Robo Advisors and Systematic Investing Final Project

-Devansh Purohit

Introduction

Two trading strategies have been presented, the first of them being a Combination of Moving Averages and Counter Trend Systems, and the other in introducing the RSI (Relative Strength index) Indicator and AverageTrueRange to both take advantage of momentum in the market, while also accounting for market volatility. We've used python to write code to implement these strategies and imported stock data from Yahoo! Finance.

Combining Trend and Counter-Trend Systems:

While we're already aware of how trend-following systems work, let's go through what we do in these so that the combined strategy makes sense.

Trend-Following Component: Moving Average Crossover

1. Use two simple moving averages (SMA), a shorter one (like a 10-day SMA) and a longer one (like a 30-day SMA). We also experiment with a 30/100 and an 80/160 as seen below.
2. Buy signal: When the 10-day SMA crosses above the 30-day SMA, this is a buy signal. It suggests that the trend is upward.
3. Sell signal: When the 10-day SMA crosses below the 30-day SMA, this is a sell signal. It suggests that the trend is downward.

Counter-Trend Component: Mean-Reversion Strategy

1. Again, choose a financial instrument.
2. Calculate the z-score of the price over a certain period, say 20 days. The z-score indicates how many standard deviations an element is from the mean.
3. Buy signal: When the z-score is below a certain threshold (like -1), this is a buy signal. It suggests that the price is significantly below the mean and is likely to revert back.
4. Sell signal: When the z-score is above a certain threshold (like 1), this is a sell signal. It suggests that the price is significantly above the mean and is likely to revert back.

Integrating the Two Components

The idea is to trade primarily based on the trend-following system, but to use the counter-trend system to avoid trading against significant price reversals.

1. When the trend-following system generates a buy signal, check the counter-trend system. If it's not showing a sell signal, then proceed with the buy.
2. Similarly, when the trend-following system generates a sell signal, check the counter-trend system. If it's not showing a buy signal, then proceed with the sell.

RSI and ATR:

First, let's introduce ourselves to the following terms:

RSI, or **Relative Strength Index (RSI)**: The RSI is a momentum oscillator that measures the speed and change of price movements. It oscillates between 0 and 100 and is typically used to identify overbought or oversold conditions in a market. When the RSI of a security approaches 70, it is often considered overbought, suggesting that it might be a good time to sell. Conversely, if the RSI approaches 30, it is considered oversold, which could indicate a good time to buy. In the strategy above, we're using the RSI to generate buy and sell signals: we buy when the RSI falls below 30 (an oversold condition suggesting an upcoming price increase), and sell when it rises above 70 (an overbought condition suggesting an upcoming price decrease). We also ran 60/40 and 80/20 strategies, as seen below.

Average True Range (ATR): The ATR is a technical analysis indicator that measures market volatility by decomposing the entire range of an asset price for that period. Specifically, it measures the average of true price ranges over a time period. A higher ATR indicates higher volatility, and a lower ATR indicates lower volatility.

In this strategy, we calculate the ATR for the high, low, and closing prices. We then use the ATR as a volatility filter. Specifically, we only consider the RSI Buy_Signals and Sell_Signals when the ATR is above its mean. This means we're only interested in trading when the market volatility (as measured by the ATR) is relatively high.

By combining the RSI and ATR, this strategy aims to take advantage of momentum in the market while also accounting for market volatility. The RSI generates trading signals based on momentum, while the ATR helps to manage risk by setting adaptive stop-loss levels.

Data:

To evaluate these models, we selected publicly traded stocks of more than 50 S&P 500 companies, along with the S&P500 index. We took data from 2011 to 2021, and ran 3 strategies – Simple Moving Averages, Combined Moving Averages & Mean Reversion, and RSI/ATR.

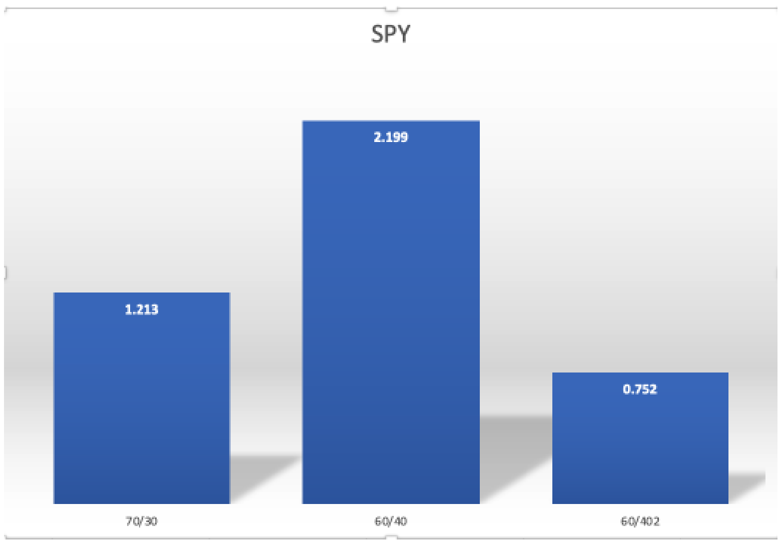
3 strategies were run for all these three approaches. While Moving Averages and the Combined Strategy were run with 10/30, 30/100 and 80/160 strategies (to compute Moving Averages), the RSI/ATR strategy was run with 80-20, 70-30, and 60-40 approaches.

The code used to generate the following results is mentioned in Appendix A.

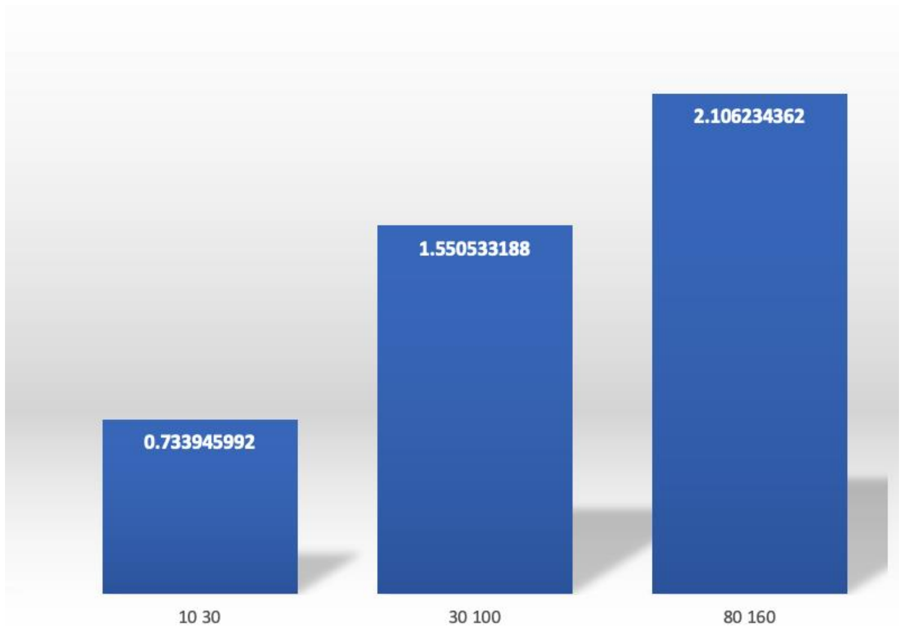
Results

As mentioned before, we ran 3 strategies for each of the approaches we have undertaken.

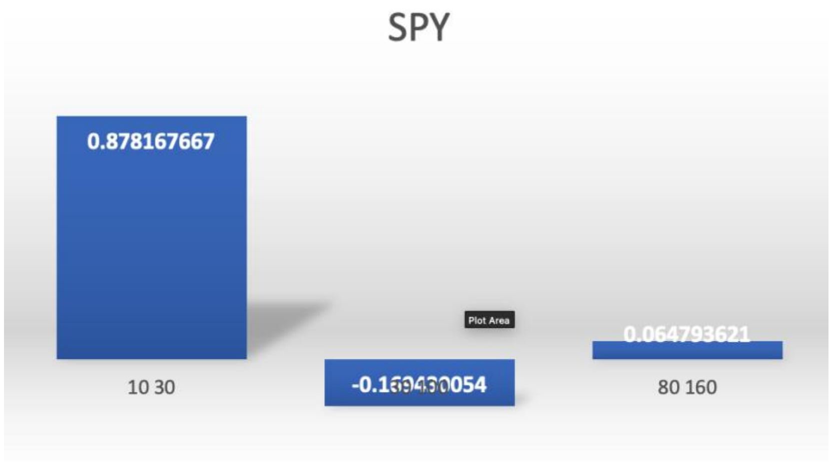
The RSI/ATR strategy performed best with a 60/40 approach, with an **average sharpe of 2.061** across all stocks. Here is an illustration of the performance of the different approaches with the S&P 500 stock:



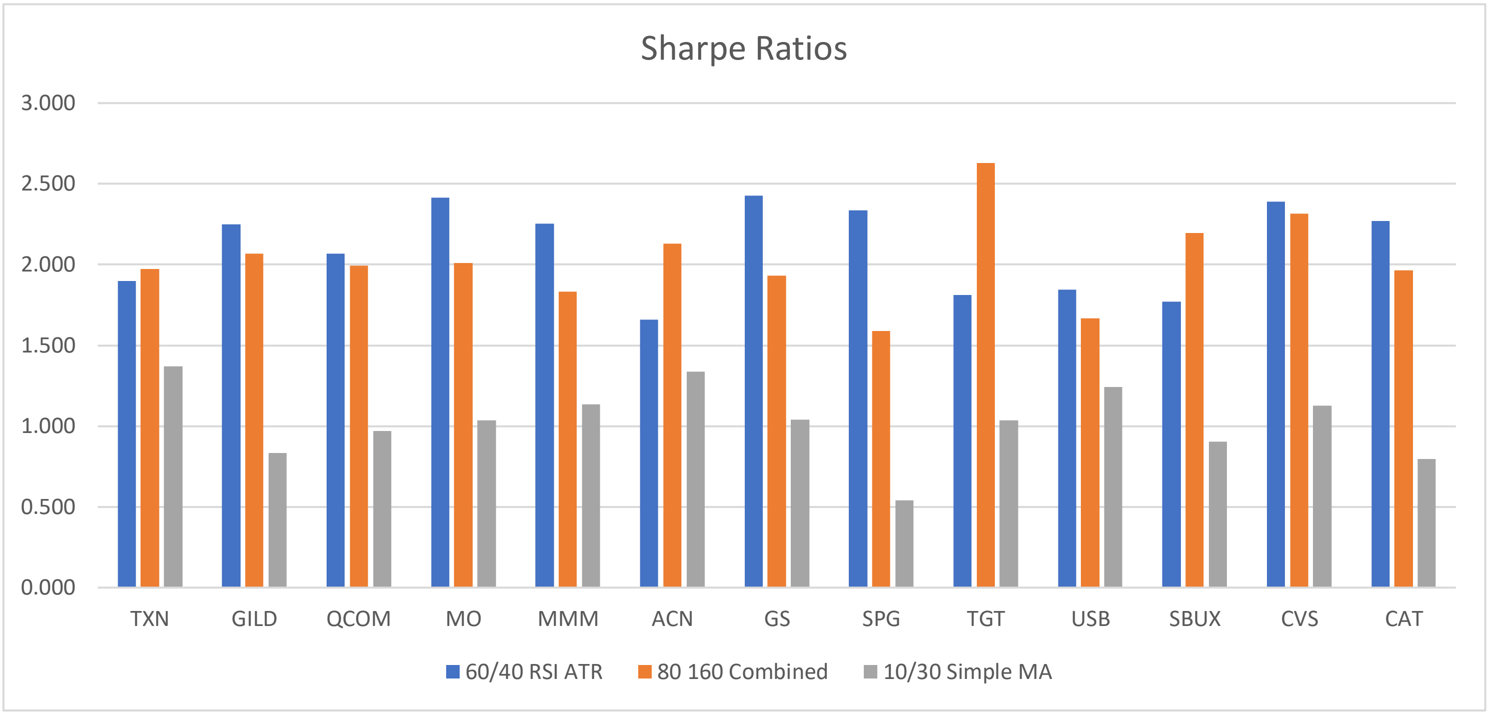
In the combined strategy, we saw that the 80/160 strategy performs the best, with an **average sharpe of 2.059**. Here is an illustration of the performance of the different approaches with the S&P 500 stock:

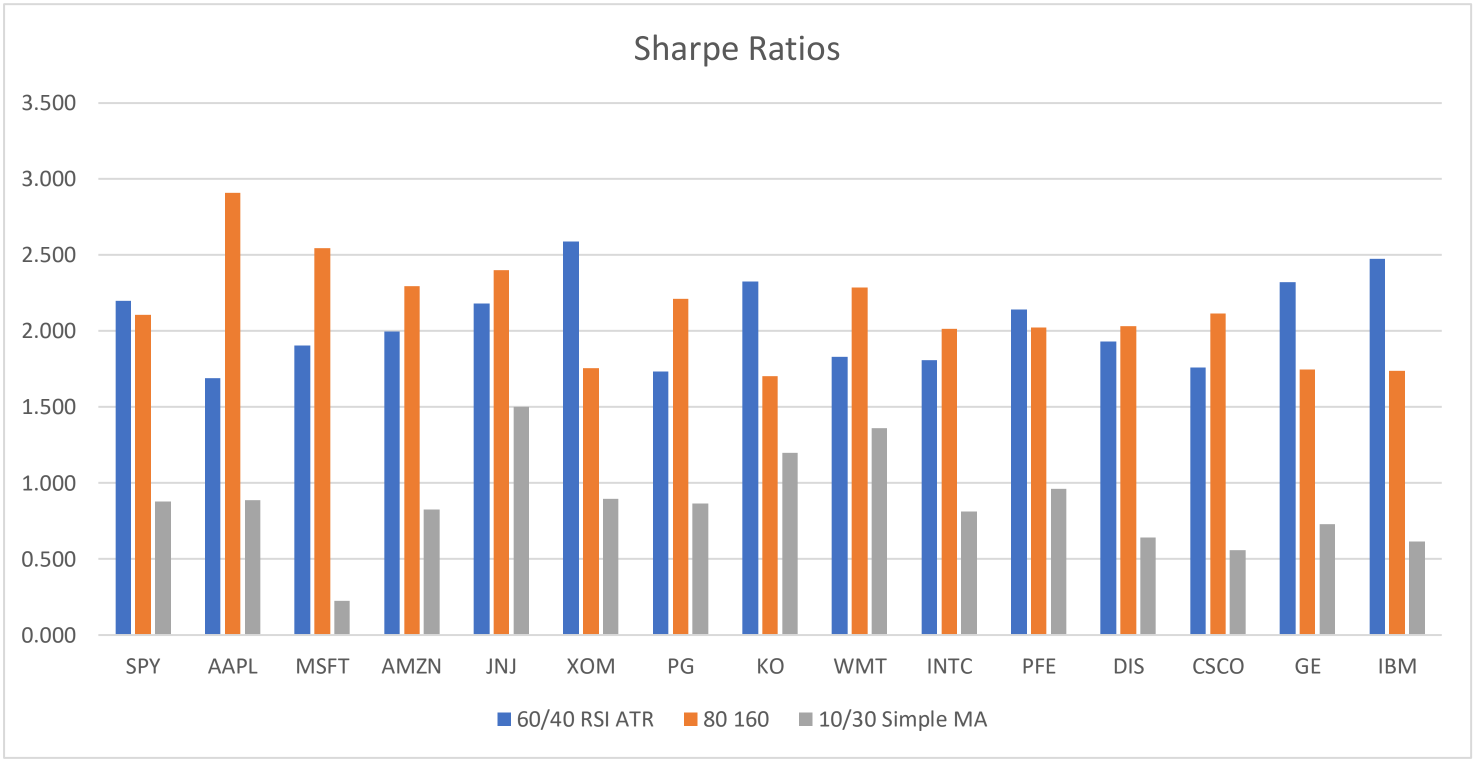


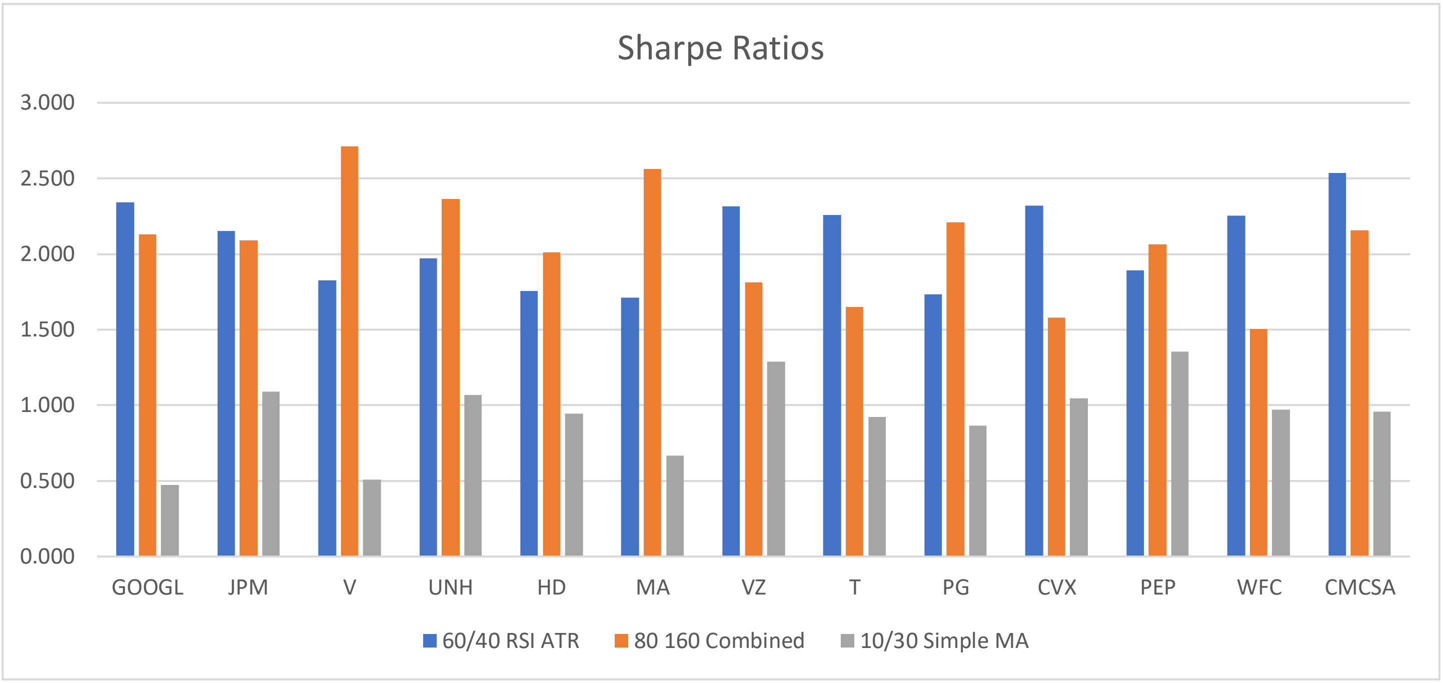
Finally, we see that the 10/30 approach worked best for the Simple Moving Average algorithm, **with an average Sharpe of 0.949.** Here is an illustration of the performance of the different approaches with the S&P 500 stock:

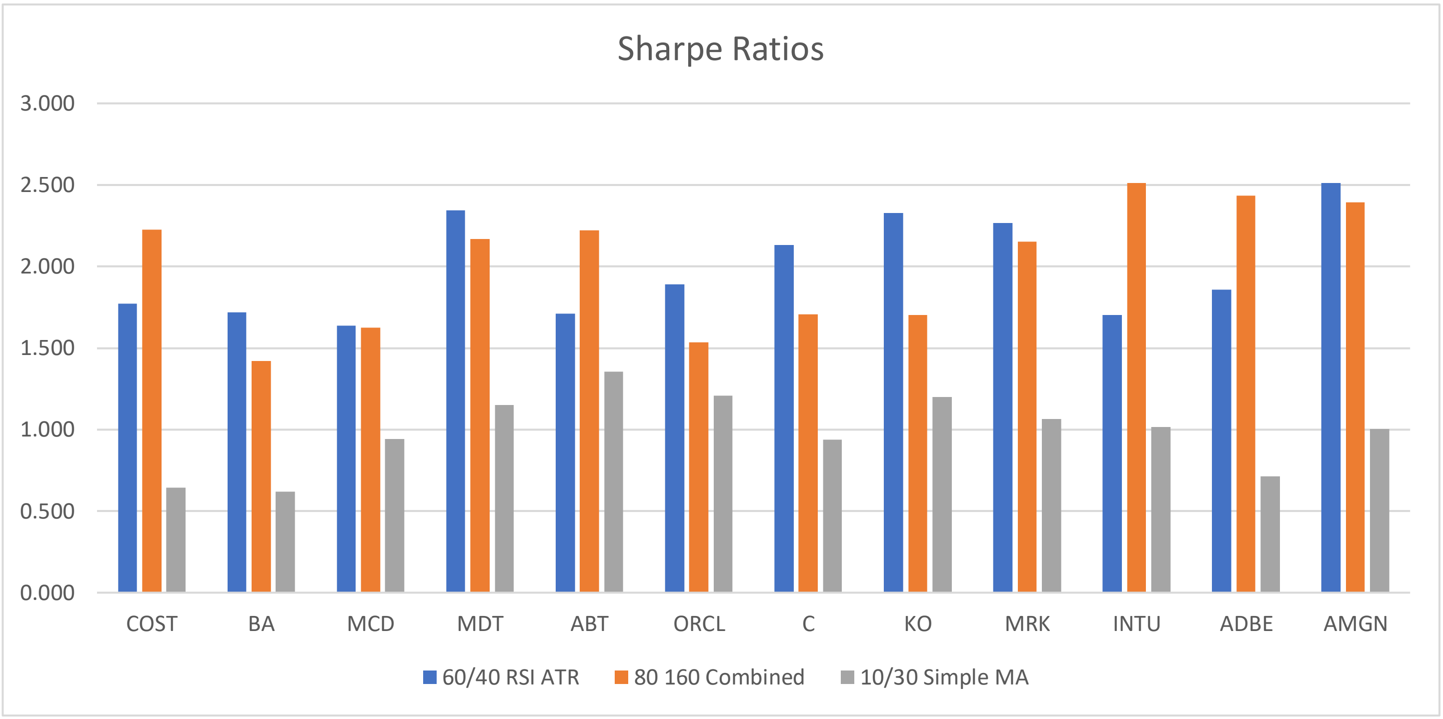


Now that we have recognized the top performers in each strategy, let's look at the Sharpe ratios of the combined dataset.






Sharpe Ratios
3.000
2.500
2.000
1.500
1.000
0.500
0.000
GOOGL
JPM
V
UNH
HD
MA
VZ
T
PG
CVX
PEP
WFC
CMCSA
60/40 RSI ATR
80 160 Combined
10/30 Simple MA


Sharpe Ratios
3.000
2.500
2.000
1.500
1.000
0.500
0.000
COST
BA
MCD
MDT
ABT
ORCL
C
KO
MRK
INTU
ADBE
AMGN
60/40 RSI ATR
80 160 Combined
10/30 Simple MA

Observations:

The strategies that we came up with eviscerate the Simple Moving average in every single stock we tested. Furthermore, both these approaches have an average Sharpe of ~2, which is impressive.

Additionally, we see that the two strategies have similar performance on many stocks. This makes sense, since both approaches are based on counting on market growth, while keeping cautious of a turnaround.

Next Steps:

- To check whether these strategies work while shorting these stocks as well. A preliminary analysis showed good results, but there were a few bugs with the output.

- To perform thorough testing across a wide range of stocks, to confirm the hypotheses that these models have put forth.

Appendix A

Code for RSI/ATR Strategy:

```
import pandas as pd
import yfinance as yf
import numpy as np
import math
from ta.volatility import AverageTrueRange
from ta.momentum import RSIIndicator
import json

# Download historical data for desired ticker symbol
# sp500_stocks = ["AAPL"]
sp500_stocks = ["SPY", "AAPL", "MSFT", "AMZN", "JNJ", "XOM", "PG", "KO", "WMT", "INTC", "PFE", "DIS", "CSCO",
"GE", "IBM", "GOOGL", "JPM", "V", "UNH", "HD", "MA", "VZ", "T", "PG", "CVX", "PEP", "WFC", "CMCSA", "COST",
"BA", "MCD", "MDT", "ABT", "ORCL", "C", "KO", "MRK", "INTU", "ADBE", "AMGN", "TXN", "GILD", "QCOM", "MO",
"MMM", "ACN", "GS", "SPG", "TGT", "USB", "SBUX", "CVS", "CAT"]


# ticker = 'AAPL'
res = []
res2 = []

for stock in sp500_stocks:

    data = yf.download(stock, start='2011-01-01', end='2021-01-01')

    # Calculate RSI
    data['RSI'] = RSIIndicator(data['Close']).rsi()

    investment = 100
```

```python
data['Daily_Returns'] = data['Close'].pct_change() + 1
data['Synthetic_URO'] = investment * data['Daily_Returns'].cumprod()

# Calculate ATR
data['ATR'] = AverageTrueRange(data['High'], data['Low'], data['Close']).average_true_range()

data['Buy_Signal_7030'] = (data['RSI'] > 30) & (data['RSI'].shift(1) <= 30) & (data['ATR'] > data['ATR'].mean())
data['Sell_Signal_7030'] = (data['RSI'] < 70) & (data['RSI'].shift(1) >= 70) & (data['ATR'] > data['ATR'].mean())

data['Buy_Signal_8020'] = (data['RSI'] > 20) & (data['RSI'].shift(1) <= 20) & (data['ATR'] > data['ATR'].mean())
data['Sell_Signal_8020'] = (data['RSI'] < 80) & (data['RSI'].shift(1) >= 80) & (data['ATR'] > data['ATR'].mean())


data['Buy_Signal_6040'] = (data['RSI'] > 40) & (data['RSI'].shift(1) <= 40) & (data['ATR'] > data['ATR'].mean())
data['Sell_Signal_6040'] = (data['RSI'] < 60) & (data['RSI'].shift(1) >= 60) & (data['ATR'] > data['ATR'].mean())


data['Position_70-30'] = np.where(data['Sell_Signal_7030'], -1, np.where(data['Buy_Signal_7030'], 1, 0))
data['Position_70-30'] = data['Position_70-30'].ffill()

data['Position_80-20'] = np.where(data['Sell_Signal_8020'], -1, np.where(data['Buy_Signal_8020'], 1, 0))
data['Position_60-40'] = np.where(data['Sell_Signal_6040'], -1, np.where(data['Buy_Signal_6040'], 1, 0))
data['Position_80-20'] = data['Position_80-20'].ffill()
data['Position_60-40'] = data['Position_60-40'].ffill()



data['Strategy_Returns_70-30'] = data['Close'].pct_change() * data['Position_70-30']
data['Strategy_Returns_70-30'] = data['Strategy_Returns_70-30'] * 100

data['Strategy_Returns_80-20'] = data['Close'].pct_change() * data['Position_80-20']
data['Strategy_Returns_80-20'] = data['Strategy_Returns_80-20'] * 100
```

```
    data['Strategy_Returns_60-40'] = data['Close'].pct_change() * data['Position_60-40']

    data['Strategy_Returns_60-40'] = data['Strategy_Returns_60-40'] * 100


    excess_returns = data['Strategy_Returns_70-30']

    sharpe_ratio_7030 = math.sqrt(260) * excess_returns.mean() / excess_returns.std()


    excess_returns = data['Strategy_Returns_80-20']

    sharpe_ratio_8020 = math.sqrt(260) * excess_returns.mean() / excess_returns.std()


    excess_returns = data['Strategy_Returns_60-40']

    sharpe_ratio_6040 = math.sqrt(260) * excess_returns.mean() / excess_returns.std()


    res.append([stock, sharpe_ratio_7030, sharpe_ratio_6040, sharpe_ratio_8020])



with open('list_data_RSI_ATR.json', 'w') as file:

    # Convert the list to JSON format and write it into the file

    json.dump(res, file)
```

Code for Combined Moving average and Mean Reversion:

```
import pandas as pd

import yfinance as yf

import numpy as np

import math

from ta.volatility import AverageTrueRange

from ta.momentum import RSIIndicator

import json


# Download historical data for desired ticker symbol

# sp500_stocks = ["AAPL"]

sp500_stocks = ["SPY", "AAPL", "MSFT", "AMZN", "JNJ", "XOM", "PG", "KO", "WMT", "INTC", "PFE", "DIS", "CSCO",

"GE", "IBM", "GOOGL", "JPM", "V", "UNH", "HD", "MA", "VZ", "T", "PG", "CVX", "PEP", "WFC", "CMCSA", "COST",
"BA", "MCD", "MDT", "ABT", "ORCL", "C", "KO", "MRK", "INTU", "ADBE", "AMGN", "TXN", "GILD", "QCOM", "MO",
"MMM", "ACN", "GS", "SPG", "TGT", "USB", "SBUX", "CVS", "CAT"]


# ticker = 'AAPL'
res = []
res2 = []

for stock in sp500_stocks:

    data = yf.download(stock, start='2011-01-01', end='2021-01-01')

    investment = 100
    data['Daily_Returns'] = data['Close'].pct_change() + 1
    data['Synthetic_URO'] = investment * data['Daily_Returns'].cumprod()


    data['SMA_10'] = data['Synthetic_URO'].rolling(window=10).mean()
    data['SMA_30'] = data['Synthetic_URO'].rolling(window=30).mean()
    data['SMA_80'] = data['Synthetic_URO'].rolling(window=80).mean()
    data['SMA_100'] = data['Synthetic_URO'].rolling(window=100).mean()
    data['SMA_160'] = data['Synthetic_URO'].rolling(window=160).mean()

    # Calculate z-score of the last 20 days
    data['z_score'] = (data['Close'] - data['Close'].rolling(window=30).mean()) /
data['Close'].rolling(window=30).std(ddof=0)

    data['Buy_Signal_10-30'] = (data['SMA_10'] > data['SMA_30']) & (data['z_score'] > -1)
    data['Sell_Signal_10-30'] = (data['SMA_10'] < data['SMA_30']) & (data['z_score'] < 1)

    data['Buy_Signal_30-100'] = (data['SMA_30'] > data['SMA_100']) & (data['z_score'] > -1)
    data['Sell_Signal_30-100'] = (data['SMA_30'] < data['SMA_100']) & (data['z_score'] < 1)
```

```python
    data['Buy_Signal_80-160'] = (data['SMA_80'] > data['SMA_160']) & (data['z_score'] > -1)
    data['Sell_Signal_80-160'] = (data['SMA_80'] < data['SMA_160']) & (data['z_score'] < 1)

    # Create a column 'Position' where we are long (+1) when Buy_Signal is True, short (-1) when Sell_Signal is True,
and flat (0) otherwise
    data['Position_10-30'] = np.where(data['Sell_Signal_10-30'], -1, np.where(data['Buy_Signal_10-30'], 1, 0))
    data['Position_30-100'] = np.where(data['Sell_Signal_30-100'], -1, np.where(data['Buy_Signal_30-100'], 1, 0))
    data['Position_80-160'] = np.where(data['Sell_Signal_80-160'], -1, np.where(data['Buy_Signal_80-160'], 1, 0))

    # Forward fill our Position column to simulate holding our position
    data['Position_10-30'] = data['Position_10-30'].ffill()
    data['Position_30-100'] = data['Position_30-100'].ffill()
    data['Position_80-160'] = data['Position_80-160'].ffill()

    # Calculate daily returns of strategy
    data['Strategy_Returns_10-30'] = data['Close'].pct_change() * data['Position_10-30']
    data['Strategy_Returns_10-30'] = data['Strategy_Returns_10-30'] * 100

    data['Strategy_Returns_30-100'] = data['Close'].pct_change() * data['Position_30-100']
    data['Strategy_Returns_30-100'] = data['Strategy_Returns_30-100'] * 100

    data['Strategy_Returns_80-160'] = data['Close'].pct_change() * data['Position_80-160']
    data['Strategy_Returns_80-160'] = data['Strategy_Returns_80-160'] * 100

    # Calculate Sharpe Ratio
    excess_returns = data['Strategy_Returns_10-30']
    sharpe_ratio_1030 = math.sqrt(260) * excess_returns.mean() / excess_returns.std()

    excess_returns = data['Strategy_Returns_30-100']
    sharpe_ratio_30100 = math.sqrt(260) * excess_returns.mean() / excess_returns.std()

    excess_returns = data['Strategy_Returns_80-160']
    sharpe_ratio_80160 = math.sqrt(260) * excess_returns.mean() / excess_returns.std()


    # print(sharpe_ratio_1030)

    # print(sharpe_ratio_30100)

    # print(sharpe_ratio_80160)


    res.append([stock, sharpe_ratio_1030, sharpe_ratio_30100, sharpe_ratio_80160])



with open('list_data_combStrat.json', 'w') as file:

    # Convert the list to JSON format and write it into the file

    json.dump(res, file)
```