**做市策略**

**1预测价差**

**1. df["spread\_change\_3d"] = df["spread"].shift(-3) - df["spread"] 的作用**

**这段代码用于构造目标变量（未来3天的价差变化），具体逻辑如下：**

* **df["spread"]：当前价差（如国债ETF与期货的价差）。**
* **df["spread"].shift(-3)：将价差列向前移动3行（即取3天后的价差值）。**
* **相减操作：未来3天价差 - 当前价差，得到未来3天的价差变化量。**
  + **正值：价差扩大（未来上涨）。**
  + **负值：价差缩小（未来下跌）。**

**为什么这样设计？  
模型的目标是预测未来价差的方向性变化（涨/跌），而非绝对值。这种构造方式直接量化未来变化，适合用于交易信号生成。**

**2. LARS变量筛选的运行原理**

**LARS（Least Angle Regression，最小角回归）是一种线性回归变量选择算法，核心步骤如下：**

**(1) 算法流程**

1. **初始化：所有因子系数设为0，残差初始化为目标变量 y。**
2. **选择最相关因子：计算所有因子与当前残差的相关系数，选择相关性最强的因子。**
3. **沿“最小角”方向移动：沿已选因子的角平分线方向调整系数，直到另一个因子与残差的相关性与之相等。**
4. **迭代：重复上述过程，直到选满指定数量的因子（如5个）或残差足够小。**

**(2) 代码实现**

**python**

**lars = Lars(n\_nonzero\_coefs=max\_vars) *# 限制最多选max\_vars个因子***

**lars.fit(X\_win, y\_win) *# 在滚动窗口数据上训练***

**idx = np.where(lars.coef\_ != 0)[0] *# 获取被选中的因子索引***

* **输出：selected\_factors 是LARS筛选出的对价差变化影响最大的因子。**

**(3) 优势**

* **稀疏性：自动选择关键变量，避免过拟合。**
* **路径连续性：系数逐步调整，稳定性强于逐步回归。**

**3. 备选子模型的生成逻辑**

**(1) 组合生成**

**python**

**for k in range(1, min(len(selected\_factors), max\_vars)+1):**

**for comb in combinations(selected\_factors, k):**

* **遍历所有可能的子集：从LARS选中的因子中，生成所有1个、2个、...、最多5个因子的组合（共 C(5,1) + C(5,2) + ... + C(5,5) = 31 种可能）。**

**(2) 拟合线性回归**

**python**

**X\_sub = sm.add\_constant(X\_win[list(comb)]) *# 添加截距项***

**model = sm.OLS(y\_win, X\_sub).fit() *# 普通最小二乘回归***

**if model.rsquared\_adj > 0: *# 保留调整R²>0的模型***

**models.append((comb, model))**

* **目的：通过不同因子组合的线性回归，捕捉价差变化的多种解释路径。**

**(3) 为什么生成多模型？**

* **避免单一模型偏差：不同因子组合可能反映不同市场机制。**
* **集成学习：通过加权平均提升预测鲁棒性。**

**4. BIC加权定义与计算**

**(1) BIC（贝叶斯信息准则）**

**公式：**

**BIC=kln⁡n−2ln⁡L^BIC=*k*ln*n*−2ln*L*^**

* **k：模型参数数量（因子数+截距）。**
* **n：样本量（窗口期数据点数）。**
* **\hat{L}：模型似然函数最大值。**

**意义：BIC越小，模型在拟合优度和复杂度之间的平衡越好。**

**(2) 加权计算**

**python**

**bics = np.array([m[1].bic for m in models]) *# 所有模型的BIC值***

**dBIC = bics - bics.min() *# 相对BIC（减去最小值）***

**weights = np.exp(-dBIC/2) *# 权重公式：e^{-dBIC/2}***

**weights = weights / weights.sum() *# 归一化为概率***

* **权重逻辑：BIC越低的模型权重越大，但通过指数变换避免极端差异。**

**(3) 预测加权**

**python**

**y\_bayes = np.dot(weights, X\_pred\_subs) *# 加权平均预测值***

* **物理意义：高权重模型的预测结果对最终输出影响更大。**

**5. np.sign(y\_hat) 的作用**

**(1) 功能**

* **输入：y\_hat 是单个子模型对价差变化的预测值（可能为正/负）。**
* **输出：**
  + **np.sign(1.5) = 1**
  + **np.sign(-0.3) = -1**
  + **np.sign(0) = 0（但价差变化极少严格为0）。**

**(2) 在代码中的应用**

**python**

**X\_pred\_subs.append(np.sign(y\_hat) \* abs(y\_hat))**

* **保留方向信息：符号决定涨跌方向，绝对值用于加权计算。**
* **最终决策：加权平均后的 y\_bayes 的符号决定交易信号：**
  + **y\_bayes > 0 → 看涨**
  + **y\_bayes < 0 → 看跌**

**总结：模型与代码的对应关系**

| **模型步骤** | **代码实现** |
| --- | --- |
| **目标变量构造** | **df["spread"].shift(-3) - df["spread"]** |
| **LARS变量筛选** | **Lars(n\_nonzero\_coefs=5).fit() + 非零系数提取** |
| **备选子模型生成** | **combinations(selected\_factors, k) + sm.OLS().fit()** |
| **BIC加权** | **weights = np.exp(-(bics - bics.min())/2) + 归一化** |
| **方向信号** | **np.sign(y\_hat) 和最终 y\_bayes 的符号** |

**通过这种设计，模型实现了从因子选择到方向预测的端到端自动化流程，适合量化交易场景。**

**2对冲回测策略（LARS+BIC加权因子线性预测）**

* **预测信号**（如价差未来3日变化）作为开平仓/对冲的触发条件。
* **对冲手数**用你utils.dv01\_calc的hedge\_ratio动态调整。
* **做市报价**可在utils中先生成理论买卖价，然后模拟成交与对冲。

**3. 代码结构**

* utils/market\_maker.py：做市报价函数（如生成买卖价、盘口模拟等）
* utils/hedge\_engine.py：对冲执行与持仓管理
* backtest/backtest\_engine.py：回测主控，调用预测、做市、对冲、收益统计
* [data](vscode-file://vscode-app/f:/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)：存放日频/分钟级别数据

**4进一步改进**

* 日频只能做日内或隔夜策略，信号响应慢，回测结果仅供趋势判断。
* **分钟级别数据**可做更真实的做市与对冲回测，信号更及时，收益更贴近实盘。
* 你需要申请API获取分钟级别的ETF和期货数据（如Wind、聚宽、TuShare等）。