Name:Pravallika Kuchika,

Web 3 trading Assignment Overview 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.

Objective is to explore the relationship between trader performance and market sentiment, uncover hidden patterns, and deliver insights that can drive smarter trading strategies. Drive Links for the Datasets:

Historical Data:https://drive.google.com/file/d/1yvNUiAk4mKZHxplWI2xIL5c5ozvr-R8L/view?usp=drive_link

Feargreedindex:https://drive.google.com/file/d/1UhNGmLqwSc-bFxt38XreNyTCo2qWwCTU/view?usp=drive_link

Short guidance for Analysis:- Sentiment vs profitability: compare daily market profit rates during Fear vs Greed.

Leverage behavior: do traders use higher leverage during Greed? Does higher leverage reduce win rate?

Volume shifts: is notional/volume higher during Greed — any correlation with volatility?

Survivorship & tail risks: analyze extreme losses by account and whether these cluster by sentiment.

Lagged effects: compute sentiment vs profitability to find predictive signals.

Sub-populations: separate by symbol, by account activity level, by side.

```
!pip install --quiet plotly scikit-learn

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
import os

sentiment_path = '/content/Feargreedindex.csv'
trades_path = '/content/historicaldata.csv'


sent = pd.read_csv(sentiment_path)
trades = pd.read_csv(trades_path)

sent['date'] = pd.to_datetime(sent['date'], errors='coerce')
trades['Timestamp'] = pd.to_datetime(trades['Timestamp'],
errors='coerce')
```

```python
# Quick heads-up: ensure column names exactly match; use
trades.columns to inspect
print("Columns in sentiment data:")
print(sent.columns)
print("\nColumns in trades data:")
print(trades.columns)
print(sent.head())
print(trades.head())
```

```
Columns in sentiment data:
Index(['timestamp', 'value', 'classification', 'date'],
dtype='object')

Columns in trades data:
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')
    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
                                      Account  Coin  Execution
Price  \
0  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9769

1  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9800

2  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9855

3  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9874

4  0xae5eacaf9c6b9111fd53034a602c192a04e082ed  @107           7.9894


   Size Tokens  Size USD Side     Timestamp IST  Start Position
Direction  \
0       986.87   7872.16  BUY  02-12-2024 22:50        0.000000
Buy
1        16.00    127.68  BUY  02-12-2024 22:50      986.524596
Buy
2       144.09   1150.63  BUY  02-12-2024 22:50     1002.518996
Buy
3       142.98   1142.04  BUY  02-12-2024 22:50     1146.558564
```

```
Buy
4         8.73     69.75  BUY  02-12-2024 22:50     1289.488521
Buy

   Closed PnL                                  Transaction Hash
Order ID  \
0         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...
52017706630
1         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...
52017706630
2         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...
52017706630
3         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...
52017706630
4         0.0  0xec09451986a1874e3a980418412fcd0201f500c95bac...
52017706630

   Crossed       Fee      Trade ID           Timestamp
0     True  0.345404  8.950000e+14 1970-01-01 00:28:50
1     True  0.005600  4.430000e+14 1970-01-01 00:28:50
2     True  0.050431  6.600000e+14 1970-01-01 00:28:50
3     True  0.050043  1.080000e+15 1970-01-01 00:28:50
4     True  0.003055  1.050000e+15 1970-01-01 00:28:50

# Standardize sentiment labels
sent['classification'] =
sent['classification'].str.strip().str.title()
# Convert trades timestamp to date and then to datetime objects for
merging
trades['date'] = pd.to_datetime(trades['Timestamp'].dt.date)

# Merge daily sentiment to trades by date
trades = trades.merge(sent[['date','classification']], on='date',
how='left')

# Example features
trades['notional'] = trades['Execution Price'] * trades['Size USD']
trades['profit_flag'] = (trades['Closed PnL'] > 0).astype(int)

# Convert trades timestamp to date and then to datetime objects for
merging
trades['date'] = pd.to_datetime(trades['Timestamp'].dt.date)

# Daily market-level summary
daily = trades.groupby('date').agg(
total_trades=('Account','count'),
total_notional=('notional','sum'),
profit_rate=('profit_flag','mean')
).reset_index()
```

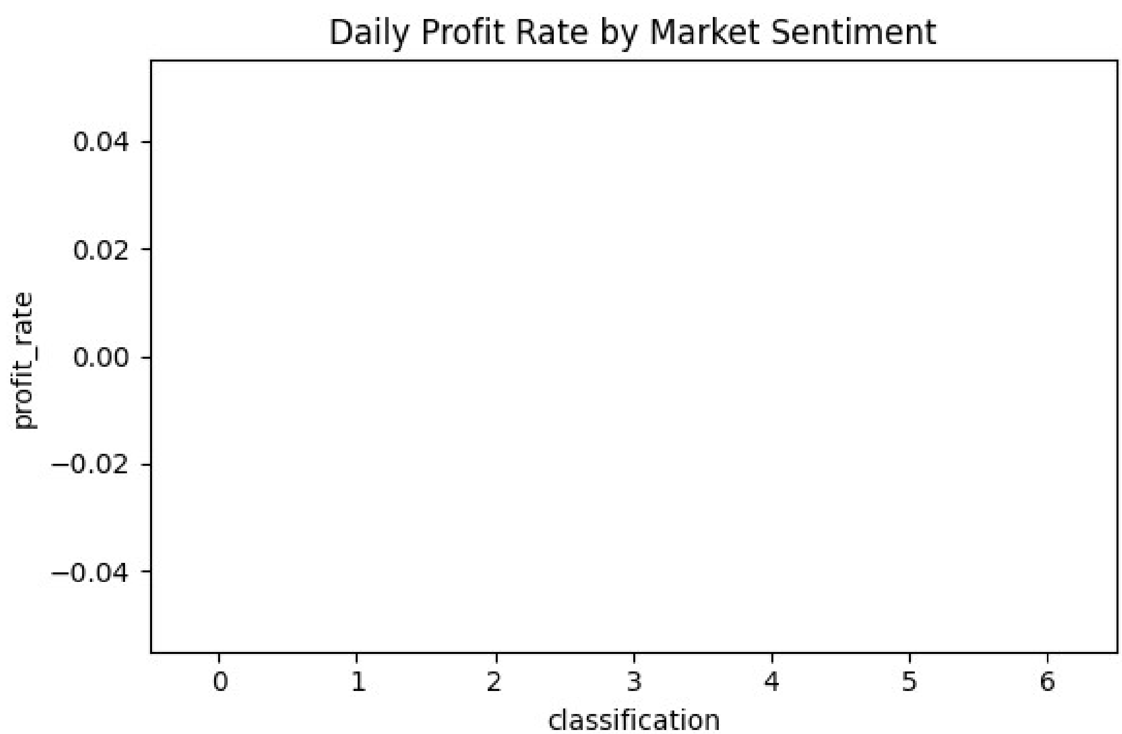
```python
# Attach sentiment
daily = daily.merge(sent[['date','classification']], on='date',
how='left')


# Per-account summary
acct = trades.groupby('Account').agg(
trades_count=('Account','count'),
win_rate=('profit_flag','mean'),
avg_notional=('notional','mean'),
pnl_sum=('Closed PnL','sum')
).reset_index()

import os

# Create the output directory if it doesn't exist
output_dir = '/content/outputs'
os.makedirs(output_dir, exist_ok=True)

# Profit rate by sentiment
plt.figure(figsize=(6,4))
sns.barplot(x='classification', y='profit_rate', data=daily)
plt.title('Daily Profit Rate by Market Sentiment')
plt.tight_layout()
plt.savefig(os.path.join(output_dir, 'profit_rate_by_sentiment.png'))
plt.show()
```



```python
# Compare mean win_rate when sentiment == Greed vs Fear
grp = daily.groupby('classification')
['profit_rate'].agg(['mean','count','std']).reset_index()
print(grp)

# Example: logistic regression (binary profitable trade) using sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, accuracy_score

# Create the features before performing logistic regression
trades['notional'] = trades['Execution Price'] * trades['Size USD']
trades['profit_flag'] = (trades['Closed PnL'] > 0).astype(int)

features = ['notional']
# Removed 'leverage' as it does not exist
X = trades[features].fillna(0)
y = trades['profit_flag']
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=42)
model = LogisticRegression(max_iter=200)
model.fit(X_train,y_train)
pred = model.predict_proba(X_test)[:,1]
print('AUC:', roc_auc_score(y_test,pred))
```

```
Empty DataFrame
Columns: [classification, mean, count, std]
Index: []
AUC: 0.5304811042053073
```

```python
Path('/content/csv_files').mkdir(parents=True, exist_ok=True)
Path('/content/outputs').mkdir(parents=True, exist_ok=True)


daily.to_csv('/content/csv_files/daily_summary.csv', index=False)
acct.to_csv('/content/csv_files/account_summary.csv', index=False)
```