In [1]:
```
# Initial imports
import os
import requests
import pandas as pd
from dotenv import load_dotenv
import alpaca_trade_api as tradeapi
import numpy as np
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from datetime import datetime, timedelta
import panel as pn
pn.extension('plotly')
import plotly.express as px
import hvplot.pandas
import matplotlib.pyplot as plt
from pathlib import Path
from holoviews import opts

%matplotlib inline
```

In [2]:
```
#Load environment Variables
load_dotenv()

## Set Alpaca and Twitter API keys and secret keys
alpaca_api_key = os.getenv("ALPACA_API_KEY")
alpaca_secret_key = os.getenv("ALPACA_SECRET_KEY")
twitter_bearer_token = os.getenv("TWITTER_BEARER_TOKEN")
```

In [13]:
```
# Define functions to be used

## Function to read 100 recent tweets related to ticker and from the date time speci
def read_100_Tweets(ticker, tweet_date_time):
    ## Function to read 100 recent tweets from the specific dates
    ##  Input : ticker - Stock Ticker
    ##          tweet_date_time - UTC Date/Time Format YYYY-MM-DDTHH:mm:ssZ (ISO 860
    ##
    ##  Output: List of 100 tweets
    auth_token = "Bearer " + twitter_bearer_token
    headers = {"Authorization": auth_token}
    twitter_api_url  = f"https://api.twitter.com/2/tweets/search/recent?max_results=
    response = requests.get(twitter_api_url, headers=headers)
    ## Check for 200 status code which means it was successful
    tweets_list = [];
    if(response.status_code == 200):
        json_response = response.json()
        #Check if there are any tweets at all
        if('data' in json_response.keys()):
            all_tweets = response.json()["data"]
            for tweet in all_tweets:
                tweets_list.append(tweet["text"])

    else:
        print(f"Response code: {response.status_code}. Error in getting the tweet")
        print(response.text)
    return tweets_list
```

```
## Perform Vader Sentiment Analysis
## Define Sentiment Object for Sentiment Analysis
sentiment_obj = SentimentIntensityAnalyzer()
def perform_sentiment_analysis(tweets_list):
    ## Function to read 100 recent tweets from the specific dates
    ##  Input : tweets_list - List of 100 tweets
    ##
    ##
    ##  Output: sentiment score average

    ##Check if there are tweets to analyse
    if (len(tweets_list) > 0):
        sentiment_scores_all = []
        for tweet in tweets_list:
            sentiment_dict = sentiment_obj.polarity_scores(tweet)
            sentiment_scores_all.append(sentiment_dict["compound"])

        #Average the sentiment of all tweets
        average_sentiment = np.average(sentiment_scores_all)
        return average_sentiment
    else:
        return 0;
```

In [44]:

```
# Input ticker to analyse using Select Widget

# Import Ticker list
ticker_list = pd.read_csv('Data/TickerList.csv',header=None)
ticker_list = list(ticker_list[0])
ticker_list.sort()

# Set up Select ticker widget
select_ticker = pn.widgets.Select(options = ticker_list, name = 'Choose Ticker')


def collect_clean_data_API(input_ticker):

    if type(input_ticker) == str:
        ticker = input_ticker
    else:
        ticker = input_ticker.value

#     ticker = input_ticker.value


    ## Set time Variables
    # Set adjustable variables
    seven_day_delta = timedelta(days=6)
    one_day_delta = timedelta(days=1)
    #Set the end date as yesterday
    end_date_time = datetime.today() - one_day_delta
    end_date = end_date_time #.date()
    #Set the start date as end date - 7 days
    start_date_time = end_date_time - seven_day_delta
    start_date = start_date_time#.date()
    #Set the time to 1PM for tweet retrieval
    start_date_time = start_date_time.replace(minute=0, hour=13, second=0)
    end_date_time = end_date_time.replace(minute=0, hour=13, second=0)



    ## Fetch and capture Ticker price data
```

```
    # Create the Alpaca API object
    alpaca = tradeapi.REST(
    alpaca_api_key,
    alpaca_secret_key,
    api_version="v2")
    # Format current date as ISO format
    start_date = pd.Timestamp(start_date, tz="America/New_York").isoformat()
    end_date = pd.Timestamp(end_date, tz="America/New_York").isoformat()
    # Set timeframe of stock bars to collect
    timeframe = "4Hour"
    # make API call to Alpaca to receive a data frame of selected ticker stock data
    df_stock_data = alpaca.get_bars(
        ticker,
        timeframe,
        end = end_date,
        start = start_date
    ).df
    # clean df_stock_data
    # remove unneeded columns
    df_stock_data.drop(['open','high','low','volume','trade_count','vwap'], axis='co
    # reset index
    df_stock_data.reset_index(inplace = True)
    # change timestamp to date only
#     df_stock_data.loc[:,'timestamp'] = df_stock_data.loc[:,'timestamp'].dt.date
    # change column names to more suitable names
    df_stock_data.columns = ['Date', 'Close']



    ## Fetch and capture Ticker sentiments
    tweet_sentiments = []
    analysis_date_time = start_date_time
    while analysis_date_time <= end_date_time:
    #    print(f"Executing Tweet Analysis for {ticker} on {analysis_date_time.isofor
        tweets_list = read_100_Tweets(ticker, analysis_date_time.isoformat() + "Z")
        sentiment_score = perform_sentiment_analysis(tweets_list)
        tweet_sentiment = {}
        tweet_sentiment["Ticker"] = ticker
        tweet_sentiment["Date"] = analysis_date_time
        tweet_sentiment["Sentiment_Score"] = sentiment_score
        tweet_sentiments.append(tweet_sentiment)
            #print("Ticker: " + tweet_sentiment["ticker"] + ", Date : " + str(tweet_
        analysis_date_time += one_day_delta

    # convert Tweet sentiment data to Data Frame
    stock_tweet_sentiment_df = pd.DataFrame(tweet_sentiments)

    return stock_tweet_sentiment_df, df_stock_data




# Plot the Data from collect_clean_data_API()
@pn.depends(select_ticker) # this will automattically push plot_recent_sentiment_vs_
def plot_recent_sentiment_vs_close(input_ticker):
    stock_tweet_sentiment_df, df_stock_data = collect_clean_data_API(input_ticker)


    ##Create Line plot output
    stock_plot = df_stock_data.hvplot.line(x = 'Date',
                                                                y = 'Close',
                                                                height = 400,
                                                                width = 600,
```

```
                                                                title = 'Stock Close Pr
                                                                ylabel = 'Closeing Pric
                                                                )

    sent_plot = stock_tweet_sentiment_df.hvplot.line(x = 'Date',
                                                     y = 'Sentiment_Score',
                                                     height = 400,
                                                     width = 600,
                                                     title = 'Sentiment Score
                                                     ylabel = 'Sentiment Scor
                                                     color = 'red')


    line_plot = (stock_plot + sent_plot)

    return line_plot
```

In [46]:

```
# Create output for Dashboard Tab

pn.Column(select_ticker, plot_recent_sentiment_vs_close).servable()
```

Out[46]:

In [ ]: