trading_bot_macd_melissa

April 13, 2025

[1]:
```
import datetime as dt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error,␣
 ↪accuracy_score, precision_score, recall_score, f1_score
from alpaca.data.requests import StockBarsRequest
from alpaca.data.timeframe import TimeFrame
from alpaca.data.historical import StockHistoricalDataClient
from alpaca.trading.client import TradingClient
from alpaca.trading.requests import MarketOrderRequest
from alpaca.trading.enums import OrderSide, TimeInForce
```

[2]:
```
# Alpaca API credentials
api_key = "PK2DSD4BN8QNOYWSC5I1"
secret_key = "eGSHvlwHLcuf24k0cTKXDPYFMjs3n1per4SsAys8"
```

[3]:
```
client = StockHistoricalDataClient(api_key, secret_key)
trading_client = TradingClient(api_key, secret_key, paper=True)
```

[4]:
```
def compute_macd(data, fast=12, slow=26, signal=9):
    data['ema_fast'] = data['close'].ewm(span=fast, adjust=False).mean()
    data['ema_slow'] = data['close'].ewm(span=slow, adjust=False).mean()
    data['macd'] = data['ema_fast'] - data['ema_slow']
    data['signal'] = data['macd'].ewm(span=signal, adjust=False).mean()
    data['histogram'] = data['macd'] - data['signal']
    return data
```

[5]:
```
# Fetch historical stock data for MAG6
symbols = ["AAPL", "GOOGL", "AMZN", "META", "MSFT", "NVDA"]
start_date = dt.datetime.now() - dt.timedelta(days=365)
end_date = dt.datetime.now()
```

```python
request_params = StockBarsRequest(
    symbol_or_symbols=symbols,
    timeframe=TimeFrame.Day,
    start=start_date,
    end=end_date
)

bars = client.get_stock_bars(request_params).df
stock_dic = {}
scaler = MinMaxScaler(feature_range=(0, 1))

for symbol in symbols:
    df = bars[bars.index.get_level_values(0) == symbol].copy()
    df.reset_index(inplace=True)
    df['timestamp'] = pd.to_datetime(df['timestamp'])
    df.set_index('timestamp', inplace=True)
    df = compute_macd(df)
    df[['close', 'macd', 'signal']] = scaler.fit_transform(df[['close', 'macd',␣
 ↪'signal']])
    stock_dic[symbol] = df
```

[6]:

```python
# Prepare dataset for LSTM
sequence_length = 50  # Use past 50 days for prediction

def create_sequences(data):
    X, y = [], []
    for i in range(len(data) - sequence_length):
        X.append(data[i:i + sequence_length])
        y.append(data[i + sequence_length, 0])  # Predicting closing price
    return np.array(X), np.array(y)

X_train, y_train = [], []
for symbol in symbols:
    train_data = stock_dic[symbol][['close', 'macd', 'signal']].values
    X, y = create_sequences(train_data)
    X_train.append(X)
    y_train.append(y)

X_train = np.concatenate(X_train, axis=0)
y_train = np.concatenate(y_train, axis=0).reshape(-1, 1)  # Ensure y_train is 2D

# Build LSTM model
model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(sequence_length, 3)),
    Dropout(0.2),
    LSTM(50, return_sequences=False),
    Dropout(0.2),
    Dense(25, activation='relu'),
    Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=10, batch_size=16)
```

```
Epoch 1/10

C:\Users\sassy\anaconda3\envs\tf_env\lib\site-
packages\keras\src\layers\rnn\rnn.py:200: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)

75/75                 4s 26ms/step -
loss: 0.0811
Epoch 2/10
75/75                 2s 26ms/step -
loss: 0.0150
Epoch 3/10
75/75                 2s 26ms/step -
loss: 0.0121
Epoch 4/10
75/75                 2s 25ms/step -
loss: 0.0106
Epoch 5/10
75/75                 2s 26ms/step -
loss: 0.0098
Epoch 6/10
75/75                 2s 26ms/step -
loss: 0.0087
Epoch 7/10
75/75                 2s 26ms/step -
loss: 0.0084
Epoch 8/10
75/75                 2s 26ms/step -
loss: 0.0078
Epoch 9/10
75/75                 2s 26ms/step -
loss: 0.0071
Epoch 10/10
75/75                 2s 26ms/step -
loss: 0.0065
```

```
[6]: <keras.src.callbacks.history.History at 0x22665d5ddf0>
```

[7]:

```python
# Predict future prices and execute trades
predictions = {}
for symbol in symbols:
    test_data = stock_dic[symbol][['close', 'macd', 'signal']].values
    X_test, y_test = create_sequences(test_data)
    pred = model.predict(X_test)
    predictions[symbol] = pred

    last_actual_price = stock_dic[symbol]['close'].iloc[-1]
    last_predicted_price = pred[-1][0]

    if last_predicted_price > last_actual_price:  # Buy condition
        order = MarketOrderRequest(
            symbol=symbol,
            qty=1,
            side=OrderSide.BUY,
            time_in_force=TimeInForce.GTC
        )
        trading_client.submit_order(order)
        print(f"BUY Order placed for {symbol}")
    elif last_predicted_price < last_actual_price:  # Sell condition
        order = MarketOrderRequest(
            symbol=symbol,
            qty=1,
            side=OrderSide.SELL,
            time_in_force=TimeInForce.GTC
        )
        trading_client.submit_order(order)
        print(f"SELL Order placed for {symbol}")

# Print trade results
print("Trading bot executed based on LSTM predictions and MACD analysis.")
```

```
7/7               1s 60ms/step
BUY Order placed for AAPL
7/7               0s 23ms/step
BUY Order placed for GOOGL
7/7               0s 21ms/step
BUY Order placed for AMZN
7/7               0s 21ms/step
BUY Order placed for META
7/7               0s 22ms/step
BUY Order placed for MSFT
7/7               0s 21ms/step
BUY Order placed for NVDA
Trading bot executed based on LSTM predictions and MACD analysis.
```

[ ]:

```python

```