Rolling Windows for Train/Test

BUSI 722: Data-Driven Finance II

Kerry Back

Train on the Past, Test on the Future

Why Not a Random Split?

In a standard ML course, you randomly split data into training and test sets.

The Problem with Financial Data

If we randomly assign months to train and test, the training set will contain **future** data relative to some test observations. The model sees the future before predicting it.

- Financial data has a natural **time ordering** that must be respected.
- The model should never train on data that comes after the prediction date.
- This is the same look-ahead bias we avoided when constructing features.

Time-Series Split

Split the data at a fixed date: everything before is training, everything after is testing.

$$\underbrace{2011 \quad \cdots \quad 2019}_{\text{Training}} \;\Big|\; \underbrace{2020 \quad \cdots \quad 2025}_{\text{Testing}}$$

- Train the model on all stock-months through 2019.
- Generate predictions for 2020–2025.
- Evaluate: do the predictions correctly rank stocks?

Limitation: a single split gives only one evaluation. The result depends heavily on where you cut.

A Simple Implementation

Prompt: "Read the merged data. Standardize features cross-sectionally each month. Use months through 2019 as training and 2020 onward as test. Fit a LightGBM model and predict return ranks in the test set."

- This is a reasonable starting point but has drawbacks.
- The model is trained once and never updated.
- By 2025, the model is using weights learned from data ending in 2019 — six years stale.
- Markets change. Relationships between features and returns evolve. The model should be **retrained** as new data arrives.

Moving Windows

The Idea

Instead of training once, **retrain the model each month** (or each quarter) using data up to that point. Predict only the next period.

1. Train on months 1 through $t-1$.
2. Predict month t.
3. Record the predictions.
4. Move forward: train on months 1 through t, predict month $t+1$.
5. Repeat until the end of the sample.

This produces a time series of **out-of-sample predictions**, each made without any future information.

Expanding vs. Rolling Windows

- **Expanding window:** the training set always starts at the beginning and grows over time. More data = more stable estimates.
- **Rolling window:** the training set is a fixed-length window (e.g., 5 years) that slides forward. Adapts faster to changing markets.

Month	Expanding	Rolling (5 yr)
Predict 2020-01	Train 2011–2019	Train 2015–2019
Predict 2021-01	Train 2011–2020	Train 2016–2020
Predict 2022-01	Train 2011–2021	Train 2017–2021

Predicting One Period Ahead

Key principle: at each step, we predict **only the next month**.

- We are not forecasting 6 or 12 months ahead.
- Each month, we rank stocks based on the model's prediction for that month's return.
- We then observe the actual returns and record the results.
- This simulates what we would do in real time: retrain, predict, trade, repeat.

The collection of one-step-ahead predictions across all months forms the **backtest**.

Moving Window in Practice

Prompt: "Using an expanding window starting with 5 years of training data, retrain a LightGBM model each month and predict return ranks one month ahead. Collect all out-of-sample predictions."

```
months = sorted(df['month'].unique())
predictions = []
for t in range(60, len(months)):
    train = df[df['month'] < months[t]]
    test  = df[df['month'] == months[t]]
    model.fit(train[features], train[target])
    pred  = model.predict(test[features])
    predictions.append(...)
```

Spearman Rank Correlation

Measuring Prediction Quality

We predict return **ranks**, not dollar returns. So the question is: do our predicted ranks agree with the actual ranks?

- We don't need the predicted values to be accurate in magnitude.
- We need the **ordering** to be right: stocks we predict to be at the top should actually outperform.
- This calls for a **rank-based** metric.

Spearman Rank Correlation

The **Spearman correlation** measures the agreement between two rankings.

- Rank the predicted values 1 through n.
- Rank the actual returns 1 through n.
- Compute the Pearson correlation of the two rank vectors.

$$\rho_S = \mathrm{corr}\big(\mathrm{rank}(\hat{y}),\ \mathrm{rank}(y)\big)$$

- $\rho_S = 1$: perfect agreement in rankings.
- $\rho_S = 0$: predictions are no better than random.
- $\rho_S < 0$: predictions are inversely related to actual outcomes.

Why Spearman?

- Directly measures what we care about: can the model **rank stocks correctly**?
- Consistent with using ranked returns as the target.
- Insensitive to outliers — a stock that doubles or crashes doesn't distort the metric.
- Does not require forming portfolios or choosing portfolio weights.

We will use Spearman as the **cross-validation metric** for selecting hyperparameters: the hyperparameter setting that produces the highest Spearman on validation data is the one we choose.

Cross-Validation in the Moving Window

The Hyperparameter Problem

Every model has **hyperparameters** that must be set before training:

- **LightGBM:** number of trees, learning rate, max depth, min samples per leaf
- **Ridge:** regularization strength λ
- **Neural net:** number of layers, hidden units, learning rate, dropout rate
- **Random Fourier features:** number of features D, bandwidth γ

How do we choose them without using future data?

Cross-Validation Inside the Training Window

At each step of the moving window, **before** training the final model:

1. Split the training window into K time-ordered folds.
2. For each candidate hyperparameter setting:
   2.1 For each fold k: train on folds before k, predict fold k.
   2.2 Compute the Spearman correlation on each fold's predictions.
   2.3 Average across folds.
3. Select the hyperparameters with the highest average Spearman.
4. Retrain on the full training window with those hyperparameters.

Time-Series Cross-Validation

Standard K-fold CV assigns folds randomly. For time series, folds must respect the time order.

Example with 5 folds (training window = 2011–2019):

Fold	**Train**	**Validate**
1	2011–2014	2015
2	2011–2015	2016
3	2011–2016	2017
4	2011–2017	2018
5	2011–2018	2019

Each validation fold comes **after** its training fold — no look-ahead bias.

Spearman in Each Validation Fold

In each validation fold, we have predictions and actual returns for all stocks in that fold's months.

- Compute the Spearman correlation between predicted and actual return ranks **within each month** of the fold.
- Average the monthly Spearman correlations within the fold.
- Then average across all K folds.

This gives a single number summarizing how well a given hyperparameter setting ranks stocks on held-out future data — exactly what we need.

The Complete Loop

1. For each prediction month t:
   1.1 Define the training window (months 1 through $t-1$).
   1.2 Run time-series CV within the training window, selecting hyperparameters by Spearman.
   1.3 Train the model on the full training window with the best hyperparameters.
   1.4 Predict month t.
2. Collect all one-step-ahead predictions.

We now have out-of-sample predicted ranks for every stock in every month of the test period. In subsequent sessions, we will use these predictions to form and evaluate portfolios.

Computational Cost

This loop is expensive: at each step, we run K-fold CV (training the model K times per hyperparameter setting) and then train once more on the full window.

Practical shortcuts:

- Retrain quarterly instead of monthly (predict 3 months with the same model).
- Use a coarse hyperparameter grid.
- Fix hyperparameters after an initial CV and only retrain the model weights in subsequent months.
- Use fast models (LightGBM, ridge) for the full loop; reserve slow models (neural nets) for fewer retraining dates.