MATRİKS

BİLGİ DAĞITIM HİZMETLERİ

AlgoTrader
Help File/Tanıtım

Algo Trader

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

1.Sistem özellikleri

- C# modülü
- Intellisense
- İndikatörlerin çift tıklanarak eklenebilmesi
- Backtesting, optimization
- C# ile gelişmiş algoritmalar oluşturma imkanı
- Yazılan tek bir stratejinin backtest, optimizasyon ya da realtime çalıştırma için ortak kullanımı
- Çoklu sembol kullanımı
- Çoklu periyot kullanımı
- Kar Al/Zarar Durdur, Trailing Stop kullanımı
- Tarihsel bar data ile strateji oluşturma
- Yüzeysel veriler üzerinden strateji oluşturma
- Derinlik verileri ile strateji oluşturma
- Yapay Zeka kütüphanelerini algoritma içinde kullanım imkanı
- Hazır şablonlarla hızlıca strateji yazımı
- Hazır stratejilerde parametre değişiklikleri ile kod yazmadan kullanım olanağı
- Overall'un stratejide kullanımı
- Emir miktarlarının strateji overall değeri ya da trading hesap bilgilerine göre ayarlanması
- Sıralı ya da sırasız emir gönderimi
- Bar data açılışı ya da istenilen zaman aralıklarında stratejinin tetiklenmesinin sağlanması
- Uyarı yapısı ile strateji üzerinden uyarı oluşturma
- Habere bağlı strateji yazımı
- Debug'a istenen değerlerin yazdırılması
- Kullanıcı grafiklerinin oluşturulabilmesi
- Loglama

Strateji Editörü

- Hatalı yazımların olduğu satırların tespiti ve gerekli açıklamaların derleme sırasında Error List alanında gösterimi
- Intellisense ile yazım sırasında otomatik tamamlama ve kullanılabilecek alanların listelenmesi
- Kodu düzenleme, formatlama için kısayollar
- Parametre bilgilerinin, açıklamaların gösterimi
- Hızlı kod ekleme kısayolları (for, switch, while)
- Araç kutusu üzerinden indikatör ve fonksiyon ekleme kolaylığı

Backtest

İstenilen zaman aralıklarında backtest imkanı

Parametrik değerlerin değiştirilerek backtest edilmesi

İşlemlerin puan, yüzde ya da fiyat adımı kadar kayması ile gösterimi ve overall hesaplaması

Sembol ve overall eğrisi grafikleri

Backtest raporu

Dock modu ile overall, rapor ve strateji seçenekleri, emir listeleri ekranlarının istendiği gibi dizayn edilebilmesi ve kaydı

Geçmiş backtest raporlarına erişim

Optimizasyon

Grid search ve bayesian yöntemleri ile optimizasyon imkanı

Aralık vererek ya da değişkenlerin tek tek tanımlanarak optimize edilebilmesi

Çoklu sembol, periyot ile optimizasyon

Optimizasyon sonuçlarının excele aktarımı

Realtime Çalıştırma

Sonraki bar açılışı ya da timer ile istenen aralıklarda strateji tetiklenmesi

Portföy verileriyle kullanım olanağı

Uygulama açık kaldığı sürece ekranın bağımsız çalışma olanağı

Çalışan stratejiler ekranı üzerinden toplu durdurma, rapor açma işlemleri

Çalıştırılmış stratejilere erişim

Explorer

İstenilen sayıda kolon ekleme ve kolonlarda hesaplanan değerleri listeleme

Explorer çalıştırılmadan önce otomatik data tamamlama

Çoklu sembol ve periyot seçimi ile pratik şekilde filtreleme

Hazır stratejilerde parametre değişiklikleri ile kod yazmadan kullanım olanağı

Sonuçlar üzerinden yeni explorer başlatma

Sonuçlar üzerinden toplu emir gönderimi

Sonuçları fiyat penceresine atama

Sonuçları grafik döngü sembolü olarak atama

Excele aktarım

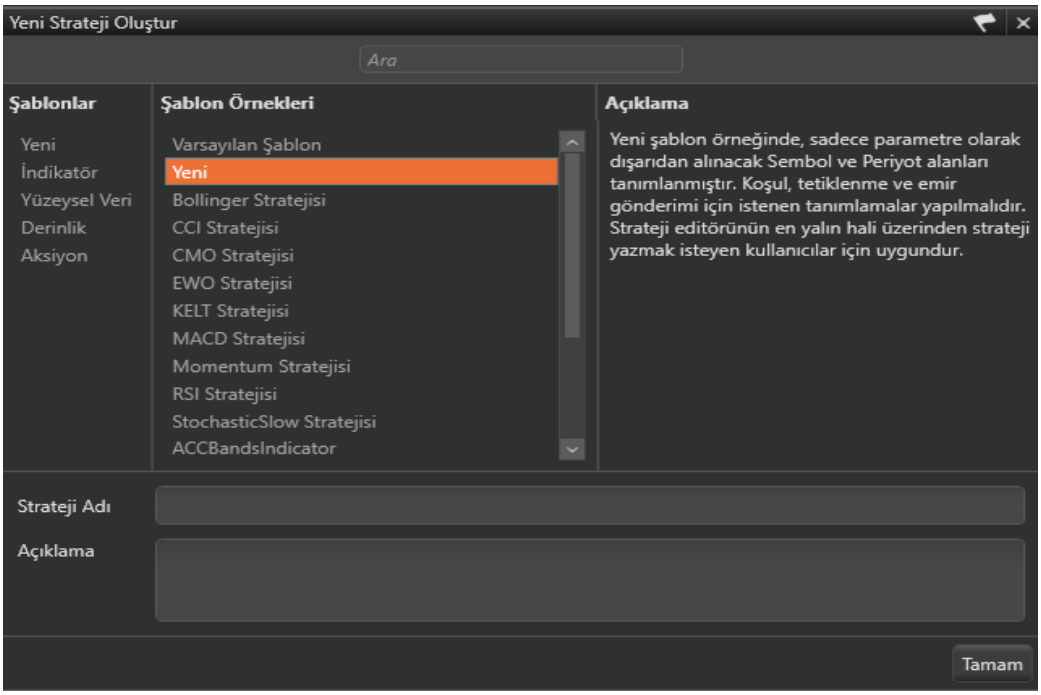
2. Strateji Yapısı

A. Algo Trader Menüsü

Yeni Strateji Oluştur

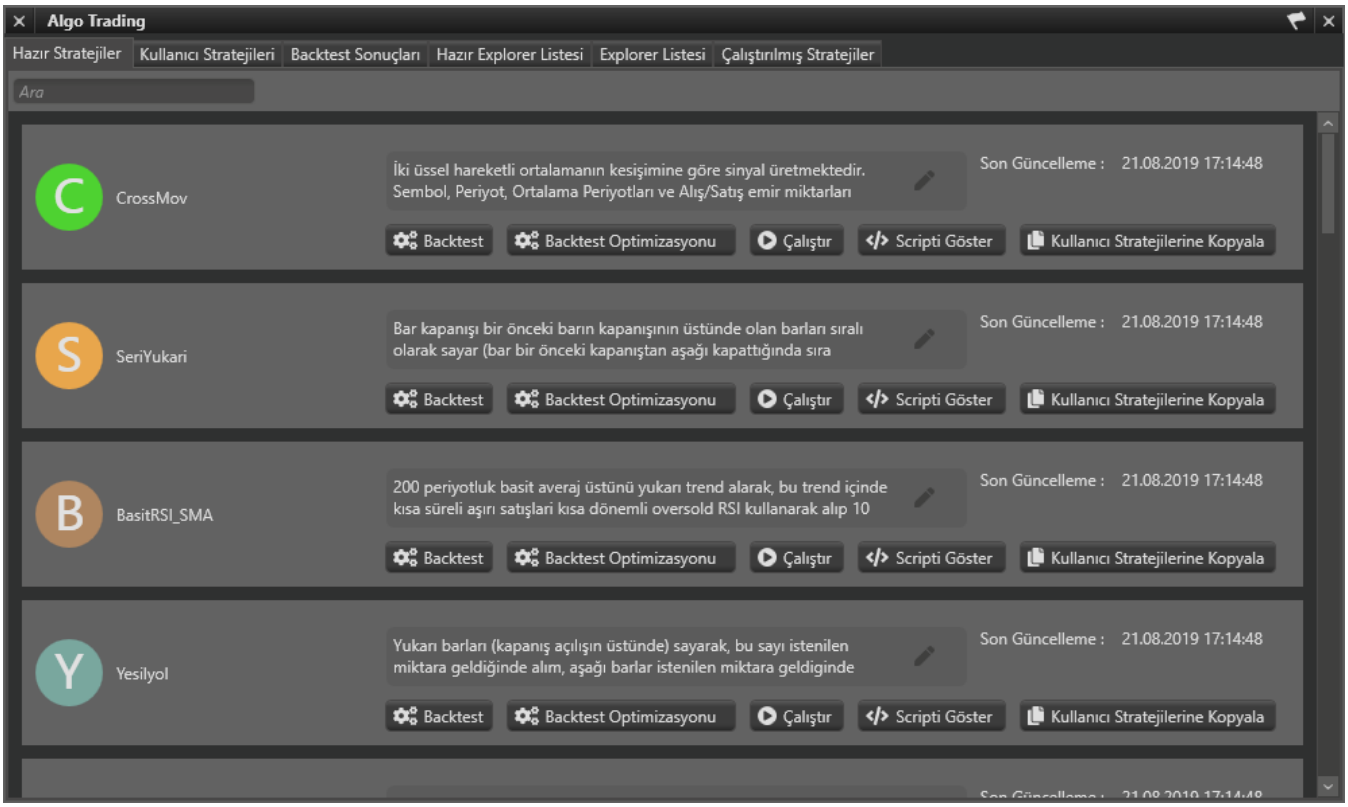
Yeni strateji oluşturmak için kullanılır. Kullanıcılar kendi stratejilerini yaratabilir veya hazır örnek şablonlardan kullanabilir.

Yeni bir strateji oluşturulduğunda C# modülünün kullanıldığı kod penceresi açılacaktır. Yeni bir strateji oluşturmak için “Strateji Adı” bölümü doldurulmak zorundadır. Bu alana yazılacak isim daha sonra “Strateji Listesi” içinden bulunabilir ve tekrar kullanılabilir. “Açıklama” kısmının doldurulması zorunlu değildir. Bu alana stratejinin nasıl çalıştığı ve/veya içerisinde kullanılan semboller, indikatörler yazılabilir.



Hazır Stratejiler

Bu alanda uygulama ile birlikte gelen örnek stratejiler bulunur. Hazır stratejiler içersinde bulunan örnek stratejileri kullanıcılar ister direk, isterlerse üzerinde değişikler yapıp kullanabilir. “Kullanıcı Stratejilerine Kopyala” butonuna basılarak burada bulunan stratejilerden herhangi birini “Kullanıcı Stratejileri” bölümüne eklenebilir ve üzerinde değişiklik yapılabilir.

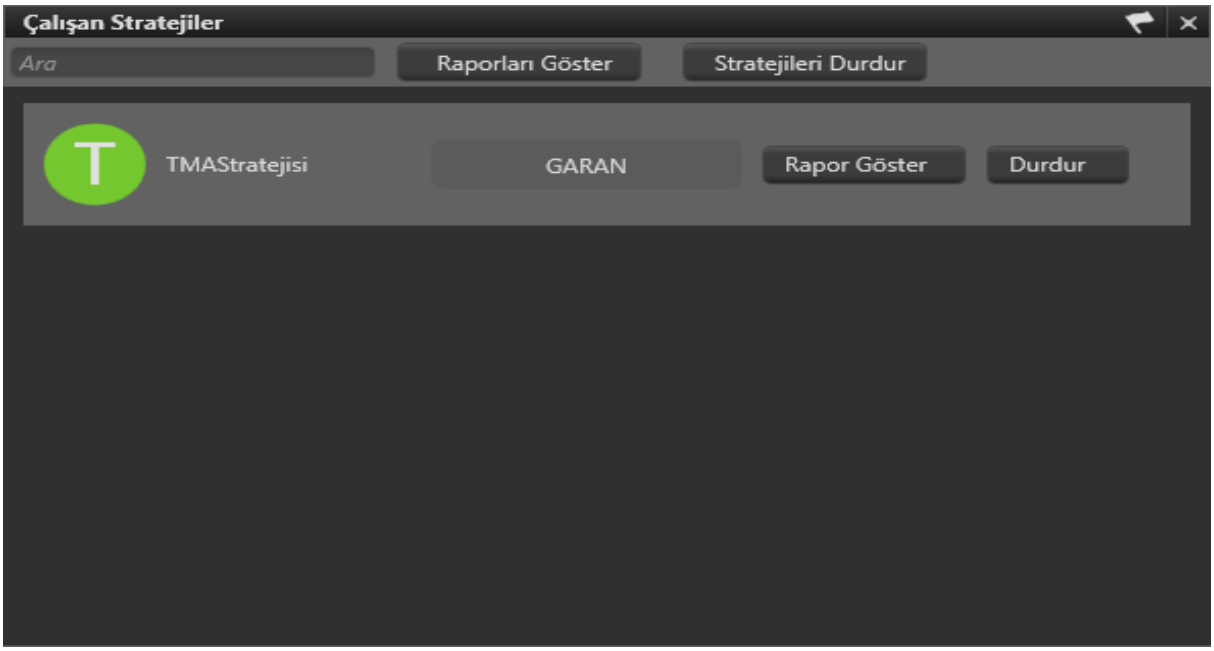


Kullanıcı Stratejileri

Bu alanda kullanıcıların oluşturdukları stratejiler bulunur. Oluşturulan her strateji bu alana kayıt edilir. Hazır stratejiler kısmından kopyalanan stratejiler de bu alana eklenir.

Çalışan Stratejiler

Oluşturalan veya hazır olarak bulunan stratejiler çalıştırıldığında “Çalışan Stratejiler” penceresine eklenir. Rapor takibini yapmak ve stratejiyi durdurmak için kolaylıklar sağlar.

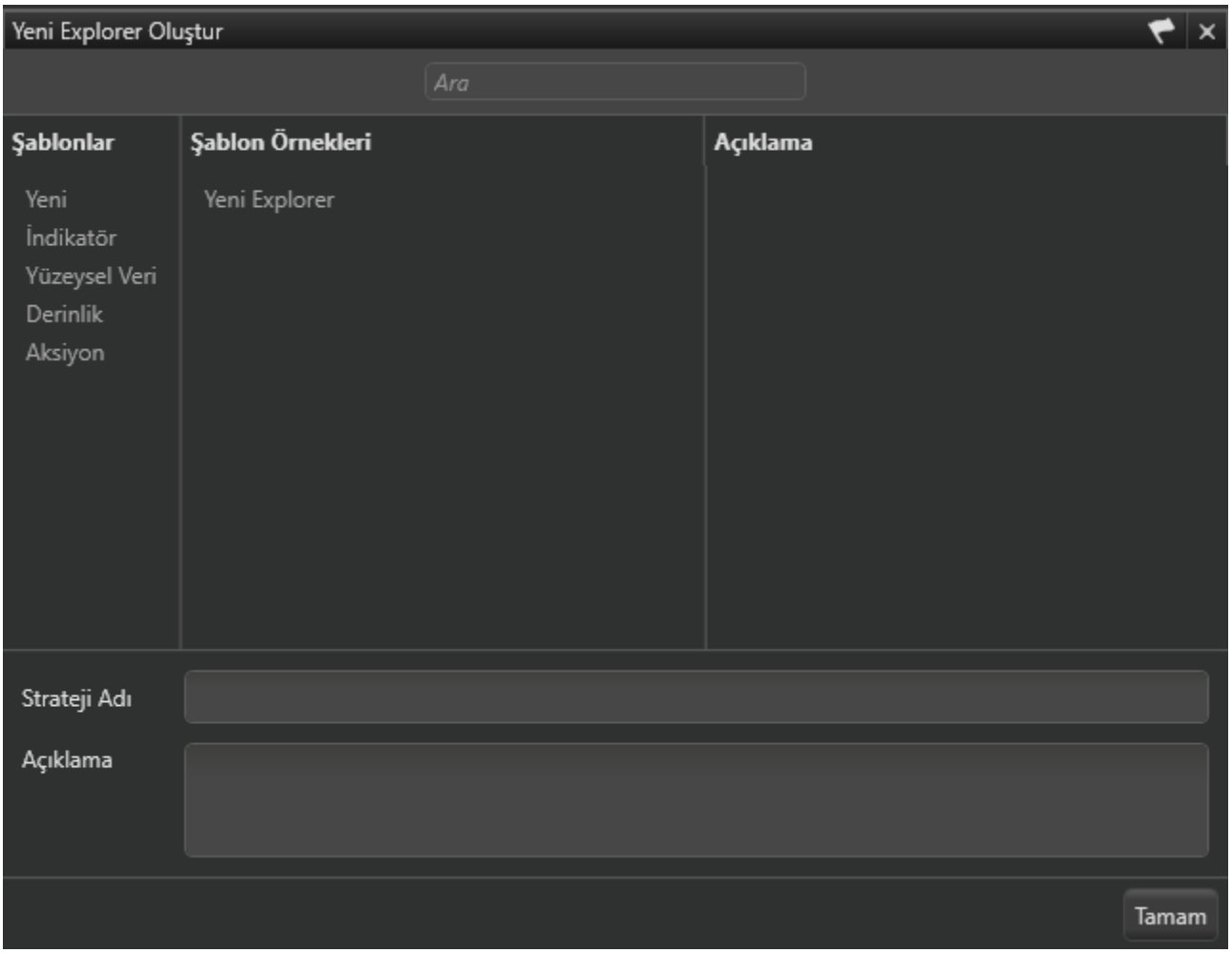


Backtest Sonuçları

Backtest yapılan stratejilerin geçmiş kayıtlarının tutulduğu bölümdür. Burada bulunan kayıtlardan backtest raporlarına ulaşabilir veya tekrar backteste tabii tutabilir.

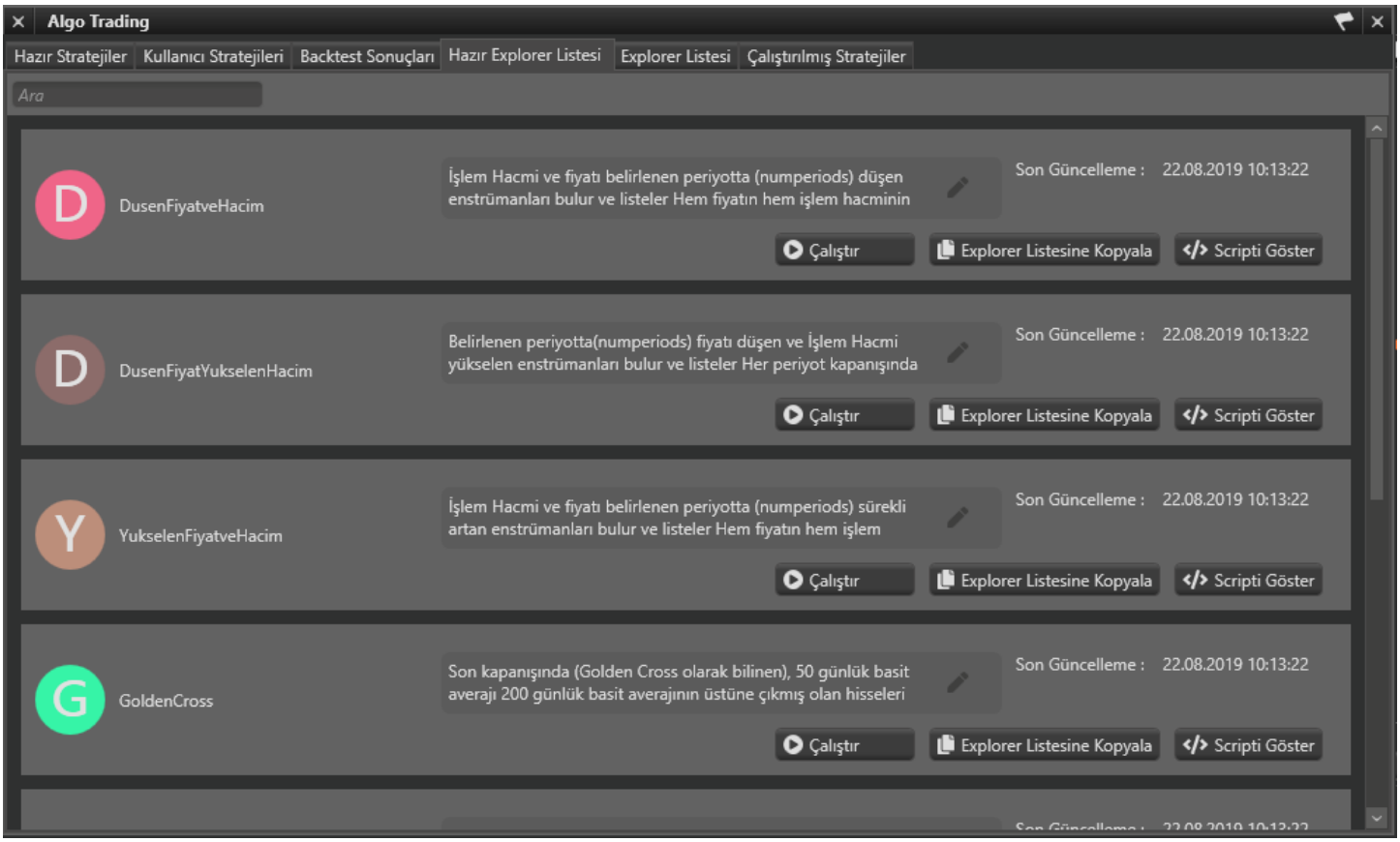
Yeni Explorer Oluştur

Kullanıcıların kendi explorerlarını oluşturmasına olanak sağlar. Burda oluşturulan explorerlar “Explorer Listesi” bölümüne kayıt edilir.



Hazır Explorer Listesi

Bu alanda uygulama ile birlikte gelen hazır explorer'lar bulunur. Hazır explorer içersinde bulunan örnek explorer'ları kullanıcılar ister direkt isterlerse üzerinde değişikler yapıp kullanabilir. "Kullanıcı Stratejilerine Kopyala" butonuna basılarak burada bulunan stratejilerden herhangi birini "Explorer Listesi" bölümüne eklenebilir ve üzerinde değişiklik yapılabilir.



Çalıştırılmış Stratejiler

Çalıştırılan tüm stratejilerin kaydının tutulduğu bölümdür. Geçmiş stratejilerin rapor kaydına ulaşımını veya tekrar çalıştırılabilmesini sağlar.

Algo Trader Ayarları

Bu bölümde AlgoTrader'ın çalışma şekline dair ayarlar bulunmaktadır.

Uygulama tarafından üretilen bildirimlere izin ver: Bu seçenek kapatılırsa AlgoTrader'da oluşan, örneğin emir gönderimi bildirimleri, ekrana yansımayacaktır. Alarm, hisse bazlı hızlı seviye düşüş/artış, pair trading gibi MatriksIQ'dan gelen uyarılar iletilmeye devam edecektir.

Strateji editöründeki kodun dakikada bir otomatik kayıt edilmesine izin ver: Strateji editöründe açtığımız stratejilerin otomatik olarak kaydedilmesini etkinleştirir. **Bu seçenek kapatıldığında strateji sadece derlendiğinde kayıt edilecektir. Derlenmeden kapatılan stratejilerde bulunan kayıt edilmemiş değişiklikler silinir.**

Kullanılacak Çekirdek Sayısı: Backtest Optimizasyonu tarafından kullanılan CPU çekirdek sayısı bu seçenek kullanılarak değiştirilebilir. Bilgisayar kaynaklarının seçilen düzeyde kullanılmasını sağlar.

Çekirdek ayarları sadece AlgoTrader Backtest Optimizasyonu tarafından kullanılır, AlgoTrader ya da MatriksIQ'nun diğer process'leri için işletim sisteminiz tarafından belirlenen (varsayılan – bütün çekirdekler) çekirdek sayısı kullanılmaya devam edecektir.

Tamamı(varsayılan), Toplam çekirdek sayısı-1(1 çekirdeği kullanmaz, serbest bırakır), Toplam çekirdek sayısının yarısı(çekirdeklerin yarısını serbest bırakır), Tek Çekirdek (sadece 1 çekirdeği AlgoTrader kullanımı için ayırır)

B. C# Modülü

MatriksIQ AlgoTrader'ın C# modülünde stratejiler, okunabilirliği yüksek ve kolayca düzenlenebilir şekilde bulunmaktadır.

Yeni strateji oluşturma

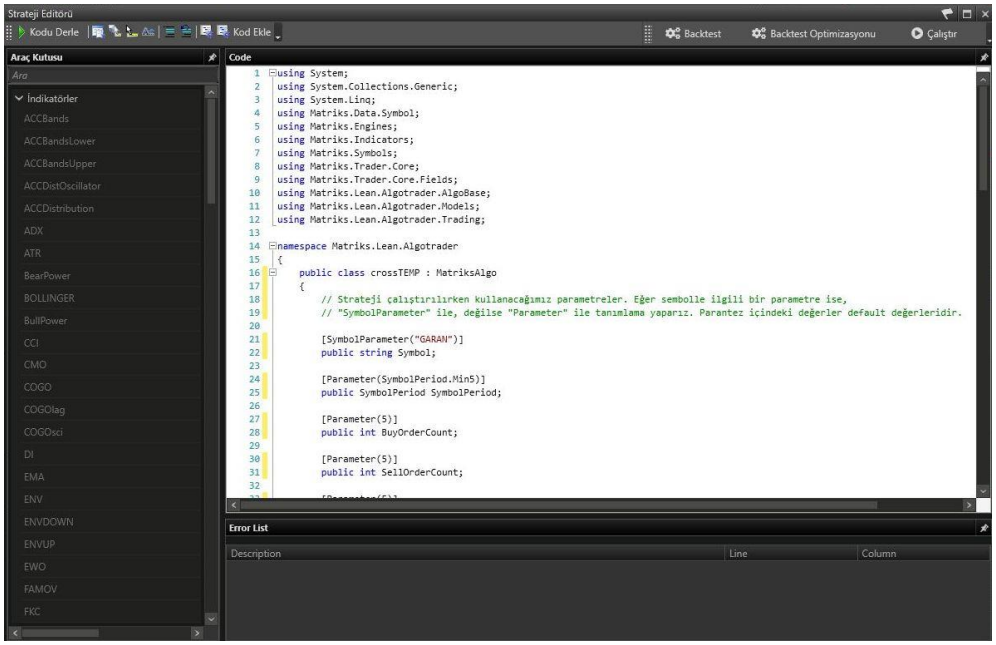
Menüden AlgoTrader->Yeni Strateji Oluştur->Yeni->Yeni seçilip, Strateji Adı yazıldıktan sonra Tamam butonuna tıklanarak boş bir şablon açılabilir.

Şablon bazlı strateji oluşturma

Eğer boş bir strateji değil de, örneğin, indikatör bazlı bir strateji kullanılmak isteniyorsa, solda Şablonlar menüsü altından seçilebilecek, İndikatör, Yüzeysel Veri, Derinlik gibi seçenekler bulunmaktadır. Örnek olarak, MACD indikatörü kullanılarak bir strateji yazılmak isteniyorsa, Şablonlar altında indikatör seçilip stratejiye isim verilip Tamam butonuna basıldıktan sonra, içerisinde MACD indikatörü tanımlanmış, çalışmaya hazır bir şablon MACD stratejisi açılacaktır. Bu tür stratejilerin örnekleri AlgoTrader menüsü Hazır Stratejiler altında da bulunmakta, aynı zamanda bazıları şablon olarak da Yeni Strateji Oluştur sekmesinden açılabilmektedir.

Örnek bir stratejinin kodunu görüntüleme

Menüden AlgoTrader->Hazır Stratejiler->Kullanıcı Stratejilerine Kopyala butonuna tıklayıp, uygun bir strateji ismi yazın. Kopyalanmış stratejiyi Kullanıcı Stratejileri sekmesinde bulabilirsiniz. Düzenle butonuna tıklandığında Strateji Editörü açılacaktır.



Parametre Tanımları

C# yapısı içerisinde **public class isim tanımı**ndan sonra, parametre tanımları bulunmaktadır. Backtest ve backtest optimizasyonunda, stratejinin canlı çalıştırılmasındabu parametreler kullanılmaktadır. Soldaki indikatörler menüsünden eklenen indikatörlerin tanımları da bu bölümde yapılmaktadır.

OnInit()

OnInit() bölümünde parametrelerde tanımlanan ve eklendiyse soldaki indikatörlerin fonksiyon tanımlamaları bulunmaktadır.

Bu bölümde ayrıca, stratejinin çalışma koşullarını **bütünüyle etkileyecek önemli fonksiyonlar** bulunmakta ve eklenebilmektedir:

AddSymbol(Symbol, SymbolPeriod): parametreler kısmında tanımlanan, Symbol ve SymbolPeriod değerlerini alır. Varsayılan şablon ile ya da yeni strateji oluşturulduğunda halihazırda varsayılan olarak eklenmektedir ve stratejinin belirlenen bir sembol ile çalışması için gereklidir.

AddSymbolMarketData(Symbol): parametreler kısmında tanımlanan, Symbol tanımlamasını alır. Klasik fiyat ekranı penceresinde kolon seçimi menüsünden ekleyebileceğimiz, ağırlıklı ortalama marjı, alış satış, yüksek düşük, açılış kapanış, temel/teknik analiz bilgileri gibi (örn. Pivot, direnç, net dönem karı, amortisman, günlük/7 günlük ağırlıklı ortalama vb.) değerleri stratejimiz içinde kullanılabilmesini sağlar. ***Bu fonksiyon eklendikten sonra backtest ve backtest optimizasyonu yapılamaz.***

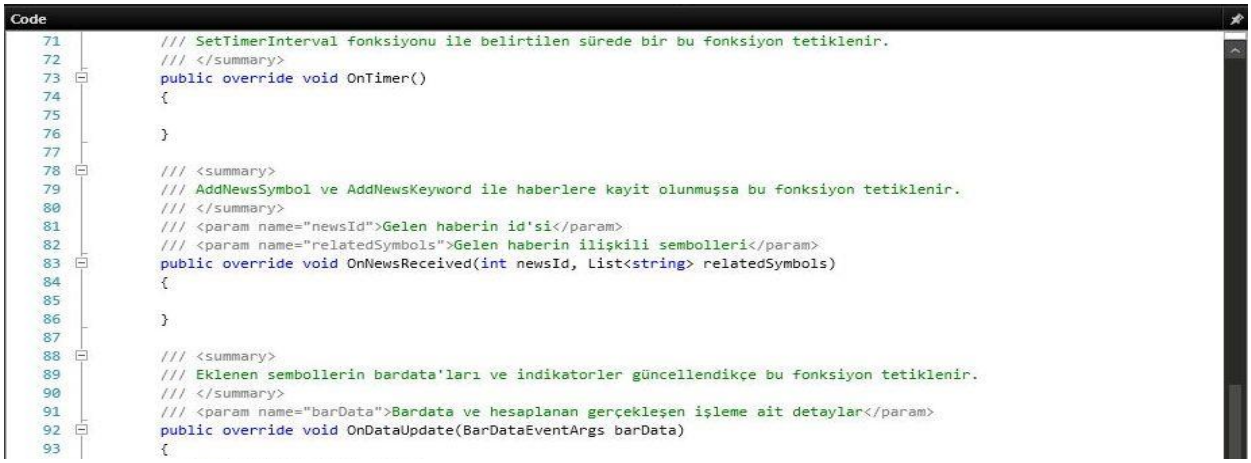
AddSymbolMarketDepth(Symbol): parametreler kısmında tanımlanan, Symbol tanımlamasını alır. Stratejimizde derinlik datası kullanabilmemizi sağlar. ***Bu fonksiyon eklendikten sonra backtest ve backtest optimizasyonu yapılamaz.***

WorkWithPermanentSignal(): true/false değer alır. Algoritmanın kalıcı veya geçici sinyal ile çalışıp çalışmayacağını belirleyen fonksiyondur. WorkWithPermanentSignal(true) şeklinde belirlenirse algoritma sadece yeni bar açılışlarında çalışır. Bu fonksiyonu çağırmazsak veya WorkWithPermanentSignal(false) olarak belirlersek algoritma her işlem olduğunda tetiklenir.

SendOrderSequential(): true/false değer alır. SendOrderSequential(true) yazılırsa, emirler önce al, sonra sat şeklinde dizin halinde ilerler. Yani sat emri gerçekleştikten sonra algoritma al emri gelene kadar oluşan koşulları hesaba katmaz. Aynı şekilde al emri gerçekleştikten sonra algoritma sat emrini bekler. Bu arada bir al emri koşulu daha tetiklense bile bu emir gönderilmez. Bu satırı silerek veya false geçerek emirlerin sırayla gönderilmesi engellenebilir.

SetTimerInterval(): sayısal değer alır. Parametre olarak verilen saniyede bir OnTimer fonksiyonu tetiklenir. Örneğin SetTimerInterval(3); olarak yazılırsa 3 saniyede bir OnTimer() fonksiyonu tetiklenecektir. Dolayısıyla bu

fonksiyon açık olduğu takdirde, belirtilen zamanlarda tetiklenmesini istediğimiz kod/stratejiyi, aşağıdaki örnek resimde gördüğümüz gibi, public override void OnTimer() {} kod sekmesinin içerisine yazmamız gerekmektedir.



AddNewsSymbol(Symbol): parametreler kısmında tanımlanan, Symbol tanımlamasını alır. Bu fonksiyon ile tanımlanan sembol ile ilgili haber geldiğinde OnNewsReceived fonksiyonu tetiklenir. OnTimer fonksiyonunun tetiklenme mekanizmasıyla aynıdır.

AddNewsKeyword("KAP"): Bu fonksiyon ile tanımlanan anahtar kelime ile ilgili haber geldiğinde OnNewsReceived fonksiyonu tetiklenir.

OnInitCompleted(): OnInit() fonksiyonu tamamlandığında çalışacak fonksiyondur. Tek kere çalıştığı için, üzerinde işlem yapılıp daha sonra program boyunca sabit olarak kullanılacak öğeler için uygundur. Mesela Machine Language kullanırken data hazırlama ve eğitim fonksiyonları,Train() ve PrepareData(), bu bölümde çalıştırılır.

OnDataUpdate(BarDataEventArgs barData) bölümünde stratejinin asıl mantıksal kısmı yer almaktadır. Her bar data açılışında tetiklenmektedir. Hangi durumlarda stratejinin aktif olacağı, alış/satış koşullarının oluştuğunun değerlendirilmesi bu bölümde yapılmaktadır.

BarDataUpdate her tetiklendiğinde barData ismindeki obje güncellenmektedir. Bu objede metotlar ile erişebileceğimiz bir çok kullanışlı öğe bulunmaktadır.

barData: BarData, BarDataIndex, IsNewBar, LastPrice, LastQuantity, LastTickTime, PeriodIndo, SymbolId, Tuple
barData.BarData: BarType, Open, Close, Diff, DiffPercent, Dtime, High, Low gibi bar hakkında bilgiler içerir. Aşağıda daha detaylı açıklanmıştır.
barData.BarDataIndex: Bar'ın sayısını/kaçıncı bar olduğunu döner. Bar sayımı ilk bar (yani en eski bar) sıfır'dan başlayarak ilerler.
barData.IsNewBar: Bar'ın yeni açılıp açılmadığına bakar. Yeni bar açılışında true, daha sonraki bar güncellemelerinde false döner.
barData.LastPrice: Son işlem fiyatı (backtestte bu değer gelmez)
barData.LastQuantity: Son işlem miktarı (backtestte bu değer gelmez)
barData.LastTickTime: Son işlem zamanı (backtestte bu değer gelmez)
barData.PeriodIndo: Bardata'da kullanılan periyot
barData.SymbolId: Bardata'sı gelen sembol/enstrümanın mevcut atanmış unique sembol id'si
barData.SymbolBarInfo: Sembol ve periyot için kullanılan değişken
barData.BarData.Open: Barın açılış değeri
barData.BarData.Close: Barın kapanış değeri (son/canlı bar datasında ise enstrümanın güncel fiyatı)
barData.BarData.Diff: Bar içerisinde oluşan fiyat farkı
barData.BarData.DiffPercent: Bar içerisinde oluşan yüzde fiyat farkı
barData.BarData.Dtime: gg.aa.yyyy ss:dd:ss formatında tarih ve zaman
barData.BarData.High: Barda oluşmuş en yüksek değer
barData.BarData.Low: Barda oluşmuş en düşük değer
Bardata.Volume: Barda oluşmuş hacim
Bardata.WClose: Barın ağırlıklı ortalama değer

OnOrderUpdate(IOrder order) emir güncellemelerini alan fonksiyondur. Emirlerin durumu değiştiğinde (düzeltme, iptal, gerçekleşme, parçalı gerçekleşme) bu fonksiyona düşer.

Emirlerin durumunu gösteren önemli fonksiyon OrdStatus ve metotlarıdır:

OrdStatus.New: Yeni Emir
OrdStatus.PartiallyFilled: Parçalı gerçekleşme
OrdStatus.Filled: Gerçekleşmiş Emir
OrdStatus.Canceled: İptal
OrdStatus.PendingCancel: İptal Bekliyor;
OrdStatus.Rejected: Reddedilen emir
OrdStatus.PendingNew: İletilmeyi bekleyen emir
OrdStatus.Expired: Süresi dolmuş tarihli Emir
OrdStatus.PendingReplace: Düzeltilmeyi bekleyen emir
OrdStatus.Replaced: Düzeltilen emir
OrdStatus.PendingCancelreplace: İptal bekleyen emir

OnOrderUpdate fonksiyonu içerisinde kullanılabilecek diğer öğeler:

string CliOrdID: Kullanıcı tanımlı emir id'si
DateTime TradeDate: Emir zamanı
string Account: Emir gönderilen hesap
Side Side: Emir yönü
TimeSpan TransactTime: Gerçekleşme zamanı
OrdType OrdType: Emir Tipi
TransactionType TransactionType: İşlem tipi (Normal emir, açığa satış vs)
decimal Price: Emir fiyatı
decimal StopPx: Şart fiyatı
TimeInForce TimeInForce: Geçerlilik süresi tipi
DateTime ExpireDate: Tarihli emirde son geçerli olduğu zaman
string Symbol: Emir gönderilen sembol
decimal OrderQty: Emir miktarı
decimal Amount: Emir tutarı

decimal FilledQty: Gerçekleşen toplam miktar
decimal FilledAmount: Gerçekleşen toplam tutar
string OrderID: Emir Id
OrdStatus OrdStatus: Emrin durumu (Yukarıda metotları ayrıcana açıklanmıştır)
OrdRejReason OrdRejReason: Emir iptal sebebi
decimal LastQty: Son gerçekleşme miktarı
decimal LastPx: Son gerçekleşme fiyatı
decimal LeavesQty: Gerçekleşmeyen kalan miktar
decimal AvgPx: Gerçekleşen emirlerin ortalama fiyatı
DateTime BarDateTime: Emrin iletildiği bardata zamanı
decimal SignalPrice: Emrin iletildiği bar fiyatı

OnDataUpdate(BarDataCurrentValues barDataCurrentValues):

(OnDataUpdate() ile aynı işlevi görmekte, fakat farklı kullanımları vardır, alternatif olarak yazılmıştır) İçerisine stratejinin asıl mantıksal kısmı yazılır. Her bar açılışında tetiklenmektedir. Hangi durumlarda stratejinin aktif olacağı, alış/satış koşullarının oluştuğunun değerlendirilmesi bu bölümde yapılmaktadır.

BarDataUpdate her tetiklendiğinde barDataCurrentValues ismindeki obje güncellenmektedir. Bu objede metotlar ile erişebileceğimiz birçok kullanışlı öğe bulunmaktadır.

barDataCurrentValues.barDataValues: Tanımlanmış tüm sembollere dair en güncel verileri içeren bir liste döner. Bu liste BarDataValue tipindeki objelerden oluşmaktadır.
***barDataCurrentValues*.LastUpdate:** Tanımlanmış semboller arasından en son güncellenen sembole ait verileri barındırmaktadır. Veriyi BarDataValue tipindeki bir obje olarak döner.
barDataCurrentValues.GetLastUpdateForSymbol(Symbol, SymbolPeriod): İstenilen sembol ve periyot için güncel veriyi BarDataValue tipindeki bir obje olarak elde etmek için kullanılır.

Yukarıdaki yöntemlerle elde edilen BarDataValue objeleri aşağıdaki öğeleri içermektedir.

SymbolName: String olarak sembol ismi.
SymbolPeriod: Sembol için kullanılan barların periyodu.
PeriodInfo: Sembol için kullanılan barların PeriodInfo class formatındaki periyodu.

SymbolDefinition: Sembol tanımı.
SymbolId: Sembole ait integer ID.
BarDataIndex: Sembolün son update değerinin indeksi.
LastTickTime: Sembol verisinin son güncellenme zamanı.
Open: Bar açılış fiyatı
High: Bar kapanış fiyatı
Low: Barın en düşük değeri.
Close: Barın en yüksek değeri.
Diff: Güncel fiyat ile bir önceki kapanış arasındaki fark.
DiffPercent: Güncel fiyat ile bir önceki kapanış arasındaki yüzdesel fark.
WClose: Ağırlıklı ortalama
Volume: Hacim
LastQuantity: Son işlem hacmi
LastPrice: Son fiyat
IsNewBar: Bar açılışlarını belirlemek için boolean flag. Bar yeni açılmış ise true, açık bara ait veri güncellenmişse false döner.
IsLastDataUpdate: Son veri güncellemesinin bu sembole ait olup olmadığını belirten flag. OnDataUpdate bu sembole gelen bir güncelleme tarafından tetiklenmişse true, aksi halde false olacaktır.

Örnek kullanımlar:
barDataCurrentValues.LastUpdate.SymbolName son güncellenen sembolün ismini döner.

barDataCurrentValues.LastUpdate.Close son güncellenen sembolün fiyatını döner.

barDataCurrentValues.GetLastUpdateForSymbol(SymbolName, SymbolPeriod).Close ismi ve periyodu verilen sembol için güncel bara ait kapanış değerini verir.

barDataCurrentValues.GetLastUpdateForSymbol(SymbolName, SymbolPeriod).IsLastUpdate ile OnDataUpdate'i tetikleyen veri güncellemesinin ismi ve periyodu verilen sembole mi, yoksa başka bir sembole mi ait olduğu öğrenilebilir.

3. Fonksiyonlar

A. Matematiksel Fonksiyonlar

MatriksIQ'da C# Math kütüphanesi fonksiyonları, kütüphane çağırılarak rahatlıkla çalıştırılabilmektedir (Abs, Acos, Asin, Atan, Atan2, BigMul, Ceiling, Cos, Cosh, DivRem, Exp, Floor, IEEERemainder, Log, Log10, Max, Min, Pow, Round, Sign, Sin, Sinh, Sqrt, Tan, Tanh, Truncate).

Örnek: Math.Round(volumeTL.CurrentValue,0), güncel işlem hacmini birler basamağına yuvarlanmış olarak dönecektir.

Bunun dışında direkt olarak MatriksIQ'dan çağırılabilecek fonksiyonlar aşağıda listelenmiştir.

Absolute(data): Bir sayı ya da değerin mutlak değerini alır.

Maximum(data1,data2): İki data arasından en yüksek olanı döner.

Minimum(data1,data2): İki data arasından en düşük olanı döner.

Power(data, power): Verilen ilk değerin 2. Değer miktarında(power) üssünü döner. Örneğin Power(X,2): x değerinin karesini, Power(X,3): küpünü döner.

B. Genel Fonksiyonlar

AddChart(String, Int32): Kullanıcı tanımlı grafik eklemek için kullanılır.

Örnek: AddChart("ChartName",2), fonksiyona verdiğimiz ilk değer grafiğin adını, ikinci değer ise grafiğe kaç veri ekleneceğini belirler. Örneğin grafiğe iki indikatörü birlikte çizdirmek istenir ve bu iki indikatörlerin de ikişer tane göstergesi varsa fonksiyona yazılması değer "4" 'tür.

AddChartLineName(String, Int32, String): Kullanıcı tanımlı grafikteki fiyat bandlarını isimlendirmek için kullanılır.

Örnek: AddChartLineName("ChartName", 1, "Most "), fonksiyonundaki ilk değer "AddChart" fonksiyonundaki grafiğin adı ile aynı olmalıdır. İkinci değer ise isimlendireceğimiz verinin indeksini belirtir. "Most" ise çizdirilecek olan bandın isimlendirilmesine yarar.

AddColumns(int columnCount): Explorer'da eklenmesini istediğimiz kolon sayısını belirtmek için kullanılır. Trading stratejilerinde kullanımı yoktur.

AddNewsSymbolKeyword(String, List< String>): Haber filtresi için sembol ve anahtar kelime grubu eklenir. Bunların hepsi aynı anda gerçekleştiğinde ilgili fonksiyon tetiklenir

Alert(string Data): Masaüstünde alarm göstermek için kullanılır.

CrossAbove(IIndicator, IIndicator): Birinci indikatörün ikinci indikatörü yukarı yönde kırıp kırmadığının kontrolünü yapar.

CrossAbove(IIndicator, Int32): İndikatörün verilen sayısal değeri yukarı yönde kırıp kırmadığının kontrolünü yapar.

CrossBelow(IIndicator, IIndicator): Birinci indikatörün ikinci indikatörü aşağı yönde kırıp kırmadığının kontrolünü yapar.

CrossBelow(IIndicator, Int32): İndikatörün verilen sayısal değeri aşağı yönde kırıp kırmadığının kontrolünü yapar.

Cumulate(IIndicator): İndikatör serisinin toplamını alır.

Cumulate(IIndicator, Int32): İndikatör serisi adedince, sayısal kümülatif toplam değerini bulur.

Cumulate(ISymbolBarData, OHLCType): Bar verilerinin ohlc tipine göre toplamını alır.

Cumulate(ISymbolBarData, Int32): Bar verileri adedince, sayısal kümülatif toplam değerini bulur.

DayOfMonth(BarData barData): İlgili barın ayını verir.

DayOfWeek(BarData barData): İlgili barın haftasını verir.

Debug(String): Debug ekranına log yazdırmak için kullanılır.

Decreasing(Int32, OHLCType, Boolean): Ana sembol için seçilen aralıkta hep düşen mi olup olmadığı kontrolünü yapar. **Örnek:** *Decreasing(8, OHLCType.Close, true)*, 8 bar içerisinde kapanış değerinin (son yazılan boolean değer true olduğunda, güncel barda değil bir önceki bardan itibaren saymaya başlar) sürekli olarak (her bar üst üste) düşüp düşmediğine bakar. Düşüyorsa true, düşmüyorsa false döner.

Decreasing(IIndicator, Int32, Int32, Boolean): Verilen indikatör için seçilen aralıkta hep yükselen mi olup olmadığı kontrolünü yapar.

Decreasing(SymbolDef, Int32, OHLCType, Boolean): Verilen sembol için seçilen aralıkta hep düşen mi olup olmadığı kontrolünü yapar.

GetBarData(): Default sembol için olan bar datayı döner.

GetBarData(SymbolDef): Parametrelerde kayıt olunan bar dataya erişmek için kullanılır.

GetMarketData(string Symbol, SymbolUpdateField symbolUpdateField): Kayıt olunan yüzeysel veriye erişmek için kullanılır.

GetMarketDepth(string Symbol): Kayıt olunan derinlik verisine erişmek için kullanılır.

GetOverall(): Strateji içinde hesaplanan overall bilgisini döner.

GetSessionTimes(string symbolName): Enstrümanın seans zamanlarını döner.

GetSymbolDef(String, IPeriodInfo): Sembol adı ve periyot bilgisi ile SymbolDef getirir.

GetSymbolId(string Symbol): Sembole atanmış id numarasını döner.

GetSymbolName(int SymbolId): Sembol id'sinden sembol adına erişmek için kullanılır.

Highest(ISymbolBarData, OHLCType): Belirtilen enstrümanın elimizdeki data içerisindeki en yüksek değerini döner.

HighestHigh(OHLCType, Int32): Belirtilen periyotta data serisinin en yüksek değerini hesaplar. (HHV)

Hour(BarData barData): İlgili barın saatini verir.

Increasing(Int32, OHLCType, Boolean): Ana sembol için seçilen aralıkta hep yükselip yükselmediğini kontrol eder.

Örnek: *Increasing(5, OHLCType.Close, true)*, 5 bar içerisinde kapanış değerinin (son yazılan boolean değer true olduğunda, güncel barda değil bir önceki bardan itibaren saymaya başlar) sürekli olarak (her bar üst üste) yükselip yükselmediğine bakar. Yükseliyorsa true, yükselmiyorsa false döner.

Increasing(IIndicator, Int32, Int32, Boolean): Verilen indikatör için seçilen aralıkta hep yükselen mi olup olmadığı kontrolünü yapar.

Increasing(SymbolDef, Int32, OHLCType, Boolean): Verilen sembol için seçilen aralıkta hep yükselen mi olup olmadığı kontrolünü yapar.

LastValue(ISymbolBarData, OHLCType): İndikatör veya bar datanın backtestte bulunduğu değer yerine, en son değeri döner.

Lowest(ISymbolBarData, OHLCType): Belirtilen enstrümanın elimizdeki data içerisindeki en düşük değerini döner.

LowestLow(OHLCType, Int32): Belirtilen periyotta data serisinin en düşük değerini hesaplar. (LLV)

Minute(BarData barData): İlgili barın dakikasını verir.

Month(BarData barData): İlgili barın ayını verir.

Plot(String, Int32, Decimal): Grafiğe data eklemek için kullanılır. Backtest'te en son tetiklenen bardata zamanını kullanır. Canlı stratejide en son tick data zamanını kullanır. Bu fonksiyonu kullanmak için AddChart fonksiyonu ile grafik eklenmelidir. Parantez içindeki 2. değer olarak (Int32), eğer plot ettiğimiz değer ya da indikatörün birden fazla değeri(çizgisi) varsa, bunu seçmekte kullanılır, varsayılanı 0'dır.

Örnek: Plot("ChartName", 1, most.CurrentValue), ilk değer addcharta oluşturduğumuz grafiğin adıdır. İkinci değer çizdirilecek verinin indeksini belirler. Son değerse

SendCancelOrder(string clOrdId): Istenilen emir için emir ID'si kullanılarak iptal emri gönderir.

SendLimitOrder(String, int Quantity, OrderSide, Decimal, TimeInForce, ChartIcon): Stratejiden limit alım/satım emri göndermek için kullanılır.

Örnek: *SendLimitOrder(Symbol, 100, (OrderSide.Buy), 11.88m, TimeInForce.GoodTillCancel, ChartIcon.Buy):* Parametrelerde belirttiğimiz enstrümandan (Symbol ismiyle tanımlanan. Buraya "GARAN" olarak hisse ismi de yazılabilir ama bu durumda strateji dışından değiştiremeyiz), 100 adet, 11.88 fiyatından, iptal edilene kadar aktif kalacak şekilde alım emri gönderir. ChartIcon.Buy grafikte alım emri olarak gözükmesini sağlayacaktır. Limit emrinin 11.88m olarak yazılma nedeni, burada fonksiyonun tanımında yazıldığı gibi decimal beklemesidir. Bu yüzden sadece 11.88 yazarsak bunu double alacağından error verecektir. Biz girdiğimiz değeri 11.88m olarak yazdığımızda, değer otomatik olarak decimal olarak tanımlanmaktadır.

SendMarketOrder(String, Int32, OrderSide, TimeInForce, ChartIcon): Stratejiden market(piyasa) alım/satım emri göndermek için kullanılır.

Örnek: *SendMarketOrder(Symbol, 100, (OrderSide.Buy)),ChartIcon.Buy):* Parametrelerde belirttiğimiz enstrümandan (Symbol ismiyle tanımlanan. Buraya "GARAN" olarak hisse ismi de yazılabilir ama bu durumda strateji dışından değiştiremeyiz), 100 adet, piyasa fiyatından, alım emri gönderir. ChartIcon.Buy grafikte alım emri olarak gözükmesini sağlayacaktır.

SendOrder(string symbol,int quantity,decimal price,Side side,OrdType ordType,TimeInForce timeInForce,TransactionType transactionType,ChartIcon chartIcon): Stratejiden kişiselleştirilmiş emir göndermek için kullanılır. Birçok ek parametre tanımlanabilmektedir. Daha serbest yazıldığından metotlara dikkat edilmesi gerekir. İçerisinde bulunan bazı fonksiyonları işlem yapmak istediğiniz borsa ya da kurum desteklemeyebilir.

Örnek: *SendOrder("ASELS", 100, 0, new Side(Side.Sell), new OrdType(OrdType.Market), new TimeInForce(TimeInForce.Day), new TransactionType(TransactionType.Normal), ChartIcon.Sell)*: 100 adet ASELS, için piyasa fiyatından, satış emri gönderir. TimeInForce.Day emrin gün sonuna kadar açık kalmasını sağlar. TransactionType shortdaily, virman, closeshort, credit gibi metotlar alabilmektedir, fakat bu metotların işlem yaptığınız borsa tarafından desteklenip desteklenmediğini bilmediğiniz taktirde, normal olarak tanımlanması tavsiye edilir. ChartIcon.Sell grafikte satış emri olarak gözükmesini sağlayacaktır. Emir piyasa emri olduğu için fonksiyon içerisinde 3. Parametre (decimal price), 0 (sıfır) olarak yazılmalıdır. OrdType.Limit olarak tanımlasaydık, sıfır yerine işlem yapmak istediğimiz limit fiyatın yazılması gerekirdi.

SendReplaceOrder(string clOrdId, int quantity): ClOrdId ile string olarak belirtilen emir, decimal olarak belirtilen fiyat ve Int32 olarak belirtilen miktar ile değiştirilir.

SendShortSaleLimitOrder(String, Int32, Decimal, TimeInForce): Stratejiden açığa satış limit emri göndermek için kullanılır.

Örnek: *SendShortSaleLimitOrder(Symbol,100,11.22m,new TimeInForce(TimeInForce.Day)):* Parametrelerde belirttiğimiz enstrümandan (Symbol ismiyle tanımlanan. Buraya "GARAN" olarak hisse ismi de yazılabilir ama bu durumda strateji dışından değiştiremeyiz), 100 adet, 11.22 fiyatından, gün sonuna kadar aktif olacak şekilde açığa satış emri iletir.

SendShortSaleMarketOrder(String, Int32, TimeInForce): Stratejiden market(piyasa) açığa satış emri göndermek için kullanılır.

Örnek: *SendShortSaleMarketOrder(Symbol,100,new TimeInForce(TimeInForce.Day)):* Parametrelerde belirttiğimiz enstrümandan (Symbol ismiyle tanımlanan. Buraya "GARAN" olarak hisse ismi de yazılabilir ama bu durumda strateji dışından değiştiremeyiz), 100 adet, piyasa fiyatından, açığa satış emri gönderir.

SetColumn(int column, object value): Explorer'da ilgili kolonun değerini, explorer çalıştırıldıktan sonra çıkan, sonuçlar ve filtrelenenler sayfalarına basar.

Örnek: *SetColumn(0, Math.Round(mov.CurrentValue, 2)):* İlk yazdığımız değer (0) datanın hangi kolona yazılacağını belirler (bu durumda ilk kolon). İkinci parametre ise (Math.Round(mov.CurrentValue, 2)), mov'a atadığımız (indikatör, fonksiyon vs.) değerin son değerini Math kütüphanesindeki Round fonksiyonunu kullanarak yuvarlar ve 2 ondalık basamaklı olarak basar.

SetColumnText(int column, object value): Explorer'da ilgili kolonun ismini, explorer çalıştırıldıktan sonra çıkan, sonuçlar ve filtrelenenler sayfalarına basar.

Örnek: *SetColumnText(1, "Mov1"):* İlk yazdığımız değer (1) datanın hangi kolona yazılacağını belirler (bu durumda ikinci kolon). İkinci parametre ise kolonun ismini belirler.

SetTimerInterval(int Second): Timer fonksiyonunun kaç saniyede bir tetikleneceği ayarlanır.

ToString(): Güncel objeyi string olarak döner.

Year(BarData barData): İlgili barın sadece yılını verir.

Sentetik Emir Tanımlama Fonksiyonları

StopLoss(string symbol, SyntheticOrderPriceType SyntheticOrderPriceType, decimal stopLevel): Girdiğiniz sembol için zarar durdur emri tanımlamakta kullanılır. Sembol başına bir adet zarar durdur emri tanımlanabilir. Önceden tanımlı bir zarar durdur emri varsa yenisiyle değiştirilir.

Emir tanımlandığında sembolün güncel fiyatına göre short ve long pozisyon için ayrı stop fiyatları belirlenir. Sembol fiyatı değiştikçe, pozisyonunuzun yönüne göre kullanılacak stop fiyatı seçilir.

Fiyat seçilen stop değerine ulaştığında, pozisyonunuz kadar ve zıt yönde bir market emri gönderilir.

TakeProfit(string symbol, SyntheticOrderPriceType SyntheticOrderPriceType, decimal stopLevel): Girdiğiniz sembol için kar al emri tanımlamakta kullanılır. Sembol başına bir adet kar al emri tanımlanabilir. Önceden tanımlı bir kar al emri varsa yenisiyle değiştirilir.

Emir tanımlandığında sembolün güncel fiyatına göre short ve long pozisyon için ayrı stop fiyatları belirlenir. Sembol fiyatı değiştikçe, pozisyonunuzun yönüne göre kullanılacak stop fiyatı seçilir.

Fiyat seçilen stop değerine ulaştığında, pozisyonunuz kadar ve zıt yönde bir market emri gönderilir.

TrailingStopLoss(string symbol, SyntheticOrderPriceType SyntheticOrderPriceType, decimal stopLevel): Girdiğiniz sembol için hareketli zarar durdur emri tanımlamakta kullanılır. Sembol başına bir adet zarar durdur emri tanımlanabilir. Önceden tanımlı bir zarar durdur emri varsa yenisiyle değiştirilir.

Emir tanımlandığında sembolün güncel fiyatına göre short ve long pozisyon için ayrı stop fiyatları belirlenir. Sembol fiyatı değiştikçe, pozisyonunuzun yönüne göre kullanılacak stop fiyatı seçilir.

Fiyat seçilen stop değerine ulaştığında, pozisyonunuz kadar ve zıt yönde bir market emri gönderilir. Fiyat değişimlerinde stop koşulu gerçekleşmemiş ise, stop fiyatları güncellenir.

Parametreler:

SyntheticOrderPriceType: Stop noktaları için fiyat hesaplama yöntemini seçmekte kullanılır. SyntheticOrderPriceType.Percent veya SyntheticOrderPriceType.PricePoint seçenekleri ile yöntem, yüzdesel fark veya fiyat farkı olarak seçilir.

stopLevel: Stop değerleri hesaplanırken kullanılacak değeri seçer. SyntheticOrderPriceType parametresi seçimine bağlı olarak yüzde fark veya fiyat farkı değeri olarak kullanılır.

Sentetik Emir İptal Fonksiyonları

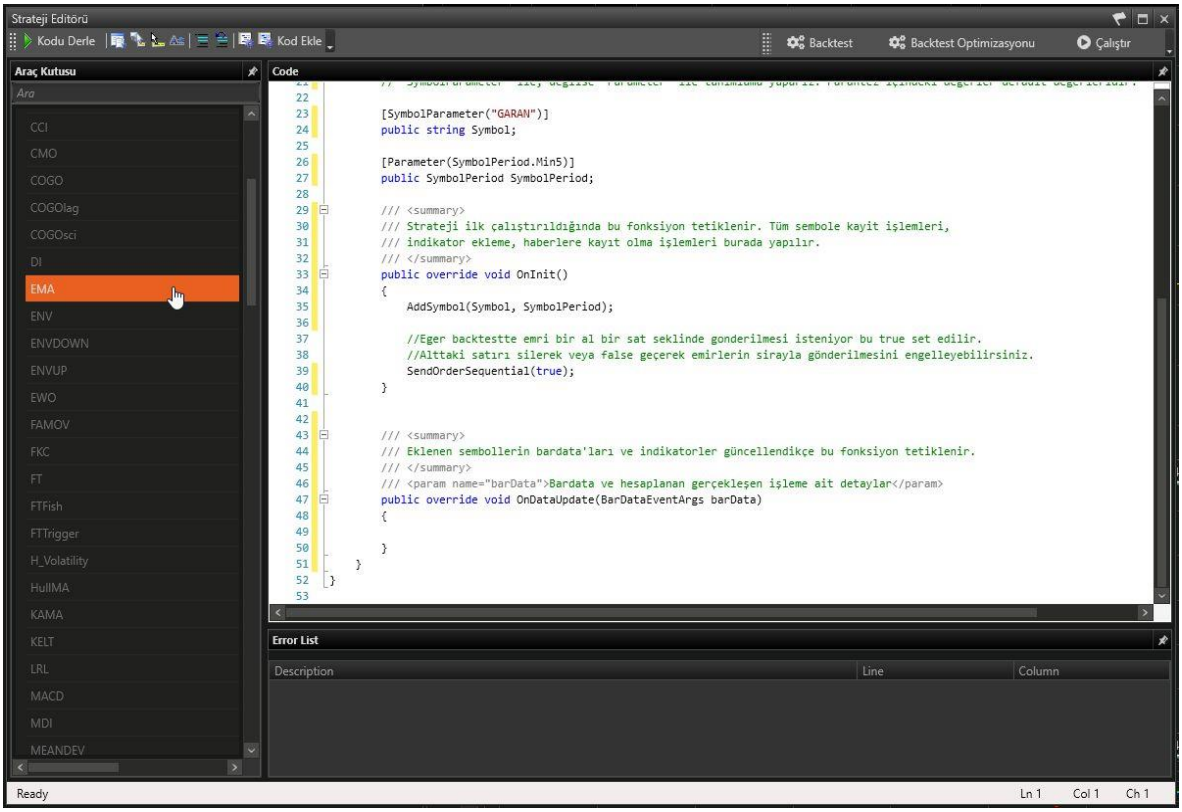
CancelStopLoss(string symbol): Sembol üzerine tanımlı zarar durdur emrini iptal eder.

CancelTakeProfit(string symbol): Sembol üzerine tanımlı kar al emrini iptal eder.

CancelTrailingStopLoss(string symbol): Sembol üzerine tanımlı hareketli zarar durdur emrini iptal eder.

C. İndikatörler

MatriksIQ Algotrader'da birçok indikatör hazır şekilde sol taraftaki araç menüsünde bulunmaktadır.



Bu indikatörlerden istenilene çift tıklandığında, otomatik olarak gerekli yerlerde o indikatörle ilgili tanımlar oluşturulmakta, indikatör stratejide kullanıma hazır hale gelmektedir.

Örneğin, EMA öğesine çift tıklandığında ilk kod bloğu içerisine ema isimli bir EMA öğesi ve OnInit() kod bloğu içerisine *ema = EMAIndicator(Symbol, SymbolPeriod, OHLCType.Close, 22);* satırı ile bu öğeye ait indikatör, varsayılan değerleriyle tanımlanmaktadır. Bu tanımlamalardan sonra OnDataUpdate kod bloğu içerisinde bu indikatör istenilen strateji uygulanarak kullanılabilir.

Her indikatörün genel çalışma prensibi ve stratejinin nasıl oluşturulabileceği dokümanın devamında anlatılmaktadır. MatriksIQ strateji editöründe IntelliSense de bulunduğundan, mevcut tanımlanmış indikatör objesinin ne metodlar alabileceğini CTRL+space'e basarak görebiliriz.

Örneğin yukarıda tanımlanmış ema objesi, ema.CurrentIndex, ema.CurrentValue, ema.LastBarIndex, ema.Period, ema.Value gibi bir çok metod alabilmektedir.

** ema.CurrentValue: EMA indikatörünün o andaki değerini döndürür.

** ema.CurrentIndex: EMA indikatörünün istenilen andaki indeksini döndürür.

** ema.LastBarIndex: EMA indikatörünün istenilen andaki indeksini döndürür.

** ema.Period: EMA indikatörünün periodunu döndürür.

ACCBandsIndicator(String, SymbolPeriod, OHLCType, Int32, Decimal)

(Acceleration bands) Hızlanma bantları göstergesinin değerini hesaplamak için kullanılır. Bollinger bantları gibi (genelde son 20 bar kullanır) basit averaj etrafında üst ve alt bant çizer. Enstrüman fiyatı bantların üstüne kırdığında, alım yapılarak yukarı doğru hızlanması, altına kırdığında da satış yapılarak aşağı doğru hızlanması beklenmektedir. Enstrüman fiyatı, tekrar bantların içine geldiğinde pozisyondan çıkılır.

Örnek:

```
if (CrossAbove(accBands.Lower, barData.BarData.Close))
{
    SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
    Debug("Alış emri verildi.");
}
if (CrossBelow(accBands.Upper, barData.BarData.Close))
{
    SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
    Debug("Satış emri verildi.");
}
```

accBands. Lower: Acceleration bands indikatörünün o andaki alt bant değeri**
accBands. Upper: Acceleration bands indikatörünün o andaki üst bandı değeri**

AccumulationDistributionIndicator(String, SymbolPeriod, OHLCType)

Accumulation/distribution hacim ve fiyatı kümülatif olarak kullanarak, enstrümanın toplamda satım ya da alım altında olduğunu belirlemeye çalışan bir indikatördür. Eğer hisse fiyatı düşerken ADI yükseliyorsa, bu hissenin aslında toplanma durumunda olduğunu gösterdiğinden fiyatta da bir süre sonra yukarı hareket olacağı beklenmektedir.

Bu indikatör tek başına teknik bir strateji belirlemede yeterli değildir. O nedenle diğer indikatörlerle birlikte kullanılır.

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

AccumulationDistributionOscillatorIndicator(String, SymbolPeriod, OHLCType, Int32, Int32)/ (IIndicator, Int32, Int32)

Chaikin Accumulation/Distribution Oscillator göstergesinin değerini hesaplamak için kullanılır. Fiyat hareketleri ve işlem hacmini birlikte analiz ederek yön hakkında fikir vermeyi hedefler.

A/D Oscillator bir volume göstergesidir. Yürütülen mantık, fiyat hareketleri ile hacmin uyumuyla alakalıdır. Hacimdeki artışla fiyatlardaki yükselmenin ya da hacimdeki düşüşle fiyatlardaki düşüşün uyumlu hareket edeceği, aksi durumlarda ise trend değişikliklerinin meydana geldiği varsayılmaktadır. Göstergenin

yorumlanmasında en yaygın yol, göstergenin yükselmesinin senette alım yapıldığı, düşmesinin ise senetten çıkıldığı şeklinde yorumlanmasıdır. Aynı şekilde, fiyatlar yükselirken bunun hacimle de desteklenmesi gerektiği varsayımından hareketle, Chaikin Oscillator göstergesinin yükselen fiyatlara aynı şekilde yükselerek cevap verememesi yükseliş trendinin sonlanabileceğine, düşen fiyatlara aynı şekilde düşerek cevap verememesi ise düşüş trendinin sonlanabileceğine işaret etmektedir.

Örnek:

```
if (temp != null)
{
    if (temp<accDistOscillator.CurrentValue)
    {
        SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
        Debug("Alış emri verildi.");
    }
    if (temp>accDistOscillator.CurrentValue)
    {
         SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
         Debug("Satış emri verildi.");
    }
    temp = accDistOscillator.CurrentValue;
}
else
{
     temp = accDistOscillator.CurrentValue;
}
```

accDistOscillator. CurrentValue: Accumulation/Distribution Oscillator indikatörünün o andaki değeri**

ADXIndicator(String, SymbolPeriod, OHLCType, Int32)

Average Directional Movement Index göstergesinin değerini hesaplamak için kullanılır. J.W.Wilder tarafından geliştirilen Directional Movement, fiyatların hangi yönde hareket etme eğiliminde olduğunu araştırır. Trend belirleme ve yanlış sinyallerin filtre edilebilmesi açısından oldukça önemli bilgiler sunan Directional Movement, ADX, ADXR, DIS, DX ve DI+DI- gibi birçok göstergenin de çıkış noktasıdır.

Hesaplanması kolay ama oldukça uzun olan Average Directional Movement Index (ADX) için öncelikle DM değerinin hesaplanması gerekmektedir. Bu değer, hareketin yönü yukarı ise pozitif aşağı ise negatif olacaktır.

+DM ve -DM değerlerini bulduktan sonra ise bunların toplamlarını yine bunların farklarına bölerek DX değerine ulaşırız. Oldukça hızlı hareket eden DX göstergesinin 14 günlük hareketli ortalamasının alınarak yumuşatılmasıyla da ADX eğrisini elde etmiş oluruz. Average Directional Movement Index (ADX), 0 ile 100 arasında dolaşan bir göstergedir. Diğer birçok göstergeden farklı olarak ADX, alım satım sinyalleri üretmekten çok, bir trendin var olup olmadığı ve gücüyle ilgili bilgiler verir. ADX'in sıfıra yakın değerleri trendin olmadığına ve kararsız bir piyasayı göstermektedir. Gösterge değerinin artmaya başlaması ise fiyat hareketinin olduğu yönde bir trendin varlığına işaret etmektedir.

Bu indikatör tek başına teknik bir strateji belirlemede yeterli değildir. O nedenle diğer indikatörlerle birlikte kullanılır.

ATRIndicator(String, SymbolPeriod, OHLCType, Int32): Average True Range göstergesinin değerini hesaplamak için kullanılır. Fiyat hareketliliğinin artış / azalışı konusunda fikir verir.

J. Welles Wilder tarafından geliştirilen Average True Range göstergesi hisse senetlerinin hareketliliği üzerine kurulmuştur. Bu yüzden vereceği bilgi de senedin hareket yönünden bağımsız olarak, fiyatların ne kadar değişkenlik göstermekte olduğudur.

Average True Range göstergesindeki yüksek değerler fiyat hareketliliğindeki artışa işaret etmektedir. Bu ise trend dönüşlerinin zamanlamasında kullanılmaktadır. Nispeten düşük değerler ise fiyatlardaki hareketliliğin azalmasıdır ve piyasadaki bir sıkışmayı göstermektedir.

Bu indikatör tek başına teknik bir strateji belirlemede yeterli değildir. O nedenle diğer indikatörlerle birlikte kullanılır.

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

BearPowerIndicator(String, SymbolPeriod, OHLCType, Int32, MovMethod)

Bears Power bir aksi yönde trend indikatörüdür. Fiyatın yönünü belirlemenize ve o trendin dönüş yapacağı noktayı bulmanıza yarar.

Bears indikatöründe çubuklar eksi tarafta iken yükselme görülmeye başladığında alış sinyali verir. Satış sinyali ise bunun tam tersidir.

Örnek:

```
decimal temp;

if ((temp != null))
{
    if(temp>barData.BarData.Close && bearPower.CurrentValue>0)
    {
        SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
        Debug("Alış emri verildi.");
    }
    if (temp<barData.BarData.Close && bearPower.CurrentValue<0)
    {
        SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
        Debug("Satış emri verildi.");
    }
    temp = barData.BarData.Close;
}
 else
{
    temp = barData.BarData.Close;
}
```

** bearPower.** CurrentValue: Bear Power indikatörünün o andaki değeri**

BollingerIndicator(String, SymbolPeriod, OHLCType, Int32, Decimal, MovMethod)

Buradaki bantlar fiyatları saran kılıf gibidir. Daralmaya başlayan bant, yakında bant dışına doğru bir hareket olasılığı demektir. Yönü ve süresi belli değildir. Orta – Alt ve Üst olmak üzere 3 çizgiden oluşur.

Hareketli ortalamanın belli bir standart sapması alınarak hesaplandıktan sonra hareketli ortalamadan aşağı ve yukarı yönlü kaydırılarak çizilen Bollinger bantları durgun piyasalarda daralıp, hareketli piyasalarda genişleyerek farklı yorumlar yüklenebilecek sinyaller üretirler.

Bollinger bantlarındaki iki değişkenden biri olan periyot için Bollinger kendi uygulamalarında 20 günlük periyodu önerse de fiyat hareketleri daha az olan senetlerde daha kısa periyotlar kullanılabilir iken, aşırı fiyat hareketleri olan senetlerde daha uzun periyotlar da kullanılabilir.

Aynı periyot seçimlerinde olduğu gibi standart sapma değerinin seçiminde de incelemekte olduğunuz senedin hareketliliği belirleyici rol oynamaktadır. Nasıl ki fiyat salınımları daha az olan senetlerde hareketli ortalamamızın periyodunu küçültüyorsak aynı mantıkla standart sapma değeri de küçültülebilir, tersine aşırı

fiyat hareketlerine sahip bir senette de aynı hareketli ortalama periyodunu büyüttüğümüz gibi standart sapma değerini de büyütebiliriz.

Bollinger bantlarının kullanımında fiyatların bantlar arasında gidip geldiği, bandın bir kenarına gelen fiyatların bunu takiben diğer banda doğru hareketlendiği ve bandın dışına taşan fiyat hareketlerinde de yeniden bandın içine dönüleceği varsayılır.

Bollinger bantlarındaki bir diğer önemli özellikse daralmalardır. Daralmaya başlayan bir bandın anlamı çok yakında fiyatlarda sert bir hareket olacağıdır. Hareketlerin yönü hakkında kesin bir bilgi içermese de büyük marjlarda bir fiyat değişikliği uyarısıdır.

Örnek:

```
if (CrossAbove(bollinger.BollingerDown, barData.BarData.Close))
{
    SendMarketOrder(Symbol, BuyOrderQuantity, (OrderSide.Buy));
    Debug("Alış emri verildi.");
}
if (CrossBelow(bollinger.Bollingerup, barData.BarData.Close))
{
    SendMarketOrder(Symbol, SellOrderQuantity, (OrderSide.Sell));
    Debug("Satış emri verildi.");
}
```

** bollinger.** BollingerDown: Bollinger bands indikatörünün alt bandının o andaki değeri**

BullPowerIndicator(String, SymbolPeriod, OHLCType, Int32, MovMethod)

Bulls Power bir aksi yönde trend indikatörüdür. Bulls Power fiyatlara bakmak ve piyasanın ardındaki gücü görmek için kullanışlı bir yoldur. Bir trendin yönünü belirlemenize ve dönüş noktasını bulmanıza yarar. Histogram giderek azalıyor ancak henüz olumlu bölgeden çıkmadıysa, o zaman satış pozisyonları açmak için iyi bir andır.

CCIIndicator(String, SymbolPeriod, OHLCType, Int32)

Commodity Channel Index göstergesinin değerini hesaplamak için kullanılır.

D. Lambert tarafından mal piyasaları için geliştirilen ancak yapısı itibariyle hisse senedi piyasaları için de uygun bir gösterge olan CCI da fiyatların istatistiksel ortalamadan ne ölçüde saptığı bulunmaya çalışılmaktadır.

İstatistiksel ortalamadan sapma, trendin ve tercihlerin ne yönde değişmekte olduğunun anlaşılabilmesi açısından önemlidir. Daha çok kısa vadeli trend değişimlerini kovalayan ve yatay piyasalarda daha iyi sonuçlar veren CCI, +100 ve -100 de yer alan referans değerlerinin tanımladığı aşırı alım ve aşırı satım bölgelerinin kullanımına dayanan bir osilatördür.

Aşırı alım ve satım bölgelerini kullanan diğer göstergelerde olduğu gibi, bu bölgelerde dolaşan CCI değerleri bize trendin yakın bir gelecekte sonlanabileceği sinyallerini vermektedir. Aşırı alım bölgesi, fiyatların aşırı yükseldiğini ve her an satışların gelebileceğini söylerken bu bölgeden yapılacak alımların riskinin arttığını da

anlatmaktadır. Aynı şekilde aşırı satım bölgesi de fiyatların aşırı düştüğünü ve burada alıma hazır olunması gerektiğini söylerken hala satmamış olanlar açısından satımın çok da mantıklı olmadığını anlatır.

CCI'yi yorumlamada kullanılan bir diğer metot ise fiyatlarla göstergenin uyumsuzluğudur. Fiyatlar bir önceki zirveyi geçerek yeni zirveler yaparken CCI yeni zirvesiyle buna eşlik edemiyor hatta bir önceki zirvenin altında kalıyorsa bu göstergenin yükselmeyi desteklemediğine bir işarettir ve fiyatlarda bir düzeltme beklenmelidir. Yeni dipler yapan fiyatlara yeni diplerle eşlik etmeyen bir CCI ise düşüşü desteklemiyor ve yukarı yönlü bir düzeltme sinyali veriyor demektir.

Örnek:

```
if (CrossAbove(cci, DownLevel))
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
     Debug("Alış emri verildi.");
}
if (CrossBelow(cci, UpLevel))
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
     Debug("Satış emri verildi.");
}
```

**cci: Commodity Channel Index göstergesinin o andaki değeri
**DownLevel: Commodity Channel Index göstergesinin alt seviyesi
**UpLevel: Commodity Channel Index göstergesinin üst seviyesi

CenterOfGravityOscillatorIndicator(String, SymbolPeriod, OHLCType, Int32)

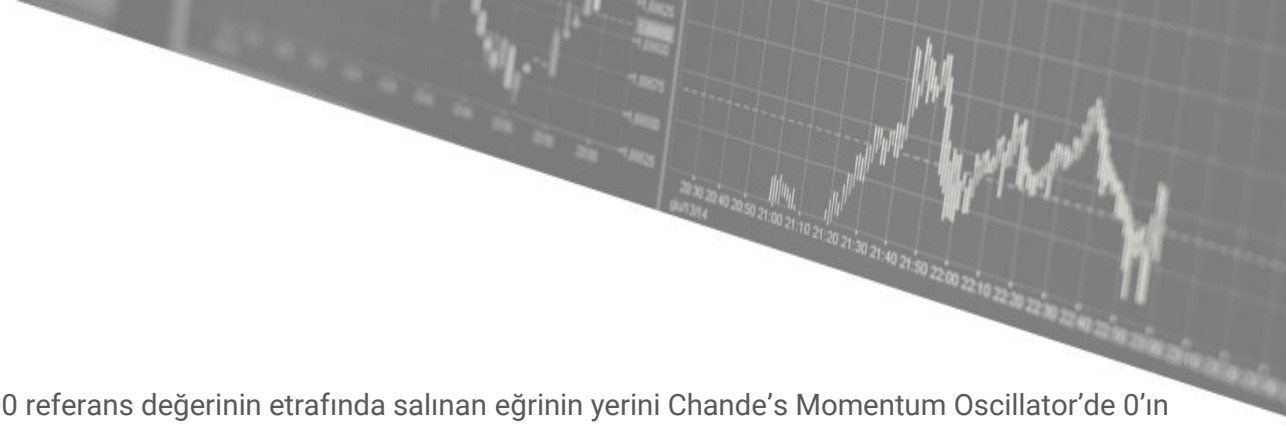
Destek ve direnç seviyeleri belirlemeye yardımcı olur. Birçok iyi bilinen göstergelerle karşılaştırıldığında yeni olmakla beraber gelecekteki fiyat salınım analizleri yapmak için uygun bir yöntem olarak çalışmaktadır, popülaritesi artmaktadır.

Bir osilatör olarak, başlıca iki faydası vardır: fiyata tepki olarak düşük gecikme ve net dönüm noktaları.

CMOIndicator(String, SymbolPeriod, OHLCType, Int32)

Chande's Momentum Oscillator göstergesinin değerini hesaplamak için kullanılır. Fiyatların yukarı / aşağı yönü konusunda sinyal verir.

Tushar Cande tarafından geliştirilen bir Momentum göstergesi olan CMO, belli bir zaman dilimi içerisinde fiyatların ne yönde ve ne şiddette hareket ettiğini gösterir.



Momentum'daki 100 referans değerinin etrafında salınan eğrinin yerini Chande's Momentum Oscillator'de 0'ın etrafında salınan bir eğri almıştır. CMO değerinin "0"'ın üzerinde olduğu değerler fiyatların yukarı yönlü arzusunu anlatırken, altındaki her değer fiyatlardaki düşüş eğilimine işaret eder.

CMO da kullanılan diğer iki referans çizgisi ise +50 ve -50 değerleridir. +50 seviyesinin üzerindeki değerler aşırı alımı ve fiyatların yakın bir zamanda düşüş gösterebileceğine işaret ederken, -50 seviyesinin altındaki değerler ise aşırı satımı ve fiyatların yakın zamanda yükselebileceğini göstermektedir.

CMO yorumlanmasında kullanılan bir diğer unsur da uyumsuzluklardır. Uyum, gevşeyen fiyatlar ve alçalan trend ile beraber oluşmakta olan yeni diplerin ve tepelerin bir öncekinden daha düşük seviyelerde oluşması, yükselen fiyatlar ve yükselen trend ile beraber de yeni oluşan tepelerinin ve diplerin bir öncekinden yukarıda olması demektir. Oluşumun bu şekilde gelişmediği durumlar uyumsuzluk olarak adlandırılır ve yakın bir zamanda trendin gücünü kaybederek ters yönde bir hareket yapacağı kabul edilir.

Örnek:

```
if (CrossAbove(cmo, DownLevel))
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
     Debug("Alış emri verildi.");
}
if (CrossBelow(cmo, UpLevel))
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
     Debug("Satış emri verildi.");
}
```

cmo: Chande's Momentum Oscillator göstergesinin o andaki değeri
DownLevel: Chande's Momentum Oscillator göstergesinin alt seviyesi
UpLevel: Chande's Momentum Oscillator göstergesinin üst seviyesi

DIIndicator(String, SymbolPeriod, OHLCType, Int32)

Directional Indicator göstergesinde pozitif ve negatif olmak üzere 2 ayrı çizgi vardır.

J.W.Wilder tarafından geliştirilen Directional Movement fiyatların hangi yönde hareket etme eğiliminde olduğunu araştırır. Trend belirleme ve yanlış sinyallerin filtre edilebilmesi açısından oldukça önemli bilgiler sunan Directional Movement, ADX, ADXR, DIS, DX ve DI+DI- gibi birçok göstergenin de çıkış noktasıdır.

Hesaplanması kolay ama oldukça uzun olan Directional Movement için öncelikle DM değerinin hesaplanması gerekmektedir. Bu değer, bir önceki güne göre hareketin yönü yukarı ise pozitif aşağı ise negatif olacaktır. +DM ve -DM değerlerini bulduktan sonra ise +DM değerinden DI+, -DM değerinden de DI- türetilir.

DI+ ve DI- grafik üzerinde beraber çizilerek gösterilirler. Fiyatın yükseldiği dönemlerde pozitif yönlü hareketin bir türevi olan DI+ yükselirken negatif yönlü hareketin bir türevi olan DI- düşecektir. Farkın DI+ lehine açılması yukarı yönlü bir trendin varlığına işaret etmektedir. Aynı şekilde fiyatların düştüğü dönemlerde bu sefer DI- artacak ve DI+ düşecektir. Farkın DI- lehine açılması ise aşağı yönlü bir trendin varlığını göstermektedir.

Yorumlama açısından genel kullanım DI+ göstergesinin DI-'yi yukarı doğru keserek üzerine çıkmasıyla "al" sinyalinin üretildiği, DI- eğrisinin DI+ eğrisini yukarı doğru keserek üzerine çıkmasıyla ise "sat" sinyalinin üretildiği şeklindedir. Ancak trendin varlığında oldukça etkili alım satım noktaları veren DI+DI- trendin olmadığı dönemlerde çok sık kesişerek hatalı sinyaller de üretebilmektedir.

Al ve sat noktalarına biraz daha temkinli yaklaşan ve hatalı sinyalleri filtre etmeyi amaçlayan bir diğer yöntemde ise DI+'nın DI-'yi yukarı keserek "al" verdiği noktada hemen alım yapılmamalıdır.

Kesişim gününün görülen en yüksek fiyatının bir sonraki işlem gününde geçilmesi beklenmelidir. Aynı şekilde DI-'nin DI+'yı yukarı keserek "sat" verdiği noktada da satım yapılmamalı ve bir sonraki iş gününde fiyatın kesişim gününde gördüğü en düşük değerin altına inmesi beklenmelidir.

EMAIndicator(String, SymbolPeriod, OHLCType, Int32)

Exponential Moving Average (EMA) göstergesinin değerini hesaplamak için kullanılır.

Üssel Hareketli Ortalama (EMA), hareketli ortalamayla çok benzerdir (ve bir türüdür). Hareketli ortalamada basit ortalama kullanılırken Üssel Hareketli Ortalamada exponential ortalama kullanılır.

Örnek:

```
decimal prevValue;

if (prevValue != null)
{
     if (prevValue<ema.CurrentValue)
     {
          SendMarketOrder(Symbol, BuyOrderQuantity, (OrderSide.Buy));
          Debug("Alış emri verildi.");
     }
     if (prevValue>ema.CurrentValue)
     {
          SendMarketOrder(Symbol, SellOrderQuantity, (OrderSide.Sell));
          Debug("Satış emri verildi.");
     }
     prevValue = ema.CurrentValue;
}
else
{
     prevValue = ema.CurrentValue;
}
```

ema. CurrentValue: Exponential Moving Average (EMA) göstergesinin o andaki değeri**
** prevValue: **Exponential Moving Average (EMA) göstergesinin bir önceki değeri**

EnvelopeIndicator(String, SymbolPeriod, OHLCType, Int32, Decimal, MovMethod)

Bollinger bandına benzer ama daha basitidir. Hareketli ortalamanın belli bir miktar üstü ve altından geçen 2 ad çizgisi vardır.

Envelopes'in iki değişkeninden biri olan hareketli ortalama periyoduna karar verirken aynı Bollinger da olduğu gibi fiyat hareketleri daha az olan senetlerde daha kısa periyotlar kullanılabilir iken, aşırı fiyat hareketleri olan senetlerde daha uzun periyotlar da kullanılabilir.

Bantların aşağı ve yukarı yönde ne ölçüde kaydırılacağını gösteren kaydırma oranının belirlenmesinde de fiyat hareketliliğine bakılması ve fiyat hareketleri daha az olan senetlerde daha küçük kaydırma oranları belirlenirken, aşırı fiyat hareketleri olan senetlerde daha büyük kaydırma oranlarının belirlenmesi gerekmektedir.

Envelopes'in yorumlanmasında fiyat salınımlarının bant içinde kaldığı ve bir banda ulaşan fiyat hareketlerinin buradan dönerek diğer banda doğru hareket edeceği varsayılır. Bu şekliyle bantların destek direnç gibi algılandığı söylenebilir. Alt banda ulaşılması "al", üst banda ulaşılması ise "sat" sinyali üretmektedir.

Örnek:

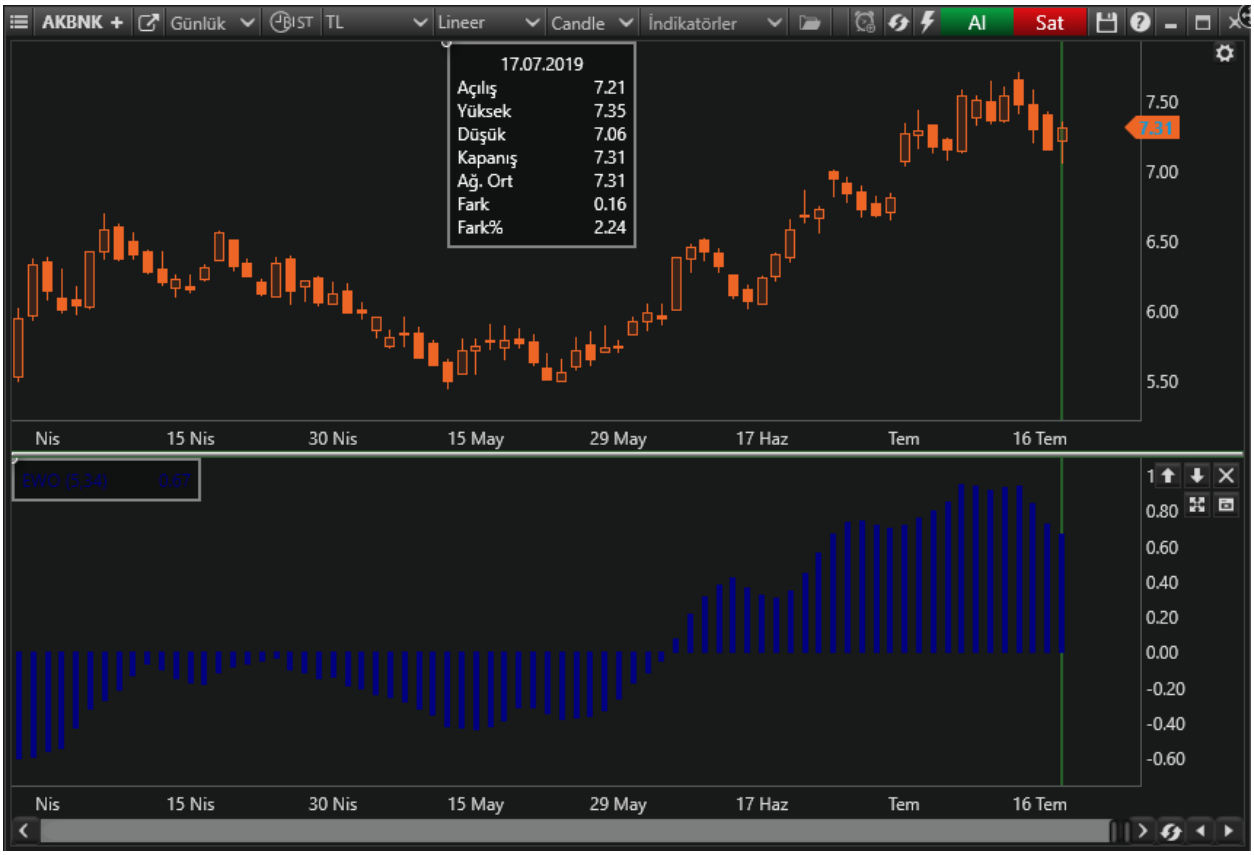
```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, env.Down, OHLCType.Close))
{
    SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
    Debug("Alış emri verildi.");
    Debug("Close:" + barData.BarData.Close);
    Debug("Env.Down:" + env.Down.CurrentValue);
    Debug("Env.Up:" + env.Up.CurrentValue);
}
if (CrossBelow(barDataModel, env.Up, OHLCType.Close))
{
    SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
    Debug("Satış emri verildi.");
    Debug("Close:" + barData.BarData.Close);
    Debug("Env.Down:" + env.Down);
    Debug("Env.Up:" + env.Up);
}
```

** env.** Down.CurrentValue: Envelope göstergesinin Down bandının değeri**
** env.** Up.CurrentValue: Envelope göstergesinin Up bandının değeri**

EWOIndicator(String, SymbolPeriod, OHLCType, Int32, Int32)

Elliott wave oscillator göstergesinin değerini hesaplamak için kullanılır.



Elliott Wave Oscillatör, 5 ve 34 günlük basit ortalamalar arasındaki farktır. Gösterim şekli olarak 'Histogram' seçilmiştir. Fiyat hareketlerinin dalgalar şeklinde ilerlediğini varsayar. Yeni zirve, bir önceki zirvenin üzerinde oldukça, yükselişin devam edeceğini, altında kaldığında ise, trendin düşüşe döneceğini varsayar.

Örnek:

```
if (CrossAbove(ewo, 0))
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
     Debug("Alış emri verildi.");
}
if (CrossBelow(ewo, 0))
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
     Debug("Satış emri verildi.");
}
```

ewo: **Elliott wave oscillator göstergesinin o andaki değeri

FAMOVIndicator(String, SymbolPeriod, OHLCType, Int32, Int32)

Fraktal Uyarlamalı Hareketli Ortalama (FRAMA), John Ehlers tarafından geliştirilen akıllı, uyarlanabilir bir hareketli ortalamadır. Fiyat değişimlerinin önemini dikkate alır ve fiyat dalgalanırken, düz kalırken belirgin değişimlerde fiyatı yakından takip eder. FRAMA, piyasaların fraktal olduğu gerçeğinden yararlanarak bu fraktal geometriye dayalı olarak geriye doğru inceleme süresini dinamik olarak ayarlar. Hesaplama çok ayrıntılı ve karmaşıktır. FRAMA genellikle diğer sinyaller ve analiz teknikleri ile birlikte kullanılır.

Örnek:

```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, famov, OHLCType.Close))
{
     SendMarketOrder(Symbol, BuyOrderQuantity, (OrderSide.Buy));
     Debug("Alış emri gönderildi.");
     Debug("Close = " + barData.BarData.Close);
     Debug("Famov = " + famov.CurrentValue);
}
if (CrossBelow(barDataModel, famov, OHLCType.Close))
{
     SendMarketOrder(Symbol, SellOrderQuantity, (OrderSide.Sell));
     Debug("Satış emri gönderildi.");
     Debug("Close = " + barData.BarData.Close);
     Debug("Famov = " + famov.CurrentValue);
}
```

famov. CurrentValue: Fraktal Uyarlamalı Hareketli Ortalama göstergesinin o andaki değeri**
OHLCType.Close: Barların kapanış değerlerine göre işlem yapılmasını sağlar

FTIndicator(String, SymbolPeriod, OHLCType, Int32)

Fisher Transform(FT) indikatörünün 2 ayrı çizgisi vardır.

Ehler tarafından tasarlanmış bir indikatördür. Çizmiş olduğu 2 tane çizginin kesişme yerine göre sinyal verir. Yeşil çizginin sarı çizgiyi yukarı doğru kırması al sinyali üretirken aşağı doğru kırması sat sinyali üretir.

Örnek:

```
if (CrossAbove(ft.Fish, ft.Trigger))
{
    SendMarketOrder(Symbol, BuyOrderQuantity, (OrderSide.Buy));
    Debug"Alış emri verildi."); Debug("FtFish:" + ft.Fish.CurrentValue);
    Debug("FtTrigger" + ft.Trigger.CurrentValue);
}
if (CrossBelow(ft.Fish, ft.Trigger))
{
    SendMarketOrder(Symbol, SellOrderQuantity, (OrderSide.Sell));
    Debug("Satış emri verildi."); Debug("FtFish:" + ft.Fish.CurrentValue);
    Debug("FtTrigger" + ft.Trigger.CurrentValue);
}
```

** ft.** Fish.CurrentValue: Fisher Transform göstergesinin Fish bandının değeri**
** ft.** Trigger.CurrentValue: Fisher Transform göstergesinin Trigger bandının değeri**

FKCIndicator(String, SymbolPeriod, OHLCType, Int32, Int32, Decimal, Decimal)

Fraktal Keltner kanalları uyarlanabilir bir fraktal hareketli ortalamaya odaklanır. Yeşil kanala yukarıdan giren bir çubuk kısa bir eğilim gösterirken mavi kanala alttan gelen sinyallere uzun bir eğilim izler. Önceki kanalının tamamen dışında kalan bir bar, trendin sonunu gösterir.

HullMAIndicator(String, SymbolPeriod, OHLCType, Int32)

Hull Moving Average (HMA) göstergesinin değerini hesaplamak için kullanılır.

Hull Hareketli Ortalaması (HMA), Alan Hull tarafından gecikmeyi azaltmak, yanıt vermeyi artırmak ve aynı zamanda gürültüyü ortadan kaldırmak amacıyla geliştirilmiştir. Hesaplaması ayrıntılıdır ve Ağırlıklı Ortalama(WMA) hesabını kullanır. Gelecekteki piyasa trendini belirlemek için, hızlı hareket eden bir hareketli ortalama üreten eski fiyatlar üzerinden son fiyatları vurgular. Giriş ve çıkış sinyalleri için de kullanılabilir.

Gösterge çoğunlukla salınım tacirleri ve uzun vadeli işlem yapanlar tarafından diğer sinyaller ve analiz teknikleriyle birlikte kullanılır.

Örnek:

```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, hullMA, OHLCType.Close))
{
     SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
     Debug("Alış Emri Gönderildi");
     Debug("Close:" + barData.BarData.Close);
     Debug("hullMA:" + hullMA.CurrentValue);
}
if (CrossBelow(barDataModel, hullMA, OHLCType.Close))
{
     SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
     Debug("Satış Emri Gönderildi");
     Debug("Close:" + barData.BarData.Close);
     Debug("hullMA:" + hullMA.CurrentValue);
}
```

** hullMA.** CurrentValue: Hull Moving Average göstergesinin o andaki değeri**

HVolatilityIndicator(String, SymbolPeriod, OHLCType, Int32)

Volatility göstergesinin değerini hesaplamak için kullanılır.

Historical (Tarihi) Volatility (Hareketlilik), son bir yılda gerçekleşen günlük bazdaki volatiliteyi ölçer. Normal volatilitenin aksine geçmişteki gerçekleşmeye bakmaktadır.

Bu indikatör tek başına teknik bir strateji belirlemede yeterli değildir. O nedenle diğer indikatörlerle birlikte kullanılır.

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

KAMAIndicator(String, SymbolPeriod, OHLCType, Int32, Int32, Int32)

Kaufman's Adaptive Moving Average (KAMA), Perry Kaufman tarafından geliştirilen akıllı bir hareketli ortalamadır. Güçlü bir trend takip göstergesi olup Üssel Hareketli Ortalama (EMA)'ya dayanır, trend ve volatiliteye karşı duyarlıdır.

Geleneksel hareketli ortalamaların en zayıf noktalarından biri, alım satım sinyalleri için kullanıldığında birçok yanlış sinyal üretme eğiliminde olmalarıdır. KAMA göstergesi bu eğilimi azaltmayı hedefler. Daha az yanlış sinyal üretir. Kısa vadeli, önemsiz fiyat hareketlerine cevap vermez.

Gürültü düşük olduğunda fiyatı yakından takip eder ve fiyat dalgalandığında gürültüyü düzleştirir. Tüm hareketli ortalamalar gibi, KAMA da trendi görselleştirmek için kullanılabilir. Fiyat kesişmesi, bir yön değişimini belirtir. Fiyat, dinamik destek ve direnç noktaları olarak görülebilecek şekilde KAMA'dan sıçramalar yapabilir. Genellikle diğer sinyaller ve analiz teknikleri ile birlikte kullanılır.

Örnek:

```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, kama, OHLCType.Close))
{
    SendMarketOrder(Symbol, BuyOrderQuantity, (OrderSide.Buy));
    Debug("Alış emri verildi."); Debug("KAMA:" + kama.CurrentValue);
    Debug("Bardata.Close:" + barData.BarData.Close);
}
if (CrossBelow(barDataModel, kama, OHLCType.Close))
{
    SendMarketOrder(Symbol, SellOrderQuantity, (OrderSide.Sell));
    Debug("Satış emri verildi."); Debug("KAMA:" + kama.CurrentValue);
    Debug("Bardata.Close:" + barData.BarData.Close);
}
```

** kama.** CurrentValue: Kaufman's Adaptive Moving Average göstergesinin değeri**

KELTIndicator(String, SymbolPeriod, OHLCType, Int32, Int32)

Keltner Kanalları göstergesinin değerini hesaplamak için kullanılır.

Keltner Kanalları göstergesi, Bollinger Bantları ve Hareketli Ortalama Zarflarına benzer bantlı bir göstergedir. Bunlar Orta Çizginin üzerinde ve altındaki birer zarftan oluşur. Orta Çizgi, kullanıcı tanımlı bir zaman aralığında hesaplanan fiyatın bir hareketli ortalamasıdır. Genellikle basit hareketli ortalama veya üstel hareketli ortalama kullanılır. Üst ve Alt Zarflar (kullanıcı tanımlı), Orta Çizgiden bir mesafede olarak ayarlanır. Bu günlük yüksek / düşük aralığının katları veya daha yaygın olarak Ortalama Gerçek Aralık'ın bir katı olabilir.

Örnek:

```
if (barData.BarData.Close < kelt.KeltDown.CurrentValue)
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
     Debug("Alış Emri Gönderildi");
}

if (barData.BarData.Close > kelt.KeltUp.CurrentValue)
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
     Debug("Satış Emri Gönderildi");
}
```

kelt. KeltDown.CurrentValue: Keltner Kanalları göstergesinin alt çizgisinin o andaki değeri**
kelt. KeltUp.CurrentValue: Keltner Kanalları göstergesinin üst çizgisinin o andaki değeri**
kelt. CurrentValue: Keltner Kanalları göstergesinin o andaki değeri**

LRLIndicator(String, SymbolPeriod, OHLCType, Int32)

Linear Regression göstergesinin değerini hesaplamak için kullanılır.

Doğrusal bağlanım olarak da tanımlanan bu gösterge, fiyatların Linear regression çizgisinin üzerinde kaldığı süreci AL sinyalini, altına düştüğü zamanlarda ise SAT sinyalini işaret eder.

Örnek:

```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, lrl, OHLCType.Close))
{
     SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
     Debug("Alış Emri Gönderildi");
     Debug("Close:" + barData.BarData.Close);
     Debug("lrl:" + lrl.CurrentValue);
}

if (CrossBelow(barDataModel, lrl, OHLCType.Close))
{
     SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
     Debug("Satış Emri Gönderildi");
     Debug("Close:" + barData.BarData.Close);
     Debug("lrl:" + lrl.CurrentValue);
}
```

lrl. CurrentValue: Linear Regression göstergesinin o andaki değeri**

LRSIndicator(String, SymbolPeriod, OHLCType, Int32)

Linear Regression Slope göstergesinin değerini hesaplamak için kullanılır.

Linear Regression'dan üretilen bu grafik, sıfırın üstüne çıktığında AL sinyali, sıfırın altına düştüğünde ise SAT sinyalini işaret eder.

Örnek:

```
if (CrossAbove(lrs, 0))
{
    SendMarketOrder(Symbol, BuyOrderQuantity, (OrderSide.Buy));
    Debug("Alış emri verildi."); Debug("LRS:" + lrs.CurrentValue);
    Debug("Bardata.Close:" + barData.BarData.Close);
}
if (CrossBelow(lrs, 0))
{
    SendMarketOrder(Symbol, SellOrderQuantity, (OrderSide.Sell));
    Debug("Satış emri verildi.");
    Debug("LRS:" + lrs.CurrentValue);
    Debug("Bardata.Close:" + barData.BarData.Close);
}
```

** lrs.** CurrentValue: LRS göstergesinin o anda ki değeri**

MACDIndicator(String, SymbolPeriod, OHLCType, Int32, Int32, Int32)

MACD göstergesinin değerini hesaplamak için kullanılır.

Genelde 26 ve 12 günlük olarak kullanılan iki üssel hareketli ortalamanın farkı olan MACD, kısa vadeli üstel ortalamanın uzun vadeliye göre olan pozisyonlarını değerlendirerek trendin yönü hakkında bilgi vermeye çalışır.

Macd'nin 0 olduğu seviyede 12 günlük üssel hareketli ortalama 26 günlük üssel ortalamaya eşittir. 12 günlüğün 26 günlüğün üzerine çıktığı durumlarda ise Macd pozitif değerler alacaktır. Tersine 12 günlüğün 26 günlüğün altına indiği durumlarda ise Macd negatif olacaktır.

Yorumlamalarda 12 günlük hareketli ortalamanın 26 günlük hareketli ortalamadan uzaklaştığı yani Macd'nin 0 değerinin altında ya da üzerinde olduğu durumlar aşırı alım ve aşırı satımlar olarak değerlendirilebilir. Ancak bu bölgelerde olmanın ürettikleri sinyaller referans değeri kullanan diğer göstergelere göre daha zayıftır. Fiyatların trend yönünde aşırı bir hareketi olduğunu söylese de dönüş zamanı hakkında kesin bilgiler içermez.

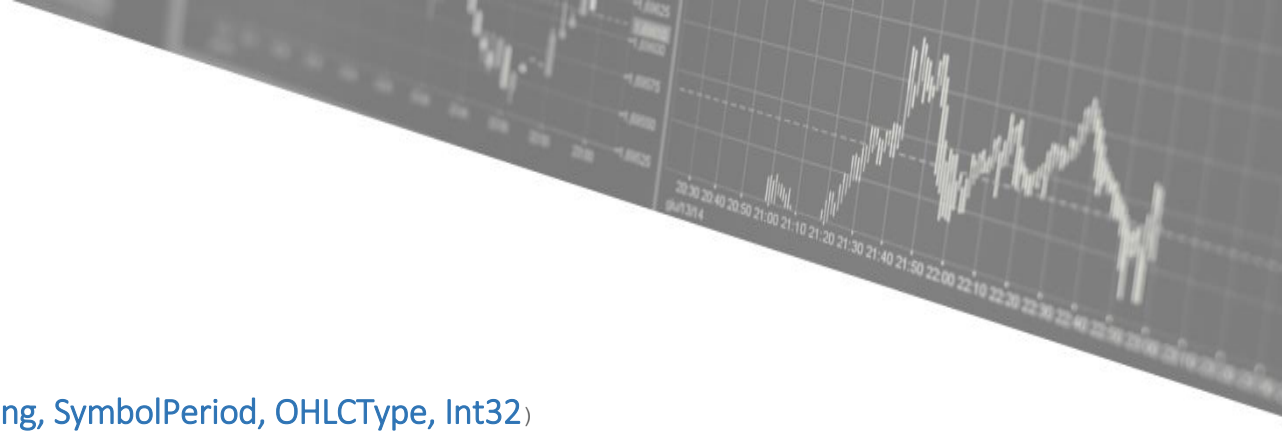
Macd eğrisinin 9 günlük hareketli ortalamasının alınarak buna göre olan pozisyonların değerlendirilmesidir. Bu yöntemde Macd eğrisi, 9 günlük hareketli ortalaması olan trigger eğrisini yukarı yönlü kestiğinde "al", aşağı yönlü kestiğinde ise "sat" sinyali verilmiş olur.

Macd'nin yorumlanmasında önem taşıyan bir diğer yöntemde uyumsuzlukların araştırılmasıdır. Uyumun, gevşeyen fiyatlar ve alçalan trend ile beraber oluşmakta olan yeni diplerin ve tepelerin bir öncekinden daha düşük seviyelerde oluşması, yükselen fiyatlar ve yükselen trend ile beraber de yeni oluşan tepelerinin ve diplerin bir öncekinden yukarıda olması demek olduğunu kabul edersek oluşumun bu şekilde gelişmediği durumlar uyumsuzluk olarak adlandırılır ve yakın bir zamanda trendin gücünü kaybederek ters yönde bir hareket yapacağı kabul edilir.

Örnek:

```
if (CrossAbove(macd, macd.MacdTrigger))
{
     SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
     Debug("Alış Emri Gönderildi");
}
if (CrossBelow(macd, macd.MacdTrigger))
{
     SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
     Debug("Satış Emri Gönderildi");
}
```

macd. MacdTrigger: MACD göstergesinin Trigger bandının o andaki değeri**



MDIIndicator(String, SymbolPeriod, OHLCType, Int32)

McGinley Dynamic göstergesi (**MDI**), piyasayı mevcut hareketli ortalama göstergelerinden daha iyi izlemek için tasarlanmış bir hareketli ortalama türüdür.

Piyasa hızındaki değişimleri ayarlayarak hareketli ortalama çizgilerini iyileştiren teknik bir göstergedir. MDI çizgisi, volatilite yükselmesi olmadıkça fiyatlar ile birlikte hareket eder.

Örnek:

```
if (CrossAbove(mdi, barData.BarData.Close))
{
     SendMarketOrder(Symbol, OrderQuantity, OrderSide.Buy);
     Debug("Alış Emri Gönderildi");
}
if (CrossBelow(mdi, barData.BarData.Close))
{
     SendMarketOrder(Symbol, OrderQuantity, OrderSide.Sell);
     Debug("Satış Emri Gönderildi");
}
```

MeanDevIndicator(String, SymbolPeriod, OHLCType, Int32): Mean Deviation göstergesi William Blau tarafından yazılmıştır. MeanDev göstergesi, fiyat değerini ve üssel hareketli ortalama (EMA) farkını kullanır.

Mean Deviation x dönem hareketli ortalama fiyat konumunu gösterir. MeanDev değeri, pozitif ise aşağı yönlü trend, hareketli ortalamanın altında ise negatif yönlü trend vardır.

MOMIndicator(String, SymbolPeriod, OHLCType, Int32): Momentum göstergesinin değerini hesaplamak için kullanılır.

Momentum, belli bir zaman dilimi içinde fiyatların ne yönde ne miktarda ve ne şiddette hareket ettiğini anlatan bir göstergedir. Momentumun asıl hedefi, periyot kadar önceki kapanışa göre şu anki kapanışın nerede olduğunun bulunmasıdır.

Kısa vadeli bir gösterge olarak kullanacağımız momentumda 12- 14 günlük periyotlar daha olumlu sonuçlar vermesine karşın kullanacağınız vade, ilgilenilen senede ve uygulanan stratejilere göre değişebilmektedir.

Hareketliliğin yüksek olduğu senetlerde vade uzatılabilirken hareketlilik azaldıkça vade de kısaltılabilir.

Son günün kapanışının, x gün önceki kapanışı bölünmesinin 100 ile çarpımı olarak hesaplanan Momentum'da bu iki günün eşit olduğu durumlarda bulunacak olan 100 değeri referans değeri olarak kabul edilir.

Kullanım açısından diğer referans değerleri kullanan indikatörlerde olduğu gibi bu seviyenin üzerinde "aşırı alım" altında ise "aşırı satım" olarak kabul edilmesi mümkündür. Ayrıca Momentumla bu çizginin kesişme noktalarında Momentumun referans değerini yukarı kesmesi "al" aşağı kesmesi ise "sat" sinyali olarak kabul edilebilir.

Momentumun yorumlanmasında bir diğer unsur da uyumsuzluklardır. Uyum, gevşeyen fiyatlar ve alçalan trend ile beraber oluşmakta olan yeni diplerin ve tepelerin bir öncekinden daha düşük seviyelerde oluşması, yükselen fiyatlar ve yükselen trend ile beraber de yeni oluşan tepelerinin ve diplerin bir öncekinden yukarıda olması demektir. Oluşumun bu şekilde gelişmediği durumlar uyumsuzluk olarak adlandırılır ve yakın bir zamanda trendin gücünü kaybederek ters yönde bir hareket yapacağı kabul edilir.

Örnek:

```
if (CrossAbove(mom, ema))
{
     SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
     Debug("Alış Emri Gönderildi");
}
else if (CrossBelow(mom, ema))
{
     SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
     Debug("Satış Emri Gönderildi");
}
```

MOSTIndicator(String, SymbolPeriod, OHLCType, Int32, Decimal, MovMethod)

MOST göstergesinin değerini hesaplamak için kullanılır.

Anıl Özekşi tarafından geliştirilen Moving Stoploss göstergesi, adından da anlaşılacağı üzere hareketli stoploss mantığı üzerine inşa edilmiş bir modeldir. Ancak bu göstergedeki hareketli stoploss, klasik stoploss tanımından farklı olarak, fiyatların düşmesi durumunda alınmış olan pozisyonun kapatılması eyleminin yanına fiyatların yükselmesi durumunda kapatılmış olan pozisyonun yeniden açılması eylemini de ekleyerek fiyat salınımlarına mükemmele uyan ve bu salınımları bir kılıf gibi örten bir yapıda sunulmaktadır.

Klasik anlayıştaki hareketli stoploss mantığından ayıran bir diğer özelliği ise, MOST göstergesinin direk fiyatlarla ilişkilendirilmeyerek al ve sat sinyallerini fiyatların üç günlük üstel ortalaması yardımıyla üretiyor

olmasıdır. Fiyat hareketlerindeki aşırılıkların bir ölçüde filtrelendiği bu üç günlük üstel ortalamanın MOST eğrisini yukarı doğru keserek üzerine çıkması al sinyali, aşağı doğru keserek altına inmesi ise sat sinyali olarak kabul edilir. Bu ise, üç günlük üstel ortalamanın üstte olduğu tüm durumlar için alınmış pozisyonların korunacağı, MOST eğrisinin üstte olması durumunda ise satış yapılarak nakit pozisyonda kalınacağı anlamına gelecektir.Burada bir diğer önemli nokta ise MOST'un sadece fiyat hareketlerine odaklanmış bir gösterge olmasıdır. Yani çok kısa vadeli hareketli ortalamamızı yumuşatılmış fiyatlar, MOST'u ise bir hareketli ortalama gibi düşünecek olursak, bu göstergenin çalışma prensibinin hareketli ortalamalardaki gibi olduğu görülecektir. MOST, fiyatları temsil eden hareketli ortalamanın üzerine çıkarsa satım, altına inerse de alım yapılacaktır. Tek fark, MOST kendi ile aynı yöndeki hareketli ortalama hareketinde aradaki stoploss mesafesini koruyarak ortalamayı izlerken, hareketli ortalamada oluşan ters yönlü bir harekete yataya girerek cevap vermektedir.

MOST eğrisindeki iki değişkenden ilki ilişki içinde olduğu hareketli ortalamanın periyodudur. Yani fiyatlar için kullanmış olduğunuz yumuşatma oranı ne ise MOST'un periyodu da o olacaktır. Burada üç günlük bir ortalama seçmemizin nedeni ise, daha kısa vadelerin bizi fiyatların aşırı hareketlerini filtre etme amacından uzaklaştırması, daha uzun vadelerde ise hareketli ortalamanın fiyatları taklit kabiliyetinin azalmasıdır.

MOST eğrisini oluşturan ikinci değişken ise, kullanılan hareketli ortalama ile sinyalizasyonunu sağlayan kaydırma oranı yani stoploss mesafesidir. Analizlerimizde varsayılan olarak %2 kabul ettiğimiz bu kaydırma oranı da diğer göstergelerde olduğu gibi vadeli tercihlere konu olabilse de yarım puanlık oynamalara bile duyarlı olduğu ve küçülmesinin hatalı sinyal, büyümesinin ise üretilecek sinyallerde gecikme riski taşıdığı unutulmamalıdır.

Örnek:

```
if (CrossAbove(most, most.ExMOV))
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
     Debug("Alış emri verildi.");
     Debug("Most.ExMov:" + most.ExMOV.CurrentValue);
     Debug("Most:" + most.CurrentValue);
}
if (CrossBelow(most, most.ExMOV))
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
     Debug("Satış emri verildi.");
     Debug("Most.ExMov:" + most.ExMOV.CurrentValue);
     Debug("Most:" + most.CurrentValue);
```

```
}
```

```
**most: Moving Stoploss göstergesinin o andaki değeri
**most.ExMov: Moving Stoploss göstergesinin ExMov bandının o andaki değeri
```

MOVIndicator(String, SymbolPeriod, OHLCType, Int32, MovMethod)

Moving Average göstergesinin değerini hesaplamak için kullanılır.

Hareketli Ortalama, herhangi bir hisse senedindeki fiyatların, belli bir zaman aralığındaki ortalamasıdır. Örneğin kapanış verisinin 22'lik basit hareketli ortalamasının hesaplanması şu şekildedir: Fiyat serisi üzerindeki her bar için, kendi dahil olmak üzere geriye doğru toplam 22 barın kapanışlarının toplanarak 22 ye bölünmesi suretiyle elde edilmiştir.

Bir hareketli ortalamanın hesaplanabilmesi için öncelikle üç değişken üzerinde karar vermemiz gerekir. Bunlar veri tipi, periyot ve hesaplama şeklidir.

Veri tipi: Hareketli ortalama hesaplamasında kullanacağımız ilk değişken veri tipidir. Açılış, kapanış, en yüksek, en düşük, ağırlıklı ortalama ya da başka bir veri kullanarak hareketli ortalamayı hesaplamak mümkündür. Kullanım açısından en çok kullanılanı kapanışa göre hesaplamaktır.

İpucu: Veri olarak bir başka indikatörü kullanabilir ve o indikatörün hareketli ortalamasını da hesaplayabilirsiniz.

Periyot: Periyodun seçimi daha çok ne tür bir yatırım stratejisi ile ilgilenildiği ile alakalıdır. Hareketli ortalamayı kısa vadeli alım satımlar için kullanmak isteyenlerin tercihleri çoğunlukla 5, 9 ya da 14 günlük ortalamalar olurken orta ve uzun vadeli yatırımcıların tercihleri 50, 100 günlük olur. Daha uzun vadeli bakanlar ise 200 ve üzeri hareketli ortalamalar kullanmaktadır.

İpucu: Bunlar kesin değerler değildir. Ayrıca, yakın vadeli bakanlar 5-60 dakikalık periyot kullanabilir. Uzun vadeli bakanlar da Haftalık periyot kullanabilirler.

Hesaplama yöntemi: Hareketli ortalama yöntemi olarak oldukça çok sayıda seçenek vardır. Bunlar hesaplama yaptıkları barların verilerini farklı yöntemlerle hesaplamaya dahil ederler.

Yöntemler:

S: Simple (Basit)) (Aynı İsimde Bir İndikatör var. Talep üzerine MatriksIQ' ya eklenmiştir)
E: Exponential (Üstel) (Aynı İsimde Bir İndikatör var. Talep üzerine MatriksIQ' ya eklenmiştir)
W: Weighted (Ağırlıklı) (Aynı İsimde Bir İndikatör var. Talep üzerine MatriksIQ' ya eklenmiştir)
TRI: Triangular (Üçgensel) (Aynı İsimde Bir İndikatör var. Talep üzerine MatriksIQ' ya eklenmiştir)
VAR: Variable (Değişken) (Aynı İsimde Bir İndikatör var. Talep üzerine MatriksIQ' ya eklenmiştir)
ZL: Zero Lag (Yakınsayan Diyebiliriz) (Aynı İsimde Bir İndikatör var. Talep üzerine MatriksIQ' ya eklenmiştir)
WW: Welles Wilder (Hesaplayan kişinin adı ile anılmaktadır) (Aynı İsimde Bir İndikatör var. Talep üzerine MatriksIQ' ya eklenmiştir.)
TSF: Time Series Forecast (Aynı İsimde Bir İndikatör var. Talep üzerine MatriksIQ' ya eklenmiştir.)

Ancak son günlerin önemini atlayan bu yaklaşıma karşı ağırlıklı hareketli ortalamada da ilk günlerin önemi azaltılarak ağırlık son günlere kaydırılır. Üssel hareketli ortalamada ise ağırlığın yine son günlere verilmesine karşın ilk günler ağırlıklı hareketli ortalamada olduğu gibi ihmal edilmez.

Diğer yöntemler biraz daha az kullanılmaktadır. Üçgensel hareketli ortalamada ağırlık orta kısımlara verilir. Değişken hareketli ortalamada hesaplamanın içine bir de dalgalanma oranı dahil edilmiştir. Zero Lag fiyatlara oldukça yakınsayarak gider.

En basit hareketli ortalama yorumu fiyatların hareketli ortalama ile karşılaştırılmasıdır. Buna göre fiyatların hareketli ortalamanın altına inmesiyle "sat", üzerine çıkmasıyla ise "al" sinyali üretilmiş olur. Burada amaç en dip ve en tepeleri yakalamaktan çok trend dönüşlerini yakalamak ve trend süresince trendin gerektirdiği pozisyonda kalmaya çalışmaktır.

Hareketli ortalamaların alım ve satım sinyallerini yorumlarken kullandığımız bir diğer yöntem ise farklı periyotta iki hareketli ortalama birden kullanarak bunların hareketlerini hem birbirleriyle hem de fiyatlarla karşılaştırmak suretiyle alım ve satım noktalarının üretildiği bölgeleri bulmaya çalışmaktır.

Yine fiyatların hareketli ortalamaların altına inmesi "sat", üzerine çıkması ise "al" sinyali anlamına gelmektedir. Bunun yanında bu yöntemi kullananlar açısından kısa vadeli olan hareketli ortalamanın uzun vadeli olan diğerinin üzerine çıkması "al", altına inmesi ise "sat" sinyali anlamındadır.

Örnek:
(Örnek olarak Exponential Moving Average seçilmiştir.)

```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, ema, OHLCType.Close))
{
     SendMarketOrder(Symbol, BuyOrderQuantity, (OrderSide.Buy));
     Debug("Alış emri gönderildi.");
     Debug("Close = " + barData.BarData.Close);
     Debug("Ema = " + ema.CurrentValue);
}
if (CrossBelow(barDataModel, ema, OHLCType.Close))
{
     SendMarketOrder(Symbol, SellOrderQuantity, (OrderSide.Sell));
     Debug("Satış emri gönderildi.");
     Debug("Close = " + barData.BarData.Close);
     Debug("Ema = " + ema.CurrentValue);
}
```

MSLIndicator(String, SymbolPeriod, OHLCType, Int32)

Moving Stoploss göstergesinin değerini hesaplamak için kullanılır.

Kaybı sınırlamak için kullanılan stop-loss özelliğinin, kazançla birlikte, stop loss seviyesini yukarı çekerek, daha sağlıklı bir stop-loss uygulaması haline getirilmiş halidir.

Örnek:

```
if (CrossBelow(msl, barData.BarData.Close))
{
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
     Debug("Stoploss seviyesinin altına inildi. Satış emri verildi.");
}
```

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

PDIIndicator(String, SymbolPeriod, OHLCType, Int32)

Price Direction Indicator (PDI) göstergesinin değerini hesaplamak için kullanılır. Yatırımcılar, fiyatların üst ve alt sıralarını belirlemek için Price Direction Indicator (PDI) kullanabilir. PDI, çoğu alım ve satım kombinasyonunun, fiyatlar tersine çevrildiğinde para kazandırma prensibiyle çalışır. Bir analiz periyodunun her bir günü için PDI değeri, analiz periyodunun ilk tarihinden o anki tarihe kadar tüm işlemler için kümülatif geri dönüşlere eklenen yüzdelik kazananlardır. PDI değerleri doğrudan hareketli ortalamalar gibi fiyatlardan elde edilmediğinden, bir fiyat tablosunda çizildiğinde PDI değerleri gerçek fiyatların çok altında veya üstünde konumlandırılabilir. PDI değerlerinin büyüklüğü önemli değildir; faydalı olan onların kalıplarıdır. Günlük veya haftalık kapanış fiyatları ve PDI değerleri grafiği, üstleri ve altları görmenize yardımcı olur. Fiyatlar yükselirken, PDI değerleri yükselme eğilimindedir ve fiyatlar düşerken, PDI değerleri düşme eğilimindedir. PDI her zaman fiyatlar düştükten sonra düşer. PDI değerindeki eğilimler fiyatların yönündeki değişiklikleri teyit etme eğilimindedir.

Örnek:

```
decimal temp;

if (temp != null)
{
     if (temp<pdi.CurrentValue)
     {
          SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
          Debug("Alış emri verildi.");
    }
    if (temp>pdi.CurrentValue)
    {
          SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
          Debug("Satış emri verildi.");
    }
    temp = pdi.CurrentValue;
}
else
{
     temp = pdi.CurrentValue;
```

```
}
```

** pdi.** CurrentValue: Price Direction Indicator (PDI) göstergesinin o andaki değeri**

RSIIndicator(String, SymbolPeriod, OHLCType, Int32)

Relative Strength Index göstergesinin değerini hesaplamak için kullanılır.

İlk olarak J.Welles Wilder tarafından kullanılan Göreceli Güç Endeksi (RSI), bir hisse senedinin incelenen periyot içindeki bir önceki güne göre yükselen günleriyle bir önceki güne göre düşen günlerini bularak bunları birbiri ile karşılaştırmak ve bu şekilde öngörülerde bulunabilmeyi amaçlamaktadır.

Bir önceki güne göre düşen ve çıkan günlerin birbiri ile mukayesesi esasına dayandığı için de bu bilgilerin kısa vadeli bilgiler olduğu ve bunların kısa süre içinde kullanılıp tüketilmesi gerektiği de açıktır.

Her ne kadar vadeler kullanıcının tercihleri doğrultusunda belirlenecek olsa da RSI açısından vadelerin uzaması fiyat hareketlerine tepkisizleşme sorununu da beraberinde getirecektir.

RSI da genellikle 30 ve 70 referans değerleri kullanılmaktadır. Genel kabul 30 referans değerinin altına inen RSI değerlerinin aşırı satımı, 70 değerinin üzerine çıkan RSI değerlerinin de aşırı alımı işaret ettiği yönündedir. Diğer aşırı alım ve aşırı satım bölgeleri içeren göstergelerde olduğu gibi bu bölgelerde dolaşan RSI değerleri bize trendin yakın bir gelecekte sonlanabileceği sinyallerini vermektedir.

Örnek:

```
if (CrossAbove(rsi, DownLevel))
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));

if (CrossBelow(rsi, UpLevel))
     SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
```

** rsi: **Relative Strength Index göstergesinin o andaki değeri**
** **DownLevel: Relative Strength Index göstergesinin alt çzigisinin değeri**
** **UpLevel: Relative Strength Index göstergesinin üst çzigisinin değeri**

SMAIndicator(String, SymbolPeriod, OHLCType, Int32)

Simple Moving Average (SMA) göstergesinin değerini hesaplamak için kullanılır.

SMA'lar genellikle trend yönünü belirlemek için kullanılır. SMA, yükseliyorsa yukarı yönlü trend, düşüyorsa aşağı yönlü trend vardır. 200 bar SMA uzun vadeli trend için ortak vekildir. 50 bar SMA'lar tipik olarak ara eğilimi ölçmek için kullanılır. Kısa dönem SMA'lar, kısa vadeli eğilimleri belirlemek için kullanılabilir.

SMA'lar genellikle fiyat verilerini ve teknik göstergeleri düzeltmek için kullanılır. SMA süresi ne kadar uzun olursa sonuç o kadar yumuşak olur, ancak SMA ile fiyat arasında daha fazla gecikme yaşanır.

SMA genellikle alım satım sinyallerini tetiklemek için kullanılır.

Örnek:

```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, sma, OHLCType.Close))
{
     SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
     Debug("Alış Emri Gönderildi");
     Debug("Close:" + barData.BarData.Close);
     Debug("Sma:" + sma.CurrentValue);
}
if (CrossBelow(barDataModel, sma, OHLCType.Close))
{
     SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
     Debug("Satış Emri Gönderildi");
     Debug("Close:" + barData.BarData.Close);
     Debug("Sma:" + sma.CurrentValue);
}
```

STDEVIndicator(String, SymbolPeriod, OHLCType, Int32)

Standart Deviation (Varyans'ın karekökü) göstergesinin değerini hesaplamak için kullanılır.

Senedin hareketliliğini ölçer. Sıfıra yaklaştıkça hareketlilik az demektir, 1'e yaklaştıkça hareketlilik yüksektir.

Bu indikatör tek başına teknik bir strateji belirlemede yeterli değildir. O nedenle diğer indikatörlerle birlikte kullanılır.

StochasticCCIIndicator(String, SymbolPeriod, OHLCType, Int32)

StochasticCCI göstergesinin değerini hesaplamak için kullanılır.

StochasticCCI, parelel veya bitişik çubuklarda yukarı veya aşağı hareket ederse trendin devam edeceği anlamına gelir.

StochasticCCI'de mavi ok yeşil oku yukarı kestiği anda al, aşağı kestiği anda sat sinyali üretir.

StochasticFastIndicator(String, SymbolPeriod, OHLCType, Int32, Int32)

Stokastik Fast göstergesinin değerini hesaplamak için kullanılır.

Periyodun içinde yer alan kapanışları periyotta ki en yüksek ve en düşük değere olan uzaklıklarına göre değerlendiren bir osilatördür.

Stochastic, periyodun içinde yer alan kapanışları periyotta ki en yüksek ve en düşük değere olan yakınlıklarına göre değerlendiren bir osilatördür. Daha doğru deyişle, periyot içindeki en yüksek ve en düşük seviyelere olan uzaklığına göre fiyatın değerlendirilmesidir.

İlki %K ve ikincisi %D olarak isimlendirilen iki eğri ile gösterilen Stochastic'te %K, bugünün kapanışından periyodun en düşük gününün çıkarılması ile bulunan değerin, periyotta ki en yüksek en düşük farkına bölünmesinin yüzdesel bir ifadesidir.

Ana eğrimiz olan %K 'nın bulunmasının ardından ikinci eğrimiz olan %D 'yi de %K 'nın hareketli ortalamasını alarak buluruz. George C. Lane tarafından geliştirilen bu gösterge için önerilen değerler %K için 5 günlük bir periyot, bunun hareketli ortalaması için de 3 günlük bir basit ortalamadır. Bulduğumuz bu iki eğrinin aynı grafik üzerinde gösterilmesi ile elde ettiğimiz ise Fast Stochastic göstergesidir.

Ancak bu göstergenin her fiyat hareketine hemen tepki veren kararsız yapısı nedeniyle analistler yine bundan üretilen Slow Stochastic'i kullanmayı tercih ederler. Burada yapılacak olan ilk olarak %K 'nın hareketli ortalaması alınarak bulunan %D 'yi yavaşlatılmış %K değeri olarak kabul edip ana eğrinin yerine bunu koymaktır. Ardından bu yavaşlatılmış %K 'nın bir kez daha 3 günlük hareketli ortalaması alınarak yeni %D eğrisine ulaşılır. Bu yavaşlatılmış %K ve %D 'nin aynı grafikte gösterilmesi ile elde edilen gösterge ise Slow Stochastic olarak adlandırılır. Hem Fast Stochastic hem de Slow Stochastic de %K'nın %D'yi yukarı doğru kesmesi "al" sinyali olarak kabul edilirken, %K'nın %D'yi yukarıdan aşağıya kesişi ise "sat" sinyalini oluşturulmaktadır.

Stochastic 0'la 100 arasındaki değerlerden oluşur. Bu skala üzerine 20 ve 80 değerlerinden çizdiğimiz yatay çizgiler ise stochastic'in referans değerleridir. Diğer referans değeri kullanan göstergelerde olduğu gibi burada da bu değerlerin alt ve üst kısımları aşırı alım ve aşırı satım bölgeleri olarak adlandırılır. Bir senedin fiyatının 20 referans değerinin altına inmesi aşırı satıma ve bir süre sonra toparlanma olacağına, 80 referans değerinin üzerine çıkması ise aşırı alıma ve bir süre sonra gevşeme olacağına işaret eder.

Örnek:

```
if (CrossAbove(stochasticFast.StochasticFastK, stochasticFast.StochasticFastD))
{
    SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
    Debug("Alış Emri Gönderildi");
    Debug("stochasticFast.StochasticFastK:"+stochasticFast.StochasticFastK.CurrentValue);
    Debug("stochasticFast.StochasticFastD:"+stochasticFast.StochasticFastD.CurrentValue);
}
if (CrossBelow(stochasticFast.StochasticFastK, stochasticFast.StochasticFastD))
{
    SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
    Debug("Satış Emri Gönderildi");
    Debug("stochasticFast.StochasticFastK:"+stochasticFast.StochasticFastK.CurrentValue);
    Debug("stochasticFast.StochasticFastD:"+stochasticFast.StochasticFastD.CurrentValue);
}
```

** stochasticFast.** StochasticFastK.CurrentValue: Stochastic Fast göstergesinin %K değeri**

** stochasticFast.** StochasticFastD.CurrentValue: Stochastic Fast göstergesinin %D değeri**

StochasticRSIIndicator(String, SymbolPeriod, OHLCType, Int32)

Stochastic RSI göstergesinin değerini hesaplamak için kullanılır.

Stochastic ve RSI indikatörlerini birleştirir. Daha doğrusu RSI indikatörüne, Stohastic indikatörün uygulanması ile oluşmuştur. 0 ve 100 arasında değerler alır. 80 üstü aşırı alım ve 20 altı ise, aşırı satım işaretidir. Bu sınır değerler AL-SAT sinyalleri için kullanılabilir.

StochasticSlowIndicator(String, SymbolPeriod, OHLCType, Int32, Int32, Int32)

Yavaş Stokastik verilerini hesaplar. 2 ayrı çizgi oluşturmaktadır.

Bilgilendirme için yukarıda yer alan Stohastic Fast indikatörünün altındaki açıklamaya bakınız.

Örnek:

```
if (CrossAbove(stochasticSlow.StochasticSlowK, stochasticSlow.StochasticSlowD)
                                              && stochasticSlow.CurrentValue <= DownLevel)
{
     SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
     Debug("Alış Emri Gönderildi");
}
if (CrossBelow(stochasticSlow.StochasticSlowK, stochasticSlow.StochasticSlowD)
                                               && stochasticSlow.CurrentValue >= UpLevel)
{
     SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
     Debug("Satış Emri Gönderildi");
}
```

** stochasticSlow. **CurrentValue: Stochastic Slow göstergesinin o andaki değeri**
** stochasticSlow.** StochasticSlowK: Stochastic Slow göstergesinin stochastic slowK çizgisinin o andaki değeri**
** stochasticSlow. **StochasticSlowD: Stochastic Slow göstergesinin stochastic slowD çizgisinin o andaki değeri**

TMAIndicator(String, SymbolPeriod, OHLCType, Int32)

Triangular Moving Average(TMA), belirli sayıda veri noktası üzerinden ortalama fiyatı göstermesiyle diğer hareketli ortalamalara benzer. Bununla birlikte, üçgensel hareketli ortalama, çift düzgünleştirilmiş olması bakımından farklılık gösterir bu da iki kere ortalama anlamına gelir.

Üçgensel hareketli ortalamanın amacı, bir SMA kadar hızlı tepki vermeyen bir çizgi oluşturacak olan fiyat verisinin tahminini iki katına çıkarmaktır. TMA'yı ne için kullandığınıza bağlı olarak avantajlı veya dezavantajlı olabilir.

TMA değişken piyasa koşullarında hızlı bir şekilde tepki vermeyecektir; TMA'nın2ön değiştirmesi daha uzun sürecektir.

TMA gecikmeler olsa da faydalı olabilir. Fiyat ileri ve geri (aralık) hareket ederse, TMA o kadar fazla tepki vermeyecek, böylece eğilimin değişmediğini size bildirecek. TMA'nın yönünü değiştirmesine neden olmak için fiyatta daha kalıcı bir hareket gerekiyor.

Örnek:

```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, tma, OHLCType.Close))
{
     SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
     Debug("Alış Emri Gönderildi");
     Debug("Close:" + barData.BarData.Close);
     Debug("Tma:" + tma.CurrentValue);
}
if (CrossBelow(barDataModel, tma, OHLCType.Close))
{
     SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
     Debug("Satış Emri Gönderildi");
     Debug("Close:" + barData.BarData.Close);
     Debug("Tma:" + tma.CurrentValue);
}
```

** tma.** CurrentValue: Triangular Moving Average göstergesinin o andaki değeri**

TSFIndicator(String, SymbolPeriod, OHLCType, Int32)

Time Series Forecast göstergesinin değerini hesaplamak için kullanılır.

Time Series Forecast indikatörü Lineer Regresyon metodu kullanarak hesaplanır. Lineer regresyon bir istatistik aracı olarak geçmiş değerleri karşılaştırarak gelecek fiyat değerlerini tahmin içindir. Bu amaçla trendlerin yukarıya veya aşağıya doğru meyillerini tanımlar ve bu sonuçları geleceğe taşır. Örneğin, fiyatlar yukarı doğru hareket ederken, TSF fiyatın yukarı meylini o anki fiyatla karşılaştırarak bu hesaplamayı geleceğe taşır.

Time Series Forecast, fiyatlar indikatörün altına düştüğünde trendi aşağı yönlü, indikatörün üstüne çıktığında ise yukarı yönlü kabul etmektedir. TSF İndikatörü eğer yönde ve eğimde bir değişiklik yoksa devam eden trendi tanımlar.

Örnek:

```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, tsf, OHLCType.Close))
{
    SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
    Debug("Alış Emri Gönderildi");
    Debug("Close:" + barData.BarData.Close);
    Debug("Tsf:" + tsf.CurrentValue);
}
if (CrossBelow(barDataModel, tsf, OHLCType.Close))
{
    SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
    Debug("Satış Emri Gönderildi");
    Debug("Close:" + barData.BarData.Close);
    Debug("Tsf:" + tsf.CurrentValue);
}
```

** tsf.** CurrentValue: Time Series Forecast göstergesinin o andaki değeri**

VMAIndicator(String, SymbolPeriod, OHLCType, Int32):

Variable Moving Average (VMA) Tushar S. Chande tarafından geliştirilmiştir. Variable Moving Average(VMA), yumuşatma sabitini piyasa volatilitesine göre ayarlayan üstel bir hareketli ortalamadır. Verilerin değişkenliği arttıkça, hassasiyeti artar. Bu üstel hareketli ortalamanın performansı, piyasa koşulları değiştikçe yumuşatma periyodunu ayarlamak için bir Volatility Index (VI) kullanılarak geliştirilmiştir.

Örnek:

```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, vma, OHLCType.Close))
{
    SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
    Debug("Alış Emri Gönderildi");
    Debug("Close:" + barData.BarData.Close);
    Debug("Vma:" + vma.CurrentValue);
}
if (CrossBelow(barDataModel, vma, OHLCType.Close))
{
    SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
    Debug("Satış Emri Gönderildi");
    Debug("Close:" + barData.BarData.Close);
    Debug("Vma:" + vma.CurrentValue);
}
```

** vma.** CurrentValue: Variable Moving Average göstergesinin o andaki değeri**

VolumeIndicator(String, SymbolPeriod)

Piyasada belirli bir süre zarfında ne kadar hisse senedinin el değiştirdiğini anlatan bir gösterge olan volume, genellikle fiyat hareketlerinin yorumlanması ve teyidinde kullanılmaktadır.

Yükselen bir piyasada yükselişe yüksek bir hacmin eşlik etmesi beklenirken düşüş genellikle hacimde bir daralmayı da beraberinde getirir. Bu yönüyle volume trendin dönüş yerlerinin tespitinde de yardımcı olabilmektedir. Şöyle ki, düşüşün sonlarına doğru iyice azalan işlem hacmindeki artış toparlanmanın başladığına işaret edebileceği gibi yükselişin sonlarına doğru azalmaya başlayan hacimde gevşemenin habercisi olabilmektedir.

Buradan hareketle, artmakta olan hacmin azalmaya başlaması ve belirli bir periyot boyunca azalan ya da sabit kalan bir hacmin artmaya başlaması, var olan trende ters yönlü bir gelişmenin olabileceğinin ilk sinyalleri olarak kabul edilebilir.

Destek direnç bölgelerinde de durum aynıdır. Bir destek ya da direncin kırılması sırasında artan işlem hacminin azalmaya başlaması bir başka destek ya da dirence yaklaşıldığı ve burada oyalanılacağı belki de dönüleceği anlamına gelirken, artışın aynı hızda devamı trendin aynı yönlü olarak devam edeceğinin bir belirtisidir.

Aynı zamanda yüksek işlem hacmi gerek yükselişlerde gerekse düşüşlerde olsun trendin aynı yönde devam edeceği kanaatinin yaygın olduğunu gösterir. Tersine düşük işlem hacimleri fiyatların dip oluşturduğu durumlar bir kenara bırakılacak olursa bir kararsızlık halini işaret eder.

Dow kuramında da üzerinde oldukça durulan volume için "trendi onaylamalıdır" denmektedir. Bunun kuramsal açıdan anlamı bir yükseliş trendi içinde yükselişler sırasında işlem hacminin artması gevşemelerde ise azalmasıdır. Tersine bir düşüş trendi içerisinde de düşüş sırasında işlem hacmi artmalı yükselişlerde azalmalıdır.

Bu indikatör tek başına teknik bir strateji belirlemede yeterli değildir. O nedenle diğer indikatörlerle birlikte kullanılır.

VolumeTLIndicator(String, SymbolPeriod)

VolumeTL(VOLTL), fiyatlardaki değişimin yönüyle işlem hacmini ilişkilendiren bir göstergedir. Basit bir şekilde senedin kapanışının bir önceki kapanışın üzerinde olmasıyla o gün yaratılan işlem hacmi pozitif olarak işleme dahil edilirken, kapanışın bir önceki kapanışın altında kaldığı günlerde hacim negatif olarak hesaba katılmaktadır.

Genel kanı, VOLTL'deki değişimlerin fiyatlardaki değişimlerden önce olduğu şeklindedir. Bunun ardında yatan varsayım ise diğer volume index göstergelerinde olduğu gibi fiyatlardaki salınımların ve hacmin azaldığı dönemlerde bilinçli yatırımcının alıma geçmesi ile VOLTL'nin yükselmeye başlamasıdır. Fiyatların yükselişi ile senede ilgisi artan küçük yatırımcıların da gelmesiyle VOLTL'deki artış hızlanacaktır.

VOLTL'nin fiyatlarla uyum içinde olması beklenir. Yani VOLTL'deki artışı fiyatlardaki artışın, düşüşü de fiyatlardaki düşüşün izlemesi gerekir. Gösterge ile fiyatlar arasındaki uyumsuzluklar trendin kısa bir süre sonra gösterge yönünde değişeceği şeklinde yorumlanır.

Bir diğer uyumsuzluksa fiyatlardaki yükselişe VOLTL'nin yeni tepelerle düşüşe de yeni diplerle eşlik edememesidir. Çünkü yükselen bir piyasada VOLTL'deki yeni oluşan dip ve tepelerin bir önceki dip ve tepelerden daha yüksekte oluşması beklenirken, alçalan bir piyasada da VOLTL'deki dip ve tepelerin bir önceki dip ve tepelerden daha aşağıda olması beklenir. Bunun gerçekleşmediği durumlar uyumsuzluk olarak kabul edilir.

Bu indikatör tek başına teknik bir strateji belirlemede yeterli değildir. O nedenle diğer indikatörlerle birlikte kullanılır.

WildersIndicator(String, SymbolPeriod, OHLCType, Int32)

Welles Wilder's Smoothing Average (WWS) J. Welles Wilder, Jr. tarafından geliştirilmiştir. Bu gösterge fiyat hareketlerini yumuşatır, yükseliş ve düşüş eğilimlerini tanımlamamıza ve belirlememize yardımcı olmak için kullanılır.

Fiyat hareketli ortalamanın üzerinde hareket ettiğinde, yükseliş sinyalleri, fiyat hareketli ortalamanın altına düştüğünde, düşüş sinyalleri verilir.

Örnek:

```
decimal data;
bool emir;

if (emir != null)
{
    if (data > wilders.CurrentValue)
    {
       if (emir == false)
       {
           SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Buy));
           Debug("Alış emri verildi.");
           Debug("Close:" + barData.BarData.Close);
           Debug("Wilders:" + wilders.CurrentValue);
       }
       emir = true;
    }
   if (data<wilders.CurrentValue)
   {
       if (emir == true)
       {
            SendMarketOrder(Symbol, OrderQuantity, (OrderSide.Sell));
            Debug("Satış emri verildi.");
            Debug("Close:" + barData.BarData.Close);
            Debug("Wilders:" + wilders.CurrentValue);
       }
       emir = false;
    }
}
 else
{
    emir = true;
}
```

```
data = wilders.CurrentValue;
```

** wilders.** CurrentValue: Wilders göstergesinin o andaki değeri**

WMAIndicator(String, SymbolPeriod, OHLCType, Int32)

Ağırlıklı hareketli ortalamalar sinyalleri açısından basit hareketli ortalama ile aynıdır. Ayrıştığı nokta ise ortalamanın hesaplanmasında kullanılan yöntemdir. Basit hareketli ortalamada uzun dönemlerde geçmiş hareketlerin ortalamayı etkileme sorunu ağırlıklı hareketli ortalamada giderilmeye çalışılmaktadır. Seçilen periyoda göre bugünkü fiyata daha çok önem vererek geçmişe doğru da önem derecesini azaltarak hesaplama gerçekleştirir.

Örneğin grafikte periyot 21 seçildiğinde bugünkü hareketin kapanış fiyatı 21 ile çarpılırken bir önceki hareketin kapanış fiyatı 20 ile çarpılır. 21 gün önceki hareketin kapanış fiyatına ise ağırlık verilmez. Böylece geçmiş hareketin daha yakın zamanı etkileme oranı aza indirgenmeye çalışılmaktadır.

Örnek:

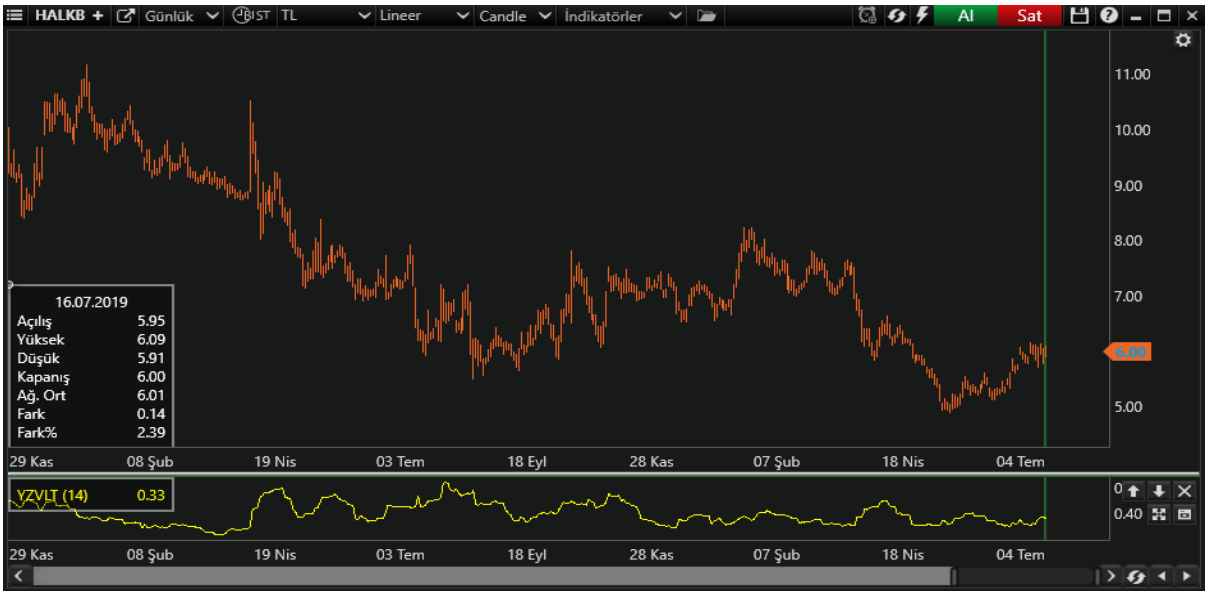
```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, wma, OHLCType.Close)
{
    SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
    Debug("Alış Emri Gönderildi");
    Debug("Close:" + barData.BarData.Close);
    Debug("Wma:" + wma.CurrentValue);
}
if (CrossBelow(barDataModel, wma, OHLCType.Close))
{
    SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
    Debug("Satış Emri Gönderildi");
    Debug("Close:" + barData.BarData.Close);
    Debug("Wma:" + wma.CurrentValue);
}
```

** wma.** CurrentValue: Weighted Moving Avegare göstergesinin o andaki değeri**

YZVLTIndicator(String, SymbolPeriod, OHLCType, Int32)

Yhang-Zhang Volatility göstergesinin değerini hesaplamak için kullanılır.

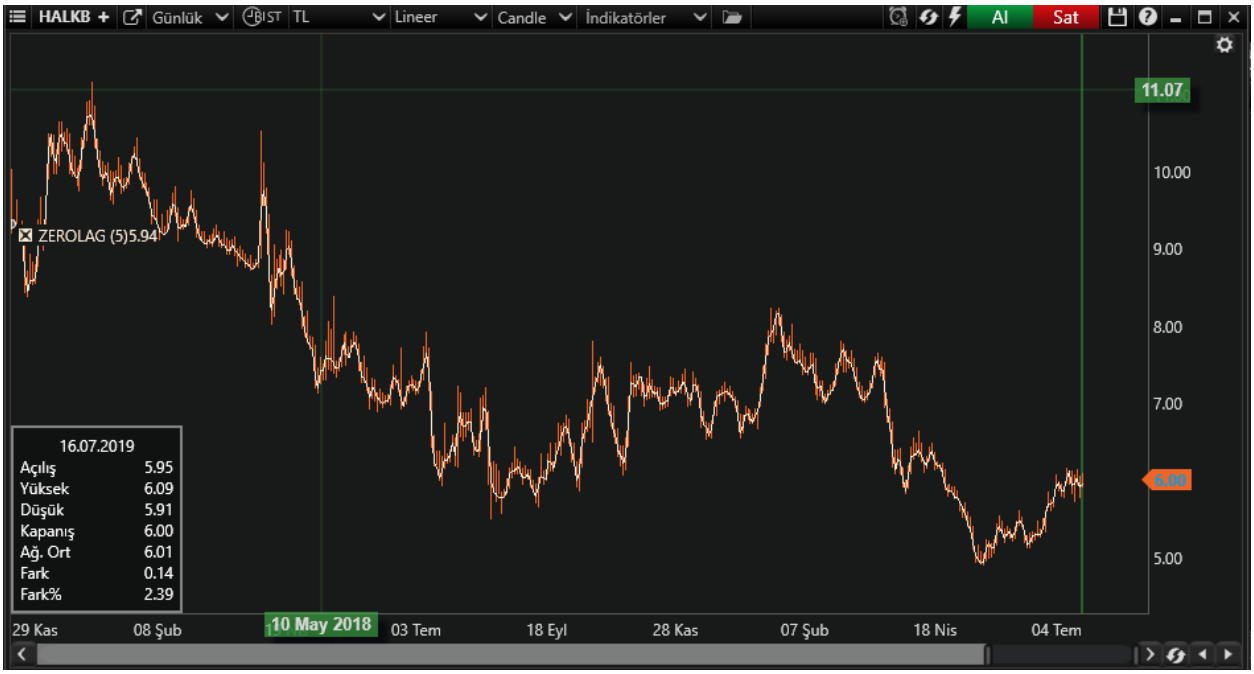


Senedin açılışlardaki fiyat sıçramalarını da hesaba katarak volatilite hesaplaması yapan bir indikatördür.

Bu indikatör tek başına teknik bir strateji belirlemede yeterli değildir. O nedenle diğer indikatörlerle birlikte kullanılır.

ZerolagIndicator(String, SymbolPeriod, OHLCType, Int32)

Zero lag exponential moving average göstergesi John Ehlers ve Ric Way tarafından yaratıldı. Amaç zaman içinde ortalama fiyat veren göstergelerin ardından tüm eğilim ile ilgili doğal gecikmeyi ortadan kaldırmaktır.



Buradaki fikir, normal üstel bir hareketli ortalama (EMA) hesaplaması yapmaktır ancak normal veriler üzerinde yapmak yerine gecikmiş bir veri üzerindedir. Veriler, günler öncesindeki 'gecikme' durumundan çıkarılır, böylece hareketli ortalamanın kümülatif etkisi ortadan kaldırılmaya (veya denenmeye) çalışılır.

Örnek:

```
var barDataModel = GetBarData();

if (CrossAbove(barDataModel, zeroLag, OHLCType.Close))
{
    SendMarketOrder(Symbol, BuyOrderQuantity, OrderSide.Buy);
    Debug("Alış Emri Gönderildi");
    Debug("Close:" + barData.BarData.Close);
    Debug("Zerolag:" + zeroLag.CurrentValue);
}
if (CrossBelow(barDataModel, zeroLag, OHLCType.Close))
{
    SendMarketOrder(Symbol, SellOrderQuantity, OrderSide.Sell);
    Debug("Satış Emri Gönderildi");
    Debug("Close:" + barData.BarData.Close);
    Debug("Zerolag:" + zeroLag.CurrentValue);
}
```

** zeroLag.** CurrentValue: Zero lag Exponential Moving Average göstergesinin o andaki değeri**

Örnek Stratejiler

Basit RSI_SMA Stratejisi

```
namespace Matriks.Lean.Algotrader
{
    public class basitRSISMA : MatriksAlgo
    {
        //strateji ismini burada deklare ediyoruz. Dosyadaki isimle stratejide yazılan
        //isim tamamen aynı olmalıdır.(küçük büyük harf duyarlı)

        //canlı, backtest ve backtest optimization kısımlarında değiştirilebilir olması
        //istenilen parametreler bu bölümde yazılır

        [SymbolParameter("GARAN")]
        public string Symbol; //Sembol ismi

        [Parameter(SymbolPeriod.Day)]
        public SymbolPeriod SymbolPeriod;
        //Stratejiyi çalıştırmak istediğimiz bar periyodu

        [Parameter(100)]
        public int BuyOrderCount;
        //alım miktarı için kullanacağımız parametre

        [Parameter(100)]
        public int SellOrderCount;
        //satım miktarı için kullanacağımız parametre

        [Parameter(10)]
        //Moving average periyodu için kullanacağımız parametre
        public int MovPeriod;

        [Parameter(2)]
        //RSI periyodu için kullanacağımız parametre
        public int RsiPeriod;
```

```
RSI rsi;
//RSI indikatörü türünde rsi isminde bir obje tanımlıyoruz

SMA sma10;
//SMA indikatörü türünde sma10 isminde bir obje tanımlıyoruz

SMA sma200;
//SMA indikatörü türünde sma200 isminde bir obje tanımlıyoruz

public override void OnInit()
//Strateji ilk çalıştırıldığında bu fonksiyon tetiklenir. Tüm sembole kayit
// işlemleri, indikator ekleme, haberlere kayıt olma işlemleri burada yapılır.
{
        //tanımladığımız objelere indikatör tanımlarını ve gerekli değerleri
        //atıyoruz

        sma10 = SMAIndicator(Symbol, SymbolPeriod, OHLCType.Close, MovPeriod);
        sma200 = SMAIndicator(Symbol, SymbolPeriod, OHLCType.Close, 200);
        rsi = RSIIndicator(Symbol, SymbolPeriod, OHLCType.Close, RsiPeriod);

        AddSymbol(Symbol, SymbolPeriod); //Sembol ve periyoduna kayıt

        WorkWithPermanentSignal(true);
        // Algoritmanın kalıcı veya geçici sinyal ile çalışıp çalışmayacağını
        //belirliyoruz. True değer, algoritmanın sadece yeni bar açılışlarında
        //çalışmasını sağlar, bu fonksiyonu çağırmazsak veya false olarak
        //belirlersek her işlem olduğunda algoritma tetiklenecektir.

        SendOrderSequential(true);
        //emirlerin sıralı gönderilmesini sağlar. Yani,strateji önce al komutu
        //bekler, sonra sat komutu gelene kadar piyasaya emir göndermez. False
        //atarsak ya da bu fonksiyonu yazmazsak stratejimiz üst üste al veya sat
        // emri gönderebilir.
}

public override void OnDataUpdate(BarDataEventArgs barData)
//kayıt olunan sembol veya indikatörler güncellendikçe bu fonksiyon tetiklenir.
//Dolayısıyla asıl al sat stratejisini yazacağımız bölümdür
{
        if (rsi.CurrentValue < 10 && barData.BarData.Close > sma200.CurrentValue)
        //statejimizin gövdesini oluşturan sorgu. RSI (parametrelerde
```

```
//tanımladığımız 2 periyotluk) değeri 10 'un altında ve seçilen sembolün
//kapanış değeri 200 periyotluk basit ortalamasının üstündeyse aşağıdaki
//kod bloğu çalışır
{
        SendMarketOrder(Symbol, BuyOrderCount, OrderSide.Buy);
        //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa
        //fiyatından alış emri gönderir

        Debug("Close = " + barData.BarData.Close);
        //bar kapanışını debug penceresine basar

        Debug("200 SMA = " + sma200.CurrentValue);
        //200 günlük basit ortalamayı debug penceresine basar

        Debug("rsi = " + rsi.CurrentValue);
        //RSI değerini debug penceresine basar

        Debug("Alış Emri Gönderildi");
        //"" içerisinde bulunan ifadeyi debug penceresine basar
}
if (barData.BarData.Close > sma10.CurrentValue)
//Seçilen sembolün bar kapanış değeri 10 periyotluk basit ortalamasının
//üstündeyse aşağıdaki kod bloğu çalışır
{
        SendMarketOrder(Symbol, SellOrderCount, OrderSide.Sell);
        //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa
        //fiyatından satış emri Gönderir

        Debug("Close = " + barData.BarData.Close);
        //bar kapanışını debug penceresine basar

        Debug("10 SMA = " + sma10.CurrentValue);
        //10 günlük basit ortalamayı debug penceresine basar

        Debug("rsi = " + rsi.CurrentValue);
        //RSI değerini debug penceresine basar

        Debug("Satış Emri Gönderildi");
        //"" içerisinde bulunan ifadeyi debug penceresine basar
}
else
```

```
            {
                //Bu bölümde kullanıcının stratejinin ne durumda olduğunu daha net
                //anlayabilmesi için, açıklayıcı debug print fonksiyonları
                //bulunmaktadır.
                Debug("Beklemede");

                if (rsi.CurrentValue>10)
                {
                    Debug("Rsi ALIS kosulu gerceklesmedi");
                    Debug("RSI = " + rsi.CurrentValue + " > 10");
                }
                if (barData.BarData.Close < sma200.CurrentValue)
                {
                    Debug("SMA ALIS kosulu gerceklesmedi");
                    Debug("Close = " + barData.BarData.Close + " < " + "sma200 = " +
                                                            sma200.CurrentValue);
                }
                if (barData.BarData.Close < sma10.CurrentValue)
                {
                    Debug("SMA SATIS kosulu gerceklesmedi");
                    Debug("Close = " + barData.BarData.Close + " < " + "10 SMA" +
                                                             sma10.CurrentValue);
                }
            }
        }
    }
}
```

Basit HullMA-TMA Stratejisi

```
namespace Matriks.Lean.Algotrader
{
 //strateji ismini burada deklare ediyoruz. Dosyadaki isimle stratejide yazılan isim
 //tamamen aynı olmalıdır. (küçük büyük harf duyarlı)

        public class BasitTMAHullMAStratejisi : MatriksAlgo
        {
                //canlı, backtest ve backtest optimization kısımlarında değiştirilebilir
                //olması istenilen parametreler bu bölümde yazılır
                // Strateji çalıştırılırken kullanacağımız parametreler. Eğer sembolle
               //ilgili bir parametre ise,

                [SymbolParameter("XU100")]
                public string Symbol;//Sembol ismi

                [Parameter(SymbolPeriod.Day)]
                public SymbolPeriod SymbolPeriod;
                //Stratejiyi çalıştırmak istediğimiz bar periyodu

                [Parameter(22)]
                public int HullMAPeriod;
                //Moving average periyodu için kullanacağımız parametre

                [Parameter(12)]
                public int TmaPeriod;
                //RSI periyodu için kullanacağımız parametre

                [Parameter(100)]
                public int BuyOrderCount;
                //alım miktarı için kullanacağımız parametre

                [Parameter(100)]
                public int SellOrderCount;
                //satım miktarı için kullanacağımız parametre

                TMA tma;
                //TMA indikatörü türünde tma isminde bir obje tanımlıyoruz
```

```
HullMA hullMA;
//HullMA indikatörü türünde hullMA isminde bir obje tanımlıyoruz

public override void OnInit()
{
     //Strateji ilk çalıştırıldığında bu fonksiyon tetiklenir. Tüm sembole
     //kayıt işlemleri, indikator ekleme, tanımladığımız objelere
     //indikatör tanımlarını ve gerekli değerleri atıyoruz
     hullMA = HullMAIndicator(Symbol, SymbolPeriod, OHLCType.Close,
                                                              HullMAPeriod);
     tma = TMAIndicator(Symbol, SymbolPeriod, OHLCType.Close, TmaPeriod);

     //Sembol ve periyoduna kayıt
     AddSymbol(Symbol, SymbolPeriod);

    // Algoritmanın kalıcı veya geçici sinyal ile çalışıp çalışmayacağını
    //belirliyoruz. true değer, algoritmanın sadece yeni bar açılışlarında
    //çalışmasını sağlar, bu fonksiyonu çağırmazsak veya false olarak
    //belirlersek her işlem olduğunda algoritma tetiklenecektir.
    WorkWithPermanentSignal(true);

    //Eger backtestte emri bir al bir sat seklinde gonderilmesi isteniyor
   // bu true set edilir. Alttaki satırı silerek veya false geçerek
   //emirlerin sirayla gönderilmesini engelleyebilirsiniz.
    SendOrderSequential(true);
}

//Kayıt olunan sembol veya indikatörler güncellendikçe bu fonksiyon
//tetiklenir.
//Dolayısıyla asıl al/sat stratejisini yazacağımız bölümdür
public override void OnDataUpdate(BarDataEventArgs barData)
{
    //Bu koşul alım emri içindir. Eğer grafikte fiyat çubukları hullMA
    //bandını yukarı kırarsa ve o anki hullMA değeri, tma değerinin
    // altındaysa al emri gönderilecek.

    if (CrossAbove(hullMA, barData.BarData.Close) &&
                                        hullMA.CurrentValue<tma.CurrentValue)
        {
            //Parametrelerde belirlenen sembolden, belirlenen miktarda,
            //piyasa fiyatından satış emri gönderir
```

```
                    SendMarketOrder(Symbol, BuyOrderCount, OrderSide.Buy);

                    //bar kapanışını debug penceresine basar
                    Debug("Close = " + barData.BarData.Close);

                    //HullMA değerini debug penceresine basar
                    Debug("HullMA = " + hullMA.CurrentValue);

                    //TMA değerini debug penceresine basar
                    Debug("TMA = " + tma.CurrentValue);

                    //"" içerisinde bulunan ifadeyi debug penceresine basar
                    Debug("Alış Emri Gönderildi");
                }
                //Bu koşul satım emri içindir. Eğer grafikte fiyat çubukları hullMA
                // bandını aşağı kırarsa ve o anki hullMA değeri, tma değerinin
                // üstündeyse sat emri gönderilecek.

                if (CrossBelow(hullMA, barData.BarData.Close) &&
                                            hullMA.CurrentValue>tma.CurrentValue)
                {
                    //Parametrelerde belirlenen sembolden, belirlenen miktarda,
                    // piyasa fiyatından satış emri gönderir
                    SendMarketOrder(Symbol, SellOrderCount, OrderSide.Sell);

                    //bar kapanışını debug penceresine basar
                    Debug("Close = " + barData.BarData.Close);

                    //HullMA değerini debug penceresine basar
                    Debug("HullMA = " + hullMA.CurrentValue);

                    //TMA değerini debug penceresine basar
                    Debug("TMA = " + tma.CurrentValue);

                    //"" içerisinde bulunan ifadeyi debug penceresine basar
                    Debug("Satış Emri Gönderildi");
                }
            }
        }
    }
```

RSI Indikatörünü MOST İçinde Kullanarak Oluşturulan Strateji

```
namespace Matriks.Lean.Algotrader
{
        public class MOSTRSIStratejisi : MatriksAlgo
        //strateji ismini burada deklare ediyoruz. Dosyadaki isimle stratejide yazılan
         // isim tamamen aynı olmalıdır. (küçük büyük harf duyarlı)
        {
            // Strateji çalıştırılırken kullanacağımız parametreler. Eğer sembolle ilgili
            // bir parametre ise, "SymbolParameter" ile, değilse "Parameter" ile tanımlama
            //yaparız. Parantez içindeki değerler default değerleridir.

            [SymbolParameter("GARAN")]
            public string Symbol;//Sembol ismi

            [Parameter(SymbolPeriod.Day)]
            public SymbolPeriod SymbolPeriod;
            //Stratejiyi çalıştırmak istediğimiz bar periyodu

            [Parameter(100)]
            public int BuyOrderCount;
            //alım miktarı için kullanacağımız parametre

            [Parameter(100)]
            public int SellOrderCount;
            //satım miktarı için kullanacağımız parametre

            [Parameter(14)]
            public int periodRsi;
            //RSI periyodu için kullanacağımız parametre

            [Parameter(3)]
            public int periodMost;
            //MOST periyodu için kullanacağımız parametre

            [Parameter(2)]
            public decimal percentage;
            //MOST yüzde paremetresi için kullanacağımız parametre
```

```
//Kullanacağımız indikatör obje tanımları
RSI rsi;
MOST most;

// Strateji ilk çalıştırıldığında bu fonksiyon tetiklenir. Tüm sembole kayit
//işlemleri,indikator ekleme, haberlere kayıt olma işlemleri burada yapılır.

public override void OnInit()
{
    //tanımladığımız objelere indikatör tanımlarını ve gerekli değerleri
    //atıyoruz
    rsi = RSIIndicator(Symbol, SymbolPeriod, OHLCType.Close, periodRsi);
    most = MOSTIndicator(rsi, periodMost, percentage, MovMethod.Exponential);

    //Sembol ve periyoduna kayıt
    AddSymbol(Symbol, SymbolPeriod);

    // Algoritmanın kalıcı veya geçici sinyal ile çalışıp çalışmayacağını
    //belirliyoruz. true değer, algoritmanın sadece yeni bar açılışlarında
    //çalışmasını sağlar, bu fonksiyonu çağırmazsak veya false olarak
    //belirlersek her işlem olduğunda algoritma
    //tetiklenecektir.
    WorkWithPermanentSignal(true);

    //Eger backtestte emri bir al bir sat seklinde gonderilmesi isteniyor bu
    //true set edilir.
    //Alttaki satırı silerek veya false geçerek emirlerin sirayla
    //gönderilmesini engelleyebilirsiniz.
    SendOrderSequential(true);
}
// Eklenen sembollerin bardata'ları ve indikatorler güncellendikçe bu
//fonksiyon tetiklenir.
//Dolayısıyla asıl al/sat stratejisini yazacağımız bölümdür

public override void OnDataUpdate(BarDataEventArgs barData)
{
    //Bu koşul alım emri içindir. Eğer grafikte MOST'un EXMOV bandı
    //most bandını yukarı kırarsa al emri gönderilecek.
    if (CrossAbove(most.CurrentValue, most.ExMOV))
    {
```

```
                //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa
                //fiyatından alış emri gönderir
                SendMarketOrder(Symbol, BuyOrderCount, (OrderSide.Buy));

                //"" içerisinde bulunan ifadeyi debug penceresine basar
                Debug("Alış Emri Gönderildi");

                //EXMOV değerini debug penceresine basar
                Debug("exmov:" + Math.Round(most.ExMOV.CurrentValue, 2));

                //MOST değerini debug penceresine basar
                Debug("most:" + Math.Round(most.CurrentValue, 2));
            }
            //Bu koşul satım emri içindir. Eğer grafikte MOST'un EXMOV bandı
            //most bandını aşağı kırarsa sat emri gönderilecek.
            if (CrossBelow(most.CurrentValue, most.ExMOV))
            {
                //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa
                //fiyatından satış emri gönderir
                SendMarketOrder(Symbol, SellOrderCount, (OrderSide.Sell));

                //"" içerisinde bulunan ifadeyi debug penceresine basar
                Debug("Satış Emri Gönderildi");

                //EXMOV değerini debug penceresine basar
                Debug("exmov:" + Math.Round(most.ExMOV.CurrentValue, 2));

                //MOST değerini debug penceresine basar
                Debug("most:" + Math.Round(most.CurrentValue, 2));
            }
        }
    }
}
```

Basit_Bollinger- RSI Stratejisi

```
namespace Matriks.Lean.Algotrader
{
 //strateji ismini burada deklare ediyoruz.
 //Dosyadaki isimle stratejide yazılan isim tamamen aynı olmalıdır. (küçük büyük harf
duyarlı)
        public class BolRsiStratejisi : MatriksAlgo
        {
            //canlı, backtest ve backtest optimization kısımlarında değiştirilebilir
            //olması istenilen parametreler bu bölümde yazılır

            [SymbolParameter("GARAN")]
            public string Symbol;//Sembol ismi

            [Parameter(SymbolPeriod.Day)]
            public SymbolPeriod SymbolPeriod;
            //Stratejiyi çalıştırmak istediğimiz bar periyodu

            [Parameter(100)]
            public int BuyOrderQuantity;
            //Alım miktarı için kullanacağımız parametre

             [Parameter(100)]
            public int SellOrderQuantity;
            //Satım miktarı için kullanacağımız parametre

            [Parameter(11)]
            //RSI periyodu için kullanacağımız parametre
            public int RsiPeriod;

            [Parameter(MovMethod.E)]
            public MovMethod MovMethod;
            //BOLLINGER indikatörünü hesaplamak için kullanacağımız hareketli ortalama
            //metodu parametre

            [Parameter(15)]
            public int BolPeriod;
            //BOLLINGER periyodu için kullanacağımız parametre
```

```
[Parameter(2)]
public decimal StandartDeviation;
//BOLLINGER indikatörünü hesaplamak için kullanacağımız standart sapma değeri
// indikator tanımları.

BOLLINGER bollinger;
//BOLLINGER indikatörü türünde bollinger isminde bir obje tanımlıyoruz

RSI rsi;
//RSI indikatörü türünde rsi isminde bir obje tanımlıyoruz

/// Strateji ilk çalıştırıldığında bu fonksiyon tetiklenir. Tüm sembole kayit
//işlemleri, indikator ekleme, haberlere kayıt olma işlemleri burada yapılır.
public override void OnInit()
{
    //tanımladığımız objelere indikatör tanımlarını ve gerekli değerleri
    //atıyoruz
    rsi = RSIIndicator(Symbol, SymbolPeriod, OHLCType.Close, RsiPeriod);
    bollinger = BollingerIndicator(Symbol, SymbolPeriod, OHLCType.Close,
                                   BolPeriod,StandartDeviation, MovMethod);
    //Sembol ve periyoduna kayıt
    AddSymbol(Symbol, SymbolPeriod);

    //Algoritmanın kalıcı veya geçici sinyal ile çalışıp çalışmayacağını
    //belirleyen fonksiyondur.
    // true geçerseniz algoritma sadece yeni bar açılışlarında çalışır, bu
    //fonksiyonu çağırmazsanız veya false geçerseniz her işlem olduğunda
    //algoritma tetiklenir.
    WorkWithPermanentSignal(true);

    //Eger emri bir al bir sat seklinde gonderilmesi isteniyor bu true set
    //edilir.
    //Alttaki satırı silerek veya false geçerek emirlerin sirayla
    //gönderilmesini engelleyebilirsiniz.
    SendOrderSequential(true);
}

//Eklenen sembollerin bardata'ları ve indikatorler güncellendikçe bu
//fonksiyon tetiklenir.

public override void OnDataUpdate(BarDataEventArgs barData)
```

```
        {
            if (CrossAbove(rsi, rsi.DownLevel) && bollinger.BollingerDown>
                                                        barData.BarData.Close)

            //Bu koşul alım emri içindir. Eğer grafikte rsi downlevel bandını yukarı
            //kırarsa ve son kapanış fiyatı BollingerDown bandından daha büyük bir
            //değerse al emri gönderilecek.
            {
                SendMarketOrder(Symbol, BuyOrderQuantity, (OrderSide.Buy));
                //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa
                //fiyatından alış emri gönderir

                Debug("bollinger = " + bollinger.CurrentValue);
                //bollinger değerini debug penceresine basar

                Debug("rsi = " + rsi.CurrentValue);
                //RSI değerini debug penceresine basar

                Debug("Alış Emri Gönderildi");
            }
            if (CrossBelow(rsi, rsi.UpLevel) && bollinger.Bollingerup<
                                                        barData.BarData.Close)

            //Bu koşul satım emri içindir. Eğer grafikte rsi upleve bandını aşağı
            //kırarsa ve son kapanış fiyatı Bollingerup bandından daha küçük bir
            //değerse sat emri gönderilecek.
            {
                SendMarketOrder(Symbol, SellOrderQuantity, (OrderSide.Sell));
                //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa
                //fiyatından alış emri gönderir

                Debug("bollinger = " + bollinger.CurrentValue);
                //bollinger değerini debug penceresine basar

                Debug("rsi = " + rsi.CurrentValue);
                //RSI değerini debug penceresine basar

                Debug("Satış Emri Gönderildi");
            }
        }
    }
```

```
}
```

Fiyat 7 gun ustu

```
namespace Matriks.Lean.Algotrader
{
        public class Fiyat7gunustu : MatriksAlgo
        //strateji ismini burada deklare ediyoruz. Dosyadaki isimle stratejide yazılan
        //isim tamamen aynı olmalıdır. (küçük büyük harf duyarlı)
        {
            //canlı, backtest ve backtest optimization kısımlarında değiştirilebilir
            //olması istenilen parametreler bu bölümde yazılır

            [SymbolParameter("AKBNK")]
            public string Symbol;
            //Sembol ismi

            [Parameter(SymbolPeriod.Min5)]
            public SymbolPeriod SymbolPeriod;
            //Stratejiyi çalıştırmak istediğimiz bar periyodu

            [Parameter(100)]
            public int BuyOrderCount;
            //alım miktarı için kullanacağımız parametre

            [Parameter(100)]
            public int SellOrderCount;
            //satım miktarı için kullanacağımız parametre

            public override void OnInit()
            //Strateji ilk çalıştırıldığında bu fonksiyon tetiklenir. Tüm sembole kayit
            //işlemleri, indikator ekleme, haberlere kayıt olma işlemleri burada yapılır.
            {
                AddSymbol(Symbol, SymbolPeriod);
                //Sembol ve periyoduna kayıt

                AddSymbolMarketData(Symbol);
                //Sembolu Marketdata, yani yüzeysel veri akışına kayıt ediyoruz. Bu veri
                //seti içerisinde temel ve teknik analiz öğeleri bulunmaktadır. Daha
```

```
        //detaylı tanım için strateji yapısı altında AddSymbolMarketData
        //fonksiyonunun tanımına bakınız.

        WorkWithPermanentSignal(true);
        //Algoritmanın kalıcı veya geçici sinyal ile çalışıp çalışmayacağını
        //belirliyoruz. true değer, algoritmanın sadece yeni baılışlarında
        //çalışmasını sağlar, bu fonksiyonu çağırmazsak veya false olarak
        //belirlersek her işlem olduğunda algoritma tetiklenecektir.

        SendOrderSequential(true);
        //emirlerin sıralı gönderilmesini sağlar. Yani strateji önce al komutu
        //bekler, sonra sat komutu gelene kadar piyasaya emir göndermez. False
        //atarsak ya da bu fonksiyonu yazmazsak stratejimiz üst üste al veya sat
        //emri gönderebilir.
    }

    public override void OnDataUpdate(BarDataEventArgs barData)
    //kayıt olunan sembol veya indikatörler güncellendikçe bu fonksiyon
    //tetiklenir. Dolayısıyla asıl al/sat stratejisini yazacağımız bölümdür
    {
        var yedigun = GetMarketData(Symbol, SymbolUpdateField.WeekClose);
        //yedigun olarak tanımladığımız objeye, enstrümanın 7 gün önceki kapanış
        //fiyatını atar

        var close = barData.BarData.Close;
        //close olarak tanımladığımız objeye güncel bar kapanış değerini atar


            if (close > yedigun)
            {
                SendMarketOrder(Symbol, BuyOrderCount, OrderSide.Buy);
                //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa
                //fiyatından alış emri gönderir

                Debug("Close = " + close);
                //Close değişkenine atadığımız değeri (barData.BarData.Close, yani
                //bar kapanışı) debug ekranına basar

                Debug(Symbol + " 7 Seans onceki kapanis = " + yedigun);
                //Sembol ismine, tırnak içindeki yazılı açıklamayı ve yedigun
                //objesinde bulunan değeri yazarak debug ekranına basar
```

```
                    Debug("Alış Emri Gönderildi");
                }
                if (close < yedigun)
                {
                    SendMarketOrder(Symbol, SellOrderCount, OrderSide.Sell);
                    //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa
                    //fiyatından satış emri gönderir

                    Debug("Close = " + close);
                    //Close değişkenine atadığımız değeri (barData.BarData.Close, yani
                    //bar kapanışı) debug ekranına basar

                    Debug(Symbol + " 7 Seans onceki kapanis = " + yedigun);
                    //Sembol ismine, tırnak içindeki yazılı açıklamayı ve yedigun
                    //objesinde bulunan değeri yazarak debug ekranına basar

                    Debug("Satış Emri Gönderildi");
                }
            }
        }
}
```

4.Yeni Eklenen Özellikler

Versiyon 3.5.0.1

Çıktı Parametreleri (Output): Canlı Algo çalıştırdığımızda açılan Rapor penceresinde, Loglar, Debug, Kod diye giden aşağıdaki tablere Çıktı Parametreleri tab'i özelliği eklenmiştir. Strateji kodu içerisinde tanımlanması gerekmektedir. Strateji içerisinde istediğimiz değişkeni bu tab'e canlı olarak iletmemize ve görüntülememize olanak sağlamaktadır. Debug penceresi yerine kullanılabilir. Seriyukarı, svmfiyatRSI, LogisticReg, Derinlik3Timer, BolRsiStratejisi, BasitRSI_SMA ve MostBitmex stratejilerinde örneklendirilmiştir.

Örnek:

```
public class seriyukari_output_release : MatriksAlgo

{

      …

        [Output]
        public decimal sonBarKapanisi;
        [Output]
        public decimal oncekiBarKapanisi;
        [Output]
        public int upcounter;
        [Output]
        public int downcounter;
}
…
public override void OnDataUpdate(BarDataEventArgs barData)
{
        …
        sonBarKapanisi = barDataModel.Close[barData.BarDataIndex -1];
        oncekiBarKapanisi = barDataModel.Close[barData.BarDataIndex - 2];
        upcounter = upCounter;
        downcounter = downCounter;
}
```

Algo Sentetik Emir Yapısı

- Sentetik emirler tanımlandığında, long ve short pozisyonlar için iki ayrı stop fiyatı hesaplanmaya başlandı.

Eski yapıda, emir tanımlandığı andaki pozisyona göre tek bir stop noktası hesaplanıyor ve sentetik emrin yönü de tanım anındaki pozisyona göre belirleniyordu.

Yeni yapı fiyat değişimleri oldukça, hesaplanan iki stop noktasından güncel pozisyona uygun olanı seçiyor ve emir yönünü de güncel pozisyona göre belirliyor.

- Sentetik emir tanımlanması için pozisyonda olma şartı kaldırıldı.

İlk maddedeki değişiklikler pozisyonda olmadan sentetik emir tanımlanmasına olanak sağlıyor.

Borsalardan emir gerçekleşme raporlarının gelmesi zaman alabildiği için, pozisyon açan bir emir gönderilmiş olsa dahi strateji pozisyona girmemiş olabiliyor. Bu ise eski yapıda emir cevabı beklenmesini gerekli kılıyordu.

- Sentetik emirler güncellenebilir hale getirildi. Tanımlı sentetik emirlerin iptal edilebilmesi için fonksiyonlar eklendi.

Eski yapıda tanımlanan bir sentetik emir, stop koşulu gerçekleşene kadar veya strateji durdurulana kadar kalmaktaydı. Aynı sembol üzerine aynı tip sentetik emirden ikinci bir tanesi tanımlandığında, eski emir kalmakta ve yeni tanımlanan emir dikkate alınmamaktaydı.

Bu durumda özellikle stratejide bir stop emri tanımlandıktan sonra, pozisyonu zıt yöne geçirecek alım/satım işlemleri yapıldığında sorunlar ortaya çıkıyordu.

5.1 Nasıl yapılır/ SSS

Q. **Mevcut varolan periyotlardan farklı bir bar periyotu nasıl kullanırım?**

A. MatriksIQ AlgoTrader'da istediğiniz değerde bir bar periyotu tanımlayabilirsiniz. (şimdilik saniyelik barları desteklememektedir)

Örnek:

```
AddSymbol(Symbol, new PeriodInfo(PeriodType.Minute,2));
//Varsayılan sembol için 2 dakikalık bir bar periyotu tanımlamaktadır.

Dakikadan küçük barlar henüz desteklenmemektedir.
```

Q. **Stratejiyi 2 farklı periyotta çalıştırabilir miyim?**

A. Evet. MatriksIQ istenildiği kadar farklı periyot ve sembol ile çalışmamıza olanak sağlamaktadır. 2 farklı periyot ve 2 farklı enstrüman ile kullanım aşağıda örneklendirilmiştir.

```
public class rsiHareketliOrtalamasi : MatriksAlgo

{
        [SymbolParameter("GARAN")]
        public string Symbol_0;
        [SymbolParameter("AKBNK")]
        public string Symbol_1;
        [Parameter(SymbolPeriod.Min5)]
        public SymbolPeriod SymbolPeriod_5;
        [Parameter(SymbolPeriod.Min10)]
        public SymbolPeriod SymbolPeriod_10;

        //… //

        public override void OnDataUpdate(BarDataEventArgs barData)
        {
                int symbolid_0 = GetSymbolId(Symbol_0);
                int symbolid_1 = GetSymbolId(Symbol_1);
                var barDataModel_0 = GetBarData(Symbol_0, SymbolPeriod.Min5);
                var barDataModel_1 = GetBarData(Symbol_0, SymbolPeriod.Min10);
                var barDataModel_2 = GetBarData(Symbol_1, SymbolPeriod.Min5);
                var barDataModel_3 = GetBarData(Symbol_1, SymbolPeriod.Min10);
                if (symbolid_0 == barData.SymbolId && barDataModel_0.PeriodInfo ==
barData.PeriodInfo)
```

// Yukarıdaki kod asıl sembollerin datalarının ayrıştırıldığı önemli bölümdür. If bölümü, barData.Symbolid, yani bar kapanışında güncellenen datadan (rastgele) gelen id verisi GetSymbolId(Symbol_0) ile atadığımız unique ID ile karşılaştırılıyor. Aynı yöntemi rastgele gelmiş olan `barData.PeriodInfo` ile eşleşmek için de kullanmamız gerekiyor.

```
                {
```

```
                    Close_0 = barDataModel.Close[barData.BarDataIndex-1];
```

// Ancak ve sadece bu id'ler ve Periotlar aynı oldugunda close olarak tanımladığımız yeni değişkene barDataModel.Close[barData.BarDataIndex-1] ile gelen Sembol'ün (yani bu örnekte ilk sembol(GARAN) ve ilk periyot(5 dakika)) bir önceki kapanışı atanıyor. Böylelikle 2 sembolü ve periyotu ayrıştırmış ve gelen doğru data ile eşleştirmiş oluyoruz. Bundan sonra artık Close_0 değişkenini kod içerisinde GARAN, 5dk'lık bir önceki kapanış olarak kullanabiliriz.

```
                }
            }
        }
}
```

Q. **Bir İndikatörün hareketli ortalamasını alabilir miyiz?**

A. Evet, MatriksIQ'da bir indikatörün hareketli ortalamasını hatta indikatörün indikatörünü almak oldukça kolay hale getirilmiştir. Aşağıda RSI indikatörünün hareketli ortalamasını almak için kod içerisinde eklenebilecek satırlar ve eklenmesi gereken bölümler örnek olarak yazılmıştır.

```
public class rsiHareketliOrtalamasi : MatriksAlgo
{
        [Parameter(10)]
        public int MovPeriod;

        MOV movrsi10;
        RSI myrsi;

        public override void OnInit()
        {
                myrsi = RSIIndicator(Symbol, SymbolPeriod, OHLCType.Close,
14);
                movrsi10 = MOVIndicator(myrsi, MovPeriod, MovMethod.Simple);
        }
```

Kısaca, normalde
mov = MOVIndicator(Symbol, SymbolPeriod, OHLCType.Close, MovPeriod, MovMethod.Simple);

şeklinde tanımlayacağımız mov indikatörünün içerisine Symbol, SymbolPeriod, OHLCType.Close parametrelerini silerek (çünkü bunlar zaten myrsi objesinde tanımlı olacak) RSI indikatörü olarak deklare

ettiğimiz rsi objesini yazdığımızda movrsi10 objesi 14 periyotluk bir RSI indikatörünün 10 periyotluk hareketli ortalamasını almış oluyor.

Q. **2 veya daha fazla, farklı sembol kullanarak strateji yazılabilir mi?**

A. Evet. Bununla ilgili örnek strateji Hazır Stratejilerde bulunmaktadır (GAOrt2Hisse *Günlük Ağırlıklı Ortalama, 2 Hisse).

Bu stratejide OnInit() fonksiyonu altına:

```
AddSymbol(Symbol, SymbolPeriod);
AddSymbol(Symbol_1, SymbolPeriod);
```

Yazılarak 2 sembol eklenmiştir. Daha sonra OnDataUpdate(BarDataEventArgs barData) fonksiyonu altına (her bar açılışında çalışacak fonksiyondur)

```
-------------------------------------
int symbolid = GetSymbolId(Symbol);
int symbolid1 = GetSymbolId(Symbol_1);
```

yazarak 2 ayrı unique sembol id saklanır.

```
var barDataModel = GetBarData(Symbol, SymbolPeriod.Min);
var barDataModel_1 = GetBarData(Symbol_1, SymbolPeriod.Min);
```

yazarak 2 ayrı sembol için bar data alınır ve barDataModel ve barDataModel_1 şeklinde isimlendirdiğimiz objelere atar.

```
if (symbolid == barData.SymbolId)
        close = barDataModel.Close[barData.BarDataIndex];

if (symbolid1 == barData.SymbolId)
        close_1 = barDataModel_1.Close[barData.BarDataIndex];
```

Yukarıdaki kod asıl sembollerin datalarının ayrıştırıldığı önemli bölümdür. If bölümü, barData.Symbolid, yani bar kapanışında güncellenen datadan (rastgele) gelen id verisi GetSymbolId(Symbol) ile atadığımız unique ID ile karşılaştırılıyor. Ancak bu id'ler aynı oldugunda close olarak tanımladığımız yeni değişkene barDataModel.Close[barData.BarDataIndex] ile gelen Sembol'ün (yani ilk sembol/enstrüman) kapanışı atanıyor. Böylelikle 2 sembolü ayrıştırmış oluyoruz.

Bu 4 satır çalıştıktan sonra close değişkeninde 1. sembolün, close_1 değişkeninde ise 2. sembolün kapanış değerleri ayrıştılarak kaydedilmiş oluyor.

Yukarıdaki şekilde sembol eklemeye devam ederek, kod içerisinde kullanılan semboller istenildiği kadar çoğaltılabilmektedir.

Q. **if(close < accBands.Lower) seklinde bir ifade tanimladim ama dogru sonuc vermiyor.**

A. İndikatorler liste (array) gibi çalışmaktadır. Dolayisiyla accBands.Lower ifadesi büyüktür/küçüktür operatörüyle kullanabileceğimiz bir ifade değildir. Eğer close objesine atadığımız değerin bir indikatörden büyük/küçük olduğunu öğrenmek istiyorsak indikatörün belli bir noktadaki sabit değeri ile kıyaslamamız mantıklı olacaktır. Dolayısıyla bu ifade if(close < accBands.Lower. CurrentValue) şeklinde yazıldığında istenilen sonuç alınabilecektir.

Q. **Algotrader rapor penceresine fiyat grafiğine ek grafik ekleyebilir miyiz?**

A. Evet. Bu işlem için Fonksiyonlar başlığı altında ki AddChart, AddChartLineName ve Plot fonksiyonlarının kullanılması gerekmektedir.

Örnek:

```
//Grafiğin adını belirlediğimiz kısım
String chartName = "MOST";
AddChart(chartName,2);

//Most indikatöründeki most ve exmov çizgilerinin isimlendirilmesi
AddChartLineName(chartName, 1, "Most");
AddChartLineName(chartName, 2, "ExMov");

//Most indikatörünün çizdirilmesi
Plot(chartName, 1, most.CurrentValue);
Plot(chartName, 2,most.ExMOV.CurrentValue);
```

Q. **AlgoTrader'da önceki barlara nasıl erişim sağlayabilirim?**

A. Önceki barlara erişim sağlayabilmek için "GetBarData" fonksiyonunu kullanmanız gerekmektedir. Fonksiyonlar başlığı altında bu fonksiyonun işlevlerinden bahsetmiştik. Aşağıdaki kod segmentinde detaylı açıklama bulabilirsiniz.

Ek olarak indikatör ve bardata serilerinin önceki datalarına erişebilmek için Ref() fonksiyonu da kullanılabilir. Örn. `Ref(mov,1)` mov indikatörünün bir önceki değerini dönecektir.

```
public override void OnDataUpdate(BarDataEventArgs barData)
{
  //Kayıt olunan bardataya erişimi sağladık. Aşağıdaki kod satırından sonra tüm
  //bardatalara erisim saglayabiliriz

  var barDataModel = GetBarData();
  //Daha sonra istediğimiz verinin indeksine göre bir koşul kurduk.


  barData.BarDataIndex
  //Son bardatanın indeksini bize döndürür.

  barDataModel.Close[barData.BarDataIndex]
  //Son bardatanın kapanış verisini döndürür.

  barDataModel.Open[barData.BarDataIndex-10]
  //Son bardatadan 10 önceki bardatanın açılış verisini döndürür.

if(barDataModel.Close[barData.BarDataIndex] > barDataModel.Open[barData.BarDataIndex-10])
  {
     Debug("Son gelen bar kapanış verisi, 10 bar önceki açılış verisinden daha
     buyuk");
  }
}
```

Q. **Kendi endeksimi yaratıp, bu endeksin moving averajını alabilir miyim?**

A. Evet! MatriksIQ Algo ile kendi oluşturduğunuz değişkenin hatta kendi indikatörünüzün bile moving averajını yaratabilir, bütün diğer indikatörlerin içerisinde de kullanabilirsiniz. Daha fazla bilgi için aşağıdaki örnek koda ve comment'lere bakınız.

```
namespace Matriks.Lean.Algotrader
{
        public class rangeMA : MatriksAlgo
        {
```

```
        // Strateji çalıştırılırken kullanacağımız parametreler. Eğer sembolle
ilgili bir parametre ise,
        // "SymbolParameter" ile, değilse "Parameter" ile tanımlama yaparız.
Parantez içindeki değerler default değerleridir.

        [SymbolParameter("GARAN")]
        public string Symbol;
        [Parameter(SymbolPeriod.Min)]
        public SymbolPeriod SymbolPeriod;
        [Parameter(4)]
        public int MovPeriod;

        MOV mov;
        /// <summary>
        /// Strateji ilk çalıştırıldığında bu fonksiyon tetiklenir. Tüm sembole
kayit işlemleri,
        /// indikator ekleme, haberlere kayıt olma işlemleri burada yapılır.
        /// </summary>
        public override void OnInit()
        {
                AddSymbol(Symbol, SymbolPeriod);
                mov = new MOV(MovPeriod, MovMethod.Exponential);
                SendOrderSequential(true);
                WorkWithPermanentSignal(true);
        }

        /// <summary>
        /// Init islemleri tamamlaninca, bardatalar kullanmaya hazir hale gelince
bu fonksiyon tetiklenir. Data uzerinde bir defa yapilacak islemler icin kullanilir
        /// </summary>
        public override void OnInitComplated()
        {

        }

        /// <summary>
        /// Eklenen sembollerin bardata'ları ve indikatorler güncellendikçe bu
fonksiyon tetiklenir.
        /// </summary>
        /// <param name="barData">Bardata ve hesaplanan gerçekleşen işleme ait
detaylar</param>
```

```
        public override void OnDataUpdate(BarDataEventArgs barData)
        {
                var range = barData.BarData.High - barData.BarData.Low;
                //Range isminde, barin en yuksek degerinden en dusuk degerini
                cikaran yeni bir degisken tanimliyoruz
                mov.Update(range, barData.BarDataIndex, barData.BarData.Dtime);
                //Yukarida (OnInit() içerisinde) oluşturduğumuz moving average'a bu
                değişkeni besliyoruz. Boylelikle range'in 4 periyotluk ussel
                hareketli ortalamasini almis oluyoruz.

                Debug(mov.CurrentValue);
                //Artik range değişkenin ussel hareketli ortalamasi kullanima
                hazirdir. Kodumuz içerisinde kullanabiliriz. Bu satirda hareketli
                ortalamanin anlik değerini debug penceresine basarak kontrol
                sagliyoruz.
        }

        /// <summary>
        /// Gönderilen emirlerin son durumu değiştikçe bu fonksiyon tetiklenir.
        /// </summary>
        /// <param name="barData">Emrin son durumu</param>
        public override void OnOrderUpdate(IOrder order)
        {
                if (order.OrdStatus.Obj == OrdStatus.Filled)
                {

                }
        }
    }
}
```

5.2 Sık Rastlanan Hatalar

Q. **error CS0246: SeriYukari' türü veya ad alanı adı bulunamadı (bir using yönergeniz veya derleme başvurunuz mu eksik?)**

A. Strateji ismi ile stratejinin kod deklerasyonunda yazılan isim arasında uyuşmazlık var, tamamen aynı olduğundan emin olunuz. Küçük büyük harfe duyarlıdır. Örn. SeriYukari olarak isimlendirdiğimiz stratejiyi public class seriyukari : MatriksAlgo şeklinde deklare edersek, bu hatayı alırız. Doğru tanım public class SeriYukari : MatriksAlgo şeklinde olmalıdır.

Q. **error CS1061: 'SymbolPeriod' bir 'Mn' tanımı içermiyor ve 'SymbolPeriod' türünde bir ilk bağımsız değişken kabul eden hiçbir erişilebilir 'Mn' genişletme yöntemi bulunamadı (bir kullanma yönergeniz veya derleme başvurunuz eksik olabilir mi?)**

A. SymbolPeriod öğesi için yanlış metod girilmiştir. SymbolPeriod yazdıktan sonra "." (nokta) yazarsanız, Intellisense sayesinde alabileceği metodları net olarak görebilirsiniz. Bu metodlar Day, Min, Min10, Min120…,Week, Month, Year şeklinde olmalıdır.

Q. **error CS1503: 3 bağımsız değişkeni: 'double' öğesinden 'decimal' öğesine dönüştürülemiyor**

A. Fonksiyonunuzda kullandığınız 3. değişken decimal olarak beklenmekte ama double olarak yazılmıştır. C# casting yaparak sorun çözülebilir.

Örn.

```
StopLoss(Symbol1, SyntheticOrderPriceType.PricePoint,1.10); //Yanlış
StopLoss(Symbol1, SyntheticOrderPriceType.PricePoint,1.10m); //Doğru. 1.10 double öğesi artık decimal olarak tanımlanmıştır (casting)
```

Q. **error CS0019: >' işleci 'decimal' ve 'double' türündeki işlenenlere uygulanamaz**

A. İki ayrı tipte eleman kıyaslanmaya çalışılmaktadır. Double olan elemanı decimal olarak cast ediniz.

Örn.

```
if (macd.Macd.CurrentValue > 0.0001) //Yanlış
if (macd.Macd.CurrentValue > 0.0001m) //Doğru. 0.0001 double öğesi artık decimal olarak tanımlanmıştır (casting)
```

Q. Hata! Strateji çalıştırılırken bir hata oluştu: Nesne başvurusu bir nesnenin örneğine ayarlanamadı.

A. Bu hata başka nedenlerle de görülebilmekle birlikte, çoğunlukla yanlış veya olmayan bir sembol kullanıldığında alınmaktadır. Bu durumda çözümü basittir.

Örn.

```
[SymbolParameter("garanti")] //Yanlış. 'garanti' diye bir sembol mevcut değildir.
[SymbolParameter("GARAN")] //Doğru
```

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

iQ