Documentation

Leverage: Friend or Enemy

Data exploration, Strategy backtesting and Analysis

Viktor Nagy Abonyi

10/20/2023

A fundamental analysis that empowers investors with valuable insights into the role of leverage in their investment strategies, ultimately enabling more informed decision-making in their financial pursuits.

Model: Leverage impact assessment model

The model is designed to evaluate and visualize the effects of leverage on an investment strategy applied to a user picked asset. The model aims to provide insights into how leverage influences investment returns and risks, catering to the needs of investors who seek to understand the implications of leverage in their investment decisions.

Comprehensive documentation is maintained, covering data sources, methodology, data transformations and assumptions. The documentation aims to ensure transparency and credibility in the analysis. Most of the visualizations are interactive and downloadable.

The work is divided into 4 sections:

- Visualization of price levels with leverage (explanation)
- Strategy execution and backtesting (data exploration and manipulation)
- Analyzing results (analysis)
- Executive summary (separate document)

Data Extraction

Initial data is extracted from a reliable source that faciliates the collection of essential financial information to start our analysis with.

- Source: Yahoo Finance
- Type: OHLC Data

The input parameters are customizable:

- Ticker
- Interval (number of days to analyze)
- Leverage levels
- Position size
- Investment / traging Strategy (more on that later)

Explanation

This section highlights the core concept of leverage, which enables greater asset control with less capital, but also emphasizes the associated increased risk. We clarify that while leverage can amplify both potential gains and losses, it must be used with caution. Visualization models in this section are simplified and do not account for fees and costs associated with investing.

Data Manipulation

We utilize the power of data analytics to manipulate the data and present detailed aggregated statistics which we use to answer our questions.

A Simple Moving Average Cross (13-21) strategy is applied to the collected data. We conduct three sets of simulations, testing 3 position sizes (100%, 50%, and 10%) with different leverage values (Spot, 5x, 10x, 20x, 50x, and 100x). Each simulation generates detailed statistical reports with graphs, including information on returns, win rates, profit factors, maximum drawdowns, volatility, Sharpe Ratios, and more. These reports are aggregated into comprehensive dataframes, creating a comparison overview of strategy performance under varying conditions. Any strategy can be applied: the flexibility of our model enables the exploration with custom simulations and analyses by changing input values and implementing different strategies. *A 0.2% fee is applied to each trade.*

Analysis

The simulations are conducted with the SMA Cross strategy on TSLA daily data for 1000 days, aggregated into comprehensive dataframes, suitable for further analysis. We try to answer the following questions:

- How does leverage affect investment risk and returns?
- What are the advantages and risks of using leverage in investments?
- How does position size impact portfolio returns when using leverage?

Presentation

An executive summary is provided alongside this documentation in a separate .pdf document, encapsulating the key findings and insights from the entire documentation, offering a concise overview of the analysis, outcomes, and recommendations related to the role of leverage in investment.

Additional comments:

- We decided to explore a subject and tools we are less familiar with. This pushed us to dive into research and learn new concepts, demonstrating our willingness to take on fresh challenges and expand our skills.
- Our model is designed with flexibility in mind. Users have the capability to adjust input data and the freedom to apply various strategies for backtesting. As a baseline, we present a standard simulation with a single instance featuring an SMA 13-21 crossover strategy.
  The topic is broad. For simplicity and to maintain focus, we don't introduce unnecessary data complexities. These options may be considered for future expansions.

Opportunities for Further Research:

A promising avenue for future research lies in the optimization of trading strategies based on leverage levels. This entails determining the optimal leverage level for each specific trading strategy and its associated position size. This research can contribute to more effective and precise decision-making for traders and investors, aiming to strike a balance between maximizing returns and managing risk.

Imports and Definitions

Our work begins by importing necessary libraries, setting up configurations, and initializing the Jupyter notebook environment for displaying graphs.

In [1]:
```
# %%capture
# Some modules are missing from the default Anaconda installation:
# - pandas_datareader library for data retrieval
# - yfinance for fetching financial data from Yahoo Finance
# - the backtesting library for strategy backtesting and aggregated results data for further analysis
!pip install pandas_datareader
!pip install yfinance
!pip install backtesting

# Dependencies
import warnings
import datetime as dt
import numpy as np
import pandas as pd
from pandas_datareader import data as pdr
import plotly.offline as pyo
import plotly.graph_objects as go
import plotly.express as px
import yfinance as yf
from backtesting import Backtest, Strategy
from backtesting.lib import crossover
from backtesting.test import SMA

# Initialize Jupyter notebook to be able to see the graphs
pyo.init_notebook_mode(connected=True)

# Quick fix for yfinance
yf.pdr_override()

# Hide warnings
warnings.filterwarnings("ignore", module="backtesting")
```

Input parameters

- **ticker**: The ticker symbol used for the analysis. Available tickers can be accessed here.
- **days**: The number of days of historical data to use for the backtesting.
- **leverages**: A list of leverage values for assessment (e.g., [5, 10, 20, 50, 100]).
- **position_size_pct**: The percentage of your capital to allocate to each trade (e.g., 100% to invest all available capital).

In [2]:
```
# inputs
ticker = "TSLA"
days = 1000
leverages = [5, 10, 20, 50, 100]
position_size_pct = 10

# Constraints:
# - `position_size_pct` should be between 0 and 100.

# Creating and storing leverage labels for x-axis
leverage_texts = ["Spot"] + [f"{x}x" for x in leverages]
```

Data retrieval

The code acquires historical price data for the designated asset symbol by leveraging the Yahoo Finance API. This data extraction encompasses the user-defined historical timeframe and is then meticulously cataloged within a Pandas DataFrame denoted as 'df.'

```
# construct dataframe
end = dt.datetime.now()
start = end - dt.timedelta(days=days)
df = pdr.get_data_yahoo(ticker, start, end)
df
```

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

Date	Open	High	Low	Close	Adj Close	Volume
2021-01-22	278.103333	282.666656	276.206665	282.213318	282.213318	60199500
2021-01-25	285.000000	300.133331	279.606659	293.600006	293.600006	123520200
2021-01-26	297.126678	298.633331	290.533325	294.363342	294.363342	69394800
2021-01-27	290.116669	297.166656	286.220001	288.053345	288.053345	82002000
2021-01-28	273.333344	282.666656	267.000000	278.476654	278.476654	79134000
...	...	...	...	...	...	...
2023-10-12	262.920013	265.410004	256.630005	258.869995	258.869995	111508100
2023-10-13	258.899994	259.600006	250.220001	251.119995	251.119995	102073800
2023-10-16	250.050003	255.399994	248.479996	253.919998	253.919998	88917200
2023-10-17	250.100006	257.179993	247.080002	254.850006	254.850006	93562900
2023-10-18	252.699997	254.630005	242.080002	242.679993	242.679993	121459300

690 rows × 6 columns

Part One: Visualization of price levels with leverage

Before we delve into our data, let's take a brief detour to explain what leverage is. In this section, we provide explanation and general visual representations of the impact of leverage on investment outcomes, allowing investors to understand the potential gains and losses associated with different leverage levels.

Leverage is a fundamental concept in the world of finance and investing. At its core, it allows traders and investors to control a larger position in an asset with a smaller amount of capital. It's like having a financial multiplier, where potential gains and potential losses are both amplified.

In simpler terms, leverage lets you 'borrow' capital to make larger investments than you could with your own funds alone. However, there's a catch: while it offers the opportunity to magnify returns, it also comes with increased risk. Leverage can be a friend when used wisely, helping traders achieve their financial goals, but it can also be an enemy if not respected.

To illustrate this with an example, imagine you have $1000 in capital, and you decide to apply 10x leverage. With this leverage, you can effectively control a position size of $10,000 ($1,000 x 10). This means that your potential gains or losses on the investment are magnified by a factor of 10. Now, let's say the asset you invest in increases in value by 5%. Without leverage, your profit would be $50 ($1,000 x 0.05). However, with 10x leverage, your profit is multiplied, resulting in a $500 profit ($10,000 x 0.05). Conversely, if the asset's value decreases by 5%, your loss without leverage would be $50. But with 10x leverage, your loss is also magnified, resulting in a $500 loss.

The Effect of Leverage on Investment with $1,000 Capital

The following table's purpose is to provide a clear visual representation of how leverage can affect investment outcomes, offering insights into the potential gains and losses associated with different leverage levels, all based on a $1,000 capital and a 5% price change scenario. It allows users to quickly compare the outcomes of varying leverage strategies and make informed decisions regarding their investments. As leverage increases, the effective position size grows significantly, potentially magnifying gains. However, this also escalates the potential for losses, as higher leverage multiplies the impact of price changes.

Disclaimer: The table does not account for fees and costs associated with investing.

```
# Define leverage levels, capital, and price change percentage
leverage_levels = [1, 5, 10, 20]
leverage_labels = ["Spot", "5x", "10x", "20x"]
capital = 1000
price_change_percent = 5

# Calculate potential gains and losses based on the given parameters
potential_gains = [
    f"${capital * (level * price_change_percent / 100):.2f}"
    for level in leverage_levels
]
potential_losses = [
    f"${-capital * (level * price_change_percent / 100):.2f}"
    for level in leverage_levels
]
price_changes = [f"{price_change_percent}%" for _ in leverage_levels]
effective_positions = [f"${capital * level:.2f}" for level in leverage_levels]

# Create a table using Plotly to visualize the effects of leverage
fig = go.Figure(
    data=[
        go.Table(
            header=dict(
                values=[
                    "Leverage",
                    "Effective Position Size",
                    "Asset's % Price Change",
                    "Potential Gain",
                    "Potential Loss",
                ],
                align="center",
                fill_color="lightgray",
            ),
            cells=dict(
                values=[
                    leverage_labels,
                    effective_positions,
                    price_changes,
                    potential_gains,
                    potential_losses,
                ],
                align=["center", "left", "center", "left", "left"],
            ),
        )
    ]
)

# Customize the layout of the table
fig.update_layout(
    title="Effect of Leverage on Investment with $1,000 Capital",
    title_x=0.5,
    margin=dict(l=50, r=50, b=30, t=50, pad=0),
    width=560,
    height=225,
)
# Display the table
fig.show()
```

Effect of Leverage on Investment with $1,000 Capital

Leverage	Effective Position Size	Asset's % Price Change	Potential Gain	Potential Loss
Spot	$1000.00	5%	$50.00	$-50.00
5x	$5000.00	5%	$250.00	$-250.00
10x	$10000.00	5%	$500.00	$-500.00
20x	$20000.00	5%	$1000.00	$-1000.00

Visual demonstration of leverage levels

The impact of leverage on the potential price movement range can be expressed mathematically as: $P_n = \frac{P}{L^n}$

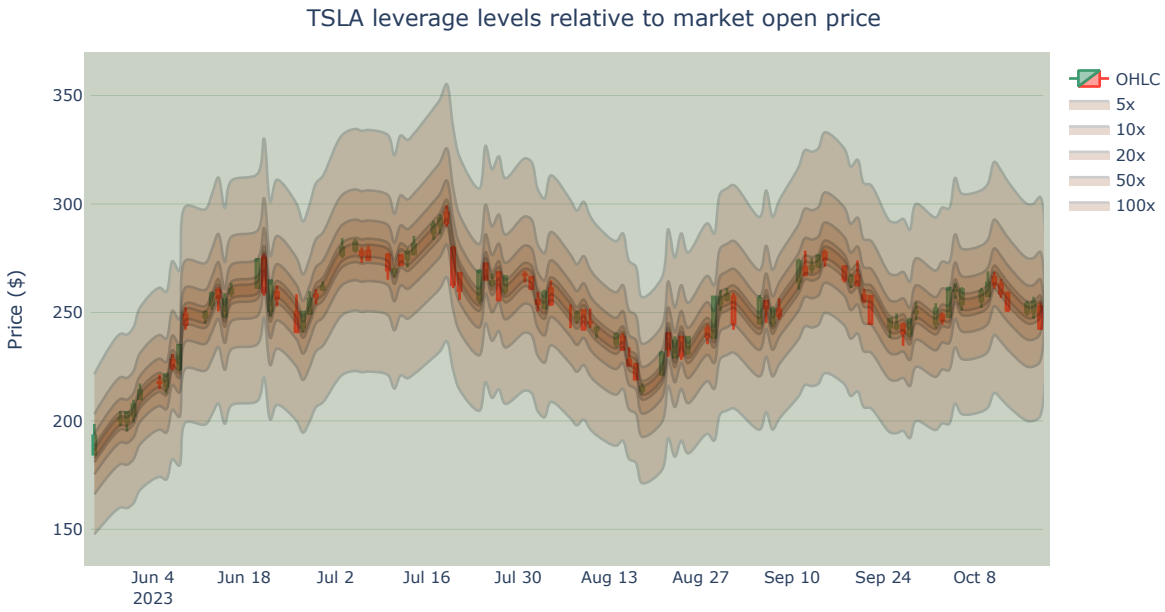
- P_n represents the potential price movement range with leverage level (n).
- P is the potential price movement range without leverage (in spot trading).
- L denotes the leverage factor.
- n is the leverage level.

The formula illustrates that as the leverage level n increases, the potential price movement range P_n narrows exponentially. Essentially, as n grows, the field of movement becomes progressively limited, indicating the exponential effect of leverage on controlling price variability.

To provide a practical illustration of the impact of leverage, we've included a visual representation of how different leverage factors influence potential price ranges for the specified asset. For each leverage factor, you'll notice a shaded area added to the chart, representing the potential price range within which the asset might fluctuate. The shading intensity corresponds to the level of leverage, with darker shading indicating higher leverage factors and, consequently, a narrower potential price range. For instance, imagine you're buying an asset, and its price hits the upper shaded boundary. In this scenario, you could potentially double your initial investment. On the other hand, if the price hits the lower shaded boundary, it could result in a complete loss of your invested capital.

It's important to emphasize that this visualization serves as a simplified demonstration model and does not take into account fees and costs associated with investing.

```
# Copy the most recent 100 data points into a new DataFrame
df_demo = df.tail(100).copy()

# Calculate upper and lower price levels for each leverage value
for leverage in leverages:
    df_demo["{}x_upper".format(leverage)] = df_demo["Open"] * (1 + (1 / leverage))
    df_demo["{}x_lower".format(leverage)] = df_demo["Open"] * (1 - (1 / leverage))

# Reverse the order of the index for the chart
x_rev = df_demo.index[::-1]

# Create a Candlestick chart for the historical price data
chart_demo = go.Figure(
    data=[
        go.Candlestick(
            x=df_demo.index,
            open=df_demo["Open"],
            high=df_demo["High"],
            low=df_demo["Low"],
            close=df_demo["Close"],
            name="OHLC",
        )
    ]
)

# Add shaded areas to the chart for each leverage factor
for e in leverages:
    y_rev = df_demo[f"{e}x_lower"][::-1]

    chart_demo.add_trace(
        go.Scatter(
            x=np.concatenate([df_demo.index, x_rev]),
            y=np.concatenate([df_demo[f"{e}x_upper"], y_rev]),
            fill="toself",
            fillcolor="#8b4513",
            opacity=0.2,
            line_color="black",
            showlegend=True,
            name=f"{e}x",
            line_shape="spline",
        )
    )

# Customize the chart layout and appearance
chart_demo.update_layout(
    title=f"{ticker} leverage levels relative to market open price",
    title_x=0.5,
    xaxis_tickfont_size=12,

    yaxis=dict(title="Price ($)", titlefont_size=14, tickfont_size=12),
    margin=dict(l=50, r=50, b=100, t=50, pad=5),
    xaxis_rangeslider_visible=False,
    xaxis_showgrid=False,
    plot_bgcolor="#cad2c5",
    yaxis_gridcolor="#84a98c",
)

# Display the chart
chart_demo.show()
```



Part Two: Strategy execution and backtesting

In the world of finance, the line between an investor and a trader can be blurred, as every investor essentially takes on the role of a trader when buying or selling assets, and conversely, a trader becomes an investor with each trade. For this reason our analysis relies on strategy simulations using historical price data.

The following code block showcases a sample trading strategy using moving averages to analyze asset price movements. It also performs backtesting to assess the strategy's performance.

This code example provides a starting point for exploring trading strategies. Users can customize the `SmaCross` strategy by adjusting parameters, or they can create their own strategies tailored to their specific requirements. It serves as a practical template for strategy development and evaluation.

To use this code with a custom strategy, users can replace the `SmaCross` class with their own trading logic and modify parameters to fit their trading preferences and objectives.

In [6]:
```
class SmaCross(Strategy):
    """
    SMA Cross Strategy:
        Trigger: SMA1 (n1) crosses SMA2 (n2)
        Stop: Liquidation level (eg. -20% for 5x, none for Spot)
        Target: 2x liquidation level (eg. +40% for 5x, none for Spot)
        Position is closed and reversed on opposite signal.
    """

    n1 = 13
    n2 = 21
    size = 0.9999
    leverage = 5

    def init(self):
        close = self.data.Close
        self.sma1 = self.I(SMA, close, self.n1)
        self.sma2 = self.I(SMA, close, self.n2)

    def next(self):
        # Calculate the price and gap for leverage
        p = self.data.Close[-1]
        gap = p * (1 / self.leverage)

        if crossover(self.sma1, self.sma2):
            if self.position.is_long or not self.position:
                self.position.close()
```

```
            # Buy with defined size, take profit, and stop loss
            self.buy(size=self.size, tp=p + 2 * gap, sl=p - gap)
        elif crossover(self.sma2, self.sma1):
            if self.position.is_short or not self.position:
                self.position.close()
            # Sell short with defined size, take profit, and stop loss
            self.sell(size=self.size, tp=p - 2 * gap, sl=p + gap)


def run_backtests(position_size=0.9999):
    """Initialize a backtest with the strategy"""
    backtest = Backtest(df, SmaCross, cash=10000, commission=0.002)
    report = backtest.run(size=position_size)
    df_report = pd.DataFrame({f"Spot (no leverage)": report})

    print(
        f'Strategy: {ticker} {report["_strategy"]}, Position size: {position_size*100}%'
    )
    backtest.plot(
        plot_pl=False,
        plot_volume=False,
        smooth_equity=True,
        superimpose=False,
        show_legend=True,
    )

    for i, x in enumerate(leverages):
        x_text = f"{x}x leverage"
        # Run backtests with different leverage levels
        backtest = Backtest(df, SmaCross, cash=10000, commission=0.002, margin=1 / x)
        report = backtest.run(leverage=x, size=position_size)
        df_report[x_text] = pd.DataFrame({x_text: report})

        print(f'Strategy: {ticker} {report["_strategy"]}')
        backtest.plot(
            plot_pl=False,
            plot_volume=False,
            smooth_equity=True,
            superimpose=False,
            show_legend=False,
        )

    return df_report
```

Simulation 1: Running backtests with 100% position size (combined results)

This code segment allows users to evaluate the strategy's performance when operating with a full 100% position size, offering a comprehensive view of the strategy's potential outcomes under different leverages.

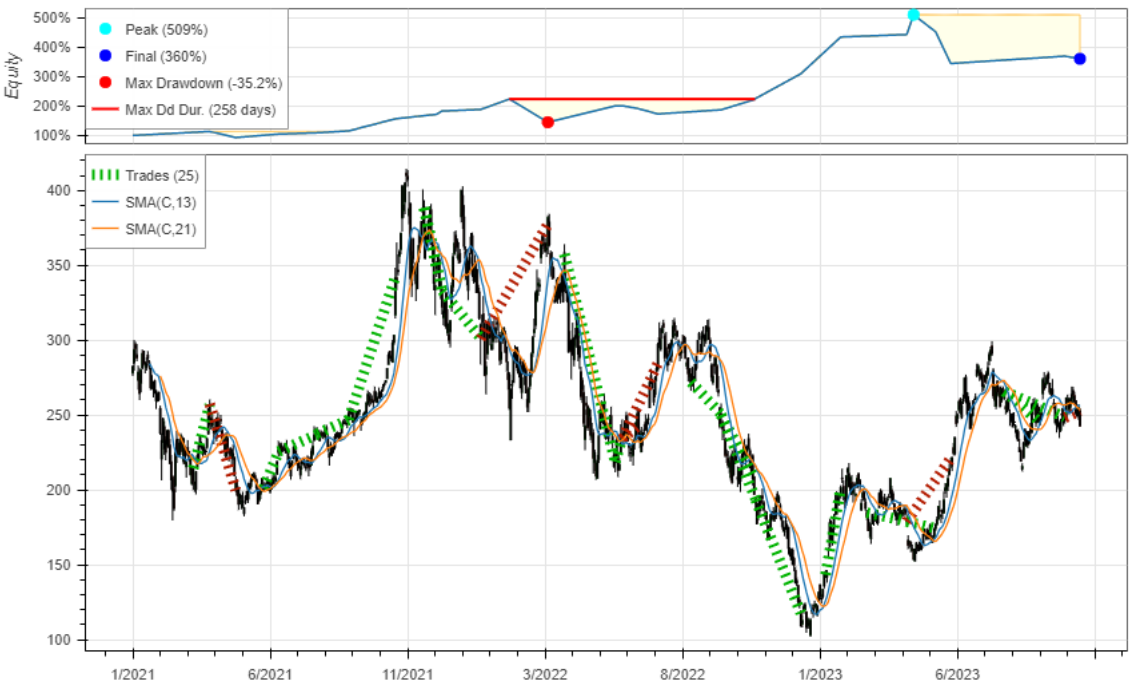
We execute a series of backtests with a 100% position size. This means that we allocate all available capital to each trade, maximizing the investment in each transaction. The run_backtests() function is called without specifying a position_size argument, which defaults to 100%.

Each backtest assesses the performance of the strategy under different leverage factors, calculates and visualizes the outcomes, helping users gain insights into how varying leverage levels impact the strategy's returns.

We have merged the results of these simulations together to create a comprehensive DataFrame for further analysis. This combined DataFrame includes the results of running the backtests with varying leverage factors, ranging from 1x to 100x.

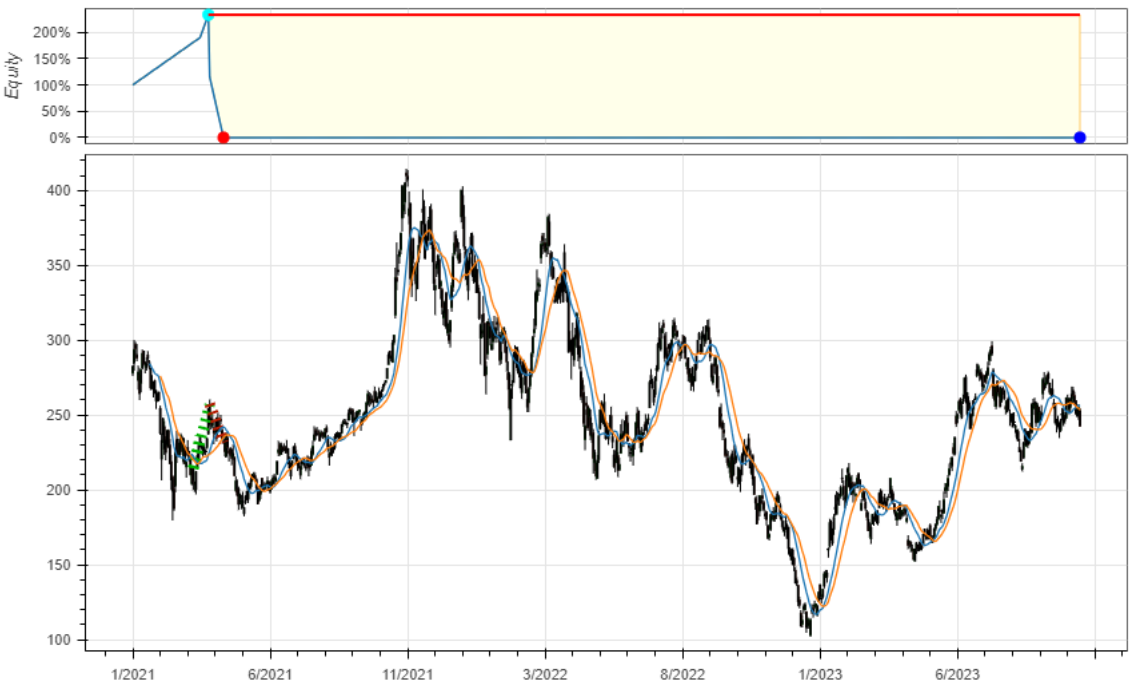
In [7]:
```
# backtests with 100% position size
stat_100 = run_backtests().iloc[:-3]
stat_100
```

```
Strategy: TSLA SmaCross(size=0.9999), Position size: 99.99%
```



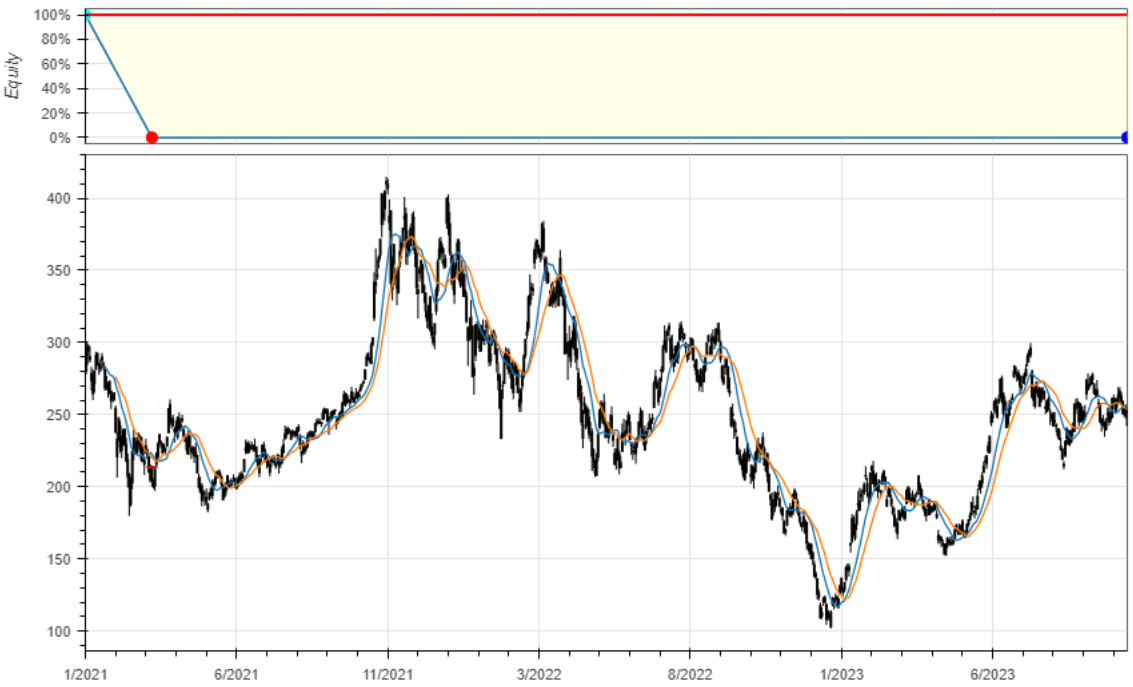
Strategy: TSLA SmaCross(leverage=5,size=0.9999)

Strategy: TSLA SmaCross(leverage=10,size=0.9999)



Strategy: TSLA SmaCross(leverage=20,size=0.9999)

Strategy: TSLA SmaCross(leverage=50,size=0.9999)



Strategy: TSLA SmaCross(leverage=100,size=0.9999)

Out[7]:

	Spot (no leverage)	5x leverage	10x leverage	20x leverage	50x leverage	100x leverage
Start	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00
End	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00
Duration	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00
Exposure Time [%]	72.463768	4.637681	3.333333	0.144928	0.144928	0.144928
Equity Final [$]	35989.702091	0.0	0.0	0.0	0.0	0.0
Equity Peak [$]	50909.132191	18108.437994	23340.748486	10000.0	10000.0	10000.0
Return [%]	259.897021	-100.0	-100.0	-100.0	-100.0	-100.0
Buy & Hold Return [%]	-14.008313	-14.008313	-14.008313	-14.008313	-14.008313	-14.008313
Return (Ann.) [%]	59.634211	0.0	0.0	0.0	0.0	0.0
Volatility (Ann.) [%]	72.980219	9181.170577	1502002.473041	1694.276759	1694.276759	1694.276759
Sharpe Ratio	0.817128	0.0	0.0	0.0	0.0	0.0
Sortino Ratio	2.108258	0.0	0.0	0.0	0.0	0.0
Calmar Ratio	1.694087	0.0	0.0	0.0	0.0	0.0
Max. Drawdown [%]	-35.20139	-100.0	-100.0	-100.0	-100.0	-100.0
Avg. Drawdown [%]	-9.088478	-34.267634	-40.750614	-100.0	-100.0	-100.0
Max. Drawdown Duration	258 days 00:00:00	918 days 00:00:00	918 days 00:00:00	937 days 00:00:00	937 days 00:00:00	937 days 00:00:00
Avg. Drawdown Duration	37 days 00:00:00	234 days 00:00:00	234 days 00:00:00	937 days 00:00:00	937 days 00:00:00	937 days 00:00:00
# Trades	25	3	3	1	1	1
Win Rate [%]	76.0	66.666667	66.666667	0.0	0.0	0.0
Best Trade [%]	40.453359	19.831378	19.831378	-5.40823	-2.421122	-1.425419
Worst Trade [%]	-25.442093	-22.402498	-10.079932	-5.40823	-2.421122	-1.425419
Avg. Trade [%]	6.631263	0.767297	5.841481	-5.40823	-2.421122	-1.425419
Max. Trade Duration	70 days 00:00:00	27 days 00:00:00	19 days 00:00:00	0 days 00:00:00	0 days 00:00:00	0 days 00:00:00
Avg. Trade Duration	34 days 00:00:00	19 days 00:00:00	15 days 00:00:00	0 days 00:00:00	0 days 00:00:00	0 days 00:00:00
Profit Factor	3.102555	1.333281	2.963198	0.0	0.0	0.0
Expectancy [%]	8.285514	2.488778	6.596301	-5.40823	-2.421122	-1.425419
SQN	1.236367	-0.476352	-0.328493	NaN	NaN	NaN

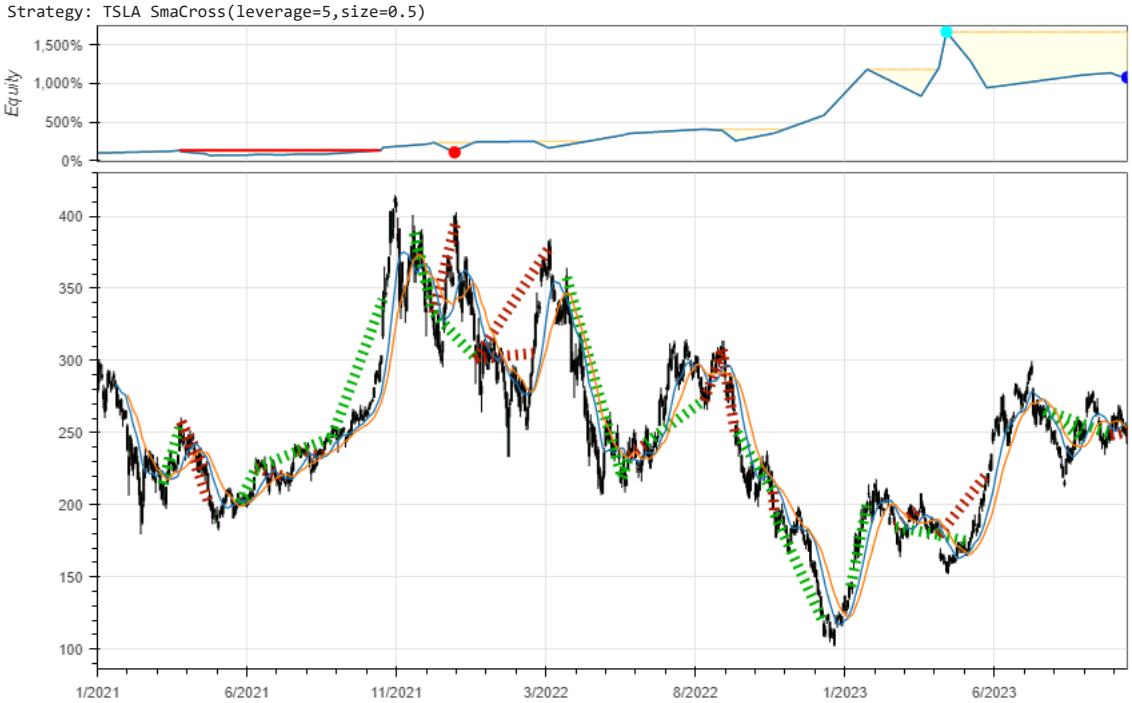
Simulation 2: Running backtests with 50% position size (combined results)

This code segment performs the same series of backtests with a 50% position size. Unlike the previous backtests with a full position size, where 100% of available capital was allocated to each trade, this time only 50% of the capital is utilized in each transaction. The results are merged together to create a comprehensive Dataframe.

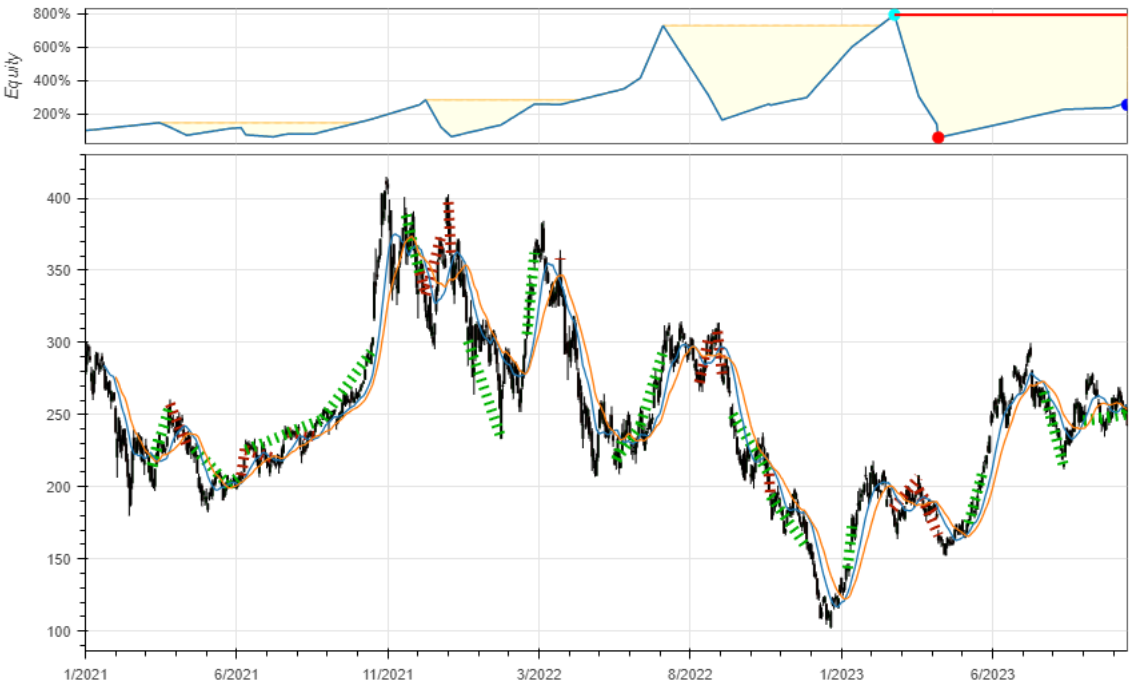
In [8]:
```
# backtests with 50% position size
stat_50 = run_backtests(position_size=0.5).iloc[:-3]
stat_50
```

```
Strategy: TSLA SmaCross(size=0.5), Position size: 50.0%
```

Strategy: TSLA SmaCross(leverage=5,size=0.5)



Strategy: TSLA SmaCross(leverage=10,size=0.5)



Strategy: TSLA SmaCross(leverage=20,size=0.5)

Strategy: TSLA SmaCross(leverage=50,size=0.5)

Strategy: TSLA SmaCross(leverage=100,size=0.5)

Out[8]:

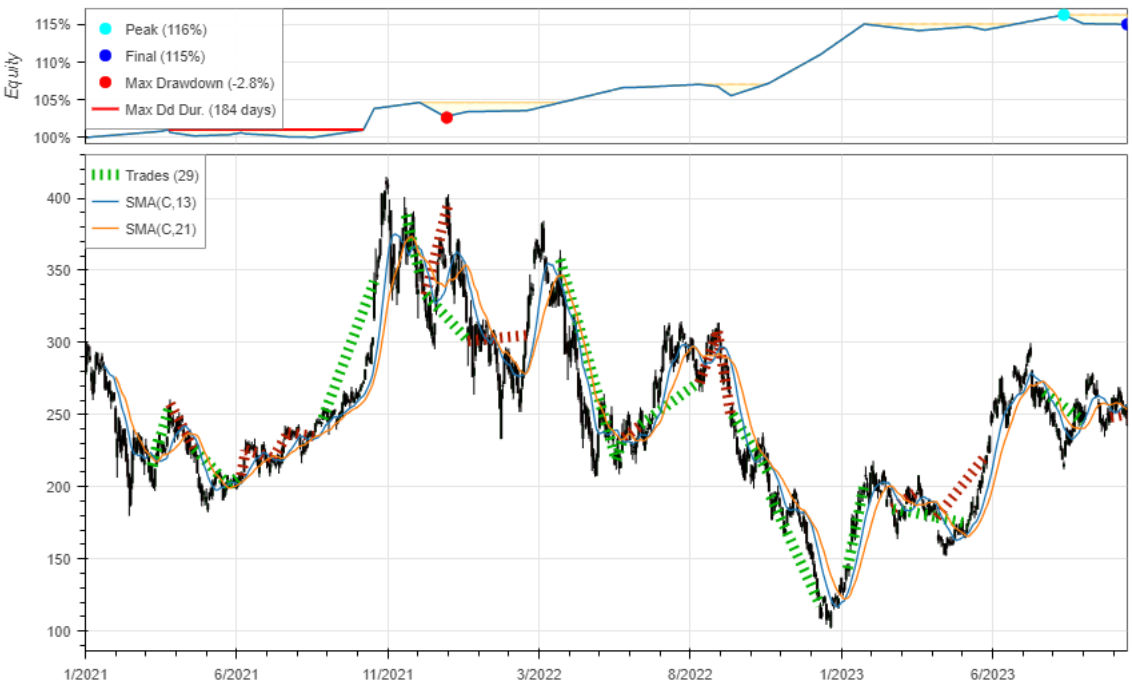
	Spot (no leverage)	5x leverage	10x leverage	20x leverage	50x leverage	100x leverage
Start	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00
End	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00
Duration	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00
Exposure Time [%]	72.463768	75.797101	55.217391	22.318841	9.565217	0.434783
Equity Final [$]	22257.262945	107702.980544	25478.493075	111.169685	46.895205	0.0
Equity Peak [$]	24337.59842	166503.524913	79514.157293	10000.0	10000.0	10000.0
Return [%]	122.572629	977.029805	154.784931	-98.888303	-99.531048	-100.0
Buy & Hold Return [%]	-14.008313	-14.008313	-14.008313	-14.008313	-14.008313	-14.008313
Return (Ann.) [%]	33.937667	138.225099	40.715427	-80.664285	-85.89232	0.0
Volatility (Ann.) [%]	26.721598	294.694363	486.422808	228.734378	448.804086	2707.517741
Sharpe Ratio	1.270046	0.469046	0.083704	0.0	0.0	0.0
Sortino Ratio	2.756281	2.247139	0.399597	0.0	0.0	0.0
Calmar Ratio	2.450848	2.305278	0.439415	0.0	0.0	0.0
Max. Drawdown [%]	-13.847314	-59.960269	-92.658211	-99.827276	-99.844278	-100.0
Avg. Drawdown [%]	-3.710279	-19.402632	-27.182523	-99.827276	-99.844278	-100.0
Max. Drawdown Duration	175 days 00:00:00	192 days 00:00:00	223 days 00:00:00	937 days 00:00:00	937 days 00:00:00	937 days 00:00:00
Avg. Drawdown Duration	29 days 00:00:00	39 days 00:00:00	51 days 00:00:00	937 days 00:00:00	937 days 00:00:00	937 days 00:00:00
# Trades	35	35	35	34	35	3
Win Rate [%]	60.0	57.142857	60.0	47.058824	42.857143	0.0
Best Trade [%]	40.453359	40.453359	22.433178	12.84893	7.033502	-0.200401
Worst Trade [%]	-25.442093	-25.442093	-16.043982	-8.384775	-3.43361	-2.271901
Avg. Trade [%]	3.284945	2.658723	3.438759	1.14927	0.339473	-1.302899
Max. Trade Duration	70 days 00:00:00	70 days 00:00:00	44 days 00:00:00	27 days 00:00:00	12 days 00:00:00	0 days 00:00:00
Avg. Trade Duration	31 days 00:00:00	30 days 00:00:00	19 days 00:00:00	6 days 00:00:00	2 days 00:00:00	0 days 00:00:00
Profit Factor	2.061161	1.880774	2.242926	1.531733	1.400169	0.0
Expectancy [%]	4.695169	4.029297	4.13354	1.410446	0.374919	-1.29924
SQN	2.036167	1.238224	0.182523	-1.239978	-1.440808	-1.732975

Simulation 3: Running backtests with custom position size (combined results)

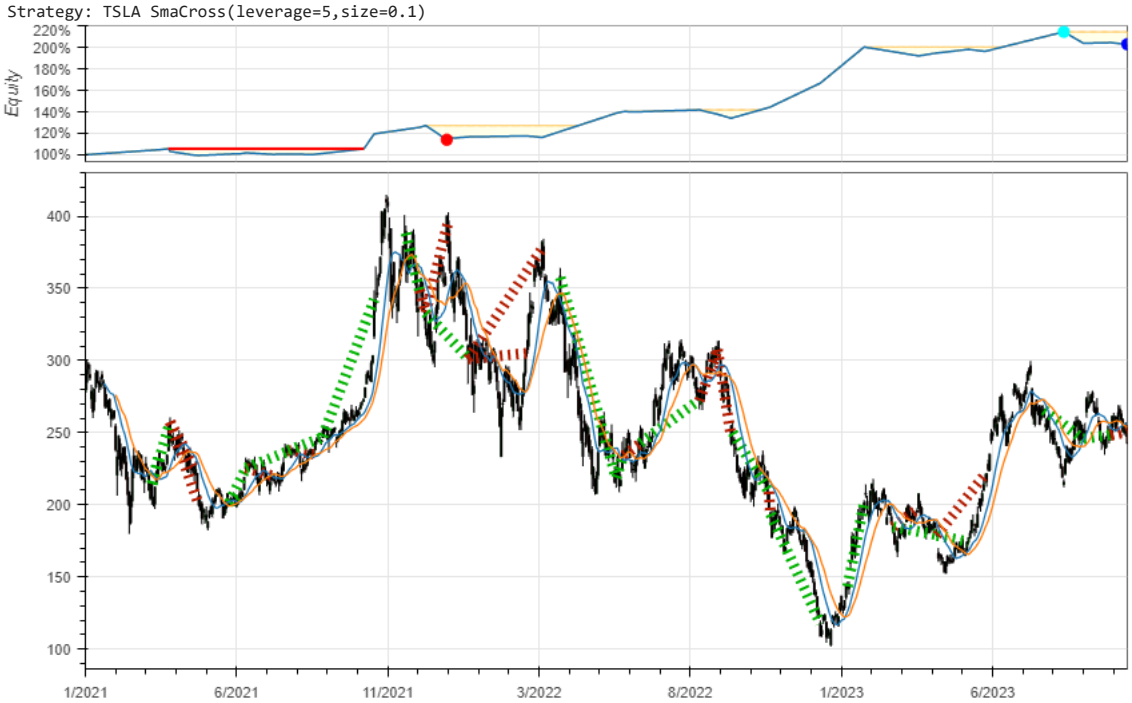
In the last set of simulations, we utilize a user-defined custom position size (10% by default). The backtests produce results for further in-depth analysis (Part Three), facilitating risk assessment and the evaluation of leverage's impact on the portfolio.

In [9]:
```
# backtests with input position size (default 10%)
stat_input = run_backtests(position_size=position_size_pct / 100).iloc[:-3]
stat_input
```

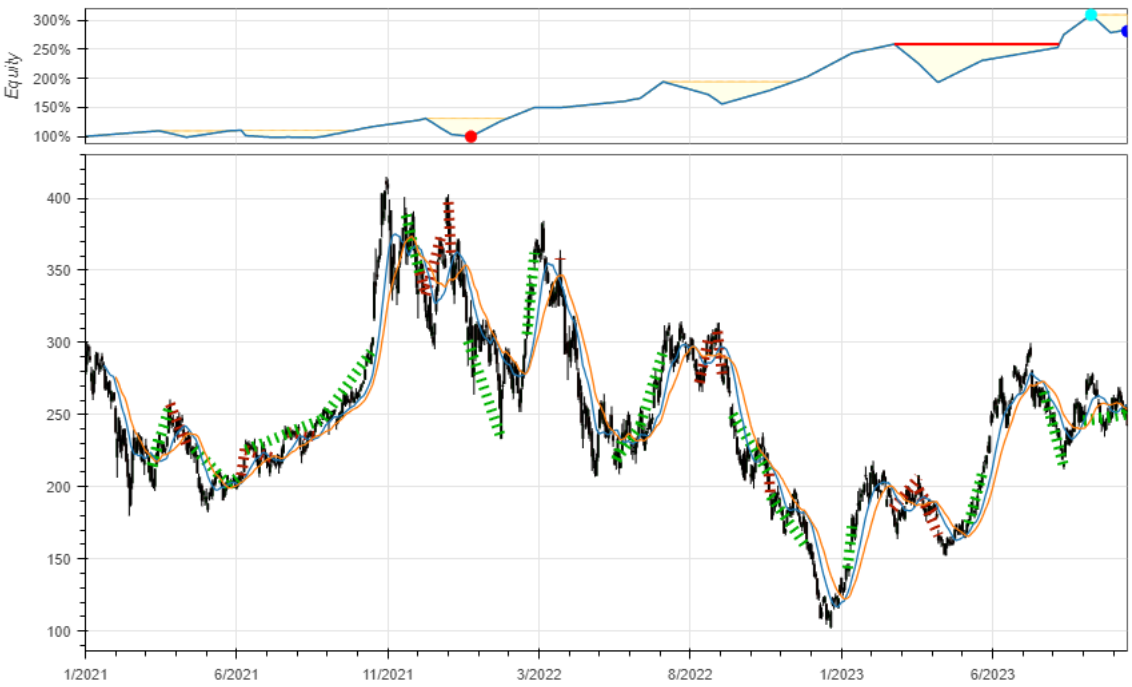
```
Strategy: TSLA SmaCross(size=0.1), Position size: 10.0%
```



Strategy: TSLA SmaCross(leverage=5,size=0.1)



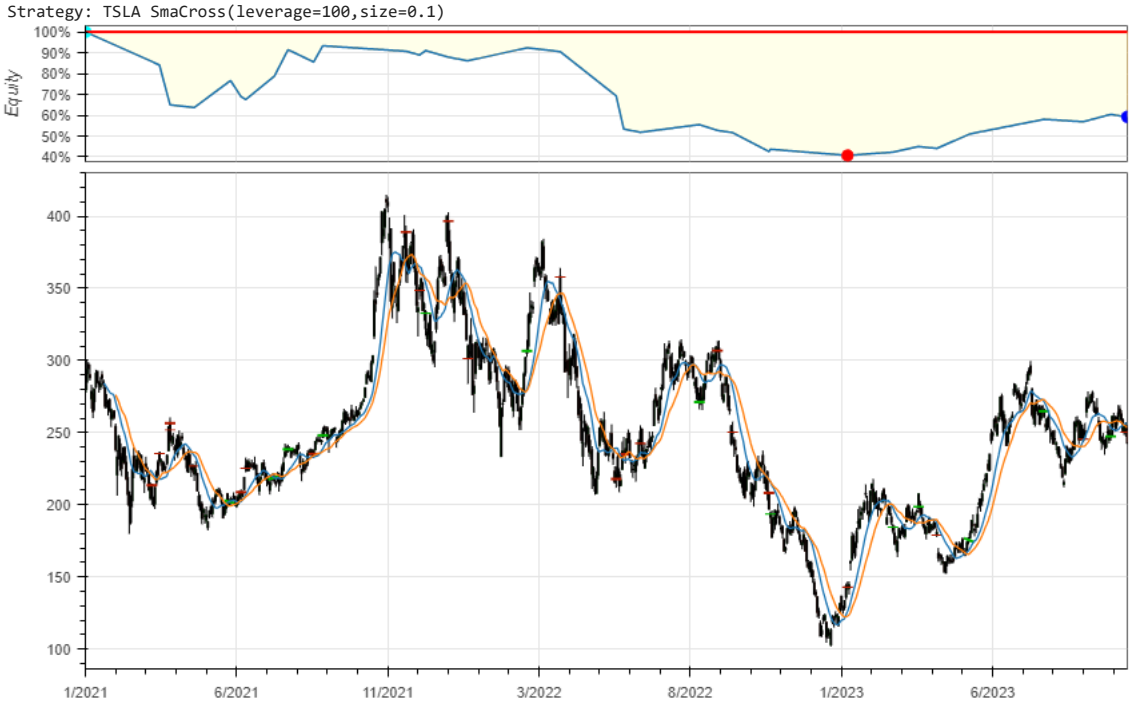
Strategy: TSLA SmaCross(leverage=10,size=0.1)



Strategy: TSLA SmaCross(leverage=20,size=0.1)

Strategy: TSLA SmaCross(leverage=50,size=0.1)

Strategy: TSLA SmaCross(leverage=100,size=0.1)



Out[9]:

	Spot (no leverage)	5x leverage	10x leverage	20x leverage	50x leverage	100x leverage
Start	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00	2021-01-22 00:00:00
End	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00	2023-10-18 00:00:00
Duration	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00	999 days 00:00:00
Exposure Time [%]	73.333333	75.797101	55.217391	22.608696	9.565217	5.507246
Equity Final [$]	11502.811681	20296.686738	28116.422644	17445.537098	14242.417061	5920.957229
Equity Peak [$]	11629.08059	21443.818717	30937.626684	19948.66675	18386.06846	10000.0
Return [%]	15.028117	102.966867	181.164226	74.455371	42.424171	-40.790428
Buy & Hold Return [%]	-14.008313	-14.008313	-14.008313	-14.008313	-14.008313	-14.008313
Return (Ann.) [%]	5.246262	29.50216	45.870685	22.537023	13.786683	-17.420248
Volatility (Ann.) [%]	3.720772	24.135499	49.691918	61.822941	63.924554	34.703292
Sharpe Ratio	1.409993	1.222355	0.923102	0.364541	0.215671	0.0
Sortino Ratio	2.357209	2.575692	2.267432	0.775577	0.438312	0.0
Calmar Ratio	1.848362	2.274489	1.640582	0.489066	0.40374	0.0
Max. Drawdown [%]	-2.838331	-12.970893	-27.960004	-46.081804	-34.147396	-59.300432
Avg. Drawdown [%]	-0.65902	-3.119228	-6.909281	-15.726803	-21.818439	-59.300432
Max. Drawdown Duration	184 days 00:00:00	185 days 00:00:00	159 days 00:00:00	352 days 00:00:00	306 days 00:00:00	937 days 00:00:00
Avg. Drawdown Duration	29 days 00:00:00	28 days 00:00:00	30 days 00:00:00	81 days 00:00:00	123 days 00:00:00	937 days 00:00:00
# Trades	29	35	35	35	35	35
Win Rate [%]	55.172414	54.285714	60.0	48.571429	42.857143	37.142857
Best Trade [%]	40.453359	40.453359	22.433178	12.84893	7.033502	2.036462
Worst Trade [%]	-23.738258	-25.442093	-16.043982	-8.384775	-3.43361	-2.351805
Avg. Trade [%]	3.873807	1.997991	3.438759	1.349473	0.339473	-0.094262
Max. Trade Duration	56 days 00:00:00	70 days 00:00:00	44 days 00:00:00	27 days 00:00:00	12 days 00:00:00	1 days 00:00:00
Avg. Trade Duration	28 days 00:00:00	29 days 00:00:00	19 days 00:00:00	6 days 00:00:00	2 days 00:00:00	1 days 00:00:00
Profit Factor	2.268086	1.714101	2.242926	1.624844	1.400169	0.804948
Expectancy [%]	5.226979	3.354952	4.13354	1.610072	0.374919	-0.088454
SQN	1.981093	1.976449	1.717467	0.853527	0.515139	-0.866637

Part Three: Analyzing results

In this phase of the process, we are examining the aggregated outcomes from the three sets of simulations. Utilizing important metrics obtained, we aim to address our initial inquiries through various visualization methods.

Returns Based on Leverage and Position Size

In this analysis, we explore how leverage and position size impact investment returns.

Return Fluctuations: The heatmap provides a visual representation of how returns fluctuate across a spectrum of leverage levels, ranging from 1x to 100x, with position sizes set at 100%, 50%, and 10%.

Impact of Leverage: Higher leverage, especially at 50x and 100x, leads to volatile returns. With a 100% position size, returns consistently drop to -100%, indicating substantial losses. Frequent liquidations can drain profit quickly with higher leverage levels due to decreased price movement range.

Position Size Matters: Position size is crucial. At any leverage level with 100% position size, a single loss wipes out the entire portfolio. A 50% position size results in a wider range of returns, including gains and losses. A 10% position size offers more stability.

Spot trading is generally safer: Opting for spot trading is generally a safer choice, providing a more conservative approach to investments.

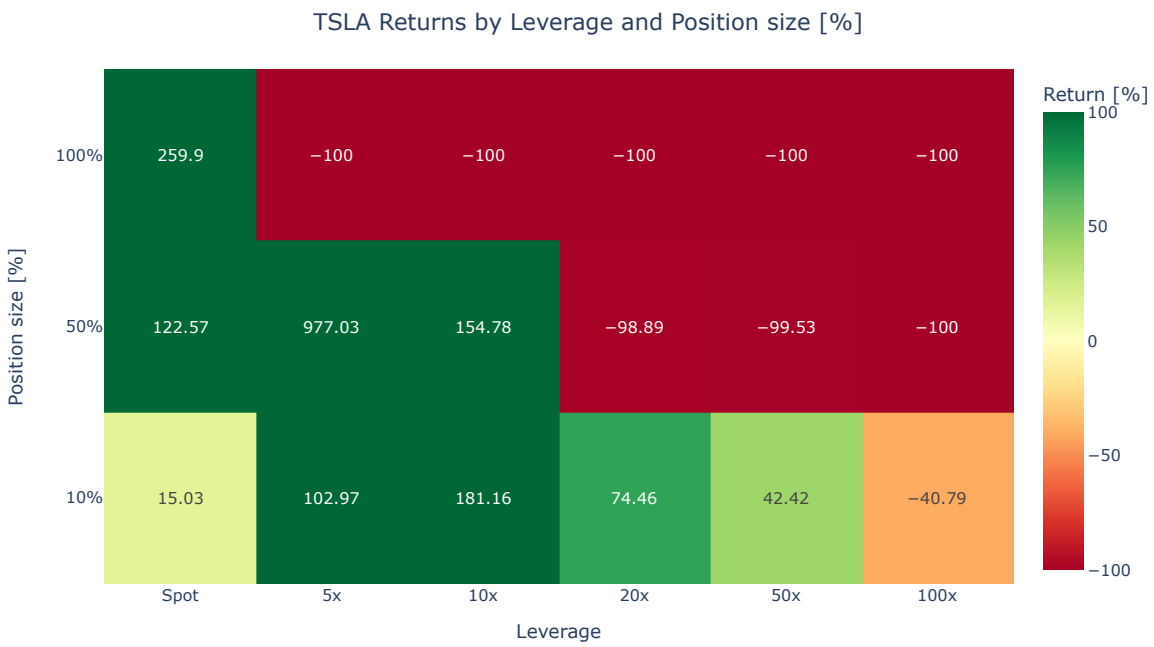
This analysis emphasizes the need to carefully consider leverage and position size. While higher leverage can boost returns, it also increases the risk of significant losses. Therefore, it's important to choose position sizes that balance risk and reward in your investment strategy.

Findings:

- While higher leverage can boost returns, it also increases the risk of significant losses.
- Frequent liquidations can drain profit quickly with higher leverage levels due to decreased price movement range.

- Smaller position sizes (e.g. 10%) provide a more balanced risk-reward profile with leverage.
- Opting for spot trading offers a safer, more stable investment approach, reducing the risk of substantial losses.

In [10]:
```python
# Merge "Return [%]" from backtesting results and create a return table
return_table = (
    pd.concat(
        [
            stat_100.loc["Return [%]"],
            stat_50.loc["Return [%]"],
            stat_input.loc["Return [%]"],
        ],
        axis=1,
        keys=["100%", "50%", f"{position_size_pct}%"],
    )
    .T.apply(pd.to_numeric)
    .round(2)
)

# Create a heatmap of returns using Plotly Express
return_heatmap = px.imshow(
    return_table,
    x=leverage_texts,
    text_auto=True,
    aspect="auto",
    color_continuous_scale="RdYlGn",
    range_color=[-100, 100],
)

# Update the layout of the heatmap
return_heatmap.update_layout(
    title=f"{ticker} Returns by Leverage and Position size [%]",
    title_x=0.5,
    xaxis_title="Leverage",
    yaxis_title="Position size [%]",
)

# Update color axes for the colorbar
return_heatmap.update_coloraxes(colorbar_title="Return [%]")

# Show the heatmap and the table
return_heatmap.show()
return_table
```



Out[10]:

	Spot (no leverage)	5x leverage	10x leverage	20x leverage	50x leverage	100x leverage
100%	259.90	-100.00	-100.00	-100.00	-100.00	-100.00
50%	122.57	977.03	154.78	-98.89	-99.53	-100.00
10%	15.03	102.97	181.16	74.46	42.42	-40.79

Comparison of Win Rate and Profit Factor across leverage levels

This analysis underscores the impact of leverage on profitability and the trade-off with win rates.

The use of Win Rate and Profit Factor as metrics provides a clear picture of risk-adjusted returns. Combining these metrics is a common practice in trading analysis. Win rate measures the proportion of successful trades in a trading strategy, while Profit Factor considers both profits and losses, offering a holistic view of a trading strategy's effectiveness.

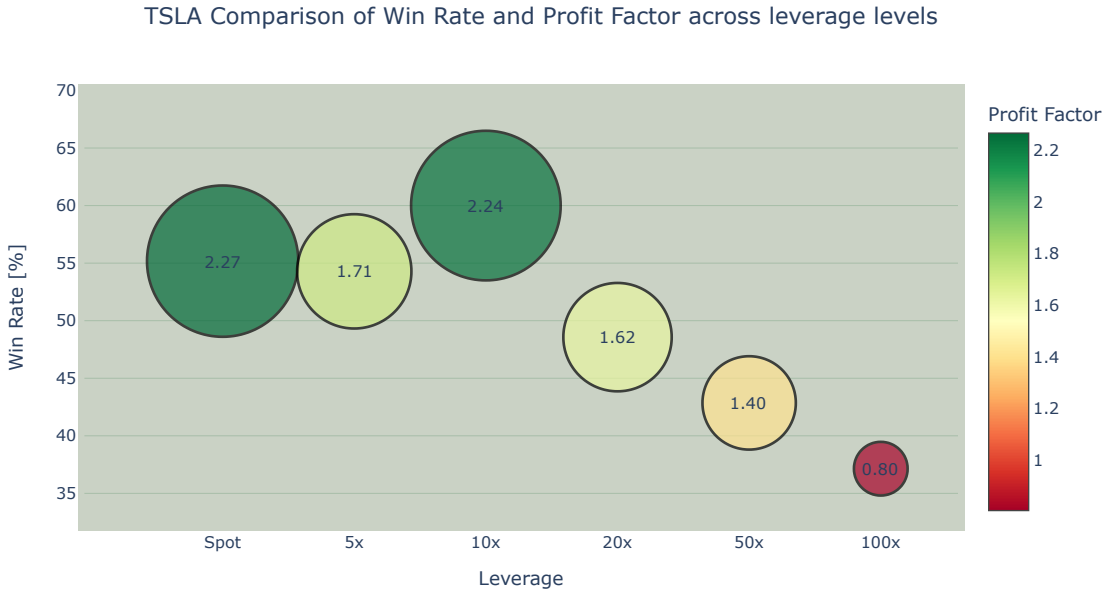
Initially we observe a general improvement in both metrics as leverage increases. This suggests that up to a point of leverage level (10x in this case), the returns tend to be more favorable relative to the risk taken.

In essence, leverage can enhance profit potential... to a degree. Higher leverage levels show a decreasing tendency in both Win Rate and Profit Factor. This suggests that with increased leverage, there may be fewer winning trades and less returns on investment.

Findings:

- Leverage impacts profitability and the balance with win rates. It enhances profit potential up to a point, but excessive leverage can reduce winning trades and returns.
- Balancing risk and reward is crucial in investing. Leverage can amplify profits but needs careful management to avoid diminishing returns and increased risk.

In [11]:
```
# Extract win rates and profit factors from backtesting results
winrates = stat_input.loc["Win Rate [%]"]
pfactors = stat_input.loc["Profit Factor"]

# Convert profit factors to numeric and format them to two decimal places
pfactors_numeric = pd.to_numeric(pfactors)
pfactors_formatted = [f"{x:.2f}" for x in pfactors_numeric]

# Create a Plotly Figure for performance comparison
fig_perf = go.Figure(
    data=[
        go.Scatter(
            x=leverage_texts,
            y=winrates,
            mode="markers",
            marker=dict(
                size=pd.to_numeric(pfactors),
                sizemode="diameter",
                sizeref=0.02,
                color=pfactors,
                showscale=True,
                colorscale="RdYlGn",
                colorbar=dict(
                    title="Profit\nFactor",
                ),
                line=dict(color="black", width=2),
            ),
        )
    ]
)

# Add annotations to the markers on the figure
for i in range(len(leverage_texts)):
    fig_perf.add_annotation(
        x=leverage_texts[i],
        y=winrates[i],
        text=str(pfactors_formatted[i]),
        showarrow=False,
        font=dict(size=12),
    )

# Update the layout of the figure
fig_perf.update_layout(
    title=f"{ticker} Comparison of Win Rate and Profit Factor across leverage levels",
    title_x=0.5,
    yaxis=dict(title="Win Rate [%]", titlefont_size=14, tickfont_size=12),
    xaxis=dict(title="Leverage", titlefont_size=14, tickfont_size=12),
    margin=dict(l=100, r=50, b=100, t=100, pad=5),
    plot_bgcolor="#cad2c5",
    yaxis_gridcolor="#84a98c",
    xaxis_showgrid=False,
)

# Show the figure
fig_perf.show()
```



Risk Assessment: Return on Investment (ROI) vs Max Drawdowns

On this chart we're looking at two things: how much you can make (Return on Investment) and how much you can lose (Max Drawdown).

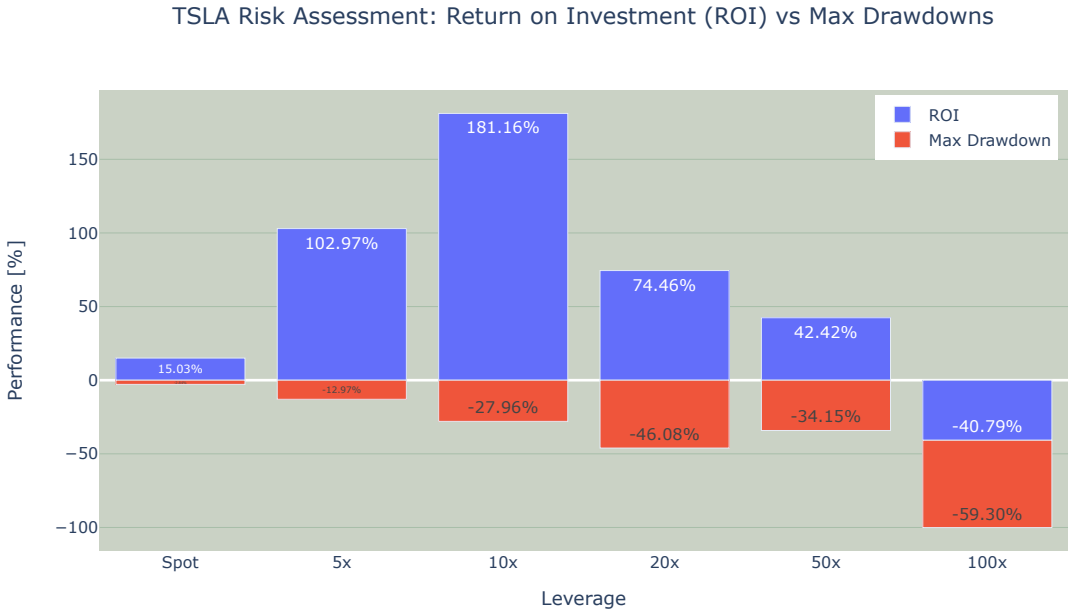
Examining the ROI bars, we notice that 10x leverage demonstrates the highest return, showcasing the potential for amplified returns through leverage. However, it's crucial to acknowledge that excessive leverage doesn't guarantee equivalent returns. This becomes evident in the subsequent leverage levels, ultimately ending in substantial losses for the 100x scenario.

Simultaneously, the Max Drawdown bars shed light on the risks associated with leveraged investment strategies. The chart vividly illustrates that higher leverage is correlated with a significantly elevated risk of drawdowns.

In conclusion, using more leverage can amplify gains, but it also increases the risk of substantial losses, as seen in the larger Max Drawdown values. This underscores the need for careful considerations before investing. Leverage is a potent tool, but it requires a full grasp of its potential risks and rewards.

Finding: Higher leverage usually comes with higher risk. As leverage increases, the chances of losing money rise, especially at higher levels of leverage.

In [12]:
```
# Extract Return [%] and Max. Drawdown [%] from backtesting results
returns = stat_input.loc["Return [%]"]
dds = stat_input.loc["Max. Drawdown [%]"]

# Create a Plotly Figure for the risk assessment comparison
fig_rr = go.Figure()

# Add two Bar traces for ROI and Max Drawdown
fig_rr.add_trace(go.Bar(x=leverage_texts, y=returns, name="ROI"))
fig_rr.add_trace(go.Bar(x=leverage_texts, y=dds, name="Max Drawdown"))

# Set the text template and position for the traces
fig_rr.update_traces(texttemplate="%{y:.2f}%", textposition="inside")

# Update the layout of the figure
fig_rr.update_layout(
    title=f"{ticker} Risk Assessment: Return on Investment (ROI) vs Max Drawdowns",
    title_x=0.5,
    yaxis=dict(title="Performance [%]", titlefont_size=14, tickfont_size=12),
    xaxis=dict(title="Leverage", titlefont_size=14, tickfont_size=12),
    barmode="relative",
    plot_bgcolor="#cad2c5",
    yaxis_gridcolor="#84a98c",
    legend=dict(x=0.99, y=0.99, xanchor="right", yanchor="top"),
)

# Show the figure
fig_rr.show()
```



Volatility and Sharpe Ratio Across Leverage Levels

This chart examines two important aspects of trading:

- Volatility: Measures how much returns fluctuate. Higher values mean a bumpier ride. (Left y-axis)
- Sharpe Ratio: Gauges risk-adjusted returns. Higher values suggest better risk-adjusted performance. (Right y-axis)

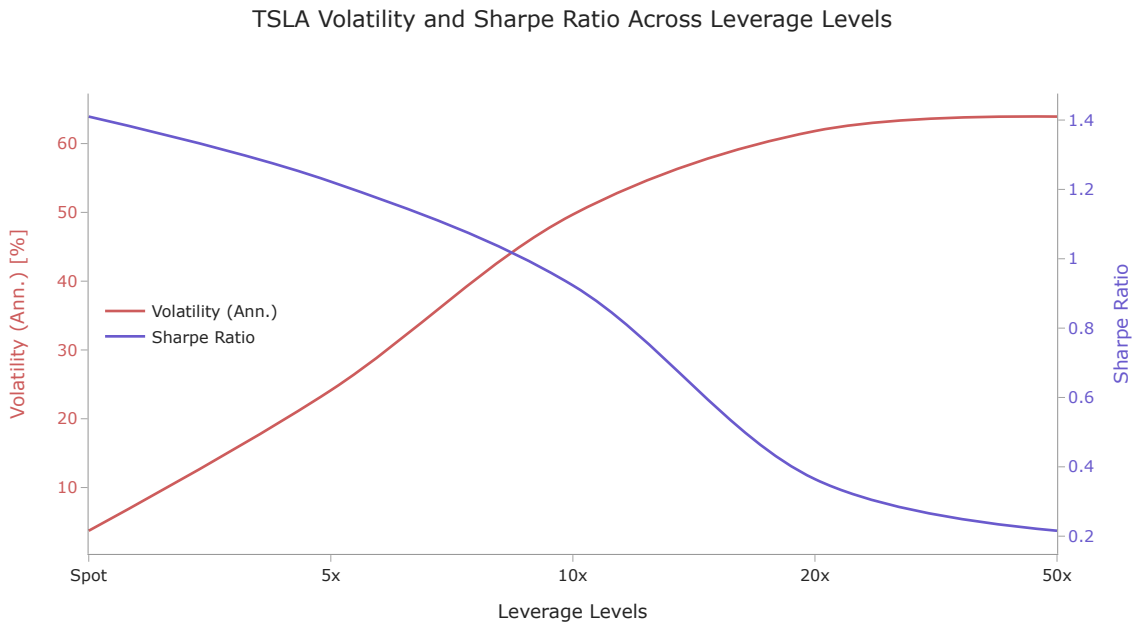
As leverage increases, volatility generally rises, indicating more price swings. Meanwhile the Sharpe Ratio decreases, meaning decreased risk-adjusted performance. This suggests that more leverage doesn't always result in better risk-adjusted returns.

Finding: Higher leverage can mean a more volatile ride, but it doesn't guarantee better risk-adjusted returns. Finding the right balance between risk and reward is key.

Disclaimer: Please note that the indicators are on different scales; therefore, we make no assumptions apart from their lack of correlation. 100x leverage level was removed, due to contaminated results.

In [13]:
```
# Extract Volatility (Ann.) [%] and Sharpe Ratios from backtesting results
volatilities = stat_input.loc["Volatility (Ann.) [%]"]
sharpe_ratios = stat_input.loc["Sharpe Ratio"]

# Create a Plotly Figure for the risk assessment comparison
fig_risk = go.Figure()

# Add a line chart for Volatility
fig_risk.add_trace(
    go.Scatter(
        x=leverage_texts[:-1],
        y=volatilities[:-1],
        mode="lines",
        name="Volatility (Ann.)",
        line=dict(color="indianred"),
        yaxis="y1",
        line_shape="spline",
    )
)

# Add a line chart for Sharpe Ratio on the secondary y-axis
fig_risk.add_trace(
    go.Scatter(
        x=leverage_texts[:-1],
        y=sharpe_ratios[:-1],
        mode="lines",
        name="Sharpe Ratio",
        line=dict(color="slateblue"),
        yaxis="y2",
        line_shape="spline",
    )
)

# Update the layout to create a secondary y-axis and apply the 'simple_white' theme
fig_risk.update_layout(
    title_text=f"{ticker} Volatility and Sharpe Ratio Across Leverage Levels",
    title_x=0.5,
    xaxis=dict(title="Leverage Levels"),
    yaxis=dict(
```

```
        title="Volatility (Ann.) [%]",
        titlefont=dict(color="indianred"),
        tickfont=dict(color="indianred"),
    ),
    yaxis2=dict(
        title="Sharpe Ratio",
        titlefont=dict(color="slateblue"),
        tickfont=dict(color="slateblue"),
        overlaying="y",
        side="right",
    ),
    template="simple_white",
    legend=dict(x=0.01, y=0.5, xanchor="left", yanchor="middle"),
)

# Show the figure
fig_risk.show()
```



Final Words

It is crucial to clarify that the **Leverage impact assessment model** presented in this work is not designed for making direct financial investment decisions. Its primary purpose is to provide investors with valuable insights into the role of leverage in their investment strategies. When the model predicts outcomes, it is highly advisable to complement these findings with thorough financial analysis and consultation with qualified experts before making any investment decisions.

This model is intended to serve as a tool to assist investors in understanding how leverage influences investment returns and risks. While it offers valuable insights, the decision to apply these findings to real-world investments should be made in a responsible and informed manner. It is essential to acknowledge that investments carry inherent risks, and leveraging can amplify both potential gains and losses.

Investors are encouraged to exercise caution, conduct further research, and consider their unique financial circumstances and goals. The model aims to empower investors with knowledge, but prudent decision-making should always involve a comprehensive evaluation of individual factors and professional guidance when necessary.

In summary, this model serves as a valuable resource for understanding the implications of leverage in investment decisions, but it is not a substitute for careful consideration and consultation with financial experts. It is a tool to facilitate informed decision-making, ultimately empowering investors to navigate the financial markets with greater insight and understanding.