Data Science Assignment – Web3 Trading

Interpretation of Sentiment Data Loading and Preprocessing:

The Bitcoin Market Sentiment data was loaded successfully, containing 2644 entries and 4 initial columns: timestamp, value, classification, and date.

- **Initial State:** The date column was initially an object (string) type, and classification was also an object type, while timestamp and value (the numerical sentiment score ranging from 0-100) were integers.
- **Missing Values:** No missing values were found in any of the columns, simplifying the cleaning process.
- **Date Conversion:** The date column (e.g., '2018-02-01') was successfully converted to the datetime64[ns] data type.
- **Sentiment Classifications:** The classification column contains 5 unique sentiment states: 'Fear', 'Extreme Fear', 'Neutral', 'Greed', and 'Extreme Greed'. "Fear" was the most frequent classification.
- **Indexing:** The date column was subsequently set as the index of the sentiment_df DataFrame.
- **Data Range:** The sentiment data spans from 2018-02-01 to 2025-05-02 (derived from min/max of the index after processing, or from later merge step).

The sentiment_df DataFrame is now cleaned and prepared for merging with the trader data. The key columns for our analysis will be the index (date), value, and classification.

Next, the historical trader data provided by Hyperliquid is loaded. This dataset is expected to contain granular details of individual trades, which will be crucial for assessing trader performance. The initial steps involve:

- Loading the data from the historical_data.csv file.
- Displaying the first and last few rows to understand the data's appearance and structure.
- Getting a summary of column names, data types, and non-null counts.
- Calculating descriptive statistics for numerical and categorical columns.
- Checking for missing values.

- Performing a preliminary inspection of potential key columns that were mentioned in the assignment outline (e.g., for account, symbol, side, event, time), while noting that the actual column names might differ

Trader Data Preprocessing

Based on the initial exploration, the raw trader data requires several preprocessing steps to make it suitable for analysis and merging:

1. **Standardize Column Names:** Column names are converted to lowercase and snake_case for consistency and ease of use in Python (e.g., 'Execution Price' becomes 'execution_price', 'Coin' becomes 'symbol', 'Direction' becomes 'event_type').
2. **Convert Timestamp:** The numeric Timestamp column (identified as timestamp_unix after renaming) needs to be converted into a proper datetime object. Based on its magnitude, it's interpreted as Unix time in milliseconds. A new time column (datetime with time) and a date column (date part only, for merging with daily sentiment) will be created.
3. **Verify Data Integrity:** After transformations, data types are checked again, and any anomalies like negative fees are briefly inspected.

A copy of the original trader DataFrame (trader_df) is made into trader_df_processed to ensure non-destructive transformations.

Further Exploration of Processed Trader Data: Symbols and Event Types

Before filtering for Bitcoin-specific trades and merging with sentiment data, it's important to further understand the diversity of symbols (crypto assets) and the nature of event_type values in the processed trader data (trader_df_processed). This step involves:

- Examining the unique symbols present and identifying those related to Bitcoin.
- Investigating the unique event_type values to understand the different trading actions recorded (e.g., opening/closing positions, liquidations).
- Briefly checking the behavior of start_position_tokens in relation to different event types to confirm understanding of position management (long vs. short, opening vs. closing).

Further Exploration of Processed Trader Data: Symbols and Event Types

Before filtering for Bitcoin-specific trades and merging with sentiment data, it's important to further understand the diversity of symbols (crypto assets) and the nature of event_type values in the processed trader data (trader_df_processed). This step involves:

- Examining the unique symbols present and identifying those related to Bitcoin.
- Investigating the unique event_type values to understand the different trading actions recorded (e.g., opening/closing positions, liquidations).
- Briefly checking the behavior of start_position_tokens in relation to different event types to confirm understanding of position management (long vs. short, opening vs. closing).

Filtering for Bitcoin Trades and Defining PnL-Realizing Events

To align with the project's objective of analyzing Bitcoin market sentiment, the trader dataset (trader_df_processed) is now filtered to include only trades involving Bitcoin-related symbols. Additionally, we need to precisely define which event_type values signify a realization of profit or loss.

This involves:

1. Creating a new DataFrame trader_df_btc containing only rows where the symbol is 'BTC' or 'AIXBT'.
2. Examining the generic 'Buy' and 'Sell' event_types within this Bitcoin-specific dataset to see if they represent PnL-realizing events or merely position adjustments.
3. Confirming the list of event_types that will be considered for PnL analysis (e.g., 'Close Long', 'Close Short', liquidations).

Merging Processed Trader Data with Sentiment Data

1. With both the Bitcoin-specific trader data (`trader_df_btc`) and the market sentiment data (`sentiment_df`) prepared, the next crucial step is to merge them. This will allow each Bitcoin trade record to be annotated with the corresponding market sentiment on the day the trade event occurred.
2. The merge is performed using the `date` column present in `trader_df_btc` and the `date` index of `sentiment_df`. A left merge ensures that all Bitcoin trade records are kept, and sentiment information is added where available.

Exploratory Data Analysis (EDA)

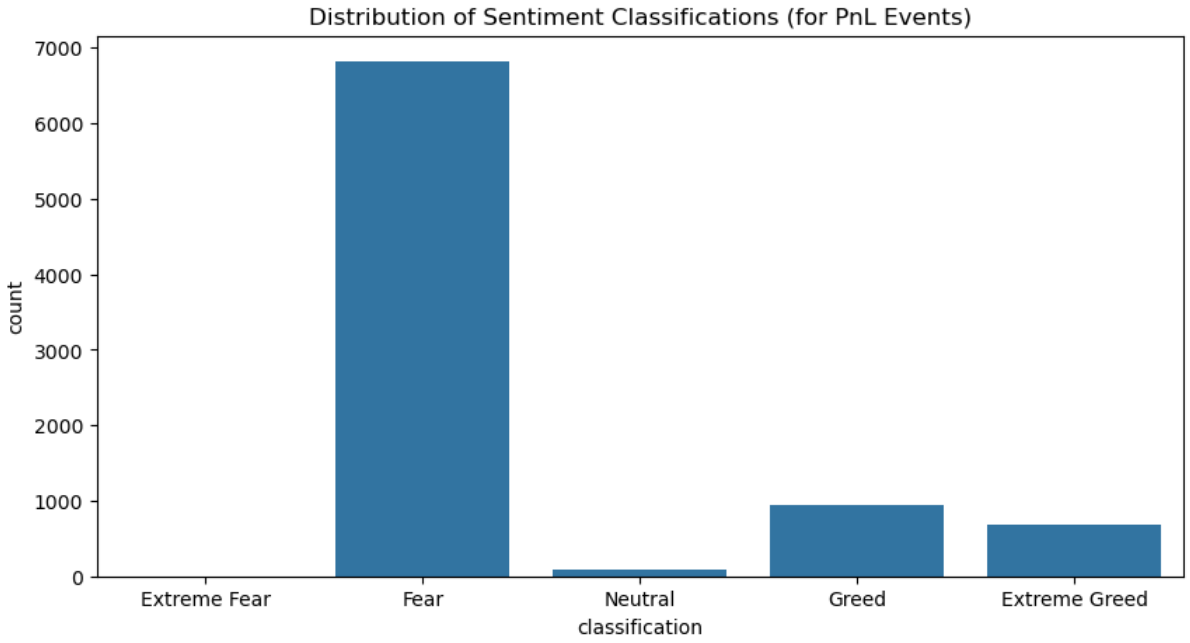
With the merged_df containing Bitcoin trades and associated daily market sentiment, this section focuses on Exploratory Data Analysis (EDA) to uncover relationships between trader performance/behavior and sentiment.

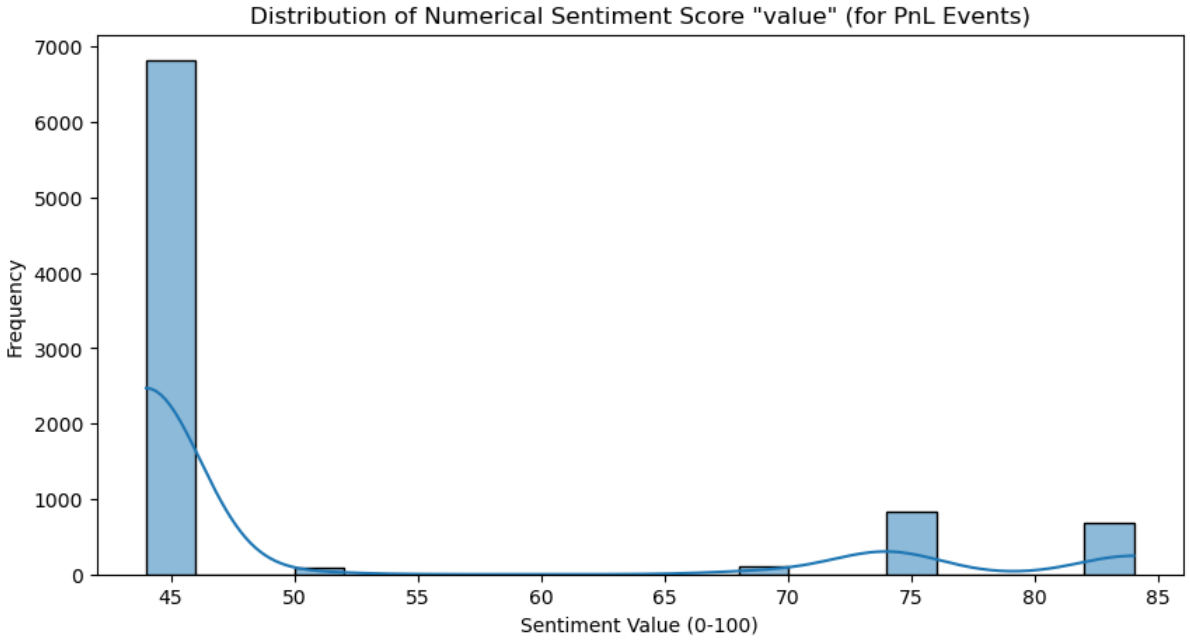
The initial steps in EDA involve:

1. **Creating an Analysis-Ready DataFrame:** Filtering merged_df to include only rows where sentiment data (classification and value) is available. This new DataFrame is named df_analysis.

2. **Focusing on PnL-Realizing Events:** Further filtering df_analysis to isolate events that directly result in a profit or loss (i.e., event_type values of 'Close Long', 'Close Short', or 'Liquidated Isolated Short'). This DataFrame is named df_pnl_events.
3. **Analyzing Sentiment Distribution:** Examining the distribution of sentiment classifications and numerical sentiment values for these PnL-realizing events to understand the market context of trades.
4. **Assessing Overall Trader Performance:** Calculating key performance indicators (total PnL, average PnL, win rate, etc.) for the df_pnl_events to establish a baseline before segmenting by sentiment.

--- Sentiment Distribution for PnL-Realizing Events ---





--- Overall Trader Performance (PnL Events with Sentiment) ---

Total Closed PnL: 710,690.32

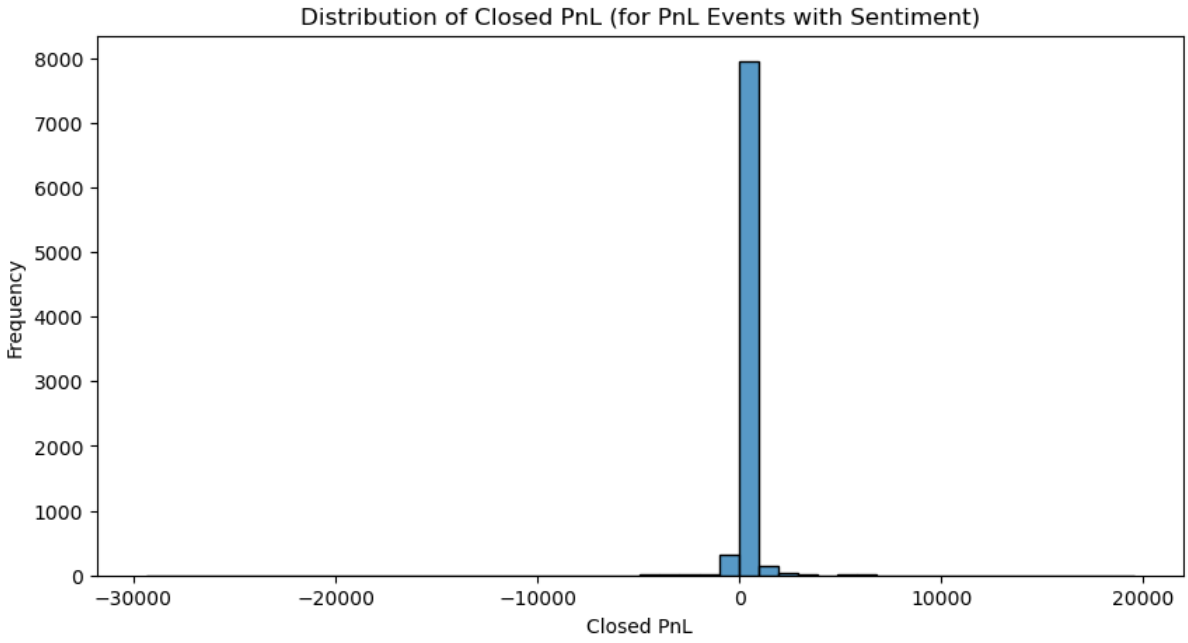
Number of PnL Events: 8524

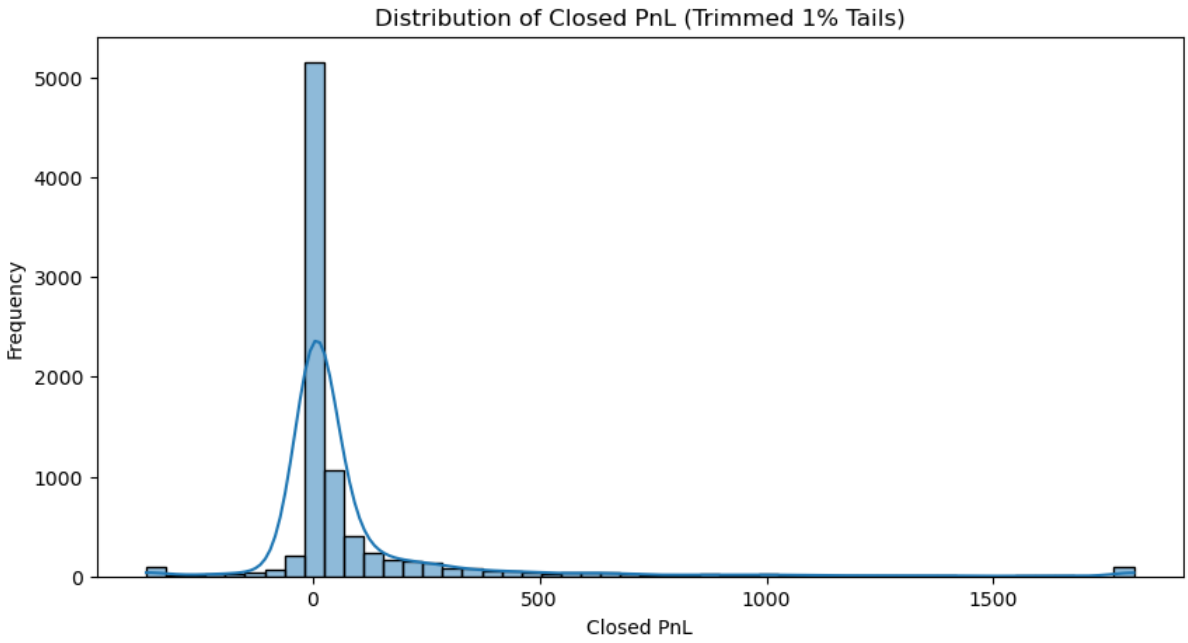
Average PnL per Event: 83.38

Median PnL per Event: 6.42

Standard Deviation of PnL: 708.85

Overall Win Rate: 86.65%



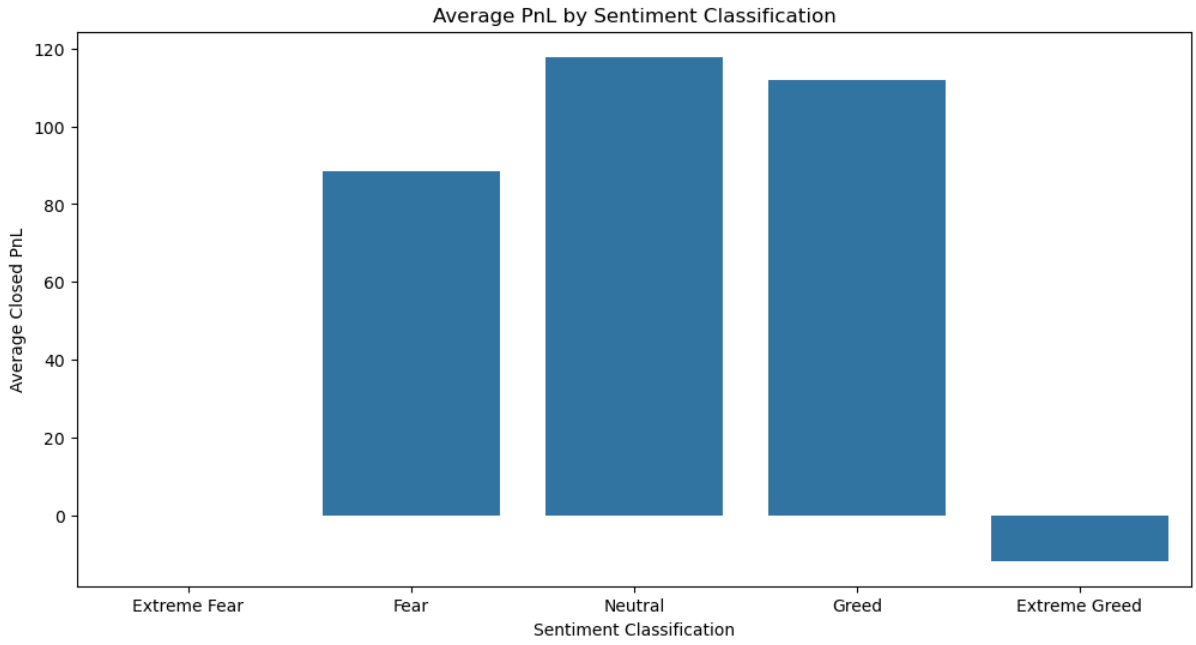


Trader Performance by Market Sentiment

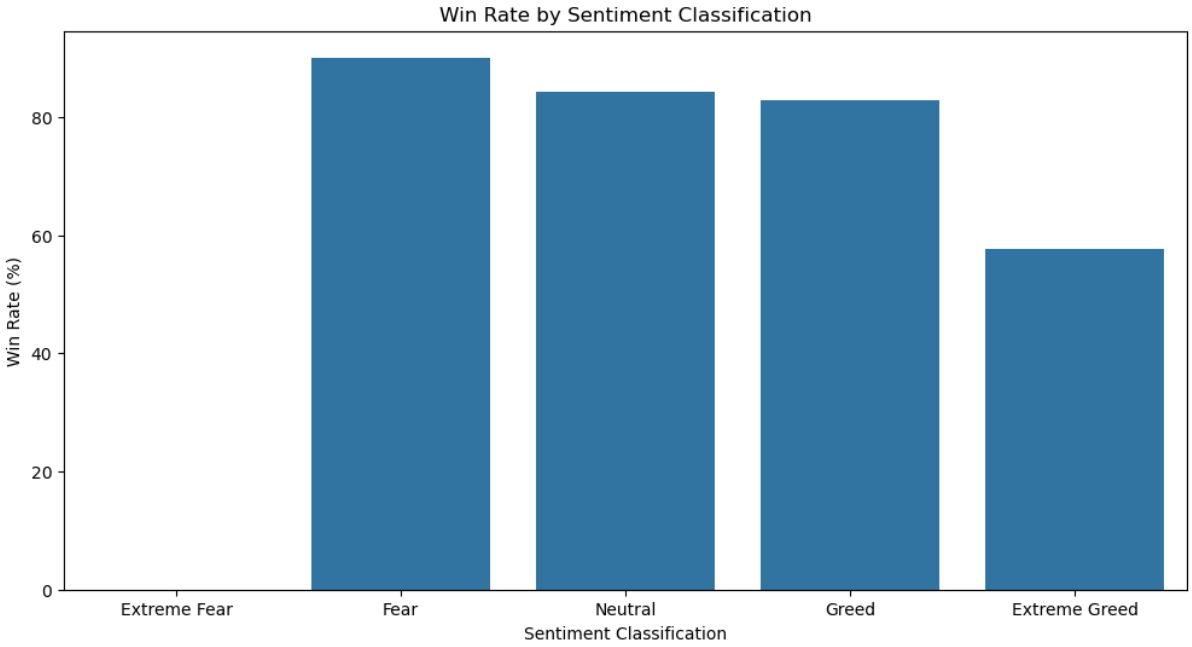
Having established an overall baseline, this section delves into how trader performance metrics (such as mean PnL, median PnL, and win rate) vary across different market sentiment states. The analysis is performed by:

1. Grouping PnL-realizing events (df_pnl_events) by the textual classification from the sentiment index.
2. Visualizing these grouped performance metrics.
3. Additionally, grouping PnL-realizing events by discretized bins of the numerical sentiment value (0-100 score) to see if a more granular view aligns with the textual classifications.

```
4. --- Performance by Sentiment Classification ---
5.   classification  count        sum_pnl    mean_pnl  median_pnl     std_pnl
   \
6. 1           Fear   6820  603545.456558   88.496401    7.074120  781.862429
7. 3        Neutral     89   10485.300779  117.812368    7.826946  357.297443
8. 2          Greed    935  104565.859356  111.835144   16.486635  278.706277
9. 0  Extreme Greed    680   -7906.299208  -11.626911    0.006058  184.221846
10.
11.     win_rate
12.1  90.087977
13.3  84.269663
14.2  82.887701
15.0  57.647059
```



16.

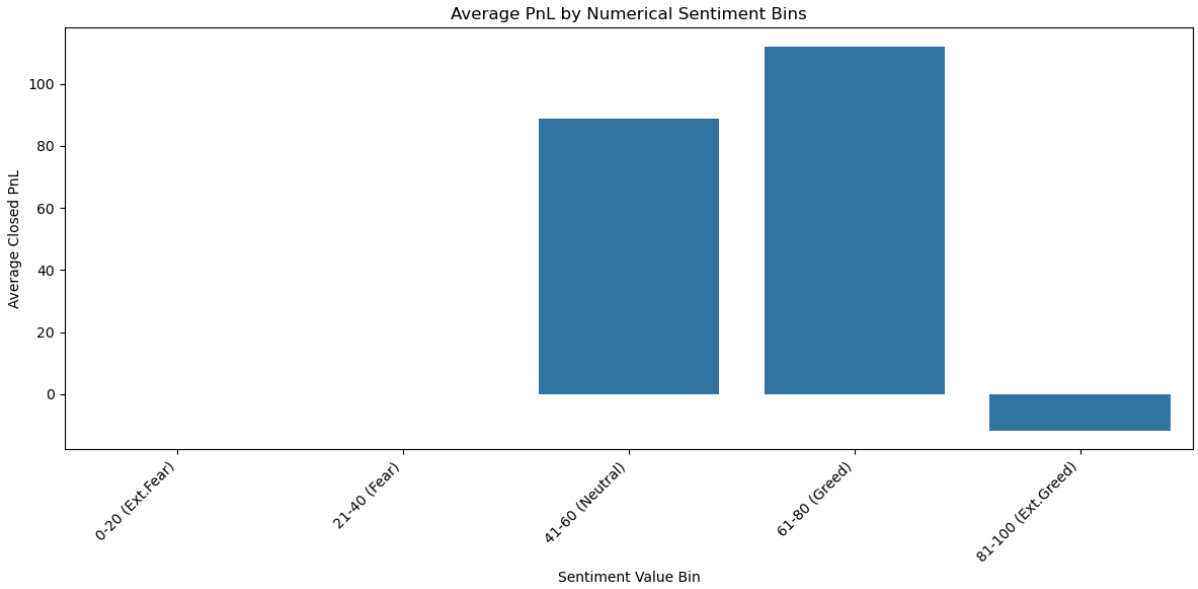


17.

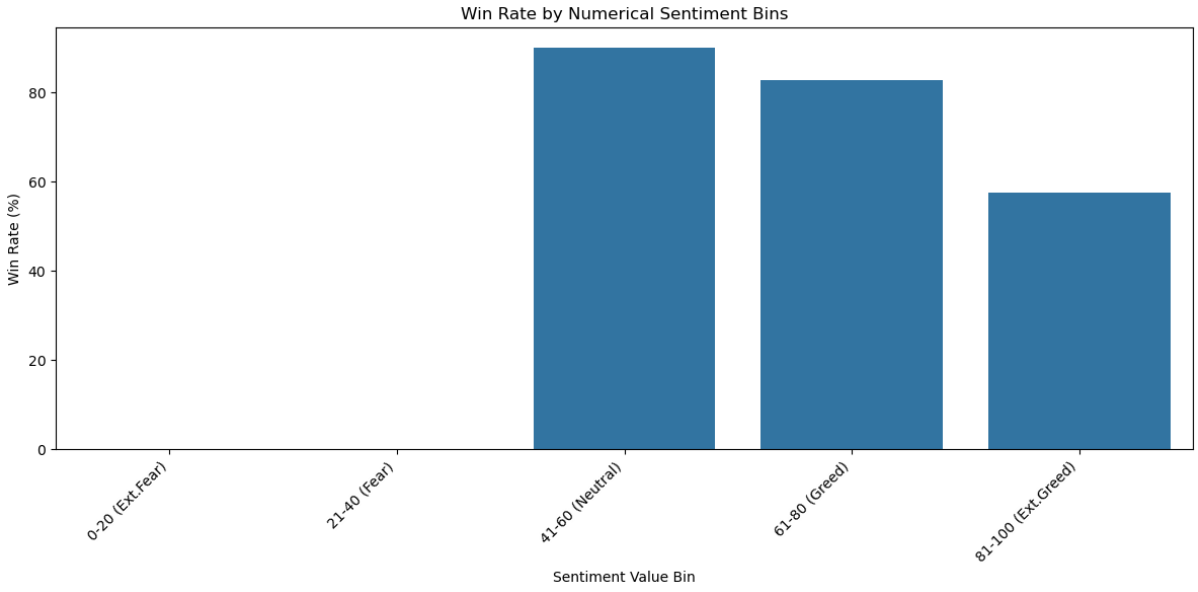
18.

19. **--- Performance by Numerical Sentiment 'value' (Binned) ---**

```
20.   sentiment_value_bin  count        sum_pnl    mean_pnl  median_pnl   win_r
    ate
21. 0     0-20 (Ext.Fear)      0       0.000000         NaN         NaN
    NaN
22. 1        21-40 (Fear)      0       0.000000         NaN         NaN
    NaN
23. 2     41-60 (Neutral)   6909  614030.757337   88.874042    7.080000  90.013
    026
24. 3       61-80 (Greed)    935  104565.859356  111.835144   16.486635  82.887
    701
25. 4  81-100 (Ext.Greed)    680   -7906.299208  -11.626911    0.006058  57.647
    059
26. C:\Users\PRAKASH SHARMA\AppData\Local\Temp\ipykernel_12688\900857246.py:54:
    FutureWarning: The default of observed=False is deprecated and will be chan
    ged to True in a future version of pandas. Pass observed=False to retain cu
    rrent behavior or observed=True to adopt the future default and silence thi
    s warning.
27.   performance_by_value_bin = df_pnl_events.groupby('sentiment_value_bin')['
    closed_pnl'].agg(
28. C:\Users\PRAKASH SHARMA\AppData\Local\Temp\ipykernel_12688\900857246.py:61:
    FutureWarning: The default of observed=False is deprecated and will be chan
    ged to True in a future version of pandas. Pass observed=False to retain cu
    rrent behavior or observed=True to adopt the future default and silence thi
    s warning.
29.   win_rate_by_value_bin = df_pnl_events.groupby('sentiment_value_bin')['is_
    win'].mean().reset_index()
```



30.



31.

32. ----------

33. **Interpretation of Trader Performance by Market Sentiment:**

34. The analysis of PnL-realizing events segmented by market sentiment reveals significant patterns:

35. **Performance by Sentiment Classification (Textual Labels):**

36. The table below summarizes key performance metrics for each sentiment category:

Classification	Count	Sum PnL	Mean PnL	Median PnL	Std Dev PnL	Win Rate (%)
Fear	6,820	+603,545.46	+88.50	+7.07	781.86	90.09%
Neutral	89	+10,485.30	+117.81	+7.83	357.30	84.27%
Greed	935	+104,565.86	+111.84	+16.49	278.71	82.89%

Classification	Count	Sum PnL	Mean PnL	Median PnL	Std Dev PnL	Win Rate (%)
Extreme Greed	680	-7,906.30	-11.63	+0.01	184.22	57.65%

--- Distribution of Numerical Sentiment 'value' for Trades Classified as 'Fear' ---

count 6820.0

mean 44.0

std 0.0

min 44.0

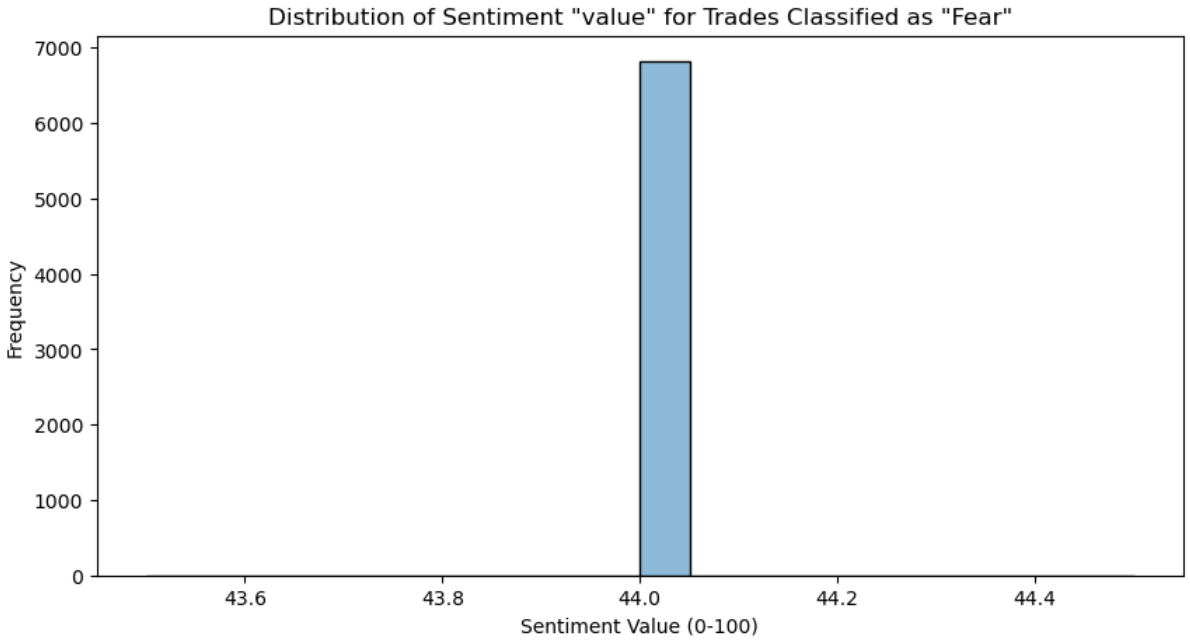
25% 44.0

50% 44.0

75% 44.0

max 44.0

Name: value, dtype: float64

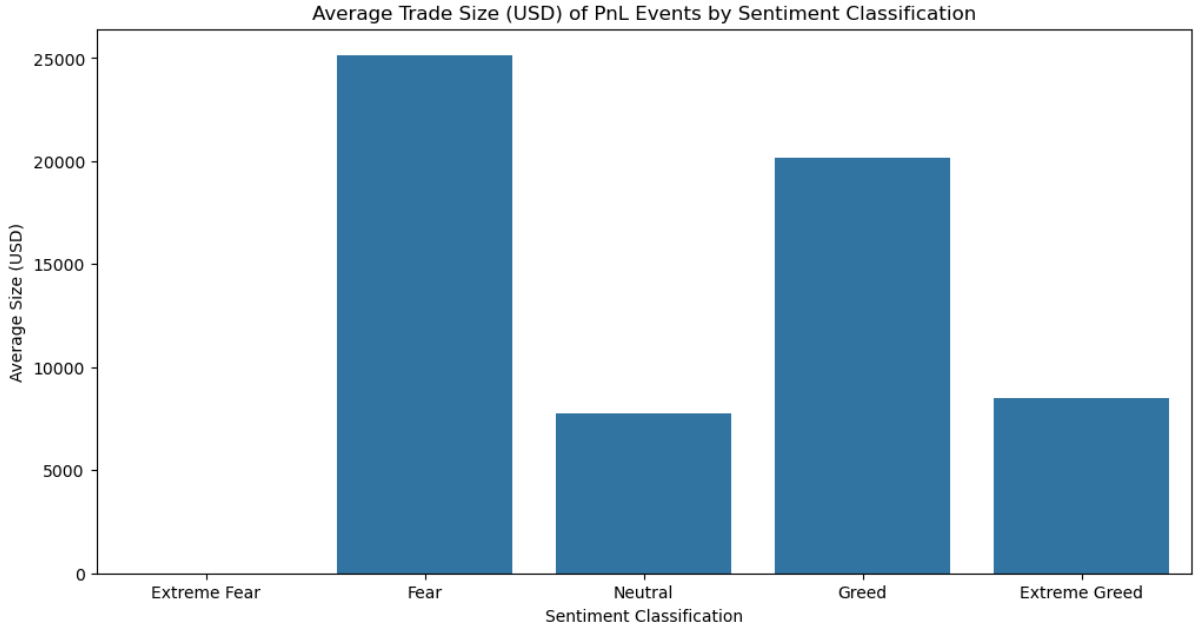


Trader Behavior by Market Sentiment

Beyond direct PnL performance, it's insightful to examine whether trader behavior—such as typical trade sizes or directional bias (long vs. short)—varies with market sentiment. This section explores:

1. **Average Trade Size:** The average and median trade size (in USD) for PnL-realizing events across different sentiment classifications.
2. **Directional Bias:** The frequency of different event_types, particularly 'Open Long' versus 'Open Short' events, for all Bitcoin trades (from df_analysis) across sentiment classifications. This helps understand if traders are more bullish or bearish in certain sentiment conditions.

```
--- Average Trade Size (USD) by Sentiment Classification (PnL Events) ---
  classification  mean_size_usd  median_size_usd  count
1           Fear   25154.366784         2336.815   6820
3        Neutral    7776.635506         2440.310     89
2          Greed   20150.710299         7398.950    935
0  Extreme Greed    8517.133956         1979.225    680
```



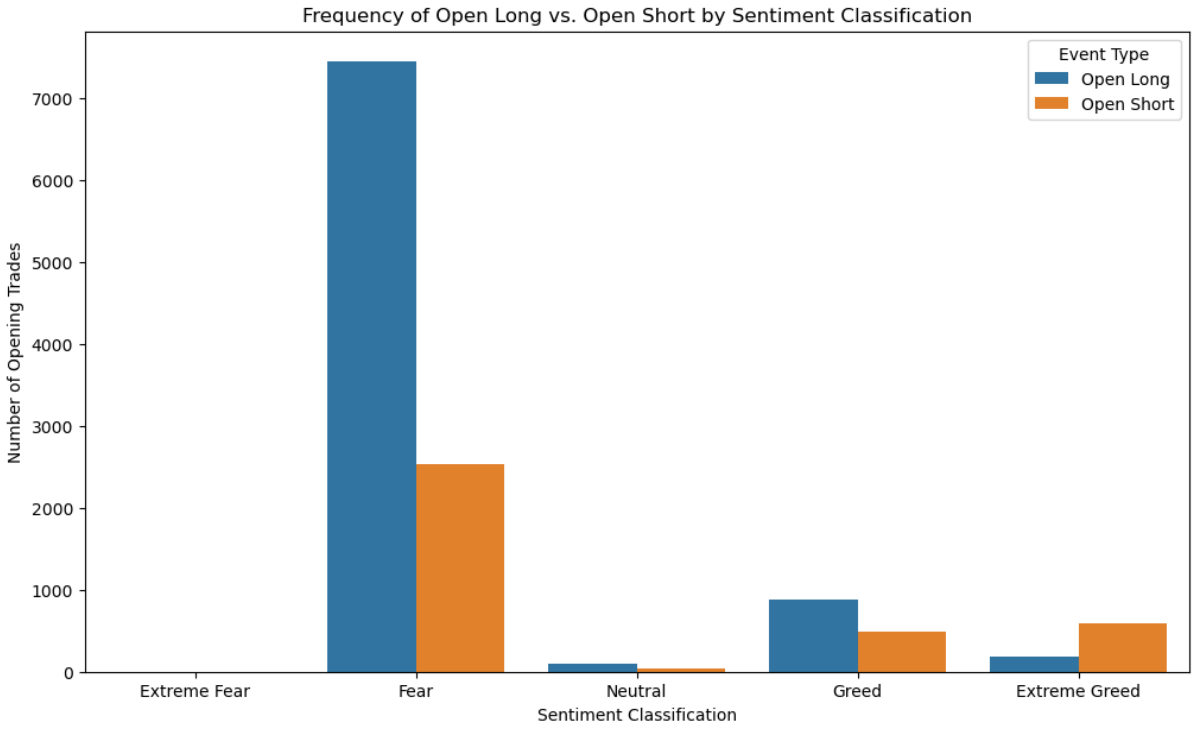

```

------------------------------------

--- Event Type Frequency by Sentiment Classification (All BTC Events with Sentiment) ---
Top event types per sentiment classification (all BTC events with sentiment):

Sentiment: Fear
event_type
Open Long       7439
Close Long      5067
Open Short      2526
Close Short     1753
Short > Long       5
Name: count, dtype: int64
```

```
23.
24. Sentiment: Greed
25. event_type
26. Open Long       881
27. Close Long      582
28. Open Short      482
29. Close Short     353
30. Short > Long      1
31. Name: count, dtype: int64
32.
33. Sentiment: Extreme Greed
34. event_type
35. Open Short      585
36. Close Short     517
37. Open Long       178
38. Close Long      163
39. Short > Long      4
40. Name: count, dtype: int64
41.
42. Sentiment: Neutral
43. event_type
44. Open Long      92
45. Close Long     70
46. Open Short     36
47. Close Short    19
48. Name: count, dtype: int64
```



49.