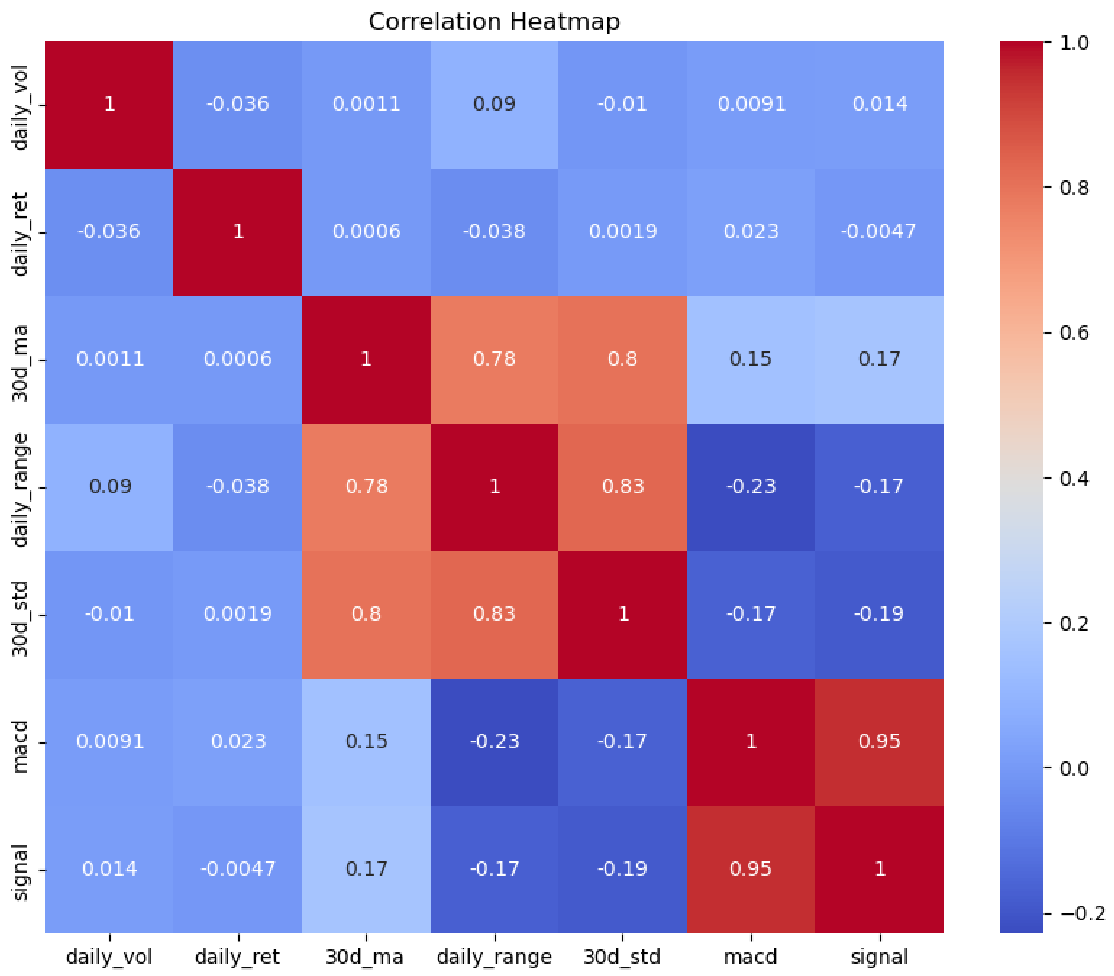
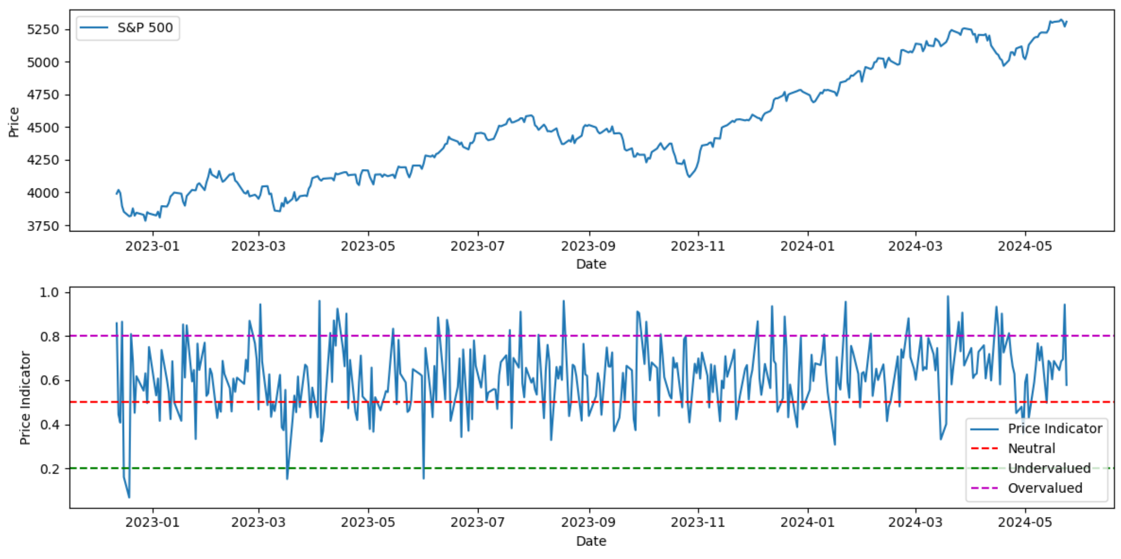
S&P Indicator using KNN

May 26, 2024

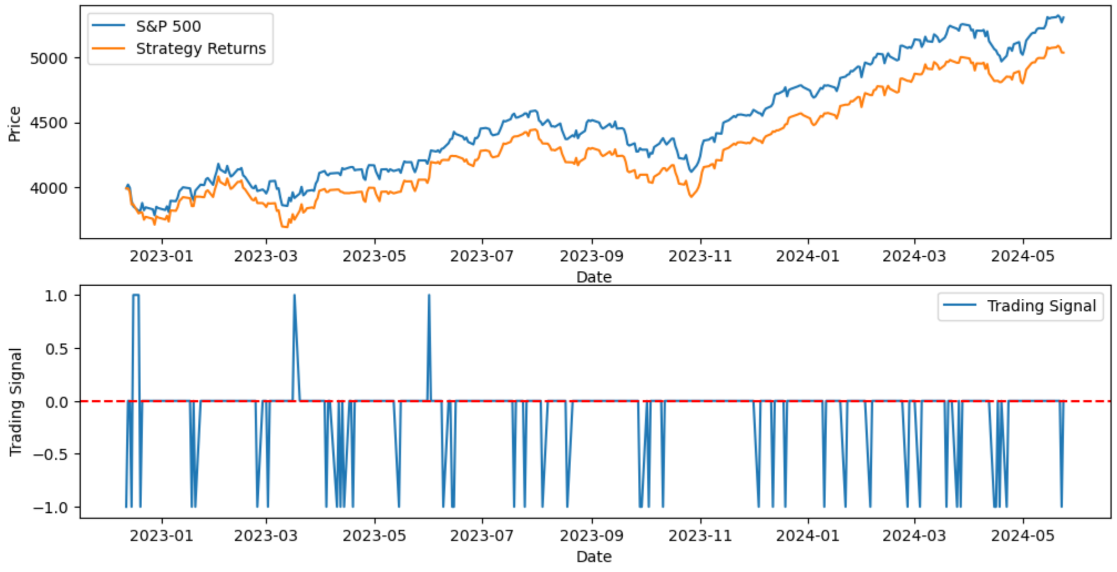
[2]:
```python
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsRegressor
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import seaborn as sns
import yfinance as yf

# ignore warnings
import warnings
warnings.filterwarnings('ignore')


# Step 1. Download S&P 500 index data from Yahoo Finance
# ============================================================
spx = yf.Ticker('^GSPC')
spx_data = spx.history(period='max')

# Step 2. Add a Selection of Indicators that you would like to incorporate in␣
 ↪the model
# ============================================================

# Create momentum indicators
spx_data['daily_vol'] = spx_data['Volume'].diff()
spx_data['daily_ret'] = spx_data['Close'].pct_change()
spx_data['30d_ma'] = spx_data['Close'].rolling(30).mean()
spx_data['daily_range'] = spx_data['High'] - spx_data['Low']
spx_data['30d_std'] = spx_data['Close'].rolling(30).std()

# Add MACD indicator
spx_data['macd'] = spx_data['Close'].ewm(span=12, adjust=False).mean() -␣
 ↪spx_data['Close'].ewm(span=26, adjust=False).mean()

# smoothed MACD line
spx_data['signal'] = spx_data['macd'].ewm(span=9, adjust=False).mean()

# Step 3. Split the Data for Training and Out-of-Sample Testing
# ============================================================

# Split data and create KNN model
train = spx_data.iloc[:-365, :]
test = spx_data.iloc[-365:, :]

# Handle missing values in the training data
imputer = SimpleImputer(strategy='mean')
X_train = train[['daily_vol', 'daily_ret', '30d_ma', 'daily_range', '30d_std',␣
 ↪'macd', 'signal']]
X_train = pd.DataFrame(imputer.fit_transform(X_train), columns=X_train.columns)
y_train = train['Close']

X_test = test[['daily_vol', 'daily_ret', '30d_ma', 'daily_range', '30d_std',␣
 ↪'macd', 'signal']]
y_test = test['Close']

# Step 4. Check for Corrlation between Features and Remove Features that are␣
 ↪too Similar and Highly Correlated
# ============================================================
# Check correlation between features
corr = X_train.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Step 5. Run the Kth Nearest Neighbours (KNN) Model
# ============================================================

knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)

# Step 6. Create price indicator
# ============================================================
y_pred = knn.predict(X_test)
test.loc[:, 'price_indicator'] = (y_test - y_pred) / y_test

# Step 7. Plot price and indicator
# ============================================================
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(test.index, test['Close'], label='S&P 500')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.subplot(2, 1, 2)
plt.plot(test.index, test['price_indicator'], label='Price Indicator')
plt.axhline(y=0.5, color='r', linestyle='--', label='Neutral')
plt.axhline(y=0.2, color='g', linestyle='--', label='Undervalued')
plt.axhline(y=0.8, color='m', linestyle='--', label='Overvalued')
plt.xlabel('Date')
plt.ylabel('Price Indicator')
plt.legend()

plt.tight_layout()
plt.show()

# Step 8. Implement a Trading Strategy Based on the Price Indicator
# ============================================================
# Initialize the trading strategy
test.loc[:, 'Trading Signal'] = 0.0
test.loc[:, 'Strategy Returns'] = 0.0
test.loc[:, 'Position'] = 0  # 1 for long, -1 for short, 0 for no position

# Generate signals
test.loc[test['price_indicator'] > 0.8, 'Trading Signal'] = -1  # Short signal
test.loc[test['price_indicator'] < 0.2, 'Trading Signal'] = 1  # Long signal
test.loc[test['price_indicator'] == 0.5, 'Trading Signal'] = 0  # Exit signal

# Calculate the strategy returns
test['Position'] = test['Trading Signal'].shift(1)
test.loc[test['Position'] == 0, 'Strategy Returns'] = 0  # No returns when no␣
 ↪position
test.loc[test['Position'] == 1, 'Strategy Returns'] = test['daily_ret']  # Long␣
 ↪returns
test.loc[test['Position'] == -1, 'Strategy Returns'] = -test['daily_ret']  #␣
 ↪Short returns

# Calculate the Sharpe ratio and maximum drawdown
strategy_returns = test['Strategy Returns'].dropna()
sharpe_ratio = strategy_returns.mean() / strategy_returns.std() * np.sqrt(252)
max_drawdown = np.max(np.maximum.accumulate(strategy_returns) -␣
 ↪strategy_returns)

print(f"Sharpe Ratio: {sharpe_ratio:.2f}")
print(f"Maximum Drawdown: {max_drawdown:.2f}")

# Plot the strategy returns and equity curve
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(test.index, test['Close'], label='S&P 500')
```

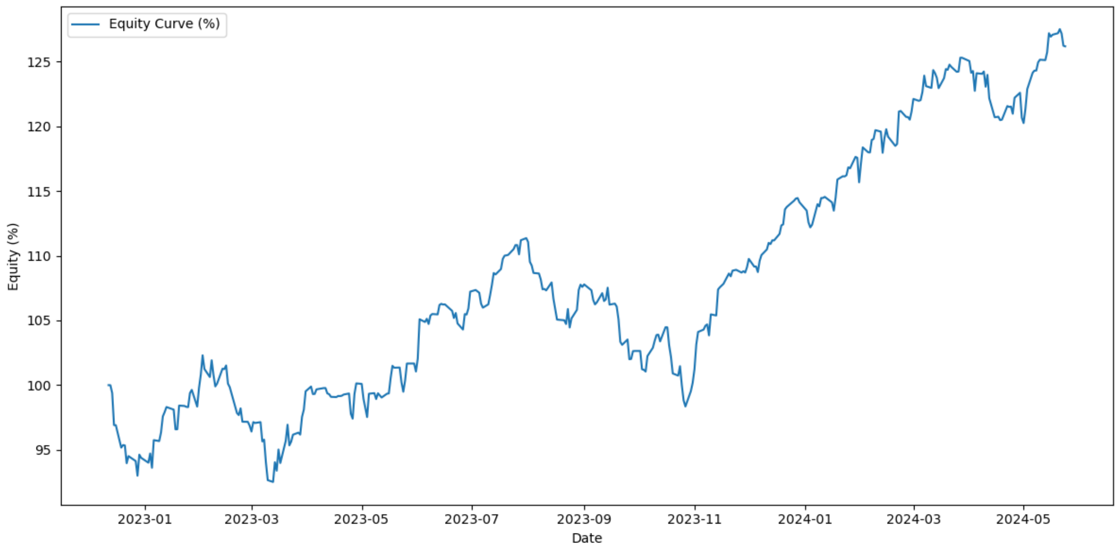
```python
plt.plot(test.index, test['Close'] * (1 + test['Strategy Returns'].cumsum().
  ↪fillna(0)), label='Strategy Returns')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(test.index, test['Trading Signal'], label='Trading Signal')
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Date')
plt.ylabel('Trading Signal')
plt.legend()

# Add a new subplot for the equity curve
plt.figure(figsize=(12, 6))
plt.subplot(1, 1, 1)
initial_investment = test['Close'][0]
equity_curve = test['Close'] * (1 + test['Strategy Returns'].cumsum().
  ↪fillna(0)) / initial_investment * 100
plt.plot(test.index, equity_curve, label='Equity Curve (%)')
plt.xlabel('Date')
plt.ylabel('Equity (%)')
plt.legend()

plt.tight_layout()
plt.show()
```


Correlation Heatmap
daily_vol
daily_ret
30d_ma
daily_range
30d_std
macd
signal
1 -0.036 0.0011 0.09 -0.01 0.0091 0.014
-0.036 1 0.0006 -0.038 0.0019 0.023 -0.0047
0.0011 0.0006 1 0.78 0.8 0.15 0.17
0.09 -0.038 0.78 1 0.83 -0.23 -0.17
-0.01 0.0019 0.8 0.83 1 -0.17 -0.19
0.0091 0.023 0.15 -0.23 -0.17 1 0.95
0.014 -0.0047 0.17 -0.17 -0.19 0.95 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


S&P 500
Price
5250
5000
4750
4500
4250
4000
3750
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
2024-03
2024-05
Date
Price Indicator
1.0
0.8
0.6
0.4
0.2
Price Indicator
Neutral
Undervalued
Overvalued
Date

```
Sharpe Ratio: -0.78
Maximum Drawdown: 0.03
```





```
[ ]:
```