# 训练集测试集分割

在时间序列数据（例如股票数据）中，合理地分割训练集和测试集需要考虑数据的时间顺序，以确保模型能够在未来的数据上泛化。以下是一些常见的训练集和测试集分割方法，以及具体适用于股票预测的建议。

### 1. \*\*简单的时间切割（Simple Time-Based Split）\*\*

这种方法按时间顺序将数据分成两部分，前部分作为训练集，后部分作为测试集。这样可以确保测试集中的数据完全位于训练集的后面，避免数据泄露。

#### 优点

- 简单且符合时间序列数据的时间依赖性。

- 适合大多数时间序列任务，尤其是股票预测。

#### 示例代码

假设 `data` 是 DataFrame，且包含按日期排序的股票数据：

```python

# 按时间顺序分割

train\_ratio = 0.8

split\_idx = int(len(data) \* train\_ratio)

train\_data = data[:split\_idx] # 前 80% 作为训练集

test\_data = data[split\_idx:] # 后 20% 作为测试集

```

#### 注意事项

- 训练数据量应尽量充足，一般将 `70% - 80%` 的数据作为训练集。

- 如果预测股票的短期趋势，可以缩短测试集长度；若关注长期趋势，可以增大测试集长度。

### 2. \*\*滚动时间窗（Rolling Window）\*\*

滚动时间窗方法将整个数据集划分为多个时间窗，每个时间窗包含一段训练期和一个测试期。模型在每个时间窗上训练和测试，这样可以模拟模型在不同时间点上的表现。

#### 优点

- 可以观察模型在不同时期的表现，适合金融数据分析。

- 减少模型过拟合某一时间段的风险，提高模型对未来数据的稳定性。

#### 示例代码

```python

window\_size = 252 # 每个时间窗包含的训练期（例如 252 个交易日即一年的数据）

test\_size = 21 # 每个时间窗包含的测试期（例如 21 个交易日即一个月的数据）

for start in range(0, len(data) - window\_size - test\_size, test\_size):

train\_data = data[start:start + window\_size]

test\_data = data[start + window\_size:start + window\_size + test\_size]

# 这里可以在每个时间窗上训练并评估模型

```

#### 注意事项

- 适合需要长期稳定预测的场景。

- 测试期的长度可以根据预测周期进行调整，例如预测月度收益率可以设置 `test\_size=21`。

### 3. \*\*扩展窗口法（Expanding Window Split）\*\*

扩展窗口法从固定时间点开始，每次将训练集扩展到包含新的数据，但测试集始终保持在时间序列的后面。这样可以最大化利用已有的数据，同时保证模型在未来数据上的泛化能力。

#### 优点

- 每次使用的训练数据量增加，可以帮助模型更好地学习数据的长期趋势。

- 非常适合金融数据和股票预测，因为股票数据的长期趋势和模式较重要。

#### 示例代码

```python

initial\_train\_size = 252 # 初始训练集大小

test\_size = 21 # 每次测试的时间长度

start = 0

while start + initial\_train\_size + test\_size < len(data):

train\_data = data[:start + initial\_train\_size]

test\_data = data[start + initial\_train\_size:start + initial\_train\_size + test\_size]

start += test\_size

# 在当前扩展窗口上训练并测试模型

```

### 4. \*\*交叉验证方法\*\*

在时间序列中，传统的交叉验证方法（例如 K 折交叉验证）会打乱数据的时间顺序，因此在时间序列数据上不适用。对于时间序列，可以使用滚动时间窗或扩展窗口法进行“时间序列交叉验证”。

#### 优点

- 模拟模型在不同时期的表现，减少过拟合风险。

- 提供不同时间窗上的验证误差，可以更稳定地评估模型的泛化能力。

#### 注意事项

- 计算量较大，因为需要在多个时间窗上重复训练模型。

- 尽量在训练集的滚动和扩展上进行交叉验证，而非打乱数据顺序。

### 5. \*\*多步预测（Multi-Step Forecasting）与多测试集分割\*\*

在股票预测中，如果模型需要进行多步预测（例如同时预测未来 5 天的价格），可以使用\*\*多步预测窗口\*\*。即每次预测的目标不仅限于一天，而是未来数天的平均价格或收盘价。

#### 示例代码

假设预测未来 `n\_steps=5` 天的价格：

```python

n\_steps = 5

data['target'] = data['Close'].shift(-n\_steps) # 预测未来 n\_steps 天的收盘价

train\_data = data[:split\_idx].dropna() # 确保没有缺失值

test\_data = data[split\_idx:].dropna()

```

### 6. \*\*实用建议\*\*

- \*\*根据目标选择分割方法\*\*：如果只预测下一个交易日的价格，简单时间切割足够；如果预测长期趋势，滚动窗口或扩展窗口更合理。

- \*\*确保训练数据充足\*\*：通常需要大量历史数据，尤其是股票价格受市场趋势和季节性因素影响较大。

- \*\*使用较短的测试集进行验证\*\*：多次滚动窗口验证更能模拟出模型在真实市场环境中的表现。

### 7. \*\*最终推荐：结合滚动和扩展窗口分割\*\*

对于股票预测，特别是希望模型在未来数据上表现良好，结合滚动时间窗和扩展时间窗的方法较佳。可以每隔一段时间（例如一个月）扩展训练集的窗口，并在每个窗口上评估模型。

#### 示例代码（推荐方案）

```python

initial\_train\_size = 252 # 初始训练集大小，约为一年

test\_size = 21 # 每次测试期的长度，约为一个月

window\_step = 21 # 每次滚动的步长

start = 0

while start + initial\_train\_size + test\_size < len(data):

train\_data = data[start:start + initial\_train\_size]

test\_data = data[start + initial\_train\_size:start + initial\_train\_size + test\_size]

# 扩展窗口训练

initial\_train\_size += window\_step

start += window\_step

# 在当前分割上训练并评估模型

```

这种结合方法可最大化使用数据，模拟模型在未来测试集上的表现，并稳定模型在不同时间段的预测能力。