Project Master Summary:
Bitcoin Sentiment & Trader Performance

Created by Nishant Sahni

September 25, 2025

Contents

1 Project Objective

Explore the relationship between Bitcoin market sentiment (Fear / Greed index) and Hyperliquid traders' realized performance — find patterns, check predictability, quantify effects, and produce reproducible deliverables (cleaned data, notebooks, analysis, event studies, regressions, report, presentation, and a demo).

2 Raw Inputs

- `/mnt/data/historical_data.csv`: Hyperliquid trades (per-trade records).
  - **Key raw columns detected**: Account, Coin, Execution Price, Size Tokens, Size USD, Side, Timestamp IST, Timestamp, Closed PnL, Trade ID, Order ID, Transaction Hash, Start Position, Direction, Fee, Crossed.
- `/mnt/data/fear_greed_index.csv`: Bitcoin Fear/Greed index.
  - **Columns detected**: timestamp, value (0–100 numeric index), classification (Extreme Fear/Fear/Neutral/Greed/Extreme Greed), date.

3 Produced Files Inventory

All generated files are saved under: `/mnt/data/submission_package/`.

- **Data (cleaned)**
  - `trades_clean.csv`: Cleaned & normalized trades (UTC time, date, size_abs, closedPnL, is_win, side_std, closedPnL_outlier, etc.)
  - `sentiment_clean.csv`: Cleaned sentiment (date, value, sent_index_raw, sent_index_norm [-1..1], sent_label_clean, sent_numeric_label)
- **Aggregates / Features**
  - `daily_account_agg.csv`: Per-account, per-day aggregates (daily_pnl, num_trades, win_rate, avg_size, median_size, pnl_std, num_outliers)
  - `daily_overall_agg.csv`: Market-level daily aggregates (sum/averages)
  - `merged_daily_sentiment.csv`: Merged daily market metrics with sentiment
- **Event Study Outputs**
  - `event_study_market_low.csv` & `event_study_market_high.csv`: Market-level event stats (rel_day -3..3)
  - `event_study_account_low.csv` & `event_study_account_high.csv`: Account-level event stats
  - `event_study_market_low.png`, `event_study_market_high.png`, `event_study_account_low.p` `event_study_account_high.png`: PNG plots
- **Causality**
  - `granger_full_summary.json`: ADF results, diff flags, Granger p-values serialized

- granger_account_summary.csv: Per-account Granger summary (top tested accounts, p-values)

- **Notebooks & Code**
  - 1_eda.ipynb
  - 2_features_and_aggregation.ipynb
  - 3_analysis.ipynb
  - streamlit_app.py: Tiny demo to inspect tables

- **Deliverables**
  - final_report.pdf: Short 2-page report
  - final_report_detailed.pdf: Detailed multi-page final report
  - presentation.pdf: 8-slide deck
  - README.md
  - requirements.txt

4 Data Cleaning Steps

4.1 Trades Cleaning

- **Column normalization**: Renamed columns to lowercase/standard names (e.g., Closed PnL -> closedPnL).

- **Timestamp parsing**:
  - Preferred timestamp_ist (assumed IST), localized to Asia/Kolkata, then converted to **UTC**.
  - Fell back to timestamp (parsed as UTC) if timestamp_ist was unavailable.
  - Final column: time (timezone-aware UTC). Derived date from time.

- **Numeric coercions**: Coerced execution_price, size_tokens, size_usd, closedPnL, fee to numeric, setting errors to NaN.

- **Derived size_abs**: Preferred size_tokens.abs(); fell back to size_usd.abs().

- **Standardize side**: Mapped common values to side_std (**+1** for buy/long, **-1** for sell/short). Inferred from size sign if contradictory.

- **Flags & derived fields**:
  - is_win = closedPnL > 0
  - closedPnL_abs, closedPnL_outlier flag where absolute PnL > 99.9th percentile.

- **Row removals**:
  - Dropped rows missing time, account, or size_abs.
  - Removed zero-size trades.
  - Removed duplicates by trade id (kept last).

4.2 Sentiment Cleaning

- **Timestamp parsing**: Final `date` field is the calendar date.
- **Value normalization**:
  - `sent_index_raw`: Retained original 0–100 value.
  - `sent_index_norm`: Normalized to [-1, 1] using the formula:

$$\text{sent\_index\_norm} = \frac{(\text{sent\_index\_raw} - \min)}{(\max - \min)} \times 2 - 1$$

- **Labels**: Kept classification as `sent_label_clean` and mapped to numeric labels (`sent_numeric_labe` Extreme Fear: -2, Fear: -1, Neutral: 0, Greed: 1, Extreme Greed: 2.
- **Deduplication**: Kept the last entry per date and forward-filled numeric values where appropriate.

4.3 Join Criteria

Daily aggregations were merged on `date` (UTC date) using a left join.

5 Reproducible Run Commands

Run from the parent directory of `submission_package`.

```
# from parent folder of submission_package
python -m venv venv
source venv/bin/activate
pip install -r submission_package/requirements.txt
# (optional) pillow for embedding/PNG handling:
pip install pillow

# Launch Jupyter to run notebooks:
jupyter notebook
# open and run:
# submission_package/1_eda.ipynb
# submission_package/2_features_and_aggregation.ipynb
# submission_package/3_analysis.ipynb

# Launch demo:
streamlit run submission_package/streamlit_app.py
```

6 Key Numeric Descriptive Results

6.1 Dataset Scale

- **Total trades processed**: 211,224
- **Total sentiment rows**: 2,644
- **Time span (trades)**: 2023-01-04 19:36:00 UTC → 2025-12-04 18:25:00 UTC
- **Critical data-quality note**: The max trade timestamp (2025-12-04) is in the future relative to the analysis date (2025-09-25) and must be verified.

- **Unique accounts**: 32
- **Unique symbols**: 246

6.2 Closed PnL (trade-level) Stats

- **Sum of closedPnL**: $10,296,958.94
- **Mean closedPnL**: $48.7490
- **Median closedPnL**: $0.00
- **Standard Deviation**: $919.165
- **Skewness**: 30.6994 (extremely heavy positive skew)
- **Interpretation**: A tiny fraction of trades generates most of the realized PnL. The median of 0 indicates many trades are break-even. Analysis must use robust statistics (median, trimmed mean) or winsorize.

7 Event Study

7.1 Methodology

- **Event Definition**: Extreme events were defined as the bottom 10% (Extreme Fear) and top 10% (Extreme Greed) of the `sent_index_norm`.
- **Window**: For each event date t, metrics were computed for the window $t-3, \ldots, t+3$.
- **Aggregation Lenses**:
  1. **Market-level**: Sum of `daily_pnl` across all accounts.
  2. **Account-level**: Average of each account's `daily_pnl` for that day, then averaged across all events.
- **Statistics**: For each relative day, the mean, standard deviation, sample count, and 95% CI were computed.

7.2 Counts

- **Extreme Fear days** (bottom 10%): 48 unique event dates
- **Extreme Greed days** (top 10%): 58 unique event dates

7.3 Mean Daily PnL at t=0

- **Market-level (Extreme Fear)**: $26,418.35
- **Market-level (Extreme Greed)**: $31,416.26
- **Account-level (Extreme Fear)**: $2,929.08
- **Account-level (Extreme Greed)**: $3,098.98
- **Interpretation**: On average, event days (both Fear and Greed) coincide with elevated PnL, suggesting increased activity or intensity. However, wide confidence intervals indicate large dispersion.

8 Granger-Causality Analysis

8.1 Preprocessing

- Daily series used: `market_daily_pnl` and `sent_index_norm`.
- **Stationarity**: Augmented Dickey-Fuller (ADF) tests were performed. Non-stationary series were differenced once.
- **Test**: `statsmodels.tsa.stattools.grangercausalitytests` was used with lags from 1 to 7.

8.2 Market-Level Results (p-values for F-test by lag)

- **Sentiment** $\rightarrow$ **`market_daily_pnl`**:
  - lag1: 0.1251, lag2: 0.1930, lag3: 0.2068, lag4: 0.3188, lag5: 0.3319, lag6: 0.3316, lag7: 0.0901
- **`market_daily_pnl`** $\rightarrow$ **Sentiment**:
  - lag1: 0.00213, lag2: 0.00383, lag3: 0.00368, lag4: 0.00822, lag5: 0.01480, lag6: 0.02242, lag7: 0.01225

8.3 Interpretation

- There is **stronger evidence that market moves drive sentiment** (`market_daily_pnl` $\rightarrow$ sentiment) than the reverse. P-values for this direction are consistently < 0.05.
- Sentiment does not robustly Granger-cause `market_daily_pnl` at a daily frequency ($p>0.05$ for most lags).
- **Implication**: The Fear/Greed index in this dataset appears to be a **reactive indicator**, following market activity rather than leading it.

9 Notebooks & Code Structure

- `1_eda.ipynb`: Loads cleaned data for basic exploratory data analysis, distributions, and plots.
- `2_features_and_aggregation.ipynb`: Demonstrates feature engineering (e.g., rolling features) and creates daily aggregate files.
- `3_analysis.ipynb`: Contains example code for OLS regression, the event study, and stubs for further analysis (Granger, VAR, ML).

10 Final Detailed Report Contents (`final_report_detailed.pdf`)

1. **Title & Executive Summary**: Key numeric findings and recommendations.
2. **Data & Cleaning Log**: Detailed list of all transformations with justifications.
3. **Exploratory Data Analysis**: Time-series plots, histograms, logarithmic tail visualizations, and full numeric stats.

4. **Rolling Correlation**: A 30-day rolling correlation plot and its interpretation.

5. **Event Study**: Numerical results and embedded charts for both market-level and account-level analyses with confidence intervals.

6. **Granger Causality**: Detailed ADF summaries, differencing decisions, p-value tables, and interpretation.

7. **Recommendations & Next Steps**: Prioritized list including panel regressions, clustering, and predictive modeling.

8. **Appendix**: Full file list and reproduction steps.

11 Limitations, Biases, and Validity Threats

- **Skew / Outliers**: Aggregate metrics are dominated by a few large PnL trades. Robust statistics are necessary.
- **Confounding (Omitted Variables)**: The analysis lacks controls for BTC returns, volatility, or macro news, which could create omitted-variable bias.
- **Reverse Causality**: Granger tests indicate market moves precede sentiment changes, making naive predictive use of the index risky.
- **Sampling & Survivorship Bias**: Data is limited to Hyperliquid and may not represent the broader market.
- **Time Resolution Mismatch**: Daily sentiment may obscure intraday patterns where it could be predictive.
- **Future-Dated Timestamps**: The presence of future timestamps must be investigated and resolved.
- **Stationarity & Small Samples**: Account-level series were often too short for robust time-series analysis after differencing.
- **Metric Selection**: Summing PnL across traders (pooling) ignores heterogeneity. Panel models with fixed effects would provide better inference.

12 Recommended Next Analyses

12.1 Panel Fixed-Effects Regression

- **Purpose**: Control for account-specific traits and test if lagged sentiment predicts daily PnL after controlling for activity.
- **Specification**:

  $$\text{daily\_pnl}_{it} = \alpha_i + \beta_1 \cdot \text{sent\_index\_norm}_t + \beta_2 \cdot \text{sent\_index\_norm}_{t-1} + \gamma \cdot X_{it} + \delta_t + \epsilon_{it}$$

  Where α_i is the account fixed effect, δ_t is an optional day fixed effect, and X_{it} includes controls like `num_trades` and `BTC_return_t`.

- **Code Snippet (`linearmodels`)**:

```python
from linearmodels.panel import PanelOLS
# ... (load and merge data with sentiment and BTC prices)
df = df.set_index(['account','date'])
df['sent_lag1'] = df.groupby('account')['sent_index_norm'].shift(1)
exog_vars = ['sent_index_norm','sent_lag1','num_trades',
             'avg_size','btc_return','realized_vol']
exog = df[exog_vars].dropna()
mod = PanelOLS(df['daily_pnl'], exog, entity_effects=True,
               time_effects=False)
res = mod.fit(cov_type='clustered', cluster_entity=True)
print(res.summary)
```

12.2 Predictive Model (Classification)

- **Target**: `profitable_day` (where `daily_pnl > 0`).
- **Approach**: Use features like lagged sentiment, rolling stats, and BTC data. Train models (Logistic Regression, XGBoost) using time-aware cross-validation (`TimeSeriesSplit`) and evaluate with ROC-AUC.

12.3 Trader Clustering (Segmentation)

- **Features**: Use account-level stats (mean PnL, volatility, trade frequency) to cluster traders using KMeans. Analyze each cluster's sensitivity to sentiment separately.

12.4 Event Study Variants

- **Robustness**: Re-run studies with different thresholds (e.g., 5%, 20%), windows (e.g., [-1, +1]), and use robust metrics like medians.

13 How to Incorporate BTC Price & Volatility

1. **Get Data**: Obtain daily BTC closing prices aligned with the sentiment data dates.
2. **Compute Returns**: Calculate daily returns: $\text{btc\_return}_t = (\frac{\text{close}_t}{\text{close}_{t-1}} - 1)$.
3. **Compute Volatility**: Calculate realized volatility, e.g., the rolling 30-day standard deviation of daily returns.
4. **Merge**: Merge these new features into the daily aggregate datasets on the `date` column.

14 Full Prioritized Action Plan

- **Priority 0 — Data Integrity**: Confirm and fix or remove future-dated timestamps.
- **Priority 1 (2–4 hours)**:
  1. Run the panel fixed-effects regression with BTC controls.
  2. Conduct robust event studies using medians.
- **Priority 2 (1 day)**:

1. Build the classification model for predicting a profitable day.
2. Cluster traders and analyze sentiment sensitivity per cluster.

- **Priority 3 (2–3 days)**:
  1. Develop a full Streamlit dashboard with interactive filters and visualizations.
  2. Prepare the final slide deck with the top 3 insights and actionable recommendations.

15 Concrete Recommendations for a Trading Team

1. **Do not use the raw Fear/Greed index as a leading signal.** Evidence suggests it is a reactive indicator. Use it conditionally, after controlling for price and volatility.
2. **Use robust risk-sizing during periods of Extreme Greed.** Event studies show elevated PnL, which likely corresponds to higher volatility. Consider reducing position sizes or tightening stop-losses.
3. **Segment traders.** Identify clusters that are particularly sensitive to sentiment changes and manage their risk accordingly.
4. **Use robust performance metrics.** Monitor daily performance using medians or winsorized means to avoid being misled by outliers.
5. **A/B test any new policy.** Before deploying a sentiment-aware strategy, backtest it rigorously using time-series cross-validation.

16 Exact Code Snippets & Helper Utilities

16.1 Auto-detect DATA_ROOT

```
from pathlib import Path
BASE = Path.cwd()
if (BASE / 'submission_package').exists() and not (BASE / 'trades_clean.csv').exists():
    DATA_ROOT = BASE / 'submission_package'
elif (BASE / 'trades_clean.csv').exists():
    DATA_ROOT = BASE
else:
    DATA_ROOT = BASE / 'submission_package'
print("DATA_ROOT:", DATA_ROOT)
```

16.2 Load Cleaned Data

```
import pandas as pd
trades = pd.read_csv(DATA_ROOT / 'trades_clean.csv', parse_dates=['time'])
sent = pd.read_csv(DATA_ROOT / 'sentiment_clean.csv', parse_dates=['date'])
daily = pd.read_csv(DATA_ROOT / 'daily_account_agg.csv', parse_dates=['date'])
merged = pd.read_csv(DATA_ROOT / 'merged_daily_sentiment.csv', parse_dates=['date'])
```

16.3 Event Study Helper

```
import numpy as np
def build_event_matrix(event_dates, series_df, value_col, window=3):
    rows=[]
    series_map = series_df.set_index('date')[value_col].to_dict()
    for ev in event_dates:
        for k in range(-window, window+1):
            d = (pd.to_datetime(ev) + pd.Timedelta(days=k)).date()
            rows.append({'event_date': ev, 'rel_day': k,
                         'value': series_map.get(d, np.nan)})
    return pd.DataFrame(rows)
```

16.4 Granger Causality Quick Call

```
from statsmodels.tsa.stattools import grangercausalitytests
data = merged[['market_daily_pnl','sent_index_norm']].dropna()
# ensure stationarity or diff once
grangercausalitytests(data[['market_daily_pnl','sent_index_norm']], maxlag=7)
```

17 Location of Important Outputs

- **Detailed Report**: /mnt/data/submission_package/final_report_detailed.pdf
- **Cleaned Data**: /mnt/data/submission_package/*.csv
- **Event Study Plots**: /mnt/data/submission_package/event_study_*.png
- **Granger Results**: /mnt/data/submission_package/granger_full_summary.json
- **Notebooks**: /mnt/data/submission_package/*.ipynb