複・欠損チェック**: 結合後のデータセットでは、主キー(date, code)の重複が無いこと [74] 、主キーにnullが無いこと [75] を確認します。重複行は結合キーの設定ミスや同一データの多重結合を意味するため、発見した場合は結合ロジックを見直します。欠損については後述の欠損処理戦略に従い、必

要ならフラグ付与やImputeを行いますが、**ID項目(date, code)の欠損はあってはならない**ため、発見時はデータ不整合として扱います [76] [75]。

以上のように、各データソースごとに**適切なキー設定と時系列方向の取り扱い**を工夫することで、データ結合時の情報リークを防ぎつつ、一貫したパネルデータセットを構築します。

4. 特徴量カテゴリとグルーピング

前述の特徴量群は、性質に応じて以下のような**カテゴリ**に分類できます。それぞれのカテゴリ内でスケール感や意味が近いため、モデル解釈や分析時に役立ちます。

- **価格系特徴量** – 個別銘柄の過去価格推移に由来する特徴量群です。リターン系列、ボラティリティ、移動平均・乖離、出来高指標、各種テクニカル指標（RSI, MACD, ボリンジャーバンド等）を含みます。短期のモメンタムやリバーサル、テクニカルな**パターン認識**に寄与します。

- **市場・マクロ特徴量** – 市場全体や経済環境に関する特徴量です。TOPIXなど株価指数由来の市場リターン・市場トレンド・市場ボラ指数、レジームフラグなど [4] [77]、さらに必要に応じて金利・為替・海外指数などマクロ指標を含みます。**マーケットベータや相場全体の方向性**をモデルに与え、個別銘柄固有要因と市場要因の分離に役立ちます。

- **クロス特徴量（市場連動）** – 個別銘柄と市場指標を組み合わせて得られる特徴量群です。β値、アルファ、相対リターン、特有ボラなど、**市場と比した銘柄の特性**を示します [40]。市場アノマリーやインデックス効果、銘柄の独立性をとらえるカテゴリです（「指数裁定」効果もここに現れ、市場との乖離がある銘柄はやがて収斂する等の動きを捉えます）。

- **ファンダメンタル特徴量** – 財務諸表やバリュエーション関連の特徴量です。売上・利益の成長率、収益性指標、予想の修正率、進捗、ROE/ROAなど [11] [12]。企業の**バリューや成長性、業績サプライズ**を示し、中期的な株価の方向性（割高/割安や業績好調/不調）に関与するカテゴリです。

- **フロー特徴量** – 投資部門別の売買フローおよび需給に関する指標群です。外国人/個人の売買動向比率、総売買高のZスコア、スマートマネー指数、ブレッドス指標など [9] [10]。**需給面の歪みや相場の過熱感/冷え込み**を捉えるカテゴリで、短期的な値動きの推進力や転換点を示唆します。

- **イベント・タイミング特徴量** – 特定イベントの発生や時間経過を示す特徴です。決算発表日フラグ・経過日数 [13]、フロー更新日フラグ・経過日数 [50]、IPO直後かどうか、上場廃止直前か、株式分割実施日か等のイベントも含み得ます。**イベントドリブンな価格変動**（例: 決算直後のギャップ）や**時間による情報減衰**を扱うカテゴリです。

- **セクター・属性特徴量** – 銘柄固有の静的属性や、それに基づくグルーピング指標です。市場区分（市場第一部/プライム等）や業種、規模指数（大型/中型/小型）、浮動株比率など。モデルにはone-hotや数値ラベルエンコードで与える他、**セクター平均やセクター指数との乖離**を動的特徴量として加えることも考えられます。同セクター内での相対的立ち位置（ピア比較）はGraph構造で捉えることも可能ですが、まずは特徴量として**「セクター平均比業績」「セクター指数に対するリターン差」**などを導入できます。

- **先物・オプション指標**（任意） – 日経225先物やVI（ボラティリティ指数）など、市場先物・オプション由来の指標です。先物夜間変動率やボラティリティ指数(VIX/VNKY)は翌日の株価に先行する情報を持つため、**オーバーナイトの情報反映**カテゴリとして追加可能です。ただしデータソースがJ-Quantsに無い場合は外部取得が必要になります。

各カテゴリごとに特徴量をグルーピングして扱うことで、**特徴量選択やモデル可視化の際に解釈しやすく**なります。またカテゴリ別に正規化や圧縮（例えば主成分分析で次元削減）を検討することで、同じカテゴリ内の共線性対策も可能です。

5. 欠損値の処理戦略

金融データは種々の理由で欠損(NaN)が発生します。適切な欠損処理によりモデルへの悪影響を避け、また欠損自体を情報として活用します。主な戦略は以下の通りです。

- **ウォームアップ不足によるNaN**: 長期移動平均や長期リターン・ボラティリティなどは、新規上場直後や計算期間不足時にNaNになります [71]。この場合は無理に値を補完せず、その期間中は**該当特徴量はNaNのまま**とします。そして併設する**成熟フラグ**（例: `is_ema200_valid` 等）が0であることでモデル側で「信頼できる値ではない」と認識できるようにします [28]。モデル学習時には、このフラグをマスクに用いるか、欠損値処理として**NaNをゼロ等に埋める前に無効フラグも入力**し、モデルが自動的に無視できるようにします。

- **財務データの欠損**: 決算発表がまだ行われていない期間（特にIPO直後数ヶ月など）や四半期間の未発表期間では財務特徴量はNaNになります [78]。これも将来情報漏洩を避けるため意図した挙動であり、**欠損のまま保持**します [78]。発表が無い＝情報が無いこと自体がモデルにとっては「決算未発表期間」という状況として捉えられます（例えば `stmt_days_since_statement` が大きい値になる）。必要であれば「財務情報未取得フラグ（is_stmt_valid）」でマスクします [73]。

- **フローデータの欠損**: 投資部門別フローについて、ある銘柄が属する市場セグメントにデータが存在しない場合（例えばジャスダック等非対象市場の場合や、過去の一部期間）、NaNになります。この場合、**セグメント自体を外す or 別途扱う**必要がありますが、前提として主要3市場（プライム・スタンダード・グロース）のデータをカバーしているため、大半の銘柄では継続的に値があります。局所的なNaNはそのまま保持し、`is_flow_valid` フラグで管理します [73]。なお、週次フロー値が0の場合はNaNではなく0として入力される点に注意が必要です（0は実際に「買い越し額ゼロ」を意味し、NaNは「データ無し」を意味するので区別） [72]。

- **ゼロ割りなど計算上のNaN**: 特徴量計算時に分母がゼロとなる可能性がある比率は、すべて式に `+1e-12` の微小値を加えることでゼロ割りを回避しています [78]。これにより無限大値の発生を防止し、NaNも生じません。また、対数を取る場合もlog(0)となる前に値チェックや+εでの処理をします。計算上のNaNは極力**式設計で出さない**方針です。

- **外れ値の処理**: 極端値については、Winsorize（上下限張り付き処理）は**データ構築段階では実施しません** [79]（将来情報を含むため）。モデル学習前の前処理フェーズで必要に応じて外れ値クリッピングを行います。したがって、生データ上は極端値もそのまま保持され、NaNにはなりません。ただし、クロスセクション正規化時にロバストなクリップ処理で対応する場合があります（後述の正規化方法参照）。

- **意図的なImputeとNaNフラグ**: 上記以外で、モデルが扱えないNaNが残る場合には以下の対策を取ります。

- **Impute（埋め値）**: 特徴量全体のNaN割合がごく小さい場合、中央値や平均、直近値での埋めで対応します。ただし時系列系では直近値によるforward-fillは未来情報を混入させない限り許容されます（例: 決算発表後はその値を次の発表まで保持＝前方複製するのは現実的にも市場参加者が最新情報をそのまま評価に使い続けることに相当）。したがって財務値などは**発表後はその値を維持**し、NaNは主にIPO前やデータなし銘柄に限られるようにします。どうしても穴が空く場合は0埋め等（例: 予想が存在しない銘柄の予想改定率は0とみなす等）をルール決めして行います。

- **NaNフラグ**: 数量を埋めても情報自体は欠測だったことを伝えるため、別途フラグを立てます。例えば `X` という特徴量をimputeしたら、`X_nan_flag = 1 if X was originally NaN else 0` を追加します。ただし既に有効フラグで代用できる場合は新設しません。財務・フローは有効フラグで賄えているため、新たなNaNフラグは主に価格系で特殊ケースがある場合などに検討します。

- **ターゲットの欠損**: 目的変数（後述）は株価終値から機械的に計算できるため、本来NaNは発生しません。ただし上場廃止直前など**将来終値が存在しない**ケースではターゲット計算時にNaNになります。そのため、データセット構築時に**末尾n日分のターゲットを持たない行**（予測に使えない行）はフィルタで除外します。具体的には、20営業日先リターンまで計算するなら各銘柄とも最終20日間はターゲット欠損になるため、これらは学習データから落とします。

以上により、**欠損値は可能な限り意味のある形で扱い**ます。NaNの箇所はモデル入力時には適宜0埋めや統計量埋めしつつ、フラグで「ここは本来データ無し」と知らせ、モデルに誤ったパターン学習をさせない工夫を凝らします。

6. 特徴量の正規化方法

金融データは銘柄間で桁違いの値（株価水準や出来高など）があり、また期間によってもスケールが変化します。適切な**正規化**を行うことで学習を安定させます。

- **クロスセクション正規化（Zスコア標準化）**: モデル学習前のデータ前処理として、**日次の横断で各特徴量を標準化**します [80]。具体的には**毎営業日ごとに**全銘柄の特徴量分布について平均・標準偏差を算出し、各値から当日平均を引いて当日標準偏差で割ります [81]。こうするとその日の銘柄間での相対的なスケールだけが残り、市場全体の水準変動（例えば出来高が年々増加する傾向など）は除去されます。**情報漏洩防止**のため、学習時はトレーニングデータで計算した分布でテストデータも変換します（つまりテスト期間の平均・標準偏差は使わない） [82]。実装的には学習データでNormalizerをfitし、同じ変換を検証データに適用します。この方法は**セクターローテーションなど日々変わる相対関係**を捉える金融モデルに有効ですが、全特徴量に適用すると業種ダミーなども平均0になる点には留意します（カテゴリ変数には適用しないか、別処理します）。

- **時系列正規化（単一銘柄内標準化）**: 特徴量によっては、各銘柄の過去履歴に対して標準化する方が適切な場合があります。例えばボラティリティの**過去1年Zスコア**やフローの52週Zスコアは特徴量として既に計算済みです [83] [48]。これは**統計的特徴量自体をNormalizationする**アプローチで、モデル入力前の処理というよりは特徴量設計の一環です。主要な数値（価格・出来高など）は対数変換や対時系列比率を取ることでスケールを抑えているため、改めてmin-maxなどは適用していません。まとめると、**特徴量計算時**には適宜対数化や比率化、Z変換を用い、**モデル入力時**には日次クロスセクションZ変換を一括適用する二段構えになっています。

- **ターゲットのスケーリング**: 回帰ターゲットであるリターン値はおおむね -1〜+1の範囲に収まり、平均0に近いので特にスケール変換は行いません（そのまま%として扱います）。必要に応じて学習安定のため0.01倍（=1%単位）にスケールすることも考えられますが、実装上は損失関数を調整することで対応可能です。分類ターゲットの場合（上昇=1/下落=0）は0/1のカテゴリなのでスケール不要です。

- **異常値クリッピング**: 正規化ではありませんが、外れ値対策として、クロスセクション標準化時に**ロバストクリップ**を導入します [81]。例えば各日の値を標準偏差5σでクリップする（5を超えるZ値は5に丸める）ことで、極端な値が学習を歪めないようにします。これはNormalizationクラスの一部で実装し、トレーニングデータfit時に適用されます。

- **カテゴリ変数のエンコード**: 市場区分やセクターなどは0/1のダミー変数に変換済みで、0-1値なのでそのまま扱います（平均0になるように-0.5するなども検討できますが、あまり効果的でないためしません）。また一部モデルではembedding層で扱うため整数カテゴリのままにする場合もありますが、その場合はembedding内で正規化相当がなされます。

以上により、**数量的な特徴量は適切にスケーリングされ、モデルに入力**されます。特に日次クロスセクションZスコア化は情報漏洩なく株式の**横の比較**を容易にし、多くの先行研究でも有効とされる手法です [81]。一方で時系列一貫の意味を持つ特徴量（例: 決算フラグなどバイナリ）はそのまま残す形になります。各特徴量について正規化方法を統一しすぎず、**意味に応じて最適な方法を選択**しています。

7. モデル入力形式

本パイプラインで構築する学習データは、多ホライズン予測に対応した**時系列パネル**となります。モデルへの入力形態は以下のように設計します。

- **テンソル次元と形状**: モデルには基本的に**(バッチサイズ N) × (時系列長 L) × (特徴量数 F)**の3次元テンソルでデータを渡します。ここでLは時系列シーケンス長で、例えば**過去60営業日**を入力に使う場合はL=60とします [84]。各シーケンスは銘柄ごとの連続した日次データを時系列順に並べたものです。Nは学習時のミニバッチ内シーケンス数です。例えばTransformer系モデルではこの3次元テンソルを入力とし、系列次元に沿って注意機構が働きます。

- **静的特徴と動的特徴の分離**: 一部の特徴量（セクター分類や銘柄固有不変の指標など）は時間によらず一定で**静的特徴**とみなせます。これらはシーケンス内で全ステップ同じ値になるため、重複計算を避けるなら別途静的入力として渡すことも可能です。しかし実装簡易のため、ここでは**静的特徴も各日付のレコードに含め**、60日のシーケンス内に同じ値が60回並ぶ形でも許容します（モデルが気にしなければ問題ありません）。Advancedなアプローチとして、Graphネットワークで静的特徴をノード属性、動的特徴を時系列属性として扱う場合もあります [85]（ATFT-GAT-FANモデルなどでは静的な銘柄embeddingと動的な時系列特徴を組み合わせます）。基本方針として、シンプルなDeepモデルでは**全特徴を動的特徴量列とみなして3次元テンソルに含める**方針です。

- **多ホライズン出力への対応**: 予測したい将来リターンが1日後・5日後・10日後・20日後の4種類あるため、モデルの出力は**4つのターゲットを同時に推論**できる形にします（マルチタスク学習） [86]。具体的には、モデルの最終出力層を4ユニットにし、それぞれ `target_1d`, `target_5d`, `target_10d`, `target_20d` に対応づけます [87]。損失関数は4つの予測について均等または重要度に応じて重み付けした平均を取ります（例えば全てMSEの平均や、期間が長いものにやや重みを置くなど調整可）。このマルチホライズン形式により、**共通の特徴量から複数期間の予測**を同時に学習させ、短期と中期の予測精度を両立させます。モデル内では短期予測と長期予測の関連を学習できるメリットもあります。一方で、モデルを**シングルホライズン**（単一期間予測）に分けて4モデル構築する方法も検討できますが、共通部分が多いため本設計では**1モデル多出力**を採用します。

- **系列データの生成方法**: データセット構築時、各銘柄ごとに連続するL日間の特徴量系列に対し、その直後の複数ホライズンターゲットを1サンプルとして扱います。例えばA銘柄の1/1〜3/31の60日間を特徴量、4/1の1日後リターン・4/7の5営業日後リターン…・5/1の20営業日後リターンをターゲットとする、のように作ります。これを全銘柄・全可能日についてスライドさせ、多数のサンプルを生成します。ただし**時系列依存**があるので、データ分割時（後述）には銘柄ごとに訓練とテスト期間を分け、未来情報が混入しないようにします。

- **モデル入力例**: もし60日履歴・145特徴量の場合、単一サンプルは60×145の行列で、これに対応して4次元のターゲットベクトルがあります [84]。このサンプルをバッチN個まとめてN×60×145テンソルとし、モデルに流します。

- **メモリ・計算量への配慮**: 上記形式だとデータ量が大きいため、メモリ効率のために `float16/bfloat16` など低精度データ型も検討します [88]。フレームワーク（PyTorchなど）上でシーケンス並列処理できるよう、データローダを工夫しI/Oを高速化します。シーケンス長Lや特徴量数Fはハイパーパラメータ扱いで、リソースに応じて調整可能です（例えばL=60は約3ヶ月の履歴ですが、短くして高速化もできます）。

まとめると、**時系列長Lの日次データを縦に連結し、横にF次元の特徴量を持つ行列**を基本構造とし、これをモデルに渡すことで**時系列文脈を考慮した予測**を可能にします。マルチホライズンでは**出力次元=予測期間数**となる点がシングルホライズンと異なりますが、入力形態自体は同じです。必要に応じて特殊なモデル（Graphネット等）では別途銘柄間の相関行列など追加入力も考えられますが、ここでは標準的な深層時系列モデルを想定しています。

8. ターゲット変数の設計

予測の目的変数（ターゲット）は**将来の株価リターン**です。複数ホライズンを設定し、それぞれに対し回帰または分類のターゲットを作成します [89]。

- **多ホライズンリターン回帰ターゲット**: **1日後, 5日後, 10日後, 20日後**の終値リターンを算出し、それを回帰タスクの教師信号とします。具体的には各銘柄について、
- `target_1d = Close[t+1] / Close[t] - 1`（翌営業日のリターン） [89]
- `target_5d = Close[t+5] / Close[t] - 1`（5営業日後のリターン）
- `target_10d`, `target_20d` も同様に計算します [89]。
  ここで**営業日カウント**で未来の日付を取り、配当等は考慮しないトータルリターンです（必要なら配当込みリターンに変更可能）。この計算は**将来の終値のみ**を使っており、リークはありません [90]。なお戻り率が極端な値になることを防ぐため（株式は通常±数十%以内）、上場廃止等で終値0になるケースは除外し、また計算時にWinsorizeしない方針です [79]。

- **分類ターゲット**: 上記リターンを符号に基づき**方向（上昇/下落）**の2値に変換したものも定義します [91]。

- `target_1d_binary = 1 if target_1d > 0 else 0` [91]
- `target_5d_binary, target_10d_binary` も同様に、リターンがプラスなら1（アウトパフォーム）、マイナスかゼロなら0とします [91]。
  （20日先については必要なら定義可能ですが、長期になるほどクラス不均衡やトレンド要因が強くなるため、場合によっては用いません。）
  バイナリターゲットは方向性予測や勝率評価に用いるほか、**分類モデル**（例えば勝率を直接予測するようなモデル）を別途構築する際に利用できます。

- **ターゲットラベルの整合**: 1d/5d/...は重複期間があります（例: 1日後リターンは5日後リターン計算の一部を含む）。マルチタスク学習ではこれらのターゲット間の関係も学習されます。一方、データセット上は各Date行に target_1d〜target_20d列を持ち、モデルは一度に4つ出力します。**ターゲット計算はデータセット構築時に一括実施**し、Nullがある行（終点付近など）は除外済みです。

- **ターゲットのラグ**: 特徴量との対応関係として、例えば**2021-01-01のデータ行**には、target_1d = 2021-01-04のリターン（休日挟み3日後実質）、target_5d = 約1週間後...といった値が入ります。学習時にはモデル入力系列の最後の日時をtとすると、それに対応する未来のt+1, t+5,...の値をターゲットとして与える形になります。これを**スライドウインドウ**で全期間カバーします。

- **特殊ターゲット派生**: 場合によっては回帰ターゲットから派生してランキングタスクやペアトレードタスクを組むことも可能ですが、本設計書では主に**回帰値予測**として定義します。ターゲットは**スケーリングせずそのまま%**で扱います（前述の通り）。

- **検証用サブ指標**:　ターゲット自体はリターンですが、モデルの出力を評価する際にはRankICや勝率などを用います（次項）。例えば**ランキング上位銘柄の平均リターン**なども評価しますが、それはターゲット設計ではなく評価設計に属します。

以上により、データセットの各行には4つ（回帰の場合）またはそれ+3つ（バイナリ）のターゲット列が含まれます [92]。この**複数ターゲット設定**により、ユーザは1つのモデルで多様な期間の予測を得られ、短期的なトレード判断から中期的なポートフォリオ構築まで一貫した出力を利用できます。

9. モデルの評価指標

本パイプラインでは、金融タスクに特有の評価指標を用いてモデル性能を多角的に評価します [93]。主な指標は以下の通りです。

- **Rank Information Coefficient (Rank IC)** – **順位相関係数**による評価です。毎営業日についてモデルの予測スコアと実際の翌日（または任意ホライズン）のリターンとの**スピアマン順位相関**を計算し、その平均を指標とします [93]。RankICは**銘柄ランキングの精度**を測る尺度で、+1に近いほど「予測上位の銘柄が実際にも高リターンであった」ことを意味します [94]。外れ値の影響を軽減し相対的な予測力を見るため、金融業界ではIC・RankICがモデル比較に用いられます。計算式:　$$RankIC　= Corr_{Spearman}(predictions, actual\ returns)$$ 日次で算出し、全期間平均します。モデル出力が確率やスコアの場合も同様に順位付けで評価可能です。

- **情報係数 (IC)** – ピアソン相関による情報係数も補助的に算出します [93]。RankICとの違いは**値そのものの相関**を見る点で、外れ値に影響を受けやすいですが予測の線形性を評価できます [94]。ICは通常RankICと傾向が似るため、本設計では主要指標はRankICとし、併記する形にします。

- **勝率（Win Rate）** – モデルがどれだけ高い確率で正しい方向を当てたかを示す指標です。計算としては、**方向予測の的中率**(Accuracy)に相当します。例えばbinaryターゲットを予測する場合は単純に正解率となりますし、連続予測の場合でも「予測が正の銘柄で実際リターンが正だった割合」などと定義できます。一般には**日次でポートフォリオを組んだ際の勝率**、すなわちRankIC>0の日数割合や、上位◯%銘柄の平均リターンがプラスだった割合を指すこともあります。ここでは「**RankICの正符号割合**」および「**トップ予測銘柄の実現勝率**」の両面で評価します。前者は例えば全日数中RankICが正だった日が60%なら勝率60%と表現し、モデルが一貫して有効かを見る指標です。後者は例えば毎日トップ10銘柄に投資したと仮定して、そのうち何銘柄が実際プラスリターンだったかの平均を求めます（これも1日ごと出して平均可能）。これら勝率指標は**直感的なモデルの当たり具合**を示します。

- **Sharpe比 (Sharpe Ratio)** – モデル予測に基づく戦略のリスク調整リターン指標です。具体的には、モデル予測に従い構築したロングショートポートフォリオの**平均超過リターンをその標準偏差で割った値**です。例えば毎日トップ銘柄をロング・ワースト銘柄をショートして等金額持つ戦略を考え、その日次リターン系列からSharpeを算出します。Sharpeは**戦略の一貫した優位性**を示し、0以上であれば有用、1を超えれば非常に優秀とされます。モデル単体評価では、**予測スコアを利用した仮想的なトレード戦略**のSharpeとして算出します（年率換算し、無リスク金利は0想定） [95]。

- **Decile（十分位）分析** – モデル予測に基づき銘柄をランキングし、上位10%、次点…下位10%の**10グループに分けた際のリターン**を分析します [93]。理想的には上位グループほど平均リターンが高く、下位は低い（負の場合も）ことが期待されます。このモノトニックな関係の度合い（例えば上位と下位のリターン差や、グループ順序の一貫性）が**分位安定性**の指標となります。具体的な数値例として

は、「トップ分位ポートフォリオの平均日次リターン」「ボトム分位ポートフォリオの平均日次リターン」「この差（スプレッド）の情報比」などを算出します。また**分位ごとのヒット率**（例えばトップ分位の中で正の銘柄割合）なども見ます。**分位安定性**とは、モデルのスコアランクによって性能が安定的に層別できているかを意味し、例えばトップ分位が常にプラス利益でボトムが常にマイナスなら極めて安定と言えます。これは長期的なモデル有効性を示す重要な観点です。

- **その他指標**:　上記以外にも、期間全体の累積リターン曲線やその最大ドローダウン、情報比（ICの平均/標準偏差）[96]、あるいは期間ごとのICトレンドなども分析します。基本的には **日次のクロスセクション評価**を重視しつつ、実際運用に耐えうるかを総合判断します[97]。

指標算出にはFinancialMetricsユーティリティを用い、日次IC計算や分位分析を自動化します[98]。例えばコード上では:

```
metrics = FinancialMetrics()
rank_ic = metrics.compute_rank_ic(preds, actuals, dates)  # 平均RankICと日次推移 [99] [100]
```

のようにして計算します。以上の各種指標により、**モデルの予測力を多面的に評価**し、単なる誤差(MSE)だけでなく実運用上有用かどうかを判断します。

10. 実装タスク分解（関数単位）

最後に、本パイプラインを実装する際の主要タスクを工程順に分解し、それぞれを関数やモジュール単位で整理します。以下に挙げる関数は擬似コード的な役割説明で、実際にはクラスメソッドとして実装する部分もありますが、ここでは責務に応じて列挙します。

1. **データ取得 (`fetch_data`)** – J-Quantsの各APIから必要な期間のデータを取得します。具体的には:
2. `fetch_trading_calendar`: 指定期間の営業日カレンダー取得（API: `/markets/trading_calendar`）。これで全ての処理日付範囲を決定します[101]。
3. `fetch_listed_info`: 指定期間の銘柄マスター取得（API: `/listed/info`）。初回全銘柄一覧と、差分検知による定期更新（毎月初や必要日だけ）でIPO/上場廃止情報を収集します[102]。取得したデータはメモリとローカルキャッシュに保存し、繰り返し呼び出しを効率化します。
4. `fetch_prices`: 日次株価OHLCV取得（API: `/prices/daily_quotes`）。全銘柄・全営業日の株価を網羅します。市場コードフィルタを活用し、対象市場（プライム等8市場）に絞った銘柄だけ取得します[103][69]。大量データのため、有料プランの場合は**並列75接続**でバッチ取得し、1時間程度で数年分取得完了します[104][105]。実装上は銘柄軸・日付軸を自動選択してAPIコール最適化も行います[106]。
5. `fetch_index`: 市場指数データ取得（API: `/indices/topix` 等）。特にTOPIXは長期履歴を含めて取得し、別途データフレーム化します。低頻度（例えば日経VIなど必要なら週次）データもここで取得します。
6. `fetch_fundamentals`: 財務諸表データ取得（API: `/fins/statements`）。営業日ループでその日に開示のあったものだけを取得する手法を採用し、空振りを減らします[107]。バックフィル時はコード軸でも不足銘柄を補います[107]。取得したRaw財務データはパース・型変換（数値項目をfloat）してデータフレーム化します[108]。
7. `fetch_trades_spec`: 投資部門別売買を取得（API: `/markets/trades_spec`）。セグメント別に期間一括取得し、過去データもまとめて入手します[109]。過誤訂正に対応するため、published_date 違いの重複も併存させます[109]。運用時は直近数週を毎日取り直し、データ修正を反映します[109]。

8. 各取得関数では**エラーハンドリング**（リトライ3回・指数バックオフ[110]）やAPIレートリミット管理（同時接続数・遅延）を組み込みます。認証情報は.envから読み込み[111]、キャッシュ（メモリ・ローカル）を用いて重複取得を避けます[110]。

9. **データ結合 (`join_data`)** – 取得した各データソースを一つのマスターデータテーブルに結合します。大まかな手順:

10. **ベーステーブル構築**: 日次株価データを基に、まず`base_df(date, code, Open, High, Low, Close, Volume, ...調整額)`を構築します。ここで銘柄マスター情報を用いて対象外市場や期間外銘柄を除去します（先に**営業日×銘柄のフルグリッド**を組む方法もありますが、株価が存在する組のみで十分です）。株価データ自体に欠損がないか確認し、なければこれがマスターフレームの土台になります [76] [75] 。
11. **市場区分・セクター付与**: base_dfに銘柄マスターから市場区分・業種セクター等をCodeキーで結合し、新しい列`market_code_name`や`sector_code`等を追加します。この情報は後のフロー結合やセクター特徴量計算に用います。
12. **市場指数結合**: TOPIXなど指数データをDateキーでleft joinし、base_dfの全銘柄各日に市場指標列（Index Closeやその派生特徴量後で計算）を付与します [112] 。結合後、INDEX由来の列は全銘柄同値になりますが、この後の処理で銘柄毎の計算に使われます。
13. **財務データ結合**: fundamentalsデータフレームをCodeキー×Dateで**後方結合**します（Polarsなら.groupby("Code").apply(... asof join ...)等を実装） [52] 。結合ルールは前述したT+1ルールを適用し、各Dateに各Codeの最新開示情報のフィールドを追加します [113] 。例えばProfit, NetSales等が新規列として加わります。結合後すぐには派生指標は計算せず、とりあえず**原始財務項目**を持たせた状態にします。
14. **フローデータ結合**: まずフローのrawデータから**日次展開テーブル**を作ります。これは(Date, section_norm)ごとに各フロー指標（ForeignersTotal等）を持つテーブルです。手順は前述の通りで、各週次データ行をそのeffective期間中のDate行に展開します [8] 。続いてbase_dfに対し、銘柄ごとに対応するsection_normをキーにこの日次フローテーブルを結合します [43] 。left joinすることで、一部NaN（対象外市場）は残りますが大半の銘柄日にフロー値が埋まります。列名は`flow_`プレフィックスにリネームしておきます [43] 。
15. **不要銘柄・日付フィルタ**: 結合後、上場廃止銘柄の廃止翌日以降の行やIPO前の行などが万一存在すれば除去します（もっとも最初から銘柄マスターで除外していれば無いはずですが、二重チェックします）。また全列NaNの行がないか確認し、あれば除去します。

16. このjoin_data工程の出力は**統合データフレーム**（主キー: date+code, 列: 原始特徴量の大半）です。まだ計算していない派生特徴もありますが、ひとまず全ソースの値が揃ったテーブルになります。

17. **特徴量計算 (`build_features` / `build_dataset`)** – 結合済みデータから各種**二次特徴量を計算**し、最終的な学習用データセットを構築します。

18. **価格系特徴量計算**: base_df上でPolars/Pandasのウィンドウ関数等を使いリターン・ボラ・移動平均等を計算します [114] [115] 。例えばPolarsの場合:

    ```
    df = df.with_columns([
        (pl.col("Close") / pl.col("Close").shift(1) - 1).alias("returns_1d"),
        (pl.col("Close") / pl.col("Close").shift(5) - 1).alias("returns_5d"),
        # ... 他リターン
        (pl.col("returns_1d").rolling_std(20) * np.sqrt(252)).alias("volatility_20d"),
    ])【8†L621-L629】
    ```

    のようにまとめて計算できます。MAやテクニカル指標も同様に計算しdfに列を追加します。pandas-taライブラリで計算する部分は、一度時系列順に並べた各銘柄DataFrameに適用してmergeする流れになります。
19. **市場・クロス特徴量計算**: すでに結合済みの指数データから市場特徴を計算します。例えばTOPIXのCloseは全銘柄同一なので、一銘柄だけ抽出して計算するか、Polarsなら`.over("Date")`ウィンドウ

で一括計算できます。先に市場側特徴（mkt_*）を別テーブルで作り、Dateで結合してもよいです [116]。クロス特徴は銘柄×日次で計算するため、Polarsなら `.over("Code")` で銘柄別グループ化してCovやVar関数で計算します [117]。例えばβ計算:

```
df = df.with_columns([
    pl.cov(pl.col("returns_1d"),
pl.col("mkt_ret_1d")).over("Code").alias("cov_60").rolling_mean(60),
    pl.var(pl.col("mkt_ret_1d")).rolling_mean(60).alias("var_60"),
]).with_columns([
    (pl.col("cov_60") / (pl.col("var_60") + eps)).alias("beta_60d")
])【8†L636-L642】
```

のように実装できます。他のcross特徴量（alphaなど）は計算式通り追加します。

20. **フロー特徴量計算**: 結合時点では週次raw項目（ForeignersBalance等）が日付に複製されて持たれているので、そこから二次計算します。例えば:

```
df = df.with_columns([
    (pl.col("ForeignersBalance") / (pl.col("ForeignersTotal") +
eps)).alias("flow_foreign_net_ratio"),
    # ... 他の比率
])
```

Zスコアは過去52週=260営業日程度のrolling_zscore関数を用意して計算します。スマートマネー指数等も算出して新規列とします。最後にrawのフロー項目（巨大な金額そのもの）は不要ならドロップし、flow_プレフィックスの派生特徴だけ残します。

21. **財務特徴量計算**: 結合時に原始財務列（Profit等）が増えたので、これを使い前述の17項目を算出します [11] [12]。計算には同一Code内でのラグ（-4期など）を取る必要があるため、まず**各銘柄ごとに決算期順**に並べ替えた上で計算します（決算期＝四半期ごとのキーが必要ですが、J-Quantsのデータには期ズレを判別するIDがあればそれでグループ化可能。無ければdisclosed_date間隔で近似可）。polarsで同Code内 `cumcount` を使い、ラグ4を安全に取ります。その上でYoYや増減率を計算し、DataFrameに列追加します [11] [12]。ついでにROE/ROA等も計算します。変更会計フラグは既にtrue/falseで入っているのでint変換します [57]。
22. **成熟フラグ計算**: 価格系の is_valid_* フラグはrow_idxベースで算出できます（row_idxはPolarsで各銘柄のcumcountまたは既存実装を利用） [118]。flowとstmtの有効フラグも、各日付でその値がNaNでないかで作ります [73]。例えば `is_flow_valid = flow_days_since.is_not_null()`、`is_stmt_valid = stmt_days_since_statement.is_not_null()` [73]。
23. **ターゲット計算**: 特徴量計算が完了したら、最後にターゲット列を計算しデータセットに加えます [119]。これも各銘柄について終値のシフト演算で求めます。Polarsなら:

```
df = df.with_columns([
    (pl.col("Close").shift(-1) / pl.col("Close") - 1).alias("target_1d"),
    (pl.col("Close").shift(-5) / pl.col("Close") - 1).alias("target_5d"),
    # ...10d, 20d
]).with_columns([
    (pl.col("target_1d"] > 0).int8().alias("target_1d_binary"),
    # ... 5d_binary, 10d_binary
])【8†L578-L587】
```

のようにします。将来がない部分（shiftでNaNになる行）はターゲット欠損となるので削除します。結果、全ターゲットが有効なレコードのみが残ります [120]。

24. **スキーマの最終整理**: 全列が出揃ったら、不要中間列（例: raw財務項目やrawフロー項目）はドロップします。欠損が多すぎる列も分析し、必要なら除外します。最終的な列数は約160弱の想定です [121]。Polars上で型も適切か確認し（日付はpl.Date、カテゴリはUtf8等） [30] [67]。

以上で**完成した特徴量データセット**（全銘柄全期間パネル）が出来上がります。これをParquet等にエクスポートし [122]、以降の工程で再利用します。

1. **データ品質検証 (`validate_dataset`)** – 構築したデータセットに対し、スキーマや統計量の検証を行います。
2. **スキーマ・ID検証**: date列とcode列が存在し、それに欠損が無いこと、(date, code)が一意であることをチェックします [76] [75]。自動スクリプト `dataset_sanity.py` でもこれらを確認できます [123]。
3. **ターゲット検証**: 期待するターゲット列（target_1d 等）が存在し、欠損率0%であることを確認します [123] [124]。また、ターゲット分布（平均0に近いか、極端値はないか）をざっと確認します。binaryターゲットについては0/1以外の値が無いかチェックします。
4. **特徴量分布検証**: いくつか代表的な特徴量についてヒストグラムや基本統計量を出し、明らかにおかしな値（例えば比率が±10000%など）が無いか確認します。特に計算式ミスで発生する定数値や異常値を洗います。開発段階では `docs/reports/DATASET_DIFF_REPORT.md` 等で差分検証も行っているので、本番時も同様のレポートを更新します。
5. **情報漏洩検証**: 時系列順にソートし、各特徴量が未来の値を参照していないか論理チェックします。例えばtarget計算にClose.shift(-1)を使ったので、特徴量側でshift(-1)が混入していないか（Polars式でshift(1)は過去参照なのでOK、shift(-1)があれば問題）をコードレビューします。財務・フローのeffective_dateロジックもテストシナリオで確認します。統合テスト `test_jquants_integration.py` を走らせ、エラーが出ないことも確認します [125]。
6. **欠損最終確認**: 有効フラグで想定通りNaNがマスクできるか、小数点桁数や型は適切かを確認します（例えばboolフラグがint8になっているかなど [73]）。また不要なNaNが残っていればここでimpute処理を追加実装します。

検証過程で問題があれば前工程に戻って修正し、再度fetchから繰り返します。最終的に `[dataset-sanity] OK` が出力されれば完成です [74] [126]。

1. **データ分割 (`split_data`)** – 機械学習モデル訓練用に、データセットを**訓練・検証・テスト**に分割します。株価予測では時系列順に過去→未来で分け、未来データが訓練に混ざらないようにします。
2. **時系列スプリット**: 例えば2015-2019年を訓練、2020年をバリデーション、2021年をテストといった**期間ベースの分割**を行います（または rolling originによるwalk-forward CV [127]）。単一モデルの評価ではホールドアウト期間を設定し、モデルは過去データのみで学習させます。複数モデルやチューニングでは**ウォークフォワード検証**で複数折に分けます [127]。いずれもsplit_data関数でインデックスを作成します。
3. **ウォークフォワードCV**: 実装例として、5-foldの時系列CVを行う `WalkForwardSplitter` を用意し、各foldごとにtrainとtestのインデックスリストを得ます [128] [129]。また**embargo期間**として最大ホライズン日数（20日）をテスト開始前に空けます [130]。これにより、例えばfold1のtrainは〜2019年末、testは2020年以降（ただし2020年最初の20日は除く）という風に、訓練データ中の末尾付近の情報がテストターゲット期間にかぶらないようにしています。
   分割後、各foldのサイズを出力しバランスを確認します（例: Fold0: 100k行 train, 20k行 test 等）。
4. **銘柄の扱い**: 原則全銘柄データを時系列でまとめて分割します（ランダムで混ぜない）。よって訓練期間には全銘柄の過去データ、テスト期間には全銘柄の未来データが入ります。ただしIPOなどで訓練期間になくテスト期間のみ登場の銘柄は**コールドスタート問題**として除外するか、特徴量欠損が多いため無視します。分割時に「trainに存在しないCodeはtestから除外」するオプションもあり得ます。

5. **標準化との関係**: クロスセクション正規化はトレーニングデータでfitした統計量を使うため、データ分割後に訓練セット全体でNormalizerを計算します [131]。その際、各foldのtrainのみ使い、その統計で

val/testを変換します。実装ではsplit_data後に正規化クラスに分割indexを渡し、内部でfit_transform/transformを行います。

6. **モデル訓練 (`train_model`)** – 準備したデータで機械学習モデルの学習を行います。

7. **モデル選択**: まずBaselineとして勾配ブースティング（LightGBM）や線形モデルで性能確認し[132]、その後本命のディープラーニングモデル（例: ATFT-GAT-FAN）を訓練します[133]。モデルインスタンスは`prediction_horizons=[1,5,10,20]`などとして多ホライズン出力を設定します[134][135]。
8. **ハイパーパラメータ設定**: 学習率・バッチサイズ・エポック数などを設定し、過学習を避けるため早期停止や正則化（ドロップアウト、重み減衰）も有効にします。シーケンス長Lや隠れ次元も調整ポイントです。FAN正規化やGAT隠れ層サイズなど高度なものはデフォルト初期にします[135]。
9. **訓練ループ**: PyTorch Lightning風のTrainerを用意し、各エポックでtrainセットでの損失を計算、バリデーションセットでの損失やICを計算します。損失は4ターゲットそれぞれのMSEの和/平均を基本とし、学習を安定させるため**多ホライズンMSE**を最小化します。場合によっては短期ターゲットに重みを高くする（短期精度重視）調整も検討します。最良バリデーション指標でモデルチェックポイントを保存し、早期停止も適用します。
10. **並列処理とGPU**: データ量が多いためGPU学習を行います。mixed precisionでメモリ削減し[136]、GPU複数枚があればDDP並列も可能です。今回はシーケンス60×145で約5万系列程度なら単GPUで十分訓練可能ですが、より長期・高頻度データでは工夫します。
11. **モデル評価**: 訓練完了後、テストセットに対して予測を生成し、前述の**評価指標**(RankIC, Sharpe等)を算出します[137]。Baselineや過去モデルと比較し、目標性能（例: Sharpe 0.8以上[138]など）に達しているか検証します。各ホライズン毎のICも出力し[137]、短期〜長期での性能プロファイルを確認します。

以上が主要な実装タスクの流れです。各タスクはそれぞれモジュール化され、スクリプトとしては: - データ取得と前処理全体を走らせる `run_pipeline.py --jquants`[139] が **fetch_data→join_data→build_features→validate_dataset** を実行し、最終データセットを出力します。 - 学習は `integrated_ml_training_pipeline.py`[140] やノートブック上で **split_data→train_model** を実行します。途中で `validate_improvements.py`[141] によるデータ品質チェックや、`hyperparameter_tuning.py`[140] でのベイズ最適化も可能です。 - こうした関数単位の設計により、例えばデータ更新時にはfetch/join部分のみ再実行、モデル構造変更時にはtrainのみ実行、など柔軟にパイプラインを運用できます。

最後に、以上の設計と実装により、J-Quants提供データを余すところなく活用した多ホライズン株価リターン予測モデルのパイプラインが完成します。各部の設計はコードにそのまま落とし込める詳細さで記述しましたので、実装時の指針として参照してください。[142][143]

---

[1][2][3][6][7][8][9][10][11][12][13][14][15][16][17][18][19][20][21][22][23][24][25][26][27][28][29][30][31][32][33][34][35][36][37][38][39][40][41][42][43][44][45][46][47][48][49][50][51][52][53][54][55][56][57][58][59][62][63][64][67][70][71][72][73][78][83][89][90][102][106][107][108][109][112][113][114][115][116][117][119][120][121] DATASET.md
https://github.com/wer-inc/gogooku3/blob/5aa73d16b460d89de08ea013934b1f78210d1dde/docs/DATASET.md

[4][5][61][77] MARKET_FEATURES_IMPLEMENTATION.md
https://github.com/wer-inc/gogooku3/blob/5aa73d16b460d89de08ea013934b1f78210d1dde/docs/ml/MARKET_FEATURES_IMPLEMENTATION.md

[60] peer_features.py
https://github.com/wer-inc/gogooku3/blob/5aa73d16b460d89de08ea013934b1f78210d1dde/src/features/peer_features.py

[65][74][75][76][123][124][126] dataset_sanity.py
https://github.com/wer-inc/gogooku3/blob/5aa73d16b460d89de08ea013934b1f78210d1dde/scripts/ci/dataset_sanity.py

66 68 69 101 103 104 105 110 111 122 125 139 140 141 JQUANTS_FULL_SCALE_IMPLEMENTATION.md

https://github.com/wer-inc/gogooku3/blob/5aa73d16b460d89de08ea013934b1f78210d1dde/docs/architecture/JQUANTS_FULL_SCALE_IMPLEMENTATION.md

79 91 92 118 142 143 ML_DATASET_COLUMNS.md

https://github.com/wer-inc/gogooku3/blob/5aa73d16b460d89de08ea013934b1f78210d1dde/docs/_archive/specifications_original/ML_DATASET_COLUMNS.md

80 81 82 84 85 86 87 88 127 128 129 130 131 132 133 134 135 136 137 138 model-training.md

https://github.com/wer-inc/gogooku3/blob/5aa73d16b460d89de08ea013934b1f78210d1dde/docs/ml/model-training.md

93 94 95 96 97 98 99 100 financial_metrics.py

https://github.com/wer-inc/gogooku3/blob/5aa73d16b460d89de08ea013934b1f78210d1dde/src/metrics/financial_metrics.py