Sarit Maitra

1.4K Followers About

PREDICTING TRADING SIGNALS OF STOCK MARKETS

Trading Signal Generation Techniques & Visualization

Technical analysis to build signals

Sarit Maitra Just now · 5 min read ★

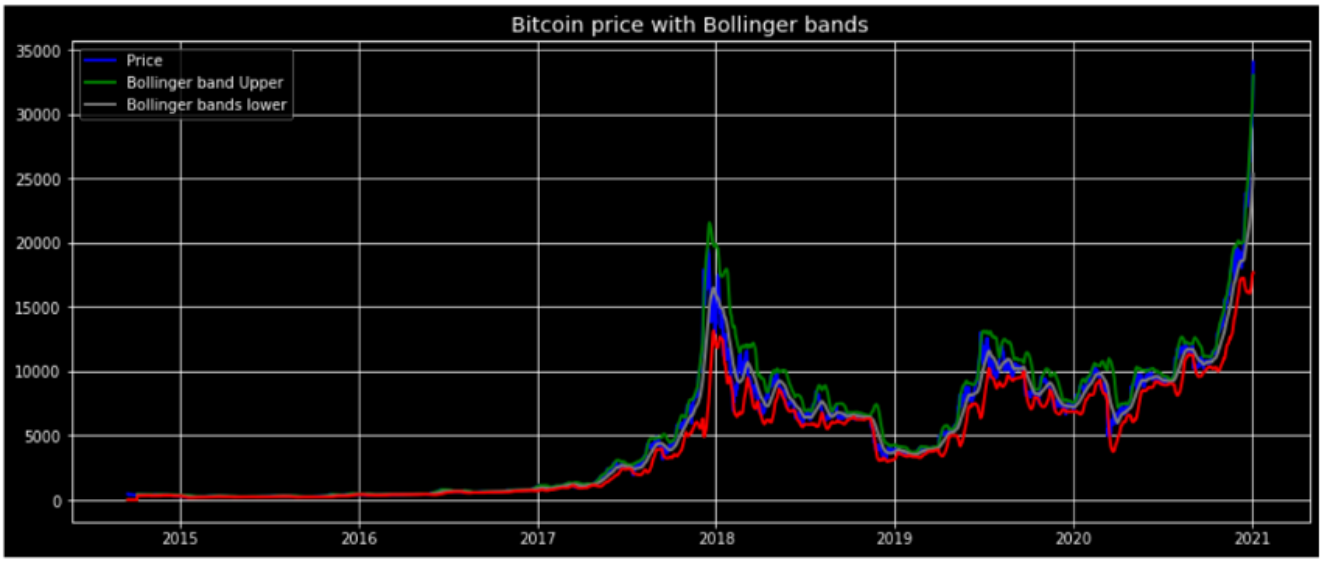


Image by author

Profitability of stock market trading is directly related to the prediction of trading signals. Here, we will discuss about some basic to advanced and popular technical analysis to build trading signals. Our focus will be on signal generation and visualization. A long list of technical indicators are available covering principal domains

However, once signal is generated, strategy is defined, the next most important task is performance testing which is not the scope of this article.

We will use free crypto currency data as shown below.

```
btc = yf.Ticker("BTC-USD")
# get historical market data
hist = btc.history(period="max")
df = hist[['Open', 'High', 'Low', 'Close', 'Volume']]
df = df.sort_index(ascending=True)
print(df.tail()); print(); print(df.shape)
```

```
                Open     High      Low    Close       Volume
Date
2020-12-30 27360.09 28937.74 27360.09 28840.95  51287442704
2020-12-31 28841.57 29244.88 28201.99 29001.72  46754964848
2021-01-01 28994.01 29600.63 28803.59 29374.15  40730301359
2021-01-02 29376.46 33155.12 29091.18 32127.27  67865420765
2021-01-03 32741.49 34538.36 32055.92 33990.76  86481952768

(2301, 5)
```

Trade Signal:

Buy low, Sell high:

```
btc = df[['close']]
btc['daily_difference'] = btc['Close'].diff() # difference in the close prices between two consecutive days
"""
differenced price is negative this means the price on the previous day was higher than the price the following day, so we
can buy. Positive value, means that we can sell because the price is higher.
"""
btc['signal'] = 0.0
btc['signal'] = np.where(btc['daily_difference'] > 0, 1.0, 0.0)
"""
positive value 1, otherwise, the value is 0
"""
btc['positions'] = btc['signal'].diff()
"""
It is not advisable to buy or sell constantly when price moving down, or up. To restrict buy/sell applied diff()
"""
btc.head()
```

2014-09-17	457.33	nan	0.00	nan
2014-09-18	424.44	-32.89	0.00	0.00
2014-09-19	394.80	-29.64	0.00	0.00
2014-09-20	408.90	14.11	1.00	1.00
2014-09-21	398.82	-10.08	0.00	-1.00

```
1  btc.positions.value_counts()
```

```
0.00     1107
1.00      597
-1.00     596
Name: positions, dtype: int64
```

This way, we could generate 597 buy signals and 596 sell signals. Let's visualize these signals in below plot.

```
print('\033[4mFor each day where Close price = Buy = red arrow head  and Sell = green arrow head \033[0m')
buys = btc.loc[btc['positions'] == 1]
sells = btc.loc[btc['positions'] == -1]
# Plot
fig = plt.figure(figsize=(15, 5))
plt.plot(btc.index, btc['Close'], color = 'gray', lw=2., label='Close price')
# Plot the buy and sell signals on the same plot
plt.plot(buys.index, btc.loc[buys.index]['Close'], '^', markersize=3, color='r',
         label='Buy')
plt.plot(sells.index, btc.loc[sells.index]['Close'], 'v', markersize=3,
         color='g', label='Sell')
plt.ylabel('BTC Price (USD)')
plt.xlabel('Date')
plt.title('Buy and sell signals plot')
plt.legend(loc=0)
plt.show()
```

For each day where Close price = Buy = red arrow head and Sell = green arrow head

Moving averages crossover:

We create two separate Simple Moving Averages (SMA) with differing lookback periods, 10 days and 30 days. If the short moving average exceeds the long moving average then we go long, if the long moving average exceeds the short moving average then we exit.

When we go long, we think that the stock price will go up and will sell at a higher price in the future (= buy signal); When we go short, we sell our stock, expecting that we can buy it back at a lower price and realize a profit (= sell signal).

Lastly, we take the difference of the signals in order to generate actual trading orders. In other words, in this column of our signals Data Frame, we'll be able to distinguish between long and short positions, whether we're buying or selling.

```
# Initialize the short and long windows
short_window = 10
long_window = 30
# Initialize the `signals` DataFrame with the `trade` column
trade = btc[['Close']]
trade['signal'] = 0.0
# short simple moving average over the short window
trade['short_ma'] = btc['Close'].rolling(window=short_window).mean()
# long simple moving average over the long window
trade['long_ma'] = btc['Close'].rolling(window=long_window).mean()
# signals generation
trade['signal'][short_window:] = np.where(trade['short_ma'][short_window:] < trade['long_ma'][short_window:], 1.0, 0.0)
# trading orders
trade['positions'] = trade['signal'].diff()
print(trade.tail())
```

```
             Close  signal  short_ma  long_ma  positions
Date
2020-12-30 28840.95    0.00  25422.57 21749.18       0.00
2020-12-31 29001.72    0.00  26042.44 22089.14       0.00
2021-01-01 29374.15    0.00  26601.55 22428.24       0.00
2021-01-02 32127.27    0.00  27490.14 22850.97       0.00
2021-01-03 34045.10    0.00  28521.06 23362.48       0.00
```

```
trade.positions.value_counts()
```

```
0.00     2224
-1.00      38
1.00       38
Name: positions, dtype: int64
```

```
plt.style.use('dark_background')
buys = trade.loc[trade['positions'] == 1]
sells = trade.loc[trade['positions'] == -1]

# Plot
fig = plt.figure(figsize=(15, 5))
plt.plot(trade.index, trade['Close'], color='g', lw=.5, label='Close price')
# Plot the short and long moving averages
plt.plot(trade[['short_ma', 'long_ma']])
plt.plot(buys.index, trade.loc[buys.index]['Close'], '^', markersize=7, color='red', label='Buy')
plt.plot(sells.index, trade.loc[sells.index]['Close'], 'v', markersize=7, color='blue', label = 'Sell')
plt.ylabel('BTC Price (USD)')
plt.xlabel('Date')
plt.legend(["Price","Short mavg","Long mavg","Buy","Sell"])
plt.title("Dual SMA Strategy")
plt.show()
```

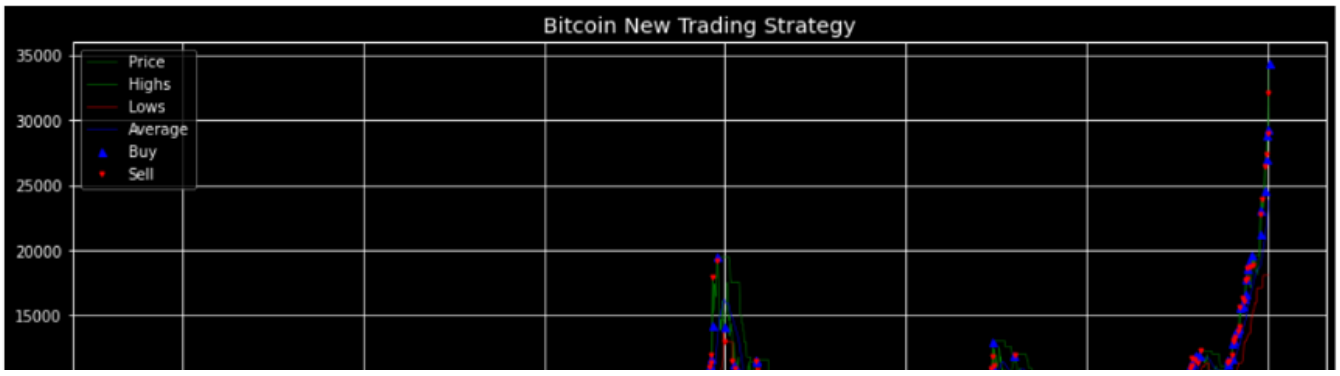
Custom Strategy:

Here, we have created a long signal when the price reaches the highest price for the last window size days (we have chosen 24).

```
def trading_strategy(df, window_size):
    newSt = df[['Close']]
    newSt['orders'] = 0
    newSt['High'] = df['Close'].shift(1).rolling(window=window_size).max()
    newSt['Low'] = df['Close'].shift(1).rolling(window=window_size).min()
    newSt['Avg'] = df['Close'].shift(1).rolling(window=window_size).mean()

    newSt['long_entry'] = df['Close'] > newSt.High
```

```
    newSt['short_exit'] = df['Close'] > newSt.Avg

    init=True
    position=0
    for k in range(len(newSt)):
        if newSt['long_entry'][k] and position==0:
            newSt.orders.values[k] = 1
            position=1
        elif newSt['short_entry'][k] and position==0:
            newSt.orders.values[k] = -1
            position=-1
        elif newSt['short_exit'][k] and position>0:
            newSt.orders.values[k] = -1
            position = 0
        elif newSt['long_exit'][k] and position < 0:
            newSt.orders.values[k] = 1
            position = 0
        else:
            newSt.orders.values[k] = 0

    return newSt
```

```
series = trading_strategy(df, 24)
buys = series.loc[ts['orders'] == 1]
sells = series.loc[ts['orders'] == -1]

fig = plt.figure(figsize=(15,6))
plt.plot(df["Close"], color='g', lw=.5); plt.plot(series["High"], color='g', lw=.5)
plt.plot(series["Low"], color='r', lw=.5); plt.plot(series["Avg"], color='b', lw=.5)
plt.plot(buys.index, series.loc[buys.index]['Close'],'^', markersize=5, color='blue', label='Buy')
plt.plot(sells.index, series.loc[sells.index]['Close'],'v', markersize = 3, color='r', label = 'Sell')
plt.legend(["Close price","High price","Low price","Average price","Buy","Sell"])
plt.title("Bitcoin New Trading Strategy"); plt.show()
```

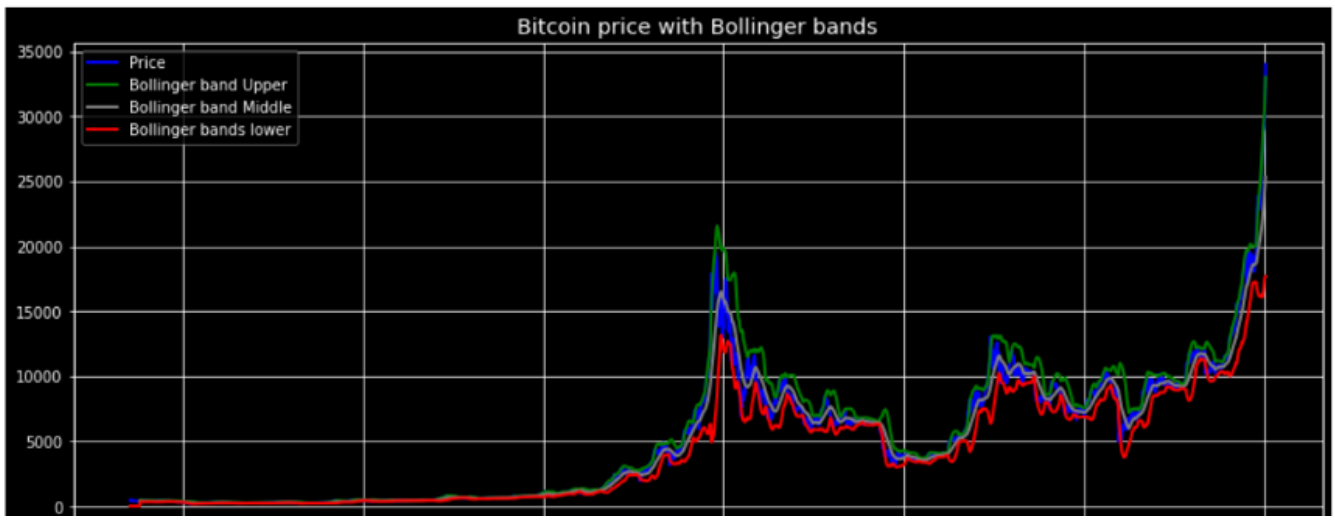



```
1   series.orders.value_counts()
```

```
 0    1670
 1     316
-1     315
Name: orders, dtype: int64
```

Bollinger Band strategy:

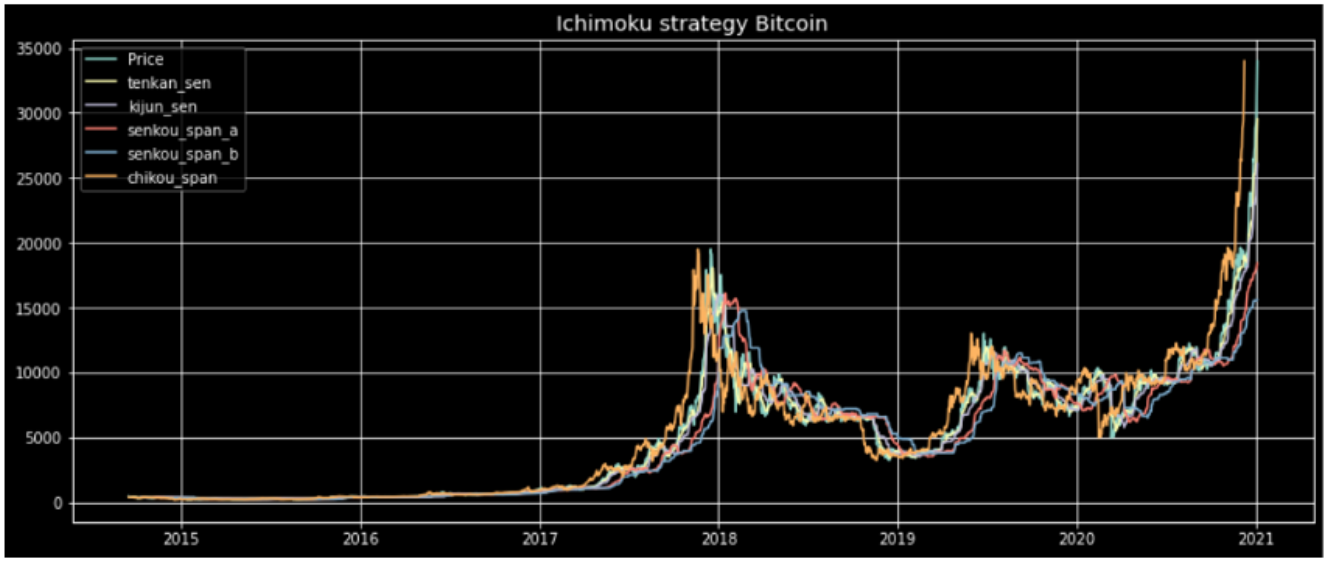
```
window = 21; no_of_std = 2

rolling_mean = df.Close.rolling(window).mean(); rolling_std = df.Close.rolling(window).std()

df['bb_up'] = (rolling_mean + (rolling_std* no_of_std)).fillna(0.0)
df['bb_middle'] =  df['Close'].rolling(window).mean()
df['bb_low'] = (rolling_mean - (rolling_std* no_of_std)).fillna(0.0)

fig = plt.figure(figsize = (15,6))
plt.plot(df.Close, color ='b', lw=2.); plt.plot(df.bb_up, color='g', lw=2.)
plt.plot(df.bb_middle, color='gray', lw=2. ); plt.plot(df.bb_low, color ='r', lw=2.)

plt.legend(["Price","Bollinger band Upper", "Bollinger band Middle","Bollinger bands lower"])
plt.title("Bitcoin price with Bollinger bands")

plt.show()
```



Here, the combination of standard deviations and daily look back periods are crucial to have an effective strategy.

Ichimoku trading strategy:

```
plt.figure(figsize= (15,6))
plt.plot(df.Close)
plt.plot(df.tenkan_sen)
plt.plot(df.kijun_sen)
plt.plot(df.senkou_span_a)
plt.plot(df.senkou_span_b)
plt.plot(df.chikou_span)
plt.legend(["Price","tenkan_sen", 'kijun_sen','senkou_span_a',
            'senkou_span_b','chikou_span'])
plt.title('Ichimoku strategy Bitcoin')
plt.show()
```



Stochastic Oscillator:

$\%K = 100(C - L14) / (H14 - L14)$

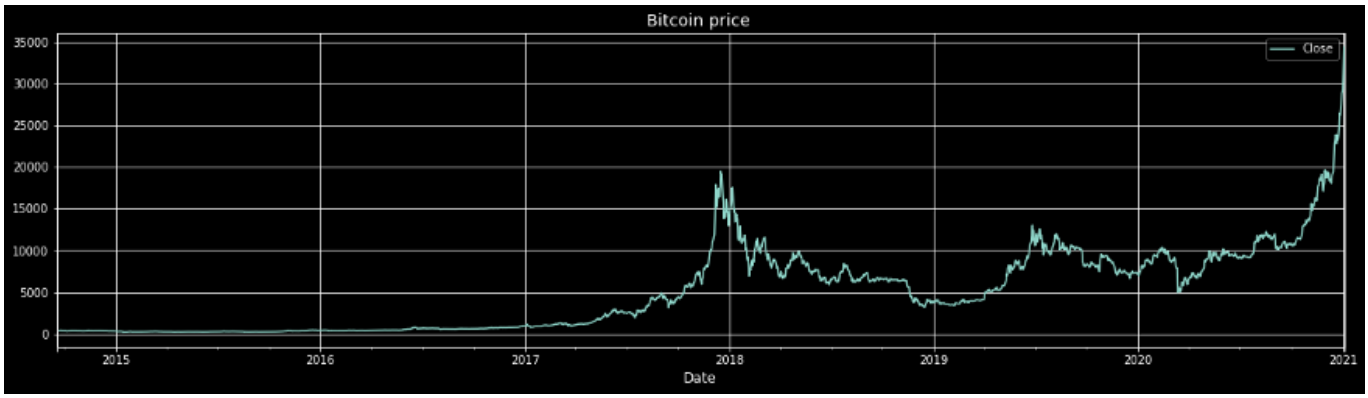
Where:

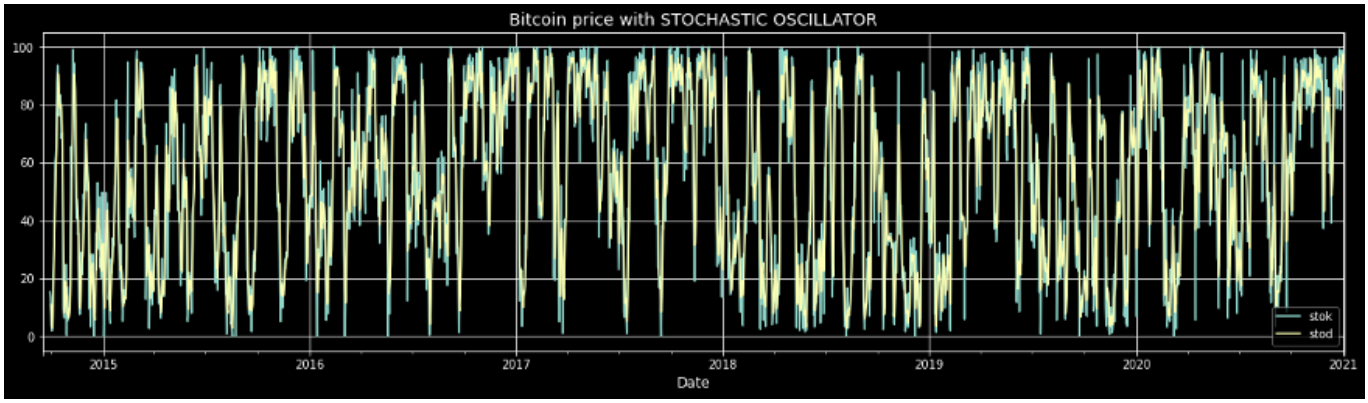
- C = the most recent closing price

- %K= the current market rate for the currency pair
- %D = 3-period moving average of %K

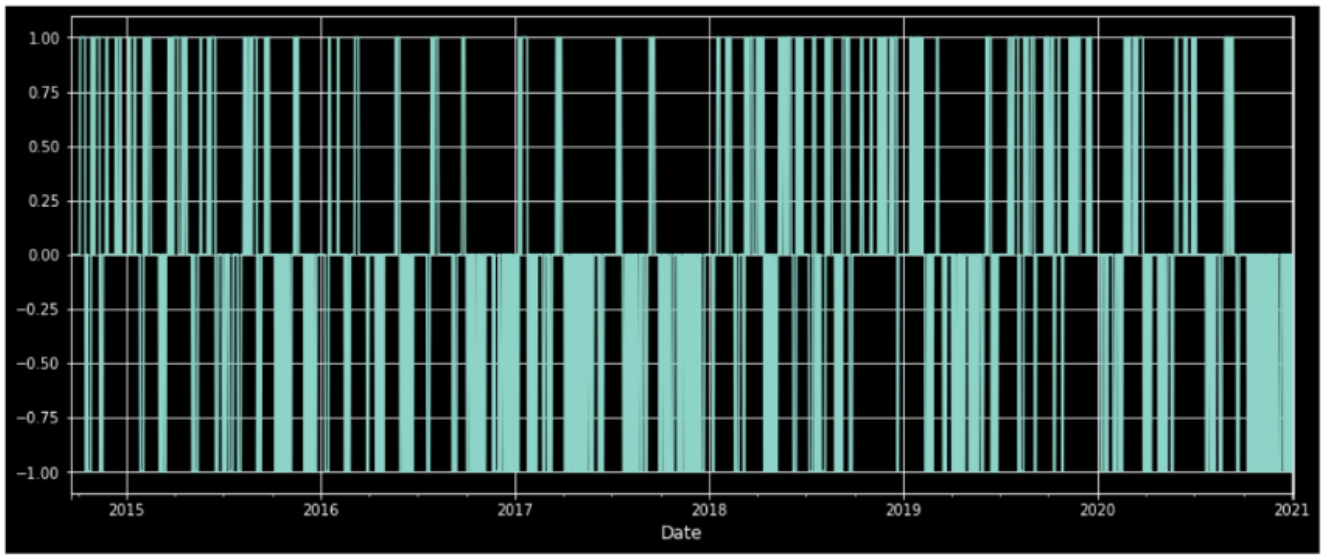
The general theory for this indicator is that in a upward trending scenario, prices will close near the high, and in a downward trending case, prices close near the low. Transaction signals are created when the %K crosses through a three-period moving average, which is called the %D.

```
n=14
df['stok'] = ((df.Close - df.Low.rolling(n).min()) / (df.High.rolling(n).max() - df.Low.rolling(n).min())) * 100
df['stod'] = df['stok'].rolling(3).mean()
df.plot(y=['Close'], figsize = (20, 5))
plt.title("Bitcoin price ")
df.plot(y=['stok', 'stod'], figsize = (20, 5))
plt.title("Bitcoin price with STOCHASTIC OSCILLATOR")

plt.show()
```

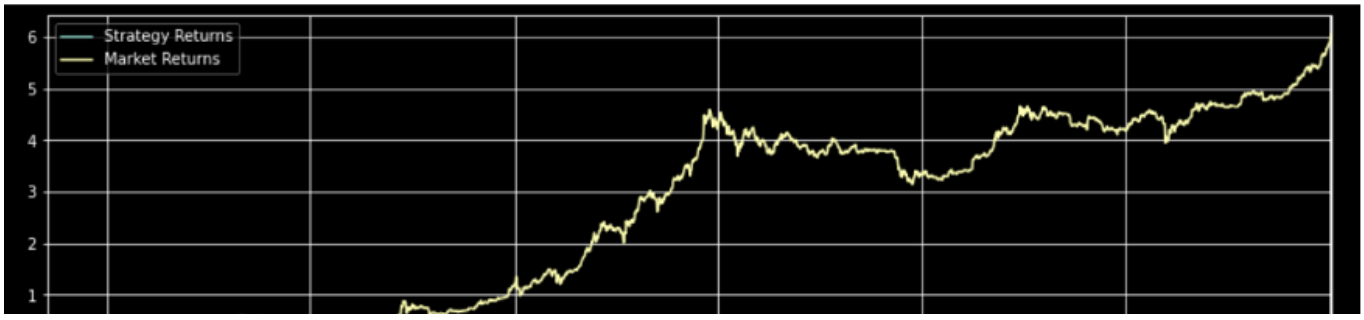




```
df['etnter_sell'] = ((df['stok'] < df['stod']) & (df['stok'].shift(1) > df['stod'].shift(1))) & (df['stod'] > 75)
df['exit_sale'] = ((df['stok'] > df['stod']) & (df['stok'].shift(1) < df['stod'].shift(1)))
df['short'] = np.nan
df.loc[df['etnter_sell'],'short'] = -1
df.loc[df['exit_sale'],'short'] = 0
# initial position on day 1 to flat
df['short'][0] = 0
df['short'] = df['short'].fillna(method='pad')
df['enter_buy'] = ((df['stok'] > df['stod']) & (df['stok'].shift(1) < df['stod'].shift(1))) & (df['stod'] < 25)
df['exit_buy'] = ((df['stok'] < df['stod']) & (df['stok'].shift(1) > df['stod'].shift(1)))
df['long'] = np.nan
df.loc[df['enter_buy'],'long'] = 1
df.loc[df['exit_buy'],'long'] = 0
df['long'][0] = 0
# Forward fill the position column to represent the holding of positions through time
df['long'] = df['long'].fillna(method='pad')
# Add Long and Short positions together to get final strategy position (1 for long, -1 for short and 0 for flat)
df['position'] = df['long'] + df['short']
df['position'].plot(figsize=(15,10))
plt.show()
```




```
df['Market Returns'] = df['Close'].pct_change()
df['Strategy Returns'] = df['Market Returns'] * df['Position'].shift(1)
df[['Strategy Returns','Market Returns']].cumsum().plot(figsize=(15,5))
plt.show()
```



Here, we see the overall return is negative and we could have done much better by just buying and holding Bitcoin.

Key takeaways:

Here, we have covered the fundamentals of signal generation using technical analysis. Once we are thorough with the fundamentals, we can work around to come us with a buy/sell strategy and test the performance to check the profitability.

I can be reached here.

Trading Signals | Technical Analysis | Time Series Data | Visualization