```
// This source code is subject to the terms of the Mozilla Public License 2.0 at
https://mozilla.org/MPL/2.0/
//
// Please, visit the main script that strategy is based in :) Machine Learning: Lorentzian
Classification/<<<< ============================================
// Indicator developed by ©jdehorty
// Strategy devoleped by ©StrategiesForEVeryone

// @version=5
strategy('Machine Learning: Lorentzian Classification', 'Lorentzian Classification Strategy',
true, precision=2, initial_capital = 100000, process_orders_on_close = true,
calc_on_every_tick = true, commission_value = 0.03)

import jdehorty/MLExtensions/2 as ml
import jdehorty/KernelFunctions/2 as kernels

initial_message = input.bool(defval = true, title = "Show initial message", group = "Initial
message")
if barstate.islast and initial_message
    label.new(bar_index, 0, "Please, read the entire post " + "\nfor a better understanding." +
"\nDo not forget to visit the original" + "\nindicator from ©jdehorty and leave your boost!",
xloc = xloc.bar_index, yloc = yloc.abovebar, textcolor = color.white, color = color.rgb(76, 175,
255, 4))

// ======================
// ==== Custom Types ====
// ======================

// This section uses PineScript's new Type syntax to define important data structures
// used throughout the script.

type Settings
    float source
    int neighborsCount
    int maxBarsBack
    int featureCount
    int colorCompression
    bool showExits
    bool useDynamicExits

type Label
    int long
    int short
    int neutral

type FeatureArrays
    array<float> f1
    array<float> f2
```

```
    array<float> f3
    array<float> f4
    array<float> f5

type FeatureSeries
    float f1
    float f2
    float f3
    float f4
    float f5

type MLModel
    int firstBarIndex
    array<int> trainingLabels
    int loopSize
    float lastDistance
    array<float> distancesArray
    array<int> predictionsArray
    int prediction

type FilterSettings
    bool useVolatilityFilter
    bool useRegimeFilter
    bool useAdxFilter
    float regimeThreshold
    int adxThreshold

type Filter
    bool volatility
    bool regime
    bool adx

// ==========================
// ==== Helper Functions ====
// ==========================

series_from(feature_string, _close, _high, _low, _hlc3, f_paramA, f_paramB) =>
    switch feature_string
        "RSI" => ml.n_rsi(_close, f_paramA, f_paramB)
        "WT" => ml.n_wt(_hlc3, f_paramA, f_paramB)
        "CCI" => ml.n_cci(_close, f_paramA, f_paramB)
        "ADX" => ml.n_adx(_high, _low, _close, f_paramA)

get_lorentzian_distance(int i, int featureCount, FeatureSeries featureSeries, FeatureArrays
featureArrays) =>
    switch featureCount
        5 => math.log(1+math.abs(featureSeries.f1 - array.get(featureArrays.f1, i))) +
            math.log(1+math.abs(featureSeries.f2 - array.get(featureArrays.f2, i))) +
```

```
            math.log(1+math.abs(featureSeries.f3 - array.get(featureArrays.f3, i))) +
            math.log(1+math.abs(featureSeries.f4 - array.get(featureArrays.f4, i))) +
            math.log(1+math.abs(featureSeries.f5 - array.get(featureArrays.f5, i)))
        4 => math.log(1+math.abs(featureSeries.f1 - array.get(featureArrays.f1, i))) +
            math.log(1+math.abs(featureSeries.f2 - array.get(featureArrays.f2, i))) +
            math.log(1+math.abs(featureSeries.f3 - array.get(featureArrays.f3, i))) +
            math.log(1+math.abs(featureSeries.f4 - array.get(featureArrays.f4, i)))
        3 => math.log(1+math.abs(featureSeries.f1 - array.get(featureArrays.f1, i))) +
            math.log(1+math.abs(featureSeries.f2 - array.get(featureArrays.f2, i))) +
            math.log(1+math.abs(featureSeries.f3 - array.get(featureArrays.f3, i)))
        2 => math.log(1+math.abs(featureSeries.f1 - array.get(featureArrays.f1, i))) +
            math.log(1+math.abs(featureSeries.f2 - array.get(featureArrays.f2, i)))

// ================
// ==== Inputs ====
// ================

// Settings Object: General User-Defined Inputs
Settings settings =
 Settings.new(
   input.source(title='Source', defval=close, group="General Settings", tooltip="Source of the
input data"),
   input.int(title='Neighbors Count', defval=8, group="General Settings", minval=1,
maxval=100, step=1, tooltip="Number of neighbors to consider"),
   input.int(title="Max Bars Back", defval=2000, group="General Settings", tooltip = "Use this
to increase or reduce the range of backtesting. If script use much time to load, reduce this"),
   input.int(title="Feature Count", defval=5, group="Feature Engineering", minval=2,
maxval=5, tooltip="Number of features to use for ML predictions."),
   input.int(title="Color Compression", defval=1, group="General Settings", minval=1,
maxval=10, tooltip="Compression factor for adjusting the intensity of the color scale."),
   input.bool(title="Show Default Exits", defval=false, group="General Settings",
tooltip="Default exits occur exactly 4 bars after an entry signal. This corresponds to the
predefined length of a trade during the model's training process.", inline="exits"),
   input.bool(title="Use Dynamic Exits", defval=false, group="General Settings",
tooltip="Dynamic exits attempt to let profits ride by dynamically adjusting the exit threshold
based on kernel regression logic.", inline="exits")
 )

// Trade Stats Settings
// Note: The trade stats section is NOT intended to be used as a replacement for proper
backtesting. It is intended to be used for calibration purposes only.
// showTradeStats = input.bool(false, 'Show Trade Stats', tooltip='Displays the trade stats for
a given configuration. Useful for optimizing the settings in the Feature Engineering section.
This should NOT replace backtesting and should be used for calibration purposes only. Early
Signal Flips represent instances where the model changes signals before 4 bars elapses;
high values can indicate choppy (ranging) market conditions.', group="General Settings")
// useWorstCase = input.bool(false, "Use Worst Case Estimates", tooltip="Whether to use
the worst case scenario for backtesting. This option can be useful for creating a conservative
```

estimate that is based on close prices only, thus avoiding the effects of intrabar repainting. This option assumes that the user does not enter when the signal first appears and instead waits for the bar to close as confirmation. On larger timeframes, this can mean entering after a large move has already occurred. Leaving this option disabled is generally better for those that use this indicator as a source of confluence and prefer estimates that demonstrate discretionary mid-bar entries. Leaving this option enabled may be more consistent with traditional backtesting results.", group="General Settings")

// Settings object for user-defined settings
FilterSettings filterSettings =
 FilterSettings.new(
   input.bool(title="Use Volatility Filter", defval=true, tooltip="Whether to use the volatility filter.", group="Filters"),
   input.bool(title="Use Regime Filter", defval=true, group="Filters", inline="regime"),
   input.bool(title="Use ADX Filter", defval=false, group="Filters", inline="adx"),
   input.float(title="Threshold", defval=-0.1, minval=-10, maxval=10, step=0.1, tooltip="Whether to use the trend detection filter. Threshold for detecting Trending/Ranging markets.", group="Filters", inline="regime"),
   input.int(title="Threshold", defval=20, minval=0, maxval=100, step=1, tooltip="Whether to use the ADX filter. Threshold for detecting Trending/Ranging markets.", group="Filters", inline="adx")
)

// Filter object for filtering the ML predictions
Filter filter =
 Filter.new(
   ml.filter_volatility(1, 10, filterSettings.useVolatilityFilter),
   ml.regime_filter(ohlc4, filterSettings.regimeThreshold, filterSettings.useRegimeFilter),
   ml.filter_adx(settings.source, 14, filterSettings.adxThreshold, filterSettings.useAdxFilter)
)

// Feature Variables: User-Defined Inputs for calculating Feature Series.
f1_string = input.string(title="Feature 1", options=["RSI", "WT", "CCI", "ADX"], defval="RSI", inline = "01", tooltip="The first feature to use for ML predictions.", group="Feature Engineering")
f1_paramA = input.int(title="Parameter A", tooltip="The primary parameter of feature 1.", defval=14, inline = "02", group="Feature Engineering")
f1_paramB = input.int(title="Parameter B", tooltip="The secondary parameter of feature 2 (if applicable).", defval=1, inline = "02", group="Feature Engineering")
f2_string = input.string(title="Feature 2", options=["RSI", "WT", "CCI", "ADX"], defval="WT", inline = "03", tooltip="The second feature to use for ML predictions.", group="Feature Engineering")
f2_paramA = input.int(title="Parameter A", tooltip="The primary parameter of feature 2.", defval=10, inline = "04", group="Feature Engineering")
f2_paramB = input.int(title="Parameter B", tooltip="The secondary parameter of feature 2 (if applicable).", defval=11, inline = "04", group="Feature Engineering")

```
f3_string = input.string(title="Feature 3", options=["RSI", "WT", "CCI", "ADX"], defval="CCI",
inline = "05", tooltip="The third feature to use for ML predictions.", group="Feature
Engineering")
f3_paramA = input.int(title="Parameter A", tooltip="The primary parameter of feature 3.",
defval=20, inline = "06", group="Feature Engineering")
f3_paramB = input.int(title="Parameter B", tooltip="The secondary parameter of feature 3 (if
applicable).", defval=1, inline = "06", group="Feature Engineering")
f4_string = input.string(title="Feature 4", options=["RSI", "WT", "CCI", "ADX"], defval="ADX",
inline = "07", tooltip="The fourth feature to use for ML predictions.", group="Feature
Engineering")
f4_paramA = input.int(title="Parameter A", tooltip="The primary parameter of feature 4.",
defval=20, inline = "08", group="Feature Engineering")
f4_paramB = input.int(title="Parameter B", tooltip="The secondary parameter of feature 4 (if
applicable).", defval=2, inline = "08", group="Feature Engineering")
f5_string = input.string(title="Feature 5", options=["RSI", "WT", "CCI", "ADX"], defval="RSI",
inline = "09", tooltip="The fifth feature to use for ML predictions.", group="Feature
Engineering")
f5_paramA = input.int(title="Parameter A", tooltip="The primary parameter of feature 5.",
defval=9, inline = "10", group="Feature Engineering")
f5_paramB = input.int(title="Parameter B", tooltip="The secondary parameter of feature 5 (if
applicable).", defval=1, inline = "10", group="Feature Engineering")

// FeatureSeries Object: Calculated Feature Series based on Feature Variables
featureSeries =
 FeatureSeries.new(
   series_from(f1_string, close, high, low, hlc3, f1_paramA, f1_paramB), // f1
   series_from(f2_string, close, high, low, hlc3, f2_paramA, f2_paramB), // f2
   series_from(f3_string, close, high, low, hlc3, f3_paramA, f3_paramB), // f3
   series_from(f4_string, close, high, low, hlc3, f4_paramA, f4_paramB), // f4
   series_from(f5_string, close, high, low, hlc3, f5_paramA, f5_paramB)  // f5
 )

// FeatureArrays Variables: Storage of Feature Series as Feature Arrays Optimized for ML
// Note: These arrays cannot be dynamically created within the FeatureArrays Object
Initialization and thus must be set-up in advance.
var f1Array = array.new_float()
var f2Array = array.new_float()
var f3Array = array.new_float()
var f4Array = array.new_float()
var f5Array = array.new_float()
array.push(f1Array, featureSeries.f1)
array.push(f2Array, featureSeries.f2)
array.push(f3Array, featureSeries.f3)
array.push(f4Array, featureSeries.f4)
array.push(f5Array, featureSeries.f5)

// FeatureArrays Object: Storage of the calculated FeatureArrays into a single object
featureArrays =
```

```
 FeatureArrays.new(
  f1Array, // f1
  f2Array, // f2
  f3Array, // f3
  f4Array, // f4
  f5Array  // f5
 )

// Label Object: Used for classifying historical data as training data for the ML Model
Label direction_s =
 Label.new(
   long=1,
   short=-1,
   neutral=0
  )

// Derived from General Settings
maxBarsBackIndex = last_bar_index >= settings.maxBarsBack ? last_bar_index -
settings.maxBarsBack : 0

// EMA Settings
useEmaFilter = input.bool(title="Use EMA Filter", defval=false, group="Filters", inline="ema")
emaPeriod = input.int(title="Period", defval=200, minval=1, step=1, group="Filters",
inline="ema", tooltip="The period of the EMA used for the EMA Filter.")
isEmaUptrend = useEmaFilter ? close > ta.ema(close, emaPeriod) : true
isEmaDowntrend = useEmaFilter ? close < ta.ema(close, emaPeriod) : true
useSmaFilter = input.bool(title="Use SMA Filter", defval=false, group="Filters", inline="sma")
smaPeriod = input.int(title="Period", defval=200, minval=1, step=1, group="Filters",
inline="sma", tooltip="The period of the SMA used for the SMA Filter.")
isSmaUptrend = useSmaFilter ? close > ta.sma(close, smaPeriod) : true
isSmaDowntrend = useSmaFilter ? close < ta.sma(close, smaPeriod) : true

// Nadaraya-Watson Kernel Regression Settings
useKernelFilter = input.bool(true, "Trade with Kernel", group="Kernel Settings",
inline="kernel")
showKernelEstimate = input.bool(false, "Show Kernel Estimate", group="Kernel Settings",
inline="kernel")
useKernelSmoothing = input.bool(false, "Enhance Kernel Smoothing", tooltip="Uses a
crossover based mechanism to smoothen kernel color changes. This often results in less
color transitions overall and may result in more ML entry signals being generated.",
inline='1', group='Kernel Settings')
h = input.int(8, 'Lookback Window', minval=3, tooltip='The number of bars used for the
estimation. This is a sliding value that represents the most recent historical bars.
Recommended range: 3-50', group="Kernel Settings", inline="kernel")
r = input.float(8., 'Relative Weighting', step=0.25, tooltip='Relative weighting of time frames.
As this value approaches zero, the longer time frames will exert more influence on the
estimation. As this value approaches infinity, the behavior of the Rational Quadratic Kernel
```

```
will become identical to the Gaussian kernel. Recommended range: 0.25-25', group="Kernel Settings", inline="kernel")
x = input.int(25, "Regression Level", tooltip='Bar index on which to start regression. Controls how tightly fit the kernel estimate is to the data. Smaller values are a tighter fit. Larger values are a looser fit. Recommended range: 2-25', group="Kernel Settings", inline="kernel")
lag = input.int(2, "Lag", tooltip="Lag for crossover detection. Lower values result in earlier crossovers. Recommended range: 1-2", inline='1', group='Kernel Settings')

// Display Settings
showBarColors = input.bool(false, "Show Bar Colors", tooltip="Whether to show the bar colors.", group="Display Settings")
showBarPredictions = input.bool(defval = false, title = "Show Bar Prediction Values", tooltip = "Will show the ML model's evaluation of each bar as an integer.", group="Display Settings")
useAtrOffset = input.bool(defval = false, title = "Use ATR Offset", tooltip = "Will use the ATR offset instead of the bar prediction offset.", group="Display Settings")
barPredictionsOffset = input.float(0, "Bar Prediction Offset", minval=0, tooltip="The offset of the bar predictions as a percentage from the bar high or close.", group="Display Settings")

// ==================================
// ==== Next Bar Classification ====
// ==================================

// This model specializes specifically in predicting the direction_s of price action over the course of the next 4 bars.
// To avoid complications with the ML model, this value is hardcoded to 4 bars but support for other training lengths may be added in the future.
src = settings.source
y_train_series = src[4] < src[0] ? direction_s.short : src[4] > src[0] ? direction_s.long : direction_s.neutral
var y_train_array = array.new_int(0)

// Variables used for ML Logic
var predictions = array.new_float(0)
var prediction = 0.
var signal = direction_s.neutral
var distances = array.new_float(0)

array.push(y_train_array, y_train_series)

// ==========================
// ====  Core ML Logic  ====
// ==========================

lastDistance = -1.0
size = math.min(settings.maxBarsBack-1, array.size(y_train_array)-1)
sizeLoop = math.min(settings.maxBarsBack-1, size)

if bar_index >= maxBarsBackIndex //{
```

```
    for i = 0 to sizeLoop //{
        d = get_lorentzian_distance(i, settings.featureCount, featureSeries, featureArrays)
        if d >= lastDistance and i%4 //{
            lastDistance := d
            array.push(distances, d)
            array.push(predictions, math.round(array.get(y_train_array, i)))
            if array.size(predictions) > settings.neighborsCount //{
                lastDistance := array.get(distances, math.round(settings.neighborsCount*3/4))
                array.shift(distances)
                array.shift(predictions)
            //}
        //}
    //}
    prediction := array.sum(predictions)
//}

// ============================
// ==== Prediction Filters ====
// ============================

// User Defined Filters: Used for adjusting the frequency of the ML Model's predictions
filter_all = filter.volatility and filter.regime and filter.adx

// Filtered Signal: The model's prediction of future price movement direction_s with
user-defined filters applied
signal := prediction > 0 and filter_all ? direction_s.long : prediction < 0 and filter_all ?
direction_s.short : nz(signal[1])

// Bar-Count Filters: Represents strict filters based on a pre-defined holding period of 4 bars
var int barsHeld = 0
barsHeld := ta.change(signal) ? 0 : barsHeld + 1
isHeldFourBars = barsHeld == 4
isHeldLessThanFourBars = 0 < barsHeld and barsHeld < 4

// Fractal Filters: Derived from relative ============ Appearance ============s of
signals in a given time series fractal/segment with a default length of 4 bars
isDifferentSignalType = ta.change(signal)
isEarlySignalFlip = ta.change(signal) and (ta.change(signal[1]) or ta.change(signal[2]) or
ta.change(signal[3]))
isBuySignal = signal == direction_s.long and isEmaUptrend and isSmaUptrend
isSellSignal = signal == direction_s.short and isEmaDowntrend and isSmaDowntrend
isLastSignalBuy = signal[4] == direction_s.long and isEmaUptrend[4] and isSmaUptrend[4]
isLastSignalSell = signal[4] == direction_s.short and isEmaDowntrend[4] and
isSmaDowntrend[4]
isNewBuySignal = isBuySignal and isDifferentSignalType
isNewSellSignal = isSellSignal and isDifferentSignalType
```

```
// Kernel Regression Filters: Filters based on Nadaraya-Watson Kernel Regression using the Rational Quadratic Kernel
// For more information on this technique refer to my other open source indicator located here:
//
https://www.tradingview.com/script/AWNvbPRM-Nadaraya-Watson-Rational-Quadratic-Kernel-Non-Repainting/
c_green = color.new(#009988, 20)
c_red = color.new(#CC3311, 20)
transparent = color.new(#000000, 100)
yhat1 = kernels.rationalQuadratic(settings.source, h, r, x)
yhat2 = kernels.gaussian(settings.source, h-lag, x)
kernelEstimate = yhat1
// Kernel Rates of Change
bool wasBearishRate = yhat1[2] > yhat1[1]
bool wasBullishRate = yhat1[2] < yhat1[1]
bool isBearishRate = yhat1[1] > yhat1
bool isBullishRate = yhat1[1] < yhat1
isBearishChange = isBearishRate and wasBullishRate
isBullishChange = isBullishRate and wasBearishRate
// Kernel Crossovers
bool isBullishCrossAlert = ta.crossover(yhat2, yhat1)
bool isBearishCrossAlert = ta.crossunder(yhat2, yhat1)
bool isBullishSmooth = yhat2 >= yhat1
bool isBearishSmooth = yhat2 <= yhat1
// Kernel Colors
color colorByCross = isBullishSmooth ? c_green : c_red
color colorByRate = isBullishRate ? c_green : c_red
color plotColor = showKernelEstimate ? (useKernelSmoothing ? colorByCross : colorByRate) : transparent
plot(kernelEstimate, color=plotColor, linewidth=2, title="Kernel Regression Estimate", display = display.all - display.price_scale - display.status_line)
// Alert Variables
bool alertBullish = useKernelSmoothing ? isBullishCrossAlert : isBullishChange
bool alertBearish = useKernelSmoothing ? isBearishCrossAlert : isBearishChange
// Bullish and Bearish Filters based on Kernel
isBullish = useKernelFilter ? (useKernelSmoothing ? isBullishSmooth : isBullishRate) : true
isBearish = useKernelFilter ? (useKernelSmoothing ? isBearishSmooth : isBearishRate) : true

// ===========================
// ==== Entries and Exits ====
// ===========================

// Entry Conditions: Booleans for ML Model Position Entries
startLongTrade = isNewBuySignal and isBullish and isEmaUptrend and isSmaUptrend
startShortTrade = isNewSellSignal and isBearish and isEmaDowntrend and isSmaDowntrend
```

```
// Dynamic Exit Conditions: Booleans for ML Model Position Exits based on Fractal Filters and Kernel Regression Filters
lastSignalWasBullish = ta.barssince(startLongTrade) < ta.barssince(startShortTrade)
lastSignalWasBearish = ta.barssince(startShortTrade) < ta.barssince(startLongTrade)
barsSinceRedEntry = ta.barssince(startShortTrade)
barsSinceRedExit = ta.barssince(alertBullish)
barsSinceGreenEntry = ta.barssince(startLongTrade)
barsSinceGreenExit = ta.barssince(alertBearish)
isValidShortExit = barsSinceRedExit > barsSinceRedEntry
isValidLongExit = barsSinceGreenExit > barsSinceGreenEntry
endLongTradeDynamic = (isBearishChange and isValidLongExit[1])
endShortTradeDynamic = (isBullishChange and isValidShortExit[1])

// Fixed Exit Conditions: Booleans for ML Model Position Exits based on a Bar-Count Filters
endLongTradeStrict = ((isHeldFourBars and isLastSignalBuy) or (isHeldLessThanFourBars and isNewSellSignal and isLastSignalBuy)) and startLongTrade[4]
endShortTradeStrict = ((isHeldFourBars and isLastSignalSell) or (isHeldLessThanFourBars and isNewBuySignal and isLastSignalSell)) and startShortTrade[4]
isDynamicExitValid = not useEmaFilter and not useSmaFilter and not useKernelSmoothing
endLongTrade = settings.useDynamicExits and isDynamicExitValid ? endLongTradeDynamic : endLongTradeStrict
endShortTrade = settings.useDynamicExits and isDynamicExitValid ? endShortTradeDynamic : endShortTradeStrict

// =========================
// ==== Plotting Labels ====
// =========================

// Note: These will not repaint once the most recent bar has fully closed. By default, signals appear over the last closed bar; to override this behavior set offset=0.
//plotshape(startLongTrade ? low : na, 'Buy', shape.labelup, location.belowbar, color=ml.color_green(prediction), size=size.small, offset=0, display = display.all - display.price_scale - display.status_line)
//plotshape(startShortTrade ? high : na, 'Sell', shape.labeldown, location.abovebar, ml.color_red(-prediction), size=size.small, offset=0, display = display.all - display.price_scale - display.status_line)
//plotshape(endLongTrade and settings.showExits ? high : na, 'StopBuy', shape.xcross, location.absolute, color=#3AFF17, size=size.tiny, offset=0, display = display.all - display.price_scale - display.status_line)
//plotshape(endShortTrade and settings.showExits ? low : na, 'StopSell', shape.xcross, location.absolute, color=#FD1707, size=size.tiny, offset=0, display = display.all - display.price_scale - display.status_line)

// ================
// ==== Alerts ====
// ================
```

```
// Separate Alerts for Entries and Exits
//alertcondition(startLongTrade, title='Open Long ▲', message='LDC Open Long ▲ |
{{ticker}}@{{close}} | ({{interval}})')
//alertcondition(endLongTrade, title='Close Long ▲', message='LDC Close Long ▲ |
{{ticker}}@{{close}} | ({{interval}})')
//alertcondition(startShortTrade, title='Open Short ▼', message='LDC Open Short  |
{{ticker}}@{{close}} | ({{interval}})')
//alertcondition(endShortTrade, title='Close Short ▼', message='LDC Close Short ▼ |
{{ticker}}@{{close}} | ({{interval}})')
//
//// Combined Alerts for Entries and Exits
//alertcondition(startShortTrade or startLongTrade, title='Open Position ▲▼', message='LDC
Open Position ▲▼ | {{ticker}}@{{close}} | ({{interval}})')
//alertcondition(endShortTrade or endLongTrade, title='Close Position ▲▼', message='LDC
Close Position  ▲▼ | {{ticker}}@[{{close}}] | ({{interval}})')
//
//// Kernel Estimate Alerts
//alertcondition(condition=alertBullish, title='Kernel Bullish Color Change', message='LDC
Kernel Bullish ▲ | {{ticker}}@{{close}} | ({{interval}})')
//alertcondition(condition=alertBearish, title='Kernel Bearish Color Change', message='LDC
Kernel Bearish ▼ | {{ticker}}@{{close}} | ({{interval}})')

// =========================
// ==== Display Signals ====
// =========================

atrSpaced = useAtrOffset ? ta.atr(1) : na
compressionFactor = settings.neighborsCount / settings.colorCompression
c_pred = prediction > 0 ? color.from_gradient(prediction, 0, compressionFactor, #787b86,
#009988) : prediction <= 0 ? color.from_gradient(prediction, -compressionFactor, 0,
#CC3311, #787b86) : na
c_label = showBarPredictions ? c_pred : na
c_bars = showBarColors ? color.new(c_pred, 50) : na
x_val = bar_index
y_val = useAtrOffset ? prediction > 0 ? high + atrSpaced: low - atrSpaced : prediction > 0 ?
high + hl2*barPredictionsOffset/20 : low - hl2*barPredictionsOffset/30
label.new(x_val, y_val, str.tostring(prediction), xloc.bar_index, yloc.price,
color.new(color.white, 100), label.style_label_up, c_label, size.normal, text.align_left)
barcolor(showBarColors ? color.new(c_pred, 50) : na)
//
================================================================================
================================================================================
================================================================================
================================================================================
======
// ==================================== Strategy code
==============================================================
```

```
// ------ Inputs for calculating our amount position --------

initial_actual_capital = input.float(defval=10000, title = "Enter initial/current capital", group =
"============ Position amount calculator ==========")
risk_c = input.float(2.5, '% account risk per trade', step=1, group = "============ Position
amount calculator ==========", tooltip = "Percentage of total account to risk per trade.
The USD value that should be used to risk the inserted percentage of the account. Appears
as a yellow number in the upper left corner")

// ------ Date filter ---------

initial_date = input.time(title="Initial date", defval=timestamp("10 Feb 2014 13:30 +0000"),
group="============ Time filter ============ ", tooltip="Enter the start date and time of
the strategy")
final_date   = input.time(title="Final date", defval=timestamp("01 Jan 2030 19:30 +0000"),
group="============ Time filter ============ ", tooltip="Enter the end date and time of
the strategy")
dateFilter(int st, int et) => time >= st and time <= et
colorDate = input.bool(defval=false, title="Date background", tooltip = "Add color to the
period of time of the strategy tester")
bgcolor(colorDate and dateFilter(initial_date, final_date) ? color.new(color.blue, transp=90) :
na)
date = dateFilter(initial_date, final_date)

// ------ Session limits ------

timeSession = input.session(title="Time session", defval="0000-2400",
group="============ Time filter ============ ", tooltip="Session time to operate. It may
be different depending on your time zone, you have to find the correct hours manually.")
colorBG     = input.bool(title="Session background", defval=false, tooltip = "Add color to
session time background")
inSession(sess) => na(time(timeframe.period, sess + ':1234567')) == false
bgcolor(inSession(timeSession) and colorBG ? color.rgb(0, 38, 255, 84) : na)

//
==========================================================================
==========================================================================
===
// ------------ Super Trend ----------

atrPeriod = input(9, "ATR Length SuperTrend", group = "========= Super Trend filter
==========")
factor = input.float(2.5, "Factor SuperTrend", step = 0.05, group = "========= Super Trend
filter ==========")
[supertrend, direction_supertrend] = ta.supertrend(factor, atrPeriod)
show_supertrend = input.bool(defval = false, title="Show supertrend ?", group =
"============ Appearance ============")
bodyMiddle = plot(show_supertrend ? ((open + close) / 2) : na, display=display.none)
```

```
upTrend = plot(show_supertrend and direction_supertrend < 0 ? supertrend : na, "Up Trend",
color = color.green, style=plot.style_linebr, display = display.all - display.status_line)
downTrend = plot(show_supertrend and direction_supertrend > 0 ? supertrend : na, "Down
Trend", color = color.red, style=plot.style_linebr, display = display.all - display.status_line)
fill(bodyMiddle, upTrend, color.new(color.green, 95), fillgaps=false, title = "Supertrend
background")
fill(bodyMiddle, downTrend, color.new(color.red, 95), fillgaps=false, title = "Supertrend
background")
up_trend_plot = direction_supertrend < 0
down_trend_plot = direction_supertrend > 0

// ---------- Ema ----------------------

ema = input.int(200, title='Ema length', minval=1, maxval=500, group = "============
Trend ============")
ema200 = ta.ema(close, ema)
bullish = close > ema200
bearish = close < ema200
show_ema = input.bool(defval=true, title="Show ema ?", group = "============
Appearance ============")
plot(show_ema ? ema200 : na, title = "Ema", color=color.white, linewidth=2, display =
display.all - display.status_line - display.price_scale)

//
=================================================================================
=================================================================================
===
// ---------- Atr stop loss by garethyeo (modified) -----------------

source_atr = input(close, title='Source', group = "================== Stop loss
==================", inline = "A")
length_atr = input.int(14, minval=1, title='Period', group = "================== Stop loss
==================" , inline = "A")
multiplier = input.float(1.5, minval=0.1, step=0.1, title='Atr multiplier', group =
"================== Stop loss ==================", inline = "A", tooltip = "Defines
the stop loss distance based on the Atr stop loss indicator")
show_stoploss = input.bool(defval = true, title = "Show stop loss ?", group = "============
Appearance ============")
var float shortStopLoss = na
var float longStopLoss = na
var float atr_past_candle_long = na
var float atr_past_candle_short = na
candle_of_stoploss = input.string(defval = "Current candle", title = "Stop loss source for atr
stoploss", group = "================== Stop loss ==================", options =
["Current candle","Past candle"])
if candle_of_stoploss == "Current candle"
    shortStopLoss := source_atr + ta.atr(length_atr) * multiplier
    longStopLoss := source_atr - ta.atr(length_atr) * multiplier
```

```
if candle_of_stoploss == "Past candle"
    shortStopLoss := close[1] + ta.atr(length_atr)[1] * multiplier[1]
    longStopLoss := close[1] - ta.atr(length_atr)[1] * multiplier[1]

// --------- Stop loss based in last swing high/low --------

high_bars = input.int(defval = 10, title = "Highest price bars: ", group =
"================== Stop loss ==================")
low_bars = input.int(defval = 10, title = "Lowest price bars: ", group =
"================== Stop loss ==================")
stop_high = ta.highest(high, high_bars)
stop_low = ta.lowest(low, low_bars)

// --------- Stop loss source selection ----------

stoploss_type = input.string(defval = "Atr stop loss", title = "General stop loss source", group
= "================== Stop loss ==================", options = ["Atr stop
loss","Swing high/low"])
if stoploss_type == "Atr stop loss"
    shortStopLoss := source_atr + ta.atr(length_atr) * multiplier
    longStopLoss := source_atr - ta.atr(length_atr) * multiplier
if candle_of_stoploss == "Past candle" and stoploss_type == "Atr stop loss"
    shortStopLoss := close[1] + ta.atr(length_atr)[1] * multiplier[1]
    longStopLoss := close[1] - ta.atr(length_atr)[1] * multiplier[1]
if stoploss_type == "Swing high/low"
    shortStopLoss := stop_high
    longStopLoss := stop_low

// ======================================== Add/withdraw money frequently
(>>>Beta<<<)
============================================================

// Declare some variables
========================================================

var initial_capital = strategy.initial_capital
var initial_capital_a = strategy.initial_capital
var initial_capital_w = strategy.initial_capital
var float capital_added = 0
var float balance = strategy.initial_capital
var prev_month = 0
var float amount = 0
var add_frequency = 0
var withdraw_frequency = 0

// Choose how often the strategy adds money
========================================================
```

```
add_money_frequency = input.string("Monthly", title = "Choose how often would you add
money", options = ["Monthly","Weekly","Daily","Yearly"], group = "============== Adding
money frequently ==============")
amount_to_add = input.float(defval = 10, title = "How much you want to add ?", group =
"============== Adding money frequently ==============")

if add_money_frequency == "Monthly"
    add_frequency := month
if add_money_frequency == "Weekly"
    add_frequency := weekofyear
if add_money_frequency == "Daily"
    add_frequency := dayofweek
if add_money_frequency == "Yearly"
    add_frequency := year

// Choose if you want to add money or not ===================================

add_money = input.string("No", title = "Add money from time to time ?", options =
["Yes","No"] , group = "============== Adding money frequently ==============")

if add_frequency != add_frequency[1] and add_money == "Yes" and date
    initial_capital_a += amount_to_add
    initial_capital += amount_to_add
    balance := strategy.netprofit + initial_capital

// Choose how often the strategy withdraws money
=========================================================

amount_to_withdraw = input.string("Fixed", title = "Withdraw based in fixed amounts or
percentage of earnings", options = ["%","Fixed"], group = "============== Withdraw
money frequently ==============")
amount_for_withdraw = input.float(defval = 2, title = "How much you want to withdraw (fixed
amount) ?", group = "============== Withdraw money frequently ==============")
withdraw_money_frequency = input.string("Monthly", title = "Choose how often would you
withdraw money", options = ["Monthly","Weekly","Daily","Yearly"], group =
"============== Withdraw money frequently ==============")

// We use this for being able to choose the frequency of withdrawing money:
if withdraw_money_frequency == "Monthly"
    withdraw_frequency := month
if withdraw_money_frequency == "Weekly"
    withdraw_frequency := weekofyear
if withdraw_money_frequency == "Daily"
    withdraw_frequency := dayofweek
if withdraw_money_frequency == "Yearly"
    withdraw_frequency := year

// Choose if you want to withdraw money or not ============================
```

```
withdraw_money = input.string("No", title = "Withdraw money from time to time ?", options =
["Yes","No"], group = "=============== Withdraw money frequently ===============")
percentage_of_earnings = input.float(10, title = "Percentage of earnings", group =
"=============== Withdraw money frequently ===============", tooltip = "Use this if
withdraw is based in % of earnings")

// Percentage of earnings:
if withdraw_frequency != withdraw_frequency[1] and withdraw_money == "Yes" and
amount_to_withdraw == "%" and date and strategy.netprofit>0
    initial_capital_w -= strategy.netprofit * (percentage_of_earnings / 100)
    initial_capital -=  strategy.netprofit * (percentage_of_earnings / 100)
    balance := strategy.netprofit + initial_capital
// Fixed amount:
if withdraw_frequency != withdraw_frequency[1] and withdraw_money == "Yes" and
amount_to_withdraw == "Fixed" and date and strategy.netprofit>0
    initial_capital_w -= amount_for_withdraw
    initial_capital -= amount_for_withdraw
    balance := strategy.netprofit + initial_capital

// Logic

if withdraw_money == "Yes" and add_money == "No"
    amount := balance
if withdraw_money =="Yes" and add_money =="Yes"
    amount := balance
if withdraw_money == "No" and add_money == "Yes"
    amount := balance
if withdraw_money == "No" and add_money == "No"
    amount := strategy.equity

//
================================================================================
================================================================================
===
// ---------- Money management ----------

strategy_contracts = amount / close
distance_sl_long = -1 * (longStopLoss - close) / close
distance_sl_short = (shortStopLoss - close) / close
risk = input.float(2.5, '% Account risk per trade for backtesting', step=1, group =
"============ Risk management for trades ============", tooltip = "Percentage of total
account to risk per trade if fixed amounts is deactivated")
long_amount = strategy_contracts * (risk / 100) / distance_sl_long
short_amount = strategy_contracts * (risk / 100) / distance_sl_short

leverage=input.bool(defval=true, title="Use leverage for backtesting ?", group =
"============ Risk management for trades ============", tooltip = "If it is activated,
```

```
there will be no monetary units or amount of assets limit for each operation (That is, each
operation will not be affected by the initial / current capital since it would be using leverage).
If it is deactivated, the monetary units or the amount of assets to use for each operation will
be limited by the initial/current capital.")
if not leverage and long_amount>strategy_contracts
    long_amount:=amount/close

if not leverage and short_amount>strategy_contracts
    short_amount:=amount/close

// ---- Fixed amounts ----

fixed_amounts = input.bool(defval = false, title = "Fixed amounts ?", group =
"============ Risk management for trades ============", tooltip = "If you activate this,
the backtester will use fixed amounts")
fixed_amount_input = input.float(defval = 1000, title = "Fixed amount in usd", group =
"============ Risk management for trades ============")
if fixed_amounts
    long_amount := fixed_amount_input / close
    short_amount := fixed_amount_input / close

// ---------- Risk management ----------

risk_reward_breakeven_long= input.float(title="Risk/reward for breakeven long", defval=1.0,
step=0.1, group = "============ Risk management for trades ============")
risk_reward_take_profit_long= input.float(title="Risk/reward for take profit long", defval=3.0,
step=0.1, group = "============ Risk management for trades ============")
risk_reward_breakeven_short= input.float(title="Risk/reward for break even short",
defval=1.0, step=0.1, group = "============ Risk management for trades
============")
risk_reward_take_profit_short= input.float(title="Risk/reward for take profit short", defval=3.0,
step=0.1, group = "============ Risk management for trades ============")
tp_percent=input.float(title="% of trade for first take profit", defval=50, step=5, group =
"============ Risk management for trades ============", tooltip = "Closing
percentage of the current position when the first take profit is reached.")

//
==========================================================================
==========================================================================
===
// ---------- Trade conditions ----------
// Entry Conditions: Booleans for ML Model Position Entries
//startLongTrade = isNewBuySignal and isBullish and isEmaUptrend and isSmaUptrend
//startShortTrade = isNewSellSignal and isBearish and isEmaDowntrend and
isSmaDowntrend
//endLongTrade = settings.useDynamicExits and isDynamicExitValid ?
endLongTradeDynamic : endLongTradeStrict
```

```
//endShortTrade = settings.useDynamicExits and isDynamicExitValid ?
endShortTradeDynamic : endShortTradeStrict
//bullish := close > ema200
//bearish := close < ema200
bought = strategy.position_size > 0
sold = strategy.position_size < 0
buy  = startLongTrade
sell = startShortTrade
var float total_commissions_value = 0
var float commission_value_l = 0
var float commission_value_s = 0
var float sl_long = na
var float sl_short = na
var float be_long = na
var float be_short = na
var float tp_long = na
var float tp_short = na
var int totaltrades = 0

close_withsupertrend = input.bool(defval = true, title = "Close positions with supertrend ?",
group = "============ Positions management ============")
closeshort_supertrend=ta.crossover(close, supertrend)
closelong_supertrend=ta.crossunder(close, supertrend)

if not bought
    be_long:=na
    sl_long:=na
    tp_long:=na
if not sold
    be_short:=na
    sl_short:=na
    tp_short:=na
long_positions = input.bool(defval = true, title = "Long positions ?", group = "============
Positions management ============")
short_positions = input.bool(defval = true, title = "Short positions ?", group =
"============ Positions management ============")
use_takeprofit = input.bool(defval = true, title = "Use take profit ?", group = "============
Risk management for trades ============")
use_breakeven = input.bool(defval = true, title = "Use break even ?", group =
"============ Risk management for trades ============")
close_only_tp = input.bool(defval = false, title = "Close just with take profit ?", group =
"============ Risk management for trades ============", tooltip = "Activate if you just
want to exit from a position until reaching take profit or stop loss. If it´s activated, change %
of closing by tp to 100%")
ema_filter_long = input.bool(defval = true, title = "Ema filter for long positions ?", group =
"============ Positions management ============", tooltip = "Activate if you just want
to long above 200 ema")
```

```
ema_filter_short = input.bool(defval = true, title = "Ema filter for short positions ?", group =
"============ Positions management ============", tooltip = "Activate if you just want
to short under 200 ema")
commission_percent = input.float(0.03, title = "Commission value in %", group =
"============ Positions management ============", tooltip = "Set the % of
commission. For example, when you enter into a position, you have a commission of 0.04%
per entry and 0.04% per exit. You have also to change this value in properties for getting a
real return in backtest. (in this case, 0.04%)")

if fixed_amounts
    commission_value_l := (close * (long_amount) * (commission_percent/100))
    commission_value_s := (close * (short_amount) * (commission_percent/100))
if not fixed_amounts
    commission_value_l := (close * ((strategy_contracts * (risk / 100)) / distance_sl_long) *
(commission_percent/100))
    commission_value_s := (close * ((strategy_contracts * (risk / 100)) / distance_sl_short) *
(commission_percent/100))

// ======================== Strategy
================================================================================
==============================================

// Long position with take profit

if not bought and buy and date and long_positions and inSession(timeSession) and
use_takeprofit and not ema_filter_long
    sl_long:=longStopLoss
    long_stoploss_distance = close - longStopLoss
    be_long := close + long_stoploss_distance * risk_reward_breakeven_long
    tp_long:=close+(long_stoploss_distance*risk_reward_take_profit_long)
    total_commissions_value += commission_value_l
    strategy.entry('L', strategy.long, long_amount, alert_message = "Long")
    strategy.exit("Tp", "L", stop=sl_long, limit=tp_long, qty_percent=tp_percent)
    strategy.exit('Exit', 'L', stop=sl_long)
if bought and high > be_long and use_breakeven
    sl_long := strategy.position_avg_price
    strategy.exit("Tp", "L", stop=sl_long, limit=tp_long, qty_percent=tp_percent)
    strategy.exit('Exit', 'L', stop=sl_long)
if bought and sell and strategy.openprofit>0 and not close_only_tp or bought and
closelong_supertrend and close_withsupertrend and strategy.openprofit>0  and not
close_only_tp
    strategy.close("L", comment="CL")
    balance := balance + strategy.openprofit

// Long position without take profit

if not bought and buy and date and long_positions and inSession(timeSession) and not
ema_filter_long
```

```
    sl_long:=longStopLoss
    long_stoploss_distance = close - longStopLoss
    be_long := close + long_stoploss_distance * risk_reward_breakeven_long
    total_commissions_value += commission_value_l
    strategy.entry('L', strategy.long, long_amount, alert_message = "Long")
    strategy.exit('Exit', 'L', stop=sl_long)
if bought and high > be_long and use_breakeven
    sl_long := strategy.position_avg_price
    strategy.exit('Exit', 'L', stop=sl_long)
if bought and sell and strategy.openprofit>0
    strategy.close("L", comment="CL")
    balance := balance + strategy.openprofit

// Short position with take profit

if not sold and sell and date and short_positions and inSession(timeSession) and
use_takeprofit and not ema_filter_short
    sl_short:=shortStopLoss
    short_stoploss_distance=shortStopLoss - close
    be_short:=((short_stoploss_distance*risk_reward_breakeven_short)-close)*-1
    tp_short:=((short_stoploss_distance*risk_reward_take_profit_short)-close)*-1
    total_commissions_value += commission_value_s
    strategy.entry("S", strategy.short, short_amount, alert_message = "Short")
    strategy.exit("Tp", "S", stop=sl_short, limit=tp_short, qty_percent=tp_percent)
    strategy.exit("Exit", "S", stop=sl_short)
if sold and low < be_short and use_breakeven
    sl_short:=strategy.position_avg_price
    strategy.exit("Tp", "S", stop=sl_short, limit=tp_short, qty_percent=tp_percent)
    strategy.exit("Exit", "S", stop=sl_short)
if sold and buy and strategy.openprofit>0 and not close_only_tp or sold and
closeshort_supertrend and close_withsupertrend and strategy.openprofit>0 and not
close_only_tp
    strategy.close("S", comment="CS")
    balance := balance + strategy.openprofit

// Short position without take profit

if not sold and sell and date and short_positions and inSession(timeSession) and not
ema_filter_short
    sl_short:=shortStopLoss
    short_stoploss_distance=shortStopLoss - close
    be_short:=((short_stoploss_distance*risk_reward_breakeven_short)-close)*-1
    total_commissions_value += commission_value_s
    strategy.entry("S", strategy.short, short_amount, alert_message = "Short")
    strategy.exit("Exit", "S", stop=sl_short)
if sold and low < be_short and use_breakeven
    sl_short:=strategy.position_avg_price
    strategy.exit("Exit", "S", stop=sl_short)
```

```
if sold and buy and strategy.openprofit>0
    strategy.close("S", comment="CS")
    balance := balance + strategy.openprofit

//
=============================================================================
=============================================================================
===

// Long position with ema filter

// With take profit

if not bought and buy and date and long_positions and inSession(timeSession) and
use_takeprofit and bullish and ema_filter_long
    sl_long:=longStopLoss
    long_stoploss_distance = close - longStopLoss
    be_long := close + long_stoploss_distance * risk_reward_breakeven_long
    tp_long:=close+(long_stoploss_distance*risk_reward_take_profit_long)
    total_commissions_value += commission_value_l
    strategy.entry('L', strategy.long, long_amount, alert_message = "Long")
    strategy.exit("Tp", "L", stop=sl_long, limit=tp_long, qty_percent=tp_percent)
    strategy.exit('Exit', 'L', stop=sl_long)
if bought and high > be_long and use_breakeven
    sl_long := strategy.position_avg_price
    strategy.exit("Tp", "L", stop=sl_long, limit=tp_long, qty_percent=tp_percent)
    strategy.exit('Exit', 'L', stop=sl_long)
if bought and sell and strategy.openprofit>0 and not close_only_tp or bought and
closelong_supertrend and close_withsupertrend and strategy.openprofit>0  and not
close_only_tp
    strategy.close("L", comment="CL")
    balance := balance + strategy.openprofit

// Without take profit

if not bought and buy and date and long_positions and inSession(timeSession) and bullish
and ema_filter_long
    sl_long:=longStopLoss
    long_stoploss_distance = close - longStopLoss
    be_long := close + long_stoploss_distance * risk_reward_breakeven_long
    total_commissions_value += commission_value_l
    strategy.entry('L', strategy.long, long_amount, alert_message = "Long")
    strategy.exit('Exit', 'L', stop=sl_long)
if bought and high > be_long and use_breakeven
    sl_long := strategy.position_avg_price
    strategy.exit('Exit', 'L', stop=sl_long)
if bought and sell and strategy.openprofit>0
    strategy.close("L", comment="CL")
```

```
    balance := balance + strategy.openprofit

// Short positon with ema filter

// With take profit
if not sold and sell and date and short_positions and inSession(timeSession) and
use_takeprofit and bearish and ema_filter_short
    sl_short:=shortStopLoss
    short_stoploss_distance=shortStopLoss - close
    be_short:=((short_stoploss_distance*risk_reward_breakeven_short)-close)*-1
    tp_short:=((short_stoploss_distance*risk_reward_take_profit_short)-close)*-1
    total_commissions_value += commission_value_s
    strategy.entry("S", strategy.short, short_amount, alert_message = "Short")
    strategy.exit("Tp", "S", stop=sl_short, limit=tp_short, qty_percent=tp_percent)
    strategy.exit("Exit", "S", stop=sl_short)
if sold and low < be_short and use_breakeven
    sl_short:=strategy.position_avg_price
    strategy.exit("Tp", "S", stop=sl_short, limit=tp_short, qty_percent=tp_percent)
    strategy.exit("Exit", "S", stop=sl_short)
if sold and buy and strategy.openprofit>0 and not close_only_tp or sold and
closeshort_supertrend and close_withsupertrend and strategy.openprofit>0 and not
close_only_tp
    strategy.close("S", comment="CS")
    balance := balance + strategy.openprofit

// Without take profit

if not sold and sell and date and short_positions and inSession(timeSession) and bearish
and ema_filter_short
    sl_short:=shortStopLoss
    short_stoploss_distance=shortStopLoss - close
    be_short:=((short_stoploss_distance*risk_reward_breakeven_short)-close)*-1
    total_commissions_value += commission_value_s
    strategy.entry("S", strategy.short, short_amount, alert_message = "Short")
    strategy.exit("Exit", "S", stop=sl_short)
if sold and low < be_short and use_breakeven
    sl_short:=strategy.position_avg_price
    strategy.exit("Exit", "S", stop=sl_short)
if sold and buy and strategy.openprofit>0
    strategy.close("S", comment="CS")
    balance := balance + strategy.openprofit

//
==========================================================================
==========================================================================
===

// ---------- Draw positions and signals on chart (strategy as an indicator) ----------
```

```
if high>tp_long
    tp_long:=na
if low<tp_short
    tp_short:=na
if high>be_long
    be_long:=na
if low<be_short
    be_short:=na

show_position_on_chart = input.bool(defval=true, title="Draw position on chart ?", group =
"============ Appearance ============", tooltip = "Activate to graphically display
profit, stop loss and break even")
position_price = plot(show_position_on_chart? strategy.position_avg_price : na,
style=plot.style_linebr, color = color.new(#ffffff, 10), linewidth = 1, display = display.all -
display.status_line - display.price_scale)

sl_long_price = plot(show_position_on_chart and bought ? sl_long : na, style =
plot.style_linebr, color = color.new(color.red, 10), linewidth = 1, display = display.all -
display.status_line - display.price_scale)
sl_short_price = plot(show_position_on_chart and sold ? sl_short : na, style =
plot.style_linebr, color = color.new(color.red, 10), linewidth = 1, display = display.all -
display.status_line - display.price_scale)

tp_long_price = plot(bought and show_position_on_chart and use_takeprofit? tp_long : na,
style = plot.style_linebr, color = color.new(#4cd350, 10), linewidth = 1, display = display.all -
display.status_line - display.price_scale)
tp_short_price = plot(sold and show_position_on_chart and use_takeprofit? tp_short : na,
style = plot.style_linebr, color = color.new(#4cd350, 10), linewidth = 1, display = display.all -
display.status_line - display.price_scale)

breakeven_long = plot(bought and high<be_long and show_position_on_chart and
use_breakeven? be_long : na , style = plot.style_linebr, color = color.new(#1fc9fd, 60),
linewidth = 1, display = display.all - display.status_line - display.price_scale)
breakeven_short = plot(sold and low>be_short and show_position_on_chart and
use_breakeven? be_short : na , style = plot.style_linebr, color = color.new(#1fc9fd, 60),
linewidth = 1, display = display.all - display.status_line - display.price_scale)

show_tpbe_on_chart = input.bool(defval=true, title="Draw first take profit/breakeven price on
chart ?", group = "============ Appearance ============", tooltip = "Activate to
display take profit and breakeven price. It appears as a green point in the chart")
long_stoploss_distance = close - longStopLoss
short_stoploss_distance=shortStopLoss - close
be_long_plot = close + long_stoploss_distance * risk_reward_breakeven_long
be_short_plot =((short_stoploss_distance*risk_reward_breakeven_short)-close)*-1
tp_long_plot = close+(long_stoploss_distance*risk_reward_take_profit_long)
tp_short_plot = ((short_stoploss_distance*risk_reward_take_profit_short)-close)*-1
```

```
plot(show_tpbe_on_chart and buy and use_breakeven and bullish and ema_filter_long or show_tpbe_on_chart and buy and use_breakeven and not ema_filter_long ? be_long_plot : na, color=color.new(#1fc9fd, 10), style = plot.style_circles, linewidth = 2, display = display.all - display.price_scale)
plot(show_tpbe_on_chart and sell and use_breakeven and bearish and ema_filter_short or show_tpbe_on_chart and sell and use_breakeven and not ema_filter_short ? be_short_plot : na, color=color.new(#1fc9fd, 10), style = plot.style_circles, linewidth = 2, display = display.all - display.price_scale)
plot(show_tpbe_on_chart and buy and use_takeprofit and bullish and ema_filter_long or show_tpbe_on_chart and buy and use_takeprofit and not ema_filter_long? tp_long_plot : na, color=color.new(#4cd350, 10), style = plot.style_circles, linewidth = 2, display = display.all - display.price_scale)
plot(show_tpbe_on_chart and sell and use_takeprofit and bearish and ema_filter_short or show_tpbe_on_chart and sell and use_takeprofit and not ema_filter_short? tp_short_plot : na, color=color.new(#4cd350, 10), style = plot.style_circles, linewidth = 2, display = display.all - display.price_scale)
plot(show_stoploss and buy and bullish and ema_filter_long or show_stoploss and buy and not ema_filter_long ? longStopLoss : na, title = "stop loss long", color = color.white, style = plot.style_circles, linewidth = 2)
plot(show_stoploss and sell and bearish and ema_filter_short or show_stoploss and sell and not ema_filter_long ? shortStopLoss : na, title = "stop loss short", color = color.white, style = plot.style_circles, linewidth = 2)

position_profit_long = plot(bought and show_position_on_chart and strategy.openprofit>0 ? close : na, style = plot.style_linebr, color = color.new(#4cd350, 10), linewidth = 1, display = display.all - display.status_line - display.price_scale)
position_profit_short = plot(sold and show_position_on_chart and strategy.openprofit>0 ? close : na, style = plot.style_linebr, color = color.new(#4cd350, 10), linewidth = 1, display = display.all - display.status_line - display.price_scale)

fill(plot1 = position_price, plot2 = position_profit_long, color = color.new(#4cd350, 90))
fill(plot1 = position_price, plot2 = position_profit_short, color = color.new(#4cd350, 90))

fill(plot1 = position_price, plot2 = sl_long_price, color = color.new(color.red,90))
fill(plot1 = position_price, plot2 = sl_short_price, color = color.new(color.red,90))

fill(plot1 = position_price, plot2 = tp_long_price, color = color.new(color.green,90))
fill(plot1 = position_price, plot2 = tp_short_price, color = color.new(color.green,90))

show_signals = input.bool(defval=true, title="Show signals on chart ?", group = "============ Appearance ============")
plotshape(show_signals and buy and bullish and ema_filter_long or show_signals and buy and not ema_filter_long ? low : na, title='Buy', text='Buy', style=shape.labelup, location=location.belowbar, color=color.new(color.green, 20), textcolor=color.new(color.white, 0), size=size.tiny , display = display.all - display.price_scale - display.status_line)
plotshape(show_signals and sell and bearish and ema_filter_long or show_signals and sell and not ema_filter_short ? high : na, title='Sell', text='Sell', style=shape.labeldown,
```

```
location=location.abovebar, color=color.new(color.red, 20), textcolor=color.new(color.white,
0), size=size.tiny, display = display.all - display.price_scale - display.status_line)
plotshape(show_signals and closelong_supertrend and close_withsupertrend and bought or
show_signals and sell and bought ? low : na, title='Cl Buy', text='Cl Buy',
style=shape.labelup, location=location.belowbar, color=color.new(#4cafaf, 30),
textcolor=color.new(color.white, 0), size=size.tiny , display = display.all - display.price_scale -
display.status_line)
plotshape(show_signals and closeshort_supertrend and close_withsupertrend and sold or
show_signals and buy and bought? high : na, title='Cl Buy', text='Cl sell',
style=shape.labeldown, location=location.abovebar, color=color.new(#4cafaf, 30),
textcolor=color.new(color.white, 0), size=size.tiny, display = display.all - display.price_scale -
display.status_line)

//
==========================================================================
==========================================================================
===
// ---------- Positions amount calculator  ----------

contracts_amount_c = initial_actual_capital / close
distance_sl_long_c = -1 * (longStopLoss - close) / close
distance_sl_short_c = (shortStopLoss - close) / close
long_amount_c = close * (contracts_amount_c * (risk_c / 100) / distance_sl_long_c)
short_amount_c = close * (contracts_amount_c * (risk_c / 100) / distance_sl_short_c)
long_amount_lev = close * (contracts_amount_c * (risk_c / 100) / distance_sl_long_c)
short_amount_lev = close * (contracts_amount_c * (risk_c / 100) / distance_sl_short_c)

leverage_for_calculator=input.bool(defval=true, title="Use leverage ?", group =
"============ Position amount calculator ===========", tooltip = "If it is activated, there
will be no monetary units or amount of assets limit for each operation (That is, each
operation will not be affected by the initial / current capital since it would be using leverage).
If it is deactivated, the monetary units or the amount of assets to use for each operation will
be limited by the initial/current capital.")

if not leverage_for_calculator and long_amount_lev>initial_actual_capital
    long_amount_lev:=initial_actual_capital

if not leverage_for_calculator and short_amount_lev>initial_actual_capital
    short_amount_lev:=initial_actual_capital

plot(buy and leverage_for_calculator ? long_amount_c : na, color = color.rgb(255, 230, 0),
display = display.all - display.pane - display.price_scale)
plot(sell and leverage_for_calculator ? short_amount_c : na, color = color.rgb(255, 230, 0),
display = display.all - display.pane - display.price_scale)
plot(buy and not leverage_for_calculator ? long_amount_lev : na, color = color.rgb(255, 230,
0), display = display.all - display.pane - display.price_scale)
plot(sell and not leverage_for_calculator ? short_amount_lev : na, color = color.rgb(255, 230,
0), display = display.all - display.pane - display.price_scale)
```

```
//
================================================================================
================================================================================
===
// ==================== Drawing stats about add and withdraw money frequently and
others on chart ==================

if not bought and buy and date and not ema_filter_long
    totaltrades += 1
if not sold and sell and date and not ema_filter_short
    totaltrades += 1
if not bought and buy and date and bearish and ema_filter_long
    totaltrades += 0
if not sold and sell and date and bullish and ema_filter_short
    totaltrades += 0
if not bought and buy and date and bullish and ema_filter_long
    totaltrades += 1
if not sold and sell and date and bearish and ema_filter_short
    totaltrades += 1

total_money_added = initial_capital_a - strategy.initial_capital
total_money_withdrawn = initial_capital_w - strategy.initial_capital
final_money = total_money_added + strategy.netprofit + strategy.initial_capital -
total_money_withdrawn  // or current money avaliable
// plot(commission_value_l, color = color.rgb(59, 245, 255), display = display.all -
display.pane)
// plot(commission_value_s, color = color.rgb(59, 245, 255), display = display.all -
display.pane)
// plot(total_commissions_value, color = color.rgb(252, 59, 255), display = display.all -
display.pane)
// plot(total_trades/2, color = color.rgb(59, 245, 255), display = display.all - display.pane)
// plot(final_money, color = color.yellow, display = display.all - display.pane)
// plot(total_money_added, color = color.blue, display = display.all - display.pane)
// plot(total_money_withdrawn, color = color.red, display = display.all - display.pane)
// plot(strategy.netprofit, color = color.green, display = display.all - display.pane)

truncate(_number, _decimalPlaces) =>
    _factor = math.pow(10, _decimalPlaces)
    int(_number * _factor) / _factor

draw_stats = input.bool(true, "Show stats from add/withdraw money frequently ?", group =
"============ Appearance ============")

// Prepare stats table

var table testTable = table.new(position.top_right, 5, 2, border_width=2, frame_color =
color.rgb(0, 0, 0), frame_width = 2)
```

```
f_fillCell(_table, _column, _row, _title, _value, _bgcolor, _txtcolor,_size,_tooltip) =>
    _cellText = _title + "\n" + _value
    table.cell(_table, _column, _row, _cellText, bgcolor=_bgcolor, text_color=_txtcolor,
text_size=_size, tooltip = _tooltip)

// Draw stats table

var bgcolor = color.new(#2f5cda, 50)
var bgcolor2 = color.new(color.green, 50)
var bgcolor3 = color.new(color.red,50)
if draw_stats
    if barstate.islastconfirmedhistory
        f_fillCell(testTable, 0, 0, "Final/current money:", "$" +
str.tostring(truncate(final_money,2)), bgcolor, color.white, _size = size.normal, _tooltip =
"Total money added + total return from strategy + initial capital - total money withdrawn")
        f_fillCell(testTable, 0, 1, "Total money added:", "$" +
str.tostring(truncate(total_money_added,2)), bgcolor, color.white, _size = size.normal,
_tooltip = "Sum of total money added at the end of the date")
        f_fillCell(testTable, 1, 0, "Total money withdrawn:", "$" +
str.tostring(truncate(total_money_withdrawn*-1,2)), bgcolor, color.white, _size = size.normal,
_tooltip = "Sum of total money withdrawn at the end of the date")
        f_fillCell(testTable, 1, 1, "Total return:", "$" + str.tostring(truncate(strategy.netprofit,2)),
strategy.netprofit > 0 ? bgcolor2 : bgcolor3 , color.white, _size = size.normal, _tooltip = "Total
return from strategy until the end of the date (it could be different from default backtesting if
last position have not been closed completly)")
        f_fillCell(testTable, 2, 0, "Total trades:", "" + str.tostring(truncate(totaltrades,2)), bgcolor,
color.white, _size = size.normal, _tooltip = "Sum of real total trades. The value from default
backtester it´s not precise")
        f_fillCell(testTable, 2, 1, "Total commissions value:", "" +
str.tostring(truncate(total_commissions_value,2)), bgcolor, color.white, _size = size.normal,
_tooltip = "Sum of commissions value from all trades. You must set the % value in the script
settings (Positions management).")
//
==========================================================================
```