Name – Nilansh Kumar Singh

Assignment Overview

You will have to work with two primary datasets:

1. Bitcoin Market Sentiment Dataset o Columns: Date, Classification (Fear/Greed

2. Historical Trader Data from Hyperliquid o Columns include: account, symbol, execution price, size, side, time, start position, event, closedPnL, leverage, etc. Your objective is to explore the relationship between trader performance and market sentiment, uncover hidden patterns, and deliver insights that can drive smarter trading strategies.

PROBLEM STATEMENT: -

You will have to work with two primary datasets:

1. Bitcoin Market Sentiment Dataset
   - Columns: Date, Classification (Fear/Greed)
2. Historical Trader Data from Hyperliquid
   - Columns: account, symbol, execution price, size, side, time, start position, event, closedPnL, leverage, etc.

Your objective is to explore the relationship between trader performance and market sentiment, uncover hidden patterns, and deliver insights that can drive smarter trading strategies.

Datasets: -

You are given two datasets:

1. **Sentiment Data**

This tells you the emotion or mood of the crypto market on each day:

- Some days the market feels "Fear"
- Some days it's "Greed"

2. **Trader Data**

This contains **real trading activity**:

- What people **bought or sold**
- At what **price and size**
- Whether they made a **profit or loss (PnL)** on the trade

Goal: -

1. Find out if **market emotions (fear/greed)** affect how well traders perform.
2. Provide the Insight

APPROACH: -

STEP 1: Load the data

- Load both files:
    - **Sentiment data** (Fear/Greed)
    - **Trader data** (Buy/Sell history)

STEP 2: Clean the data

- Make sure the **dates** in both datasets are in the correct format (i.e., YYYY-MM-DD)
- Check for:
    - Missing values
    - Duplicates
    - Weird or incorrect entries

STEP 3: Connect both datasets

- Both files have a **"date" column**
- Use this column to **merge** the two datasets
    - This means: **For each trade**, attach the **sentiment of that day**

STEP 4: Analyze performance based on sentiment

STEP 5: Visualize

Use **matplotlib** or **seaborn** to create charts

STEP 6: Summarize your insights

```python
import pandas as pd
fear_greed_df = pd.read_csv('fear_greed_index.csv')
historical_df = pd.read_csv('historical_data.csv')
```

✓ 1.1s

```python
print("Fear & Greed Index Sample:")
display(fear_greed_df.head())

print("Historical Trader Data Sample:")
display(historical_df.head())
```

✓ 0.0s

```
Fear & Greed Index Sample:
```

	timestamp	value	classification	date
0	1517463000	30	Fear	2018-02-01
1	1517549400	15	Extreme Fear	2018-02-02
2	1517635800	40	Fear	2018-02-03
3	1517722200	24	Extreme Fear	2018-02-04
4	1517808600	11	Extreme Fear	2018-02-05

```
Historical Trader Data Sample:
```

	Account	Coin	Execution Price	Size Tokens	Size USD	Side	Timestamp IST	Start Position	Direction	Closed PnL	
0	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9769	986.87	7872.16	BUY	02-12-2024 22:50	0.000000	Buy	0.0	0xec09451986a1874e3a9
1	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9800	16.00	127.68	BUY	02-12-2024 22:50	986.524596	Buy	0.0	0xec09451986a1874e3a9
2	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9855	144.09	1150.63	BUY	02-12-2024 22:50	1002.518996	Buy	0.0	0xec09451986a1874e3a9
3	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9874	142.98	1142.04	BUY	02-12-2024 22:50	1146.558564	Buy	0.0	0xec09451986a1874e3a9
4	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9894	8.73	69.75	BUY	02-12-2024 22:50	1289.488521	Buy	0.0	0xec09451986a1874e3a9

```python
# 4. Check data types and missing values
print("Fear & Greed Index Info:")
fear_greed_df.info()
print("\nHistorical Trader Data Info:")
historical_df.info()
```

✓ 0.1s

```
Fear & Greed Index Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2644 entries, 0 to 2643
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   timestamp       2644 non-null   int64 
 1   value           2644 non-null   int64 
 2   classification  2644 non-null   object
 3   date            2644 non-null   object
dtypes: int64(2), object(2)
memory usage: 82.8+ KB

Historical Trader Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211224 entries, 0 to 211223
Data columns (total 16 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Account           211224 non-null  object 
 1   Coin              211224 non-null  object 
 2   Execution Price   211224 non-null  float64
 3   Size Tokens       211224 non-null  float64
 4   Size USD          211224 non-null  float64
 5   Side              211224 non-null  object 
...
 14  Trade ID          211224 non-null  float64
 15  Timestamp         211224 non-null  float64
dtypes: bool(1), float64(8), int64(1), object(6)
memory usage: 24.4+ MB
```

```python
fear_greed_df['date'] = pd.to_datetime(fear_greed_df['date'], errors='coerce', dayfirst=True)    "dayfirst": Unknown word.
```
✓ 0.0s

```python
print("Unique classifications in Fear & Greed Index:")
print(fear_greed_df['classification'].unique())

# Check for duplicates
print("Duplicates in Fear & Greed Index:", fear_greed_df.duplicated().sum())
```
✓ 0.0s

```
Unique classifications in Fear & Greed Index:
['Fear' 'Extreme Fear' 'Neutral' 'Greed' 'Extreme Greed']
Duplicates in Fear & Greed Index: 0
```

```python
if pd.api.types.is_numeric_dtype(historical_df['Timestamp']):
    # Try milliseconds
    historical_df['Timestamp'] = pd.to_datetime(historical_df['Timestamp'], unit='ms', errors='coerce')
else:
    historical_df['Timestamp'] = pd.to_datetime(historical_df['Timestamp'], errors='coerce')
```
✓ 0.0s

```python
# Check for missing values
print("\nMissing values in Historical Trader Data:")
print(historical_df.isnull().sum())

# Show column names for reference
print("\nHistorical Trader Data columns:")
print(historical_df.columns)
```
✓ 0.1s

```
Missing values in Historical Trader Data:
Account             0
Coin                0
Execution Price     0
Size Tokens         0
Size USD            0
Side                0
Timestamp IST       0
Start Position      0
Direction           0
Closed PnL          0
Transaction Hash    0
Order ID            0
Crossed             0
Fee                 0
Trade ID            0
Timestamp           0
dtype: int64

Historical Trader Data columns:
Index(['Account', 'Coin', 'Execution Price', 'Size Tokens', 'Size USD', 'Side',
       'Timestamp IST', 'Start Position', 'Direction', 'Closed PnL',
       'Transaction Hash', 'Order ID', 'Crossed', 'Fee', 'Trade ID',
       'Timestamp'],
      dtype='object')
```

```python
# Date ranges
print("Fear & Greed Index date range:", fear_greed_df['date'].min(), "to", fear_greed_df['date'].max())
print("Historical Data date range:", historical_df['Timestamp'].min(), "to", historical_df['Timestamp'].max())

# Unique values
print("\nUnique accounts:", historical_df['Account'].nunique())
print("Unique coins:", historical_df['Coin'].nunique())
print("Unique sides:", historical_df['Side'].unique())

# Distribution of sentiment
print("\nFear & Greed Index distribution:")
print(fear_greed_df['classification'].value_counts())
```

✓ 0.1s

```
Fear & Greed Index date range: 2018-01-02 00:00:00 to 2025-12-04 00:00:00
Historical Data date range: 2023-03-28 10:40:00 to 2025-06-15 15:06:40

Unique accounts: 32
Unique coins: 246
Unique sides: ['BUY' 'SELL']

Fear & Greed Index distribution:
classification
Fear             781
Greed            633
Extreme Fear     508
Neutral          396
Extreme Greed    326
Name: count, dtype: int64
```

```python
# --- Prepare for merging ---
# Add a 'date' column to historical_df to match with fear_greed_df
historical_df['date'] = historical_df['Timestamp'].dt.normalize()  # sets time to 00:00:00

# Check a sample
display(historical_df[['Timestamp', 'date']].head())
```

✓ 0.0s

	Timestamp	date
0	2024-10-27 03:33:20	2024-10-27
1	2024-10-27 03:33:20	2024-10-27
2	2024-10-27 03:33:20	2024-10-27
3	2024-10-27 03:33:20	2024-10-27
4	2024-10-27 03:33:20	2024-10-27

```python
# Merge historical trader data with fear & greed index on 'date'
merged_df = pd.merge(
    historical_df,
    fear_greed_df[['date', 'classification', 'value']],  # Only bring in relevant columns
    on='date',
    how='left'  # Use 'left' to keep all trades, even if some dates have no sentiment data
)
```

✓ 0.1s

```python
# Check the result
print("Merged Data Sample:")
display(merged_df.head())

# Check for any trades with missing sentiment data
missing_sentiment = merged_df['classification'].isna().sum()
print(f"Number of trades with missing sentiment data: {missing_sentiment}")
```

✓ 0.0s Python

```
Merged Data Sample:
```

	Account	Coin	Execution Price	Size Tokens	Size USD	Side	Timestamp IST	Start Position	Direction	Closed PnL	
0	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9769	986.87	7872.16	BUY	02-12-2024 22:50	0.000000	Buy	0.0	0xec09451986a1874e3a9
1	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9800	16.00	127.68	BUY	02-12-2024 22:50	986.524596	Buy	0.0	0xec09451986a1874e3a9
2	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9855	144.09	1150.63	BUY	02-12-2024 22:50	1002.518996	Buy	0.0	0xec09451986a1874e3a9
3	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9874	142.98	1142.04	BUY	02-12-2024 22:50	1146.558564	Buy	0.0	0xec09451986a1874e3a9
4	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9894	8.73	69.75	BUY	02-12-2024 22:50	1289.488521	Buy	0.0	0xec09451986a1874e3a9

```
Number of trades with missing sentiment data: 197121
```


```
Number of trades with missing sentiment data: 197121
```


```python
print("Fear & Greed Index date range:", fear_greed_df['date'].min(), "to", fear_greed_df['date'].max())
print("Historical Data date range:", historical_df['date'].min(), "to", historical_df['date'].max())
```

✓ 0.0s

```
Fear & Greed Index date range: 2018-01-02 00:00:00 to 2025-12-04 00:00:00
Historical Data date range: 2023-03-28 00:00:00 to 2025-06-15 00:00:00
```

```python
# Find the overlap
min_sentiment_date = fear_greed_df['date'].min()
max_sentiment_date = fear_greed_df['date'].max()
```

✓ 0.0s

```python
filtered_trades = historical_df[
    (historical_df['date'] >= min_sentiment_date) &
    (historical_df['date'] <= max_sentiment_date)
]
```

✓ 0.0s

```python
merged_df = pd.merge(
    filtered_trades,
    fear_greed_df[['date', 'classification', 'value']],
    on='date',
    how='left'
)
```

✓ 0.0s

```python
print("Number of trades in overlap:", len(merged_df))
print("Sample after filtering:")
display(merged_df.head())
```

✓ 0.0s Python

```
Number of trades in overlap: 211224
Sample after filtering:
```

	Account	Coin	Execution Price	Size Tokens	Size USD	Side	Timestamp IST	Start Position	Direction	Closed PnL	
0	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9769	986.87	7872.16	BUY	02-12-2024 22:50	0.000000	Buy	0.0	0xec09451986a1874e3a9
1	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9800	16.00	127.68	BUY	02-12-2024 22:50	986.524596	Buy	0.0	0xec09451986a1874e3a9
2	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9855	144.09	1150.63	BUY	02-12-2024 22:50	1002.518996	Buy	0.0	0xec09451986a1874e3a9
3	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9874	142.98	1142.04	BUY	02-12-2024 22:50	1146.558564	Buy	0.0	0xec09451986a1874e3a9
4	0xae5eacaf9c6b9111fd53034a602c192a04e082ed	@107	7.9894	8.73	69.75	BUY	02-12-2024 22:50	1289.488521	Buy	0.0	0xec09451986a1874e3a9

```python
print("Fear & Greed Index date range:", fear_greed_df['date'].min(), "to", fear_greed_df['date'].max())
print("Historical Data date range:", historical_df['date'].min(), "to", historical_df['date'].max())
```

✓ 0.0s Python

```
Fear & Greed Index date range: 2018-01-02 00:00:00 to 2025-12-04 00:00:00
Historical Data date range: 2023-03-28 00:00:00 to 2025-06-15 00:00:00
```

```python
print("fear_greed_df['date'] dtype:", fear_greed_df['date'].dtype)
print("historical_df['date'] dtype:", historical_df['date'].dtype)
print("Sample dates in fear_greed_df:", fear_greed_df['date'].sort_values().unique()[:5])
print("Sample dates in historical_df:", historical_df['date'].sort_values().unique()[:5])
```

✓ 0.0s

```
fear_greed_df['date'] dtype: datetime64[ns]
historical_df['date'] dtype: datetime64[ns]
Sample dates in fear_greed_df: <DatetimeArray>
['2018-01-02 00:00:00', '2018-01-03 00:00:00', '2018-01-04 00:00:00',
 '2018-01-05 00:00:00', '2018-01-06 00:00:00']
Length: 5, dtype: datetime64[ns]
Sample dates in historical_df: <DatetimeArray>
['2023-03-28 00:00:00', '2023-11-14 00:00:00', '2024-03-09 00:00:00',
 '2024-07-03 00:00:00', '2024-10-27 00:00:00']
Length: 5, dtype: datetime64[ns]
```

```python
# Normalize both to remove time component
fear_greed_df['date'] = pd.to_datetime(fear_greed_df['date']).dt.normalize()
historical_df['date'] = pd.to_datetime(historical_df['date']).dt.normalize()
```

✓ 0.0s

```python
merged_df = pd.merge(
    historical_df,
    fear_greed_df[['date', 'classification', 'value']],
    on='date',
    how='left'
)

print("Number of trades with sentiment data:", merged_df['classification'].notna().sum())
print("Sample with sentiment data:")
display(merged_df[merged_df['classification'].notna()].head())
```

✓ 0.1s Pytho

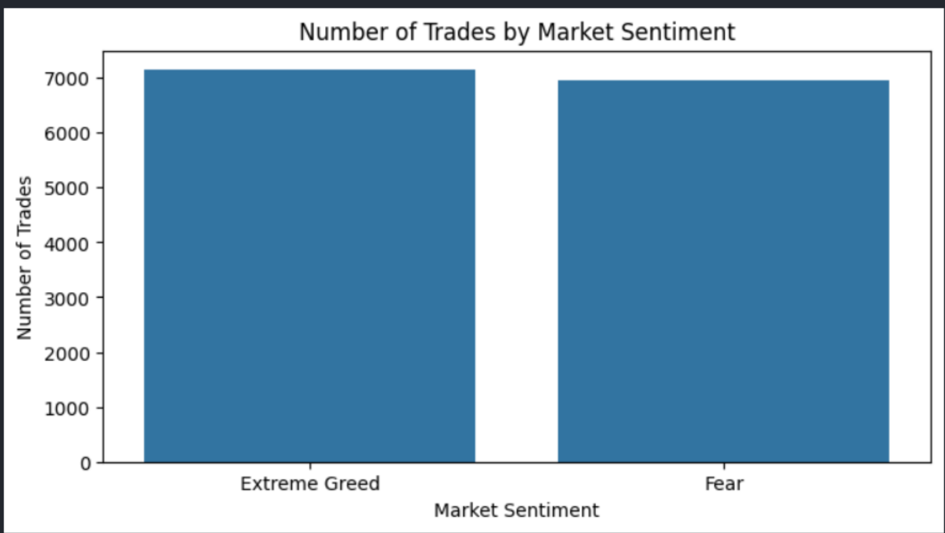
```
Number of trades with sentiment data: 14103
Sample with sentiment data:
```

	Account	Coin	Execution Price	Size Tokens	Size USD	Side	Timestamp IST	Start Position	Direction	Closed PnL	
18047	0x430f09841d65beb3f27765503d0f850b8bce7713	PURR/USDC	0.13097	22382.0	2931.37	BUY	20-04-2024 12:28	0.0	Buy	0.0	0xbe658417a7c
18048	0x430f09841d65beb3f27765503d0f850b8bce7713	PURR/USDC	0.13100	447.0	58.56	BUY	20-04-2024 12:28	22374.0	Buy	0.0	0xbe658417a7c
18049	0x430f09841d65beb3f27765503d0f850b8bce7713	PURR/USDC	0.13100	503.0	65.89	BUY	20-04-2024 12:28	22821.0	Buy	0.0	0xb2429810a
18050	0x430f09841d65beb3f27765503d0f850b8bce7713	PURR/USDC	0.13100	39139.0	5127.21	BUY	20-04-2024 12:28	23323.0	Buy	0.0	0x5fdfde429fa
18051	0x430f09841d65beb3f27765503d0f850b8bce7713	PURR/USDC	0.13100	726.0	95.11	BUY	20-04-2024 12:28	62459.0	Buy	0.0	0xb4b03e0b0

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Count of trades by sentiment classification
plt.figure(figsize=(8,4))    "figsize": Unknown word.
sns.countplot(data=merged_df, x='classification', order=merged_df['classification'].value_counts().index)
plt.title('Number of Trades by Market Sentiment')
plt.xlabel('Market Sentiment')    "xlabel": Unknown word.
plt.ylabel('Number of Trades')    "ylabel": Unknown word.
plt.show()
```

✓ 6.1s



Bar chart that shows how many trades occurred under each type of market sentiment, such as Fear, Greed, Extreme Fear, and so on.

It uses the merged dataset, which includes trading data along with the sentiment classification for each trade date. The x-axis of the chart represents the different sentiment categories, while the y-axis shows the number of trades made on days when that sentiment was recorded. By looking at the height of each bar, we can say which types of sentiment had more or fewer trades associated with them. It helps identify whether traders were more active during times of fear, greed, or neutrality in the market.

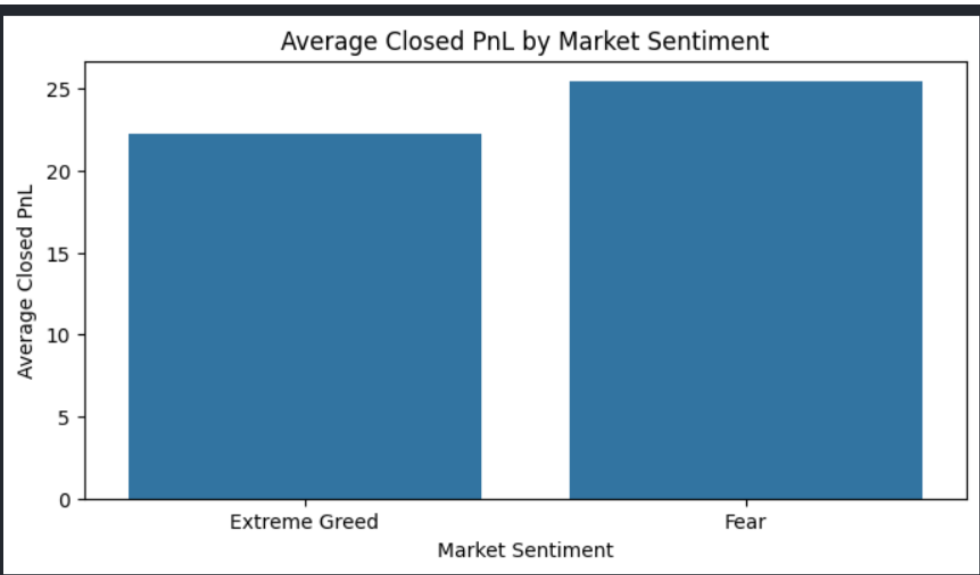
```
# Ensure Closed PnL is numeric
merged_df['Closed PnL'] = pd.to_numeric(merged_df['Closed PnL'], errors='coerce')

# Group by sentiment and calculate mean Closed PnL
performance_by_sentiment = merged_df.groupby('classification')['Closed PnL'].mean().sort_values()
print(performance_by_sentiment)

# Visualize
plt.figure(figsize=(8,4))
sns.barplot(x=performance_by_sentiment.index, y=performance_by_sentiment.values)
plt.title('Average Closed PnL by Market Sentiment')
plt.xlabel('Market Sentiment')
plt.ylabel('Average Closed PnL')
plt.show()
```

✓ 0.2s

```
classification
Extreme Greed     22.229713
Fear              25.418772
Name: Closed PnL, dtype: float64
```




```
# Calculate win rate (percentage of trades with positive Closed PnL) by sentiment
win_rate = merged_df.groupby('classification')['Closed PnL'].apply(lambda x: (x > 0).mean() * 100)
print(win_rate)
```

✓ 0.0s

```
classification
Extreme Greed     31.718247
Fear              49.008905
Name: Closed PnL, dtype: float64
```

We Calculates and visualizes the average profit or loss made by traders under different market sentiment conditions like "Fear", "Greed", etc.

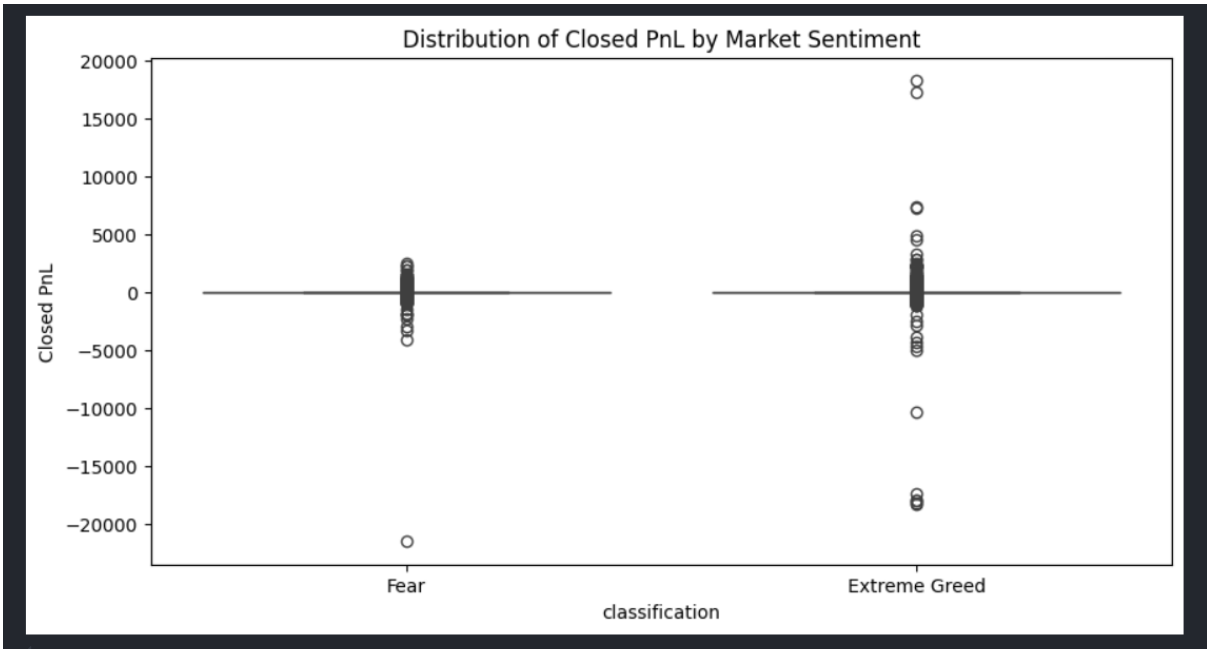
```python
# Total USD traded by sentiment
volume_by_sentiment = merged_df.groupby('classification')['Size USD'].sum()
print(volume_by_sentiment)
```

✓ 0.0s

```
classification
Extreme Greed     21843234.35
Fear              39406770.25
Name: Size USD, dtype: float64
```

```python
plt.figure(figsize=(10,5))    "figsize": Unknown word.
sns.boxplot(data=merged_df, x='classification', y='Closed PnL')
plt.title('Distribution of Closed PnL by Market Sentiment')
plt.show()
```

✓ 0.5s



It creates a box plot that shows the distribution of trader profits and losses for each market sentiment category like Fear, Greed.

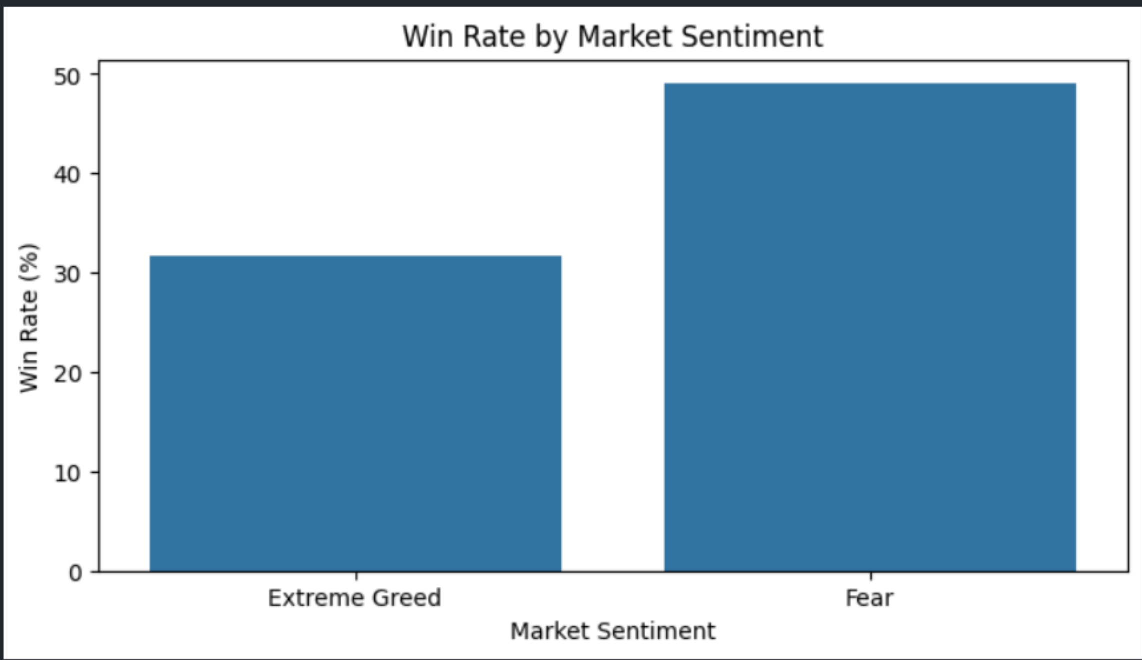
Traders seem to perform better or take more risks when the market is fearful, and make less or play safer when the market is greedy. This could help traders decide when to be aggressive and when to be careful, depending on the market mood.

```
Click to add a breakpoint e (percentage of trades with positive Closed PnL) by sentiment
win_rate = merged_df.groupby('classification')['Closed PnL'].apply(lambda x: (x > 0).mean() * 100)
print("Win Rate by Sentiment (%):")
print(win_rate)

# Visualize
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,4))    "figsize": Unknown word.
sns.barplot(x=win_rate.index, y=win_rate.values)    "barplot": Unknown word.
plt.title('Win Rate by Market Sentiment')
plt.xlabel('Market Sentiment')    "xlabel": Unknown word.
plt.ylabel('Win Rate (%)')    "ylabel": Unknown word.
plt.show()
```
✓ 0.2s

```
Win Rate by Sentiment (%):
classification
Extreme Greed    31.718247
Fear             49.008905
Name: Closed PnL, dtype: float64
```



This code calculates how often traders made a profit (Closed PnL > 0) under each market sentiment condition like Fear, Greed.

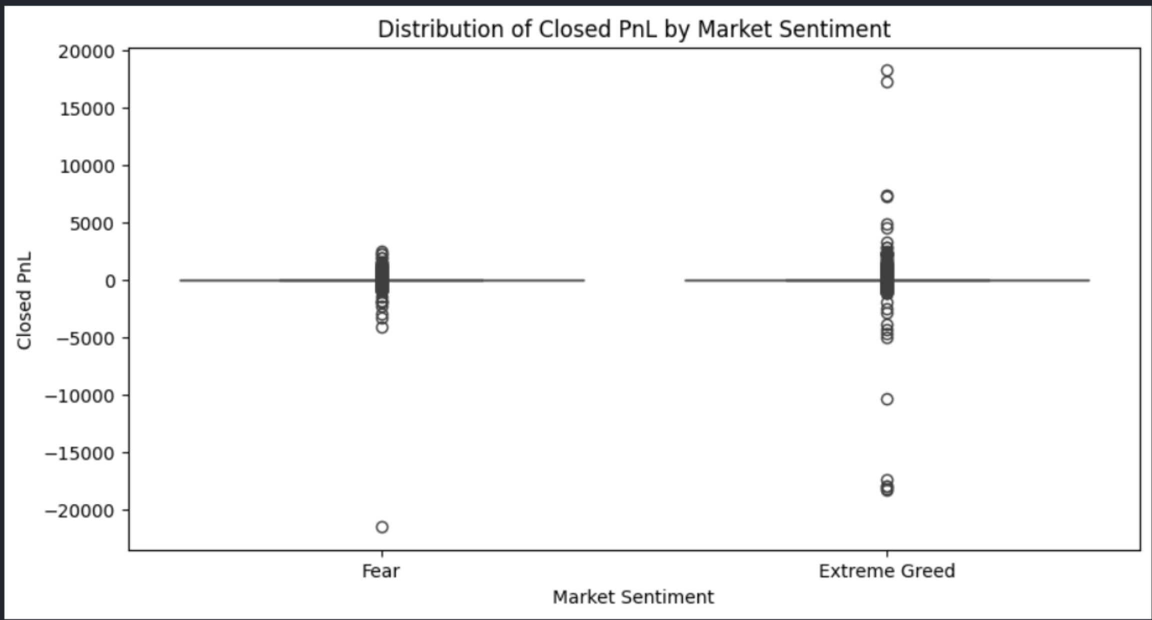
If the bar for Fear is higher than Extreme Greed, it means traders had a better chance of making money when the market was fearful than when it was greedy.

Traders may perform better or find more profitable opportunities during fearful markets compared to greedy ones.

So, fear in the market might be a better time to trade, while greedy times may carry more risk or fewer profits.

```python
plt.figure(figsize=(10,5))
sns.boxplot(data=merged_df, x='classification', y='Closed PnL')
plt.title('Distribution of Closed PnL by Market Sentiment')
plt.xlabel('Market Sentiment')
plt.ylabel('Closed PnL')
plt.show()
```

✓ 0.5s



We can understand:

In market mood traders had more stable or more profitable outcomes, and where they faced more losses or unpredictability.

If the Fear boxplot shows a higher median and tighter spread than Extreme Greed, it means:

Traders had better and more consistent profits during fearful times compared to greedy times.

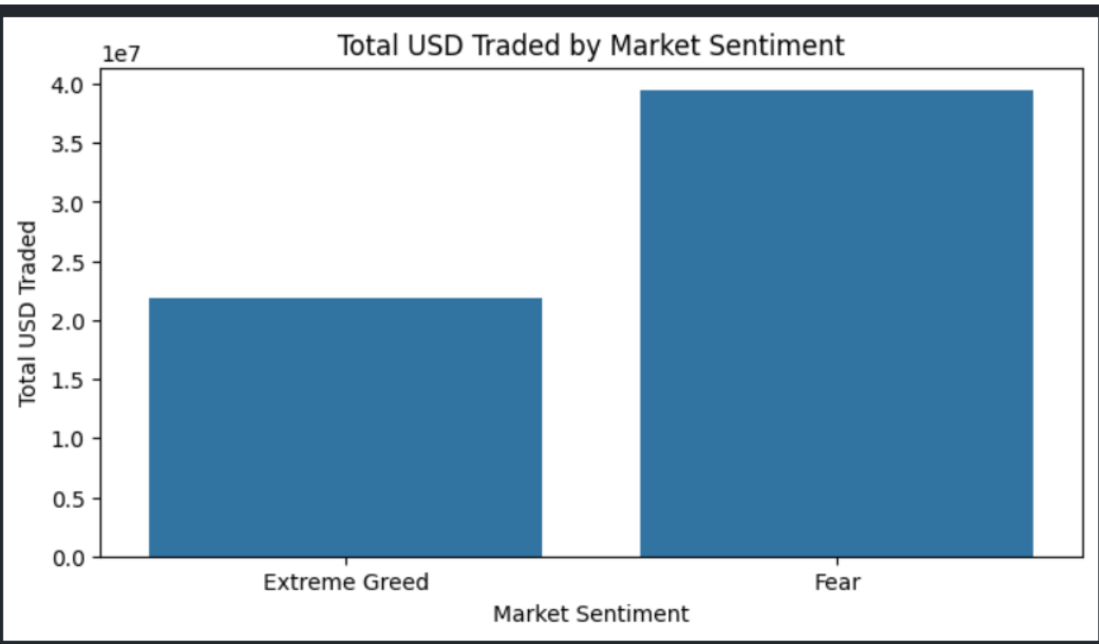
This helps us see not just how often traders win, but also how big or small their profits and losses are, depending on the market sentiment.

```python
volume_by_sentiment = merged_df.groupby('classification')['Size USD'].sum()
print("Total USD Traded by Sentiment:")
print(volume_by_sentiment)

plt.figure(figsize=(8,4))
sns.barplot(x=volume_by_sentiment.index, y=volume_by_sentiment.values)
plt.title('Total USD Traded by Market Sentiment')
plt.xlabel('Market Sentiment')
plt.ylabel('Total USD Traded')
plt.show()
```

✓ 0.2s

```
Total USD Traded by Sentiment:
classification
Extreme Greed    21843234.35
Fear             39406770.25
Name: Size USD, dtype: float64
```




```python
summary = merged_df.groupby('classification').agg(
    trade_count=('Closed PnL', 'count'),
    avg_pnl=('Closed PnL', 'mean'),
    median_pnl=('Closed PnL', 'median'),
    win_rate=('Closed PnL', lambda x: (x > 0).mean() * 100),
    total_usd_traded=('Size USD', 'sum')
)
print(summary)
```

✓ 0.0s

We analyse how much total trading volume (in USD) happened during different market sentiments (like Fear or Greed).

```
               trade_count    avg_pnl  median_pnl   win_rate  \
classification
Extreme Greed         7141  22.229713         0.0  31.718247
Fear                  6962  25.418772         0.0  49.008905

               total_usd_traded
classification
Extreme Greed       21843234.35
Fear                39406770.25
```