因子1：柴油波动率调整动量（diesel_vol_adj_mom_20d_v1）

来源：单表USA Daily Diesel Spot Price.csv

公式：过去20日的柴油价格平均收益率 / 过去20日的收益率标准差

逻辑: 基于美国柴油现货价格的波动率调整动量因子,用于预测原油价格

1. 裂解价差联动: 柴油是原油的下游产品,柴油价格的变化往往领先或同步反映原油市场的供需变化。柴油的强劲动量可能预示原油需求上升。
2. 风险调整的动量: 单纯的柴油价格上涨可能由短期投机或供应中断导致,但如果是稳定的低波动上涨,则更可能反映基本面的改善,对原油价格有更强的预测性。
3. 炼厂利润信号: 柴油/原油价差(裂解价差)是炼厂利润的重要指标。柴油价格的稳健上涨往往意味着下游需求旺盛,会传导至原油采购需求,推动油价上涨。
4. 跨品种套利: 当柴油相对原油表现出强动量时,可能触发炼厂增加开工率,进而增加原油需求,形成价格传导机制。

```
def build_diesel_mom_factor(raw_data: pd.DataFrame, config: dict = None) →
pd.Series:
    """
    因子名称: 柴油波动率调整动量 (Diesel Volatility Adjusted Momentum)

    Args:
        raw_data (pd.DataFrame): 包含 'Date' 和 'Price' 列的柴油价格数据,包含多地区数据
        config (dict): 可选参数
                    - window: 滚动窗口大小,默认20
                    - region: 指定地区代码,可选: 'Y35NY'(纽约港), 'RGC'(墨西哥湾),
'Y05LA'(洛杉矶)

    Returns:
        pd.Series: 因子值序列,索引为日期
    """

    # 1. 参数设置
    if config is None:
        config = {}
    window = config.get('window', 20)
    target_region = config.get('region')

    # 2. 数据清洗与筛选
    df = raw_data.copy()
```

```python
    # 优先使用指定地区,如果没有则使用所有数据的平均
    region_data = df[df['duoarea'] == target_region]
    if len(region_data) == 0:
        # 按日期分组取平均
        df = df.groupby('Date').agg({'Price': 'mean'}).reset_index()
    else:
        df = region_data[['Date', 'Price']].copy()

    # 3. 设置时间索引
    df['Date'] = pd.to_datetime(df['Date'])
    df = df.set_index('Date')
    df = df.sort_index()

    # 4. 处理重复日期(取平均值)
    df = df.groupby(df.index).mean()

    # 5. 数据预处理
    # 前向填充处理少量缺失值(最多3天)
    prices = df['Price'].ffill(limit=3)

    # 过滤异常值(价格变化超过50%视为异常)
    returns = prices.pct_change()
    returns = returns.clip(lower=-0.5, upper=0.5)  # 限制极端值

    # 6. 核心因子计算
    # 滚动窗口计算均值和标准差
    rolling_mean = returns.rolling(window=window, min_periods=window).mean()
    rolling_std = returns.rolling(window=window, min_periods=window).std()

    # 计算波动率调整后的动量
    # 加入极小值1e-9防止除零错误
    factor = rolling_mean / (rolling_std + 1e-9)

    # 7. 异常值处理
    # 限制因子值在合理范围内(-10, 10)
    factor = factor.clip(lower=-10, upper=10)

    # 8. 格式化输出
    factor.name = "diesel_vol_adj_mom_20d_v1"

    # 移除缺失值
    return factor.dropna()
```

基于美国汽油现货价格的原油价格预测多因子模块（包含因子2~4）：

1. gasoline_vol_adj_mom_20d_v1: 波动率调整动量因子
2. gasoline_crack_spread_mom_10d_v1: 裂解价差动量因子
3. gasoline_demand_strength_5d_v1: 短期需求强度因子

```
# 主函数：构建所有因子
def build_all_gas_factors(gasoline_data: pd.DataFrame, crude_data: pd.DataFrame =
None) → pd.DataFrame:
    """
    构建所有三个因子并合并为一个DataFrame
    Args:
        gasoline_data: 汽油价格数据
        crude_data: 原油价格数据(因子2需要)

    Returns:
        pd.DataFrame: 包含所有因子的数据框
    """
    factor1 = build_gas_mom_factor_(gasoline_data)
    factor2 = build_gas_spread_factor(gasoline_data, crude_data)
    factor3 = build_gas_demand_factor(gasoline_data)

    # 合并因子
    all_factors = pd.DataFrame({
        'factor1_vol_adj_mom': factor1,
        'factor2_crack_spread_mom': factor2,
        'factor3_demand_strength': factor3
    })

    return all_factors
```

因子2：汽油波动率调整动量因子（gasoline_vol_adj_mom_20d_v1）

来源：单表USA Daily Spot Prices for Conventional Gasoline.csv

公式：过去20日的汽油价格平均收益率 / 过去20日的收益率标准差

逻辑：

1. 裂解价差传导: 汽油是原油最主要的下游产品(约占炼油产出的45%),汽油价格的强劲上涨反映终端消费需求旺盛,会传导至炼厂增加开工率,进而增加原油采购需求,推动原油价格上涨。这种传导在成品油库存偏低时尤为显著。

2. 风险调整的需求信号: 单纯的汽油价格上涨可能由季节性因素(夏季驾驶高峰)或供应中断(炼厂检修)导致,持续性较弱。但如果汽油价格是在低波动环境下稳步上涨,则更可能反映基本面的持续改善,对原油价格有更强的预测性。
3. 套利机制: 当汽油/原油价差(Crack Spread)扩大时,会吸引炼厂增产汽油以获取更高利润。炼厂增产需要更多原油原料,这会推动原油现货采购,形成价格传导。该因子通过捕捉汽油的稳定动量,间接反映这种套利驱动的需求。

```python
def build_gas_mom_factor_(raw_data: pd.DataFrame, config: dict = None) →
pd.Series:
    """
    因子名称: 汽油波动率调整动量 (Gasoline Volatility Adjusted Momentum)

    Args:
        raw_data (pd.DataFrame): 包含 'Date' 和 'Value' 列的汽油价格数据
        config (dict): 可选参数,默认 window=20

    Returns:
        pd.Series: 因子值序列,索引为日期
    """
    # 1. 数据清洗与索引设置
    df = raw_data.copy()

    # 兼容不同的列名输入
    price_col = 'Value' if 'Value' in df.columns else df.columns[2]

    # 确保时间索引格式正确
    if 'Date' in df.columns:
        df = df.set_index('Date')
    elif 'date' in df.columns:
        df = df.set_index('date')

    df.index = pd.to_datetime(df.index)
    df = df.sort_index()

    # 处理多地区数据,按日期取平均
    if len(df.index.unique()) < len(df):
        df = df.groupby(df.index)[price_col].mean().to_frame()
        df.columns = [price_col]

    # 2. 计算日收益率
    # 简单的 ffill 处理微小的缺失,避免因子计算中断
    prices = df[price_col].ffill(limit=3)
    returns = prices.pct_change()
```

```
    # 3. 核心逻辑:滚动窗口计算
    window = 20  # 对应一个月交易日
    rolling_mean = returns.rolling(window=window).mean()
    rolling_std = returns.rolling(window=window).std()

    # 4. 计算因子值（加上极小值 1e-9 防止除零）
    factor = rolling_mean / (rolling_std + 1e-9)

    # 5. 防泄露处理（Lag）
    # t 时刻看到的因子,只能用 t 时刻收盘前的数据计算
    # 回测时通常取 factor.shift(1) 匹配 return_t
    # 但在构建因子文件本身时,我们输出"t 时刻对应的因子值"即可

    # 格式化输出
    factor.name = "gasoline_vol_adj_mom_20d_v1"
    return factor.dropna()
```

因子3：汽油裂解价差动量因子（gasoline_crack_spread_mom_10d_v1）

来源：双表 USA Daily Spot Prices for Conventional Gasoline.csv, Cushing_OK_WTI_Spot_Price_FOB.csv（删去表头说明行）

公式：过去10日汽油价格收益率 - 过去10日原油价格收益率的滚动均值

逻辑：过去 10 个交易日汽油价格相对于布伦特原油价格的超额收益率动量

1. 炼厂利润指标: 裂解价差(Crack Spread)直接反映炼厂的加工利润。当汽油价格涨幅超过原油时,裂解价差扩大,炼厂利润增加,这会激励炼厂提高开工率,增加原油需求,从而推动原油价格上涨。
2. 供需错配信号: 裂解价差扩大往往意味着下游成品油需求强于上游原油供应的增长,这种错配最终会通过价格机制自我修正——原油价格上涨以平衡供需。
3. 领先指标特性: 由于汽油消费更直接反映终端需求,汽油价格的相对强势往往领先于原油市场对需求改善的认知,提供了1-5个交易日的预测窗口。

```
def build_gas_spread_factor(gasoline_data: pd.DataFrame, crude_data: pd.DataFrame,
                            config: dict = None) -> pd.Series:
    """
    因子名称: 汽油裂解价差动量 (Crack Spread Momentum)

    Args:
        gasoline_data (pd.DataFrame): 汽油价格数据
        crude_data (pd.DataFrame): 原油价格数据
```

```
        config (dict): 可选参数,默认 window=10

    Returns:
        pd.Series: 因子值序列,索引为日期
    """
    # 1. 处理汽油数据
    gas_df = gasoline_data.copy()
    gas_price_col = 'Value' if 'Value' in gas_df.columns else gas_df.columns[2]

    if 'Date' in gas_df.columns:
        gas_df = gas_df.set_index('Date')
    gas_df.index = pd.to_datetime(gas_df.index)
    gas_df = gas_df.sort_index()

    # 处理多地区数据
    if len(gas_df.index.unique()) < len(gas_df):
        gas_df = gas_df.groupby(gas_df.index)[gas_price_col].mean().to_frame()
        gas_df.columns = [gas_price_col]

    gas_prices = gas_df[gas_price_col].ffill(limit=3)
    gas_returns = gas_prices.pct_change()

    # 2. 处理原油数据
    crude_df = crude_data.copy()
    # 布伦特数据格式: Day, Price
    if len(crude_df.columns) > 1:
        crude_df.columns = ['Date', 'Price']
        crude_df['Date'] = pd.to_datetime(crude_df['Date'], format='%m/%d/%Y',
errors='coerce')
        crude_df = crude_df.dropna(subset=['Date'])
        crude_df = crude_df.set_index('Date')
        crude_df['Price'] = pd.to_numeric(crude_df['Price'], errors='coerce')

    crude_df = crude_df.sort_index()
    crude_prices = crude_df['Price'].ffill(limit=3)
    crude_returns = crude_prices.pct_change()

    # 3. 合并数据计算超额收益
    combined = pd.DataFrame({
        'gas_ret': gas_returns,
        'crude_ret': crude_returns
    }).dropna()

    # 计算超额收益率(汽油相对原油)
```

```
    excess_return = combined['gas_ret'] - combined['crude_ret']

    # 4. 滚动窗口计算动量
    window = 10  # 10日窗口捕捉短期价差趋势
    factor = excess_return.rolling(window=window).mean()

    # 格式化输出
    factor.name = "gasoline_crack_spread_mom_10d_v1"
    return factor.dropna()
```

因子4： 汽油短期需求强度因子（gasoline_demand_strength_5d_v1）

来源：单表USA Daily Spot Prices for Conventional Gasoline.csv

公式：过去5日上涨天数占比 - 下跌天数占比

逻辑：过去 5 个交易日汽油价格上涨天数占比减去下跌天数占比

1. 需求持续性: 与传统动量因子不同,本因子关注价格变化的"方向一致性"而非幅度。连续多日上涨(即使涨幅不大)往往反映持续的买盘压力和稳定的需求增长,这种持续性需求最终会传导至原油市场。
2. 市场情绪: 该因子捕捉短期市场情绪和交易者预期。当汽油价格连续上涨时,市场参与者会预期原油需求增加,提前布局原油多头,形成自我实现的预期。
3. 噪音过滤: 通过统计上涨/下跌天数而非价格变化幅度,该因子对极端值和异常波动不敏感,能更稳定地反映市场趋势,适合作为主因子的补充验证信号。

```
def build_gas_demand_factor(raw_data: pd.DataFrame, config: dict = None) →
pd.Series:
    """
    因子名称: 5日需求强度因子 (Demand Strength Indicator)

    Args:
        raw_data (pd.DataFrame): 汽油价格数据
        config (dict): 可选参数,默认 window=5

    Returns:
        pd.Series: 因子值序列,范围 [-1, 1]
    """
    # 1. 数据清洗
    df = raw_data.copy()
    price_col = 'Value' if 'Value' in df.columns else df.columns[2]

    if 'Date' in df.columns:
```

```
        df = df.set_index('Date')
    df.index = pd.to_datetime(df.index)
    df = df.sort_index()

    # 处理多地区数据
    if len(df.index.unique()) < len(df):
        df = df.groupby(df.index)[price_col].mean().to_frame()
        df.columns = [price_col]

    prices = df[price_col].ffill(limit=3)

    # 2. 计算日收益率
    returns = prices.pct_change()

    # 3. 判断涨跌方向（1=上涨, -1=下跌, 0=持平）
    direction = returns.apply(lambda x: 1 if x > 0 else (-1 if x < 0 else 0))

    # 4. 滚动窗口统计
    window = 5  # 5日短期窗口

    # 计算上涨天数占比
    up_ratio = direction.rolling(window=window).apply(
        lambda x: (x > 0).sum() / len(x), raw=True
    )

    # 计算下跌天数占比
    down_ratio = direction.rolling(window=window).apply(
        lambda x: (x < 0).sum() / len(x), raw=True
    )

    # 5. 计算因子值: 上涨占比 - 下跌占比
    # 范围 [-1, 1]: 1表示5天全涨, -1表示5天全跌, 0表示涨跌均衡
    factor = up_ratio - down_ratio

    # 格式化输出
    factor.name = "gasoline_demand_strength_5d_v1"
    return factor.dropna()
```

基于美国取暖油现货价格的原油价格预测多因子模块（包含因子5~8）：

1. heating_vol_adj_mom_20d_v1: 波动率调整动量因子
2. heating_seasonal_strength_15d_v1: 季节性需求强度因子
3. heating_crack_reversal_10d_v1: 裂解价差均值回归因子
4. heating_vol_regime_5d_v1: 波动率制度转换因子

```
# 主函数：构建所有因子
def build_all_heating_factors(heating_data: pd.DataFrame, crude_data: pd.DataFrame
= None) → pd.DataFrame:
    """
    构建所有四个因子并合并为一个DataFrame

    Args:
        heating_data: 取暖油价格数据
        crude_data: 原油价格数据(因子3需要)

    Returns:
        pd.DataFrame: 包含所有因子的数据框
    """
    factor1 = build_heating_mom_factor(heating_data)
    factor2 = build_heating_demand_factor(heating_data)
    factor3 = build_heating_rev_factor(heating_data, crude_data)
    factor4 = build_heating_shift_factor(heating_data)

    # 合并因子
    all_factors = pd.DataFrame({
        'factor1_vol_adj_mom': factor1,
        'factor2_seasonal_strength': factor2,
        'factor3_crack_reversal': factor3,
        'factor4_vol_regime': factor4
    })

    return all_factors
```

因子5：取暖油波动率调整动量（heating_vol_adj_mom_20d_v1）

来源：单表USA Daily Spot Prices for Heating Oil.csv

公式：过去20日的取暖油价格平均收益率 / 过去20日的收益率标准差

逻辑：过去 20 个交易日的取暖油现货价格日均收益率除以日收益率的标准差

1. 冬季采暖需求传导: 取暖油主要用于北方地区冬季供暖,其需求具有明显季节性。当取暖油价格稳步上涨时(高动量低波动),往往反映真实的采暖需求增长,这会传导至炼厂增加开工率,进而推动原油采购需求,最终使原油价格上涨。
2. 风险调整的供需信号: 取暖油价格的单纯上涨可能由寒潮预期、库存偏低等短期因素驱动,持续性较弱。但如果价格是在低波动环境下稳定上涨,则更可能反映基本面的结构性变化(如长期气候模式改变、替代能源价格上升等),对原油价格有更强的预测能力。
3. 炼厂套利机制: 取暖油/原油价差扩大会激励炼厂调整产品结构,增加取暖油产出比例。由于炼厂总产能有限,增产取暖油意味着需要更多原油原料,形成对原油的额外需求,推动价格上涨。该因子通过稳定动量捕捉这种套利驱动。

```
def build_heating_mom_factor(raw_data: pd.DataFrame, config: dict = None) →
pd.Series:
    """
    因子名称: 取暖油波动率调整动量 (Heating Oil Volatility Adjusted Momentum)

    Args:
        raw_data (pd.DataFrame): 包含 'Date' 和价格列的取暖油价格数据
        config (dict): 可选参数,默认 window=20

    Returns:
        pd.Series: 因子值序列,索引为日期
    """
    # 1. 数据清洗与索引设置
    df = raw_data.copy()

    # 兼容不同的列名输入
    price_col = 'New York Harbor No. 2 Heating Oil Spot Price FOB' if \
                'New York Harbor No. 2 Heating Oil Spot Price FOB' in df.columns
else df.columns[1]

    # 确保时间索引格式正确
    if 'Date' in df.columns:
        df = df.set_index('Date')
    elif 'date' in df.columns:
        df = df.set_index('date')
```

```
    df.index = pd.to_datetime(df.index)
    df = df.sort_index()

    # 2. 计算日收益率
    # 简单的 ffill 处理微小的缺失,避免因子计算中断
    prices = df[price_col].ffill(limit=3)
    returns = prices.pct_change()

    # 3. 核心逻辑:滚动窗口计算
    window = 20  # 对应一个月交易日
    rolling_mean = returns.rolling(window=window).mean()
    rolling_std = returns.rolling(window=window).std()

    # 4. 计算因子值（加上极小值 1e-9 防止除零）
    factor = rolling_mean / (rolling_std + 1e-9)

    # 格式化输出
    factor.name = "heating_vol_adj_mom_20d_v1"
    return factor.dropna()
```

因子6：取暖油季节性需求强度因子（heating_seasonal_strength_15d_v1）

来源：单表USA Daily Spot Prices for Heating Oil.csv

公式：(当前15日均价 - 历史同期15日均价) / 历史同期标准差

逻辑：过去 15 日取暖油价格相对于历史同期均值的偏离程度

1. 季节性异常捕捉: 取暖油需求具有强烈的季节性特征(冬季需求高,夏季需求低)。本因子通过比较当前价格与历史同期均值,识别超出正常季节性波动的异常需求。这种异常往往预示着供需失衡,会传导至原油市场。
2. 气候风险溢价: 当取暖油价格显著高于历史同期水平时,可能反映极端天气事件(如超级寒潮)的风险溢价。这种风险会提升市场对能源供应安全的担忧,推高整个能源板块包括原油的风险溢价,导致价格联动上涨。
3. 库存周期领先: 取暖油价格的季节性偏离往往领先于库存数据的公布。当价格异常高企时,意味着实际库存可能低于预期,这会触发市场参与者提前锁定供应,增加对上游原油的采购,形成领先指标。

```
def build_heating_demand_factor(raw_data: pd.DataFrame, config: dict = None) →
pd.Series:
    """
```

```
    因子名称: 季节性需求强度 (Seasonal Demand Strength)

    Args:
        raw_data (pd.DataFrame): 取暖油价格数据
        config (dict): 可选参数

    Returns:
        pd.Series: 因子值序列,标准化后的季节性偏离
    """
    # 1. 数据处理
    df = raw_data.copy()
    price_col = 'New York Harbor No. 2 Heating Oil Spot Price FOB' if \
                'New York Harbor No. 2 Heating Oil Spot Price FOB' in df.columns
else df.columns[1]

    if 'Date' in df.columns:
        df = df.set_index('Date')
    df.index = pd.to_datetime(df.index)
    df = df.sort_index()

    prices = df[price_col].ffill(limit=3)

    # 2. 计算15日移动平均
    window = 15
    ma_15 = prices.rolling(window=window).mean()

    # 3. 提取月份和日期信息
    ma_15_df = ma_15.to_frame()
    ma_15_df.columns = ['price']
    ma_15_df['month'] = ma_15_df.index.month
    ma_15_df['day'] = ma_15_df.index.day

    # 4. 计算历史同期(相同月份±7天)的均值和标准差
    factor_values = []
    for idx, row in ma_15_df.iterrows():
        if pd.isna(row['price']):
            factor_values.append(np.nan)
            continue

        current_month = row['month']
        current_day = row['day']
        current_year = idx.year

        # 筛选历史同期数据(相同月份,且年份<当前年份)
```

```
            historical = ma_15_df[
                (ma_15_df['month'] == current_month) &
                (ma_15_df.index.year < current_year)
            ]['price'].dropna()

            if len(historical) < 5:  # 历史数据不足
                factor_values.append(np.nan)
                continue

            # 计算季节性偏离 =（当前 - 历史均值）/ 历史标准差
            hist_mean = historical.mean()
            hist_std = historical.std()

            deviation = (row['price'] - hist_mean) / (hist_std + 1e-9)
            factor_values.append(deviation)

        # 5. 构建因子序列
        factor = pd.Series(factor_values, index=ma_15_df.index)
        factor.name = "heating_seasonal_strength_15d_v1"

        return factor.dropna()
```

因子7：取暖油裂解价差均值回归因子（heating_crack_reversal_10d_v1）

来源：双表USA Daily Spot Prices for Heating Oil.csv, Cushing_OK_WTI_Spot_Price_FOB.csv（删去表头说明行）

公式：-(当前价差 - 长期均值价差) / 长期标准差

负号表示反向：价差过高时预期原油上涨以收窄价差

逻辑：取暖油/原油价差相对于其长期均值的偏离程度的反向信号

1. 价差均值回归: 裂解价差(取暖油价格-原油价格)受炼厂成本约束,长期围绕稳定中枢波动。当价差过度偏离历史均值时,市场会通过套利机制使其回归:
   - 价差过高→炼厂增产取暖油→原油需求增加→原油价格上涨→价差收窄
   - 价差过低→炼厂减产取暖油→原油需求减少→原油价格下跌→价差扩大
2. 套利力量平衡: 专业套利者(如对冲基金、炼厂自营)会监控裂解价差。当价差偏离合理区间时,大规模套利交易会迅速启动,这种交易行为本身就会推动原油价格向均衡方向调整,产生可预测的价格动量。
3. 炼厂利润指引: 价差是炼厂利润的直接指标。过高的价差虽短期利好炼厂,但会刺激产能扩张和原油

采购,中期反而推高原油成本;过低的价差则抑制炼厂开工,减少原油需求。该因子捕捉这种动态平衡的调整过程。

```
def build_heating_rev_factor(heating_data: pd.DataFrame, crude_data: pd.DataFrame,
                             config: dict = None) -> pd.Series:
    """
    因子名称: 裂解价差均值回归 (Crack Spread Mean Reversion)

    Args:
        heating_data: 取暖油价格数据
        crude_data: 原油价格数据
        config: 可选参数

    Returns:
        pd.Series: 因子值序列
    """
    # 1. 处理取暖油数据
    ho_df = heating_data.copy()
    ho_price_col = 'New York Harbor No. 2 Heating Oil Spot Price FOB' if \
                   'New York Harbor No. 2 Heating Oil Spot Price FOB' in
ho_df.columns else ho_df.columns[1]

    if 'Date' in ho_df.columns:
        ho_df = ho_df.set_index('Date')
    ho_df.index = pd.to_datetime(ho_df.index)
    ho_df = ho_df.sort_index()

    ho_prices = ho_df[ho_price_col].ffill(limit=3)

    # 2. 处理原油数据
    crude_df = crude_data.copy()
    if len(crude_df.columns) > 1:
        crude_df.columns = ['Date', 'Price']
        crude_df['Date'] = pd.to_datetime(crude_df['Date'], format='%m/%d/%Y',
errors='coerce')
        crude_df = crude_df.dropna(subset=['Date'])
        crude_df = crude_df.set_index('Date')
        crude_df['Price'] = pd.to_numeric(crude_df['Price'], errors='coerce')

    crude_df = crude_df.sort_index()
    crude_prices = crude_df['Price'].ffill(limit=3)

    # 3. 对齐数据并计算价差
    # 注意:取暖油是$/加仑,原油是$/桶,需要单位换算
```

```
        # 1桶原油 ≈ 42加仑,所以原油$/桶 ÷ 42 = $/加仑
        combined = pd.DataFrame({
            'heating': ho_prices,
            'crude_per_gallon': crude_prices / 42.0
        }).dropna()

        # 裂解价差 = 取暖油价格 - 原油价格(转换为$/加仑)
        crack_spread = combined['heating'] - combined['crude_per_gallon']

        # 4. 计算价差的长期均值和标准差(使用60日窗口)
        lookback = 60
        spread_mean = crack_spread.rolling(window=lookback).mean()
        spread_std = crack_spread.rolling(window=lookback).std()

        # 5. 计算价差偏离(标准化)
        spread_deviation = (crack_spread - spread_mean) / (spread_std + 1e-9)

        # 6. 均值回归信号 = -偏离度
        # 负号的含义:当价差过高(正偏离)时,预期原油上涨以收窄价差,因此返回负值作为原油看涨信号
        # 当价差过低(负偏离)时,预期原油下跌以扩大价差,因此返回正值作为原油看跌信号
        # 为了统一语义取负号,使因子值为正时表示看涨原油
        factor = -spread_deviation

        # 7. 平滑处理(10日移动平均)
        factor = factor.rolling(window=10).mean()

        # 格式化输出
        factor.name = "heating_crack_reversal_10d_v1"
        return factor.dropna()
```

因子8：取暖油波动率制度转换因子（heating_vol_regime_5d_v1）

来源：单表USA Daily Spot Prices for Heating Oil.csv

公式：(过去5日波动率 / 过去60日波动率) - 1
正值表示波动率上升,负值表示波动率下降

逻辑：过去 5 日取暖油价格波动率相对于过去 60 日波动率的比值

1. 市场情绪转换: 波动率的突然上升往往标志着市场从平静状态进入不确定状态, 这可能由突发事件(寒潮预警、供应中断、地缘风险)引发。这种不确定性会迅速传染至原油市场,因为能源品种高度关联,波动率会产生联动效应。

2. 流动性与价格发现: 低波动环境下,市场流动性充裕,价格发现机制有效;高波动环境下,流动性下降,价格容易超调。通过识别波动率制度转换,可以预判原油市场即将进入高波动期,此时趋势跟踪策略效果更好。
3. VIX效应: 类似于股市的VIX指数,取暖油波动率的上升反映市场恐慌情绪。在能源市场,恐慌往往导致囤积行为,推高包括原油在内的所有能源品种价格。该因子捕捉这种情绪传染带来的价格联动。

```
def build_heating_shift_factor(raw_data: pd.DataFrame, config: dict = None) →
pd.Series:
    """
    因子名称: 波动率制度转换 (Volatility Regime Shift)

    Args:
        raw_data: 取暖油价格数据
        config: 可选参数

    Returns:
        pd.Series: 因子值序列
    """
    # 1. 数据处理
    df = raw_data.copy()
    price_col = 'New York Harbor No. 2 Heating Oil Spot Price FOB' if \
                'New York Harbor No. 2 Heating Oil Spot Price FOB' in df.columns
else df.columns[1]

    if 'Date' in df.columns:
        df = df.set_index('Date')
    df.index = pd.to_datetime(df.index)
    df = df.sort_index()

    prices = df[price_col].ffill(limit=3)
    returns = prices.pct_change()

    # 2. 计算短期和长期波动率
    short_window = 5
    long_window = 60

    vol_short = returns.rolling(window=short_window).std()
    vol_long = returns.rolling(window=long_window).std()

    # 3. 计算波动率比率
    # 比率 > 1 表示波动率上升,< 1 表示波动率下降
    vol_ratio = vol_short / (vol_long + 1e-9)
```

```
    # 4. 转换为偏离度
    factor = vol_ratio - 1

    # 格式化输出
    factor.name = "heating_vol_regime_5d_v1"
    return factor.dropna()
```

基于美国汽油和柴油价格的原油价格预测多因子模块（包含因子9~13）：

1. composite_vol_adj_mom_20d_v1: 综合成品油波动率调整动量因子
2. product_spread_divergence_15d_v1: 成品油价差分化因子
3. diesel_gasoline_ratio_mom_10d_v1: 柴汽比价动量因子
4. refinery_margin_pressure_20d_v1: 炼厂利润压力因子
5. demand_structure_shift_30d_v1: 需求结构转换因子

```
# 主函数：构建所有因子
def build_all_diesel_gas_factors(gasoline_data: pd.DataFrame, diesel_data:
pd.DataFrame,
                      crude_data: pd.DataFrame = None) -> pd.DataFrame:
    """
    构建所有五个因子并合并为一个DataFrame

    Args:
        gasoline_data: 汽油价格数据
        diesel_data: 柴油价格数据
        crude_data: 原油价格数据(因子4需要)

    Returns:
        pd.DataFrame: 包含所有因子的数据框
    """
    factor1 = build_diesel_gas_vol_mom_factor(gasoline_data, diesel_data)
    factor2 = build_diesel_gas_div_factor(gasoline_data, diesel_data)
    factor3 = build_diesel_gas_ratio_mom_factor(gasoline_data, diesel_data)
    factor4 = build_diesel_gas_margin_pressure_factor(gasoline_data, diesel_data,
crude_data)
    factor5 = build_diesel_gas_demand_shift_factor(gasoline_data, diesel_data)

    # 合并因子
    all_factors = pd.DataFrame({
        'factor1_composite_mom': factor1,
```

```
        'factor2_spread_divergence': factor2,
        'factor3_ratio_mom': factor3,
        'factor4_margin_pressure': factor4,
        'factor5_demand_shift': factor5
    })

    return all_factors
```

因子9：综合成品油波动率调整动量因子（composite_vol_adj_mom_20d_v1）

来源：双表USA Daily Spot Prices for Conventional Gasoline.csv, USA Daily Diesel Spot Price.csv

公式：(0.6 * 汽油动量 + 0.4 * 柴油动量)
　　权重基于产量占比:汽油约占45%,柴油约占30%,归一化后得到6:4

逻辑：汽油和柴油价格的加权平均动量,经波动率调整后的综合信号

1. 全产业链需求信号: 汽油反映交通出行需求,柴油反映工业运输需求,两者结合可以更全面地反映经济活动强度。当两者同步走强时,表明经济基本面改善,对原油需求形成强劲支撑,预示原油价格上涨。
2. 风险分散与信号增强: 单一成品油价格可能受到特定因素扰动(如汽油受季节性影响,柴油受工业周期影响),通过加权平均可以过滤掉特异性噪音,提取共性的需求信号,使预测更稳健。
3. 炼厂综合利润: 炼厂同时生产汽油和柴油,综合成品油价格的强势意味着炼厂整体盈利能力强,会激励炼厂扩大产能和提高开工率,这必然增加原油采购需求,形成对原油价格的支撑。

```
def build_diesel_gas_vol_mom_factor(gasoline_data: pd.DataFrame, diesel_data:
pd.DataFrame,
                    config: dict = None) -> pd.Series:
    """
    因子名称: 综合成品油波动率调整动量 (Composite Products Volatility Adjusted Momentum)

    Args:
        gasoline_data (pd.DataFrame): 汽油价格数据
        diesel_data (pd.DataFrame): 柴油价格数据
        config (dict): 可选参数,默认 window=20

    Returns:
        pd.Series: 因子值序列,索引为日期
    """

    def calc_vol_adj_mom(df, price_col_name):
        """计算单个产品的波动率调整动量"""
        df = df.copy()
```

```
    # 自动识别价格列
    if price_col_name in df.columns:
        price_col = price_col_name
    elif 'Value' in df.columns:
        price_col = 'Value'
    else:
        price_col = df.columns[2] if len(df.columns) > 2 else df.columns[1]

    # 设置时间索引
    if 'Date' in df.columns:
        df = df.set_index('Date')
    df.index = pd.to_datetime(df.index)
    df = df.sort_index()

    # 处理多地区数据
    if len(df.index.unique()) < len(df):
        df = df.groupby(df.index)[price_col].mean().to_frame()
        df.columns = [price_col]

    # 计算收益率
    prices = df[price_col].ffill(limit=3)
    returns = prices.pct_change()

    # 滚动窗口计算
    window = 20
    rolling_mean = returns.rolling(window=window).mean()
    rolling_std = returns.rolling(window=window).std()

    # 波动率调整动量
    factor = rolling_mean / (rolling_std + 1e-9)

    return factor

# 1. 计算汽油动量
gasoline_mom = calc_vol_adj_mom(gasoline_data, 'Value')

# 2. 计算柴油动量
diesel_mom = calc_vol_adj_mom(diesel_data, 'Price')

# 3. 加权合成
combined = pd.DataFrame({
    'gas': gasoline_mom,
    'diesel': diesel_mom
```

```
    }).dropna()

    # 权重:汽油60%,柴油40%（基于典型炼厂产出结构）
    factor = 0.6 * combined['gas'] + 0.4 * combined['diesel']

    # 格式化输出
    factor.name = "composite_vol_adj_mom_20d_v1"
    return factor.dropna()
```

因子10：成品油价差分化因子（product_spread_divergence_15d_v1）

来源：双表USA Daily Spot Prices for Conventional Gasoline.csv, USA Daily Diesel Spot Price.csv

公式：(汽油15日收益率 - 柴油15日收益率) 的绝对值
高值表示两者走势分化,低值表示同步

逻辑：汽油和柴油价格走势的分化程度,衡量成品油市场的内部结构变化

1. 需求结构失衡: 当汽油和柴油价格走势出现显著分化时,往往反映需求结构的不平衡。例如汽油强而柴油弱,可能意味着消费性需求强但工业需求弱,这种结构性矛盾最终会传导至原油市场,引发价格调整。
2. 炼厂产能约束: 炼厂在固定原油投入下,汽油和柴油的产出比例相对稳定。当某一产品价格单边走强时,炼厂无法快速调整产品结构,会转而增加原油采购以满足强势产品的需求,推动原油价格上涨。
3. 套利机会信号: 价差分化为套利者提供了交易机会。大规模的跨品种套利交易会影响各品种的供需平衡,这种套利力量的再平衡过程会对原油价格产生可预测的影响。

```
def build_diesel_gas_div_factor(gasoline_data: pd.DataFrame, diesel_data:
pd.DataFrame,
                    config: dict = None) -> pd.Series:
    """
    因子名称: 成品油价差分化 (Products Spread Divergence)

    Args:
        gasoline_data: 汽油价格数据
        diesel_data: 柴油价格数据
        config: 可选参数

    Returns:
        pd.Series: 因子值序列
    """
```

```
    def get_returns(df, price_col_pattern):
        """提取收益率序列"""
        df = df.copy()

        # 识别价格列
        if isinstance(price_col_pattern, str) and price_col_pattern in df.columns:
            price_col = price_col_pattern
        elif 'Value' in df.columns:
            price_col = 'Value'
        elif 'Price' in df.columns:
            price_col = 'Price'
        else:
            price_col = df.columns[2] if len(df.columns) > 2 else df.columns[1]

        if 'Date' in df.columns:
            df = df.set_index('Date')
        df.index = pd.to_datetime(df.index)
        df = df.sort_index()

        # 处理多地区
        if len(df.index.unique()) < len(df):
            df = df.groupby(df.index)[price_col].mean()
        else:
            df = df[price_col]

        prices = df.ffill(limit=3)
        return prices.pct_change()

    # 1. 获取收益率
    gas_returns = get_returns(gasoline_data, 'Value')
    diesel_returns = get_returns(diesel_data, 'Price')

    # 2. 对齐数据
    combined = pd.DataFrame({
        'gas_ret': gas_returns,
        'diesel_ret': diesel_returns
    }).dropna()

    # 3. 计算15日累计收益率
    window = 15
    gas_cum_ret = combined['gas_ret'].rolling(window=window).sum()
    diesel_cum_ret = combined['diesel_ret'].rolling(window=window).sum()

    # 4. 计算分化度(差值的绝对值)
```

```
    divergence = (gas_cum_ret - diesel_cum_ret).abs()

    # 5. 标准化处理
    factor = (divergence - divergence.rolling(60).mean()) /
(divergence.rolling(60).std() + 1e-9)

    # 格式化输出
    factor.name = "product_spread_divergence_15d_v1"
    return factor.dropna()
```

因子11：柴汽比价动量因子（diesel_gasoline_ratio_mom_10d_v1）

来源：双表USA Daily Spot Prices for Conventional Gasoline.csv, USA Daily Diesel Spot Price.csv

公式：(当前柴汽比价 - 10日前柴汽比价) / 10日前柴汽比价

逻辑：柴油/汽油价格比值的变化趋势,反映产品结构和需求偏好

1. 经济周期指示: 柴汽比价是经济周期的领先指标。当柴油相对汽油走强(比价上升)时,往往意味着工业和物流需求旺盛,经济处于扩张期;反之则可能预示经济放缓。这种周期性变化会影响整体能源需求,从而影响原油价格。
2. 运输成本传导: 柴油主要用于货运,柴油价格上涨会提高物流成本,进而推高通胀预期。通胀预期的上升通常伴随大宗商品价格上涨,包括原油。因此柴汽比价上升可能预示原油价格上行压力。
3. 炼厂产品优化: 柴汽比价的变化会引导炼厂调整产品结构。当柴油相对利润更高时,炼厂会倾向于增加柴油产出,但这需要采购更多原油(尤其是重质原油),从而支撑原油价格。

```
def build_diesel_gas_ratio_mom_factor(gasoline_data: pd.DataFrame, diesel_data:
pd.DataFrame,
                                      config: dict = None) -> pd.Series:
    """
    因子名称: 柴汽比价动量 (Diesel/Gasoline Ratio Momentum)

    Args:
        gasoline_data: 汽油价格数据
        diesel_data: 柴油价格数据
        config: 可选参数

    Returns:
        pd.Series: 因子值序列
    """

    def get_prices(df, price_col_pattern):
```

```
    """提取价格序列"""
    df = df.copy()

    if isinstance(price_col_pattern, str) and price_col_pattern in df.columns:
        price_col = price_col_pattern
    elif 'Value' in df.columns:
        price_col = 'Value'
    elif 'Price' in df.columns:
        price_col = 'Price'
    else:
        price_col = df.columns[2] if len(df.columns) > 2 else df.columns[1]

    if 'Date' in df.columns:
        df = df.set_index('Date')
    df.index = pd.to_datetime(df.index)
    df = df.sort_index()

    if len(df.index.unique()) < len(df):
        df = df.groupby(df.index)[price_col].mean()
    else:
        df = df[price_col]

    return df.ffill(limit=3)

# 1. 获取价格
gas_prices = get_prices(gasoline_data, 'Value')
diesel_prices = get_prices(diesel_data, 'Price')

# 2. 对齐数据
combined = pd.DataFrame({
    'gas': gas_prices,
    'diesel': diesel_prices
}).dropna()

# 3. 计算柴汽比价
ratio = combined['diesel'] / (combined['gas'] + 1e-9)

# 4. 计算10日动量
window = 10
ratio_change = ratio.pct_change(periods=window)

# 5. 平滑处理
factor = ratio_change.rolling(window=5).mean()
```

```
    # 格式化输出
    factor.name = "diesel_gasoline_ratio_mom_10d_v1"
    return factor.dropna()
```

因子12：炼厂利润压力因子（refinery_margin_pressure_20d_v1）

来源： 三表USA Daily Spot Prices for Conventional Gasoline.csv, USA Daily Diesel Spot Price.csv, Cushing_OK_WTI_Spot_Price_FOB.csv（删去表头说明行）

公式：(综合裂解价差 - 60日均值) / 60日标准差
综合裂解价差 = 0.6*汽油价差* + *0.4*柴油价差

逻辑：综合裂解价差(成品油-原油价差)相对历史水平的偏离,衡量炼厂利润压力

1. 利润均值回归: 炼厂利润长期受成本约束,围绕合理中枢波动。当综合裂解价差过度偏离历史均值时,市场会通过调整原油和成品油价格使其回归。价差过低时,炼厂减产降低原油需求,原油价格承压;价差过高时,炼厂增产提振原油需求,原油价格获得支撑。
2. 产能利用率调整: 裂解价差直接决定炼厂的产能利用率。当价差压缩至盈亏平衡线附近时,边际炼厂会停工检修,减少原油采购;当价差扩大至高位时,闲置产能重启,增加原油需求。这种产能的弹性调整会对原油价格产生可预测影响。
3. 库存策略转变: 裂解价差的变化会影响炼厂的库存策略。低价差时,炼厂倾向于去库存、减少原油储备;高价差时,炼厂积极建库、增加原油采购。这种集体行为会放大对原油市场的影响。

```
def build_diesel_gas_margin_pressure_factor(gasoline_data: pd.DataFrame,
diesel_data: pd.DataFrame,
                   crude_data: pd.DataFrame, config: dict = None) → pd.Series:
    """
    因子名称: 炼厂利润压力 (Refinery Margin Pressure)

    Args:
        gasoline_data: 汽油价格数据
        diesel_data: 柴油价格数据
        crude_data: 原油价格数据
        config: 可选参数

    Returns:
        pd.Series: 因子值序列
    """

    def get_prices(df, price_col_pattern):
        """提取价格序列"""
```

```
        df = df.copy()

        if isinstance(price_col_pattern, str) and price_col_pattern in df.columns:
            price_col = price_col_pattern
        elif 'Value' in df.columns:
            price_col = 'Value'
        elif 'Price' in df.columns:
            price_col = 'Price'
        else:
            price_col = df.columns[1]

        if 'Date' in df.columns:
            df = df.set_index('Date')
        df.index = pd.to_datetime(df.index)
        df = df.sort_index()

        if len(df.index.unique()) < len(df):
            df = df.groupby(df.index)[price_col].mean()
        else:
            df = df[price_col]

        return df.ffill(limit=3)

    # 1. 获取价格数据
    gas_prices = get_prices(gasoline_data, 'Value')
    diesel_prices = get_prices(diesel_data, 'Price')

    # 处理原油数据
    crude_df = crude_data.copy()
    crude_df.columns = ['Date', 'Price']
    crude_df = crude_df.iloc[5:]
    crude_df['Date'] = pd.to_datetime(crude_df['Date'], format='%m/%d/%Y',
errors='coerce')
    crude_df = crude_df.dropna(subset=['Date'])
    crude_df = crude_df.set_index('Date')
    crude_df['Price'] = pd.to_numeric(crude_df['Price'], errors='coerce')
    crude_df = crude_df.sort_index()
    crude_prices = crude_df['Price'].ffill(limit=3)

    # 2. 对齐数据并转换单位
    # 汽油/柴油: $/加仑, 原油: $/桶, 转换: $/桶 ÷ 42 = $/加仑
    combined = pd.DataFrame({
        'gas': gas_prices,
        'diesel': diesel_prices,
```

```
            'crude_gallon': crude_prices / 42.0
        }).dropna()

        # 3. 计算各产品的裂解价差
        gas_crack = combined['gas'] - combined['crude_gallon']
        diesel_crack = combined['diesel'] - combined['crude_gallon']

        # 4. 计算综合裂解价差(加权平均)
        composite_crack = 0.6 * gas_crack + 0.4 * diesel_crack

        # 5. 计算相对历史水平的偏离
        lookback = 60
        crack_mean = composite_crack.rolling(window=lookback).mean()
        crack_std = composite_crack.rolling(window=lookback).std()

        # 标准化偏离度
        factor = (composite_crack - crack_mean) / (crack_std + 1e-9)

        # 6. 平滑处理
        factor = factor.rolling(window=20).mean()

        # 格式化输出
        factor.name = "refinery_margin_pressure_20d_v1"
        return factor.dropna()
```

因子13：需求结构转换因子（demand_structure_shift_30d_v1）

来源：双表USA Daily Spot Prices for Conventional Gasoline.csv, USA Daily Diesel Spot Price.csv

公式：(汽油30日动量 - 柴油30日动量) 的移动平均

逻辑：汽油和柴油需求强度的相对变化,识别经济结构转型

1. 经济结构变迁: 汽油消费主导的经济体(如美国)与柴油消费主导的经济体(如欧洲、中国)对原油需求的特征不同。通过监测汽柴油需求强度的相对变化,可以捕捉经济结构转型,这对中长期原油需求预测具有重要意义。
2. 消费升级信号: 当汽油需求相对柴油加速增长时,可能反映居民消费升级和私人汽车保有量增加,这种消费驱动型增长对原油需求的支撑更稳定,价格弹性更低,往往预示原油价格中枢上移。
3. 能源转型压力: 柴油需求的相对走弱可能反映新能源替代加速(如电动卡车、LNG卡车),这种结构性变化会影响市场对原油长期需求的预期,从而影响原油价格的风险溢价。

```python
def build_diesel_gas_demand_shift_factor(gasoline_data: pd.DataFrame, diesel_data:
pd.DataFrame,
                  config: dict = None) -> pd.Series:
    """
    因子名称: 需求结构转换 (Demand Structure Shift)

    Args:
        gasoline_data: 汽油价格数据
        diesel_data: 柴油价格数据
        config: 可选参数

    Returns:
        pd.Series: 因子值序列
    """

    def calc_momentum(df, price_col_pattern):
        """计算价格动量"""
        df = df.copy()

        if isinstance(price_col_pattern, str) and price_col_pattern in df.columns:
            price_col = price_col_pattern
        elif 'Value' in df.columns:
            price_col = 'Value'
        elif 'Price' in df.columns:
            price_col = 'Price'
        else:
            price_col = df.columns[2] if len(df.columns) > 2 else df.columns[1]

        if 'Date' in df.columns:
            df = df.set_index('Date')
        df.index = pd.to_datetime(df.index)
        df = df.sort_index()

        if len(df.index.unique()) < len(df):
            df = df.groupby(df.index)[price_col].mean()
        else:
            df = df[price_col]

        prices = df.ffill(limit=3)
        returns = prices.pct_change()

        # 30日累计收益率作为动量
        window = 30
        momentum = returns.rolling(window=window).sum()
```

```
    return momentum

# 1. 计算各自动量
gas_mom = calc_momentum(gasoline_data, 'Value')
diesel_mom = calc_momentum(diesel_data, 'Price')

# 2. 对齐数据
combined = pd.DataFrame({
    'gas_mom': gas_mom,
    'diesel_mom': diesel_mom
}).dropna()

# 3. 计算结构转换指标(差值)
structure_shift = combined['gas_mom'] - combined['diesel_mom']

# 4. 平滑处理(10日移动平均)
factor = structure_shift.rolling(window=10).mean()

# 格式化输出
factor.name = "demand_structure_shift_30d_v1"
return factor.dropna()
```