from lumibot.brokers import Alpaca

from lumibot.backtesting import YahooDataBacktesting

from lumibot.strategies.strategy import Strategy

from datetime import datetime

from alpaca\_trade\_api.rest import REST

from pandas.tseries.offsets import Timedelta

from sentiment import estimate\_sentiment

class MLTrader(Strategy):

def \_\_init\_\_(self, symbol='BANKNIFTY', cash\_at\_risk=0.5, api\_key='', api\_secret='', base\_url=''):

"""

Initialize MLTrader strategy.

Args:

symbol (str): Symbol to trade.

cash\_at\_risk (float): Percentage of cash at risk per trade.

api\_key (str): Alpaca API key.

api\_secret (str): Alpaca API secret.

base\_url (str): Base URL for Alpaca API.

"""

super().\_\_init\_\_()

self.symbol = symbol

self.cash\_at\_risk = cash\_at\_risk

self.api = REST(base\_url=base\_url, key\_id=api\_key, secret\_key=api\_secret)

self.last\_trade = None

def position\_sizing(self):

"""

Calculate position size based on available cash and risk percentage.

Returns:

tuple: Cash, last price, and quantity.

"""

cash = self.get\_cash()

last\_price = self.get\_last\_price(self.symbol)

quantity = round(cash \* self.cash\_at\_risk / last\_price, 0)

return cash, last\_price, quantity

def get\_dates(self):

"""

Get today's date and date three days prior.

Returns:

tuple: Today's date and three days prior date.

"""

today = self.get\_datetime()

three\_days\_prior = today - Timedelta(days=3)

return today.strftime('%y-%m-%d'), three\_days\_prior.strftime('%y-%m-%d')

def get\_sentiment(self):

"""

Estimate sentiment based on news headlines.

Returns:

tuple: Probability and sentiment.

"""

today, three\_days\_prior = self.get\_dates()

news = self.api.get\_news(symbol=self.symbol, start=three\_days\_prior, end=today)

news = [ev.\_raw["headline"] for ev in news]

probability, sentiment = estimate\_sentiment(news)

return probability, sentiment

def on\_trading\_iteration(self):

"""

Implement trading logic based on sentiment analysis.

"""

cash, last\_price, quantity = self.position\_sizing()

probability, sentiment = self.get\_sentiment()

if cash > last\_price:

if sentiment == "positive" and probability > 0.999:

if self.last\_trade == "sell":

self.sell\_all()

order = self.create\_order(

self.symbol,

quantity,

"buy",

type="bracket",

take\_profit\_price=last\_price \* 1.20,

stop\_loss\_price=last\_price \* 0.95

)

self.submit\_order(order)

self.last\_trade = "buy"

elif cash > last\_price:

if sentiment == "negative" and probability > 0.999:

if self.last\_order == "buy":

self.sell\_all()

order = self.create\_order(

self.symbol,

quantity,

"sell",

type="bracket",

take\_profit\_price=last\_price \* 0.8,

stop\_loss\_price=last\_price \* 1.05

)

self.submit\_order(order)

self.last\_trade = "sell"

if \_\_name\_\_ == "\_\_main\_\_":

API\_KEY = "YOUR\_ALPACA\_API\_KEY"

API\_SECRET = "YOUR\_ALPACA\_API\_SECRET"

BASE\_URL = "https://paper-api.alpaca.markets" # Or use the live URL if you are trading with a live account

start\_date = datetime(2024, 4, 2)

end\_date = datetime(2024, 12, 2)

try:

broker = Alpaca(API\_KEY, API\_SECRET, base\_url=BASE\_URL)

strategy = MLTrader(api\_key=API\_KEY, api\_secret=API\_SECRET, base\_url=BASE\_URL)

strategy.backtest(

YahooDataBacktesting,

start\_date,

end\_date,

parameters={"symbol": strategy.symbol, "cash\_at\_risk": strategy.cash\_at\_risk}

)

except Exception as e:

print(f"An error occurred: {e}")