Backtesting Report: Moving Average Crossover Strategy on EUR/USD

Anirudh

September 26, 2024

1 Introduction

This report presents an analysis of a simple moving average crossover strategy applied to the EUR/USD forex pair. The strategy utilizes a 20-day short-term moving average and a 50-day long-term moving average to generate trading signals. Specifically:

- **Buy Signal**: When the 20-day moving average crosses above the 50-day moving average, indicating potential upward momentum.
- **Sell Signal**: When the 20-day moving average crosses below the 50-day moving average, indicating potential downward momentum.

We evaluate the strategy using historical data from 2015 to 2023. Basic risk management techniques are implemented to assess the strategy's profitability and sustainability.

2 Data Pipeline Setup

We fetched historical daily price data for EUR/USD from Yahoo Finance using the `yfinance` Python library. The dataset spans from January 1, 2015, to October 1, 2023, and includes Open, High, Low, and Close (OHLC) prices. Missing data was addressed by forward-filling and backward-filling to ensure data consistency.

2.1 Data Fetching

The following Python function was used to fetch and preprocess the data:

```
import yfinance as yf
import pandas as pd

def fetch_forex_data(symbol='EURUSD=X', start_date='2015-01-01', end_date='
    ↪ 2023-10-01'):
    forex = yf.download(symbol, start=start_date, end=end_date, interval='1d',
        ↪ progress=False)
    forex = forex[['Open', 'High', 'Low', 'Close']]
    forex.ffill(inplace=True)
    forex.bfill(inplace=True)
    return forex

data = fetch_forex_data()
```

This function downloads the historical OHLC data for EUR/USD and handles missing values by forward-filling and backward-filling.

3 Backtesting Engine and Strategy

The strategy is built around the crossover of two moving averages:

- **Buy Signal**: When the 20-day moving average crosses above the 50-day moving average.
- **Sell Signal**: When the 20-day moving average crosses below the 50-day moving average.

3.1 Calculating Moving Averages

We calculate the short-term and long-term moving averages using the closing prices:

```
def calculate_moving_averages(data, short_window=20, long_window=50):
    data['MA_Short'] = data['Close'].rolling(window=short_window, min_periods=1).
        ↪ mean()
    data['MA_Long'] = data['Close'].rolling(window=long_window, min_periods=1).
        ↪ mean()
    return data

data = calculate_moving_averages(data)
```

3.2 Generating Trading Signals

Trading signals are generated based on the crossover of the moving averages:

```
import numpy as np

def generate_signals(data):
    data['Signal'] = 0
    data['Signal'] = np.where(data['MA_Short'] > data['MA_Long'], 1, -1)
    data['Signal'] = data['Signal'].shift(1)  # Shift signals to represent action
        ↪ at next open
    data['Signal'].fillna(0, inplace=True)
    return data

data = generate_signals(data)
```

3.3 Risk Management

Basic risk management was implemented as follows:

- **Position Sizing**: Each trade's position size is limited to 5% of the total account capital.
- **Maximum Loss Threshold**: Trading halts if total losses exceed 10% of the initial capital.
- **Stop-Loss Mechanism**: Positions are exited when losses exceed predetermined levels to limit extreme losses.

3.4 Backtesting Framework

The backtesting framework simulates trade execution and tracks performance metrics such as profit and loss (PnL), number of trades, and overall portfolio value. The core of the backtesting logic is outlined below:

```
def backtest_strategy(data, initial_capital=100000, position_size_percent=0.05,
    ↪ max_loss_percent=0.10):
    data = data.copy()
    data['Position'] = data['Signal']
    data['Market Returns'] = data['Close'].pct_change()
```

```
    data['Capital'] = initial_capital
    data['Position Size'] = 0
    data['PnL'] = 0
    data['Cumulative PnL'] = 0
    data['Stop Trading'] = False

    for i in range(1, len(data)):
        if data['Stop Trading'].iloc[i-1]:
            data['Position'].iloc[i] = 0
            data['Capital'].iloc[i] = data['Capital'].iloc[i-1]
            data['Cumulative PnL'].iloc[i] = data['Cumulative PnL'].iloc[i-1]
            data['Stop Trading'].iloc[i] = True
            continue

        data['Position Size'].iloc[i] = data['Capital'].iloc[i-1] *
            ↪ position_size_percent * data['Position'].iloc[i]
        data['PnL'].iloc[i] = data['Position Size'].iloc[i-1] * data['Market
            ↪ Returns'].iloc[i]
        data['Capital'].iloc[i] = data['Capital'].iloc[i-1] + data['PnL'].iloc[i]
        data['Cumulative PnL'].iloc[i] = data['Capital'].iloc[i] - initial_capital

        if data['Capital'].iloc[i] <= initial_capital * (1 - max_loss_percent):
            data['Stop Trading'].iloc[i] = True
            print(f"Trading stopped on {data.index[i].date()} due to maximum loss
                ↪ threshold reached.")
        else:
            data['Stop Trading'].iloc[i] = False

    return data
```

4 Performance Evaluation

The backtest results are summarized in Table 1.

Table 1: Performance Metrics

Initial Capital	$100,000.00
Final Portfolio Value	$100,116.62
Total Profit/Loss (PnL)	$116.62
Total Number of Trades	54
Winning Percentage	44.44%
Sharpe Ratio	0.18
Maximum Drawdown	1.21%

4.1 Key Metrics Analysis

The key metrics provide insights into the performance and risk profile of the strategy:

- **Total PnL**: The strategy yielded a modest profit of $116.62, indicating limited profitability.
- **Sharpe Ratio**: A Sharpe Ratio of 0.18 suggests low risk-adjusted returns, implying that the returns are not substantial relative to the risk taken.
- **Winning Percentage**: With a winning percentage of 44.44%, less than half of the trades were profitable.

- **Maximum Drawdown**: The maximum drawdown was 1.21%, demonstrating effective downside risk management.

5 Visualization

The following figures illustrate the strategy's performance and trading signals.

5.1 Equity Curve

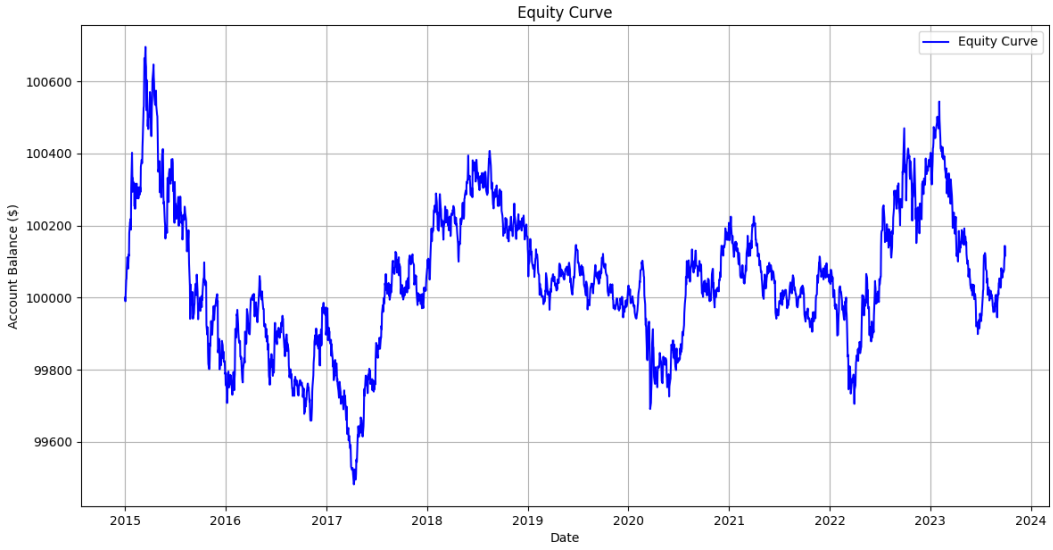

Figure 1: Equity Curve: Account balance over time during the backtest.

5.2 Drawdown Over Time

5.3 EUR/USD Price with Moving Averages and Trading Signals

6 Conclusion

The simple moving average crossover strategy applied to EUR/USD generated a small profit of $116.62 over the backtest period. While the strategy effectively limited downside risk, as indicated by the low maximum drawdown of 1.21%, the overall returns were minimal. The low Sharpe Ratio and winning percentage suggest that the strategy may require refinement. Potential improvements could include optimizing the moving average periods, incorporating additional technical indicators, or enhancing risk management techniques to improve profitability and risk-adjusted returns.

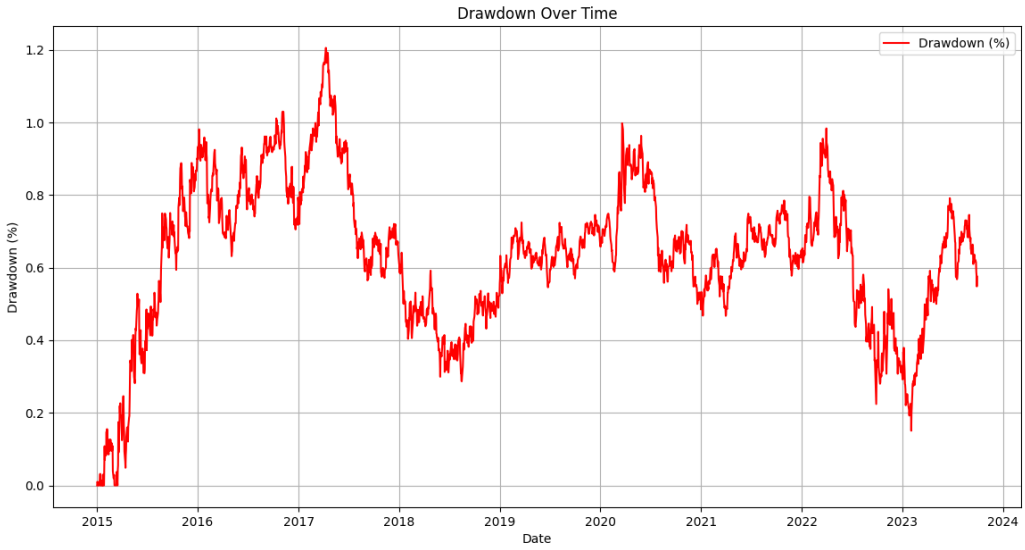

Figure 2: Drawdown Over Time: Peak-to-trough percentage decline in portfolio value.

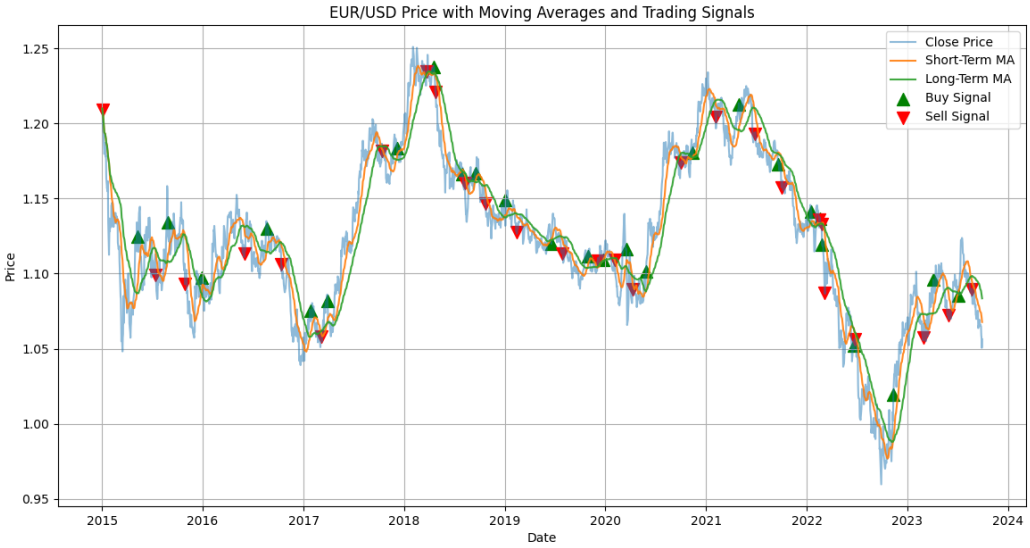

Figure 3: Moving Average Crossover Strategy: EUR/USD price with short-term and long-term moving averages, alongside buy/sell signals.