```
!pip install dash
!pip install jupyter-dash
!pip install yfinance
!pip install plotly
```

```
import pandas as pd
import numpy as np
import yfinance as yf
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import plotly.io as pio
import os
import glob
```

```
# merging all the csvs into one easier to use for dashboard
folder_path = '/content/merge_csvs'
csv_files = glob.glob(os.path.join(folder_path, '*.csv'))
dfs = [pd.read_csv(file) for file in csv_files]
merged_df = pd.concat(dfs, ignore_index=True)
merged_df.to_csv('tsla_new_merged.csv', index=False)
```

```
df = pd.read_csv('/content/tsla_new_merged.csv', low_memory=False)
df['timestamp'] = pd.to_datetime(df['timestamp'])
df['date'] = df['timestamp'].dt.date
df['type'] = df['type'].str.lower()

start_date = '2025-01-01'
end_date = '2025-03-12'
tsla = yf.download('TSLA', start=start_date, end=end_date, auto_adjust=False)

if tsla.empty:
    rng = pd.date_range(start_date, end_date, freq='D')
    tsla = pd.DataFrame({'Close': np.linspace(600, 800, len(rng))}, index=rng)
    tsla.reset_index(inplace=True)
    tsla.rename(columns={'index': 'Date'}, inplace=True)
else:
    if isinstance(tsla.columns, pd.MultiIndex):
        tsla.columns = tsla.columns.map(
            lambda x: x[0] if x[0]=='Date' else (f"{x[0]}_{x[1]}" if x[1] else x[0])
        )
        tsla = tsla.loc[:, ~tsla.columns.duplicated()]
    tsla.reset_index(inplace=True)
    tsla['Date'] = pd.to_datetime(tsla['Date']).dt.date


close_col = 'Close_TSLA' if 'Close_TSLA' in tsla.columns else 'Close'
```

```
[**********100%**********]  1 of 1 completed
```

```
tsla.set_index(pd.to_datetime(tsla['Date']), inplace=True)
rng = pd.date_range(start_date, end_date, freq='D')
tsla = tsla.reindex(rng)
tsla[close_col] = tsla[close_col].ffill()
tsla.reset_index(inplace=True)
tsla.rename(columns={'index': 'Date'}, inplace=True)
tsla = tsla.loc[:, ~tsla.columns.duplicated()]
tsla['Date'] = pd.to_datetime(tsla['Date']).dt.date

if close_col not in tsla.columns:
    close_col = 'Close'


sentiment_dict = {}
platforms = df['platform'].unique()
types = df['type'].unique()

for p in platforms:
    for t in types:
        dff = df[(df['platform'] == p) & (df['type'] == t)]
        if not dff.empty:
            grp = dff.groupby('date')['sentiment_score'].mean().reset_index()
            grp['date'] = pd.to_datetime(grp['date'])
            sentiment_dict[(p, t)] = grp
```

```
fig = make_subplots(specs=[[{"secondary_y": True}]])


fig.add_trace(
    go.Scatter(
        x=tsla['Date'], y=tsla[close_col],
        mode='lines',
        name='TSLA Close',
        line=dict(color='blue', width=3),
        connectgaps=True
    ),
    secondary_y=False
)

sentiment_traces = []
for key, grp in sentiment_dict.items():
    p, t = key
    trace = go.Scatter(
        x=grp['date'], y=grp['sentiment_score'],
        mode='lines+markers',
        name=f'{p} {t} sentiment',
        visible=False
    )
    sentiment_traces.append(trace)
    fig.add_trace(trace, secondary_y=True)


if sentiment_traces:
    fig.data[1].visible = True


buttons = []
num_sentiment = len(sentiment_traces)
for i, key in enumerate(sentiment_dict.keys()):
    p, t = key
    vis = [True] + [False] * num_sentiment
    vis[1+i] = True
    buttons.append(dict(
        label=f'{p} {t}',
        method='update',
        args=[{'visible': vis},
              {'title': f'TSLA vs {p} {t} sentiment'}]
    ))
```

```
fig.update_layout(
    updatemenus=[dict(
        active=0,
        buttons=buttons,
        x=1.1,
        y=1
    )],
```

```
    title="TSLA Stock Performance vs. Sentiment",
    xaxis_title="Date",
    legend=dict(x=0, y=1)
)

tsla_min = tsla[close_col].min() * 0.95
tsla_max = tsla[close_col].max() * 1.05
fig.update_xaxes(type='date', range=[start_date, end_date])
fig.update_yaxes(title_text="TSLA Close Price", range=[tsla_min, tsla_max], secondary_y=False)
fig.update_yaxes(title_text="Average Sentiment Score", secondary_y=True)
```