volume-detection

October 12, 2024

[ ]:
```
import ccxt
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
from tabulate import tabulate
import time

class VolumeMachine:
    def __init__(self, exchange_name='binance'):
        self.exchange = getattr(ccxt, exchange_name)()
        self.scan_interval = 30  # minutes
        self.volume_acceleration_threshold = 50  # 50% acceleration
        self.price_momentum_threshold = 2  # 2% momentum

    def fetch_data(self, symbol, timeframe='5m', limit=30):
        ohlcv = self.exchange.fetch_ohlcv(symbol, timeframe, limit=limit)
        df = pd.DataFrame(ohlcv, columns=['timestamp', 'open', 'high', 'low',␣
 ↪'close', 'volume'])
        df['timestamp'] = pd.to_datetime(df['timestamp'], unit='ms')
        return df

    def calculate_buy_sell_ratio(self, symbol):
        order_book = self.exchange.fetch_order_book(symbol, limit=20)
        bids = order_book['bids']
        asks = order_book['asks']

        bid_volume = sum(bid[1] for bid in bids)
        ask_volume = sum(ask[1] for ask in asks)

        total_volume = bid_volume + ask_volume
        buy_percentage = (bid_volume / total_volume) * 100
        sell_percentage = (ask_volume / total_volume) * 100

        return f"{buy_percentage:.2f}%\n-{sell_percentage:.2f}%"

    def identify_support_resistance(self, df):
        support = df['low'].tail(24).min()
```

```
        resistance = df['high'].tail(24).max()
        return support, resistance

    def calculate_technic_score(self, df):
        # Implement a more predictive technical analysis scoring system
        rsi = self.calculate_rsi(df)
        volume_acceleration = self.calculate_volume_acceleration(df)
        price_momentum = self.calculate_price_momentum(df)

        score = 50  # Neutral base score
        if rsi < 40:
            score += 10  # Potential for upward movement
        if volume_acceleration > self.volume_acceleration_threshold:
            score += 20  # Significant volume increase
        if price_momentum > self.price_momentum_threshold:
            score += 20  # Strong upward price movement

        return min(score, 100)  # Cap at 100

    def calculate_rsi(self, df, period=14):
        close_delta = df['close'].diff()
        up = close_delta.clip(lower=0)
        down = -1 * close_delta.clip(upper=0)
        ma_up = up.ewm(com = period - 1, adjust=True, min_periods = period).
↪mean()
        ma_down = down.ewm(com = period - 1, adjust=True, min_periods = period).
↪mean()
        rsi = ma_up / ma_down
        rsi = 100 - (100/(1 + rsi))
        return rsi.iloc[-1]

    def calculate_volume_acceleration(self, df):
        recent_volume = df['volume'].tail(6).mean()  # Last 30 minutes
        previous_volume = df['volume'].iloc[-12:-6].mean()  # Previous 30␣
↪minutes
        return (recent_volume / previous_volume - 1) * 100

    def calculate_price_momentum(self, df):
        return (df['close'].iloc[-1] / df['close'].iloc[-6] - 1) * 100  # Last␣
↪30 minutes price change

    def extract_coin_name(self, symbol):
        return symbol.replace('USDT', '')

    def has_pump_potential(self, df):
        volume_acceleration = self.calculate_volume_acceleration(df)
        price_momentum = self.calculate_price_momentum(df)
```

```
        return volume_acceleration > self.volume_acceleration_threshold and␣
↪price_momentum > self.price_momentum_threshold

    def scan_market(self):
        results = []
        markets = self.exchange.load_markets()
        usdt_pairs = [symbol for symbol in markets if symbol.endswith('USDT')]

        for symbol in usdt_pairs:
            try:
                df = self.fetch_data(symbol)
                if self.has_pump_potential(df):
                    support, resistance = self.identify_support_resistance(df)
                    buy_sell_ratio = self.calculate_buy_sell_ratio(symbol)
                    technic_score = self.calculate_technic_score(df)
                    coin_name = self.extract_coin_name(symbol)

                    results.append({
                        'symbol': coin_name,
                        'volume': f"{df['volume'].iloc[-1]:.2f}M",
                        'buy_sell': buy_sell_ratio,
                        'change': f"{self.calculate_price_momentum(df):.2f}",
                        'volume_accel': f"{self.
↪calculate_volume_acceleration(df):.2f}%",
                        'technic_score': technic_score,
                        'support_resistance': f"[{support:.4f} - {resistance:.
↪4f}]",
                        'data': df
                    })
            except Exception as e:
                print(f"Error processing {symbol}: {str(e)}")

        return results

    def display_results(self, results):
        summary_table = []
        for coin in results:
            summary_table.append([
                coin['symbol'],
                coin['volume'],
                coin['buy_sell'],
                coin['change'],
                coin['volume_accel'],
                coin['technic_score'],
                coin['support_resistance']
            ])
```

```
        print(tabulate(summary_table, headers=['Symbol', 'Volume', 'Buy/Sell␣
↪(%)', 'Momentum\n(%)', 'Vol Accel\n(%)', 'Technic\nScore',␣
↪'Support-Resistance'], tablefmt='grid'))
        print("\n")

        for coin in results:
            coin_data = coin['data'].tail(6)[::-1]
            coin_table = []
            for _, row in coin_data.iterrows():
                coin_table.append([
                    f"{row['volume']:.2f}M",
                    f"{row['close']:.4f}",
                    row['timestamp'].strftime('%d.%m.%Y\n%H:%M')
                ])
            print(f"{coin['symbol']}")
            print(tabulate(coin_table, headers=['Volume', 'Price', 'Date'],␣
↪tablefmt='grid'))
            print("\n")

    def run(self):
        while True:
            print(f"Scanning market at {datetime.now()}")
            results = self.scan_market()
            self.display_results(results)

            next_scan = datetime.now() + timedelta(minutes=self.scan_interval)
            print(f"Next scan at {next_scan}")
            sleep_time = (next_scan - datetime.now()).total_seconds()
            time.sleep(sleep_time)

if __name__ == "__main__":
    vm = VolumeMachine()
    vm.run()
```

```
Scanning market at 2024-10-12 14:10:06.424749
Error processing WAVES/USDT: division by zero
Error processing XMR/USDT: division by zero
Error processing OMG/USDT: division by zero
Error processing MITH/USDT: division by zero

C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100

Error processing COCOS/USDT: division by zero
Error processing STORM/USDT: division by zero
Error processing BTS/USDT: division by zero
Error processing BNBBEAR/USDT: division by zero
```

```
Error processing PNT/USDT: division by zero
Error processing XEM/USDT: division by zero
Error processing MDX/USDT: division by zero
Error processing PLA/USDT: division by zero

C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100

Error processing BTCST/USDT:USDT: binance {"code":-1122,"msg":"Invalid symbol
status."}

C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
```

```
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100
C:\Users\John\AppData\Local\Temp\ipykernel_6556\3487021073.py:69:
RuntimeWarning: invalid value encountered in scalar divide
  return (recent_volume / previous_volume - 1) * 100

+----------+------------+---------------+-----------+------------+-----------
+---------------------+
| Symbol   | Volume     | Buy/Sell (%)  |   Momentum | Vol Accel  |   Technic
| Support-Resistance  |
|          |            |               |        (%) | (%)        |     Score
|                     |
+==========+============+===============+===========+============+===========
+=====================+
| KDA/     | 69484.48M  | 46.10%        |       3.8  | 72.96%     |        90
| [0.5340 - 0.5740]   |
|          |            | -53.90%       |           |            |
|                     |
+----------+------------+---------------+-----------+------------+-----------
+---------------------+
| KDA/:    | 608028.00M | 56.14%        |       2.97 | 87.65%     |        90
| [0.5345 - 0.5736]   |
|          |            | -43.86%       |           |            |
|                     |
+----------+------------+---------------+-----------+------------+-----------
+---------------------+


KDA/
+------------+---------+------------+
| Volume     |   Price | Date       |
+============+=========+============+
| 69484.48M  |   0.573 | 12.10.2024 |
|            |         | 08:40      |
+------------+---------+------------+
| 396255.51M |   0.567 | 12.10.2024 |
|            |         | 08:35      |
+------------+---------+------------+
| 22590.76M  |   0.554 | 12.10.2024 |
```

		08:30
20811.88M	0.553	12.10.2024 08:25
96464.64M	0.554	12.10.2024 08:20
73427.08M	0.552	12.10.2024 08:15

KDA/:

Volume	Price	Date
608028.00M	0.5685	12.10.2024 08:40
995048.00M	0.5671	12.10.2024 08:35
71931.00M	0.5536	12.10.2024 08:30
80947.00M	0.5538	12.10.2024 08:25
606076.00M	0.5542	12.10.2024 08:20
185432.00M	0.5521	12.10.2024 08:15

Next scan at 2024-10-12 14:42:32.510952