Noon Delta Signal Analysis on DAX

It's 4 months; 4 months that my friend Paolo *warmly invites* me for statistically testing and characterizing this strategy. Something he heard from somebody else; a gossip traveling the seas of trading forums and chats for years probably. I guess that Paolo now could be happy, but always possible undetected bugs in my code. More than 82% successful, simple strategy! Or … not?

Index

1. Versions

- V4, 2024–04–14: ML studies
- V3, 2024–03–10: Optimizations and implementation of statistical suggestions
- V2, 2024–03–18: statistical analysis
- V1, 2024–02–04: first version

2. Preface

Some expectations you may have on this publication

- Find new ways for trading DAX or defining a new powerful confirmation signal for you base and custom strategies
- Understand how deep statistical investigation can help to formalize a trading strategy
- Be exposed to different approaches and inspired by new ideas
- Contribute to the further development of the described approach
- Create a community of motivated people working on this and similar challenges

What you won’t find for sure in this publication

- You won't find a guaranteed, foolproof science for successful trading
- You won't find material without errors; despite double-checking the code and addressing initial bugs in the resulting data, there may still be some oversights
- You won't find a comprehensive course on Python, its libraries and trading

Prerequisites

- While I will publish my *Jupyter Notebook* code, it's essential to note that embarking on this journey will be of limited value if you don't already possess a solid understanding of the world of trading.
- Furthermore, having some background in mathematics and statistics is crucial for a comprehensive understanding. This doesn't necessitate a degree in mathematics or statistics, but it does require a foundation in advanced calculus and basic statistics, which can be acquired also through disciplines like engineering, physics, architecture, biology, economics, and similar fields.

My expectations

- Share my results with the hope that they can be valuable to others for defining they or own conscious and rational way for a successful trading
- Welcome constructive feedbacks to potentially improve and refine these results.
- Establish a motivated community of people from around the world collaborating to enhance trading capabilities through the application of mathematics, statistics, and AI. I've created a Telegram group, and you are encouraged to consider joining it; it’s completely new so you will find a white paper for which contributing with enjoyable and valuable content.

Welcome everybody to this reading, my name is Federico, I hold a couple of engineering master degrees and I am NOT a *data scientist*.

Code
Small portions of code are copied in this post text in a highlighted colored box; all the rest of the code is available in my Jupiter Notebook that you can find at this URL: https://github.com/fede72bari/DAX-Noon-Delta-Strategy-Jupyter-Notebook

Time
Since the analysis is based on a European index, the time reference used is Central European Time (CET).

Uncorrelations with my job
As I mentioned, I am not a data scientist or developer; this post summarizes some studies I have conducted in my spare time, reflecting my amateur interest and hobbyist pursuits. None of these studies or results are related to my primary job or professional skills.

3. Noon Signal Description

Here is a simple description of the noon Signal calculation, as understood from the N-th description received from Paolo during a late and tired phone call:

- Between 9:00 and 12:00 each day, detect the maximum and minimum prices.
- Calculate the difference between the maximum and minimum prices (delta).
- Divide the delta by 3 to get delta_div_3.
- Calculate two new thresholds:
- Subtract delta_div_3 from the maximum (max_minus_delta).
- Add delta_div_3 to the minimum (min_plus_delta).
- Based on the opening price at noon:
    - If the opening price is between max_minus_delta and the maximum: enter a long position, take profit at the maximum, set stop loss at min_plus_delta or the minimum.
    - If the opening price is between the minimum and min_plus_delta: enter a short position, take profit at the minimum, set stop loss at max_minus_delta or the maximum.

These were the notes used for the initial part of the subsequent analysis and development, at least until my next call with Paolo!

4. Libraries import

The main function responsible for enhancing the features is called *create_noon_strategy_statistics* and it accepts a dataset containing DAX index data (timestamp, OHLC prices), along with parameters to adjust the DAX Noon Delta Signal thresholds slightly from the original (max-min)/3 to search for an optimal setting. This function computes various KPIs, which can be categorized as follows:

1. KPIs from 9:00 to 12:00
2. KPIs from 12:00 to closing time
3. Ratios and deltas between them
4. Figures from previous days

I believe that the names I used for each KPI are self-explanatory. For more details, you can refer directly to the code in already mentioned Jupiter Notebook .

Additionally, if you need to use the standard indicator calculation library Talib in a Google Colab Notebook, here is the workaround I found for installing it:

```
Value url = 'https://anaconda.org/conda-forge/libta-lib/0.4.0/download/linux-64/libta-lib-0.4.0-h166bdaf_1.tar.bz2'
!curl -L $url | tar xj -C /usr/lib/x86_64-linux-gnu/ lib --strip-components=1
url = 'https://anaconda.org/conda-forge/ta-lib/0.4.19/download/linux-64/ta-lib-0.4.19-py310hde88566_4.tar.bz2'
!curl -L $url | tar xj -C /usr/local/lib/python3.10/dist-packages/ lib/python3.10/site-packages/talib --strip-components=3
```

5. Data sources

For the initial phase of the strategy analysis, I utilized a 1-minute timeframe dataset of the DAX index covering the period from 2020 to early 2021, exported from MultiChart. The subsequent phase of this analysis involves an extended dataset spanning from 2000 to 2024, captured at a 5-minute timeframe. For your reference, the dataset for the extended period was sourced from www.backtestmarket.com. Below is the code snippet for importing this latter dataset.

```
df = pd.read_csv(r'D:\Dropbox\TRADING\DATA\dax-5m.csv', delimiter=';')
df.columns = ['Date', 'Time', 'Open', 'High', 'Low', 'Close', 'TotalVolume']

df['datetime'] = pd.to_datetime(df['Date'] + ' ' + df['Time'], dayfirst=True)

chicago_timezone = pytz.timezone('America/Chicago')

# Converte la colonna datetime alla time zone di Chicago
df['datetime_chicago'] = df['datetime'].dt.tz_localize(chicago_timezone)

# Specifica la time zone di Roma
rome_timezone = pytz.timezone('Europe/Rome')

# Converte la colonna datetime alla time zone di Roma
df['datetime_rome'] = df['datetime_chicago'].dt.tz_convert(rome_timezone)
df['datetime'] = df['datetime_rome'].dt.tz_localize(None)

df['Volume'] = df['TotalVolume']

# corrupted data on this dates
date_threshold = pd.to_datetime('2001-10-08').date()
df = df[df['datetime'].dt.date != date_threshold]
date_threshold = pd.to_datetime('2004-01-12').date()
df = df[df['datetime'].dt.date != date_threshold]
date_threshold = pd.to_datetime('2004-01-13').date()
df = df[df['datetime'].dt.date != date_threshold]


print("Dataframe column names:")
print(df.columns)
```

As you can infer from the code, there are at least 3 days with seemingly corrupted or incomplete data, which was causing issues during feature calculations. To address this, I chose to skip these days in order to skip the problems.

6. Trading this strategy directly

One initial approach could be to implement this strategy directly for trading. This involves initiating a long position at the noon open price if the strategy indicates that the maximum price from 9:00 to 12:00 will likely be reached and potentially surpassed. Conversely, a short position would be opened at the noon open price if the strategy suggests the opposite scenario. In all other cases—such as when the noon open price falls within the middle band or exceeds the minimum or maximum thresholds—it would be prudent to remain neutral (flat). Let's proceed by implementing the function to generate the *df_stats* working dataframe.

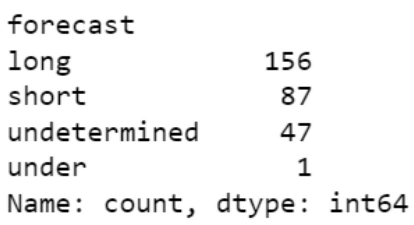
```
df_stats = create_noon_strategy_statistics(df)
```

Preliminary analysis: is it worth to continue? A glance to statistics

Before coding the back-test function, let's take a look at the occurrences of all possible scenarios:

```
occurrences = df_stats['forecast'].value_counts()
display(occurrences)
```

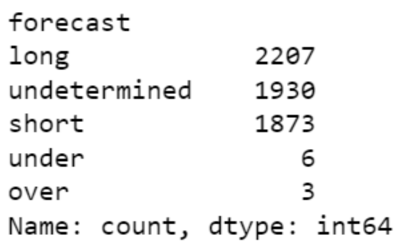
For the one-year long, 1-minute time series dataset, we have the following occurrences of the five cases based on the described strategy:



```
forecast
long            156
short            87
undetermined     47
under             1
Name: count, dtype: int64
```

Figure 1 - Statistics of DAX Noon Delta Signal over 1-year long dataset, 1 minutes timeframe

For the 24-year long, 5-minute time series dataset, we observe the occurrences of the described strategy's cases as follows:



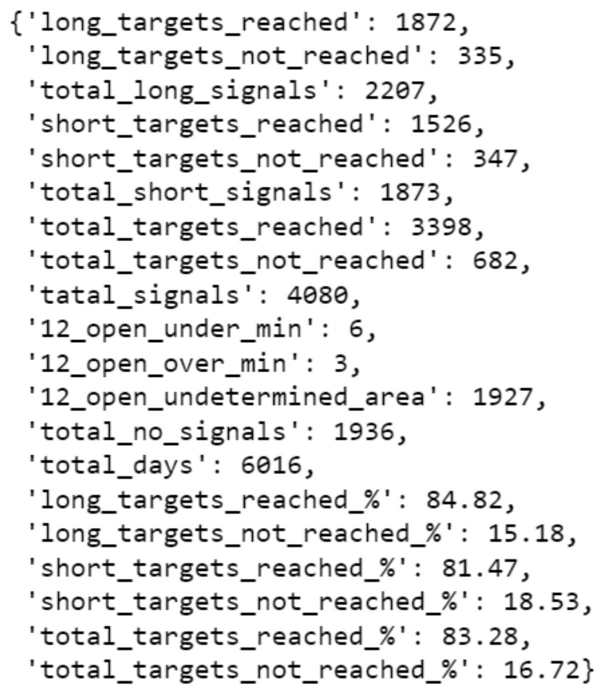
```
forecast
long            2207
undetermined    1930
short           1873
under              6
over               3
Name: count, dtype: int64
```

Figure 2 - Statistics of DAX Noon Delta Signal over 24-years long dataset, 5 minutes timeframe

Even though the long trend forecast occurs more frequently compared to the other forecast types, we can conclude that forecasts for long, undetermined, and short scenarios each account for roughly one-third of the cases. However, how accurate are these forecasts? Before coding a trading backtest, it would be beneficial to assess the rough success or failure probabilities of the strategy. The following code accomplishes this purpose.

```
def targets_reached_resutls(df_stats):
    # Conta le occorrenze di True e False per 'long'
    long_gain_counts = df_stats[df_stats['forecast'] == 'long']['gain'].value_counts()

    # Conta le occorrenze di True e False per 'short'
    short_gain_counts = df_stats[df_stats['forecast'] == 'short']['gain'].value_counts()

    results = {}

    results['long_targets_reached'] = long_gain_counts.get(1, 0)
    results['long_targets_not_reached'] = long_gain_counts.get(-1, 0)
    results['total_long_signals'] = results['long_targets_reached'] + results['long_targets_not_reached']

    results['short_targets_reached'] = short_gain_counts.get(1, 0)
    results['short_targets_not_reached'] = short_gain_counts.get(-1, 0)
    results['total_short_signals'] = results['short_targets_reached'] + results['short_targets_not_reached']

    results['total_targets_reached'] = results['long_targets_reached'] + results['short_targets_reached']
    results['total_targets_not_reached'] = results['long_targets_not_reached'] + results['short_targets_not_reached']
    results['tatal_signals'] = results['total_targets_reached'] + results['total_targets_not_reached']


    results['12_open_under_min'] = df_stats[df_stats['forecast'] == 'under'].shape[0]
    results['12_open_over_min'] = df_stats[df_stats['forecast'] == 'over'].shape[0]
    results['12_open_undetermined_area'] = df_stats[df_stats['forecast'] == 'undetermined'].shape[0]
    results['total_no_signals'] = results['12_open_under_min'] + results['12_open_over_min'] + results['12_open_undetermined_area']

    results['total_days'] = results['tatal_signals'] + results['total_no_signals']

    results['long_targets_reached_%'] = round(results['long_targets_reached'] / results['total_long_signals'] * 100 , 2)
    results['long_targets_not_reached_%'] = round(results['long_targets_not_reached'] / results['total_long_signals'] * 100 , 2)

    results['short_targets_reached_%'] = round(results['short_targets_reached'] / results['total_short_signals'] * 100 , 2)
    results['short_targets_not_reached_%'] = round(results['short_targets_not_reached'] / results['total_short_signals'] * 100 , 2)

    results['total_targets_reached_%'] = round(results['total_targets_reached'] / results['tatal_signals'] * 100 , 2)
    results['total_targets_not_reached_%'] = round(results['total_targets_not_reached'] / results['tatal_signals'] * 100 , 2)

    return results

targets_stats = targets_reached_resutls(df_stats)
display(targets_stats)
```

Running the code on the 24 years long dataset returns these results:



```
{'long_targets_reached': 1872,
 'long_targets_not_reached': 335,
 'total_long_signals': 2207,
 'short_targets_reached': 1526,
 'short_targets_not_reached': 347,
 'total_short_signals': 1873,
 'total_targets_reached': 3398,
 'total_targets_not_reached': 682,
 'tatal_signals': 4080,
 '12_open_under_min': 6,
 '12_open_over_min': 3,
 '12_open_undetermined_area': 1927,
 'total_no_signals': 1936,
 'total_days': 6016,
 'long_targets_reached_%': 84.82,
 'long_targets_not_reached_%': 15.18,
 'short_targets_reached_%': 81.47,
 'short_targets_not_reached_%': 18.53,
 'total_targets_reached_%': 83.28,
 'total_targets_not_reached_%': 16.72}
```

Figure 3 - Rough DAX Noon Delta Signal success statistics over 24 years of data, 5 minutes timeframe

We received a total of 4080 signals, broken down as follows:

- 2207 were long signals, successful in almost 85% of cases.
- 1873 were short signals, successful in more than 83% of cases.
- In 1936 instances, the opening price at 12:00 fell into the undetermined middle area.
- In just 9 cases, the opening price at 12:00 exceeded or fell below the minimum or maximum of the previous 3 hours.

Can we conclude from these statistics that directly trading the DAX Noon Delta Signal will lead to a positive and exciting equity curve? Absolutely not! Despite the promising outcomes of this initial analysis, which indeed confirm that the minimum and maximum values established between 9:00 and 12:00 are often revisited during the afternoon session of the same trading day, the direction of equity trend—whether positive or negative—is influenced by numerous other factors. These include:

- The distance between the noon opening price and the take profit target.
- Stop loss policies.
- Market price action behavior. We cannot predict in advance whether the price will revisit the minimum or maximum in the afternoon as part of a favorable trend continuation or after a swing in the opposite direction. If the swing exceeds the opposite threshold of the minimum or maximum, the stop loss condition will be triggered, even if the forecast was correct.

Back-testing the base strategy and some variations

For back-testing the strategy, I have used a library of mine; in case you would like to replicate my results (or unconfirm them) by your own, you may use one of the many back-testing libraries written in Python that you can find for freeware in GitHub. I will not share the code of my library also to stimulate different approaches and tools utilization in order to check if there could be any misalignment with my results for potential still hidden bugs. Anyway, I will provide here following the code of the callback function that define the trading strategy.

Basic trading strategy

The initial version of the strategy does not incorporate a formal stop-loss mechanism. This strategy can be outlined as follows:

- Enter at the Open price at 12:00 with a long position if the forecast was "long" or with a short position if the forecast was "short"; remain neutral (flat) in all other cases.
- Close the position at a profit if the price reaches the high (in the case of a "long" forecast) or the low (in the case of a "short" forecast).
- Close the position at a loss if the price touches the opposite threshold.
- If a position remains open at the end of the day, close it at the closing price of the day.

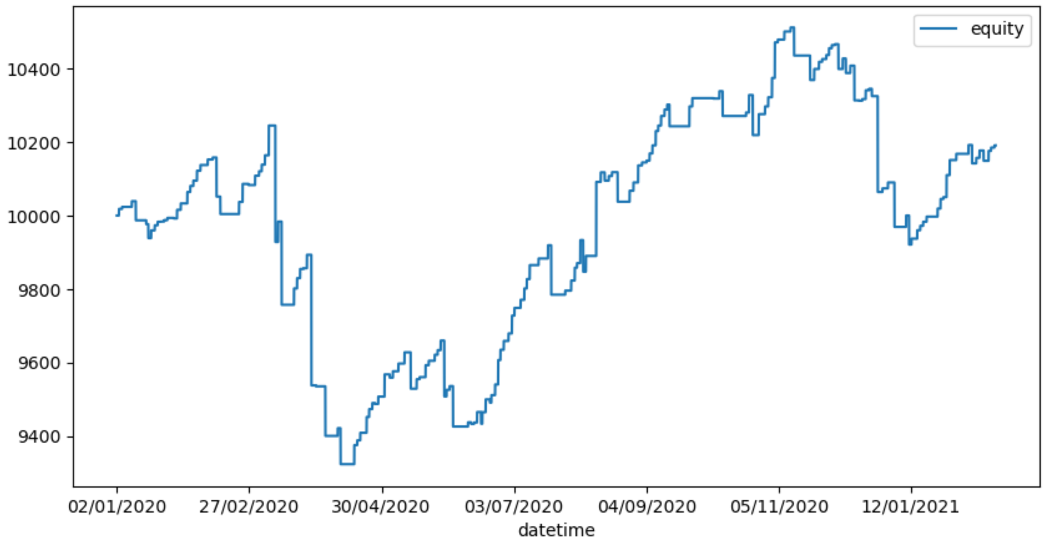
Below is the code of the callback function that defines this straightforward strategy.

```
# trading strategy callback functions
```

```
def simple_moon_DAX_stretgy(self, data, args, index, current_open_volumes):

    current_row = data[data.index == index]

    current_date = current_row['datetime'].dt.date.iloc[0]
    current_date_str = current_date.strftime('%Y-%m-%d')
    current_hour = current_row['datetime'].dt.hour.iloc[0]
    current_minute = current_row['datetime'].dt.minute.iloc[0]

    current_High = current_row['High'].iloc[0]
    current_Low = current_row['Low'].iloc[0]

    current_date_obj = datetime.strptime(current_date_str, '%Y-%m-%d')
    current_date_formatted = current_date_obj.strftime('%d/%m/%Y')

    if( ('current_date_formatted' not in self.custom_dict) | ('current_date_data' not in self.custom_dict) ):

        current_date_data = data.loc[data['Date'] == current_date_formatted]

        self.custom_dict = {'current_date_formatted': current_date_formatted,
                            'current_date_data': current_date_data
                           }

    elif(current_date_formatted != self.custom_dict['current_date_formatted']):


        current_date_data = data.loc[data['Date'] == current_date_formatted]

        self.custom_dict = {'current_date_formatted': current_date_formatted,
                            'current_date_data': current_date_data
                           }

    else:

        current_date_data = self.custom_dict['current_date_data']


    # next is last element of the day, close at next Open price
    if(index == current_date_data.index[-2]):


        if(current_open_volumes > 0):

            print('Close long position last bar.')
            return -1

        elif(current_open_volumes < 0):

            print('Close short position last bar.')
            return 1

        else:

            return 0
    # it is 12:00! track open price, calculate thresholds and set stop loss and take profit; in case open the position
    if( (data.iloc[index]['datetime'].hour == 12) &
        (data.iloc[index]['datetime'].minute == 0) ):

        open_at_12 = data.iloc[index]['Open']

        from_9_to_12_data = current_date_data[ (current_date_data['datetime'].dt.hour >= 9) &
(current_date_data['datetime'].dt.hour < 12) ]

        from_12_data = current_date_data[ current_date_data['datetime'].dt.hour >= 12 ]

        max_9_to_12 = from_9_to_12_data['High'].max()
        min_9_to_12 = from_9_to_12_data['Low'].min()

        delta_9_to_12 = max_9_to_12 - min_9_to_12
        delta_9_to_12_div_3 = delta_9_to_12 / 3

        max_9_to_12_minus_delta = max_9_to_12 - delta_9_to_12_div_3
        min_9_to_12_plus_delta = min_9_to_12 + delta_9_to_12_div_3

        self.max_9_to_12 = max_9_to_12
        self.min_9_to_12 = min_9_to_12
        self.max_9_to_12_minus_delta = max_9_to_12_minus_delta
        self.min_9_to_12_plus_delta = min_9_to_12_plus_delta

        max_from_12_data = from_12_data['High'].max()
        min_from_12_data = from_12_data['Low'].min()

        print(f'\topen_at_12 = {open_at_12}')
        print(f'\tmax_from_12_data = {max_from_12_data}')
        print(f'\tmin_from_12_data = {min_from_12_data}')
        print(f'\tmax_9_to_12 = {max_9_to_12}')
        print(f'\tmax_9_to_12_minus_delta = {max_9_to_12_minus_delta}')
        print(f'\tmin_9_to_12_plus_delta = {min_9_to_12_plus_delta}')
        print(f'\tmin_9_to_12 = {min_9_to_12}')


        # long condition
        if( (open_at_12 >= max_9_to_12_minus_delta) & (open_at_12 < max_9_to_12) ):

            self.take_profit = max_9_to_12 - next_open_price
            self.stop_loss = self.take_profit * 2 #self.take_profit * 3 # self.take_profit  # self.take_profit * 2 # 93 #
abs(min_9_to_12 - next_open_price)
            self.take_profit *= 1.5

            return 1

        # short condition
        elif( (open_at_12 <= min_9_to_12_plus_delta ) & (open_at_12 > min_9_to_12) ):

            self.take_profit = next_open_price - min_9_to_12
            self.stop_loss = self.take_profit * 2 #self.take_profit * 3 # self.take_profit  # self.take_profit * 2 # 93 #
abs(min_9_to_12 - next_open_price)
```

```
            self.take_profit *= 1.5

            return -1

    return 0
```

The charts below provide a summary of the back-test results for this strategy.



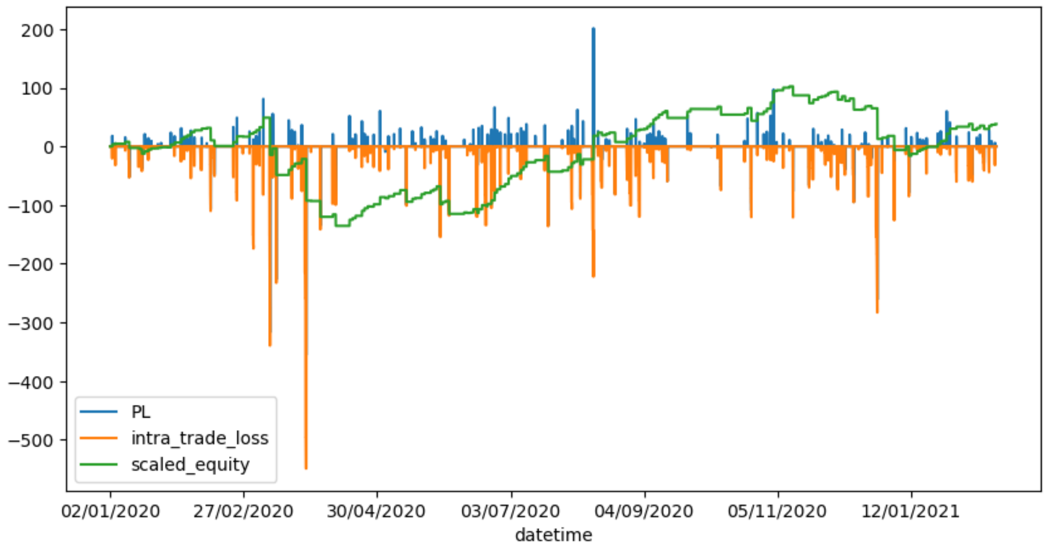


Figure 4 - DAX Noon Delta Signal backtest, 1 year data - A) Equity curve, B) PL and intra trade losses

Upon first inspection of the charts, it appears that the *profit and loss* over a one-year period is relatively flat, with both the maximum accumulated loss and maximum profit around 500 euros. Delving deeper into the equity curve, two significant observations emerge:

- The number of successful trades, indicated by small incremental steps, significantly exceeds the number of trades closed with a loss.
- Despite the higher frequency of successful trades, the overall loss from losing trades equals the gain from winning trades due to a higher average loss per losing trade.

This phenomenon can be attributed to the strategy's entry criteria, where trades are initiated only if the noon open price is within one-third of the range established between 9:00 and noon (used as the take profit level), resulting in smaller profits relative to the more distant stop loss positioned at the opposite threshold. Essentially, the stop loss condition is not optimally positioned, suggesting room for improvement.

The orange lines in the second chart above represent "intra-trade loss," which signifies the maximum loss incurred before a position is closed either with a gain or loss. By comparing these lines with the blue gain lines, one can visually identify potentially more profitable stop loss positions.

These insights are corroborated by the trading statistics, reinforcing the need to refine the stop loss strategy for enhanced performance.

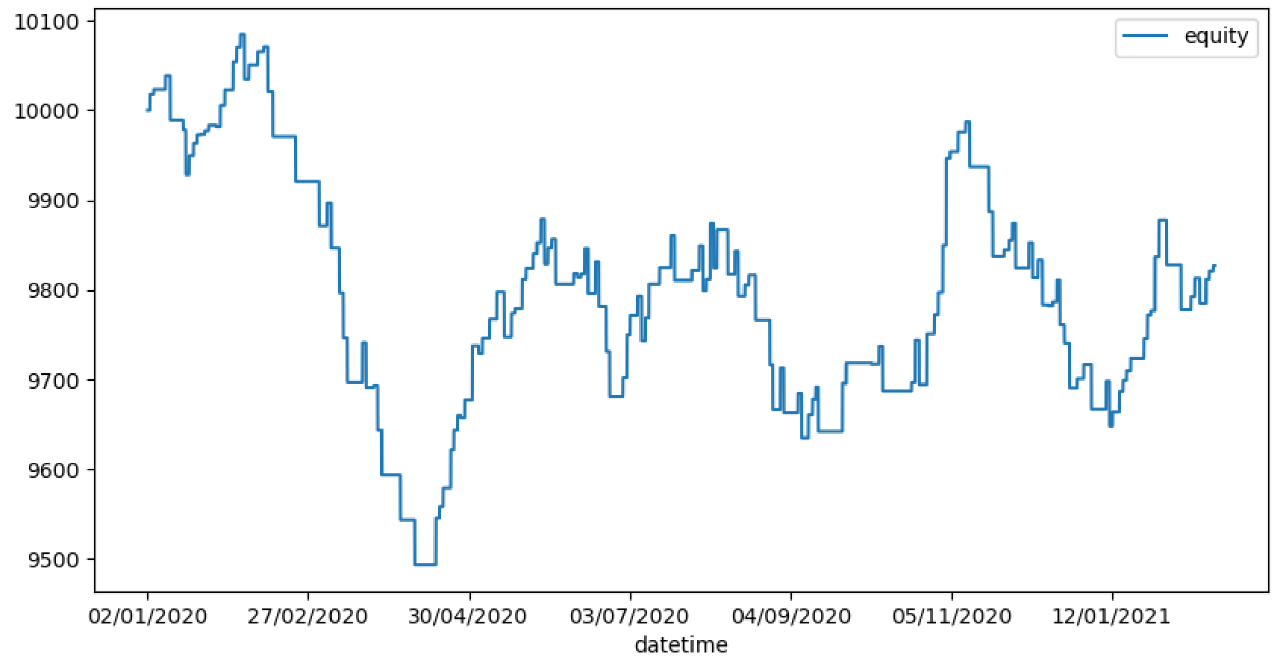
```
{'ticker': 'DAX',
 'start_date': '02/01/2020',
 'end_date': '22/02/2021',
 'number_of_periods': 333133,
 'initial_equity': 10000,
 'final_equity': 10191.0,
 'final_pl': 191.0,
 'number_of_operations': 179,
 'sum_in_trade_periods_stats': 25226,
 'mean_in_trade_periods_stats': 140.93,
 'std_in_trade_periods_stats': 173.08,
 'max_in_trade_periods_stats': 600,
 'min_in_trade_periods_stats': 1,
 'sum_flat_periods_stats': 306773,
 'mean_flat_periods_stats': 1704.29,
 'std_flat_periods_stats': 1359.28,
 'max_flat_periods_stats': 8836,
 'min_flat_periods_stats': 423,
 'total_profit': 3467.0,
 'number_profit_operations': 138,
 'percent_profit_operations': 77.09,
 'average_profit_trade': 25.1231884057971,
 'max_profit_single_operation': 201.5,
 'number_max_consecutive_profits': 14,
 'max_profit_consecutive_operations': 428.5,
 'number_profit_longs': 79,
 'percent_long_profit': 44.13,
'number_loss_longs': 21,
 'percent_long_loss': 11.73,
 'total_loss': -3276.0,
 'number_loss_operations': 41,
 'percent_loss_operations': 22.91,
 'average_loss_trade': -79.90243902439025,
 'max_loss_single_operation': -355.5,
 'number_max_consecutive_losses': 3,
 'max_loss_consecutive_operations': -493.0,
'number_profit_shorts': 59,
 'percent_short_profit': 32.96,
 'number_loss_shorts': 20,
 'percent_short_loss': 11.17,
 'mean_intra_trade_loss': -43.69832402234637,
 'std_intra_trade_loss': 64.00828200877325,
 'max_intra_trade_loss': 0.0,
 'min_intra_trade_loss': -549.0,
 'mean_intra_trade_loss_gain_trade': -27.67391304347826,
 'std_intra_trade_loss_gain_trade': 33.75397528218043,
 'max_intra_trade_loss_gain_trade': 0.0,
 'min_intra_trade_loss_gain_trade': -222.0,
 'mean_intra_trade_loss_lose_trade': -97.63414634146342,
 'std_intra_trade_loss_lose_trade': 102.28398117436596,
 'max_intra_trade_loss_lose_trade': 0.0,
 'min_intra_trade_loss_lose_trade': -549.0,
 'number_long_stop_losses': 16,
 'number_short_stop_losses': 7}
```

Stop Loss at 50 euro

Before implementing more sophisticated *stop-loss* positioning strategies, a preliminary and simpler approach could involve setting stop losses to a fixed value. This value could be determined as follows:

- Setting the stop loss to the mean intra-trade unrealized losses for winning trades plus its standard deviation (50 USD), or
- Setting the stop loss to twice its standard deviation (93 USD).

Fortunately, both of these values remain below the mean intra-trade unrealized losses (97 USD) of losing trades. Below are the results of the initial attempt to reduce the *stop-loss* condition.



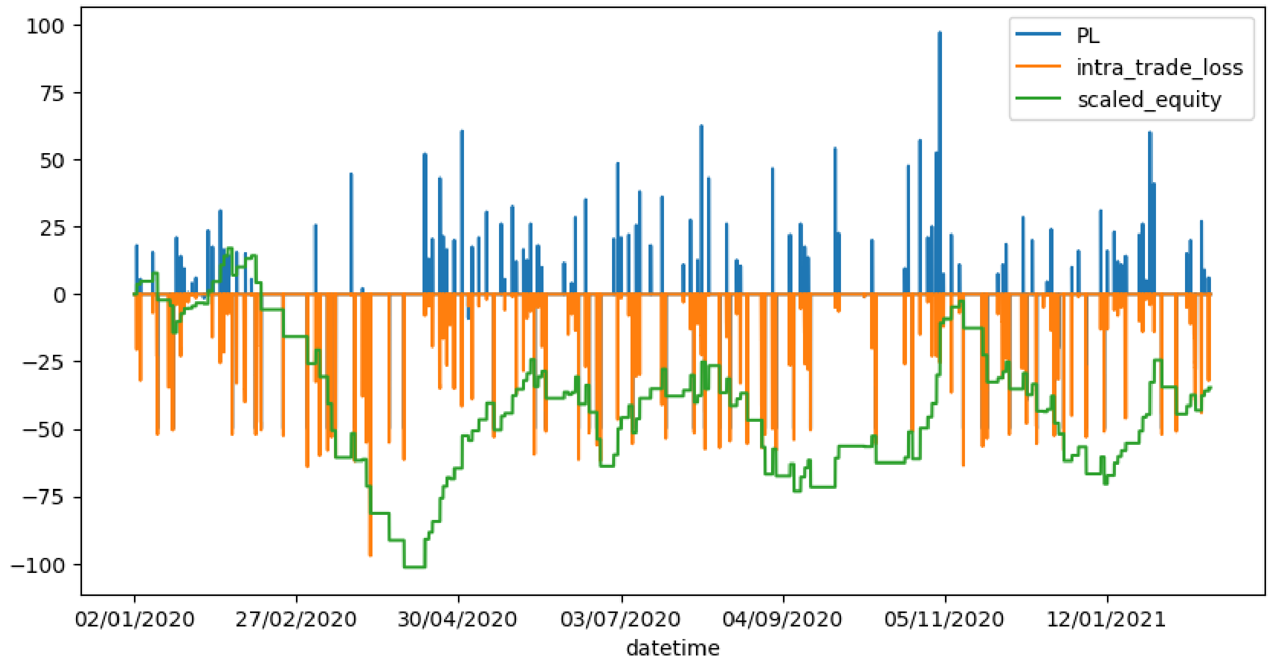


Figure 5 - DAX Noon Delta Signal backtest, 1 year data, stop loss at 1st STD of average intra-trade unrealized losses - A) Equity curve, B) PL and intra trade losses

And here the statistics:

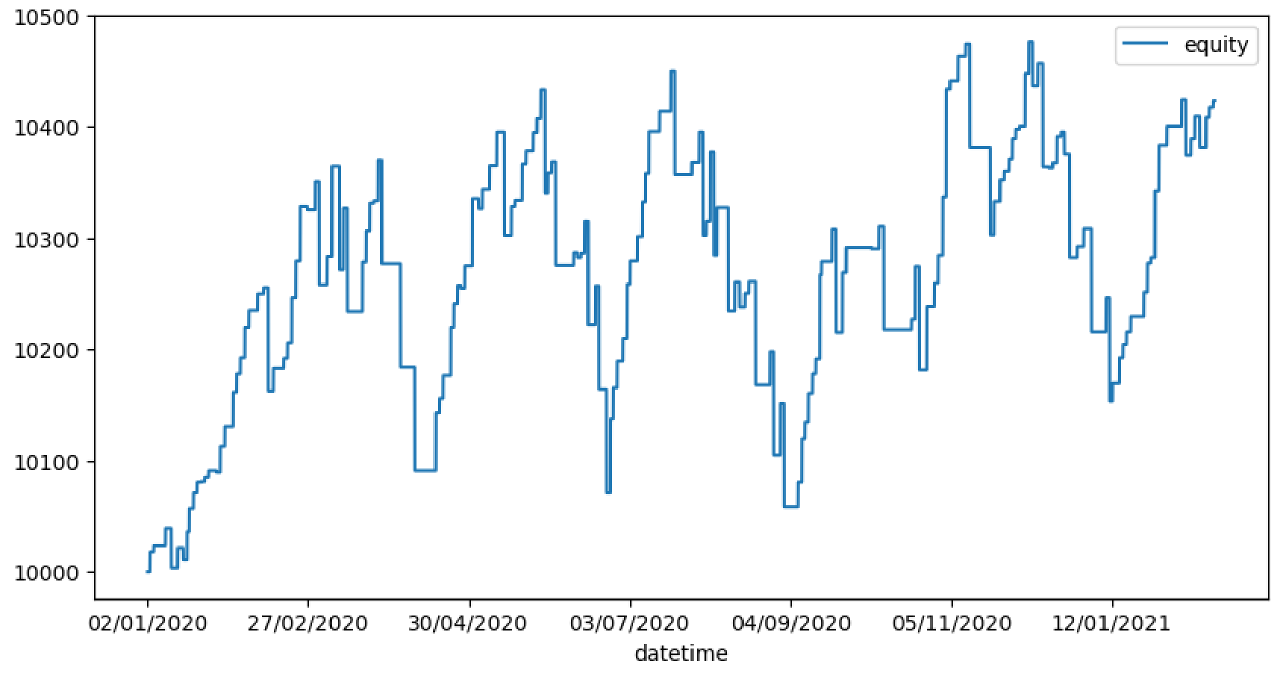
{'ticker': 'DAX', 'start_date': '02/01/2020',	'number_loss_longs': 32, 'percent_long_loss': 19.88,

```
 'end_date': '22/02/2021',
 'number_of_periods': 333133,
 'initial_equity': 10000,
 'final_equity': 9826.5,
 'final_pl': -173.5,
 'number_of_operations': 161,
 'sum_in_trade_periods_stats': 14735,
 'mean_in_trade_periods_stats': 91.52,
 'std_in_trade_periods_stats': 126.26,
 'max_in_trade_periods_stats': 600,
 'min_in_trade_periods_stats': 1,
 'sum_flat_periods_stats': 317264,
 'mean_flat_periods_stats': 1958.42,
 'std_flat_periods_stats': 1557.27,
 'max_flat_periods_stats': 8904,
 'min_flat_periods_stats': 508,
 'total_profit': 2344.0,
 'number_profit_operations': 103,
 'percent_profit_operations': 63.98,
 'average_profit_trade': 22.75728155339806,
 'max_profit_single_operation': 97.0,
 'number_max_consecutive_profits': 10,
 'max_profit_consecutive_operations': 293.0,
 'number_profit_longs': 58,
 'percent_long_profit': 36.02,
'percent_loss_operations': 36.02,
 'average_loss_trade': -43.4051724137931,
 'max_loss_single_operation': -50,
 'total_loss': -2517.5,
 'number_loss_operations': 58,
'number_max_consecutive_losses': 4,
 'max_loss_consecutive_operations': -200,
 'number_profit_shorts': 45,
 'percent_short_profit': 27.95,
 'number_loss_shorts': 26,
 'percent_short_loss': 16.15,
 'mean_intra_trade_loss': -28.369565217391305,
 'std_intra_trade_loss': 22.062023489126798,
 'max_intra_trade_loss': 0.0,
 'min_intra_trade_loss': -97.0,
 'mean_intra_trade_loss_gain_trade': -16.349514563106798,
 'std_intra_trade_loss_gain_trade': 14.014144631836821,
 'max_intra_trade_loss_gain_trade': 0.0,
 'min_intra_trade_loss_gain_trade': -50.0,
 'mean_intra_trade_loss_lose_trade': -49.71551724137931,
 'std_intra_trade_loss_lose_trade': 17.032248069153148,
 'max_intra_trade_loss_lose_trade': 0.0,
 'min_intra_trade_loss_lose_trade': -97.0,
 'number_long_stop_losses': 28,
 'number_short_stop_losses': 20}
```

The average loss for losing trades has decreased with this approach, but the increased number of losing trades has offset these gains, resulting in a slightly worse overall performance compared to the initial attempt.

Stop loss at 93 euro

Here is the chart depicting the performance of trading the basic DAX Noon Delta Signal with a stop-loss threshold set at 93 euros.



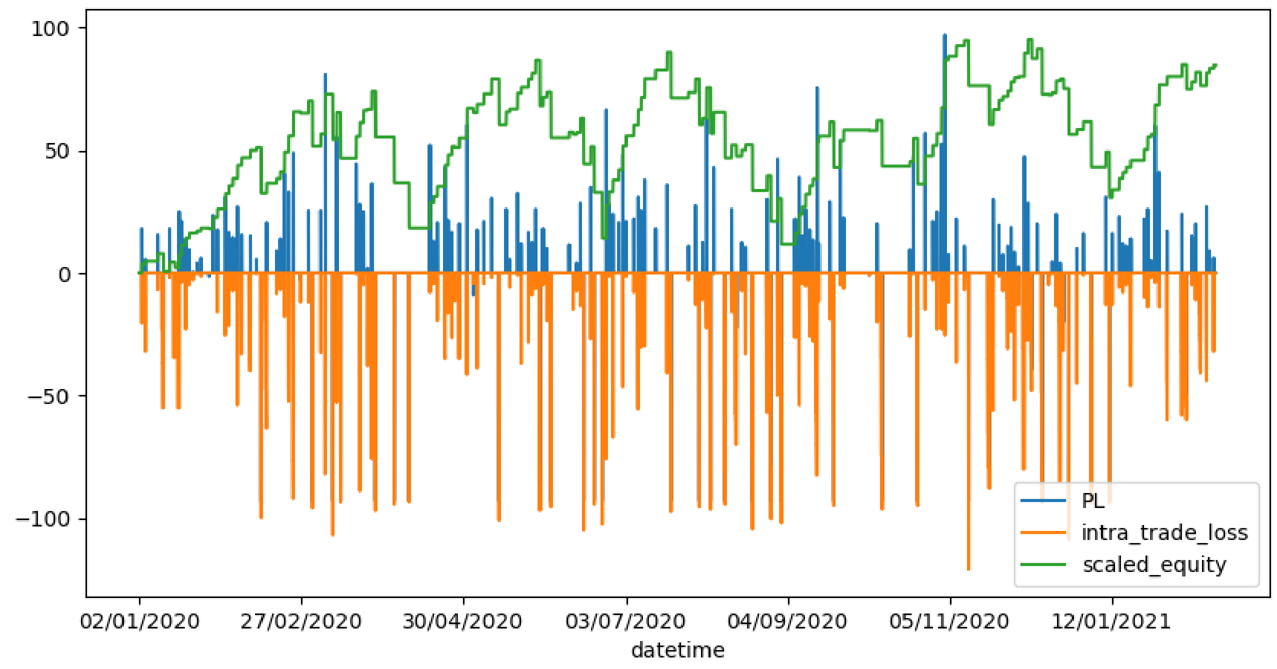


Figure 6 - DAX Noon Delta Signal backtest, 1 year data, stop loss at 2st STD of average intra-trade unrealized losses - A) Equity curve, B) PL and intra trade losses

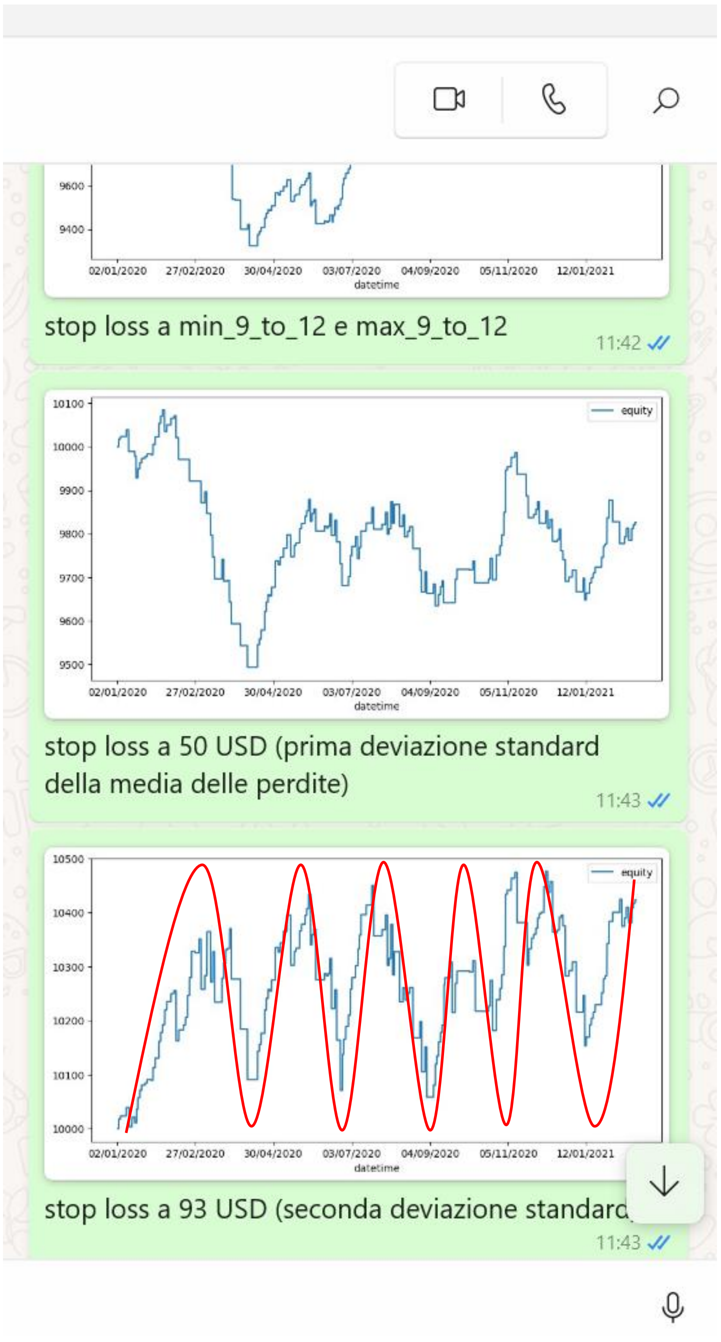
We have achieved an improvement with the average equity line positioned around 250 euros; however, there are still significant fluctuations that make the strategy suffering for its instability.

Time to call Paolo

Yes, it's a Sunday morning in February, and after dedicating two months of nights and weekends to this research, it's time to discuss the results with Paolo. My WhatsApp voice call rings, and I've already sent him the equity curve in our chat.

- Me: "So Paolo, the statistics are consistent, but as you can see, trading the strategy directly could still result in significant drawdowns. Surely, it could be optimized and better understood why this happens..."
- Paolo: "Well, to me, the interesting part is just the first one; it's more than enough."
- Me: "Meaning?"
- Paolo: "The intention wasn't to trade the strategy, but to use it for confirmation in other strategies involving dominant cycle analysis and volume profiles."
- Me: "Really!? So, what and for whom have I been working all these nights?"

The answer, of course, is for myself. Therefore, I will continue the investigation from a statistical perspective. After a while, I end the call with Paolo and return to the WhatsApp chat. I take another look at the charts I sent to Paolo some of them. OH MY GOSH!



stop loss a min_9_to_12 e max_9_to_12 11:42

stop loss a 50 USD (prima deviazione standard della media delle perdite) 11:43

stop loss a 93 USD (seconda deviazione standard 11:43

Figure 7 - Whatsapp chat screenshot - Evidence of good cyclicity of the equity curve resulting from 2nd STD of average loosing trades as stop loss

The losing and gaining periods of the equity curve exhibit a fairly stable periodicity. Conducting a dominant cycle analysis can help predict when the strategy is likely to be winning or losing most frequently. Essentially, the trading strategy with a fixed stop loss at 93 euros could produce an equity curve that serves as a reference for assessing reliability based on its main cycle analysis. This information can be used to skip the application of this strategy during identified losing periods. Moreover, it provides insights into refining the strategy to enhance potential gains during winning periods. Foreseeing these could be a way for optimizing the strategy and maximizing profitability.

7. Statistical Analysis

The following analysis refers to uptrends' signals of the DAX Noon Delta Signal and the characterization of the relative afternoon maxes. Obviously, a similar analysis can be repeated for the downtrend signals and the relative minimal values after 12:00.

After 12:00 maxes characterization

This analysis aims to characterize the distribution of occurrences within 30-minutes time slots after 12:00. The specific occurrences being analyzed are as follows:

1. **After 12:00 maxes**: Identifying the maximum values occurring after 12:00.
2. **After 12:00 profitable maxes**: Determining the maximum values after 12:00 that resulted in profitable trades.
3. **Absolute min after 12:00 below the 9:00-12:00 min**: Identifying absolute minimum values occurring after 12:00 that are below the minimum values observed between 9:00 and 12:00.
4. **Absolute min after 12:00 above the 9:00-12:00 min**: Identifying absolute minimum values occurring after 12:00 that are above the minimum values observed between 9:00 and 12:00.
5. **Min after 12:00 occurring before afternoon max below the 9:00-12:00 min**: Identifying minimum values occurring after 12:00 and before the realization of a maximum after 12:00 that are below the minimum values observed between 9:00 and 12:00.
6. **Min after 12:00 occurring before afternoon max above the 9:00-12:00 min**: Identifying minimum values occurring after 12:00 and before the realization of a maximum after 12:00 that are above the minimum values observed between 9:00 and 12:00.

It's important to note that the occurrence of the minimum value after 12:00 may precede or follow the realization of the maximum value after 12:00. The analysis of the last two occurrences (5 and 6) focuses on identifying very low minimum values occurring after 12:00, but before the realization of the subsequent maximum, which could potentially trigger a stop loss condition.

This detailed breakdown allows for a deeper understanding of the price dynamics and critical points within the specified time slots, aiding in the refinement and optimization of trading strategies based on these observed patterns.

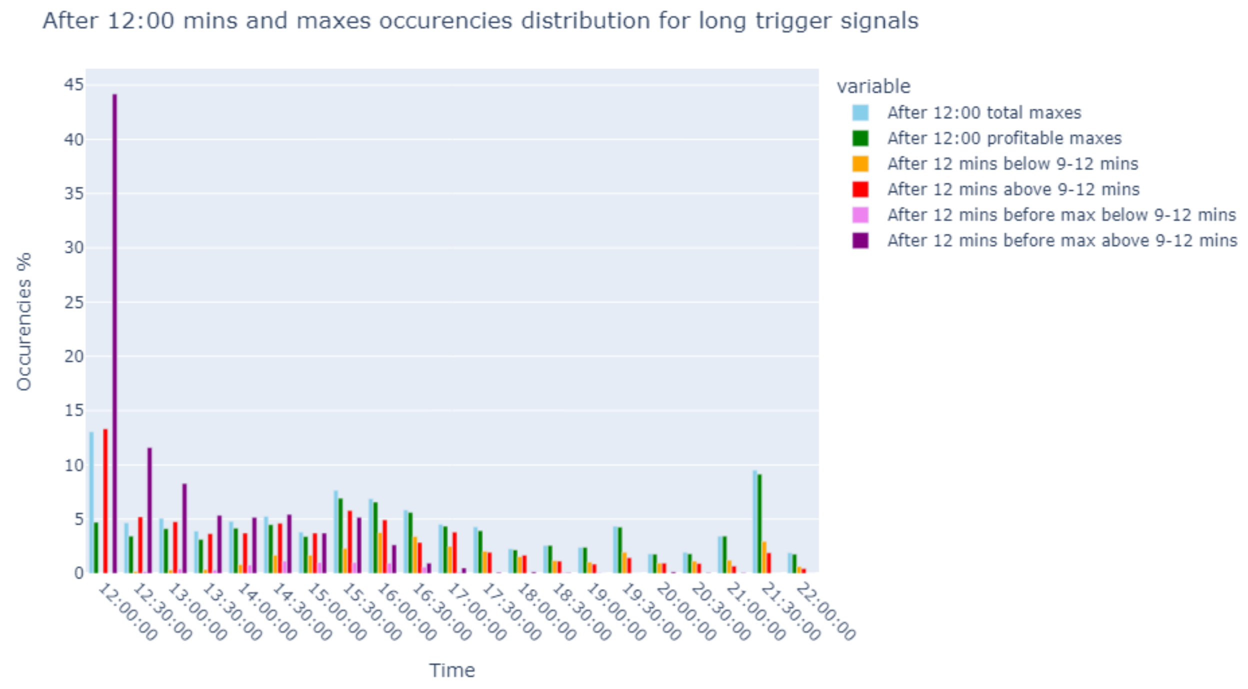


Figure 8 - Statistical characterization of occurrences useful for long trades

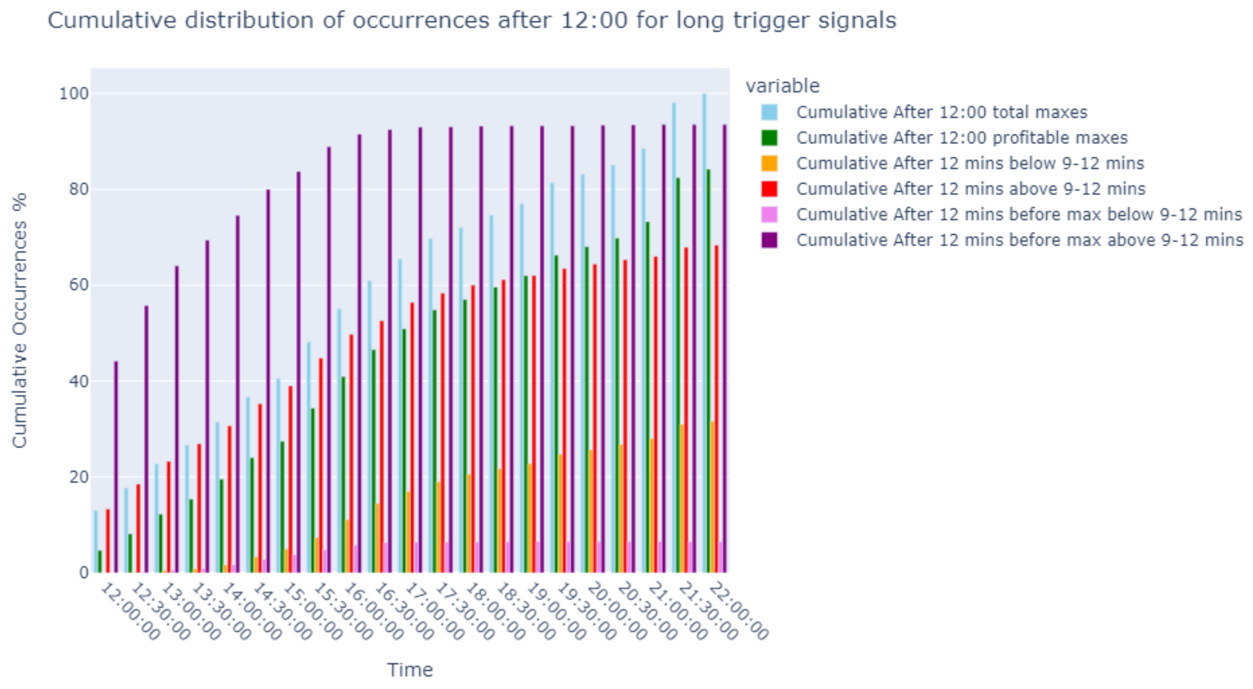


Figure 9 - Statistical cumulative characterization of occurrences useful for long trades

The first chart illustrates the distribution of occurrences every 30 minutes, while the second chart shows the cumulative distribution over the same intervals. From these charts, several interesting observations can be made:

1. Nearly 45% of minimum values preceding the maximums after 12:00 occur within the first half-hour after the 12:00 Open price, and around 80% of them occur within by 15:00. This suggests that in most cases, the take-profit threshold equal to the maximum price between 9:00 and 12:00 is reached due to a continuation of a long trend rather than an intermediate low swing. This observation could support an empirical approach to entering a long trade if at 12:00 there is a long strategy signal AND if the previous bar shows strong ascending momentum.
2. Between 12:00 and 15:00, the occurrences of maximum values are distributed relatively evenly. However, there are distinct peaks at 15:30-16:00 (around the opening of the US market), followed by a decrease throughout the later afternoon. There is a minor peak around 19:30 and a significant peak at 21:30, representing the final 30 minutes of daily trading.

Correlation between 9:00-12:00 min/max delta and after 12:00 min before max/max delta

The next scatter plot and *k-means* clusterization reveals a notable correlation between two key price differentials:

- The difference between the minimum and maximum prices realized from 9:00 to 12:00.
- The subsequent difference between the absolute maximum price after 12:00 and the average minimum price preceding this maximum after 12:00.

In this analysis, the minimum price preceding the afternoon maximum is considered instead of the absolute minimum in the afternoon time range. This choice is motivated by the fact that for uptrends, this difference represents the maximum potential gain achievable by entering at the minimum price and exiting at the subsequent maximum, which is typically above the threshold set by the 9:00-12:00 maximum. It's important to note that while there may be opportunities for trading subsequent downtrends, the DAX Noon Delta Signal is not designed to anticipate or capitalize on these secondary opportunities. The strategy's focus remains on capturing gains within the identified uptrend period based on the observed price dynamics and thresholds established during the morning session.

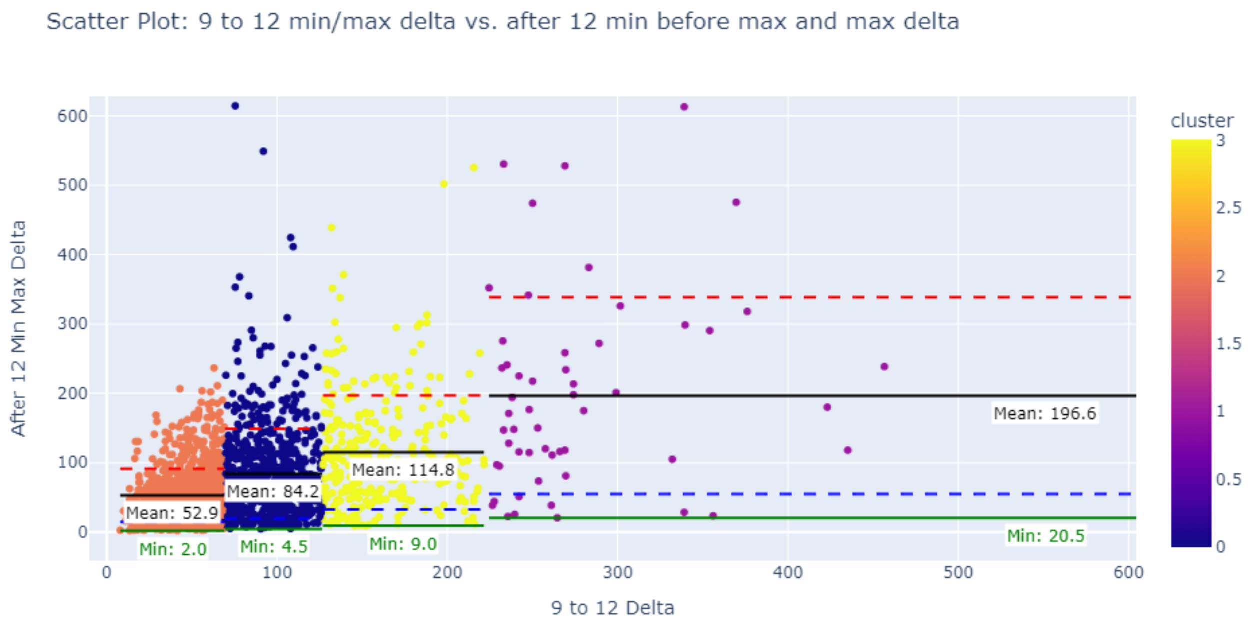


Figure 10 - Correlation scatter plot between morning and afternoon min-max deltas

This is somehow encouraging, but it must be considered that these mean figures have a relatively high standard deviation, so that they are good in average, but they cannot be considered as a possible take profit target. The minimal gains are respectively 2, 4.5, 9, 20.5 euros respectively, so much lower than the mean values.

The correlation between the two considered deltas is evident based on the following analysis:

- The mean ideal gain when the 9:00-12:00 min/max delta is less than 70 euros is approximately 53 euros.
- The mean ideal gain when the 9:00-12:00 min/max delta is between 70 and 124 euros is at least 84 euros.
- The mean ideal gain when the 9:00-12:00 min/max delta is between 124 and 218 euros is at least 114 euros.
- The mean ideal gain when the 9:00-12:00 min/max delta is higher than 218 euros is at least 196 euros.

While these figures are encouraging, it's important to note that they have a relatively high standard deviation. Therefore, while they represent good averages, they cannot be considered as reliable take-profit targets. The minimum gains associated with these scenarios are 2, 4.5, 9, and 20.5 euros respectively, significantly lower than the mean values.

Min before after 12:00 max areas occurrences

It could be valuable to understand the price ranges where the minimum before the afternoon maximum is likely to occur.

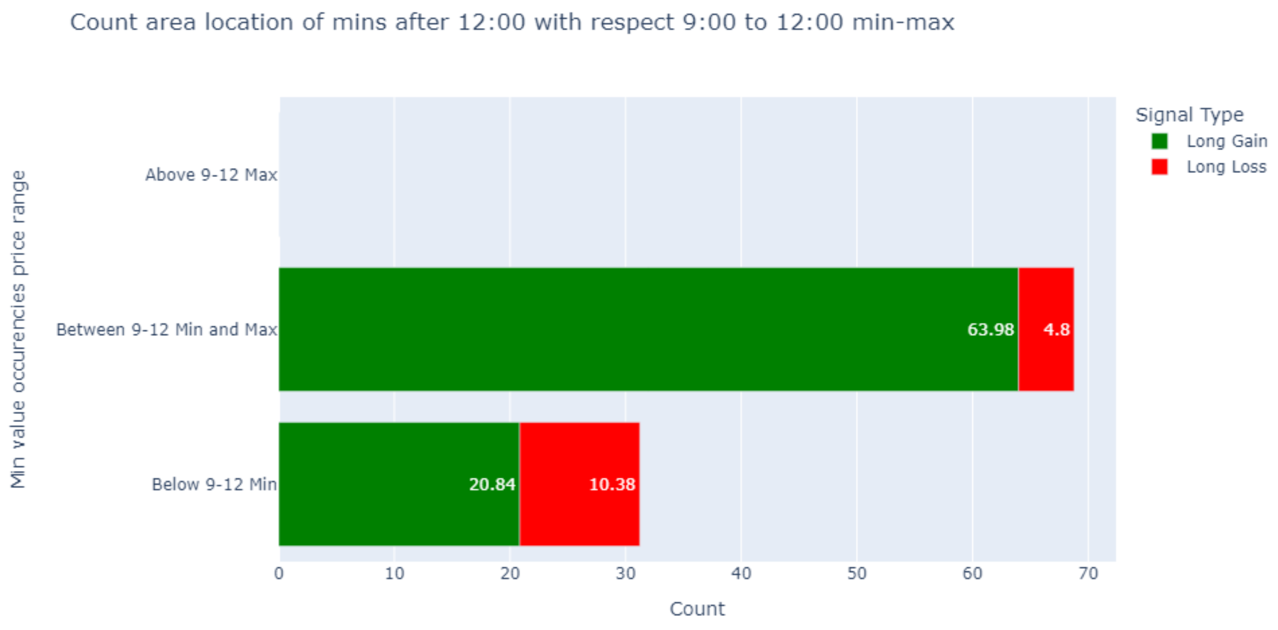


Figure 11 - Price ranging of the afternoon min before afternoon max

The chart indicates that, when considering the profitable cases (green areas of the bars), in more than three-fifth of cases, the minimum preceding the maximum value after 12:00 occurs between the minimum and maximum figures realized in the 9:00 to 12:00 time range. Conversely, in one-fifth of cases, it occurs below the minimum value observed between 9:00 and 12:00.

Number of 9:00-12:00 max threshold crosses after 12:00

A profitable uptrend signal does not necessarily imply only one crossing of the 9:00-12:00 maximum threshold. Within the same daily-afternoon trading session, multiple crossings of this threshold can occur, opening up new opportunities for profitable trades. The following distribution chart aims to characterize the occurrence of morning maximum crosses in the afternoon session of the same trading day.

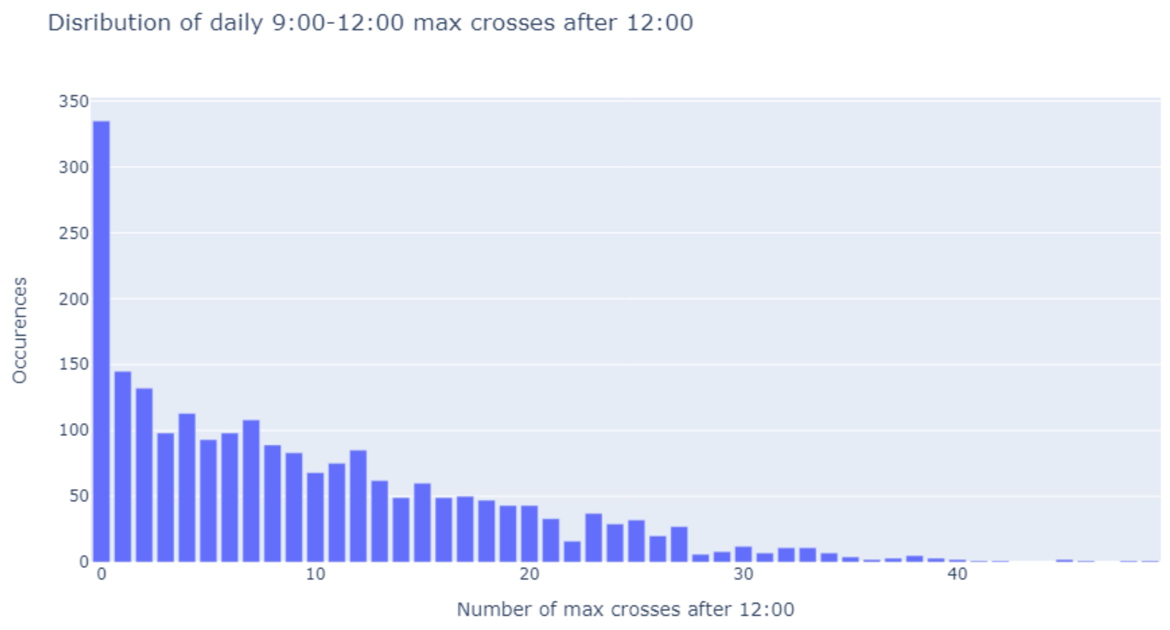


Figure 12 - Distribution, number of morning max crosses in the afternoon of the same daily trading session

The first bar with "0" occurrences represents the number of unprofitable long signals from the strategy. As inferred from this distribution, in the majority of profitable signals, the crossing of the 9:00-12:00 maximum occurs more than once during the same afternoon session.

Distribution of delta between 9:00-12:00 and after 12:00 maxes

This analysis seeks to address the question of how much an afternoon uptrend can exceed the maximum price realized between 9:00 and 12:00.

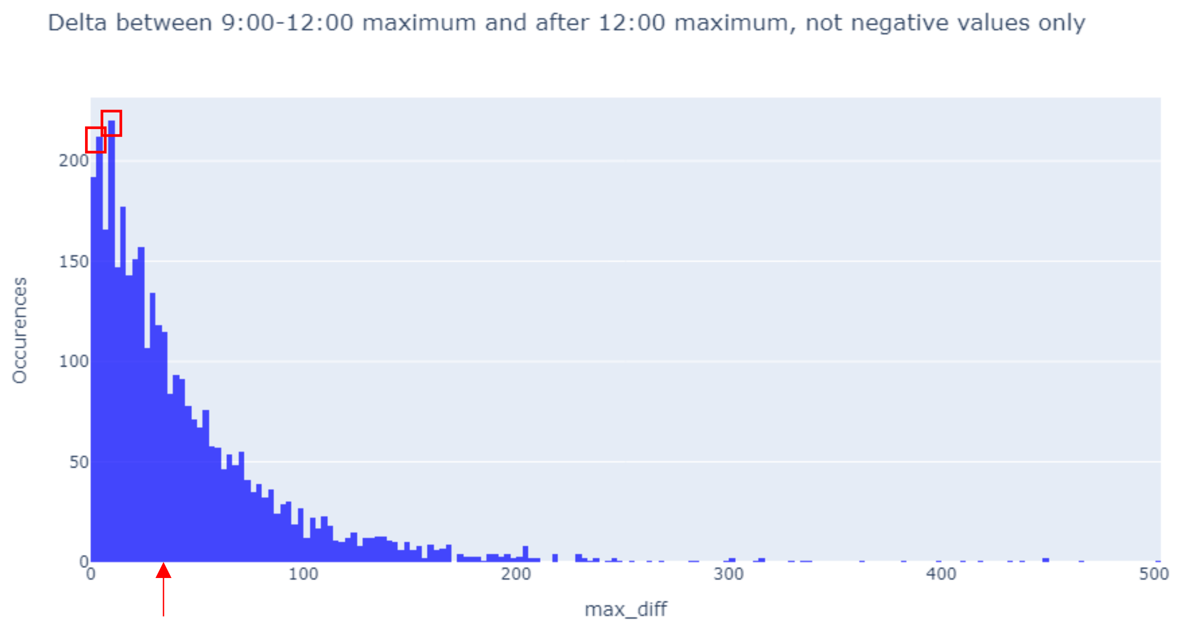


Figure 13 - Distribution of potential maximum exceed of the afternoon max with respect to the morning one

The distribution reveals a significant number of occurrences within the range of 0 to 36 euros, with two prominent peaks covering the ranges of [2.78, 5.55] euros and [8.34, 11.11] euros.

8. Strategy Optimizations

During our back-testing of the strategy, we identified opportunities for improvement, such as fine-tuning the stop-loss levels to vary with the 12:00 Open price and the difference to the take-profit target. Additionally, we observed a statistical correlation between the difference in minimum and maximum values from 9:00 to 12:00 and the subsequent difference between the minimum before the maximum after 12:00 and the maximum itself.

To further explore potential improvements, we can test variations in forecast precision by adjusting parameters such as the thresholds set at one-third above the minimum and below the maximum, as well as the take-profit thresholds initially aligned with the 9:00-12:00 minimum and maximum levels used to trigger the trading signal. For testing these optimization strategies statistically, we can employ specific algorithms such as genetic algorithms or brute force methods. Given my decision to test a ±50% variation of the inner thresholds divided into just 16 points, I implemented a straightforward *try-all* approach. Below are the results for the long signals; replicating the test for short signals can be done similarly.

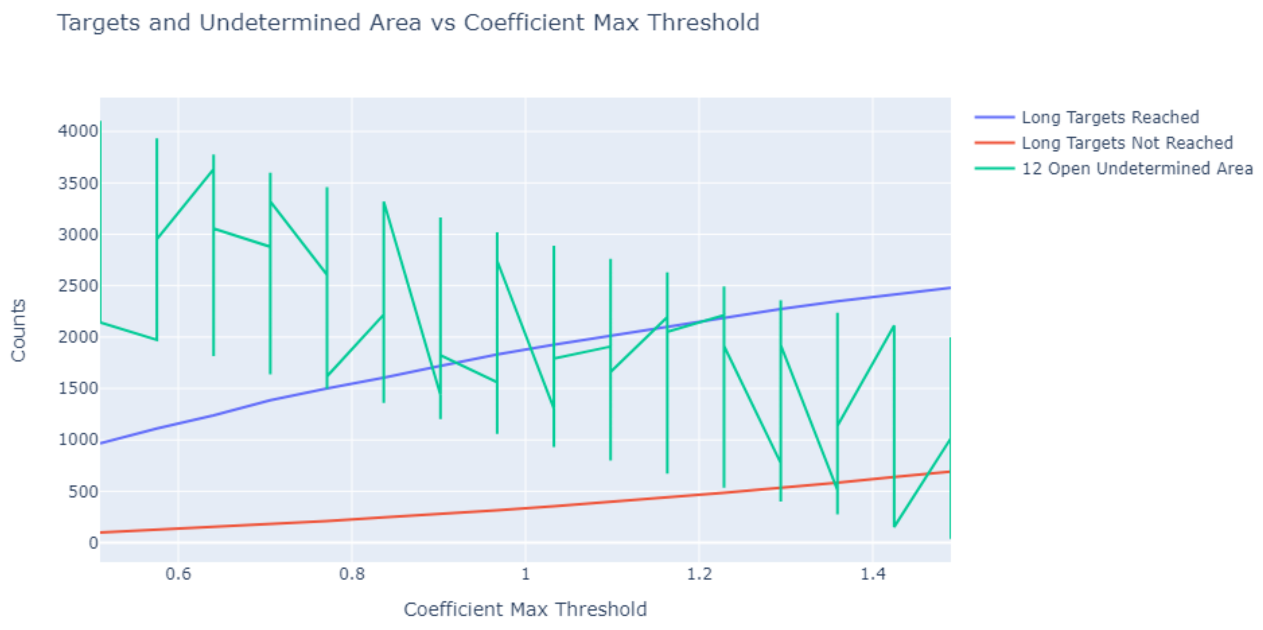


Figure 14 - Absolute number of reached and not reached target vs. variation of the inner thresholds (+-50%)

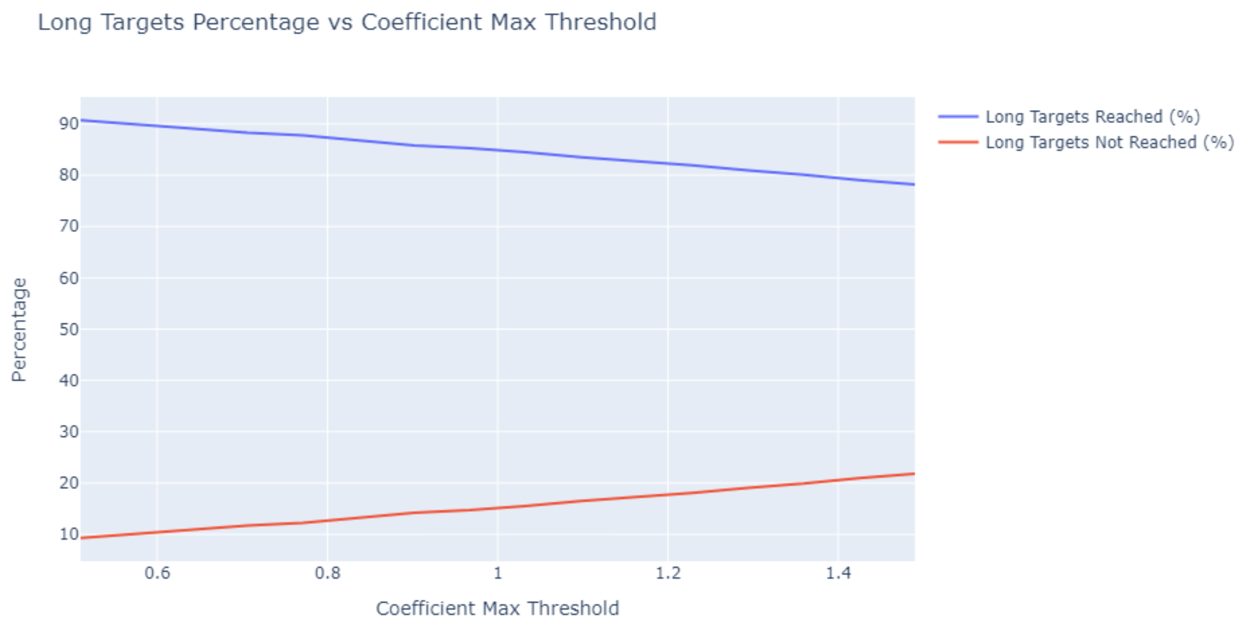


Figure 15 - Percentage number of reached and not reached target vs. variation of the inner thresholds (+-50%)

Just a note before interpretating the above charts; the green line of the first chart has multiple points on the same verticality because the optimization curves are three dimensional, whereas here we are plotting a two dimensions projection having as explicit independent variable the *variation on the max threshold* on the X axe and not displayed the *variation on the min threshold*.

The first observation is that the curve of occurrences and the consequent accuracy are monotone. Besides, narrowing the inner thresholds, we get, as we could expect, a higher precision of forecast at an expensive cost of a very minor number of occurrences of the entry signal. At the extremis, we have around 1000 cases with an accuracy of 90.7% and approximately 2500 cases with an accuracy of 78.2%. In other word we lose 120 signal opportunities every 1% of accuracy improvement. Furthermore, since narrowing the thresholds means selecting the cases in which the 12:00 Open price is closer to the threshold, and, since this is assumed to be, as first policy, the take profit reference, another consequence is that we are reducing the average gain of the profitable trades.

So where is the optimal point? It depends on personal choices. What we can do now is to run a backtest on the two opposite variations (-50%, +50%) of the inner thresholds to see what happen to the equity curve.

9. Implementing Statistical Suggestions for the DAX Noon Delta Signal

Backtesting variable stop loss as a multiple of the take profit level

Let's now experiment with a strategy that dynamically adjusts the stop-loss level based on the level of the take-profit. In other words, we will attempt to set the maximum acceptable loss based on the morning maximum/minimum delta on one side, and the remaining distance from the noon Open price to the threshold (morning minimum or maximum) in the forecasted direction on the other side. The following charts depict the equity curves for these scenarios:

- Stop-loss set at the same (opposite) distance between the noon Open price and the take profit.
- Stop-loss set at twice the distance of the take profit.
- Stop-loss set at three times the take profit distance.

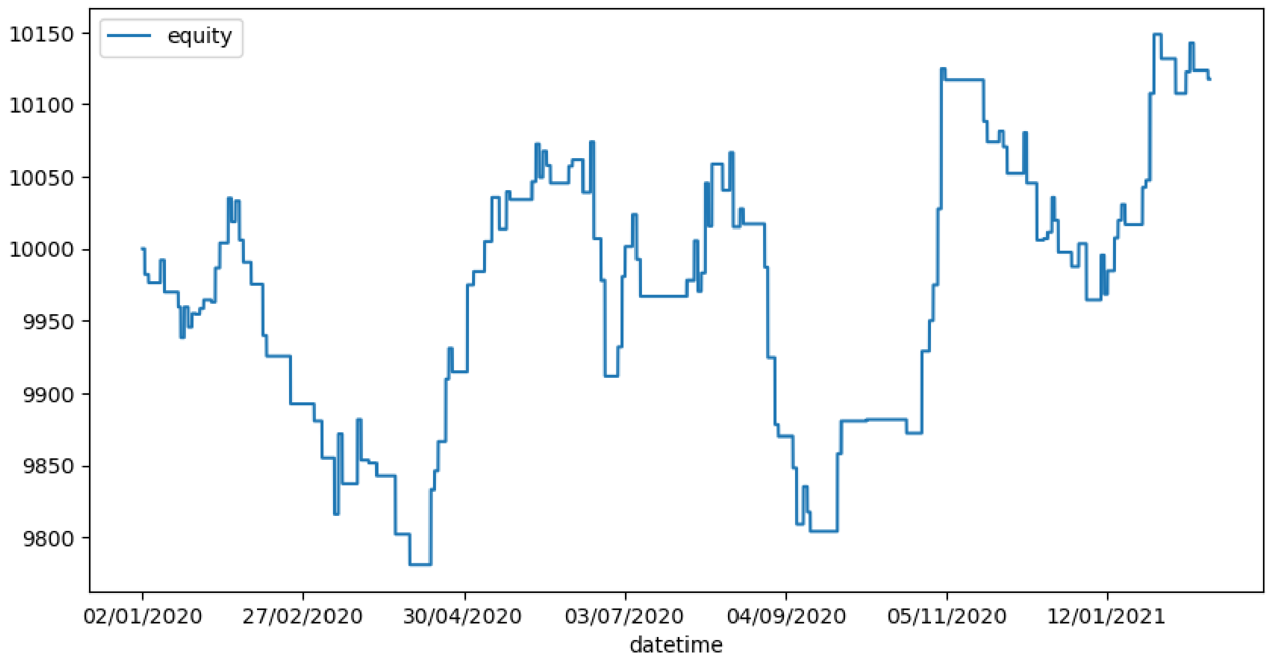


Figure 16 - DAX Noon Delta Signal - Statistical Suggestions Implementation - Equity curve - Stop-loss set at the same (opposite) distance between the noon Open price and the take profit

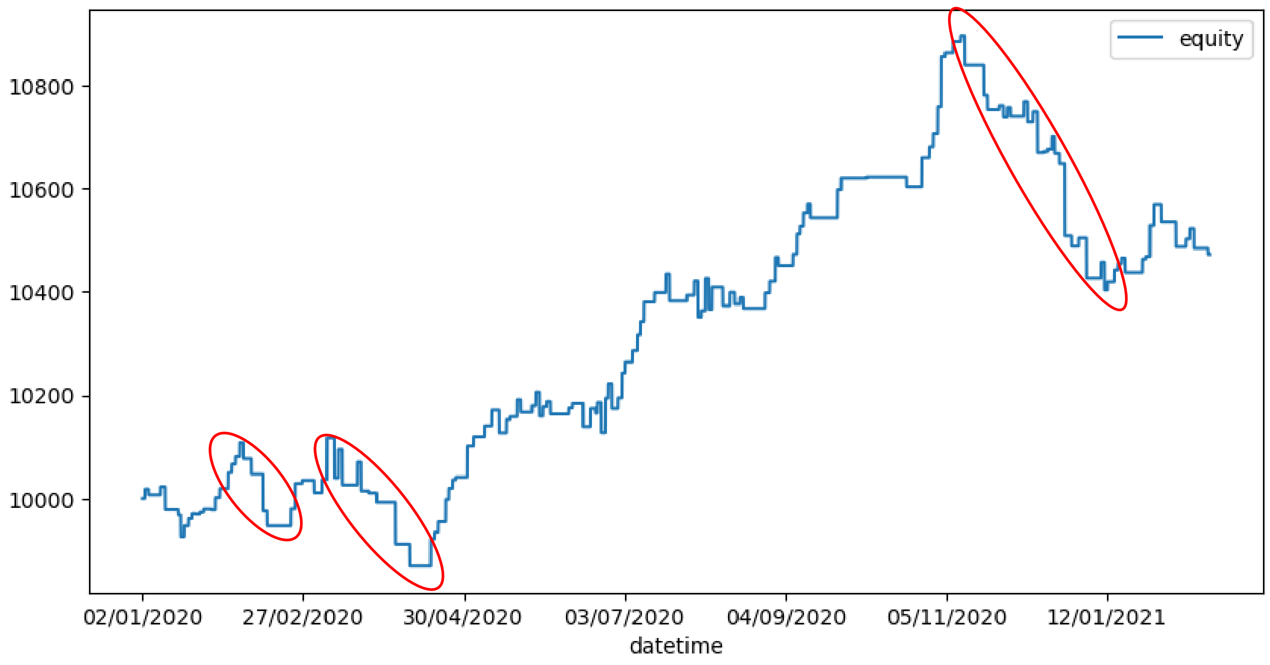


Figure 17 - DAX Noon Delta Signal - Statistical Suggestions Implementation - Equity curve - Stop-loss set at twice the distance of the take profit

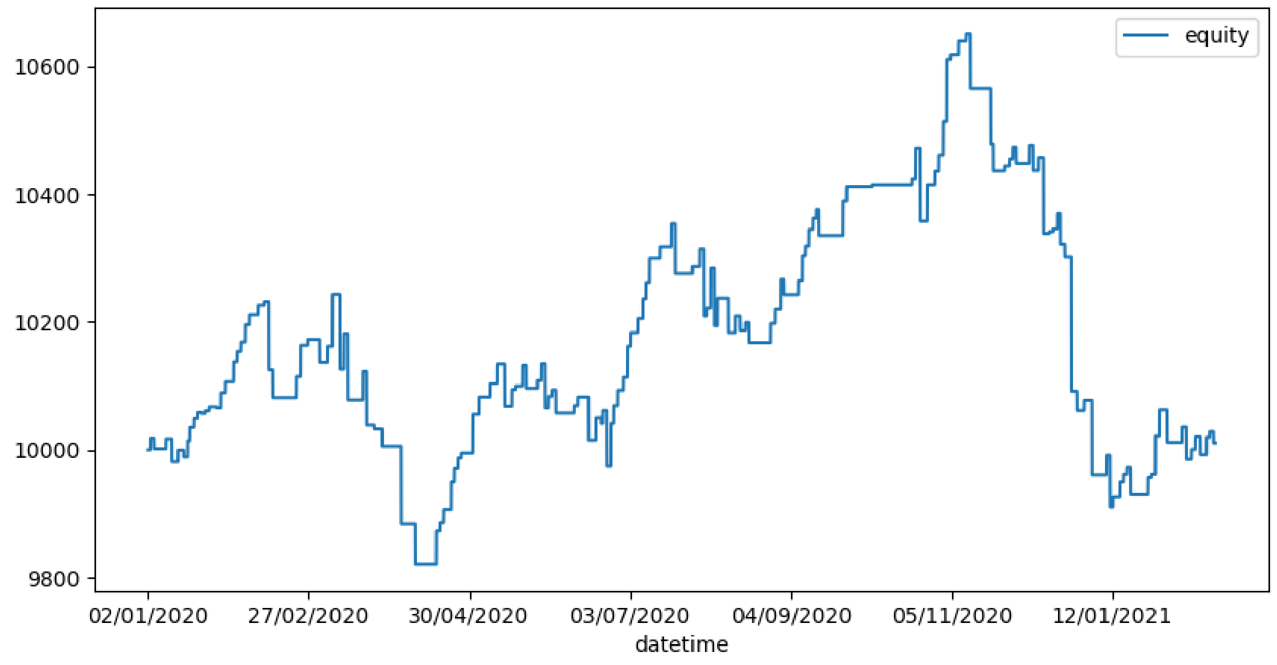


Figure 18 - DAX Noon Delta Signal - Statistical Suggestions Implementation - Equity curve - Stop-loss set at three times the take profit distance

As can be seen, the best performing stop-loss policy is the one that sets the maximum loss at twice the take-profit level. This strategy reduces the average value of losses, resulting in the equity reaching a maximum profit of over 800 euros. Unfortunately, the equity curve experiences three drawdowns, with the last one being particularly significant and disruptive.

Variation of the inner thresholds

Since we have demonstrated statistically that there is a relationship between the signal-triggering ranges and the forecast precision, with higher precision observed with smaller ranges, we can conduct some backtests by varying the inner thresholds by ±50%. The following chart shows the equity curve when decreasing the inner thresholds from 1/3 to 1/6 while keeping the stop-loss at twice the take-profit level.

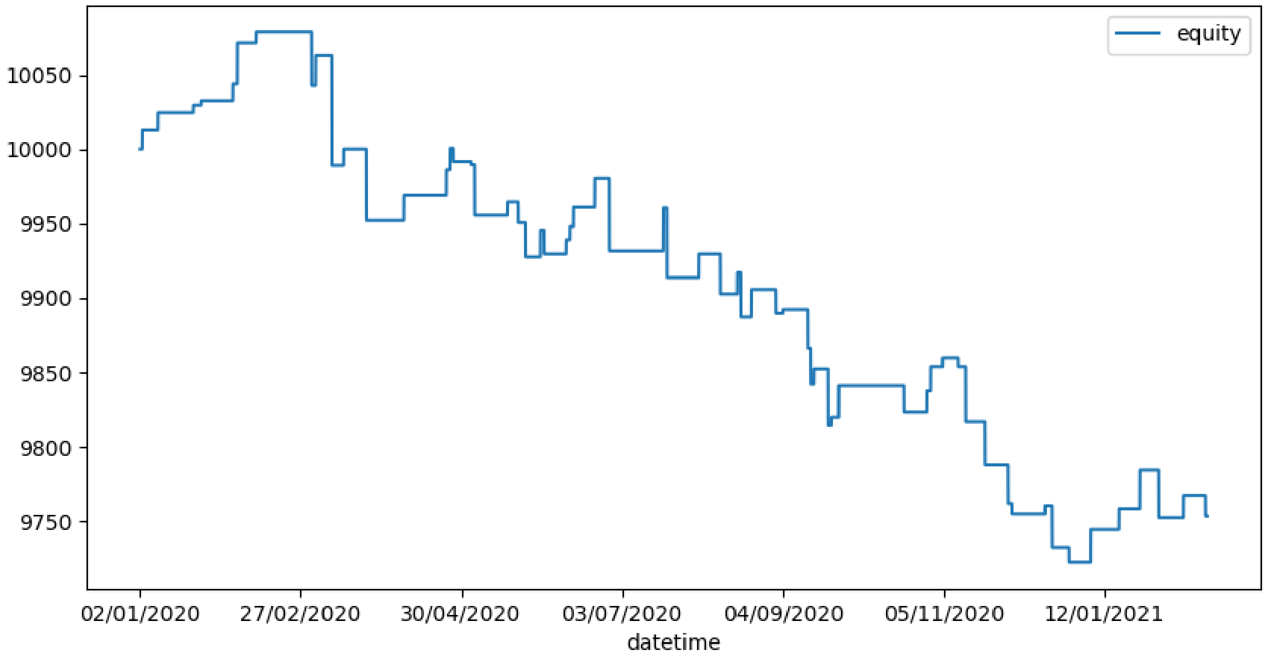


Figure 19 - DAX Noon Delta Signal - Statistical Suggestions Implementation - Equity curve - Inner thresholds 1/6 of morning min-max delta, stop-loss at twice the take-profit level

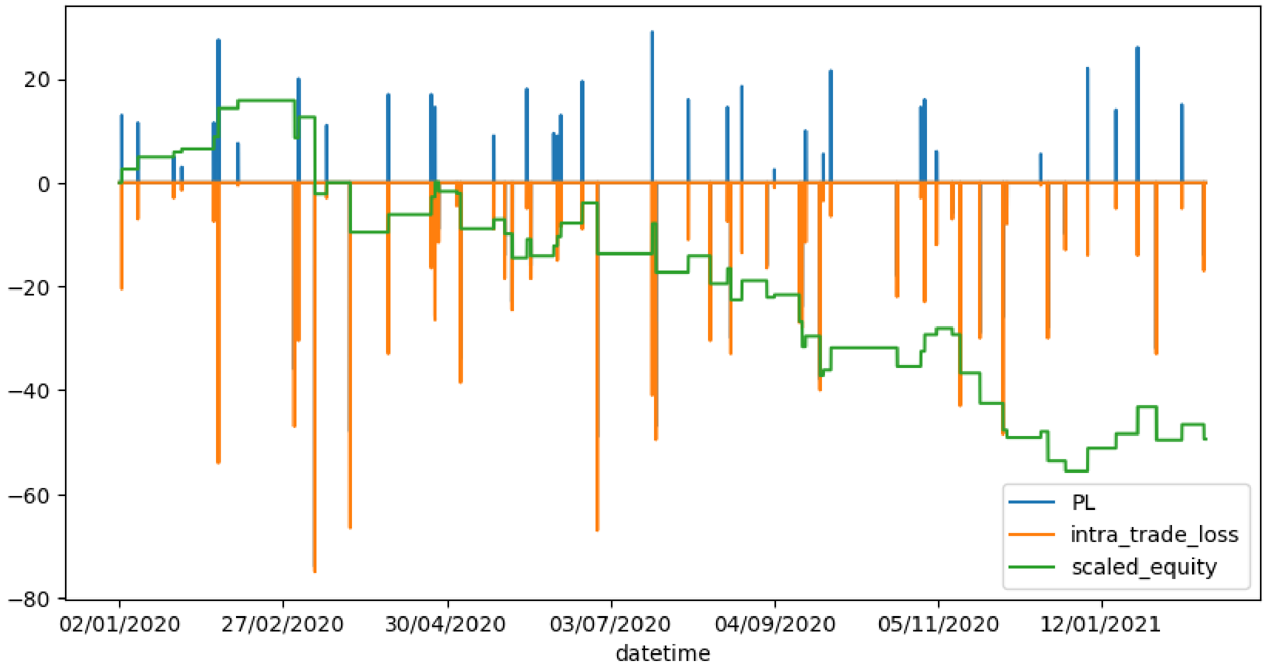


Figure 20 - DAX Noon Delta Signal - Statistical Suggestions Implementation - Intra trade loss - Inner thresholds 1/6 of morning min-max delta, stop-loss at twice the take-profit level

We positioned the thresholds at a level that, we know, gives a higher forecast precision decreasing a lot the recall. Apparently surprising, we get a monotonal decreasing equity. A first consideration could be that having smaller signal ranges also the average take profit is smaller. Since the stop loss level has been defined as a multiple of the take profit, as a consequence is that we are giving smaller breath to the trade positions probably leading to a higher number of taken stop-losses even in case of correct forecast. Therefore, it could be meaningful to test the same set-up with a stop-loss level four times higher than the take-profit one. Here is the resulting equity curve.

We positioned the thresholds at a level that, we know, provides higher forecast precision, but it significantly decreases recall. Surprisingly, we observed a monotonically decreasing resulting equity curve. However, the apparent surprise may have been expected. One initial consideration for explaining this behavior is that with smaller signal ranges, the average take profit is also reduced. Since the stop loss level is defined as a multiple of the take profit, this results in narrower trade positions, likely leading to a higher number of stop-losses even in cases of correct forecasts. Therefore, it would be meaningful to test the same setup with a stop-loss level set at four times higher than the take-profit level. Here is the resulting equity curve.

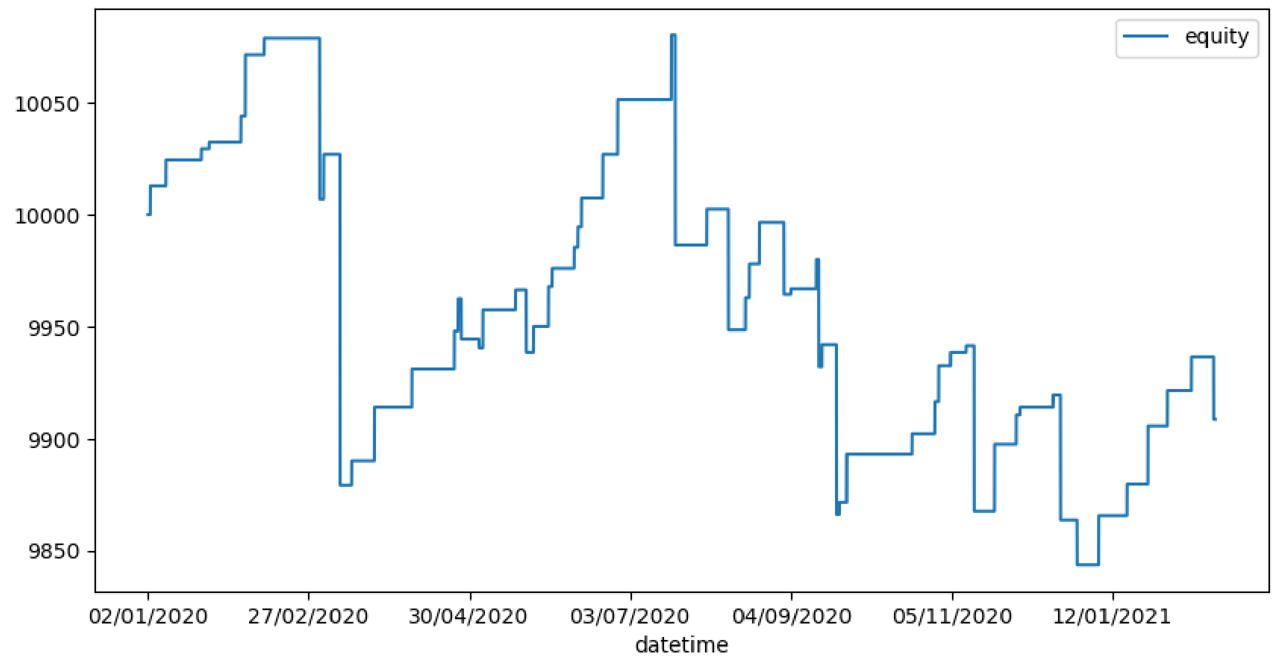


Figure 21 - DAX Noon Delta Signal - Statistical Suggestions Implementation - Equity curve - Inner thresholds 1/6 of morning min-max delta, stop-loss at four times the take-profit level

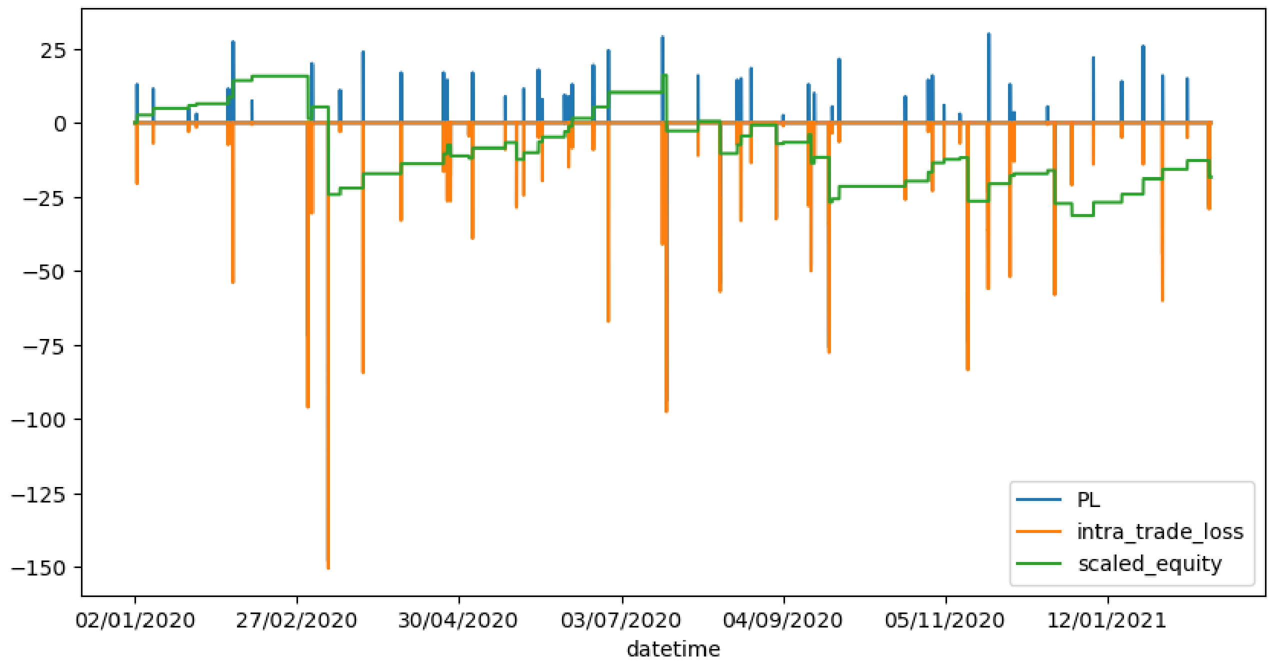


Figure 22 - DAX Noon Delta Signal - Statistical Suggestions Implementation - Intra trade loss - Inner thresholds 1/6 of morning min-max delta, stop-loss at four times the take-profit level

What we observed is an improvement in behavior in the middle of the yearly long equity curve, but this improvement was negatively impacted by a smaller number of very large losing trades. This suggests that excluding a signal in the context of a significant morning trend might be beneficial, as it's unlikely that such a large trend will continue or retrace significantly. Let's see what happens if we extend the inner thresholds by 50%; here is the resulting equity curve while keeping the stop-loss at twice the take-profit level.

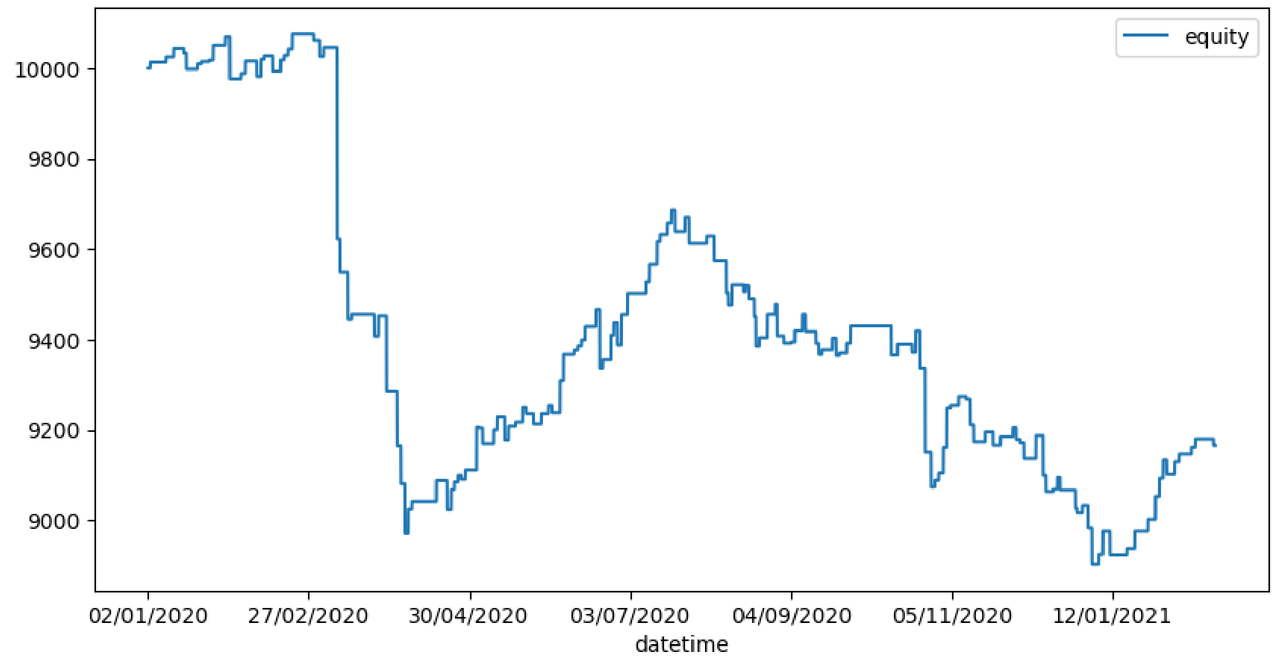


Figure 23 - DAX Noon Delta Signal - Statistical Suggestions Implementation - Equity curve - Inner thresholds 1/2 of morning min-max delta 2, stop-loss at two times the take-profit level

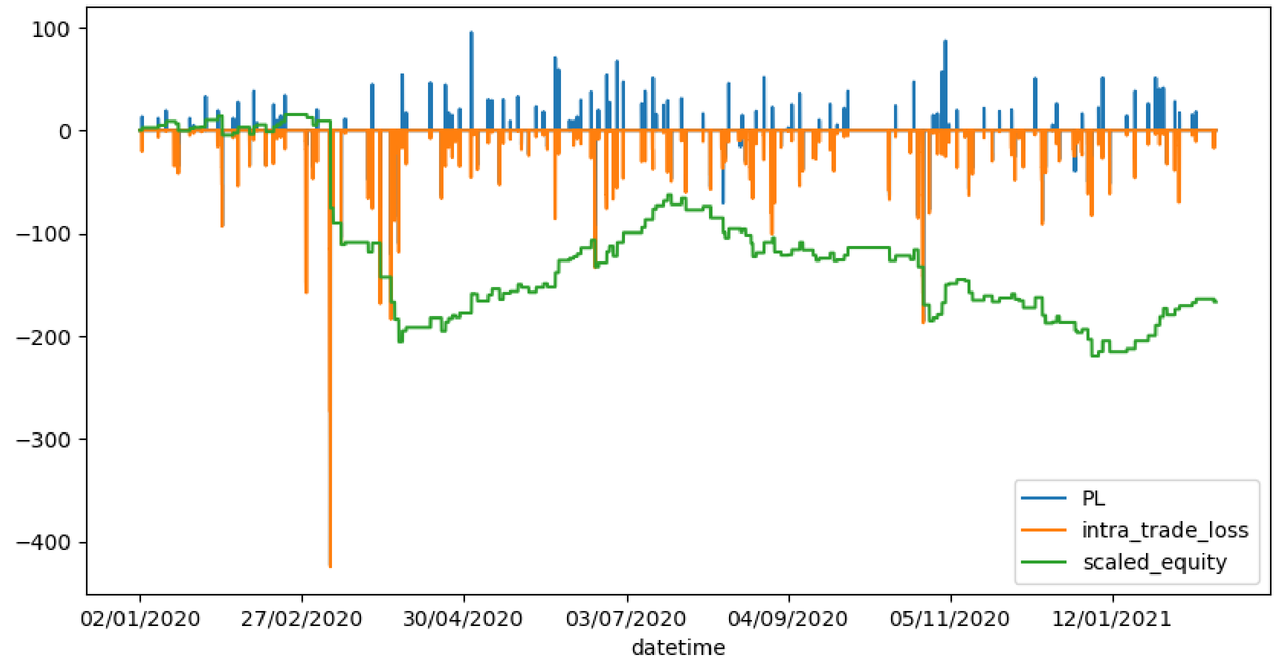


Figure 24 - DAX Noon Delta Signal - Statistical Suggestions Implementation – Intra trade loss - Inner thresholds 1/2 of morning min-max delta 2, stop-loss at two times the take-profit level

Finally, here is the equity curve obtained by decreasing the stop-loss to half the take-profit level, using the same setup for all other strategy policies.

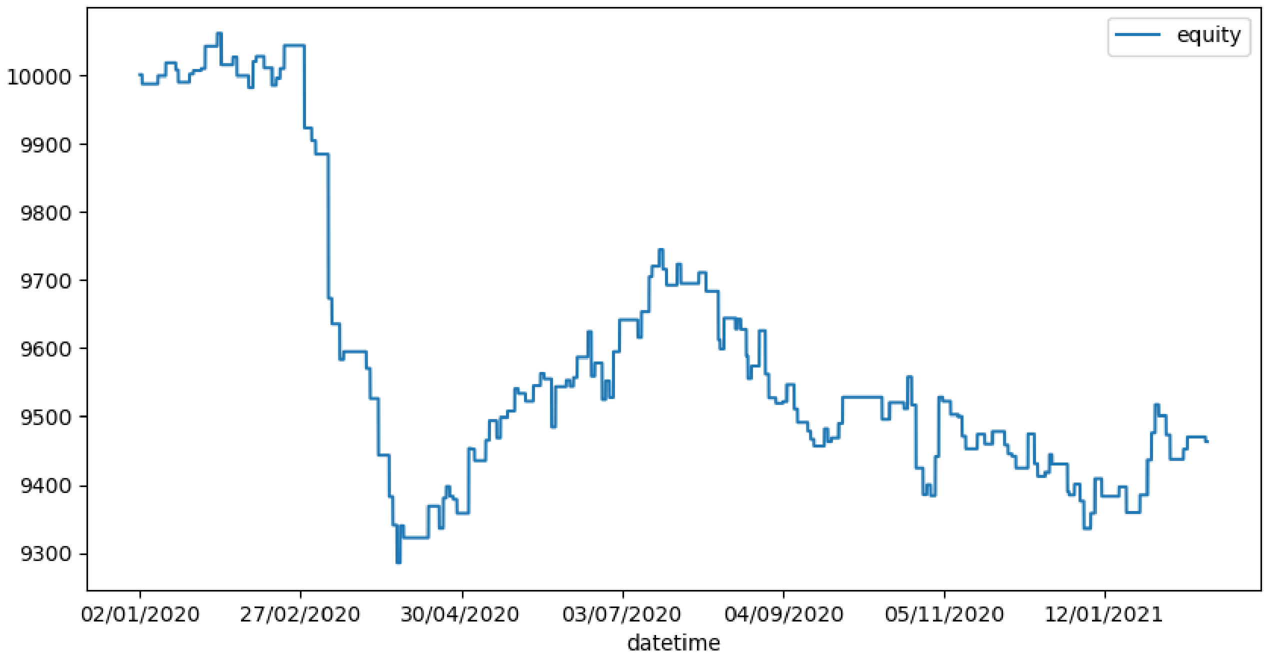


Figure 25 - DAX Noon Delta Signal - Statistical Suggestions Implementation – Intra trade loss - Inner thresholds 1/2 of morning min-max delta 2, stop-loss at one half of the take-profit level

What we can deduce is that in these last two attempts too, a few very large loss trades are responsible for ruining the equity line. By examining the morning min/max delta figures, we can understand that most of these losses occur when there was significant price action during the morning hours. Here is the chart suggesting this.

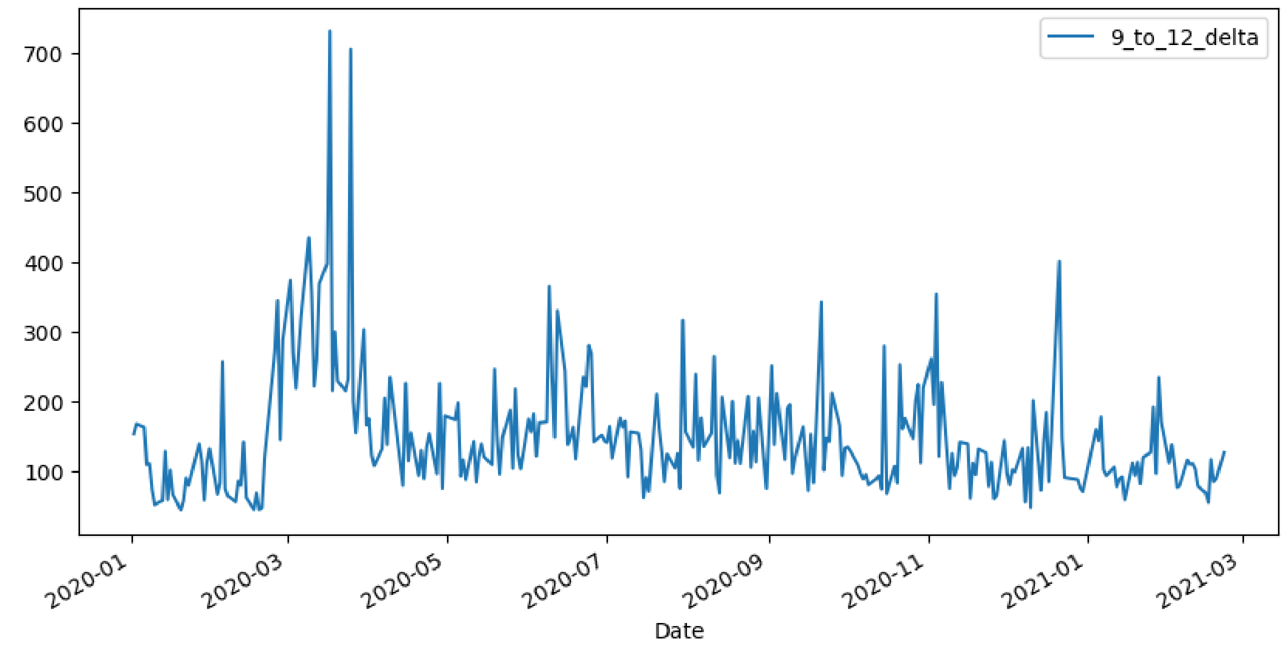


Figure 26 - DAX Noon Delta Signal - Morning max-min deltas

If this hypothesis holds true, we might expect some improvement by filtering out signals that occurred in the context of large morning min/max deltas. Let's implement a cut-off level at 200 euros; here is the resulting equity curve.

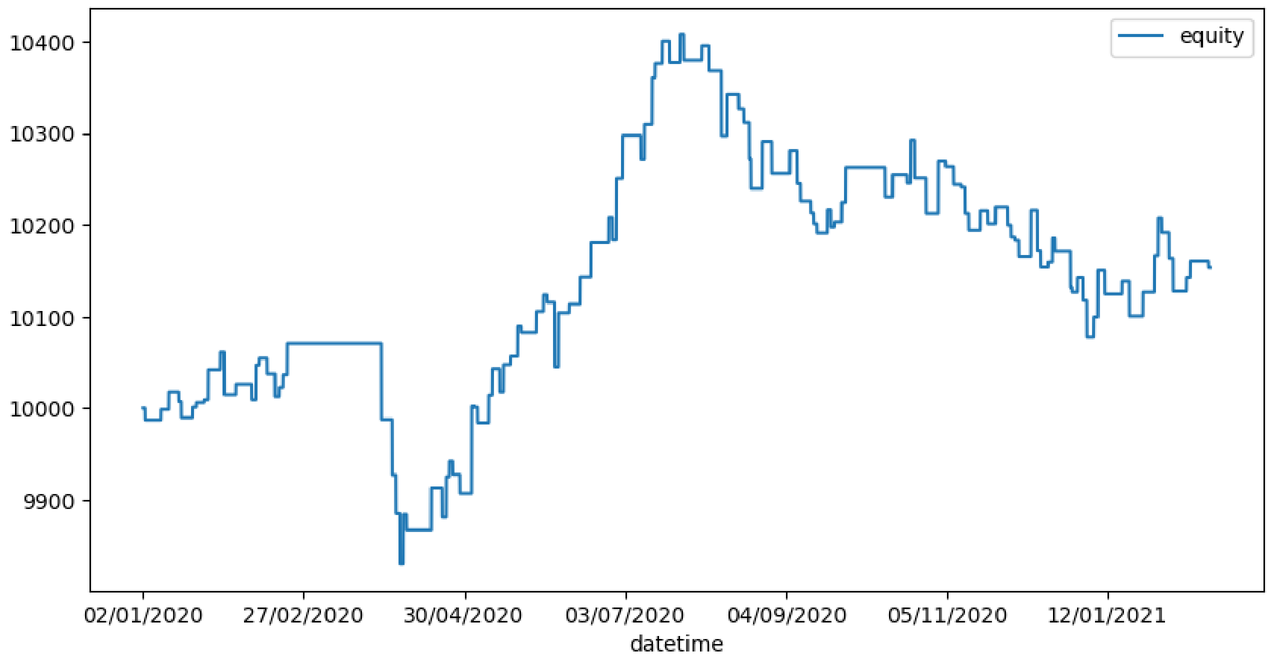


Figure 27 - DAX Noon Delta Signal - Statistical Suggestions Implementation – Equity curve - Inner thresholds 1/2 of morning min-max delta 2, stop-loss at one half of the take-profit level - Filtering out morning deltas higher than 200 euros

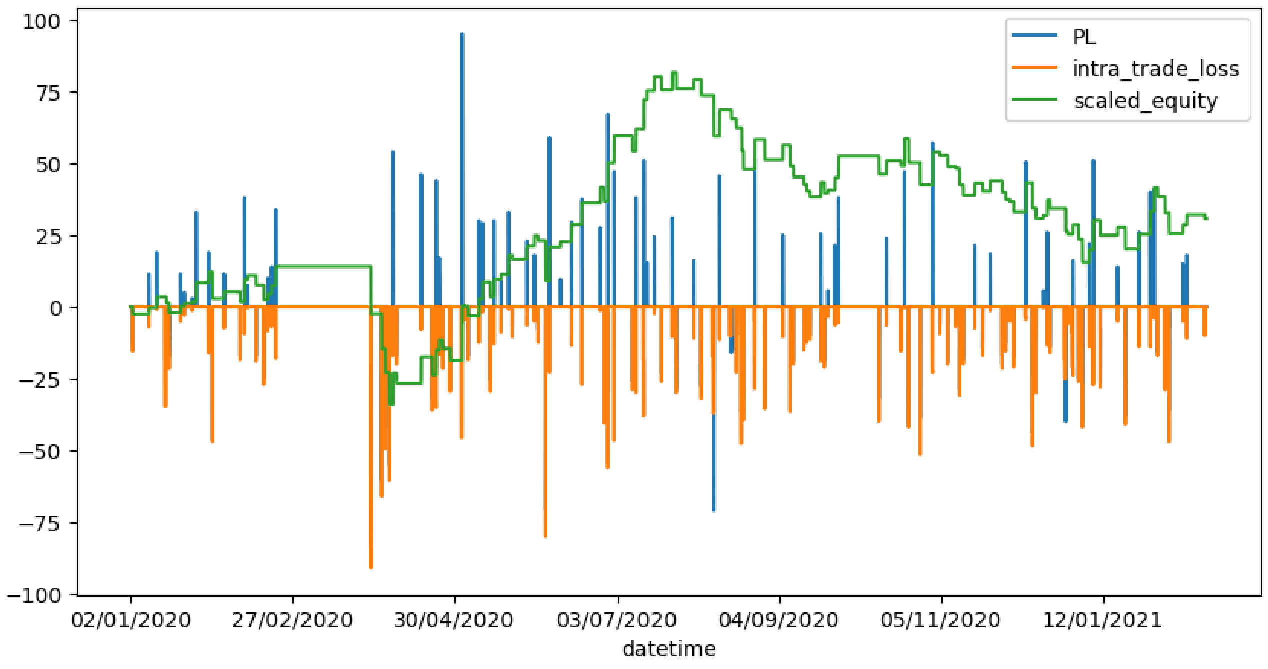


Figure 28 - DAX Noon Delta Signal - Statistical Suggestions Implementation – Intra trade loss - Inner thresholds 1/2 of morning min-max delta 2, stop-loss at one half of the take-profit level - Filtering out morning deltas higher than 200 euros

Time for metabolization

The performance of directly applying the DAX Noon Delta Signal and thresholds as a trading strategy is not particularly impressive. However, the initial attempts suggest that there is still room for improvement by optimizing the few adjustable parameters that characterize this strategy: the calculation of inner signal thresholds, the stop-loss level, and the take-profit level.

The major challenge is that the strategy accurately predicts whether the price will touch the morning min or max, but it remains uncertain when this will happen and what will occur in between, leaving uncertainty about the best entry points for a trade and the appropriate *stop-loss* levels.

10. Can Machine Learning improve this strategy?

A preliminary look on clusterization

While the main objective of this chapter is to explore supervised neural network training for implementing the DAX Noon Delta Signal, it could be valuable to also consider visual clustering to preliminary evaluate the feasibility of this approach. The Principal Component Analysis (PCA) is a method used to reduce the dimensionality of the problem while preserving as much inherent information as possible. I applied PCA to a basic metric set derived from morning period data, focusing specifically on long signals. Here is the list of input features used:

```
columns_for_clustering = ['9_to_12_delta',
                  '9_to_12_min_max_period',
                  '9_to_12_min_before_max',
                  'open_at_12_rel_distance',
                  '9_to_12_delta_day_-1',
                  '9_to_12_min_max_period_day_-1',
                  '9_to_12_min_before_max_day_-1',
                  'open_at_12_rel_distance_day_-1',
                  '9_to_12_delta_day_-2',
                  '9_to_12_min_max_period_day_-2',
                  '9_to_12_min_before_max_day_-2',
                  'open_at_12_rel_distance_day_-2',
                  ]
```

In this initial attempt, the target was the "gain" feature representing the correct (1) or incorrect (0) forecast. The goal was to determine if this feature could provide any information indicative of the accuracy of the DAX Noon Delta Signal's forecast. If this were the case, PCA should have revealed two distinct clusters of points in the two-dimensional scatter plot, each corresponding to the two values of the target. The following chart represents the result of the PCA on these features and target values.

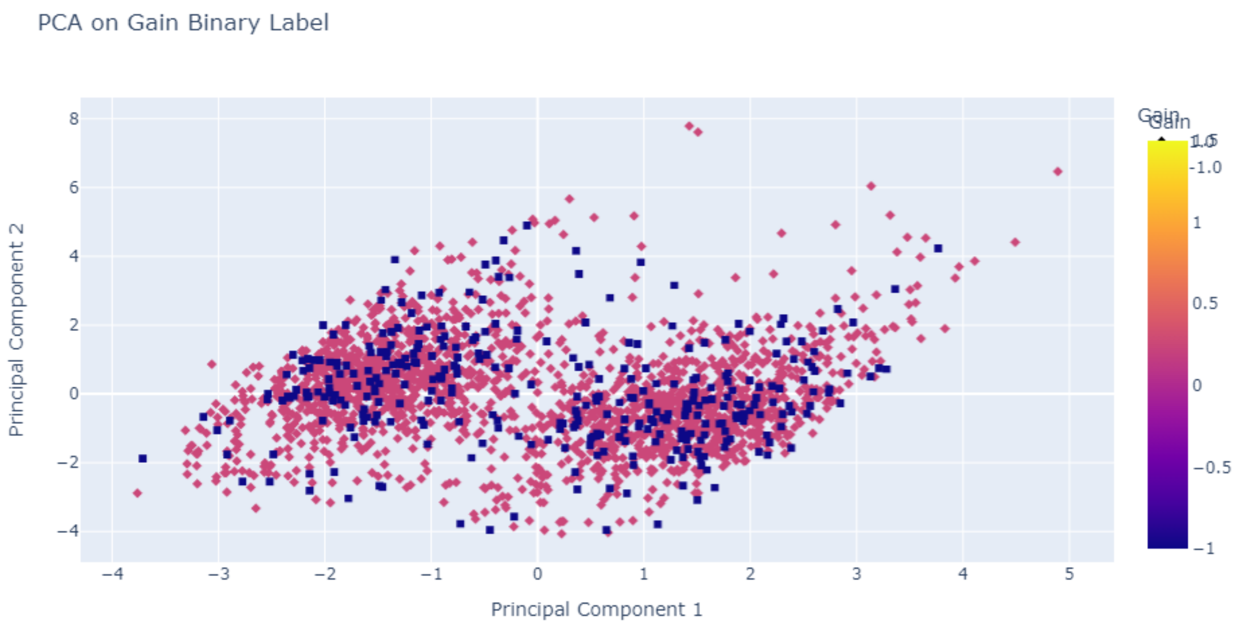


Figure 29 - PCA for clustering good and bad forecasts - Minimal set of features

The two categories (colors) are not clearly separated on the Cartesian chart, indicating a challenge that could also arise when using neural networks as a classifier. Additional features were considered to potentially address this issue, as in the following list.

```
X_data = df_stats_cleaned[[
                    '9_to_12_delta',
                    '9_to_12_min_max_period',
                    '9_to_12_min_before_max',
                    'open_at_12_rel_distance',
                    '9_to_12_delta_day_-1',
                    '9_to_12_min_max_period_day_-1',
                    '9_to_12_min_before_max_day_-1',
                    'open_at_12_rel_distance_day_-1',
                    '9_to_12_delta_day_-2',
                    '9_to_12_min_max_period_day_-2',
                    '9_to_12_min_before_max_day_-2',
                    'open_at_12_rel_distance_day_-2',
                    'gain_day_-1',
                    'gain_day_-2',
                    'weekly_day',
                    'month',

'9_to_12_delta_div_after_12_min_max_delta_day_-1',
'9_to_12_delta_div_after_12_min_max_delta_day_-2',
                    'binary_target_long_reached_day_-1',
                    'binary_target_long_reached_day_-2',
                    'binary_target_short_reached_day_-1',
                    'binary_target_short_reached_day_2',
                   '9_to_12_rel_maxes_number',
                   '9_to_12_rel_mins_number',
                    'pos_CO_div_volume_std',
                    'neg_CO_div_volume_mean',
                    'neg_CO_div_volume_std',
                    'pos_CO_HL_div_volume_mean',
                    'pos_CO_HL_div_volume_std',
                    'neg_CO_HL_div_volume_mean',
                    'neg_CO_HL_div_volume_std',
                    'pos_CO_volume',
                    'neg_CO_volume',
                    '9_to_12_max_idx_minus_min_idx',
                    '9_to_12_max_idx_minus_min_idx_day_-1',
                    '9_to_12_max_idx_minus_min_idx_day_-2' ,
                     '9_to_12_max_minus_9_to_12_max_day_-1',
                     '9_to_12_max_minus_9_to_12_max_day_-2',
                     '9_to_12_max_minus_9_to_12_min_day_-1',
                     '9_to_12_max_minus_9_to_12_min_day_-2',
                     '9_to_12_min_minus_9_to_12_max_day_-1',
                     '9_to_12_min_minus_9_to_12_max_day_-2',
                     '9_to_12_min_minus_9_to_12_min_day_-1',
                     '9_to_12_min_minus_9_to_12_min_day_-2',
                     '9_to_12_max_minus_after_12_max_day_-1',
                     '9_to_12_max_minus_after_12_max_day_-2',
                     '9_to_12_max_minus_after_12_min_day_-1',
                     '9_to_12_max_minus_after_12_min_day_-2',
                     '9_to_12_min_minus_after_12_max_day_-1',
                     '9_to_12_min_minus_after_12_max_day_-2',
```

'after_12_max_minus_9_to_12_max_day_-1', 'after_12_max_minus_9_to_12_max_day_-2', 'after_12_min_minus_9_to_12_min_day_-1', 'after_12_min_minus_9_to_12_min_day_-2', 'pos_CO', 'pos_CO_mean', 'pos_CO_std', 'neg_CO', 'neg_CO_mean', 'neg_CO_std', 'pos_CO_div_volume_mean',	'9_to_12_min_minus_after_12_min_day_-1', '9_to_12_min_minus_after_12_min_day_-2', '9_to_12_delta_div_9_to_12_delta_day_-1', '9_to_12_delta_div_9_to_12_delta_day_-2', '9_to_12_delta_div_after_12_min_max_delta_day_-1', '9_to_12_delta_div_after_12_min_max_delta_day_-2'

Including KPIs from the previous two days' mornings and afternoons, as well as current morning market behavior indicators, unfortunately did not improve the result.

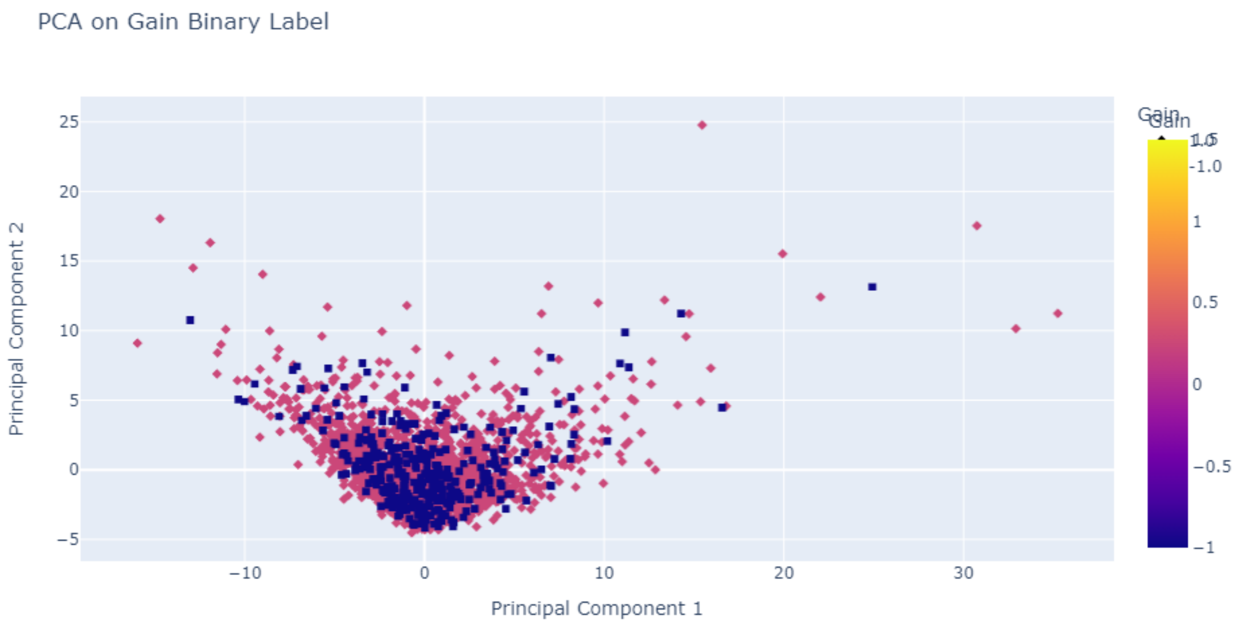


Figure 30 - PCA for clustering good and bad forecasts - Extended set of features

Instead of focusing on finding inherent information to distinguish between correct and incorrect forecasts, another attempt has been to cluster the four cases of reached targets themselves: *long* (1), *short* (2), *both* (3), and none of them (0). Unfortunately, we observed similar results with no clear separation based on the targets' colors.

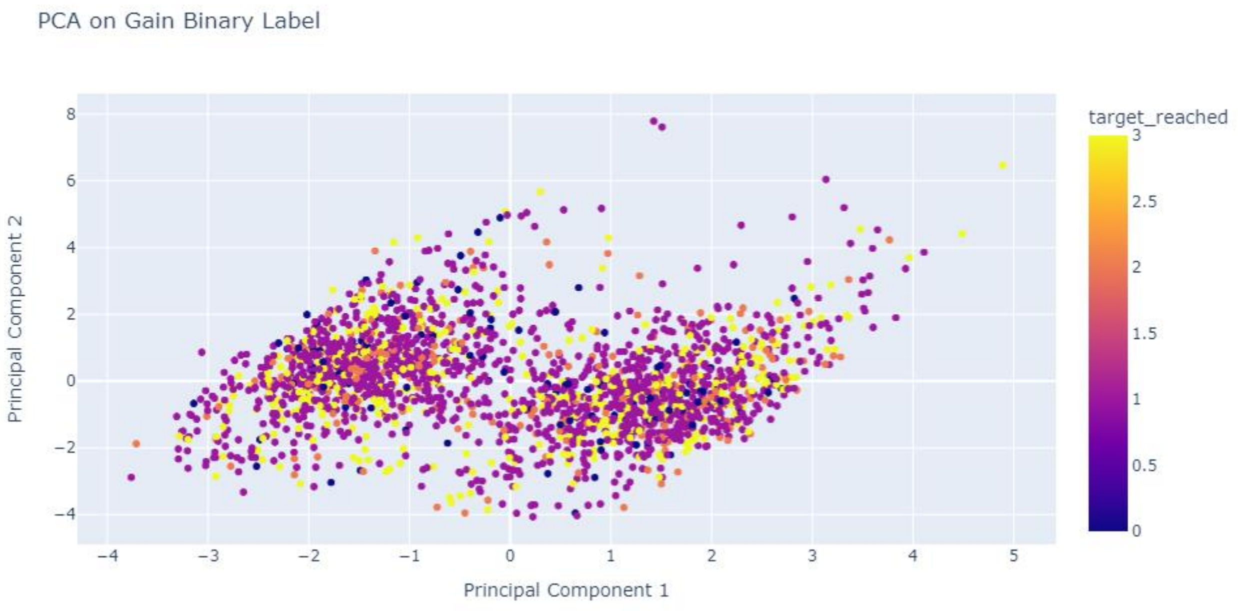


Figure 31 - PCA for clustering the four cases: target long reached, short reached, both reached, none of them – Minimal set of features

Increasing the number of features led to a worsening of the result.

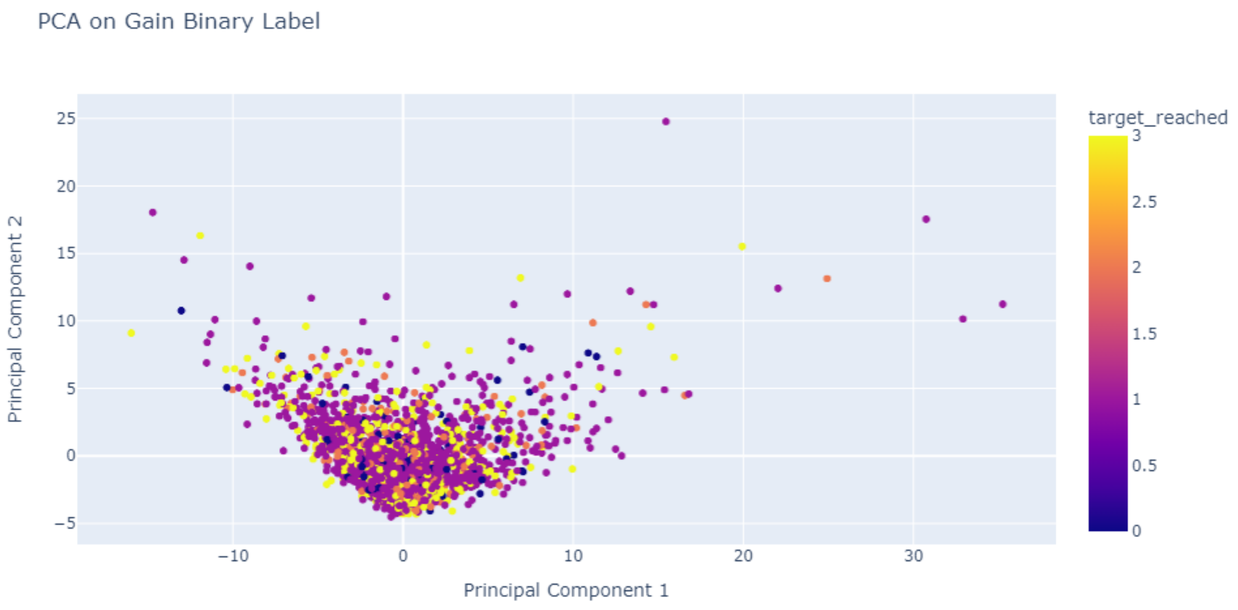


Figure 32 - PCA for clustering the four cases: target long reached, short reached, both reached, none of them – Extended set of features

The path is all uphill!

Attempts with classificators and regressors

I've made numerous attempts, but the results are not very promising. These attempts involved incremental inclusion of features, and I'll summarize the goals and outcomes of each approach:

1. **Base Strategy Signals Confirmation/Not-Confirmation ANN Network**: The objective was to train an ANN network to recognize false entry long signals from the Noon strategy. However, the dataset was unbalanced because the base strategy was quite good. The "0" values (indicating incorrect forecasts) were almost five times less frequent than the "1" values (correct forecasts). Attempts to address this imbalance by replicating "0" rows or adjusting training weights did not lead to meaningful improvements.
2. **No Rule-Based Forecast of Thresholds Touching ANN Network**: The goal was to replicate and potentially enhance the performance of the rule-based forecast by allowing the neural network to identify correlations among all morning features. While the results were good, they did not surpass those of the rule-based strategy. One potential reason could be the significant number of cases where both the morning min and max thresholds were reached in the afternoon, which may lack sufficient forecasting information in the provided features.
3. **Forecasting Afternoon Touch of Both Morning Min and Max ANN Network**: Again, the dataset was unbalanced, with cases of both morning min and max being touched in the afternoon accounting for approximately one-fourth to one-fifth of the total cases where only the morning min or max was touched. The performance of this approach was very poor.

I also made minor attempts to forecast price elongation beyond the morning max in long signals and beneath the morning min in short signals using a regressor ANN network, but these yielded no meaningful results. Additionally, I explored using an LSTM regressor network to capture cyclic patterns in the deltas between morning and afternoon maxima and minima, again without obtaining usable outcomes; this idea was suggested by the following chart representing differences between the morning and afternoon maxes (as well as mins) the peaks of which apparently manifest some cyclicity.

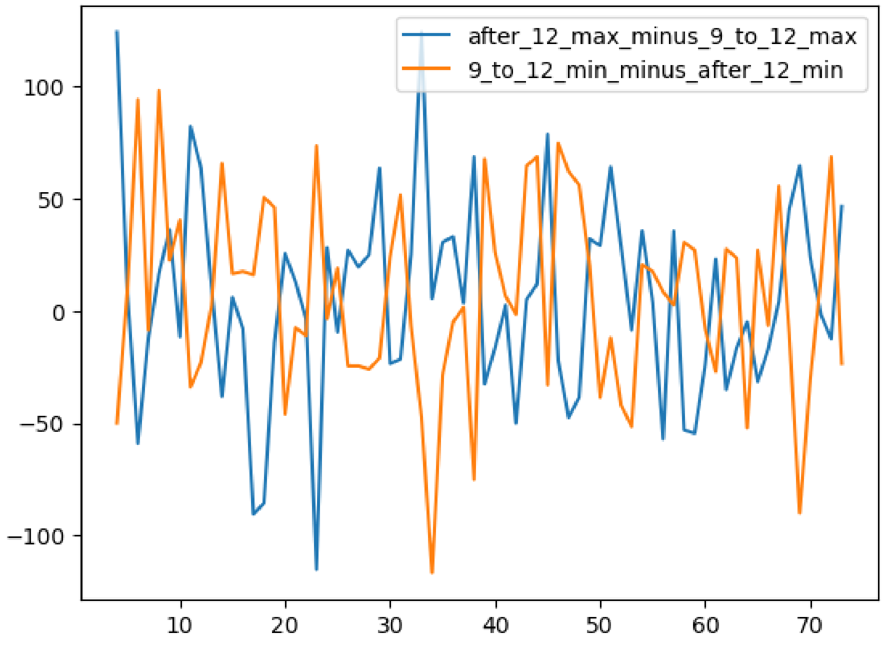


Figure 33 - morning and afternoon maxes (and mins) deltas

The final attempt I'd like to discuss involves combining signals from the rule-based strategy with those generated by the second ANN network.

A note on dataset preparation for ML training: Random or sequential split?

As a friend and former colleague pointed out, dealing with time series data, it is preferable to maintain the chronological order of the data when splitting into training and validation datasets. The rationale behind this is that when we deploy a model in real-time, it is trained on historical time series to infer the characteristics of past observed phenomena. If we randomly split the training and validation sets, we inadvertently train the model on data that chronologically follows some of the validation data. In essence, using a random split introduces a bias in the training process, as the model may learn characteristics of future behavior from the validation data, resulting in unpredictable performance.

Therefore, the most conservative and less contestable approach to splitting a time series dataset is to preserve its chronological sequence. However, it should be noted that the benefit of evaluating the model on records that precede some of the training data isn't necessarily advantageous; it simply might be. For instance, if the training dataset is sparsely populated and the observed phenomenon undergoes intrinsic behavior changes over time, the predictive capability of the trained model may be diluted across these varied behaviors, leading to a more stable but less consistently performing forecast.

Typically, in sequential splitting, the most recent data forms the validation subset. Conversely, with a random split, including more recent data in the training set implies that the resulting trained model is more up-to-date in its understanding of the phenomenon.

In this specific case, I don't believe that using one approach over the other will make a significant difference, for two main reasons. Firstly, we are training the model on a dataset spanning 24 years, which provides a comprehensive representation of inherent behaviors. It's reasonable to assume that any significant structural changes or entirely new phenomena are unlikely to occur in the future that aren't already reflected in this dataset. Secondly, the simplicity of both the strategy and the data used to train it further supports this notion. The features are derived from broad data aggregations and straightforward thresholds.

However, since this is merely an intuition, in the next chapter, I will proceed to train the model using both random and sequential data splits and compare their performance. This empirical comparison will provide a clearer understanding of whether the choice of data splitting method impacts the model's effectiveness in this context.

Mixing rule-based signals with ANN classificator ones

The general idea is that the performance of the second ANN network, as discussed in the previous section, could be compromised due to the presence of ambiguous cases where both the morning min and max thresholds are touched in the afternoon. These cases might represent a minority if the dataset is filtered using the rule-based strategy. In other words, instances of both thresholds being touched in the afternoon could primarily stem from situations where the rule-based Noon DAX signal falls into the "undetermined" area. This hypothesis is supported by the following statistics, using the original rule-based Noon DAX strategy parameters:

- Total number of considered cases (days), excluding rows with NaN values for any reason: 6016
- Number of times the morning max threshold is reached in the afternoon, regardless of the strategy signal: 2316
- Number of times the morning min threshold is reached in the afternoon, regardless of the strategy signal: 2078
- Number of times both the morning max and min thresholds are reached in the afternoon, regardless of the strategy signal: 1284
- Number of times neither the morning max nor min threshold is reached in the afternoon, regardless of the strategy signal: 338

If this is the case, the instances where both thresholds are touched in the afternoon are excluded from the rule-based Noon DAX strategy, but they are included in the training of the ANN network. Therefore, there is a possibility that by considering only the cases where the rule-based strategy gives "long" (or "short") signals, the performance of the ANN could improve. The attempt is to relax the rule-based strategy by increasing the ranges of the inner thresholds, sacrificing some accuracy for higher recall, and then filter out these signals using the ANN forecast as a confirmation signal.

To simplify the model structure, define hyperparameters, and automatically save the best-performing epoch/model and training history, I use a custom class based on TensorFlow. Here is how I defined the ANN model:

```
layers_relative_width = [1024, 64, 1]
layers_dropout = [0.8, 0.8, 0]

longTriggerANN = fastANN(
                    X_data,
                    Y_data,
                    split_type = 'random', # 'random' 'sequential'
                    model_relative_width = layers_relative_width,
                    model_dropout = layers_dropout,
                    learning_rate = 0.0003,
                    activation = 'relu',
                    last_layer_activation = 'sigmoid',
                    loss = 'binary_crossentropy',
                    save_best_only = True,
                    early_stop_patience = 150,
                    train_size_rate = 0.7,
                    data_storage_path= r"D:\Dropbox\TRADING\DATA\\",
                    model_name = 'Noon Strategy no rules based - long triggers - minimal 4 kpis - target long signal reached - layers 10-
3-1 - droput 0-0-0 - bacth 512'
                   )

longTriggerANN.network_structure_set_compile()

longTriggerANN.network_training(epochs = 1000, batch_size = 512)
```

I often find that I achieve better and more stable learning curves with models that have the following characteristics:

- They are short, with a maximum of two hidden layers
- They are extremely large, especially for the first two layers
- They have an exaggerated dropout factor for the very wide layers
- It is used a high batch size

These characteristics slightly improve the maximum achievable precision, prevent early overfitting, and contribute to improving validation accuracy. Additionally, using a large batch size stabilizes the learning curve, although it requires more epochs to reach the maximum accuracy level. Memory usage is a limiting factor, which depends on the number of input features.

In this case, the width of the layers is specified as multiplier factors relative to the number of input features. For example, the first layer is 1024 times larger than the number of input features, and the dropout factor is set at 80% of the first layer's width. The total number of layers is determined by the elements of the two series (*layers_relative_width* and *layers_dropout*), plus one for the output layer, which has a number of neurons equal to the width of the target data frame.

Let's proceed with the first attempt using a minimal dataset, as follows:

```
X_data = df_stats_cleaned[[
                    '9_to_12_delta',
                    '9_to_12_min_max_period',
                    '9_to_12_min_before_max',
                    'open_at_12_rel_distance'
                   ]]


Y_data = df_stats_cleaned[['binary_target_long_reached']]
```

resulting in the following learning curve

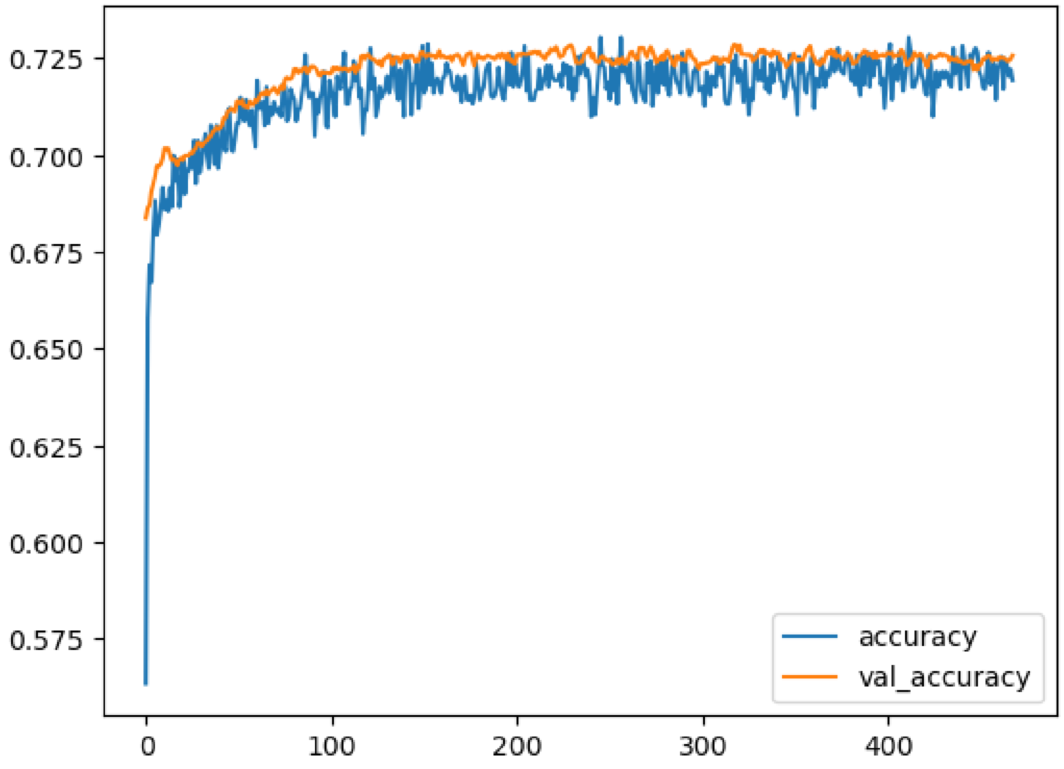


Figure 34 - Learning curve for ANN network for forecast long/short signals - Minimal set of features – Random split

Let’s see the confusion matrix for the “1” value target at the following cut-off values: 0.4, 0.5 and 0.6.

```
longTriggerANN.network_predictions_evaluation(0.4)
```

```
132/132 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.76      0.38      0.51      1708
           1       0.68      0.92      0.78      2501

    accuracy                           0.70      4209
   macro avg       0.72      0.65      0.65      4209
weighted avg       0.72      0.70      0.67      4209
```

```
longTriggerANN.network_predictions_evaluation(0.5)
```

```
132/132 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.70      0.59      0.64      1708
           1       0.75      0.83      0.78      2501

    accuracy                           0.73      4209
   macro avg       0.72      0.71      0.71      4209
weighted avg       0.73      0.73      0.72      4209
```

```
longTriggerANN.network_predictions_evaluation(0.6)
```

```
132/132 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.62      0.73      0.67      1708
           1       0.79      0.70      0.74      2501

    accuracy                           0.71      4209
   macro avg       0.71      0.71      0.71      4209
weighted avg       0.72      0.71      0.71      4209
```

As always, a better overview is given plotting a chart; here following there is the chart of the precision and recall as a function of the cut-off ratio for long triggers.

```
# Inizializza le liste per memorizzare i valori di precision e recall
precision_list = []
recall_list = []
cutoff_values = []

# Ciclo for per calcolare i risultati per ogni valore di cutoff
for cutoff in range(10, 95, 5):
    cutoff_value = cutoff / 100  # Calcola il valore di cutoff da 0.1 a 0.9
    filtered_predictions_results_df, dictionary = longTriggerANN.network_predictions_evaluation(cutoff_value, output_dict=True)

    # Aggiungi i valori di precision e recall alla lista
    precision_list.append(dictionary['1']['precision'])
    recall_list.append(dictionary['1']['recall'])
    cutoff_values.append(cutoff_value)

# Crea un DataFrame pandas con i valori di precision, recall e cutoff
df = pd.DataFrame({'Cutoff': cutoff_values, 'Precision': precision_list, 'Recall': recall_list})

# Plot di precision e recall in funzione di cutoff
plt.plot(df['Cutoff'], df['Precision'], label='Precision')
plt.plot(df['Cutoff'], df['Recall'], label='Recall')
plt.xlabel('Cutoff')
plt.ylabel('Value')
plt.title('Precision and Recall vs Cutoff')
plt.legend()
plt.show()
```

Accepting approximately 70% precision yields a recall around 90%. For a precision of 85%, we would need to lower the recall expectation to around 53%-54%.

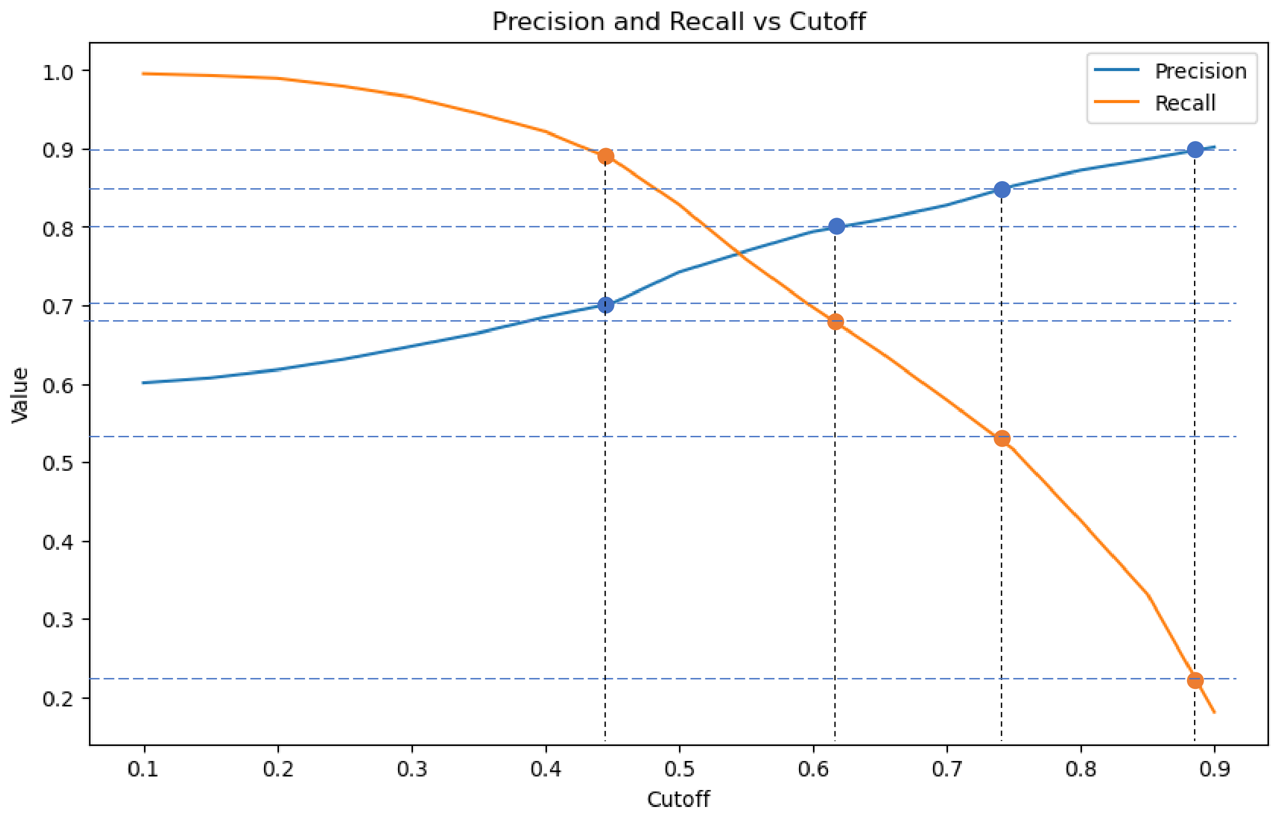


Figure 35 - Precision/recall vs. cut-off ratio curves for ANN network for forecast long/short signals - Minimal set of features

Before proceeding with further attempts on different datasets, I aim to repeat the model training using a sequential split of the original data into training and validation subsets. The chart below displays the resulting learning curves keeping identical model structures and hyperparameters.

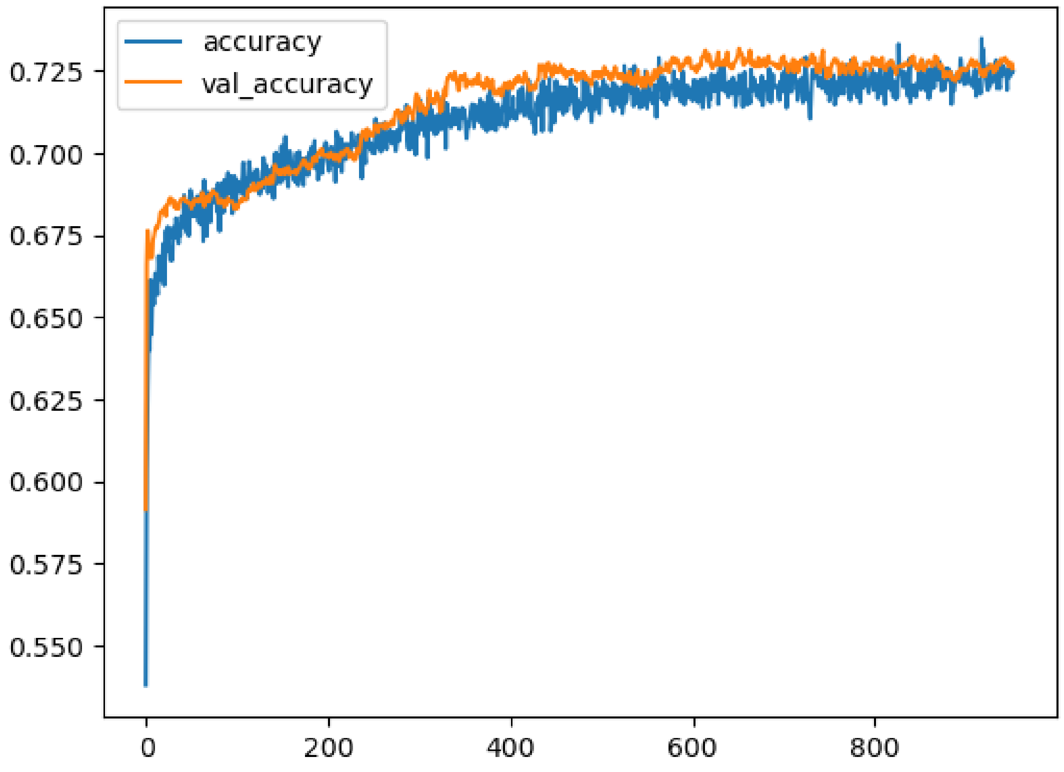


Figure 36 - Learning curve for ANN network for forecast long/short signals - Minimal set of features – Sequential split

As observed, the learning curves obtained with the sequential dataset split closely resemble those obtained with the random dataset split. In the following chart, I have overlaid the precision and recall curves from both training attempts for comparison. The green and purple lines represent the precision and recall of the sequentially split dataset, respectively. These curves are very similar, with the sequentially split dataset showing slightly better recall behavior at a precision level of 80%. This confirms the intuition that in this case, the choice between the two methods of splitting the original dataset into training and validation subsets has minimal influence on the model performance.

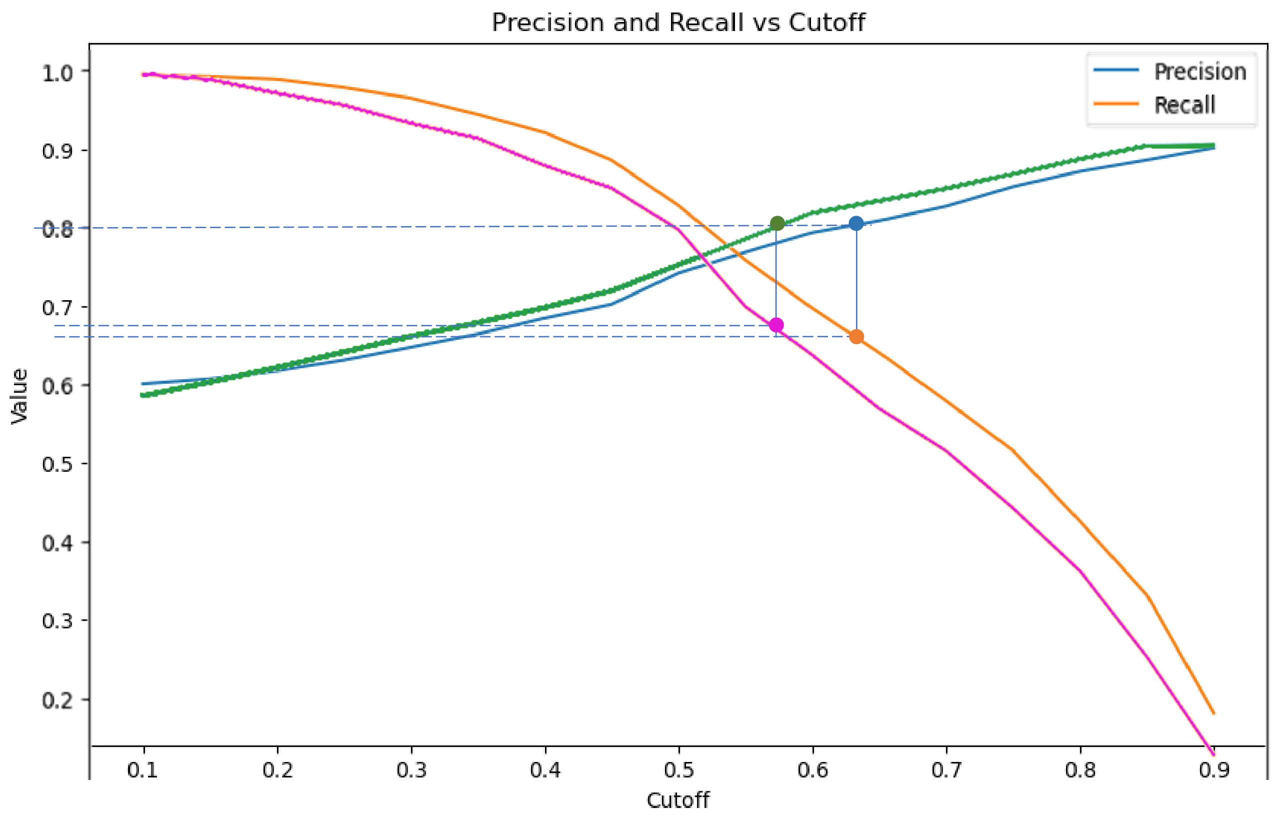


Figure 37 - Precision/recall vs. cut-off ratio curves for ANN network for forecast long/short signals - Minimal set of features - Comparison between randomnly and sequentially dataset split

We can now try adding some features such as a minimal set of previous day's KPIs, the weekday, and the month.

```
X_data = df_stats_cleaned[[
                    '9_to_12_delta',
                    '9_to_12_min_max_period',
                    '9_to_12_min_before_max',
                    'open_at_12_rel_distance',
                    '9_to_12_delta_day_-1',
                    '9_to_12_min_max_period_day_-1',
                    '9_to_12_min_before_max_day_-1',
                    'open_at_12_rel_distance_day_-1',
                    '9_to_12_delta_day_-2',
                    '9_to_12_min_max_period_day_-2',
                    '9_to_12_min_before_max_day_-2',
                    'open_at_12_rel_distance_day_-2',
                    'gain_day_-1',
                    'gain_day_-2',
                    'weekly_day',
                    'month'
                  ]]
```

Getting the following learning curve

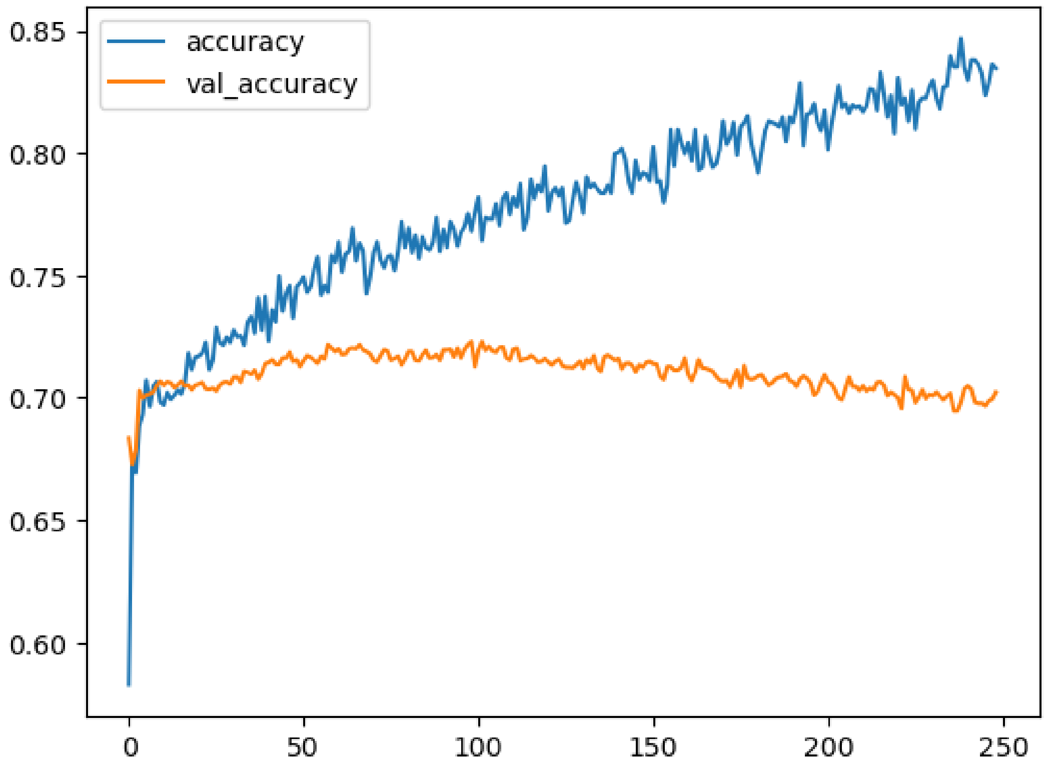


Figure 38 - Learning curve for ANN network for forecast long/short signals - Minimal set of features extended with some previous day features

and precision/recall versus cutoff ratio.

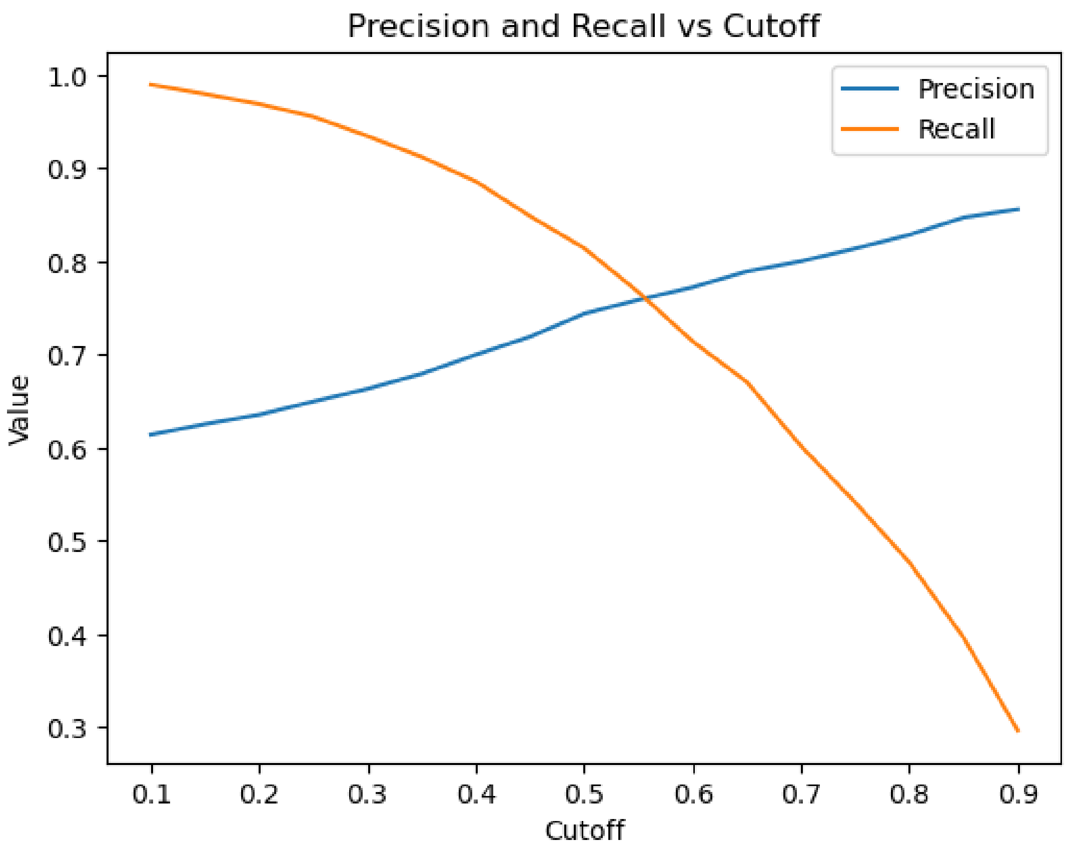


Figure 39 - Precision/recall vs. cut-off ratio curves for ANN network for forecast long/short signals - Minimal set of features extended with some previous day features

The charts above both indicate a slight deterioration in performance.

Differently, the following attempt involves adding contextual market behavior KPIs to the base features.

```
X_data = df_stats_cleaned[[
                  '9_to_12_delta',
                  '9_to_12_min_max_period',
                  '9_to_12_min_before_max',
                  'open_at_12_rel_distance',
                  '9_to_12_delta_day_-1',
                 'CO_mean',
                 'HL_mean',
                 'Close_delta_mean',
                 'High_delta_mean',
                 'Low_delta_mean',
                 'CO_std',
                 'HL_std',
                 'Close_delta_std',
                 'High_delta_std',
                 'Low_delta_std'
                 ]]
```

resulting in the following learning curves and performances.

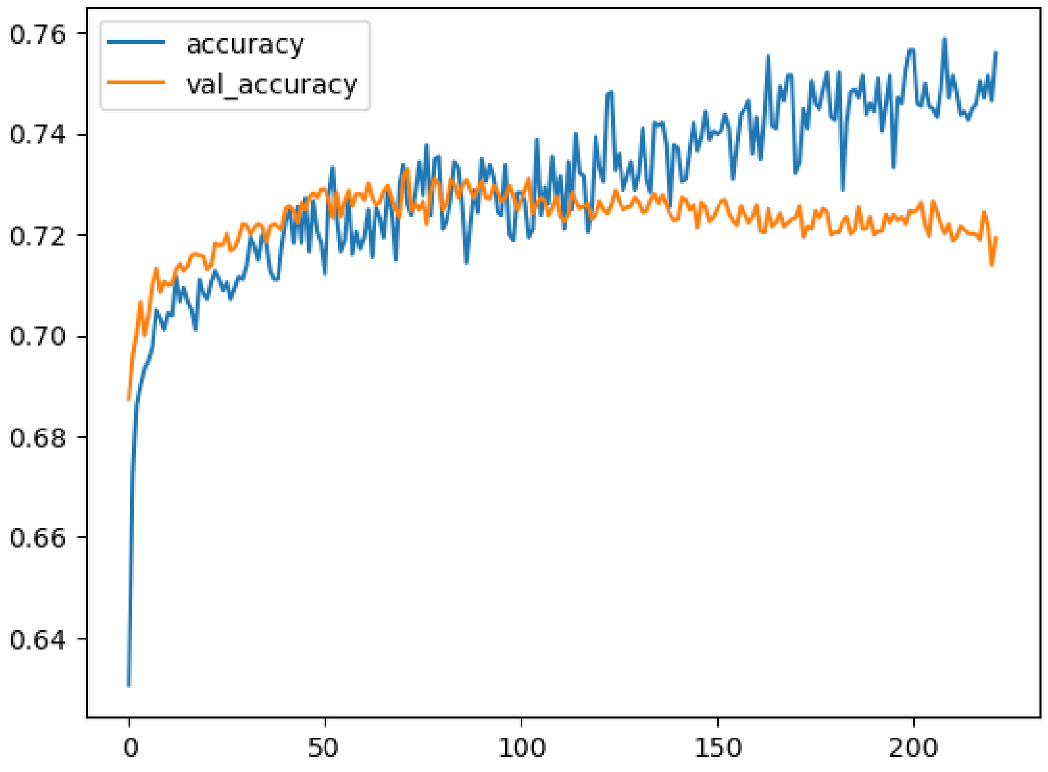


Figure 40 - Learning curve for ANN network for forecast long/short signals - Minimal set of features extended with some morning market behavioral ones

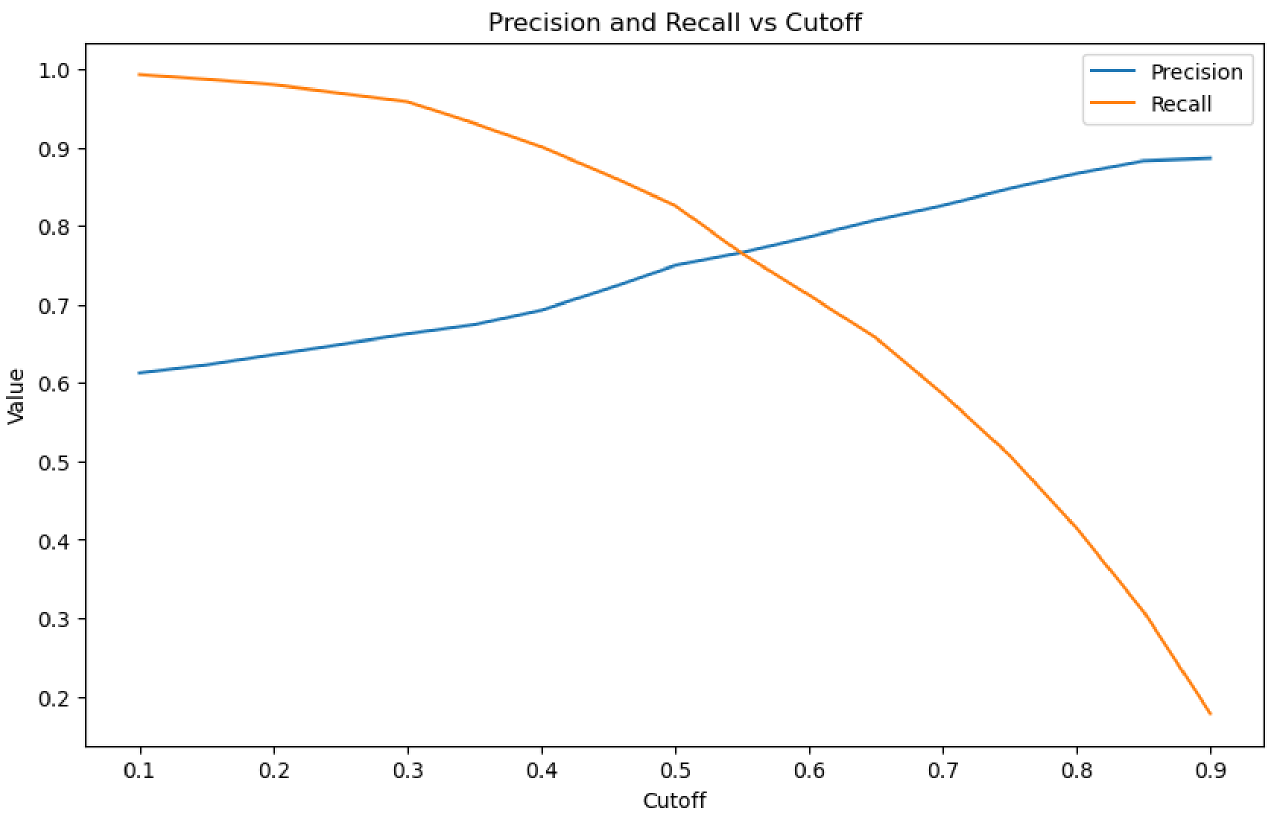


Figure 41 - Precision/recall vs. cut-off ratio curves for ANN network for forecast long/short signals - Minimal set of features extended with some morning market behavioral ones

Instead, including all the features, but the morning daily behavioral ones as following

```
X_data = df_stats_cleaned[[
                '9_to_12_delta',
                '9_to_12_min_max_period',
                '9_to_12_min_before_max',
                'open_at_12_rel_distance',
                '9_to_12_delta_day_-1',
                '9_to_12_min_max_period_day_-1',
                '9_to_12_min_before_max_day_-1',
                'open_at_12_rel_distance_day_-1',
                '9_to_12_delta_day_-2',
                '9_to_12_min_max_period_day_-2',
                '9_to_12_min_before_max_day_-2',
                'open_at_12_rel_distance_day_-2',
                'gain_day_-1',
                'gain_day_-2',
                'weekly_day',
                'month',
                '9_to_12_max_minus_9_to_12_max_day_-1',
                '9_to_12_max_minus_9_to_12_max_day_-2',
                '9_to_12_max_minus_9_to_12_min_day_-1',
                '9_to_12_max_minus_9_to_12_min_day_-2',
                '9_to_12_min_minus_9_to_12_max_day_-1',
                '9_to_12_min_minus_9_to_12_max_day_-2',
                '9_to_12_min_minus_9_to_12_min_day_-1',
                '9_to_12_min_minus_9_to_12_min_day_-2',
                '9_to_12_max_minus_after_12_max_day_-1',
                '9_to_12_max_minus_after_12_max_day_-2',
                '9_to_12_max_minus_after_12_min_day_-1',
                '9_to_12_max_minus_after_12_min_day_-2',
                '9_to_12_min_minus_after_12_max_day_-1',
                '9_to_12_min_minus_after_12_max_day_-2',
               '9_to_12_min_minus_after_12_min_day_-1',
               '9_to_12_min_minus_after_12_min_day_-2',
               '9_to_12_delta_div_9_to_12_delta_day_-1',
               '9_to_12_delta_div_9_to_12_delta_day_-2',
               '9_to_12_delta_div_after_12_min_max_delta_day_-1',
               '9_to_12_delta_div_after_12_min_max_delta_day_-2'
              ]]
```

we get even less good results.

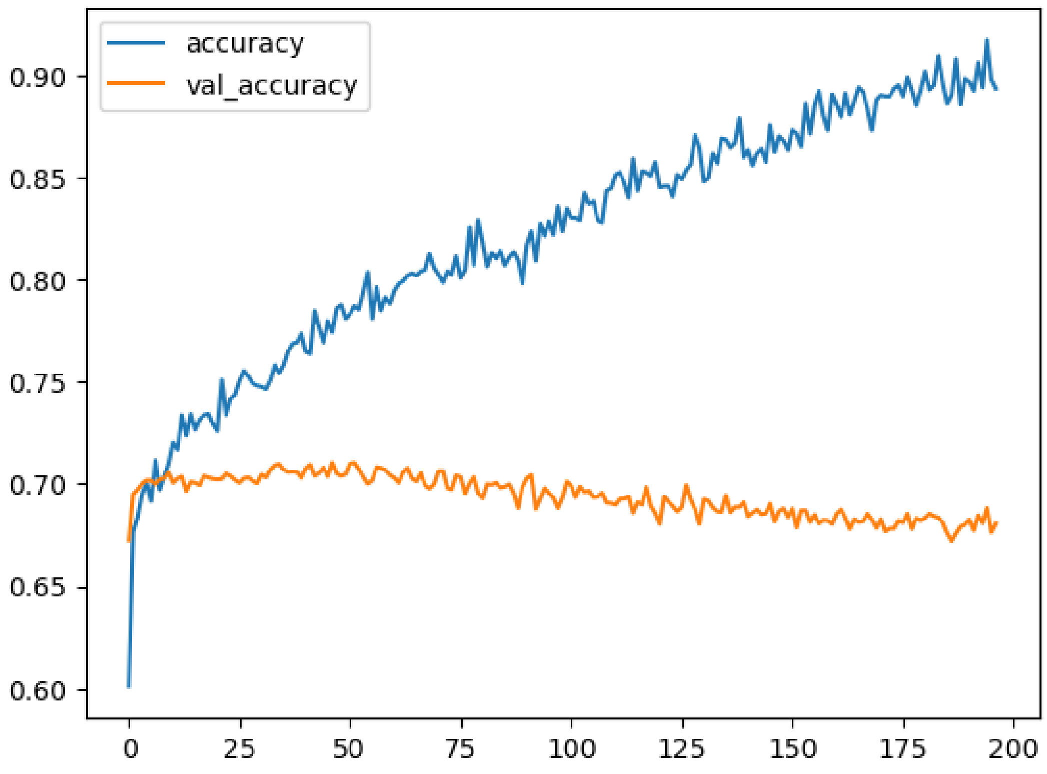


Figure 42 - Learning curve for ANN network for forecast long/short signals - Extended set of features with previous two days ones

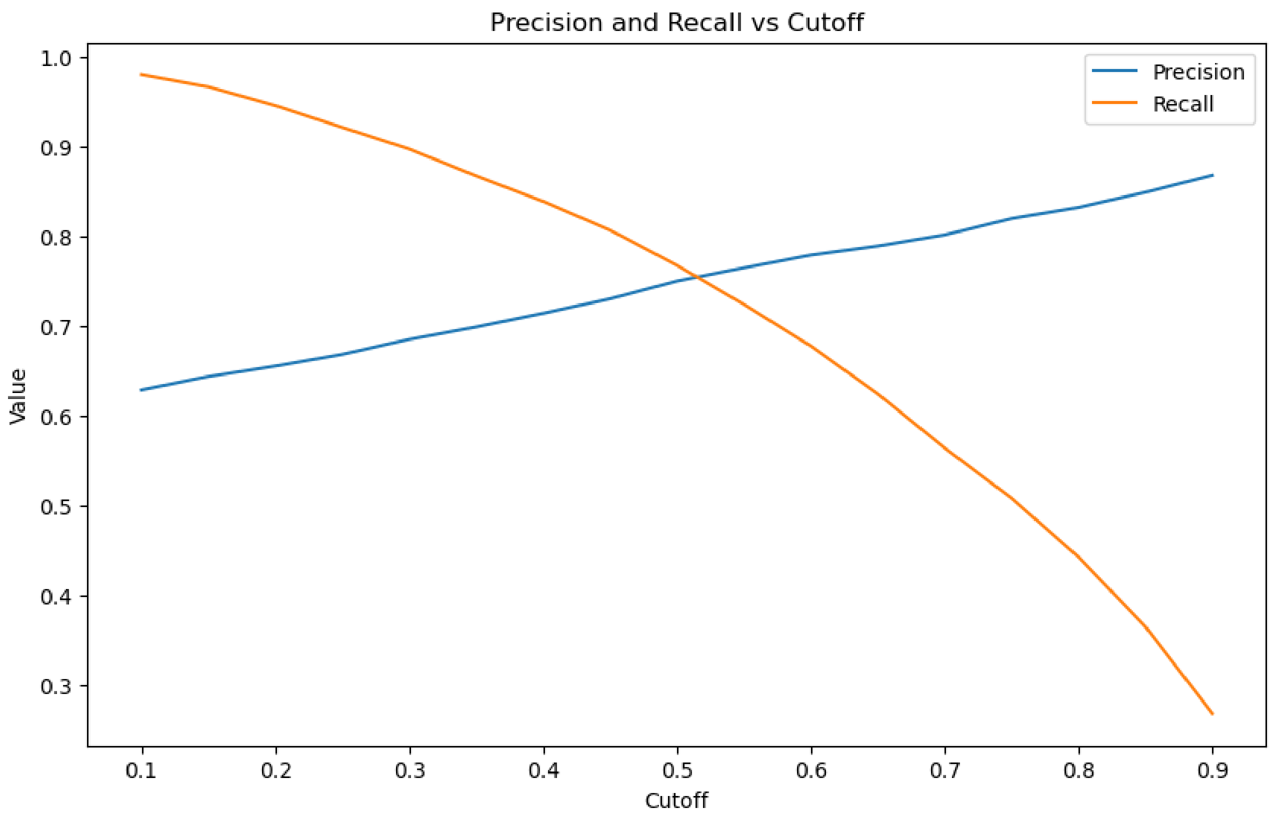


Figure 43 - Precision/recall vs. cut-off ratio for ANN network for forecast long/short signals - Extended set of features with previous two days ones

Now let's combine the two signals by filtering the dataset to include only the rows with the base strategy forecast as "long". We'll work with an ANN model trained on the simplest dataset since it has slightly better performance.

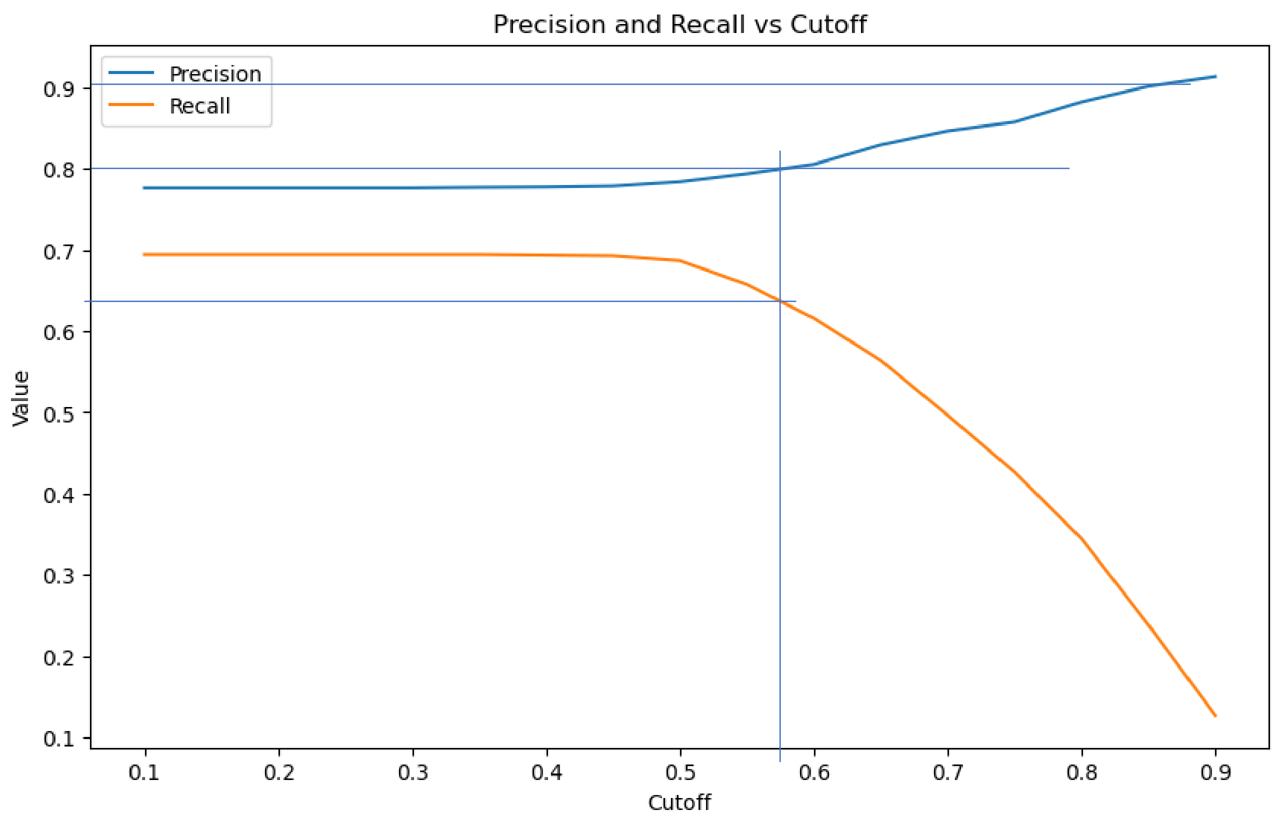


Figure 44 - Precision/recall vs. cut-off ratio for ANN network for forecast long/short signals – Minimal set of features – Mixed with basic rule-based DAX Noon Delta Signal

WE got no significant improvement compared to using the ANN model alone. The flattening behavior on the left occurs because for low cut-off values, we are essentially not using the ANN model forecasts, relying instead on the rule-based signals.

11. Conclusions

I believe that Paolo and his trading friends would be pleased with the statistical performance of the DAX Noon Delta Signal as described at its basic level. I challenge them to develop a strategy around it, as using it alone may not be as profitable, and to conduct backtesting accordingly. I look forward to their follow-up post here as a continuation of this study.

As we come to the end of this journey, before parting ways, I would like to show how the DAX Noon Delta Signal appears when plotting the levels on a candlestick chart. You can find the relevant Pine Script code for TradingView in the Annexes chapter.

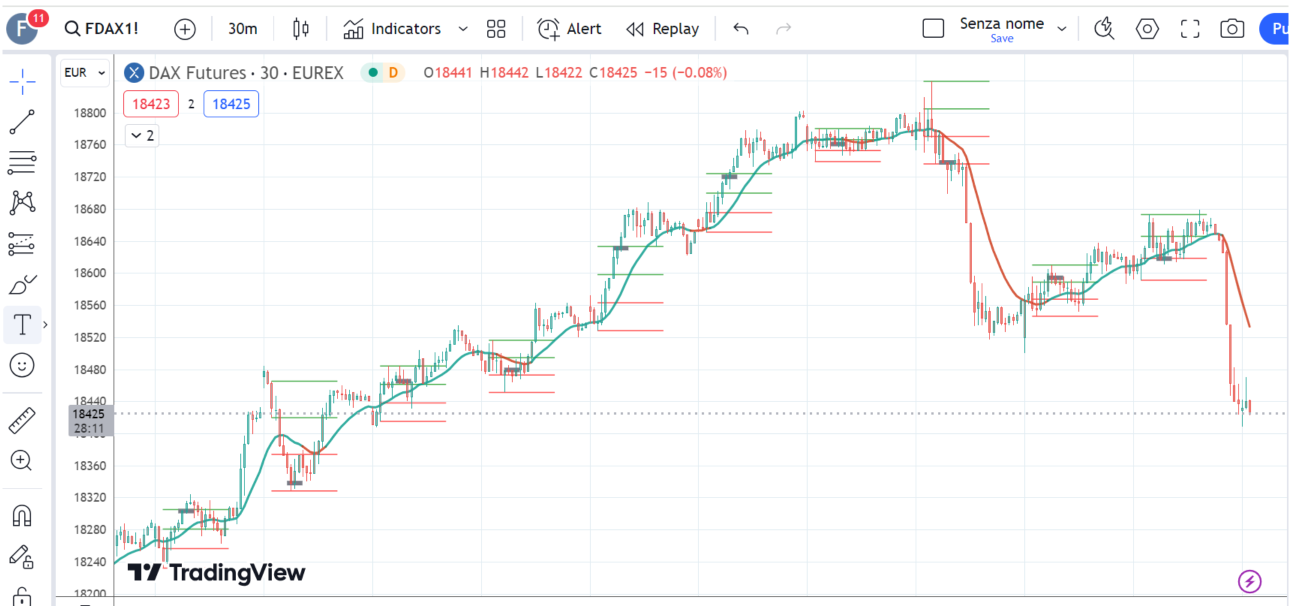


Figure 45 - TradingView screenshot of DAX Futures - 30 minutes TF - Indicators: DAX Noon Delta Signal and Kernel/Lorentzian Classifier

The chart depicts the future of the DAX rather than the pure DAX index, providing more data points over the daily 24 hours. You can identify the morning minimum, maximum, and inner thresholds (marked by red and green lines), as well as the Noon opening price (small grey line). The changing color line represents the

Kernel filter of the Lorentzian Classifier, a highly effective indicator available for free on TradingView developed by Justin Dehorty. Can you identify any advantages in combining these two indicators?

12. Annexes

Annex A: full notebook

You can download my Jupyter Notebook from Github at this URL: https://github.com/fede72bari/DAX-Noon-Delta-Strategy-Jupyter-Notebook

Annex B: DAX Noon Delta indicator in Pine 5 code

```
//@version=5
indicator(title="Noon Strategy", overlay=true)
// (c) fede72bari@gmail.com

// Accedi ai valori della candela corrente
float currentOpen = open
float currentHigh = high
float currentLow = low
float currentClose = close

// Accedi ai valori della candela precedente
float previousOpen = open[1]
float previousHigh = high[1]
float previousLow = low[1]
float previousClose = close[1]


// Definisci l'ora desiderata (9:00) come una costante
int desiredHour = 9
int desiredMinute = 0

// Imposta il fuso orario di Roma
const string ROME_TIMEZONE = "Europe/Rome"
// const string ROME_TIMEZONE = "America/Los_Angeles"

// Ottieni l'offset UTC per il fuso orario di Roma
// utcOffsetRome = timezone.offset(ROME_TIMEZONE)

// Ottieni l'orario di apertura della barra nel timeframe del grafico corrente e nel fuso
orario di Roma
chartBarOpenTime = time(timeframe.period, "", ROME_TIMEZONE)

// log.info(str.format_time(chartBarOpenTime, "yyyy-MM-dd HH:mm", ROME_TIMEZONE))


// Estrai l'ora e i minuti dall'orario di apertura della barra nel fuso orario di Roma
int chartHour = hour(chartBarOpenTime, ROME_TIMEZONE)
int chartMinute = minute(chartBarOpenTime, ROME_TIMEZONE)


// Calcola il numero di minuti tra l'ora corrente e l'ora desiderata
// int minutesSinceDesiredTime = (hour - desiredHour) * 60 + (minute - desiredMinute)
int minutesSinceDesiredTime = (chartHour - desiredHour) * 60 + (chartMinute - desiredMinute)

// Converte la stringa del periodo di tempo in minuti
// Converte il periodo di tempo del grafico corrente in minuti
int periodInMinutes = timeframe.in_seconds() / 60

// Calcola il numero di candele indietro necessarie per raggiungere l'ora desiderata
int candlesBack = minutesSinceDesiredTime / periodInMinutes



// Controlla se la candela attuale è quella delle 12:00
// bool isNoonCandle = hour == 12 and minute == 0
bool isNoonCandle = chartHour == 12 and chartMinute == 0

// Accedi ai valori della candela corrispondente all'ora desiderata
float desiredOpen = open[candlesBack]
float desiredHigh = high[candlesBack]
float desiredLow = low[candlesBack]
float desiredClose = close[candlesBack]
hi = 1.1
lo = 1.1



// log.info("close -2: " + str.tostring((close[2])))
// Stampa i valori delle candele corrente e precedente sulla console di output solo se è la
candela delle 12:00
if isNoonCandle


    // Inizializza le variabili per il massimo e il minimo
    float hi = na
    float lo = na

    // Ciclo per trovare il massimo e il minimo tra le barre precedenti
    for i = 1 to candlesBack
        // Aggiorna il massimo e il minimo
        hi := na(hi) ? high[i] : math.max(hi, high[i])
        lo := na(lo) ? low[i] : math.min(lo, low[i])

    delta = hi - lo
    delta_3 = delta / 3

    // Definisci l'ora di chiusura del mercato
    int marketCloseHour = 17
    int marketCloseMinute = 30

    // Calcola il numero di minuti rimanenti fino alla chiusura del mercato
    int minutesRemainingToMarketClose = (marketCloseHour - hour) * 60 + (marketCloseMinute -
minute)
```

```
    // Calcola il numero di barre rimanenti nella giornata
    int barsRemainingToday = minutesRemainingToMarketClose / periodInMinutes


    // Traccia le linee di massimo e minimo
    line.new(x1=bar_index - candlesBack, y1=hi, x2=bar_index+barsRemainingToday, y2=hi,
color=color.green)
    line.new(x1=bar_index - candlesBack, y1=lo, x2=bar_index+barsRemainingToday, y2=lo,
color=color.red)

    line.new(x1=bar_index - candlesBack, y1=hi - delta_3, x2=bar_index+barsRemainingToday,
y2=hi - delta_3, color=color.green)
    line.new(x1=bar_index - candlesBack, y1=lo + delta_3, x2=bar_index+barsRemainingToday,
y2=lo + delta_3, color=color.red)

    line.new(x1=bar_index - 4, y1=open, x2=bar_index+4, y2=open, color=color.black, width=4)
```