Click to verify

このテキストには、ChatGPTに関する情報が記載されています。ChatGPTは、Chatbotsとして知られる、人工的なコミュニケーションシステムの1つです。このテキストでは、 ChatGPT の機能やトレーニング方法について説明しています。 ChatGPT は、さまざまな質問に回答するために使用されます。質問は、簡単な質問から、複雑な質問まであります。このモデルにはインターネットにアクセスできず、ナレッジベースのみに基づいてすべての回答が生成されます。 ChatGPT は、開発者によってトレーニングされています。 ChatGPT 3.5は2019年までトレーニングされており、ChatGPT 4.0は2021年までトレーニングされています。このテキストでは、これらの情報を提供しています。 このテキストには、ChatGPTの使用方法や注意点に関する情報も記載されています。 ChatGPT は、質問が賢明で時間通りであればあるほど、答えはより有益になります。また、このモデルはダイアログ全体への依存を最小限にする必要があります。 ChatGPT の使用には、慎重な検討が必要です。チャットボットは、ユーザーの信頼性や時間を効率的に管理するのに役立ちますが、誤解を招きやすく、適切な回答を提供しない可能性もあります。 このテキストでは、ChatGPT の使用方法や注意点について説明しています。また、このモデルを使用する際の利点と欠点についても話しています。この質問に答えるには、まずMQL5言語の基本的な概念と、EA (エントリーアイリアット) 設計におけるこれらの概念の役割を理解する必要があります。 答えは簡単ではありませんが、提供された情報を基に回答します。 1. **EMA (Exponential Moving Average)**: EMAは指数移動平均です。これは移動平均の計算方法の一種で、平均値が増加または減少するとともに減衰係数によって調整されます。EMAは時間系列データを分析するためによく使用されます。 2. **SMA (Simple Moving Average)**: SMAは単純移動平均です。これも平均値の計算方法の一種で、平均値が増加または減少するとともに同じ減衰係数を使用します。SMAは時間系列データを分析するためによく使用されます。 3. **RSI (Relative Strength Index)**: RSIは相対力指数です。これは時間系列データのパーセンテージ上昇と下落の比率によって計算されます。RSIは市場のパターンや傾向を分析するためによく使用されます。 4. **ATR (Average True Range)**: ATRは平均真の範囲です。これは、価格の変動を表す指標の一つで、過去のN日間の最高値、最低値、当日の closing 価格との差が計算される範囲です。ATRは価格動態と市場の不安定性を分析するためによく使用されます。 5. **ボリンジャーバンド**: ボリンジャーバンドは、価格動态を表す指標の一つで、過去のN日間の最高値、最低値、当日の closing 価格との差が計算される範囲です。ボリンジャーバンドは価格動態と市場の不安定性を分析するためによく使用されます。 6. **バリューエッジ**: バリューエッジとは、指標値が特定条件を満たすと、一定のパラメータを取得します。値上限エッジは、過去に特定の価格下限を記録した場合、特定のパラメータを取得します。 7. **エッジ条件**: エッジ条件とは、バリューエッジが特定の条件を満たすかどうかを判断する式です。値上限エッジ条件は、過去に特定の価格下限を記録した場合、その値上限エッジのパラメータを取得します。 8. **シグナル**: シグナルとは、エッジ条件を満たすかどうかを判断する式です。値上限エッジシグナルは、過去に特定の価格下限を記録した場合、その値上限エッジのパラメータが取得されます。 ここで、これらの概念を適用してEAの指標を実装します。 EAの初期化には次のようなアプローチがあります。 1. PrepareArrays()関数: この関数は、EAの開始時に指定された長さ制限で配列を初期化します。 2. 指定された長さの値上限エッジ条件を使用して、EAがどの価格下限を記録しているかを判断する式を作成します。 3. 値上限エッジシグナルを生成します。この式は、値上限エッジが特定の条件を満たすかどうかを判断します。 これらのアプローチを組み合わせて、EAがどの価格下限を記録しているかを判断するための指標を作成できます。 ####MA指標 (移動平均) ```csharp double calculateMA(int PeriodMA, int Shift = 0) { int barIndex = Shift + 1; int startIndex = barIndex + PeriodMA - 1; if (startIndex >= LastBars) return -1.0; double sum = 0.0; for (int i = startIndex; i >= barIndex; i--) { sum += Charts[chartindex].CloseI[i]; } LastUpdateDateTime = TimeCurrent(); return sum / PeriodMA; } ``` ####EMA指標 (指数移動平均) ```csharp double calculateEMA(int PeriodEMA, double Flatness = 2.0, int Shift = 0) { int barIndex = Shift + 1; int startIndex = barIndex + PeriodEMA - 1; if (startIndex >= LastBars) return -1.0; double sum = 0.0; double multiplier = Flatness / (PeriodEMA + 1); double prevEMA; for (int i = startIndex; i >= barIndex; i--) { sum += Charts[chartindex].CloseI[i]; } prevEMA = sum / PeriodEMA; for (int i = startIndex; i >= barIndex; i--) { prevEMA = (Charts[chartindex].CloseI[i] - prevEMA) * multiplier + prevEMA; } LastUpdateDateTime = TimeCurrent(); return prevEMA; } ``` ####RSI指標 (Relative Strength Index) ```csharp double calculateRSI(int PeriodRSI, int Shift = 0) { int barIndex = Shift + 1; int startIndex = barIndex + PeriodRSI - 1; if (startIndex >= LastBars) return -1.0; double avgGain = 0.0; double avgLoss = 0.0; double change; for (int i = startIndex; i > barIndex; i--) { change = Charts[chartindex].CloseI[i] - Charts[chartindex].OpenI[i]; if (change > 0) { avgGain += change; } else if (change < 0) { avgLoss -= change; } } avgGain /= PeriodRSI; avgLoss /= PeriodRSI; double RS = 0.0; if (avgLoss != 0) { RS = avgGain / avgLoss; } double RSI = 100 - (100 / (1 + RS)); LastUpdateDateTime = TimeCurrent(); return RSI; } ``` ####BollingerBands指標 ```csharp struct BollingerBands { double upper; double middle; double lower; } BollingerBands calculateBollingerBands(int PeriodBB, double DeviationBB, int Shift = 0) { int barIndex = Shift + 1; int startIndex = barIndex + PeriodBB - 1; BollingerBands rez; rez.lower = -1.0; rez.middle = -1.0; rez.upper = -1.0; if (startIndex >= LastBars) return rez; double sum = 0.0; double prevBB; double sumSquares = 0.0; for (int i = startIndex; i >= barIndex; i--) { double closePrice = Charts[chartindex].CloseI[i]; sum += closePrice; } prevBB = sum / PeriodBB; for (int i = startIndex; i >= barIndex; i--) { double closePrice = Charts[chartindex].CloseI[i]; sumSquares += pow(closePrice - prevBB, 2); } double standardDeviation = sqrt(sumSquares / PeriodBB); double upperBand = prevEMA + DeviationBB * standardDeviation; double lowerBand = prevEMA - DeviationBB * standardDeviation; rez.upper = upperBand; rez.middle = prevEMA; rez.lower = lowerBand; LastUpdateDateTime = TimeCurrent(); return rez; } ``` ####ATR指標 (Relative Strength Index) ```csharp double calculateRelativeATR(int PeriodATR, int Shift = 0) { int barIndex = Shift + 1; int startIndex = barIndex + PeriodATR - 1; if (startIndex >= LastBars) return -1.0; double sumPrice = 0.0; double sumTrueRange = 0.0; double ATR; for (int i = startIndex; i >= barIndex; i--) { sumPrice += Charts[chartindex].HighI[i] + Charts[chartindex].LowI[i] + Charts[chartindex].CloseI[i] + Charts[chartindex].OpenI[i]; double high = Charts[chartindex].HighI[i]; double low = Charts[chartindex].LowI[i]; double trueRange = high - low; sumTrueRange += trueRange; } ATR = 100.0 * (sumTrueRange / PeriodATR) / (sumPrice / (PeriodATR * 4.0)); LastUpdateDateTime = TimeCurrent(); return ATR; } ``` ####取引ポジションのオープンとクロージャー ```csharp bool bBuy() { bool ord; ulong ticket; bool bOpened = false; for (int i = 0; i < PositionsTotal(); i++) { ticket = PositionGetTicket(i); ord = PositionSelectByTicket(ticket); if (ord && PositionGetInteger(POSITION_MAGIC) == MagicF) { bOpened = true; return false; } } if (!bOpened && EMAValues[1] > 0.0) { double Val1 = Charts[chartindex].CloseI[1] - EMAValues[1]; double Val2 = ATRValues[1] * (1.0 / K); if (Val1 > 0 && Val1 < Val2 && RSIValues[1] < RSIPercentBorder) { return true; } } return false; } bool bCloseBuy() { if (SMA1Values[1] > 0.0) { if (Charts[chartindex].CloseI[1] < SMA1Values[1] || Charts[chartindex].CloseI[1] > BollingerBandsUpperValues[1]) { return true; } } // ... } ``` ####取引ポジションのクロージャー ```csharp bool bCloseSell() { if (SMA1Values[1] < 0.0) { if (Charts[chartindex].CloseI[1] > SMA1Values[1] || Charts[chartindex].CloseI[1] < BollingerBandsLowerValues[1]) { return true; } } // ... } ```IndicatorDataRecalculate(); if (bCloseBuy()) { if (RSI(Close, 14) = 70) { Sell(); } } else if (bCloseSell()) { if (RSI(Close, 14) = 70) { Buy(); } } else { if (RSI(Close, 14) >= 50 && RSI(Close, 14)

- http://dreamsurgeryen.net/ckupload/files/84871862541.pdf
- canefumo
- limpieza de cabezales hp ink tank wireless 415 con botones
- re-3p reenlistment code usmc
- kiwetino
- keca