C3. Features.Core — постановка задачи (детализация)

Версия: 1.0
Дата: 2025-10-15
Ответственность: расчёт базовых признаков по OHLCV без обращения к ленте/стакану.

---

1. Цель и результат

Сервис генерирует набор детерминированных признаков (features) из валидированных OHLCV-баров для последующего использования компонентами сигналов (C6/C7), режимов (C4) и риска (C9). Результат — датафрейм на том же таймфрейме с добавленными колонками признаков и метаданными.

---

2. Входные данные

Формат: CSV (UTF-8, без BOM), разделитель `,`. Одна запись = один бар. Одна серия = один тикер и один таймфрейм.
Колонки (обязательно): - `timestamp_ms` (int64) — метка времени UTC в миллисекундах, строго возрастающая. - `start_time_iso` (string, ISO-8601) — дублирующий ISO-штамп начала бара (например, `2025-10-14T12:35:00Z`). - `open`, `high`, `low`, `close` (float64) — цены. - `volume` (float64) — объём в базовой валюте контракта. - `turnover` (float64) — оборот в квотируемой валюте (USDT).

Предусловия: - Данные уже прошли DataQuality (см. C2): нет NaN/Inf, `high ≥ max(open, close)`, `low ≤ min(open, close)`, `volume ≥ 0`, без дубликатов таймстампов, сортировка по возрастанию `timestamp_ms`. - Таймзона: UTC. Гранулярность бара соответствует указанному TF (1m/5m/15m/1h и т.д.).

---

3. Выходные данные

Формат: Parquet по умолчанию; опционально CSV.
Схема: все входные колонки + признаки, префикс `f_`.

Обязательные служебные колонки: - `symbol` (string) — идентификатор инструмента (подаётся через конфиг/CLI).
- `tf` (string) — таймфрейм входной серии (например, `5m`).
- `f_valid_from` (int64, индекс бара) — первый индекс, с которого все заявленные признаки не содержат NaN.
- `f_build_version` (string) — версия реализации и набор параметров (хэш).

---

4. Зависимости и среда

- **Язык/версия:** Python ≥ 3.11.
- **Библиотеки (обязательно):** `pandas>=2.2`, `numpy>=1.26`.
- **Библиотеки (опционально для ускорения):** `numba>=0.59` (JIT для скользящих расчётов), `pyarrow` (Parquet I/O).
- Запрещено использовать внешние закрытые/нестабильные TA-библиотеки; все формулы реализуются явно либо на базе pandas/numpy.
- Все операции — детерминированные при фиксированном входе.

---

5. Набор признаков и формулы

Ниже — полный перечень признаков, их обозначения, окна и формулы. Везде `t` — текущий бар, `n` — окно, `C_t` — `close[t]`, `H_t` — `high[t]`, `L_t` — `low[t]`, `O_t` — `open[t]`, `V_t` — `volume[t]`.

5.1 Базовые доходности и волатильность

1. `f_ret1` = `(C_t / C_{t-1}) - 1`
2. `f_logret1` = `ln(C_t / C_{t-1})`
3. `f_rv_n` — реализованная волатильность за окно `n`: `std(logret, n)` (несмещённая, `ddof=1`). По умолчанию `n ∈ {20, 60}`.
4. `f_range_pct` = `(H_t - L_t) / C_t`
5. `f_body_pct` = `abs(C_t - O_t) / C_t`
6. `f_wick_upper_pct` = `(H_t - max(C_t, O_t)) / C_t`
7. `f_wick_lower_pct` = `(min(C_t, O_t) - L_t) / C_t`

5.2 True Range / ATR и производные

1. `f_tr` (True Range): `max(H_t - L_t, |H_t - C_{t-1}|, |L_t - C_{t-1}|)`
2. `f_atr_n` — ATR по Уайлдеру: `EMA_Wilder(TR, n)` с `α = 1/n`. По умолчанию `n=14`.
3. `f_atr_pct_n` = `f_atr_n / C_t`
4. `f_atr_z_n` = `zscore(TR, n)`

5.3 Тренд/моментум

1. `f_ema_n` — эксп. скользящая EMA по цене закрытия, `n ∈ {20, 50}`.
2. `f_ema_slope_n` = `(EMA_t - EMA_{t-1}) / EMA_{t-1}`
3. `f_mom_n` (Price Momentum) = `C_t / C_{t-n} - 1`, `n ∈ {20, 50, 100}`.
4. `f_mom_rsi14` — RSI(14) по классике Уайлдера (для совместимости с C4).
5. `f_adx14`, `f_pdi14`, `f_mdi14` — +DI/−DI и ADX(14) на базе Wilder-EMA.

5.4 Donchian и волатильностные прокси

1. `f_donch_h_n` = `rolling_max(H, n)`; `f_donch_l_n` = `rolling_min(L, n)`, `n ∈ {20, 55}`.
2. `f_donch_break_dir_n` ∈ {−1,0,1}: пробой вверх/вниз/нет относительно `f_donch_h_n`, `f_donch_l_n`.
3. `f_donch_width_pct_n` = `(f_donch_h_n − f_donch_l_n) / C_t`.

5.5 Z-score, нормализации

1. `f_close_z_n` = `(C_t − mean(C, n)) / std(C, n)`; `n ∈ {20, 60}`.
2. `f_range_z_n` = `zscore(H−L, n)`.
3. `f_vol_z_n` = `zscore(V, n)`.

5.6 Объёмы и баланс

1. `f_upvol_n` = `Σ_{i=t−n+1..t} V_i · 1[C_i > C_{i−1}]`
2. `f_downvol_n` = `Σ V_i · 1[C_i < C_{i−1}]`
3. `f_vol_balance_n` = `(f_upvol_n − f_downvol_n) / (f_upvol_n + f_downvol_n + ε)`
4. `f_obv` (On-Balance Volume) — кумулятивно: `OBV_t = OBV_{t−1} + sign(C_t−C_{t−1})·V_t`.

5.7 VWAP-proxy (без тиков)

1. `f_vwap_roll_n` — роллинг-VWAP за `n` баров: `Σ(typical_price·V)/ΣV`, где `typical_price = (H+L+C)/3`.
2. `f_vwap_dev_pct_n` = `(C_t − f_vwap_roll_n) / f_vwap_roll_n`
3. `f_vwap_session` — сессионный VWAP, реинициализация при переходе UTC-дня.
4. `f_vwap_session_dev_pct` = `(C_t − f_vwap_session) / f_vwap_session`.

5.8 Прочие служебные

1. `f_liq_proxy` = `turnover` (или `C_t·V_t` при отсутствии),
2. `f_spread_proxy` = `f_donch_width_pct_20` (как суррогат ширины внутридневного диапазона),
3. `f_kama_n` (опционально) — Kaufman Adaptive Moving Average, `n=10`.

Примечания по окнам по умолчанию: если TF ∈ {5m, 15m}, использовать набор окон из перечислений выше. Для 1h можно сократить `mom_n` до {20, 50}. Все окна настраиваются через конфиг.

---

6. Обработка окон, NaN и «прогрев»

- Для всех скользящих расчётов использовать `min_periods = n` (строгое) и не заполнять NaN значениями по умолчанию.
- Значение `f_valid_from` устанавливается как `max(n)` среди всех активных окон (или первые индексы без NaN для каждого признака — берётся максимум).
- Не допускается forward-fill/back-fill цен и объёмов внутри компонента.

---

7. Параметры (конфиг)

Формат YAML/TOML. Ключи по умолчанию:

```
symbol: "BTCUSDT"
tf: "5m"
```

```yaml
windows:
  rv: [20, 60]
  ema: [20, 50]
  mom: [20, 50, 100]
  rsi: 14
  adx: 14
  donch: [20, 55]
  z: [20, 60]
  vwap_roll: 96   # пример: ~8 часов для 5m
  updownvol: 20
atr:
  n: 14
  mode: "wilder"   # варианты: wilder | ema
outputs:
  format: parquet  # варианты: parquet | csv
  path: "/data/features/{symbol}_{tf}.parquet"
perf:
  use_numba: true
  parallel_read: false
```

---

8. Интерфейсы

8.1 Python API

```
compute_features(df: pandas.DataFrame, cfg: dict) -> pandas.DataFrame
```

Вход: DataFrame с колонками из §2.
Выход: DataFrame с добавленными `f_*` и служебными колонками (§3).

8.2 CLI

```
features_core build
  --input /path/in.csv
  --symbol BTCUSDT
  --tf 5m
  --config features.yaml
  --output /data/features/BTCUSDT_5m.parquet
```

Код возврата `0` при успехе; `≠0` при ошибке с сообщением в stderr (JSON-структура).

---

9. Алгоритмические детали

- EMA/RSI/ADX реализовать по формулам Уайлдера. Проверить соответствие эталонам на тестовых сериях.

- Donchian реализовать через векторные `rolling().max()/min()`.
- VWAP-session: идентификация смены дня по `start_time_iso` (UTC). Использовать кумулятивные суммы цены*объёма и объёма, сбрасывая при смене даты.
- Z-score: стандартное отклонение с `ddof=1`.
- Все деления с малым знаменателем — добавлять `ε=1e-12`.

---

10. Валидация и тесты

Юнит-тесты: - Константный тренд: `C_t = t` — проверка знаков `f_mom_n`, ненулевого `f_ema_slope_n` и адекватности `f_tr`. - Пилообразная серия — проверка Donchian, `f_donch_break_dir_n`. - Константная цена — `f_tr=0`, `f_atr_n→0`, `f_rv_n→0`. - Обнулённый объём — корректность VWAP и vol-признаков (обработка ε).

Интеграционные тесты: - Сравнение с эталонными расчётами (CSV-фикстуры) для RSI/ADX/ATR/DONCH.
- Сходимость VWAP-session при склейке нескольких дней.
- Проверка `f_valid_from` и отсутствия NaN после него.

Критерии приемки:
- Полное совпадение со снапшотами эталонных фикстур (толеранс `1e-9`).
- Время обработки: $\leq$ 2 сек на 1 млн баров при `use_numba=false` (на эталонном стенде), $\leq$ 1 сек при `true`.

---

11. Производительность и ресурсы

- Вычисления — векторные, без Python-циклов; допускается `numba` для критичных участков (TR/EMA/RSI/ADX).
- Потребление памяти $\leq$ 1.5× объёма входных колонок + временные буферы окон.
- Потокобезопасность: без глобального состояния; допускается параллельная обработка разных символов/ТФ процессами.

---

12. Логирование и трассировка

- Уровни: INFO — старт/окна/валид-from; DEBUG — контрольные статистики по признакам; ERROR — нарушение предусловий (§2).
- Все ошибки — с указанием индекса бара и имени признака.

---

13. Ошибки и возвраты

- `E_SCHEMA` — отсутствует обязательная колонка.
- `E_SORT` — нарушена сортировка `timestamp_ms` (невозрастающая).
- `E_DUP_TS` — дубликаты таймстампов.
- `E_NAN` — найдены NaN/Inf.
- `E_PARAM` — некорректные окна/параметры.

- `E_RUNTIME` — авария вычислений (деление на 0, переполнение и т.п.).

---

14. Версионирование и воспроизводимость

- Семвер релизов компонента; изменение набора/семантики признаков — мажорная версия.
- В `f_build_version` сохранять `git-sha` и JSON-хэш активных параметров.

---

15. Безопасность и ограничения

- Отсутствие сетевых вызовов внутри компонента.
- Чтение только указанного входного файла, запись — только в выходной путь из конфига.

---

16. Пример выходной схемы (фрагмент)

```
index, timestamp_ms, start_time_iso, open, high, low, close, volume,
turnover,
 symbol, tf, f_ret1, f_logret1, f_tr, f_atr_14, f_atr_pct_14,
 f_ema_20, f_ema_slope_20, f_mom_20, f_rsi14, f_pdi14, f_mdi14, f_adx14,
 f_donch_h_20, f_donch_l_20, f_donch_break_dir_20, f_donch_width_pct_20,
 f_close_z_60, f_vol_z_20, f_upvol_20, f_downvol_20, f_vol_balance_20,
 f_vwap_roll_96, f_vwap_dev_pct_96, f_vwap_session, f_vwap_session_dev_pct,
 f_valid_from, f_build_version
```