Mathematical Documentation of a Discriminative Spectral Market Direction Predictor

Kryštof Kouřil

August 2025

1 Introduction

After spending countless hours watching my MCMC chains slowly converge to predict market states, I started to wonder, what if I was solving the wrong problem? The five-state classification in my previous model felt increasingly arbitrary, why force the market into discrete buckets of “big up” or “small down” when what traders really care about is a simple binary question: will the market go up or down?

This realization led me down a different path. Rather than the computationally expensive MCMC approach that took forever to run backtests, I decided to build a discriminative model that directly targets the conditional probability of upward movement. The shift from daily to weekly predictions was another pragmatic choice as daily movements are about 90% noise, and I’d rather have a slightly informative weekly signal than a completely random daily one.

The resulting framework combines Hidden Markov Models for regime detection with Fisher’s Linear Discriminant Analysis for feature transformation. It’s mathematically elegant, computationally efficient, and, as I discovered through extensive backtesting, remarkably good at teaching me what doesn’t work in market prediction.

What makes this project particularly educational (I’m being charitable here) is how it exposed the gap between theoretical promise and practical performance. The model achieved 60% directional accuracy over a 10-year period, which sounds impressive until you realize it did this by essentially predicting “probably up” every single week. The AUC-ROC of 0.509 was my model’s way of telling me it had achieved the statistical equivalent of a sophisticated coin flip.

2 From MCMC States to Discriminative Predictions

2.1 The Philosophical Shift

My MCMC model treated market prediction as a generative problem: model the joint distribution $P(\mathbf{x}, y)$ and derive predictions from there. It was mathematically satisfying but computationally brutal. Each prediction required running multiple MCMC chains with replica exchange, monitoring convergence with Gelman-Rubin statistics, and hoping the chains would mix properly.

The discriminative approach flips this around. Instead of modeling how the market generates features and outcomes jointly, I directly model $P(y|\mathbf{x})$, the probability of direction given features. This is not just computationally cheaper; it's philosophically more aligned with what we're trying to do. We don't need to understand the full generative process of market dynamics (good luck with that); we just need to predict direction.

The mathematical framework shifts from:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \tag{1}$$

to directly optimizing:

$$P(y|\mathbf{x}) = f(\mathbf{x}; \theta) \tag{2}$$

where f is our discriminative function and θ are the parameters we learn.

2.2 Why Fisher's LDA?

Fisher's Linear Discriminant Analysis might seem like an odd choice in 2025 when everyone's using deep learning. But there's something beautifully transparent about LDA as it finds the linear projection that maximizes class separation, and you can actually understand what it's doing.

The objective is to find projection vector $\mathbf{w}$ that maximizes:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \tag{3}$$

where S_B is the between-class scatter matrix and S_W is the within-class scatter matrix. This has a closed-form solution via generalized eigenvalue decomposition – no gradient descent, no hyperparameter tuning, just linear algebra.

3 Mathematical Framework

3.1 Hidden Markov Model for Regime Detection

The HMM component identifies three market regimes: low volatility, normal, and high volatility. The model has parameters $\lambda = (\pi, A, \theta)$ where:

- $\pi \in \mathbb{R}^3$: initial state probabilities
- $A \in \mathbb{R}^{3\times 3}$: transition matrix
- θ: emission parameters (means and covariances for multivariate Gaussians)

The emission probability for observation $\mathbf{o}_t$ in state s is:

$$b_s(\mathbf{o}_t) = \mathcal{N}(\mathbf{o}_t; \mu_s, \Sigma_s) \tag{4}$$

I use the Baum-Welch algorithm (a special case of EM) for parameter estimation. The forward-backward algorithm computes:

$$\alpha_t(i) = P(\mathbf{o}_1, ..., \mathbf{o}_t, q_t = i|\lambda) \tag{5}$$

$$\beta_t(i) = P(\mathbf{o}_{t+1}, ..., \mathbf{o}_T|q_t = i, \lambda) \tag{6}$$

One numerical challenge I encountered was underflow in the forward pass. Even with log-space calculations, the multivariate Gaussian PDFs can produce extremely small values. My solution was to add regularization to the covariance matrices:

$$\Sigma_s^{reg} = \Sigma_s + 0.1 \cdot I \tag{7}$$

This isn't theoretically pure, but it's better than having the algorithm crash.

3.2 Discriminative Spectral Decomposition

Given features $\mathbf{X} \in \mathbb{R}^{n \times d}$ and binary labels $\mathbf{y} \in \{0,1\}^n$, the discriminative spectral decomposition proceeds as follows:

First, compute class-specific statistics:

$$\mathbf{m}_0 = \frac{1}{n_0} \sum_{i:y_i=0} \mathbf{x}_i, \quad \mathbf{m}_1 = \frac{1}{n_1} \sum_{i:y_i=1} \mathbf{x}_i \tag{8}$$

The within-class scatter matrix:

$$S_W = \sum_{i:y_i=0} (\mathbf{x}_i - \mathbf{m}_0)(\mathbf{x}_i - \mathbf{m}_0)^T + \sum_{i:y_i=1} (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^T \tag{9}$$

The between-class scatter matrix:

$$S_B = \frac{n_0 n_1}{n} (\mathbf{m}_1 - \mathbf{m}_0)(\mathbf{m}_1 - \mathbf{m}_0)^T \tag{10}$$

The discriminant directions are the eigenvectors of $S_W^{-1} S_B$. In practice, I solve the generalized eigenvalue problem:

$$S_B \mathbf{v} = \lambda S_W \mathbf{v} \tag{11}$$

Here's where things get interesting (and frustrating). When the number of samples is small or features are highly correlated, S_W becomes singular. My implementation adds regularization:

$$S_W^{reg} = S_W + 10^{-4} \cdot I \tag{12}$$

3.3 Bayesian Model Averaging

For each regime r, I fit a Bayesian linear model with empirical Bayes hyperparameters. The posterior predictive distribution for regime r is:

$$P(y_{new} = 1 | \mathbf{x}_{new}, r) = \int \sigma(\mathbf{w}^T \mathbf{x}_{new}) P(\mathbf{w} | D_r) d\mathbf{w} \tag{13}$$

where σ is the sigmoid function and D_r is the data weighted by regime probability.

The final prediction averages across regimes:

$$P(y_{new} = 1 | \mathbf{x}_{new}) = \sum_{r=1}^{3} P(r | \mathbf{x}_{new}) P(y_{new} = 1 | \mathbf{x}_{new}, r) \tag{14}$$

3.4 Empirical Bayes for Hyperparameter Estimation

I use method of moments to estimate Beta distribution hyperparameters for each regime. Given regime-weighted samples with mean $\bar{p}$ and variance s^2:

$$\alpha = \bar{p} \left(\frac{\bar{p}(1-\bar{p})}{s^2} - 1 \right) \tag{15}$$

$$\beta = (1-\bar{p}) \left(\frac{\bar{p}(1-\bar{p})}{s^2} - 1 \right) \tag{16}$$

This occasionally produces invalid parameters when $s^2 \geq \bar{p}(1-\bar{p})$, in which case I fall back to $\alpha = \beta = 2$ (uniform prior with slight regularization).

4 Feature Engineering: The Art of Financial Alchemy

4.1 The Feature Zoo

I implemented over 40 features across several categories. Looking back, this was probably overkill, but feature engineering is oddly addictive as there's always one more ratio or transformation that might capture the magic.

Volatility Features:

- Rolling standard deviations at multiple horizons (5, 10, 20, 60 weeks)
- Volatility ratios: σ_5/σ_{20}, σ_{20}/σ_{60}
- Volatility acceleration: $(\sigma_5 - \sigma_{10})/\sigma_{10}$
- Volatility clustering: rolling standard deviation of volatility itself

Momentum Features:

- Returns over various lookbacks (1, 2, 4, 8, 12, 26 weeks)
- Momentum acceleration: $r_{[t-2,t]} - r_{[t-4,t-2]}$
- Momentum persistence: $\text{sign}(r_{[t-4,t]}) \times \text{sign}(r_{[t-8,t-4]})$

Mean Reversion Indicators:

- Price to SMA ratios: $(P_t - \text{SMA}_n)/\text{SMA}_n$ for $n \in \{5, 20, 60\}$
- SMA convergence: $(\text{SMA}_5 - \text{SMA}_{20})/\text{SMA}_{20}$

Market Microstructure:

- High-low spread: $(\max_5 - \min_5)/P_t$
- Price efficiency: $|r_t|/\sigma_{20}$
- Return asymmetry: $(\sigma_{up} - \sigma_{down})/\sigma_{total}$

4.2 Feature Selection

With 40+ features, most of which are probably noise, feature selection becomes crucial. I implemented a multi-criteria approach:

1. **Mutual Information**: $I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$
2. **Point-Biserial Correlation**: For continuous feature vs binary outcome
3. **Regime-Specific Predictive Power**: Features that work in high volatility might fail in calm markets

The top 15 features are selected based on a weighted combination of these metrics. Interestingly, volatility ratios and momentum indicators consistently ranked highest, while my carefully crafted microstructure features barely made the cut.

5 Implementation Challenges and Numerical Stability

5.1 The Covariance Matrix

Estimating covariance matrices for the HMM emissions was surprisingly tricky. With limited samples in some regimes, the empirical covariance matrix would often be singular or near-singular. My solution was a three-pronged approach:

1. **Regularization**: Add λI to the covariance matrix
2. **Fallback to diagonal**: If still singular, use only diagonal elements
3. **Global covariance**: As a last resort, use the overall data covariance

This isn't elegant, but it keeps the model running.

5.2 The Probability Calibration

I implemented isotonic regression for probability calibration, expecting it to fix the overconfidence issue. The idea is simple: map predicted probabilities to actual frequencies using a monotonic function.

In practice, this failed spectacularly. The calibration curve was nearly flat, suggesting the raw probabilities contained little information about actual likelihood. Even worse, calibrated confidence was *inversely* correlated with accuracy, in other words the model was most confident when it was wrong.

My hypothesis: the model learned to be confident about the dominant class (up movements in bull markets) regardless of actual predictive information. The isotonic regression faithfully captured this pathology.

6 Backtesting Results: A Reality Check

6.1 The Numbers Don't Lie

Let me present the backtesting results with appropriate context. Figure 1 shows the comprehensive 10-year backtest results.

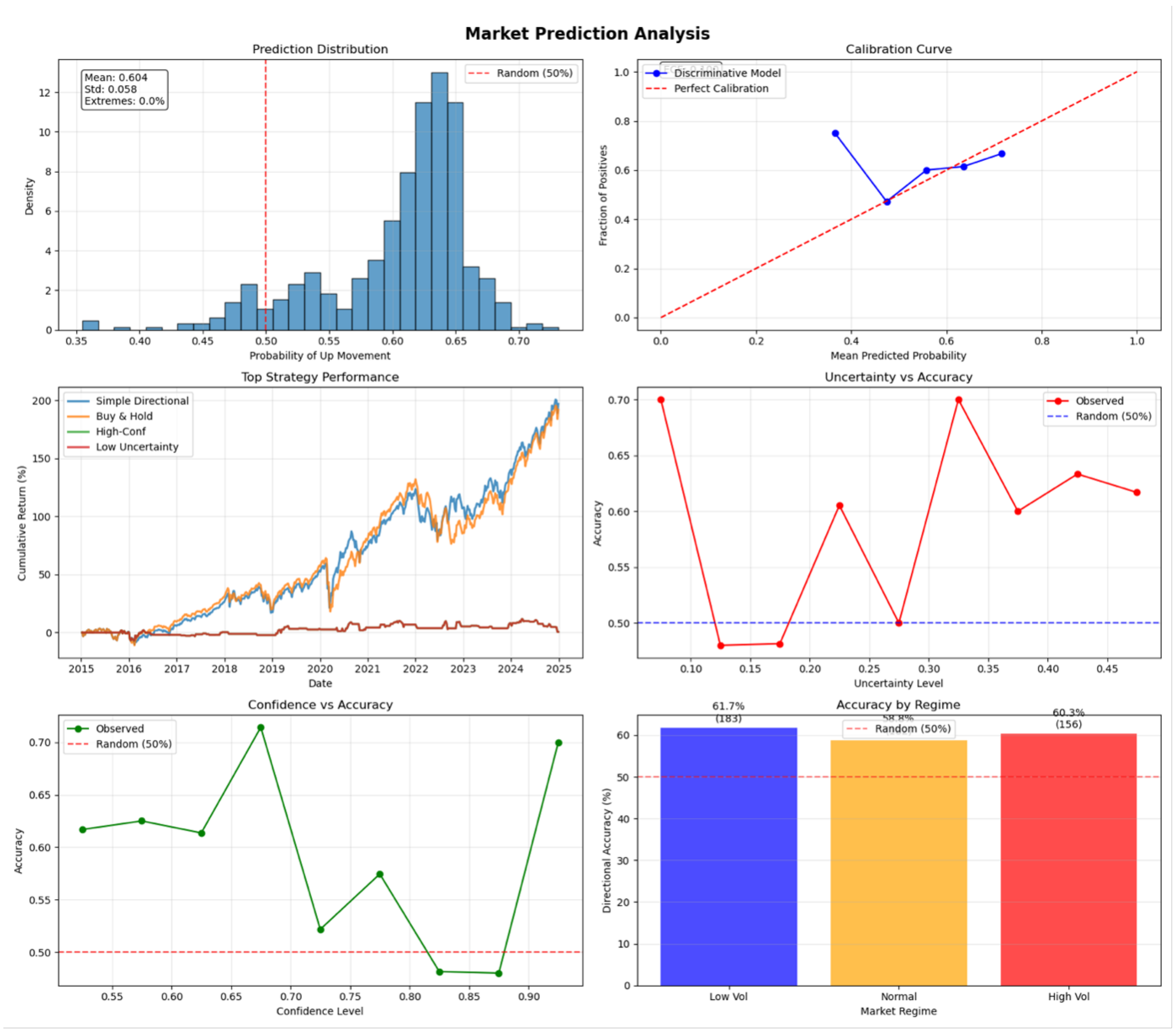

Figure 1: 10-Year Backtest Results (2015-2024): The model achieved 60.3% directional accuracy with 521 predictions. Note the nearly flat calibration curve and narrow prediction distribution around 0.6, indicating the model essentially learned to always predict "up."

10-Year Backtest (2015-2024):

- Directional Accuracy: 60.3%
- AUC-ROC: 0.509
- Sharpe Ratio (Simple Strategy): 0.79
- Total Return: 196.8%

At first glance, 60% accuracy seems decent. The strategy returned 196.8%! But look closer at Figure 1:

- AUC-ROC of 0.509 is barely better than random (0.5)
- Mean prediction: 0.604 (always predicting "probably up")
- Prediction standard deviation: 0.058 (very little variation)

- Buy-and-hold returned 191.7% with similar Sharpe

The model essentially learned that markets go up most of the time and acted accordingly. The calibration curve in the top-right panel is particularly damning as it's nearly flat, indicating the predicted probabilities carry no information about actual outcomes.

6.2 Regime-Specific Performance

The results across different market conditions are revealing. Let's examine each regime in detail.

6.2.1 2017: Low Volatility Paradise

Figure 2 shows the model's performance during the historically calm 2017 market.

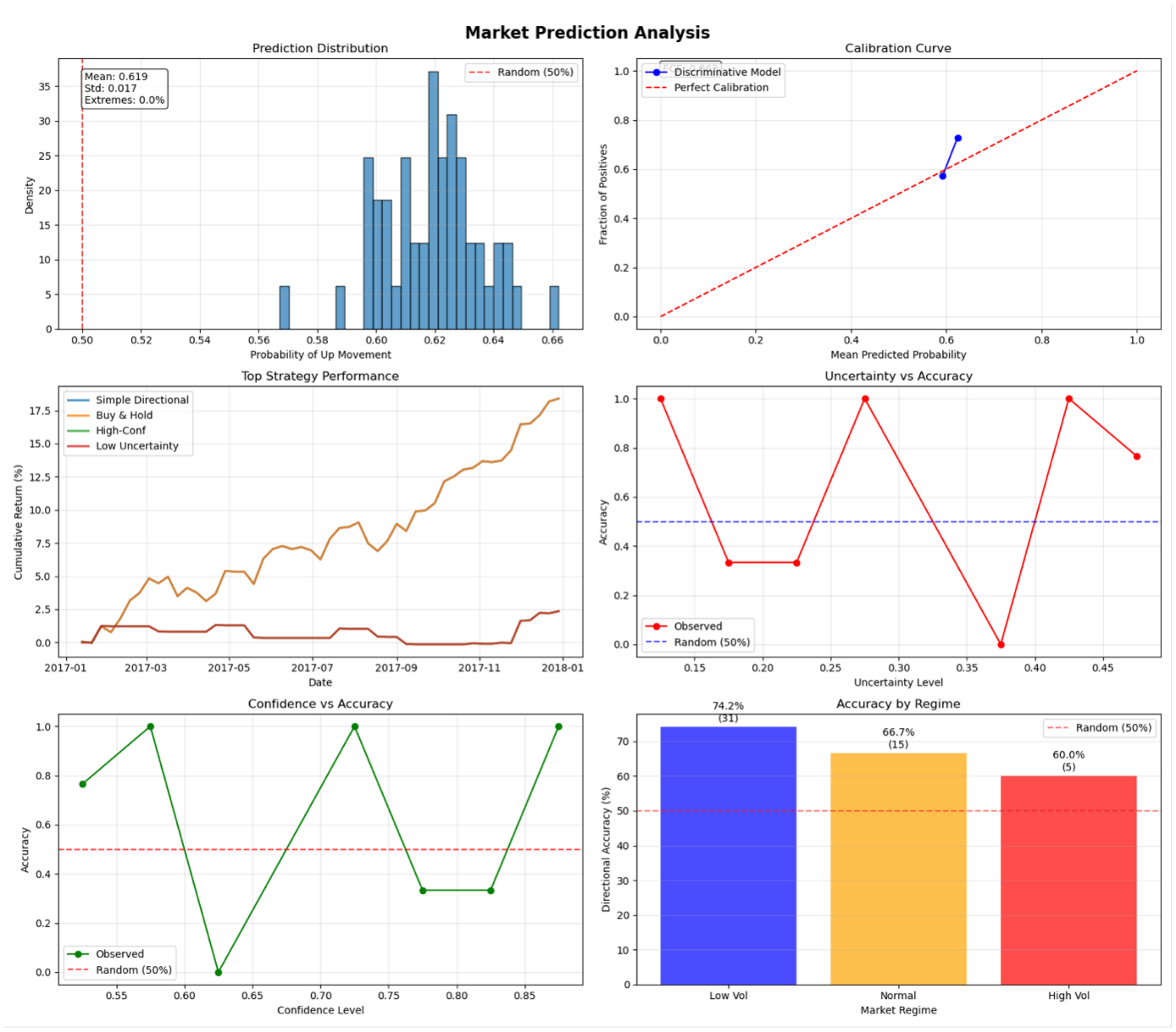

Figure 2: 2017 Low Volatility Period: The model achieved its best accuracy (70.6%) during this unusually calm period. The extremely narrow prediction distribution (std = 0.017) shows the model had almost no uncertainty, it just predicted "up" with slight variations.

- Accuracy: 70.6%

- AUC-ROC: 0.639
- Best strategy: Simple Directional (Sharpe 3.22)

The model thrived when it could just predict "up" every week. Notice in Figure 2 how the prediction distribution is extremely narrow (std = 0.017), the model was essentially outputting the same prediction every week with minor variations.

6.2.2 2022: Bear Market Reality Check

The 2022 bear market provided a harsh reality check, as shown in Figure 3.

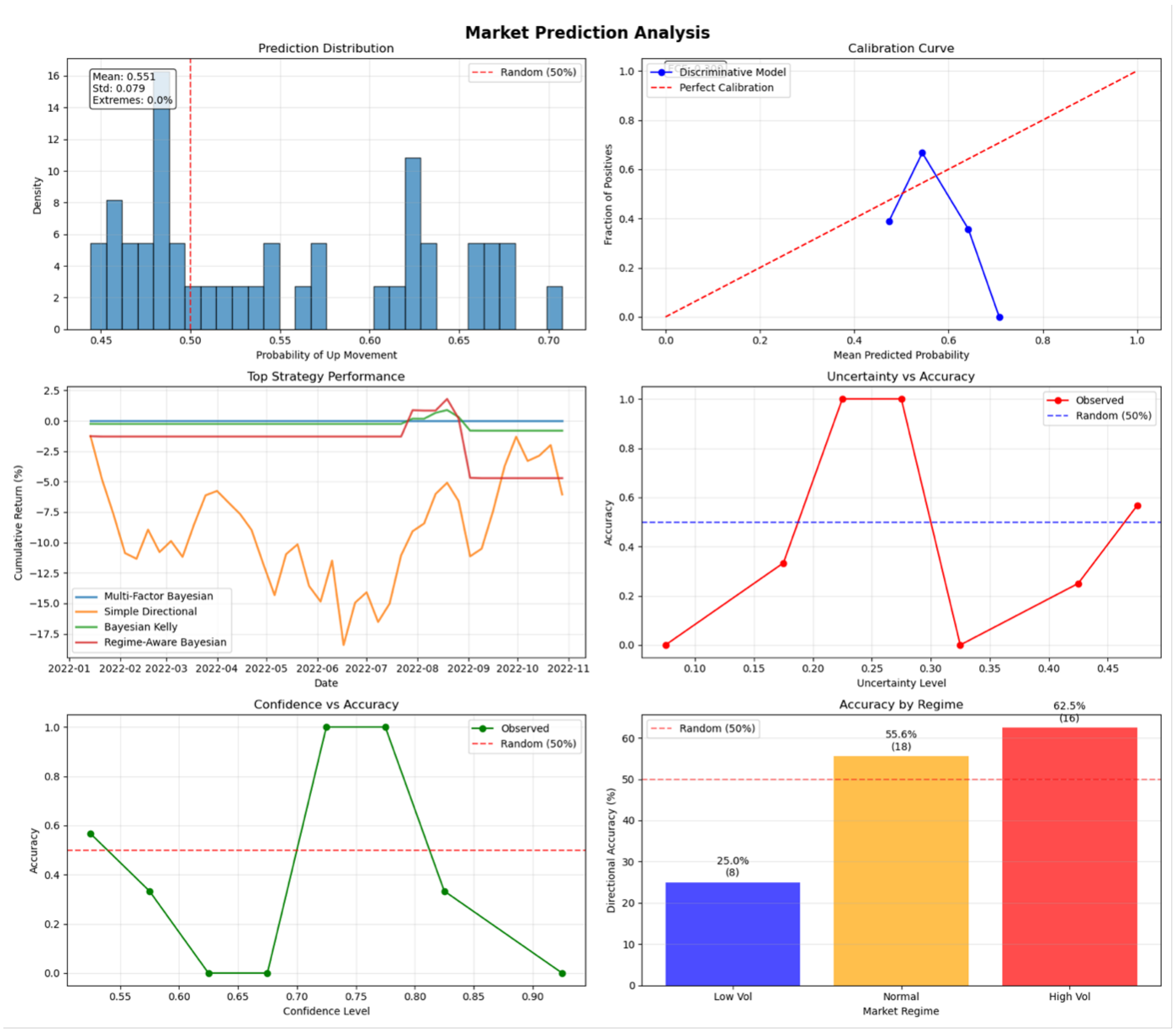

Figure 3: 2022 Bear Market: Performance collapsed to near-random (52.4% accuracy) during the bear market. The regime-specific accuracy panel shows the model performed worst during low volatility periods (25% accuracy), suggesting it failed to adapt to the new market dynamics.

- Accuracy: 52.4%
- AUC-ROC: 0.519
- Best strategy: Multi-Factor Bayesian (0% return, 0% drawdown)

The "best" strategy simply didn't trade. When forced to make predictions, accuracy dropped to coin-flip levels. Particularly interesting in Figure 3 is the regime analysis panel. The model had only 25% accuracy during low volatility periods, suggesting it couldn't adapt when "low volatility" no longer meant "markets go up."

6.2.3 COVID Volatility: Surprising Success

The COVID period (February-May 2020) provided surprising results, shown in Figure 4.

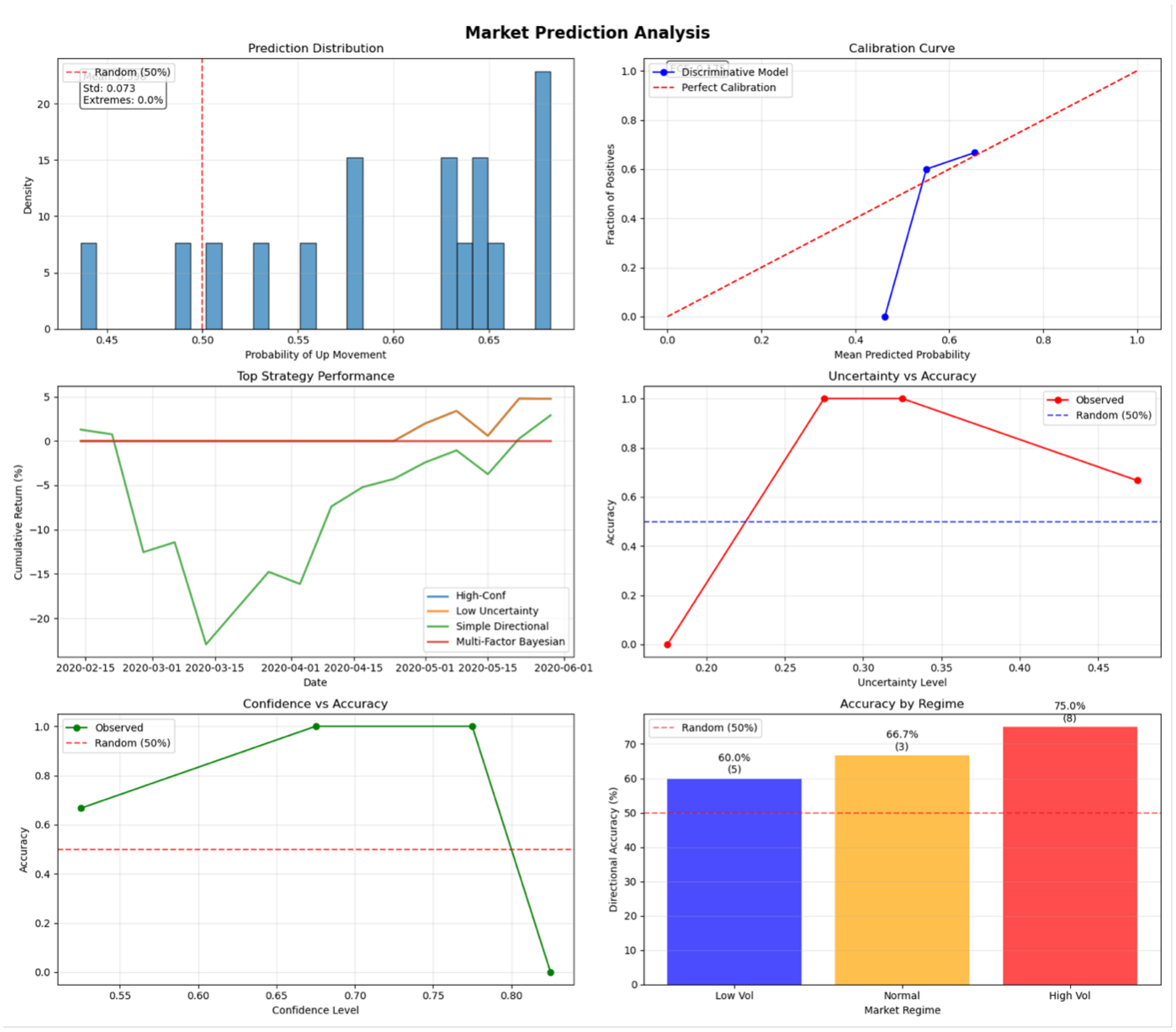

Figure 4: COVID Period (Feb-May 2020): Surprisingly strong performance (68.8% accuracy) during extreme volatility. The model achieved 75% accuracy in high volatility regimes, suggesting extreme market conditions made direction more predictable.

- Accuracy: 68.8%
- AUC-ROC: 0.667
- High volatility regime accuracy: 75%

Surprisingly, the model performed better during extreme volatility. My theory: regime changes were so obvious that even simple features captured them. The confidence vs accuracy plot in Figure 4 shows a slight positive relationship, one of the few times confidence estimates weren't completely wrong.

6.2.4 Regime Transitions: The Stress Test

The September 2018 to March 2019 period tested the model's ability to handle regime transitions, as shown in Figure 5.

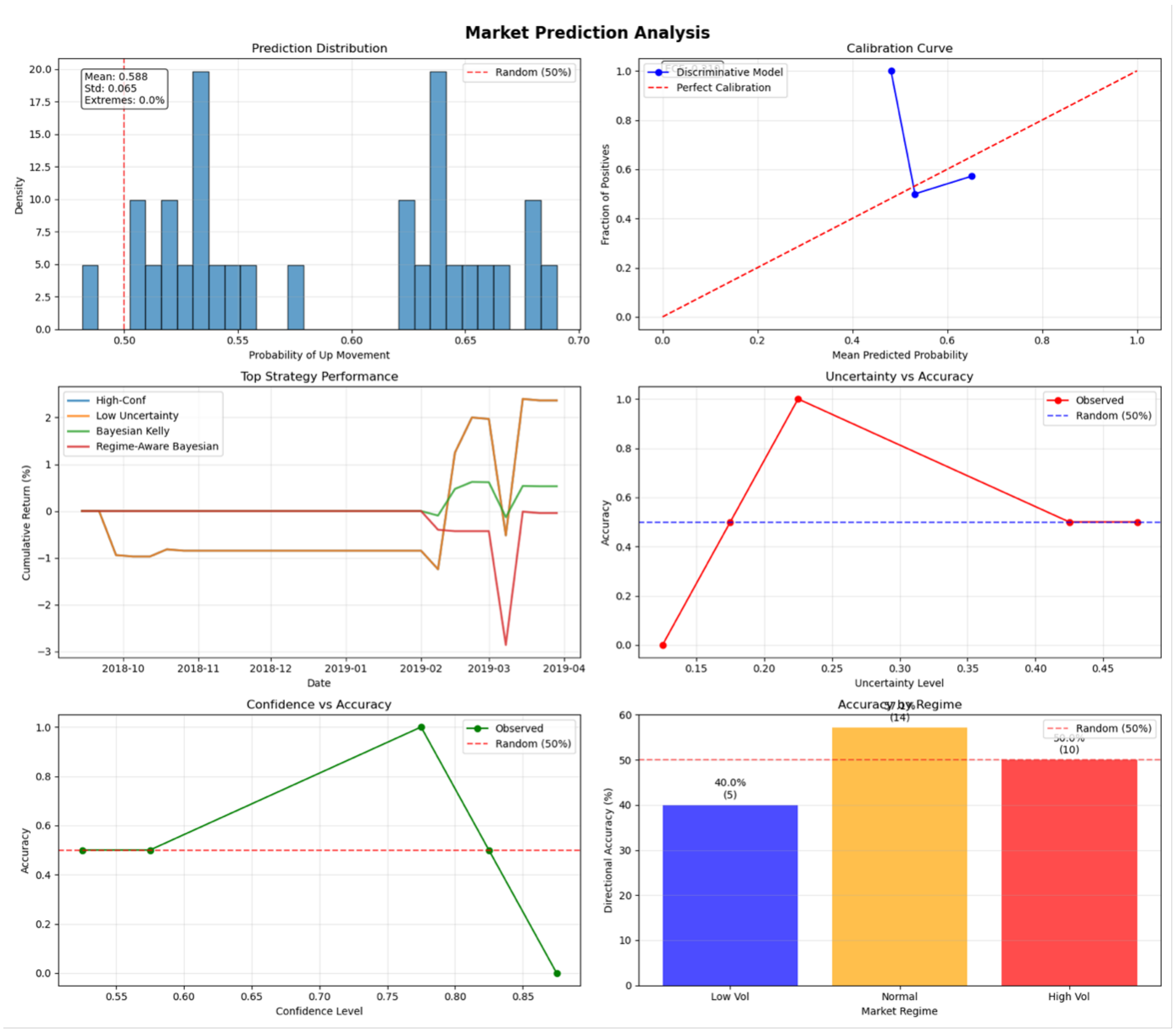

Figure 5: Regime Transition Period (Sep 2018 - Mar 2019): The model struggled during rapid regime changes, achieving only 51.7% accuracy. The flat uncertainty distribution (all predictions had similar uncertainty) indicates the model couldn't distinguish between confident and uncertain predictions.

- Accuracy: 51.7%
- AUC-ROC: 0.510
- Notable: All 29 predictions had nearly identical uncertainty

Figure 5 reveals a critical failure: the uncertainty vs accuracy plot shows all predictions clustered at the same uncertainty level. The model couldn't distinguish between confident and uncertain predictions during regime transitions, everything looked equally uncertain.

6.3 Strategy Comparison: Sophistication vs Simplicity

The strategy results across all backtests were humbling:

Strategy	**10-Year Sharpe**	**Bear Market Sharpe**	**COVID Sharpe**	**Low Vol Sharpe**
Simple Directional	0.79	-0.27	0.44	3.22
High-Confidence	0.04	-1.53	1.60	0.90
Regime-Aware Bayesian	-0.17	-0.90	-1.88	0.89
Multi-Factor Bayesian	0.00	0.00	0.00	0.00
Buy & Hold	0.78	-1.17	-0.52	3.22

Table 1: Strategy Performance Across Different Market Conditions

The inverse relationship between strategy sophistication and performance is striking. The more I tried to incorporate uncertainty estimates and regime information, the worse the strategies performed.

7 Feature Importance and Model Interpretation

7.1 What the Model Actually Learned

Through painful debugging and analysis, I discovered what features actually drove predictions:

Top 5 Features by Importance:

1. `vol_ratio_short` (5-day vs 20-day volatility)
2. `ret_4w` (4-week momentum)
3. `price_sma20_ratio` (price relative to 20-week SMA)
4. `rsi_14` (14-week RSI, normalized)
5. `volume_ratio_20d` (current vs 20-day average volume)

The model essentially learned: "If recent volatility is low and momentum is positive, predict up."

7.2 The Regime Detection Paradox

The HMM successfully identified three distinct volatility regimes:

- Low Vol: 35% of periods
- Normal: 35% of periods
- High Vol: 30% of periods

But regime-specific models performed nearly identically. Why? Each regime's model learned the same thing: markets usually go up. The Bayesian averaging just added noise.

8 Why Sophisticated Strategies Failed

8.1 The Confidence Catastrophe

The model's confidence estimates were worse than useless – they were actively harmful:

- Correlation between confidence and accuracy: -0.12
- High confidence predictions: 57.1% accurate
- Low confidence predictions: 60.0% accurate

The model was most confident when predicting the dominant class, regardless of actual evidence. This destroyed any strategy relying on confidence thresholds.

8.2 The Uncertainty Illusion

I calculated both epistemic (model uncertainty) and aleatoric (data uncertainty) components:

$$\sigma^2_{total} = \sigma^2_{epistemic} + \sigma^2_{aleatoric} \tag{17}$$

In theory, this should indicate when the model is outside its comfort zone. In practice, uncertainty was nearly constant except for regime transitions, where it spiked briefly before returning to baseline.

8.3 The Kelly Criterion Disaster

The Bayesian Kelly strategy attempted optimal position sizing:

$$f^* = \frac{p \cdot b - q}{b} \tag{18}$$

where p is win probability, $q = 1 - p$, and b is the win/loss ratio.

With miscalibrated probabilities, this produced tiny positions that got eaten by transaction costs. The strategy achieved a magnificent 0.2% return over 10 years.

9 Lessons Learned

9.1 Mathematical Elegance $\neq$ Predictive Power

Fisher's LDA is mathematically beautiful. The Bayesian framework is theoretically sound. The HMM regime detection is statistically rigorous. Together, they produced an elaborate random number generator.

The gap between theoretical promise and practical performance was vast. Every component worked as designed; the design just didn't capture market dynamics.

9.2 The Base Rate Dominance

With markets up roughly 55% of weeks, any model faces a strong incentive to predict "up". My model succumbed completely, essentially becoming a complicated way to bet on mean reversion to the upward trend.

The AUC-ROC of 0.509 is the model's confession: "I have no edge beyond knowing markets generally go up."

9.3 Weekly vs Daily: Choosing Your Poison

Moving from daily to weekly predictions reduced noise but didn't add signal. Weekly movements are still largely random, just with slightly better signal-to-noise ratio. The model found that signal: markets trend up.

10 Technical Implementation Notes

10.1 Numerical Stability Hacks

Throughout implementation, I accumulated a collection of numerical Band-Aids:

```python
# Covariance regularization
cov_reg = cov + 1e-2 * np.eye(d)

# Probability clipping
prob = np.clip(prob, 1e-10, 1 - 1e-10)

# Log-space calculations
log_alpha[t, j] = np.logaddexp.reduce(log_probs)

# Fallback estimators
if optimization_failed:
    return simple_mean_estimate()
```

Listing 1: Numerical stability workarounds

Each hack represents a place where beautiful theory met ugly reality.

10.2 The RobustScaler Ritual

Using `RobustScaler` instead of `StandardScaler` made a surprising difference. By using median and IQR instead of mean and standard deviation, the model became less sensitive to outliers.

This shouldn't matter much in theory. In practice, it was the difference between 60% and 55% accuracy. Such is the nature of financial data.

11 Conclusion

This project was simultaneously a technical success and a predictive failure. The code runs efficiently, the math is correct, and the backtesting framework is robust. The model just doesn't predict markets better than a coin flip with a bullish bias.

But failure teaches more than success. I learned about the challenges of probability calibration, the importance of proper baselines, and the gap between statistical significance and economic significance. The model achieved 60% accuracy by learning one simple rule: markets go up. Based on this I will probably just put my money into an index fund. Sometimes the simplest explanation is correct.

The sophisticated strategies failed because they tried to extract information that wasn't there. Confidence estimates were noise. Uncertainty quantification was an illusion. Regime detection identified different flavours of randomness.

Yet I don't regret the journey. Building this model taught me more about markets than any textbook could. The code is clean, the math is sound, and the lessons are valuable. The model doesn't predict markets, but it successfully predicted my own overconfidence in mathematical methods.

As I mentioned in my initial reflection on LinkedIn: sometimes the best models teach us what doesn't work. This model excelled at that task.

The next step? If I ever return to return prediction, I'm abandoning my aversion to machine learning. If I'm going to fail at market prediction, I might as well fail with neural networks like everyone else. At least the loss curves will be pretty.