Polymarket Whale Playbooks

How 3 wallets are running Taleb, Simons, and Soros strategies on prediction markets — and the Python code to track them

$1.11M **Hans323 Payout**	**98%** **0x8dxd Win Rate**	**$800K+** **Magamyman Profit**

PredictionMarketsPicks.com | March 2026

Contents

1. The Connection

Three Polymarket wallets. Three trading styles. One pattern: each maps directly to a strategy that has been printing money on Wall Street for decades.

Hans323	Nassim Taleb Barbell Strategy	Buy extreme tails at 2-8 cents. Lose small, win huge.	$92K trade $1.11M payout
0x8dxd	Jim Simons RenTech / Medallion	Thousands of tiny trades exploiting market microstructure lag.	$313 start $2.38M in 4mo
Magamyman	George Soros Reflexivity	Large directional positions on geopolitics where info asymmetry is highest.	$800K+ profit 93% win rate

The infrastructure is different — smart contracts instead of prime brokers, CLOB order books instead of exchange floors — but the logic is identical. Prediction markets are repricing the same edges Wall Street has traded for decades, with lower barriers to entry and wider mispricings.

This guide breaks down each playbook, shows you how each wallet executes it, and gives you working Python code to find the same opportunities yourself.

2. Playbook 1 — The Barbell (Taleb)

Hans323 & Weather Markets

$0.08	$92,632	$1.11M	12x
Entry Price	Position Size	Payout	Return

Nassim Taleb's barbell strategy: put most of your capital in ultra-safe positions and a small slice in extreme-upside trades. You lose pennies most of the time, but when the "impossible" hits, you collect 10x, 50x, or 100x. The key insight is that markets consistently overprice "normal" and underprice tails.

How Hans323 Executes This

Hans323 buys prediction market contracts priced between 2 and 8 cents — events the market considers nearly impossible. In February 2026, he entered a London temperature market at 8 cents, putting in $92,632. The implied probability was 8%. The temperature landed in his range. The contract resolved at $1.00. Result: $1.11M from a single position.

This wasn't luck. Hans323 checked weather models and saw they were giving this temperature range significantly more than 8% probability. The market was mispricing the tail. He's done the same with Trump executive orders at 7 cents, PGA golf markets at 2 cents, and dozens of other positions — all following one pattern: find where the market's implied probability is far below the real-world probability, and buy the tail.

gopfan2 runs an even simpler version: $2M+ across weather markets with a hard rule — buy "Yes" below $0.15, buy "No" above $0.45, $1 limit per position, thousands of small trades. Pure barbell diversification.

Why Weather Markets?

There are 415+ active weather markets on Polymarket right now, with $4.6M in volume. Weather is the ideal barbell hunting ground because: (1) NOAA forecasts are 85-90% accurate at 1-2 days out, (2) prediction markets often price the same outcome at 15-40% — that gap is the edge, (3) weather data is free and public, (4) markets are liquid enough to enter meaningful positions, and (5) resolution is objective and fast.

The Edge Formula

Edge = (Your estimated probability) - (Market implied probability). If NOAA models say a temperature range has a 45% chance and the market prices it at 15 cents (15% implied), your edge is +30 percentage points. That's a massive mispricing. Hans323's London trade had roughly this profile — weather models gave ~40-50% to his range while the market said 8%.

3. Code: NOAA vs Polymarket Weather Scanner

This scanner compares NOAA forecast data to Polymarket weather contract prices, flagging any market where the implied probability is significantly below the forecast probability. Run it daily for 1-2 day out forecasts where NOAA accuracy is highest.

Dependencies

```
pip install requests python-telegram-bot schedule
```

weather_scanner.py

```
import requests, json, time
from datetime import datetime, timedelta

# ■■ Config ■■
TELEGRAM_TOKEN = "YOUR_BOT_TOKEN"
TELEGRAM_CHAT_ID = "YOUR_CHAT_ID"
MIN_EDGE = 0.15  # 15pp minimum edge to alert
POLYMARKET_API = "https://clob.polymarket.com"
GAMMA_API = "https://gamma-api.polymarket.com"

# ■■ NOAA Forecast ■■
def get_noaa_forecast(lat, lon):
    """Get temperature forecast from NOAA API."""
    headers = {"User-Agent": "WeatherScanner/1.0"}
    point = requests.get(
        f"https://api.weather.gov/points/{lat},{lon}",
        headers=headers
    ).json()
    forecast_url = point["properties"]["forecast"]
    forecast = requests.get(
        forecast_url, headers=headers
    ).json()
    periods = forecast["properties"]["periods"]
    return [{
        "name": p["name"],
        "temp": p["temperature"],
        "unit": p["temperatureUnit"],
        "wind_speed": p["windSpeed"],
        "short": p["shortForecast"],
        "detail": p["detailedForecast"]
    } for p in periods[:4]]  # Next 2 days

# ■■ Polymarket Weather Markets ■■
def get_weather_markets():
    """Fetch active weather markets from Polymarket."""
    r = requests.get(f"{GAMMA_API}/markets", params={
        "closed": "false",
        "tag": "Weather",
        "limit": 100,
    })
    return r.json() if r.status_code == 200 else []

def get_market_prices(condition_id):
    """Get current prices for a market."""
    try:
        r = requests.get(
            f"{POLYMARKET_API}/markets/{condition_id}"
        )
        data = r.json()
        prices = data.get("outcomePrices", [])
        return {
            "yes": float(prices[0]) if prices else None,
            "no": float(prices[1]) if len(prices) > 1 else None,
        }
    except:
        return {"yes": None, "no": None}

# ■■ Edge Calculator ■■
def estimate_probability_from_noaa(noaa_temp, market_range):
    """
    Simple model: estimate probability that actual temp
    falls in market_range given NOAA forecast.
    NOAA 1-2 day error is ~3-4 deg F (std dev).
    """
    import math
    low, high = market_range
    mu = noaa_temp
    sigma = 3.5  # NOAA typical std dev at 1-2 days
    # Normal CDF approximation
    def norm_cdf(x):
        return 0.5 * (1 + math.erf(
            (x - mu) / (sigma * math.sqrt(2))
        ))
    return norm_cdf(high) - norm_cdf(low)

# ■■ Telegram Alert ■■
def send_alert(msg):
    requests.post(
        f"https://api.telegram.org/bot{TELEGRAM_TOKEN}"
        f"/sendMessage",
        json={
            "chat_id": TELEGRAM_CHAT_ID,
            "text": msg,
            "parse_mode": "Markdown"
        }
    )

# ■■ Main Scanner ■■
def scan():
    print(f"[{datetime.now()}] Scanning weather markets...")
    markets = get_weather_markets()
    weather_markets = [m for m in markets
        if "temp" in m.get("question","").lower()
        or "weather" in m.get("question","").lower()]

    alerts = []
    for m in weather_markets:
        cid = m.get("conditionId") or m.get("condition_id")
        if not cid:
            continue
        prices = get_market_prices(cid)
        yes_price = prices["yes"]
        if yes_price is None:
            continue

        # Log every market found
        title = m.get("question", "Unknown")
        print(f"  Market: {title}")
        print(f"    Yes price: ${yes_price:.2f}")

        # Flag cheap markets (< $0.25) as potential
        # barbell targets — manual NOAA check needed
        if yes_price < 0.25:
            alerts.append({
                "title": title,
                "yes_price": yes_price,
                "url": f"https://polymarket.com/event/{m.get('slug','')}",
                "volume": m.get("volume", "N/A"),
            })

    if alerts:
        msg = "*Weather Barbell Scanner*\n\n"
        for a in alerts:
            msg += (
                f"*{a['title']}*\n"
                f"Yes: ${a['yes_price']:.2f} "
                f"| Vol: {a['volume']}\n"
                f"{a['url']}\n\n"
            )
        msg += (
            "_Check NOAA forecast vs implied prob. "
            "Edge > 15pp = position candidate._"
        )
        send_alert(msg)
        print(f"  Sent {len(alerts)} alerts")
    else:
        print("  No barbell candidates found")

if __name__ == "__main__":
    scan()
```

How to use: Run this every 6-12 hours. When it flags a market under $0.25, pull up the NOAA forecast for that city/date and estimate the real probability. If your estimate is 15+ percentage points above the market price, that's a barbell candidate. Start small — $50-200 per position.

4. Playbook 2 — The Quant Engine (Simons)

0x8dxd & Crypto Microstructure

$313	$2.38M	26,738	98%
Starting Capital	Final Balance	Total Trades	Win Rate

Jim Simons built Renaissance Technologies on one idea: data beats intuition, and a thousand small edges compound into an unstoppable machine. The Medallion Fund has returned ~66% annually since 1988 — not from one big call, but from millions of tiny statistical arbitrages.

How 0x8dxd Executes This

0x8dxd trades only 15-minute Up/Down markets on BTC, ETH, and SOL. Each position is $4-5K. The edge: Binance spot prices move first, and Polymarket's 15-minute resolution markets update with a few seconds of lag. The bot reads the Binance price, computes which direction the 15-minute candle is heading, and buys the correct side on Polymarket before the market adjusts.

There is no single big win in 0x8dxd's history. The entire $2.38M P&L comes from 26,738 near-identical entries. Discipline beats conviction. This is the exact opposite of Hans323 — and the result is twice as large.

The Microstructure Edge

Prediction markets that reference real-time price feeds (crypto 15-min up/down, hourly BTC range, etc.) have an inherent latency problem. The reference price (Binance, Coinbase) updates every millisecond. The prediction market's order book updates every few seconds. That gap is the edge. It's the same principle behind high-frequency trading on traditional exchanges — just slower and more accessible.

Important: This edge degrades as more traders exploit it. 0x8dxd's 98% win rate was during a period of low competition on these markets. As volume grows, the lag window shrinks. This is a race — the first bot with the lowest latency wins.

5. Code: Binance-Polymarket Lag Detector

This script monitors Binance BTC price in real time and compares it to Polymarket 15-minute Up/Down market prices. When it detects the Binance price has moved decisively but the Polymarket market hasn't repriced yet, it alerts you.

Note: This is a detection/alerting tool, not an auto-trader. Automated trading on Polymarket requires their CLOB API with an approved API key and on-chain USDC. The code below monitors the lag so you can understand the pattern and decide if building a full bot is worth the infrastructure investment.

lag_detector.py

```
import requests, time, json
from datetime import datetime

# ■■ Config ■■
BINANCE_API = "https://api.binance.com/api/v3"
POLYMARKET_CLOB = "https://clob.polymarket.com"
GAMMA_API = "https://gamma-api.polymarket.com"
CHECK_INTERVAL = 2  # seconds between checks
PRICE_MOVE_THRESHOLD = 0.003  # 0.3% move = signal

# ■■ Binance Price Feed ■■
def get_btc_price():
    r = requests.get(f"{BINANCE_API}/ticker/price",
                     params={"symbol": "BTCUSDT"})
    return float(r.json()["price"])

def get_btc_15min_candle():
    """Get current 15-min candle open price."""
    r = requests.get(f"{BINANCE_API}/klines", params={
        "symbol": "BTCUSDT",
        "interval": "15m",
        "limit": 1
    })
    candle = r.json()[0]
    return {
        "open": float(candle[1]),
        "high": float(candle[2]),
        "low": float(candle[3]),
        "close": float(candle[4]),
        "open_time": candle[0],
    }

# ■■ Polymarket 15-min Markets ■■
def get_crypto_15min_markets():
    """Find active BTC 15-minute up/down markets."""
    r = requests.get(f"{GAMMA_API}/markets", params={
        "closed": "false",
        "limit": 50,
    })
    markets = r.json() if r.status_code == 200 else []
    return [m for m in markets
        if "btc" in m.get("question","").lower()
        and ("15" in m.get("question","")
             or "minute" in m.get("question","").lower())]
```

```python
def get_market_yes_price(condition_id):
    try:
        r = requests.get(
            f"{POLYMARKET_CLOB}/markets/{condition_id}"
        )
        prices = r.json().get("outcomePrices", [])
        return float(prices[0]) if prices else None
    except:
        return None

# ■■ Lag Detection Loop ■■
def monitor():
    print("Binance-Polymarket Lag Detector")
    print("=" * 50)
    print(f"Threshold: {PRICE_MOVE_THRESHOLD*100}% move")
    print(f"Checking every {CHECK_INTERVAL}s")
    print()

    markets = get_crypto_15min_markets()
    if not markets:
        print("No active BTC 15-min markets found.")
        return

    print(f"Tracking {len(markets)} markets")
    for m in markets:
        print(f"  - {m.get('question','')}")
    print()

    # Baseline
    candle = get_btc_15min_candle()
    open_price = candle["open"]
    print(f"Candle open: ${open_price:,.2f}")

    while True:
        try:
            current = get_btc_price()
            move = (current - open_price) / open_price

            # Determine direction
            if abs(move) > PRICE_MOVE_THRESHOLD:
                direction = "UP" if move > 0 else "DOWN"
                print(f"\n[{datetime.now().strftime('%H:%M:%S')}]"
                      f" BTC moved {move*100:+.2f}% from open"
                      f" -> {direction}")

                # Check if Polymarket has repriced
                for m in markets:
                    cid = (m.get("conditionId")
                           or m.get("condition_id"))
                    if not cid:
                        continue
                    yes_p = get_market_yes_price(cid)
                    if yes_p is None:
                        continue

                    q = m.get("question","")
                    # If market is "BTC Up?" and price is up
                    # but yes_price is still < 0.60 = lag
                    if direction == "UP" and yes_p < 0.55:
                        print(f"  ** LAG DETECTED **")
                        print(f"  {q}")
                        print(f"  Binance says UP "
                              f"but Yes={yes_p:.2f}")
                    elif direction == "DOWN" and yes_p > 0.45:
                        print(f"  ** LAG DETECTED **")
                        print(f"  {q}")
                        print(f"  Binance says DOWN "
                              f"but Yes={yes_p:.2f}")

            time.sleep(CHECK_INTERVAL)

            # Reset on new candle
            new_candle = get_btc_15min_candle()
            if new_candle["open_time"] != candle["open_time"]:
                candle = new_candle
                open_price = candle["open"]
                print(f"\n--- New candle: "
                      f"${open_price:,.2f} ---")

        except KeyboardInterrupt:
            print("\nStopped.")
            break
        except Exception as e:
            print(f"Error: {e}")
            time.sleep(5)

if __name__ == "__main__":
    monitor()
```

What to watch for: When the detector prints "LAG DETECTED", the Binance price has moved decisively but the Polymarket contract hasn't repriced. This is the window 0x8dxd trades in. Track how often lags appear, how long they last, and how large they are before committing capital.

6. Playbook 3 — The Macro Thesis (Soros)

Magamyman & Geopolitics

$800K+	93%	Iran Strikes	Concentrated
Profit	Win Rate (>$10K)	Key Market	Style

George Soros built his career on reflexivity: the idea that market prices don't just reflect reality — they shape it. His trades aren't statistical; they're thesis-driven. He famously shorted the British pound in 1992 because he understood that the Bank of England couldn't sustain its peg. He was right, made $1B in a day.

How Magamyman Executes This

Magamyman made $800,000 trading U.S. military strikes against Iran. His positions were large and concentrated — the opposite of diversification. He had a thesis about what was going to happen geopolitically, the market was underpricing it, and he loaded up. 93% win rate on positions over $10,000.

CNN and Columbia Law School are analyzing his trades because the timing raises questions about information asymmetry. But the pattern is recognizable: Soros-style macro trading is about forming a thesis on a geopolitical or policy outcome before the market prices it in. The information doesn't have to be insider knowledge — it can be deep expertise in a domain the average trader doesn't follow closely.

The Information Asymmetry Edge

Prediction markets on geopolitics are thin and populated by generalists. If you have domain expertise — military analysis, diplomatic relationships, legislative process, regulatory history — you're trading against people who read headlines. That's the edge. It's the same reason Soros beat the Bank of England: he understood the macro mechanics better than the crowd.

The playbook: (1) Identify a geopolitical event the market is pricing as unlikely, (2) Form a thesis based on deep domain knowledge, (3) Size the position proportionally to your conviction, (4) Have a clear exit if the thesis is invalidated.

7. Code: Whale Wallet Tracker

This script monitors specific Polymarket wallets for new positions and sends Telegram alerts. Track Hans323, 0x8dxd, Magamyman, or any wallet you want to follow.

whale_tracker.py

```
import requests, json, time, hashlib
from datetime import datetime
from pathlib import Path

# ■■ Config ■■
TELEGRAM_TOKEN = "YOUR_BOT_TOKEN"
TELEGRAM_CHAT_ID = "YOUR_CHAT_ID"
GAMMA_API = "https://gamma-api.polymarket.com"
POLL_INTERVAL = 300  # 5 minutes
STATE_FILE = "whale_state.json"

WHALES = {
    "hans323": {
        "name": "Hans323 (Taleb)",
        "address": "WALLET_ADDRESS_HERE",
        "style": "Barbell — tails at 2-8 cents",
    },
    "0x8dxd": {
        "name": "0x8dxd (Simons)",
        "address": "WALLET_ADDRESS_HERE",
        "style": "Quant — 15min crypto microstructure",
    },
    "magamyman": {
        "name": "Magamyman (Soros)",
        "address": "WALLET_ADDRESS_HERE",
        "style": "Macro — geopolitical thesis trades",
    },
}

# ■■ State Management ■■
def load_state():
    if Path(STATE_FILE).exists():
        return json.loads(Path(STATE_FILE).read_text())
    return {}

def save_state(state):
    Path(STATE_FILE).write_text(json.dumps(state))

# ■■ Polymarket Profile Scraper ■■
def get_wallet_positions(username):
    """
    Fetch positions for a Polymarket user.
    Note: Polymarket's public API may change.
    Check gamma-api.polymarket.com for current
    endpoints. You may need to use their GraphQL
    API or on-chain data via Polygonscan.
    """
    try:
        r = requests.get(
            f"{GAMMA_API}/users/{username}/positions",
            timeout=10
        )
        if r.status_code == 200:
            return r.json()
    except:
        pass

    # Fallback: use Polygonscan/Dune for on-chain
    # positions if the API doesn't expose this
    return []

# ■■ Alert Formatting ■■
def format_alert(whale_info, new_positions):
    msg = f"*{whale_info['name']}*\n"
    msg += f"_{whale_info['style']}_\n\n"

    for pos in new_positions[:5]:
        title = pos.get("title", pos.get("question", "Unknown"))
        side = pos.get("outcome", "?")
        size = pos.get("size", "?")
        price = pos.get("avgPrice", "?")
        msg += (f"  {title}\n"
                f"  {side} @ ${price} | ${size}\n\n")

    if len(new_positions) > 5:
        msg += f"_+{len(new_positions)-5} more..._\n"
    return msg

# ■■ Telegram ■■
def send_alert(msg):
    requests.post(
        f"https://api.telegram.org/bot{TELEGRAM_TOKEN}"
        f"/sendMessage",
        json={
            "chat_id": TELEGRAM_CHAT_ID,
            "text": msg,
            "parse_mode": "Markdown"
        }
    )

# ■■ Main Loop ■■
def track():
    print("Whale Wallet Tracker")
    print("=" * 50)
    print(f"Tracking {len(WHALES)} wallets")
    print(f"Poll interval: {POLL_INTERVAL}s")
    print()

    state = load_state()

    while True:
        for key, whale in WHALES.items():
            try:
                positions = get_wallet_positions(
                    whale.get("address", key)
                )
                if not positions:
                    continue

                # Hash current positions to detect changes
                pos_hash = hashlib.md5(
                    json.dumps(positions, sort_keys=True
                    ).encode()
                ).hexdigest()

                prev_hash = state.get(key)
                if prev_hash and prev_hash != pos_hash:
                    # Positions changed — find new ones
                    prev_positions = state.get(
                        f"{key}_positions", []
                    )
                    prev_ids = {
                        p.get("id") for p in prev_positions
                    }
                    new_positions = [
                        p for p in positions
                        if p.get("id") not in prev_ids
                    ]

                    if new_positions:
                        msg = format_alert(whale, new_positions)
                        send_alert(msg)
                        print(f"[{datetime.now():%H:%M}] "
                              f"{whale['name']}: "
                              f"{len(new_positions)} new pos")

                state[key] = pos_hash
                state[f"{key}_positions"] = positions

            except Exception as e:
                print(f"Error tracking {key}: {e}")

        save_state(state)
        time.sleep(POLL_INTERVAL)

if __name__ == "__main__":
    track()
```

Finding wallet addresses: Go to each trader's Polymarket profile, inspect the page source, or use Polygonscan to find the associated wallet. Dune Analytics dashboards like @polymarket_whales can also surface high-activity addresses.

8. Which Playbook Fits You

Capital needed	Low ($500+)	Low ($300+)	High ($10K+)
Technical skill	Basic (API calls)	Advanced (bots, latency)	None (thesis-driven)
Time commitment	Low (scan daily)	High (monitor 24/7)	Medium (research)
Edge source	Public data (NOAA)	Speed (latency arb)	Domain expertise
Win rate	Low (5-15%)	Very high (90%+)	High (80%+)
Payoff per win	Huge (10-100x)	Small (5-20%)	Large (2-10x)
Risk profile	Many small losses, rare huge win	Consistent small gains	Concentrated — high variance

Most traders should start with the Barbell. It requires the least capital, the least technical skill, and the edge (NOAA vs market mispricing) is the easiest to verify. The Quant path requires real engineering and will get more competitive over time. The Macro path requires genuine domain expertise — if you don't have an informational edge, you're just speculating.

9. Risk Management & Position Sizing

Kelly Criterion for Prediction Markets

The Kelly Criterion tells you the optimal fraction of your bankroll to risk on a trade. For prediction markets: f* = (p - q/b) where p = your estimated probability of winning, q = (1-p), and b = (payout/cost - 1). Example: You estimate 40% chance of an event. Market price is $0.15 (15%). Payout is $1.00. b = (1.00/0.15 - 1) = 5.67. f* = (0.40 - 0.60/5.67) = 0.294. Kelly says risk 29.4% of bankroll. In practice, use half-Kelly (14.7%) for safety.

Position Sizing by Playbook

[illegible]	[illegible]	[illegible]
Barbell	1-3% of bankroll	15-20% across all tail positions
Quant	0.5-1% per trade	5% at any moment (positions resolve in 15min)
Macro	5-10% of bankroll (high conviction)	25% max (concentrated by design)

The #1 rule: Never risk more than you can afford to lose entirely. Prediction markets are volatile, contracts can go to zero, and platforms carry counterparty risk. Treat your prediction market bankroll as separate from your savings.

10. Resources & API Reference

APIs

Polymarket CLOB	clob.polymarket.com	Order book, prices, trading
Polymarket Gamma	gamma-api.polymarket.com	Market search, metadata
NOAA Weather	api.weather.gov	Forecasts (free, no key)
Binance	api.binance.com	Crypto prices, candles
Polygonscan	polygonscan.com/api	On-chain wallet activity
Dune Analytics	dune.com/api	Whale dashboards, queries

On-Chain Research Tools

Dune Analytics: Search for @polymarket dashboards. Top whale trackers update daily. You can fork any query and customize filters for volume, win rate, or specific markets.

Bubblemaps: Visualizes wallet clusters. Use it to find if multiple wallets are controlled by the same entity (connected wallets trading the same markets).

Polymarket Analytics: Third-party dashboards that aggregate leaderboard data, volume trends, and market-level statistics.

Wallet Links

Hans323: https://polymarket.com/@hans323
0x8dxd: https://polymarket.com/@0x8dxd
Magamyman: https://polymarket.com/@magamyman