

LightGBM expected returns and daily shrunk VCV

Paul Geertsema, Various Agents

30 August 2026

TL;DR: We use conventional machinery: LightGBM for expected returns and Ledoit-Wolf style shrunk VCVs with Tikhonov regularisation. No GPUs, no neural networks. The non-standard bits involve 1) refinements to the way we use LightGBM and 2) not imposing a fully-invested constraint. Our submitted portfolio is the tangency portfolio minus the GMV portfolio, both formed under the Tikhonov VCV. Net equity exposure is zero at all times; gross (active) exposure is exactly 100%.

1 LightGBM refinements

1.1 Features and targets

We convert features to ranks (0 to 1) by month, which is fairly standard. But we also transform the target, which is not typical (yet). Specifically, we first standardize targets by month (subtract means and scale by standard deviation). This means that both the features and the targets now lack any systematic time-series component. This is intentional; most predictability seems to be in the cross-section of stocks, not in the time-series of the aggregate market. Trying to simultaneously also predict the market component just confuses the machines. We solve this problem by taking the market out of the data, for both features and the target. That leaves the target centered over time, but still skewed. Returns are limited to -100% but unlimited in principle (I think the highest daily historical return is of the order of 22x). In short, standardization does not in itself handle positive outliers. To take care of this, we apply an $\arctan()$ transform of the standardised target returns. $\arctan()$ is roughly an analogue of a log transform, but it also works with negative numbers. The overall effect is to remove the skew and squash outliers on both sides towards the centre.

This means that the output of the GBM is a “squashed” z-score: $y = (\arctan(z_score(r)))$, so y_hat inherits the same interpretation. One could, in principle, reverse the transform on y_hat to recover a prediction in the original return space. But we don’t do that; instead, we treat the GBM predictions as if they were returns. That has the effect of regularising predictions – the predicted values tend not to be too wild.

1.2 LightGBM settings

We change exactly four hyper-parameters in LightGBM; everything else is either left at default or relates to reproducibility or compute.

The overall idea is to give the model much more capacity than it needs, while at the same time making it hard for it to overfit. To generate that capacity, we increase the number of trees from the default of 100 to 1,000 (10x) while at the same time dropping the learning rate from its

default of 0.1 to 0.01 (a tenth of the default). Net effect is neutral in terms of adding to the GBM base (GBMs start by predicting the training data target mean == a tree with one leaf), but roughly 10x the compute. We also increase number of leaves from the default of $32-1=31$ to $256-1=255$. Most applications in finance use depths of 2-4, which maximally imply $2^2 = 4$ to $2^4 = 16$ leaves. LightGBM grows leaf-wise, so it is better to deal with the leaf count than the depth. Either way, 255 leaves are a LOT – arguably way too much. So what we end up with is a large number of trees (1,000), each very detailed (leaves = 255). What this creates is *capacity*; the model can handle very complicated DGPs. But it might overfit.

We address the potential overfitting in an unusual way. First, we rely on the target transforms above to limit the influence of non-typical observations in general. Second, we increase the minimum number of observations in each leaf from the default of 20 to 2,000 (an increase of 100x). LightGBM defaults are calibrated to typical use cases where the signal to noise ratio is much higher than is the case for stock returns. So 20 is a reasonable choice where a very high or very low target really could have been predicted. But in finance, almost nothing should ever have an ex-ante expected monthly return of 10%. But this is very common in realised returns. By imposing a minimum of 2,000 firm-months in each leaf, we make it very hard for the machine to fit to genuinely non-typical patterns: whatever it predicts in a given leaf is an average over at least 2,000 firm-months, not a handful of extremes. Finally, the GBM combines the 1,000 trees, so even if the trees still overfit, the GBM “diversifies” it away to the extent that the overfitting is idiosyncratic.

```
LGBM_PARAMS = dict(
    n_estimators=1000,          # Default = 100, so 10 times larger
    learning_rate=0.01,        # Default = 0.1, so a tenth of the default
    num_leaves=255,            # Default = 31, so 8 times larger: combined -> MUCH more compute
    min_child_samples=2000,     # Default = 20, so 100x larger. The magic sauce that controls overfitting.
    random_state=SEED,         # The rest is standard repro and perf settings
    n_jobs=N_THREADS,
    deterministic=True,
    force_row_wise=True,
    verbose=-1,
)
```

1.3 Expanding window

We train the LightGBM on an expanding window, advancing 12 months at a time. Advancing monthly is of course possible, but takes 12x longer. The DGP does not really change that much over the course of a single year. Once trained, the model is applied to month t 's characteristics to forecast the excess return over $t + 1$.

2 VCV estimation

We use daily data for the VCV estimate, scaled to monthly units by $\sqrt{21}$ (that scale does not change the weights). Daily data is noisy, but there is a lot of it. We take the extra observations and then deal with the noise. Missing days are set to zero, so a non-trading day is treated as no return. Each stock's column is then time-series demeaned, so the columns have mean zero and $S = X'X/T$ is a covariance. A stock needs 250 daily observations in the window to be held, and a month needs 50 such stocks.

The empirical VCV is first shrunk using Ledoit-Wolf, with the shrinkage intensity δ clipped to $[0, 1]$. We then add λI , where λ is the average stock variance. That doubles the average variance and leaves the covariances unchanged. The resulting $A = \Sigma + \lambda I$ is the Tikhonov (ridge)

covariance: it is the matrix we invert to form the tangency portfolio and the GMV portfolio, so that step is ridge mean-variance rather than classic Markowitz on Σ .

The VCV is estimated every month on a trailing window of 756 trading days (about three years), so the estimate can move with regime shifts (expansions versus contractions) and with structural change (computers, the internet, AI).

3 The portfolio

We take the GBM output as expected returns (even though they are not really in return space; see the comment above) and use A as the VCV in a Markowitz tangency portfolio (Tikhonov: A in place of Σ). Call that the tangency portfolio w_{tan} . The GMV portfolio w_{gmV} is formed with the same A . Both sum to one, so their difference sums to zero. Scale the difference so the absolute weights (the exposures) sum to one; that is w , the submitted portfolio.

4 Agents

This submission was coded and written up over about three days using a mix of Claude Opus 5, Grok 4.6 and GPT 5.6 Sol in Cursor. The agents did most of the coding and the math; I mostly spent my time explaining what I wanted, sanity-checking results, arguing with the agents (we both got things wrong from time to time) and asking for help or advice. So a team effort, all in all.

5 Details

What follows is the algebra behind the preceding sections. A portfolio here is a vector of weights that sums to one, except the submitted portfolio (sums to zero). The vectors a and b below are intermediates, not portfolios.

5.1 Forecast

Let i index stocks and t the position month. The field `ret_exc_lead1m` on the row for month t is the excess return already realised over $t + 1$, written $r_{i,t+1}$. The 402 characteristics are replaced, within each month, by average-tie percentile ranks on $(0, 1]$. Missing values stay missing.

The training target is

$$y_{it} = \arctan(z_{it}), \quad z_{it} = \frac{r_{i,t+1} - \bar{r}_t}{s_t}$$

where $\bar{r}_t$ and s_t are the cross-sectional mean and standard deviation of the lead return among stocks present at t . At month t the booster output is used as μ_t in that unit: we do not apply tan. The tree for month t is fit on rows strictly before t , and is refit every twelve months on the expanding window.

5.2 Covariance

Let X be the trailing 756 trading days of daily excess returns ending on the position date, restricted to that month's stocks, each column multiplied by $\sqrt{21}$ so that S is in monthly units. The scaling is cosmetic: every output is invariant to it. Missing entries are set to zero, then each

column is demeaned. A stock needs 250 observed days in the window to be held, and a month needs 50 such stocks. Filling unobserved days with zero and dividing by the full window T understates the variance of stocks with thin history, which tilts the inverse toward those stocks. The sample is $S = X'X/T$. Ledoit–Wolf shrinks toward a scaled identity,

$$\Sigma = (1 - \delta) S + \delta \nu I, \quad \nu = \text{tr}(S)/n$$

with

$$\delta = \frac{\hat{\beta}^2}{\hat{d}^2}, \quad \hat{d}^2 = \|S - \nu I\|_F^2, \quad \hat{\beta}^2 = \frac{1}{T^2} \sum_{\tau} \|x_{\tau} x'_{\tau} - S\|_F^2$$

then δ clipped to $[0, 1]$. The Tikhonov matrix is

$$\lambda = \frac{1}{n} \text{tr}(\Sigma), \quad A = \Sigma + \lambda I$$

so λ is the average stock variance.

5.3 The portfolio

One linear solve in A produces two vectors that are not yet portfolios:

$$\begin{aligned} a &= A^{-1} \mu \\ b &= A^{-1} \mathbf{1} \end{aligned}$$

The GMV portfolio is b scaled to sum to one,

$$w_{\text{gmv}} = \frac{b}{\mathbf{1}' b}$$

The tangency portfolio w_{tan} sums to one, so the tangency portfolio minus the GMV portfolio sums to zero. Scale that difference to unit gross:

$$w = \frac{a - (\mathbf{1}' a) w_{\text{gmv}}}{\|a - (\mathbf{1}' a) w_{\text{gmv}}\|_1}$$

Every quantity at t uses data at or before t only: the covariance window ends on the position date.