In [6]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from pandas_datareader import data as web
from sklearn.decomposition import PCA
import seaborn as sns
from datetime import datetime as dt, timedelta as td
from pykalman import KalmanFilter
sns.set()

data = yf.download('BTC-USD', start='2023-01-01', end='2024-07-05')

df=data.drop(columns=['Open', 'High','Low','Adj Close','Volume'])
df.tail()
```

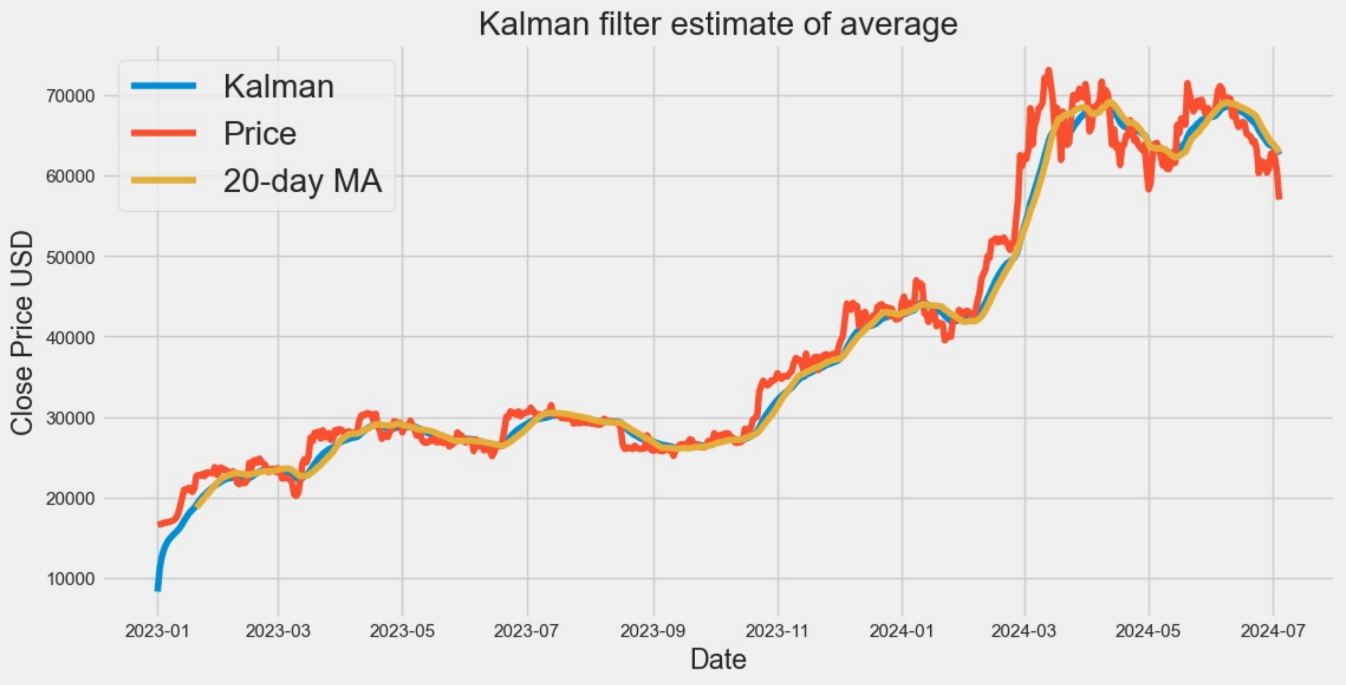
```
[*********************100%%**********************]  1 of 1 completed
```

Out[6]:

	Close
Date	
2024-06-30	62678.292969
2024-07-01	62851.980469
2024-07-02	62029.015625
2024-07-03	60173.921875
2024-07-04	56977.703125

In [7]:

```
import matplotlib.pyplot as plt
import requests
import math
from termcolor import colored as cl

from pykalman import KalmanFilter

plt.style.use('fivethirtyeight')
plt.rcParams['figure.figsize'] = (15, 8)

kf = KalmanFilter(
    transition_matrices = [1],
    observation_matrices = [1],
    initial_state_mean = 0,
    initial_state_covariance = 1,
    observation_covariance=1,
    transition_covariance=0.01
)
state_means, _ = kf.filter(df.values)
state_means = pd.Series(state_means.flatten(), index=df.index)

mean30 = df['Close'].rolling(window=20).mean()

plt.figure(figsize=(12,6))
plt.plot(state_means)
plt.plot(df['Close'])
plt.plot(mean30)
plt.title('Kalman filter estimate of average', fontsize=20)
plt.legend(['Kalman', 'Price', '20-day MA'], fontsize=20)
plt.xlabel('Date')
plt.ylabel('Close Price USD')

def sma(data, n):
    sma = data.rolling(window = n).mean()
    return pd.DataFrame(sma)
```



In [8]:

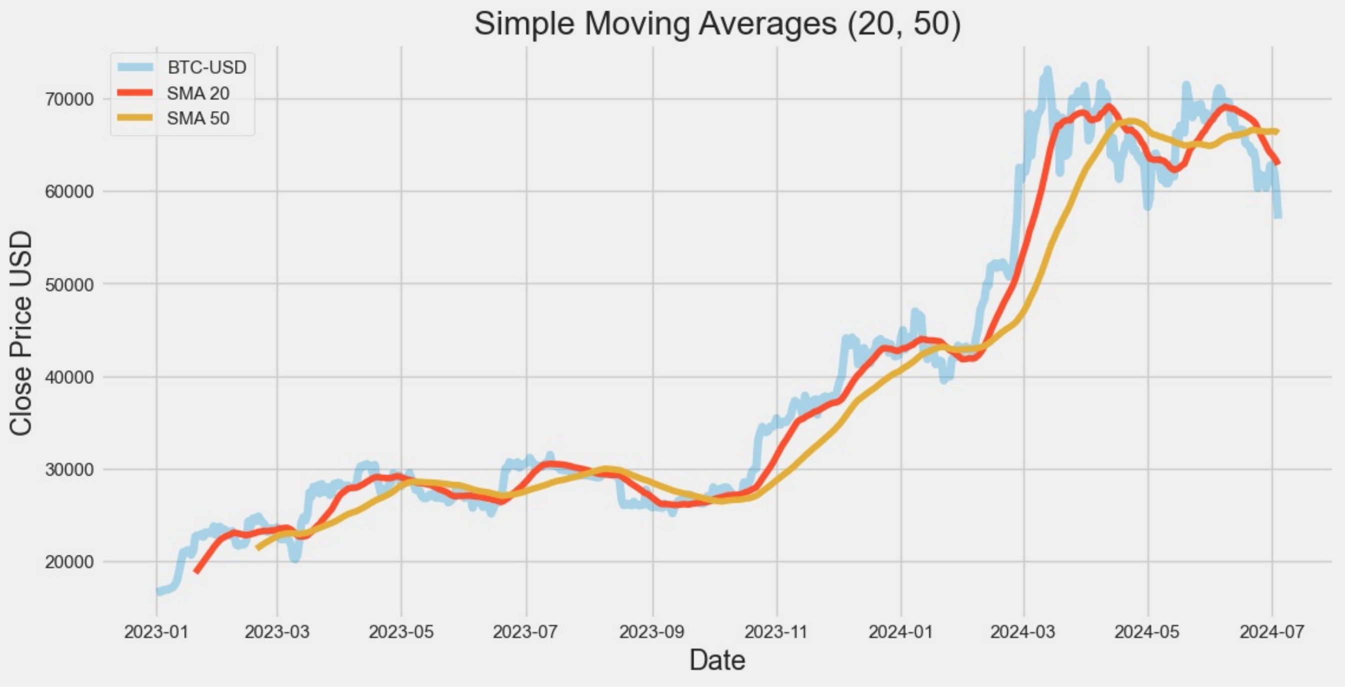
```
n = [20, 50]
for i in n:
    df[f'sma_{i}'] = sma(df['Close'], i)

df.tail()
```

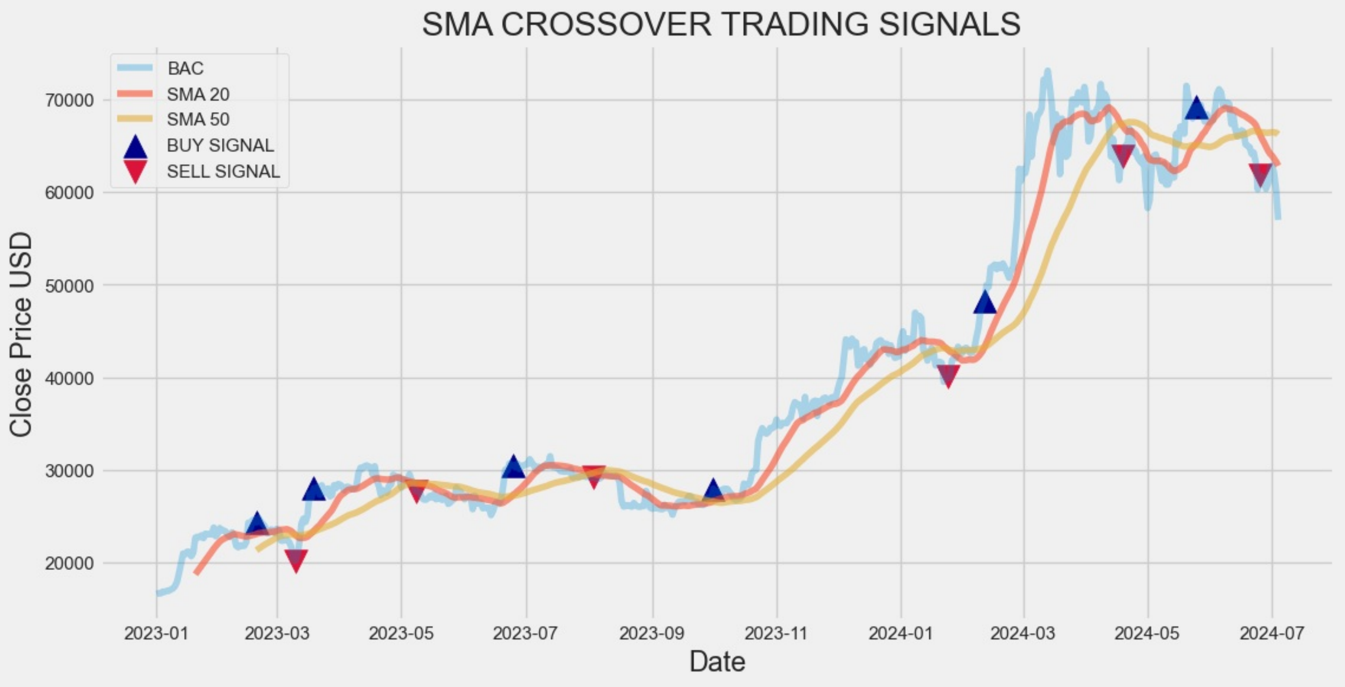
Out[8]:

Date	Close	sma_20	sma_50
2024-06-30	62678.292969	64125.213477	66414.575313
2024-07-01	62851.980469	63901.210938	66442.647031
2024-07-02	62029.015625	63590.602344	66425.198359
2024-07-03	60173.921875	63261.478516	66397.621016
2024-07-04	56977.703125	62809.808984	66211.825234

In [10]:

```
plt.figure(figsize=(12,6))
plt.plot(df['Close'], label = 'BTC-USD', linewidth = 5, alpha = 0.3)
plt.plot(df['sma_20'], label = 'SMA 20')
plt.plot(df['sma_50'], label = 'SMA 50')
plt.title('Simple Moving Averages (20, 50)')
plt.legend(loc = 'upper left')

plt.xlabel('Date')
plt.ylabel('Close Price USD')
plt.show()

def implement_sma_strategy(data, short_window, long_window):
    sma1 = short_window
    sma2 = long_window
    buy_price = []
    sell_price = []
    sma_signal = []
    signal = 0

    for i in range(len(data)):
        if sma1.iloc[i] > sma2.iloc[i]:
            if signal != 1:
                buy_price.append(data.iloc[i])
                sell_price.append(np.nan)
                signal = 1
                sma_signal.append(signal)
            else:
                buy_price.append(np.nan)
                sell_price.append(np.nan)
                sma_signal.append(0)
        elif sma2.iloc[i] > sma1.iloc[i]:
            if signal != -1:
                buy_price.append(np.nan)
                sell_price.append(data.iloc[i])
                signal = -1
                sma_signal.append(-1)
            else:
                buy_price.append(np.nan)
                sell_price.append(np.nan)
                sma_signal.append(0)
        else:
            buy_price.append(np.nan)
            sell_price.append(np.nan)
            sma_signal.append(0)

    return buy_price, sell_price, sma_signal

sma_20 = df['sma_20']
sma_50 = df['sma_50']

buy_price, sell_price, signal = implement_sma_strategy(df['Close'], sma_20, sma_50)

plt.figure(figsize=(12,6))
plt.plot(df['Close'], alpha = 0.3, label = 'BAC')
plt.plot(sma_20, alpha = 0.6, label = 'SMA 20')
plt.plot(sma_50, alpha = 0.6, label = 'SMA 50')
plt.scatter(df.index, buy_price, marker = '^', s = 200, color = 'darkblue', label = 'BUY SIGNAL')
plt.scatter(df.index, sell_price, marker = 'v', s = 200, color = 'crimson', label = 'SELL SIGNAL')
plt.legend(loc = 'upper left')
plt.title(' SMA CROSSOVER TRADING SIGNALS')
plt.xlabel('Date')
plt.ylabel('Close Price USD')
plt.show()

print(signal)
```





```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

In [11]:

```
position = []
for i in range(len(signal)):
    if signal[i] > 1:
        position.append(0)
    else:
        position.append(1)

for i in range(len(df['Close'])):
    if signal[i] == 1:
        position[i] = 1
    elif signal[i] == -1:
        position[i] = 0
    else:
        position[i] = position[i-1]

sma_20 = pd.DataFrame(sma_20).rename(columns = {0:'sma_20'})
sma_50 = pd.DataFrame(sma_50).rename(columns = {0:'sma_50'})
signal = pd.DataFrame(signal).rename(columns = {0:'sma_signal'}).set_index(df.index)
position = pd.DataFrame(position).rename(columns = {0:'sma_position'}).set_index(df.index)

frames = [sma_20, sma_50, signal, position]
strategy = pd.concat(frames, join = 'inner', axis = 1)
strategy = strategy.reset_index().drop('Date', axis = 1)

msft_ret = pd.DataFrame(np.diff(df['Close'])).rename(columns = {0:'returns'})
sma_strategy_ret = []

for i in range(len(msft_ret)):
    try:
        returns = msft_ret['returns'].iloc[i]*strategy['sma_position'].iloc[i]
        sma_strategy_ret.append(returns)
    except:
        pass

sma_strategy_ret_df = pd.DataFrame(sma_strategy_ret).rename(columns = {0:'sma_returns'})

investment_value = 100000
number_of_stocks = math.floor(investment_value/df['Close'].iloc[1])
sma_investment_ret = []

for i in range(len(sma_strategy_ret_df['sma_returns'])):
    returns = number_of_stocks*sma_strategy_ret_df['sma_returns'].iloc[i]
    sma_investment_ret.append(returns)

sma_investment_ret_df = pd.DataFrame(sma_investment_ret).rename(columns = {0:'investment_returns'})
total_investment_ret = round(sum(sma_investment_ret_df['investment_returns']), 2)
print(cl('Profit gained from the strategy by investing $100K in BTC-USD : ${}'.format(total_investment_ret), at
```

```
Profit gained from the strategy by investing $100K in BTC-USD : $110489.26
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js