write the following as a trading algo: --identify highest % market mover (100% +)
--identify peak from candle signs of degradation from peak (i.e., triple tier surges, double top-end wick, previous resistance <months timeline>, degrading candles, head-and-shoulders, etc.)
--short for 15% - 20% gain (for consistency)
--bounce / level-off price (support) (previous close + day high avg.)
--DO NOT short AI driven news stocks
--previously flat price for weeks prior is ideal

python

```python
# Trading Algorithm: Short High-Volatility Movers After Peak Degradation
# Language: Python (using pandas, numpy, yfinance for data; backtrader or similar for live execut
# Assumptions:
# - Daily timeframe (can be adapted to 1H/4H)
# - Scans US equities (extendable to crypto/others)
# - Uses yfinance for historical data
# - Filters out AI/news-driven stocks via keyword + volume spike check
# - Targets 15-20% short gains with high consistency
# - Ideal setup: flat price for 4+ weeks → explosive move → peak degradation

import yfinance as yf
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import warnings
warnings.filterwarnings("ignore")

# ================================
# CONFIGURATION
# ================================
LOOKBACK_FLAT_WEEKS = 4          # Weeks of flat price before surge
MIN_FLAT_STD = 0.015             # Max std dev of returns in flat period (<1.5%)
MIN_SURGE_PCT = 1.00             # 100%+ move from flat base
DEGRADATION_LOOKBACK = 10        # Candles to detect peak degradation
TARGET_GAIN_PCT = (0.15, 0.20)    # Short target: 15-20%
SUPPORT_METHOD = 'avg_prev'      # 'avg_prev': (prev_close + day_high)/2
AI_KEYWORDS = ['AI', 'artificial intelligence', 'neural', 'GPT', 'LLM', 'deep learning']

# ================================
# HELPER FUNCTIONS
# ================================

def is_flat_price(df, weeks=LOOKBACK_FLAT_WEEKS, threshold=MIN_FLAT_STD):
    """Check if price was flat for N weeks before current surge"""
    end_date = df.index[-1]
    start_date = end_date - timedelta(weeks=weeks)
    flat_df = df[df.index >= start_date]
    if len(flat_df) < 10:
        return False
    returns = flat_df['Close'].pct_change().dropna()
    return returns.std() < threshold

def detect_surge(df, min_pct=MIN_SURGE_PCT):
    """Find base price (flat low) and check for 100%+ surge"""
    flat_period = df.iloc[-(LOOKBACK_FLAT_WEEKS*5):]  # approx 5 trading days/week
    base_price = flat_period['Low'].min()
    peak_price = df['High'].max()
    surge_pct = (peak_price - base_price) / base_price
    return surge_pct >= min_pct, base_price, peak_price

def is_ai_news_stock(ticker):
    """Crude filter: avoid tickers with AI buzz + abnormal volume"""
    try:
        info = yf.Ticker(ticker).info
        name = info.get('longName', '').lower()
        business = info.get('longBusinessSummary', '').lower()
        sector = info.get('sector', '').lower()
        volume_avg = info.get('averageVolume', 0)
        volume_today = info.get('volume', 0)

        has_ai_keyword = any(kw in name or kw in business for kw in AI_KEYWORDS)
        volume_spike = volume_today > 3 * volume_avg if volume_avg > 0 else False
        return has_ai_keyword and volume_spike
    except:
        return True  # Err on side of caution

def detect_peak_degradation(df, lookback=DEGRADATION_LOOKBACK):
    """
    Detect signs of peak exhaustion:
    - Triple tier surges (3 diminishing up moves)
    - Double top with upper wick
    - Retest of previous resistance (last 3 months)
    - Degrading candle bodies
    - Head and shoulders (simplified)
    """
    recent = df.tail(lookback).copy()
    if len(recent) < 5:
        return False, "Insufficient data"

    high_idx = recent['High'].idxmax()
    peak_row = recent.loc[high_idx]
    post_peak = recent[recent.index > high_idx]

    signals = []

    # 1. Double top + upper wick
    highs = recent['High']
    tops = highs[highs >= 0.98 * highs.max()]
    if len(tops) >= 2:
        last_top = tops.index[-1]
        if recent.loc[last_top, 'Close'] < recent.loc[last_top, 'Open'] * 1.01:
            signals.append("Double top with rejection wick")

    # 2. Triple diminishing surges
    closes = recent['Close']
    surges = []
    for i in range(1, len(closes)):
        surge = (closes.iloc[i] - closes.iloc[i-1]) / closes.iloc[i-1]
        if surge > 0.05:
            surges.append(surge)
    if len(surges) >= 3 and surges[-1] < surges[-2] < surges[-3]:
        signals.append("Triple diminishing surges")

    # 3. Previous resistance (last 3 months)
    three_mo_ago = df.index[-1] - timedelta(days=90)
    old_df = df[df.index >= three_mo_ago]
    resistance = old_df['High'].quantile(0.9)
    if abs(peak_row['High'] - resistance) / resistance < 0.05:
        signals.append("Retesting previous resistance")

    # 4. Degrading candle bodies
    recent['body'] = abs(recent['Close'] - recent['Open'])
    recent['wick_ratio'] = (recent['High'] - recent['Low']) / (recent['body'] + 1e-6)
    if (recent['body'].pct_change().tail(3) < -0.3).all():
        signals.append("Degrading candle bodies")

    # 5. Head and Shoulders (simplified)
    prices = recent['High'].values
    if len(prices) >= 5:
        left_shoulder = np.argmax(prices[:len(prices)//3])
        head = np.argmax(prices)
        right_shoulder = len(prices) - 1 - np.argmax(prices[::-1][:len(prices)//3])
        if (left_shoulder < head > right_shoulder and
            abs(prices[left_shoulder] - prices[right_shoulder]) / prices[head] < 0.1 and
            prices[head] > prices[left_shoulder] * 1.1):
            signals.append("Head and shoulders pattern")

    return len(signals) >= 1, " | ".join(signals) if signals else "None"
```

```python
def calculate_support(df):
    """Support = average of previous close and day high (of surge day)"""
    surge_day = df[df['High'] == df['High'].max()].iloc[0]
    prev_close = df[df.index < surge_day.name]['Close'].iloc[-1]
    day_high = surge_day['High']
    return (prev_close + day_high) / 2

# ================================
# MAIN SCANNER
# ================================

def scan_for_short_setup(tickers, end_date=None):
    if end_date is None:
        end_date = datetime.now()
    start_date = end_date - timedelta(days=180)  # 6 months data

    candidates = []

    for ticker in tickers:
        try:
            df = yf.download(ticker, start=start_date, end=end_date, progress=False)
            if df.empty or len(df) < 50:
                continue

            df = df.dropna()

            # Step 1: Filter AI/news stocks
            if is_ai_news_stock(ticker):
                continue

            # Step 2: Was price flat for weeks?
            if not is_flat_price(df):
                continue

            # Step 3: Did it surge 100%+?
            surged, base, peak = detect_surge(df)
            if not surged:
                continue

            # Step 4: Detect peak degradation
            degraded, reason = detect_peak_degradation(df)
            if not degraded:
                continue

            # Step 5: Calculate entry, target, support
            entry_price = df['Close'].iloc[-1]  # Short at current close
            support = calculate_support(df)
            target_low = entry_price * (1 - np.mean(TARGET_GAIN_PCT))
            target_high = entry_price * (1 - min(TARGET_GAIN_PCT))

            # Risk: stop above recent peak
            stop_loss = df['High'].max() * 1.02

            candidate = {
                'ticker': ticker,
                'surge_pct': round((peak - base)/base * 100, 2),
                'entry': round(entry_price, 2),
                'target_15%': round(target_high, 2),
                'target_20%': round(target_low, 2),
                'support_level': round(support, 2),
                'stop_loss': round(stop_loss, 2),
                'degradation_signals': reason,
                'flat_period_std': round(df.tail(20)['Close'].pct_change().std(), 4)
            }
            candidates.append(candidate)

        except Exception as e:
            continue

    return pd.DataFrame(candidates).sort_values('surge_pct', ascending=False)

# ================================
# EXECUTION EXAMPLE
# ================================
if __name__ == "__main__":
    # Example: Scan small-cap/high-vol universe (replace with your screener)
    tickers = [
        'NVAX', 'GME', 'AMC', 'BB', 'CLSK', 'MARA', 'RIOT', 'CLOV', 'SPCE', 'PLUG',
        'FCEL', 'BLNK', 'NIO', 'XPEV', 'LI', 'RIDE', 'NKLA', 'LCID', 'RIVN', 'SOFI'
    ] * 10  # Expand as needed

    print("Scanning for short setups...\n")
    results = scan_for_short_setup(tickers)

    if not results.empty:
        print(results.to_string(index=False))
        print(f"\n{len(results)} high-conviction short setups found.")
        print("Ideal: Flat → 100%+ explosion → peak degradation → short for 15–20%")
    else:
        print("No setups found matching criteria.")
```

Strategy Summary (Embedded in Code)

Criteria	Implementation
100%+ Mover	`detect_surge()` from flat base
Peak Degradation	5+ bearish reversal patterns
Short 15–20%	Fixed % target from entry
Support Level	`(prev_close + day_high)/2`
Avoid AI News	Keyword + volume spike filter
Flat Prior Weeks	Low std dev in returns (≤1.5%)

Next Steps for Live Trading

1. **Replace** `tickers` with Finnhub/IEX/Polygon screener for real-time 100%+ movers.
2. **Add position sizing** (e.g., 1–2% risk per trade).
3. **Integrate with broker API** (Alpaca, Interactive Brokers).
4. **Add time filter**: Enter only after 2–3 degradation candles post-peak.

> ***Warning: High-risk strategy. Backtest rigorously. Not financial advice.***

Let me know if you want:

- Backtest module
- Real-time WebSocket version
- Crypto adaptation
- Risk management overlay

Ask anything