Week 4: Time-Series Forecasting

ARIMA & GARCH Models for Financial Markets

MSc Banking and Finance - FinTech Course

September 30, 2025

Abstract

This document covers Week 4's time-series forecasting methods for cryptocurrency markets. We explain stationarity testing, ARIMA models for predicting returns, GARCH models for forecasting volatility, and Monte Carlo simulation for price predictions. The focus is on intuitive understanding with practical applications to DeFi portfolio management.

Contents

1 Introduction and Setup

1.1 Cell 1: Setting Up the Environment

What We're Doing

We're building a forecasting system that can predict cryptocurrency prices, for example, and their volatility. Think of it like weather forecasting: we use historical patterns to predict future conditions, knowing that predictions become less certain the further ahead we look.

The key tools we need:

- **statsmodels**: For ARIMA models (predicting average returns)
- **arch**: For GARCH models (predicting how volatile/risky prices will be)
- **numpy/pandas**: For handling and analyzing data
- **matplotlib**: For visualizing results

Why This Matters

In traditional markets, you might predict next week's stock price. In crypto, volatility changes dramatically—Bitcoin can be calm one week and swing wildly the next. We need specialized tools to handle this.

Key Takeaways

- Time-series forecasting is different from regular prediction because order matters
- We need both return forecasts (direction) and volatility forecasts (risk)
- Setting a random seed ensures our results are reproducible for research

2 Data Preparation

2.1 Cell 2: Loading Returns and Creating Time Index

What We're Doing

We're converting prices into returns and organizing them with proper dates. This is like converting temperatures into "change in temperature"—it's the change that matters for forecasting.

Simple concept:

$$\text{Return} = \ln\left(\frac{\text{Today's Price}}{\text{Yesterday's Price}}\right) \tag{1}$$

We use logarithms because:

- A 10% gain followed by 10% loss doesn't bring you back to the start with simple returns
- Log returns are additive: weekly return = sum of daily returns
- They're more "normal" statistically (better for our models)

Why This Matters

Trying to predict prices directly is like predicting exact temperatures. Predicting returns is like predicting whether it will be warmer or cooler—much more manageable. Plus, returns are what investors actually care about for portfolio performance.

Key Takeaways

- Prices go up forever (non-stationary), returns fluctuate around zero (stationary)
- Proper date indexing allows us to use time-series specific functions
- Cryptocurrency returns show much higher volatility than traditional assets

3 Testing for Stationarity

3.1 Cell 3: Are Our Returns Predictable?

What We're Doing

Before forecasting, we need to check if our data is "stationary"—meaning it doesn't have trends or changing behavior over time. Imagine trying to predict tide heights: if you include the long-term sea level rise, your predictions will be off. We need to remove trends first.

The Two Tests:

1. ADF Test (Augmented Dickey-Fuller):

- Asks: "Does this series have a permanent trend (no unit root)?"
- If p-value < 0.05: No trend, series is stationary (good for modeling)
- If p-value ≥ 0.05: Trend exists, need to transform data

2. KPSS Test:

- Asks the opposite: "Is this series stationary?"
- If p-value > 0.05: Series is stationary (good)
- If p-value ≤ 0.05: Series is not stationary (problem)

Combined Results:

ADF Says	KPSS Says	Conclusion
Stationary	Stationary	Definitely stationary
Non-stationary	Non-stationary	Definitely not stationary
Stationary	Non-stationary	Unclear, investigate
Non-stationary	Stationary	Unclear, investigate

Why This Matters

If you try to forecast non-stationary data, you'll get nonsense predictions. It's like trying to predict how a drunk person will walk—there's no pattern to find. We need stationary data where patterns actually exist.

What We Typically Find:

- Bitcoin prices: NOT stationary (they drift upward over time)
- Bitcoin returns: Stationary (they bounce around zero)

This is why we model returns (or log-returns), not prices!

Key Takeaways

- Stationarity means the statistical properties don't change over time
- Prices trend (non-stationary), returns fluctuate (stationary)
- Use both tests together for reliable conclusions
- Only model stationary data with ARIMA

3.2 Cell 4: Visual Confirmation with Rolling Statistics

What We're Doing

We're creating "moving averages" and "moving standard deviations" to see if they stay constant or drift over time. Think of it like a fitness tracker showing your average heart rate over a moving 30-day window—if it keeps climbing, something's changing.

What We Calculate:

- **Rolling Mean**: Average over last 30 days
- **Rolling Std Dev**: Volatility over last 30 days

What to Look For:

- **Stationary series**: Both lines wiggle but stay around the same level
- **Non-stationary series**: Lines drift up/down with clear trends

Why This Matters

Numbers can lie, but charts don't. Visual inspection catches issues that statistical tests might miss, like sudden jumps or structural breaks (e.g., when COVID hit markets).

Key Takeaways

- Price charts show clear trends (non-stationary)
- Return charts bounce around zero (stationary)
- Always combine statistical tests with visual inspection
- Look for structural breaks or regime changes

4 Understanding Patterns in Returns

4.1 Cell 5: ACF and PACF—Finding Memory in Returns

What We're Doing

We're checking if today's return predicts tomorrow's return. In efficient markets, it shouldn't—but sometimes patterns exist that we can exploit.

Two Key Tools:

1. ACF (Autocorrelation Function):

- Measures: "If Bitcoin was up today, will it be up tomorrow? Next week?"
- Shows correlation between today and various days in the past
- Tells us about MA (Moving Average) patterns

2. PACF (Partial Autocorrelation):

- Measures: Direct relationship after removing intermediate effects
- Tells us about AR (Autoregressive) patterns
- Shows if yesterday's return directly affects today

How to Read the Plots:

ACF Pattern	PACF Pattern	What It Means
Spike at lag 1, then dies	Gradual decay	MA(1) model
Gradual decay	Spike at lag 1, then dies	AR(1) model
Both decay gradually	Both decay gradually	Need ARMA
Nothing significant	Nothing significant	White noise (random)

Why This Matters

If crypto returns were truly random (white noise), technical analysis wouldn't work. Finding patterns in ACF/PACF means there's predictable structure we can model and potentially profit from.

Reality Check: Most cryptocurrency returns look close to white noise—markets are fairly efficient. Any patterns tend to be weak and disappear once everyone knows about them.

Key Takeaways

- ACF/PACF reveal if past returns predict future returns
- Significant spikes suggest predictable patterns
- Crypto markets often show weak or no autocorrelation
- Patterns that exist are usually short-lived

5 ARIMA Models—Forecasting Returns

5.1 Cell 6: Building the Prediction Model

What We're Doing

ARIMA combines three components to forecast returns:

ARIMA(p, d, q) Breakdown:

- **AR(p)**: Uses p past values to predict future (like momentum)
- **I(d)**: Number of times we difference the data (for stationarity)
- **MA(q)**: Uses q past forecast errors (learn from mistakes)

Common Models:

- **ARIMA(0,0,0)**: Pure white noise, no pattern
- **ARIMA(1,0,0)**: Today predicts tomorrow (AR)
- **ARIMA(0,0,1)**: Yesterday's surprise affects today (MA)
- **ARIMA(1,0,1)**: Both effects combined

For stationary returns, $d = 0$ always, so we're really doing ARMA models.

Choosing the Best Model:

We use AIC and BIC scores—think of them as "quality per complexity" ratings:

- Lower is better
- BIC penalizes complex models more (prefer simpler)
- Compare multiple models, pick the winner

Why This Matters

ARIMA is the workhorse of time-series forecasting. It's like polynomial fitting but for time series—sophisticated enough to capture real patterns, simple enough to not overfit.

Reality Check: For crypto returns, we often end up with ARIMA(0,0,0) or very simple models. This confirms that returns are hard to predict—markets are quite efficient.

Key Takeaways

- ARIMA captures linear dependencies in returns
- Model selection uses AIC/BIC to balance fit vs. complexity
- Simple models often win for crypto (market efficiency)
- Don't expect amazing return predictions

6 Checking if the Model Works

6.1 Cell 7: Residual Diagnostics

What We're Doing

After fitting ARIMA, we check if the residuals (prediction errors) are truly random. If they're not, we missed something and need a better model.

Good residuals should be:

1. Centered around zero (no bias)
2. Same variance throughout (homoskedastic)
3. No patterns or correlations left
4. Roughly normally distributed

Our Diagnostic Tests:
1. Ljung-Box Test:

- Checks: "Are residuals truly random?"
- If p-value > 0.05: Good, no patterns left
- If p-value < 0.05: Bad, model missed something

2. Jarque-Bera Test:

- Checks: "Are residuals normally distributed?"
- Not critical, but nice to have
- Crypto often fails this (fat tails)

3. ARCH-LM Test:

- Checks: "Is volatility clustering present?"
- If p-value < 0.05: Yes! Need GARCH model
- This is almost always true for crypto

Why This Matters

If diagnostics fail, your forecasts are unreliable. It's like checking if your weather model accounts for all the important factors—if not, predictions will be systematically wrong.

What Usually Happens: Crypto returns pass Ljung-Box (ARIMA captures what little pattern exists) but fail ARCH test (volatility clusters). This tells us: returns are unpredictable, but risk is predictable!

Key Takeaways

- Diagnostic tests validate model assumptions
- ARCH effects indicate volatility clustering
- Crypto almost always shows volatility clustering
- Failed ARCH test means we need GARCH next

7 GARCH Models—Forecasting Volatility

7.1 Cell 8: Predicting Risk, Not Returns

What We're Doing

Here's the key insight: while crypto returns are hard to predict (efficient markets), volatility is very predictable. High volatility today means high volatility tomorrow. This is called "volatility clustering."

What GARCH Does:
GARCH(1,1) is the most common model:

$$\text{Today's Variance} = \text{Base Level} + \alpha \times \text{Yesterday's Shock}^2 + \beta \times \text{Yesterday's Variance} \tag{2}$$

Interpreting the Parameters:

- α: How much yesterday's surprise affects today's volatility
- β: How much yesterday's volatility persists
- $\alpha + \beta$: Total persistence (close to 1 means very persistent)

Example: If $\alpha = 0.1$ and $\beta = 0.85$:

- 10% comes from recent shocks
- 85% comes from past volatility
- Total = 0.95 means volatility is very persistent

Why This Matters

GARCH is crucial for:

- **Risk management**: Knowing when markets will be volatile
- **Position sizing**: Trade smaller when volatility is high
- **Option pricing**: Volatility drives option values
- **Portfolio allocation**: Reduce exposure during risky periods

Reality Check: Cryptocurrencies typically show $\alpha+\beta \approx 0.95-0.99$—very high persistence. When crypto gets volatile, it stays volatile for weeks.

Key Takeaways

- GARCH forecasts volatility, not returns
- Volatility clustering is extremely strong in crypto
- High $\alpha + \beta$ means shocks last a long time
- More useful for risk management than return prediction

8 Monte Carlo Forecasting

8.1 Cell 9: Simulating Future Price Paths

What We're Doing

Instead of predicting a single price, we simulate thousands of possible future paths. This is like running 1000 parallel universes to see the range of outcomes.

The Process:

1. Get Forecasts:

- ARIMA gives us expected return (usually close to zero)
- GARCH gives us expected volatility (varies day-to-day)

2. Simulate Each Path:

- Start at today's price
- For each future day: add expected return plus random shock
- Random shock size = GARCH volatility forecast
- Repeat 1000 times to get 1000 different paths

3. Calculate Confidence Intervals:
From 1000 simulations, we can say:

- 50% chance price is between 25th and 75th percentile
- 90% chance price is between 5th and 95th percentile
- Median (50th percentile) is our "best guess"

Why This Matters

Single-point forecasts are misleading. "Bitcoin will be $45,000 in 30 days" sounds confident but ignores uncertainty. Monte Carlo gives you: "Bitcoin will likely be between $40,000 and $50,000, with $45,000 as the median."

This is critical for:

- Risk assessment: How bad could it get?
- Position sizing: Don't bet more than you can afford to lose
- Scenario planning: Prepare for multiple outcomes

Key Takeaways

- Monte Carlo quantifies forecast uncertainty
- 1000 simulations give reliable confidence intervals
- Wider intervals mean more uncertainty
- Median forecast often differs from mean (asymmetry)

9 Testing the Model

9.1 Cell 10: Does It Actually Work?

What We're Doing

We split data into training (80%) and testing (20%) sets, then forecast the test period to see how well we do. This simulates real-world forecasting.

Key Approach: Rolling Forecast

- Start with training data
- Forecast one day ahead
- Add actual data, refit model
- Forecast next day
- Repeat through entire test period

This is realistic—you'd update your model as new data arrives.

Performance Metrics:

1. RMSE (Root Mean Squared Error):

- Average size of forecast errors
- Heavily penalizes big mistakes
- Compare to benchmark (naive forecast)

2. MAE (Mean Absolute Error):

- Average absolute error
- Easier to interpret than RMSE

3. Directional Accuracy:

- What % of time did we predict direction correctly?
- Random guess = 50%
- Above 50% means model has some skill

4. Comparison to Naive Forecast:

- Naive forecast: "Tomorrow = Today"
- If we can't beat this simple rule, our model is useless

Why This Matters

Many models look great in-sample (on training data) but fail out-of-sample. This is overfitting. Only out-of-sample testing reveals if your model truly works.

Reality Check for Crypto:

- Directional accuracy: 45-55% (barely better than coin flip)
- RMSE: Often similar to naive forecast
- **Conclusion**: Returns are hard to predict (market efficiency)
- **But**: Volatility forecasts are much more accurate!

Key Takeaways

- Out-of-sample testing prevents overfitting
- Beating naive forecasts is the minimum bar
- Directional accuracy around 50% is normal for returns
- Focus on volatility forecasting, not return forecasting

10 What We Learned

10.1 Key Insights

1. Market Efficiency:

- Cryptocurrency returns are nearly unpredictable
- ARIMA models often reduce to white noise
- Directional accuracy barely exceeds 50%
- This is expected in efficient markets

2. Volatility is Predictable:

- GARCH models work well for crypto
- Volatility clustering is very strong
- Today's volatility predicts tomorrow's
- This has practical value for risk management

3. Practical Applications:

- Use GARCH for position sizing (trade less when volatile)
- Monte Carlo for scenario planning and risk assessment
- Don't rely on return forecasts for trading decisions
- Focus on managing risk, not predicting direction

10.2 Limitations to Remember

1. Model Assumptions:

- ARIMA assumes linear relationships
- GARCH assumes symmetric volatility response
- Both assume parameters don't change (but they do)
- Normal distribution assumption often violated

2. Crypto-Specific Issues:

- Limited historical data
- Extreme events (flash crashes) not well-modeled

- Regulatory news causes sudden breaks
- Market structure still evolving

3. What's Missing:

- On-chain metrics (transaction volumes, whale activity)
- Market sentiment and social media
- Cross-asset correlations
- Fundamental drivers (adoption, technology)

10.3 How to Use This in Practice

For Risk Management:

- Use GARCH volatility forecasts for VaR calculations
- Adjust position sizes based on predicted volatility
- Set stop-losses using confidence intervals
- Reduce leverage during high volatility periods

For Portfolio Construction:

- Don't over-rely on return forecasts
- Use volatility forecasts for dynamic allocation
- Rebalance more frequently during volatile periods
- Maintain diversification across assets

For Further Learning:

- Explore asymmetric GARCH models (EGARCH, GJR-GARCH)
- Learn about regime-switching models
- Investigate machine learning for nonlinear patterns
- Study multivariate models for portfolio forecasting

11 Conclusion

Time-series forecasting in cryptocurrency markets reveals a fundamental truth: returns are hard to predict, but volatility is not. This matches financial theory—efficient markets make price prediction difficult, but risk patterns persist and are exploitable.

The tools covered—ARIMA for returns and GARCH for volatility—form the foundation of quantitative finance. While they have limitations, they provide rigorous frameworks for:

- Understanding market dynamics
- Quantifying forecast uncertainty
- Managing portfolio risk systematically
- Making evidence-based decisions

Remember: the goal isn't to predict the future perfectly (impossible), but to understand uncertainty well enough to manage risk effectively. That's what separates professional quantitative analysts from gamblers.