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
- Algoritma ve Explorer Sihirbazı ile kod yazmadan kullanım olanağı
- 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 kullanılarak, overall, rapor, strateji seçenekleri ve emir listeleri ekranlarının istendiği gibi tasarlanabilmesi 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 ve çoklu periyot ile optimizasyon imkanı
- Optimizasyon sonuçlarının Excel'e aktarılabilmesi

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

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

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
- Excel'e aktarım

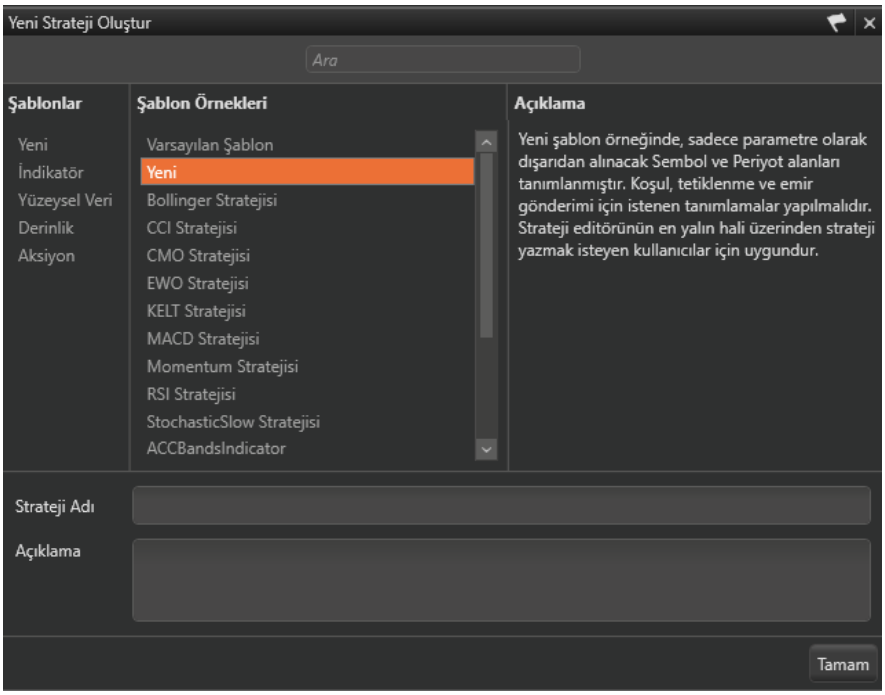
2. Strateji Yapısı

A. Algo Trader Menüsü

Yeni Strateji Oluştur

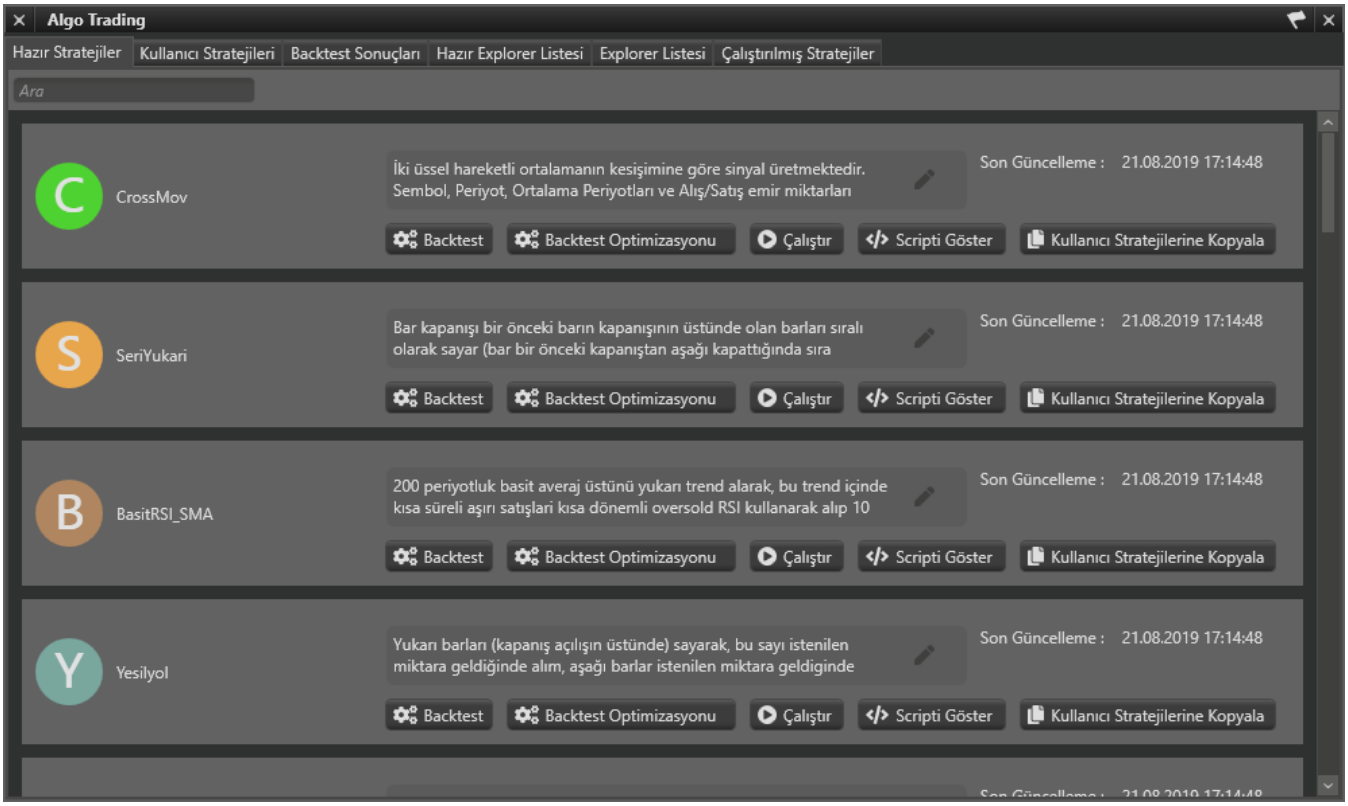
Yeni strateji oluşturmak için kullanılır. Kullanıcılar kendi stratejilerini oluşturabilir veya hazır örnek şablonlardan kullanabilir.

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.

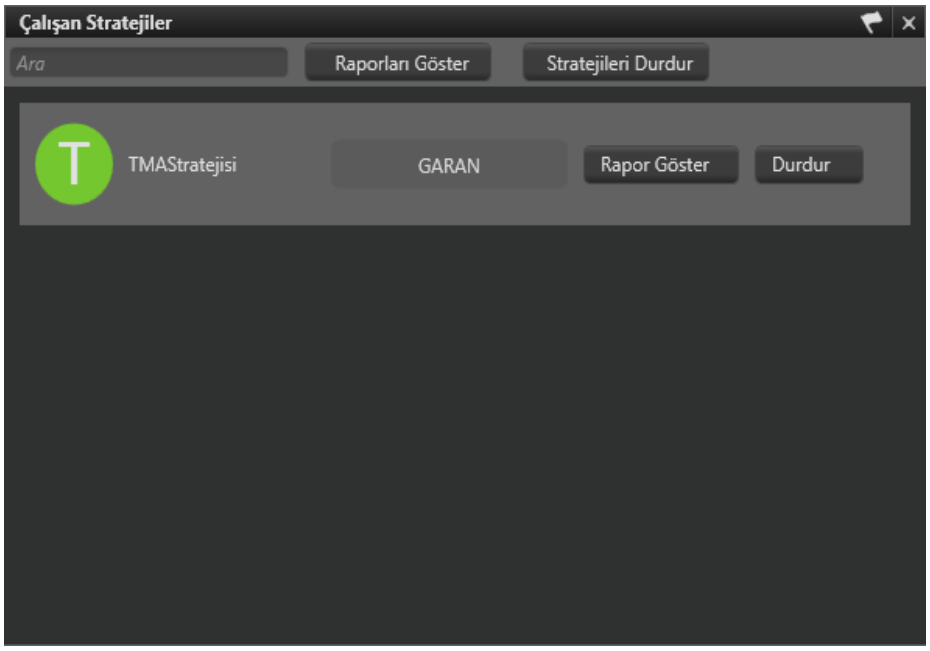


Kullanıcı Stratejileri

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.

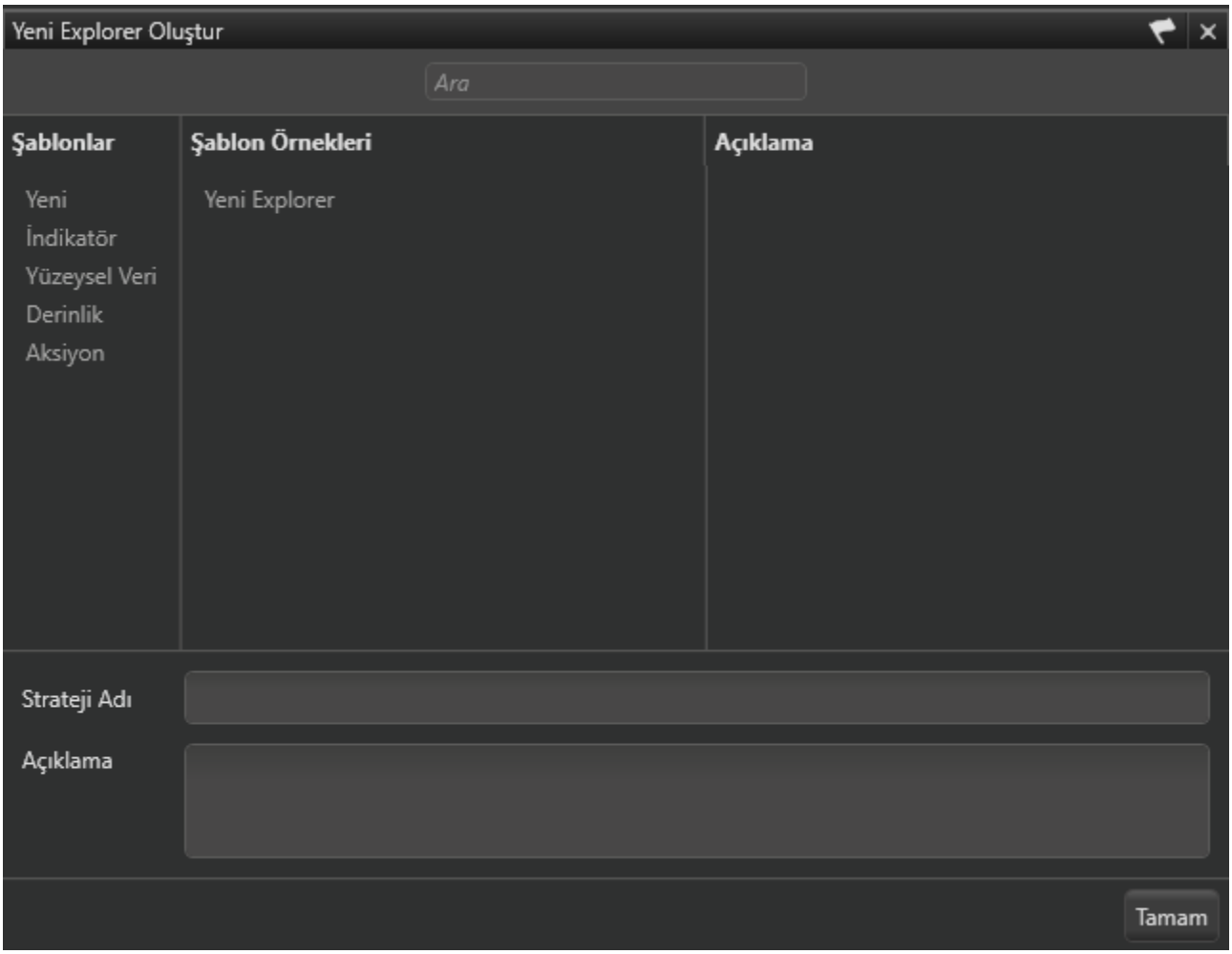


Backtest Sonuçları

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

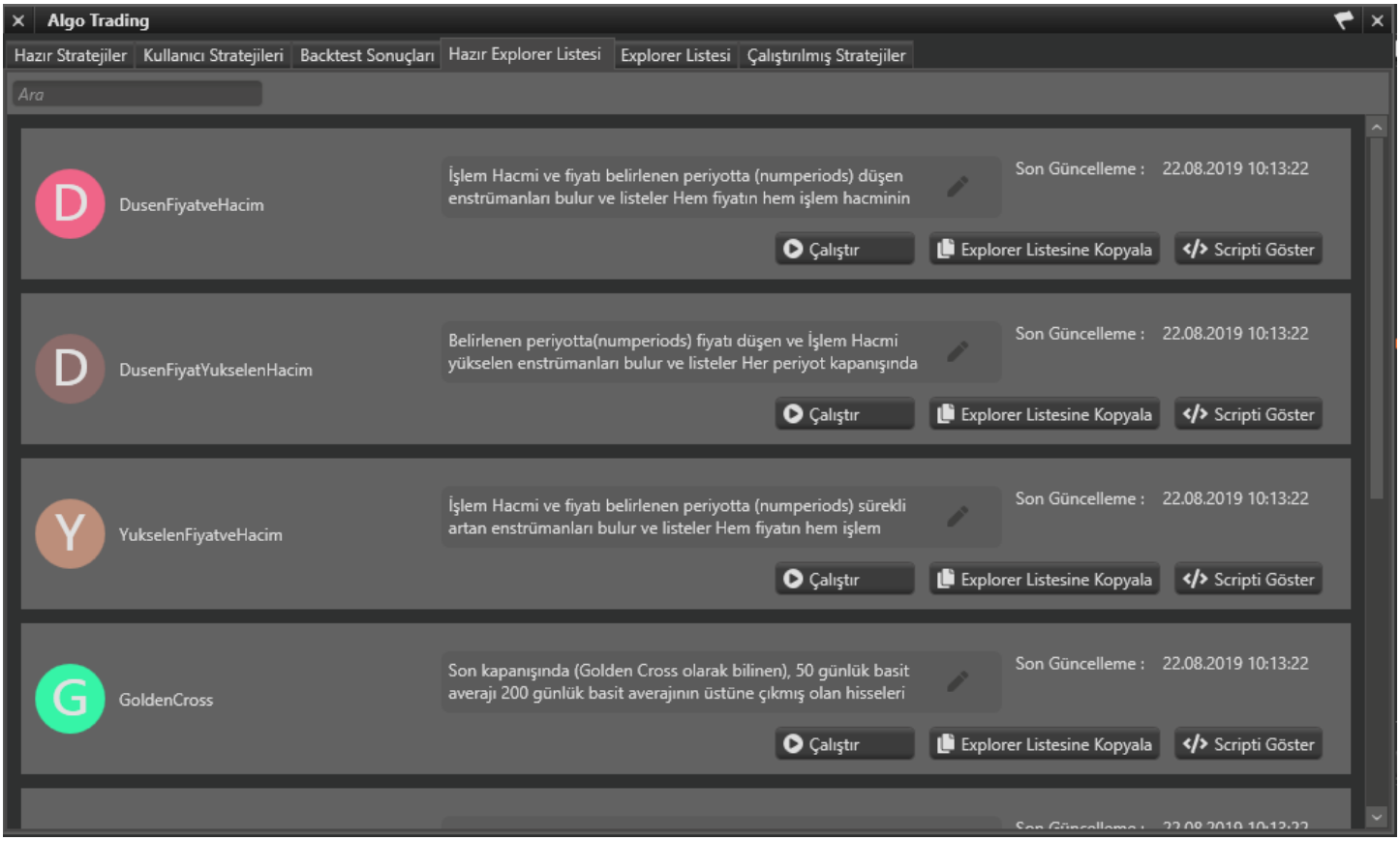
Yeni Explorer Oluştur

Kullanıcıların kendi explorerlarını oluşturmasına olanak sağlar. Burada 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

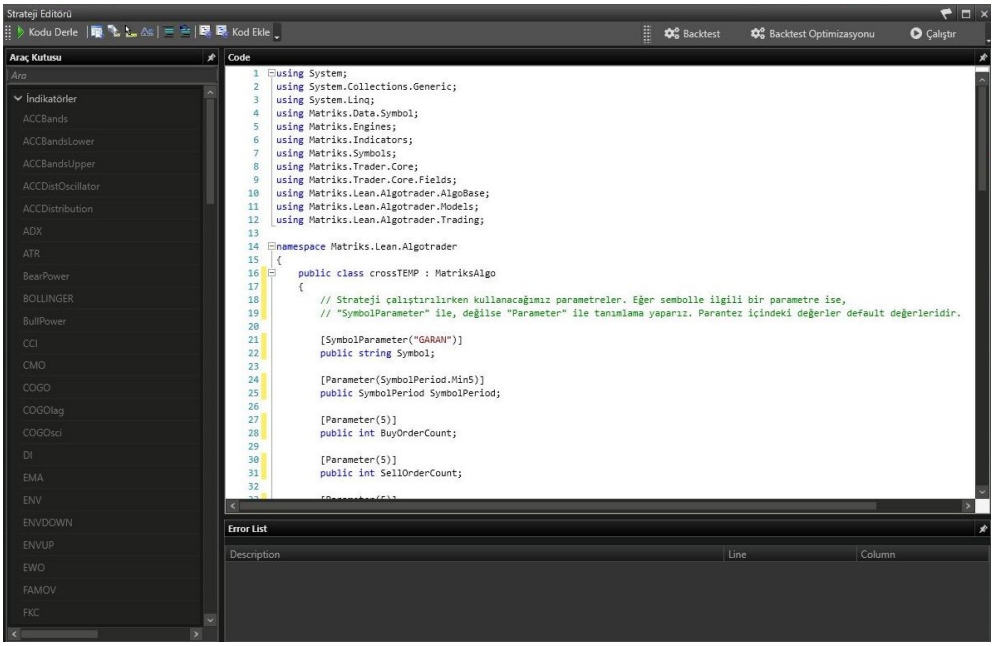
Menüden AlgoTrader->Yeni Strateji Oluştur->Yeni->Yeni seçilip, Strateji Adı yazıldıktan sonra Tamam butonuna tıklanarak boş bir şablon açılabilir.

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 çalıştırılmasında bu parametreler kullanılmaktadır. Soldaki indikatörler menüsünden eklenen indikatörlerin tanımları da bu bölümde yapılmaktadır.

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

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.

Yukaridaki mevcut SendOrderSequential fonksiyonunun geliştirilerek, sıralı emir gönderimi aktif kalacak şekilde, satış ya da alış emriyle başlama koşulu ekleyebilme vb. aşağıda detaylandırılmış özellikler eklenmiştir.

SendOrderSequential(false) : Emirler sırasız gönderilecektir.

SendOrderSequential(true) : Emirler ilk alış emri ile başlanarak (alış emri oluşmazsa, oluşması beklenecektir) sıralı gönderilecektir. {al,sat,al,sat…}

SendOrderSequential(true, true) : Emirler sıralı gönderilecektir, fakat ilk emrin ne olduğu dikkate alınmayacaktır. Örnek sıralamalar {sat, al, sat, al…} ya da {al,sat,al,sat…}

SendOrderSequential(true, false) : SendOrderSequential(true) ile aynı işlevi görür. {al,sat,al,sat…}

SendOrderSequential(true, Side.Buy) : Emirler sıralı gönderilecektir, SendOrderSequential(true) ile aynı işlevi görür. {al,sat,al,sat…}

SendOrderSequential(true, Side.Sell) : Emirler sıralı gönderilecektir, fakat ilk emrin satış emri olması beklenecektir. {sat,al,sat,al…}

Dikkat edilmesi gereken bir husus, mevcut versiyonda yukarıda sıralanan yöntemler dışında bir yöntem kullanılamamaktadır. Örn. SendOrderSequential(false, Side.Buy) yazmanız emirlerin sırasız olması dışında bir kullanım sağlamayacaktır.

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.

```
Code
71          /// SetTimerInterval fonksiyonu ile belirtilen sürede bir bu fonksiyon tetiklenir.
72          /// </summary>
73          public override void OnTimer()
74          {
75
76          }
77
78          /// <summary>
79          /// AddNewsSymbol ve AddNewsKeyword ile haberlere kayit olunmuşsa bu fonksiyon tetiklenir.
80          /// </summary>
81          /// <param name="newsId">Gelen haberin id'si</param>
82          /// <param name="relatedSymbols">Gelen haberin ilişkili sembolleri</param>
83          public override void OnNewsReceived(int newsId, List<string> relatedSymbols)
84          {
85
86          }
87
88          /// <summary>
89          /// Eklenen sembollerin bardata'ları ve indikatorler güncellendikçe bu fonksiyon tetiklenir.
90          /// </summary>
91          /// <param name="barData">Bardata ve hesaplanan gerçekleşen işleme ait detaylar</param>
92          public override void OnDataUpdate(BarDataEventArgs barData)
93          {
```

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)

Bu bölümde stratejinin asıl mantıksal kısmı yer almaktadır. Her yeni bar açılışında tick update geldikten sonra tetiklenmektedir. Burada önemli olan husus, OnDataUpdate fonksiyonunun tetiklenmesi için yeni bara ait data'nın (update'in) gelmiş olmasının gerektiğidir. Sembole ait yeni işlem gerçekleşmediği sürece, bar'ın süresi dolmuş olsa bile fonksiyon tetiklenmeyecektir. Yeni bara ait ilk veri henüz gelmediği için bar oluşturulmaya başlanamamıştır.

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

OnDataUpdate(BarDataCurrentValues barDataCurrentValues)

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

BarDataUpdate her tetiklendiğinde barDataCurrentValues ismindeki obje her 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: Barın en yüksek değeri.
Low: Barın en düşük değeri.

Close: Bar kapanış fiyatı.
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.

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

OnRealPositionUpdate(AlgoTraderPosition position)

Strateji ilk calistirildiginda ve daha sonra, portföyde bir pozisyon değişikliği olduğunda tetiklenir.
AccountId : Hesabı tanımlayan rakam
BrokerageId : Kurumu tanımlayan rakam
Symbol : Portföy bilgilerini aradığımız sembol
PositionId : Pozisyon tanıtıcısı
Currency : Pozisyonun baz para birimi
QtyT : Uzlaşma durumuna (gününe) göre, belirtilen sembolün mevcut adedini gösterir
QtyT1- QtyT3 : Uzlaşma durumuna (gününe) göre, belirtilen sembolün mevcut adedini gösterir
Amount : Sembolün güncel fiyatına göre, porföydeki anlık toplam değeri (QtyAvailable * Güncel Fiyat)
QtyAvailable : Mevcut işlem yapılabilir adet
AvgCost : Seçili sembole ait ortalama işlem fiyatı
OpeningAveragePrice : Pozisyonun açılış fiyatını ifade eder
Side : Alım 1, Satım 2 olmak üzere pozisyon yön değeri
IsSymbol : Sembolün IQ'da tanımlı olup olmadığına dair Boolean değer
ExchangeID : İşlem yapılan borsanın kodu
QtyLong : Uzun pozisyon Adedi (Uzun/Kısa pozisyon alınabilen semboller için geçerlidir, yoksa 0 döner)
QtyShort : Kısa pozisyon Adedi (Uzun/Kısa pozisyon alınabilen semboller için geçerlidir, yoksa 0 döner)
QtyNet : Toplam, net pozisyon adedi
SettlementPx : Uzlaşma fiyatını ifade eder. Pay piyasası için bu alanda ilgili sembolün son fiyat (lastpx) değeri gönderilir.

Aşağıda, Symbol olarak belirlediğimiz portföyde bulunan finansal enstrümanın mevcut toplam değerinin Strateji içerisinden, Debug sekmesine bastırılması gösterilmiştir. Bu örnekte işlemin OnDataUpdate içerisinden yürütülmesi örneklendirilmiştir.

```
public override void OnDataUpdate(BarDataCurrentValues barDataCurrentValues)
{
     //PositionReceiveComplated: Portföyden pozisyonların okunmasını beklemek için
     gereken kontrol
     if (PositionReceiveComplated)
     {
           var position = GetRealPositions();
     //GetRealPositions(): Portföydeki hisselere ait detayları çekmek için kullanılır.
           decimal Tutar = 0;
           if (position.ContainsKey(Symbol)) Tutar = position[Symbol].Amount;
           Debug($"{Symbol} sembolünden {Tutar} tutarında Portföyünüzde
bulunmaktadır.");
     }
```

```
}
```

Aşağıdaki ikinci örnekte ise konu başlığı olan OnRealPositionUpdate fonksiyonunu kullanarak yukarıdaki kod ile aynı sonucu alacağımız işlem yapılmaktadır.

```
public override void OnRealPositionUpdate(AlgoTraderPosition position)
{
     if (position.Symbol.Equals(Symbol))
     {
           if (Kripto) Kripto_kontrol = true;
           var P = position;
           Debug(P.Amount);
     }
}
```

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

Aşağıda stratejiler içerisinde kullanabileceğiniz fonksiyonlar ve kısa açıklamaları bulunmaktadır. Dikkat edilmesi gereken bir husus, fonksiyonlardan sonra parantez içine yazılan öğeler, fonksiyonun alabileceği öğeler kümesinden sadece bir tanesidir. Aşağıda bulunan birçok fonksiyon daha farklı öğeler de alabilmektedir. Bunları görebilmek için fonksiyonu stratejinize yazarken parantez açtığınızda çıkacak intellisense açıklamalarına dikkat ediniz. Fonksiyonun alabileceği farklı öğe kümelerini görebilmek için klavyenizin aşağı/yukarı oklarını kullanabilirsiniz.

Fonksiyonların aldığı Öğeler

SymbolDef: Stratejide belirtilmiş, kayıt olunmuş varsayılan semboldür. Stratejilerde sembollere, OnInit bölümü içerisinde, AddSymbol veya indikatör tanımlaması yapılırken kayıt olunmakta ve sembole ait bilgiler çekilmektedir. Birden fazla sembol tanımlanması durumunda SymbolDef ilk kayıt olunan sembol alınır. Manuel olarak tanımlanmak istenirse SymbolDef = AddSymbol(Symbol, SymbolPeriod); şeklinde yapılabilir.

Örnek: Decreasing fonksiyonu, *Decreasing(SymbolDef, Int32, OHLCType, Boolean)* olarak açıklanmıştır. Bu örnekteki kullanım şekli Decreasing(10,OHLCType.Close,false); şeklinde olabilir. Tanımda SymbolDef yazılı kısım, kod yazımında boş bırakılmıştır. Kayıt olunmuş varsayılan sembol, tekrar yazılmasına gerek kalmadan, bu fonksiyon içerisinde kullanılacaktır.

ISymbolBarData: Sembol bar datası içeren öğe.

Örnek: var bardata = GetBarData(); satırı kullanıldığında, bardata bir ISymbolBarData öğesi olacaktır.

IIndicator: Seri indikatör datası içeren öğe.

Örnek: accBands = ACCBandsIndicator(Symbol1, SymbolPeriod1, OHLCType.Close, AccbandsPeriod1, AccbandsFactor1); tanımı yapıldığında accBands bir IIndicator öğesi olacaktır.

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, her bar düşen durumda olup olmadığının 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, her bar düşen durumda olup olmadığının kontrolünü yapar.

Decreasing(SymbolDef, Int32, OHLCType, Boolean): Verilen sembol için seçilen aralıkta, her bar düşen durumda olup olmadığının 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.

GetPriceStepForBistViop(string Symbol, decimal Price): Bist ve VIOP sembolleri için minimum fiyat hareketi değerini döner. Örneğin GARAN sembolü için 0.01, EGEEN için 0.1 ve F_XU0300821 vadeli sembolü için 0.25 değerini dönecektir (sabit değer değildir).

GetSessionTimes(string symbolName): Enstrümanın seans zamanlarını döner.

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

GetSymbolDetail(int symbolId): Sembol id veya sembol ismi (string) alır. Aşağıda içeriği belirtilmiş SymbolDetail öğesini döner.

Seçilen sembol için:
ExchangeDetail ExchangeDetail: ExchangeCode, ExchangeDescription, ExchangeID isimleri altında üç adet alan döner.
Örnek:

```
var detail = GetSymbolDetail("GARAN");
exchangeDetail = detail.ExchangeDetail;
Debug($"{exchangeDetail.ExchangeCode}, {exchangeDetail.ExchangeDescription},
{exchangeDetail.ExchangeID}");
(OUTPUT) 01.01.01 13:23:02.198: BISTPP, BIST Pay Piyasası, 4
```

string SymbolName: Sembol ismi
int SymbolID: Sembol id'si
decimal StrikePrice: Varant ve opsiyonlar için geçerli bölüm, kullanım fiyatı
int TradeFraction: adet bazında işlem yapılabilecek azami ondalık basamak sayısı
int DecimalCount: fiyat bazında işlem yapılabilecek azami ondalık basamak sayısı
string UnderlyingSymbol: Vadeli sembolün baz sembolünü belirtir (VIOP)
decimal MinPosition: Alınabilecek asgari pozisyon büyüklüğü (VIOP)
DateTime MaturityDate: Sembol için geçerli ise, vade tarihi

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

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

GetTradeUser(): Kullanıcının hesap bilgisi, giriş bilgisi, emirleri ile ilgili çeşitli bilgiler döner.

Accounts: Kullanıcının açık olan hesaplarını döner
AccountId: Kullanıcının bağlı olduğu hesabın Id'sini döner
AlertAutoOrder: Alarmlarda oluşturulan emirler için yetki olup/olmadığının cevabını true/false şeklinde döner
AutoOrder: Algo'dan otomatik emir verildiği zaman "true" şekinde cevap döner
BrokerageId: İşlem yapılan aracı kurumun Id'sini döner
LoginInfo: Kullanıcının giriş bilgilerini döner
MainTradeUser: Bazı aracı kurumlarda birden fazla alt hesap olabiliyor, bu fonksiyon kullanıcının ana hesabını döner.
TestAutoOrder: Otomatik emir testi

Örnek:

```
var trader = GetTradeUser();
Debug(trader.AccountId);
```

```
01.01.0101 13:23:02.198: 0~811111
```

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

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

HighestHighWithIndex(ISymbolBarData, OHLCType, Int32): Belirtilen sembol, OHLC tipi ve periyota göre geçmiş en yüksek değeri ve bar endeksini döner. Atandığında `.high` ve `.index` metodlarıyla en yüksek ve endeks değerlerine ulaşılabilir.

Örnek: *HighestHighWithIndex(bardataModel, OHLCType.High, 10)*, 10 bar içerisinde oluşmuş en yüksek değeri ve en yüksek değerin oluştuğu bar endeksini döner.

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)

LowestLowWithIndex(ISymbolBarData, OHLCType, Int32): Belirtilen sembol, OHLC tipi ve periyota göre geçmiş en düşük değeri ve bar endeksini döner. Atandığında `.low` ve `.index` metodlarıyla en düşük ve endeks değerlerine ulaşılabilir.

Örnek: *LowestLowWithIndex(bardataModel, OHLCType.Low, 25)*, 25 bar içerisinde oluşmuş en düşük değeri ve en düşük değerin oluştuğu bar endeksini döner.

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 grafiğe çizeceğimiz veridir.

RoundPriceStepBistViop(string Symbol, decimal Price): Bist ve VIOP sembolleri için minimum fiyat hareketine uygun olacak şekilde, girilen rakamı yuvarlar. Fonksiyonun kolaylık sağlayacağı örnek bir kullanım alanı, indikatörlerden gelen decimal değeri işlem yapılabilecek ondalık hane sayısına yuvarlayabilmesidir. Bu durumda, fonksiyonun hesaplayacağı rakam ile, ek kontrole gerek kalmadan, Limit emir gönderilebilecektir.

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

IQ Algo Sentetik Emir Yapısı Yenilikler

- 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 belirleniyorken yeni yapı ile fiyat değişimleri oldukça, hesaplanan iki stop noktasından güncel pozisyona uygun olan seçiliyor ve emir yönü de güncel pozisyona göre belirleniyor.

- 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.

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.

4. Indicator Builder

IQAlgo menüsünden “Yeni İndikatör Tanımla” menü adımıyla kendi indikatörünüzü oluşturabilme imkanı eklendi. Bu pencerede sizlere örnek olması açısından farklı indikatör şablonları eklenmiştir. Bu örnekler üzerinden ilerleyerek daha kolay şekilde indikatör tanımlamaları yapabilirsiniz.

Oluşturduğunuz indikatörün kodunu derlediğinizde, editör üzerinde önizleme penceresi açılacaktır. Bu pencereden indikatörünüzü Günlük periyotta izleyerek yazım işlemini tamamlamadan önce kontrol edebilirsiniz. Editör üzerindeki “Grafikte Göster” butonu ile indikatörünüzü grafik üzerine ekleyebilirsiniz.

IQAlgo menüsünden Kullanıcı İndikatörleri menü adımıyla yazdığınız indikatörlerin listesine ulaşabilirsiniz. Buradan düzenleme, grafiğe ekleme ve indikatör paylaşma ve alma işlemlerini yapabilirsiniz.

Oluşturduğunuz indikatörler, grafik indikatör listesinde Kullanıcı İndikatörleri başlığı altında listelenir. Buradan oluşturduğunuz indikatörleri grafiklerinize ekleyebilirsiniz. Grafik sağ klik menüsü üzerinden Kullanıcı İndikatörleri menü adımıyla da ulaşabilirsiniz. Hazır indikatörlerde olduğu gibi, kendi oluşturduğunuz indikatörleri de formül penceresinde ve indikatör alarmlarında kullanabilirsiniz. Oluşturacağınız indikatörleri stratejilerinizde kullanabilirsiniz. Strateji editöründe indikatörler listesinden kendi indikatörlerinize ulaşabilirsiniz.

Indicator Builder ile ilgili kurallar ve bilgiler:

Ilk parametre indikatörün adı, sınıfın adıyla aynı olmalıdır. Ikinci parametre indikatörün Data serisinin üzerine mi yoksa yeni pencereye mi ekleneceğini belirtir. Yeni pencere için ->IndicatorDrawingArea.NewWindow, Data Serisi için IndicatorDrawingArea.OnDataSeries şeklinde tanımlanır.

Örn. [IndicatorInformationAttribute("Indikator", IndicatorDrawingArea.NewWindow)]

Indikatörün çizgilerinin isimleri

```
[IndicatorLineInformationAttribute(new []
{
     "Indikator"
})]
```

Indikatör opsiyon panelinde değerleri değiştirebildiğimiz parametreler. Int, Bool, Decimal ve Enum değerleri alabilir.Tüm değişken tiplerini DefaultValue ile tanımlarız.

```
[DefaultValue(14)]
public int Period
```

```
{ get; set; }
```

public sealed override void OnInit(): Indicator değerleri hesaplanmadan önce oluşturulacak indikatörler burada tanımlanıyor.

public override void OnDataUpdate(int currentBar, decimal inputValue, DateTime barDateTime): Seçilen sembolün bardata'ları güncellendikçe bu fonksiyon tetiklenir.

int currentBar: Güncellenen bardata'nın endeksteki sırası

decimal inputValue: Seçilen OHLC tipine göre gelen bardata'nın o anki değeri

DateTime barDateTime: Bardata'ya gelen güncelleme zamanı

Oluşturulan indikatörün o anki değerini `Indikatör.CurrentValue` şeklinde alabiliriz.

```
var val = MovingAverage.CurrentValue;
```

SetLine fonksiyonu indikatördeki noktaları kuran fonksiyondur

```
SetLine(0, currentBar, val);
```

Indicator Builder Yeni Eklenen Fonksiyon ve Özellikler

SetPointTitle(int lineIndex, int barIndex, string title, IconLocation chartLocation, decimal value, bool isDrawQuad, string textColor): indikatörün istenilen çizgisinin istenilen endeksine verilen yazıyı yazan fonksiyondur.

SetPointTitle(int lineIndex, int barIndex, IndicatorIconStyle iconStyle, IconLocation chartLocation, decimal value, bool isDrawQuad, string textColor): fonksiyonu sistemde tanımlı iconları(al, sat, stop gibi) indicator üzerinde gösterilmesine yarar.

Parametreler:

lineIndex: Çizilen indikatörun hangi çizgisine ekleneceğini belirler. 0'dan başlar.
barIndex: Indikatörün hangi noktasında yazının çıkacağını belirtir.
title: Ne yazılacağını belirtir.
chartLocation: yazının barın altında mı üstünde mi yoksa yukarısında mı çıkacağını belirtir.
value: Y axisinde verilen değerde yazı çıkar.
isDrawQuad: yazının çerçeve içine alınıp alınmayacağını belirtir.
textColor: yazının rengini belirtir.

iconStyle: Hangi ikonun konulacağını belirtir.
chartLocation: yazının barın altında mı üstünde mi yoksa yukarısında mı çıkacağını belirtir.

5. Örnek Stratejiler

Basit RSI_SMA Stratejisi

```
namespace Matriks.Lean.Algotrader
{
    public class basitRSISMA : MatriksAlgo
    {
       //strateji ismini burada deklare ediyoruz. Strateji oluştururken verilen isimle
      //stratejide (public class deklerasyonunda - yukarıda - yazılan isim tamamen aynı
      //olmalıdır, aksi takdirde ad alanı bulunamadı hatası düşecektir.(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. Dosyada
ki 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; //Tma periyodu için kullanacağımız parametre
    [Parameter(100)]
    public decimal BuyOrderQuantity; //alım miktarı için kullanacağımız parametre
    [Parameter(100)]
    public decimal SellOrderQuantity; //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() {

      //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) {
      var barDataModel = GetBarData();
      //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(barDataModel, hullMA, OHLCType.Close, 0) && hullMA.CurrentValue <
tma.CurrentValue) {
        //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa fiyatından
       alış emri gönderir
        SendMarketOrder(Symbol, BuyOrderQuantity, 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(barDataModel, hullMA, OHLCType.Close, 0) && hullMA.CurrentValue >
tma.CurrentValue) {
```

```
            //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa fiyatından
        satış emri gönderir
            SendMarketOrder(Symbol, SellOrderQuantity, 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)
```

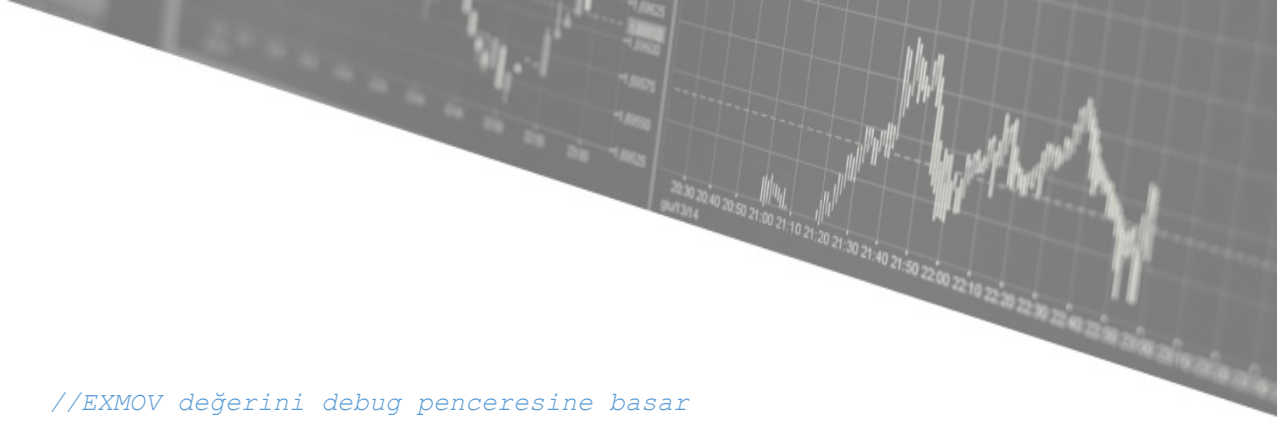
```
{
    //Bu koşul alım emri içindir. Eğer grafikte MOST'un EXMOV bandı
    //most bandını yukarı kırarsa al emri gönderilecek.
    if (CrossAbove(most.CurrentValue, most.ExMOV))
    {
        //Parametrelerde belirlenen sembolden, belirlenen miktarda, piyasa
        //fiyatından alış emri gönderir
        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");
            }
        }
    }
}
```

6. Güncellemeler ve Yeni Eklenen Özellikler

Versiyon 3.5.0.1

Çıktı Parametreleri (Output):

Canlı Algo çalıştırdığımızda açılan Rapor penceresinde, Loglar, Debug, Kod seklinde giden aşağıdaki tab'lere Çıktı Parametreleri tab'i özelliği eklenmiştir. Strateji kodu içerisinde 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;
}
```

Sentetik Emir Yapısı (bknz. Algo Sentetik Emir Yapısı Yenilikler)

Versiyon 4.0.0

Indicator Builder (bknz. Indicator Builder)

Indicator Builder Yeni Eklenen Fonksiyon ve Özellikler

Versiyon 4.0.6

Bknz. GetTradeUser

Bknz. OnRealPositionUpdate

Bknz. SendOrderSequential

Versiyon 4.0.7

Cross fonksiyonu içerisinde Indikatorlerin Endeksleriyle Kullanılabilmesi

CrossAbove ve Crossbelow fonksiyonlarında, kullanılan indikatörler için endeksi ile çağırma özelliği eklenmiştir. Aslen Algoritma Sihirbazının kullanıcı girdilerinin daha etkin bir şekilde koda yansıtabilmesi için yazılmıştır. Aşağıda örneklerle detaylandırılmıştır:

CrossAbove(IIndicator indicator, int value, [int indicatorLineIndex])

Örnek: *CrossAbove(rsi,50,0):* RSI indikatorunun, 0. endeksteki çizgisi, 50 değerinin üzerine çıktığında true döner. RSI indikatörünün tek çizgisi olduğu için ve indikatör endeksleri sıfırdan başladığı için 0 indikatör çizgi endeksi kullanılmaktadır.

CrossAbove(IIndicator indicator, IIndicator indicator2, [int indicator1LineIndex], [int indicator2LineIndex])

Örnek: *CrossAbove(most, most, 1, 0):* Most indikatörünün 1. endeksli çizgisi 0. endeksli çizgisinin üzerine kırdığında true döner. Most indikatörünün sıfır endeksindeki çizgisi MOST, birinci endeksteki çizgisi ise ExMov'dur.

Hangi indikatör çizgisinin hangi endekste olduğunun bir listesi bulunmamakla birlikte, aşağıdaki kod stratejinizde OnDataUpdate fonksiyonu içerisine eklediğiniz taktirde, strateji raporu penceresindeki debug sekmesine basılan değerler ile, grafikteki canlı değerleri kontrol ederek doğru endeksi kullanıp kullanmadığınızı anlayabilirsiniz.

```
var currentBar = barDataCurrentValues.LastUpdate.BarDataIndex;
Debug(bollinger.Value[0][currentBar]);
Debug(bollinger.Value[1][currentBar]);
Debug(bollinger.Value[2][currentBar]);
```


TRX_USDT_BIN
1 Dk.
TL
Lineer
Candle
İndikatörler
TMP
Al
Sat
Seçimler
15.03.2021 18:21
Açılış 0.05029
Yüksek 0.05034
Düşük 0.05026
Kapanış 0.05034
Ağ. Ort 0.05030
Fark 0.00004
Fark% 0.07952
Hacim 78,103
MOST (3,% 2)0.05
ExMOV (3)0.05
BOL U (20,2)0.05041188
BOL M (20,2)0.05032350
BOL D (20,2)0.05023512
18:21:21.832 :0.05041187
18:21:21.833 :0.0503235
18:21:21.834 :0.05023512
0.05045
0.05041
0.05040
0.05035
0.05034
0.05032
0.05030
0.05025
0.05024
0.05020
0.05015
0.05034 - 1499 - 18:21:20
18:06
18:08
18:10
18:12
18:14
18:16
18:18
18:20

Ayrıca, kural olmamakla birlikte, çoğu durumda, grafiğe eklediğiniz indikatörün grafikte sol üstte gözüken güncellenen değerleri endeks sırasına göre yazılmaktadır. Örn. Yukarıdaki ekran görüntüsünde ilk yazılmış bollinger değeri aynı zamanda 0. endekstedir. 2. yazılan 1. endekste, 3. rakam ise 2. indikatör endeksindedir.

Algoritma ve Explorer Sihirbazı

Basit algoritmaların, istenilen sembol, indikatör ve parametreler, tıkla ve seç kullanılarak, koşul ve emirler ile beraber eklenerek oluşturulması için kullanılır. Sihirbaz kullanıldığında, kodlamaya gerek kalmadan algoritma oluşturturulabildiğinden, kullanışlı ve pratiktir. Sihirbaz kullanıldıktan sonra oluşturulan kod incelenebildiğinden, aynı zamanda kodlamaya başlangıç basamağı olarak da faydalanılabilir.

Algoritma sihirbazının kullanımı ile ilgili detaylı bilgi içeren dosyalar, dokumanlar arasına eklenmiştir. Ayrıca destek sitesi üzerinden ulaşabileceğiniz, ayrıntılı açıklamaların bulunduğu sayfaya Algoritma Sihirbazı linkinden, Explorer sihirbazı için hazırlanmış video'ya Explorer Sihirbazı linkinden ulaşabilirsiniz.

Versiyon 4.0.8

MyTrend Fonksiyonu ile ilgili Geliştirmeler

1. Algo çalışmaya başladıktan sonra trend oluşturulabilmesi.
Önceki versiyonlarda trendler, indikatörler gibi sadece Init fonksiyonu içerisinde tanımlanabiliyorlardı. Geliştirmeler sonrası, OnDataUpdate gibi algo başlatıldıktan sonra çalışan yerlerde de MyTrend fonksiyonu kullanılabiliyor ve yeni trendler oluşturulabiliyor.

Örnek: *SetColumnText(1, "Mov1"):*

2. Trend'den snapshot oluşturulması:
Önceden oluşturmuş olduğumuz bir trendin, oluşturulduğu zamandaki halinden bir kopya oluşturmak için kullanılır. Autotrend'lerde yeni veri geldiğinde, trend parametreleri değişiklik gösterebildiğinden, güncellenme öncesi durumu muhafaza edebilme ve sonradan kullanabilme olanağı snapshot ile sağlanmıştır.

Örnek: *var snapshot = myTrend.GetSnapshot():* Önceden oluşturulmuş, myTrend isimli bir trend üzerinden bu şekilde çağrılarak kullanılır.

Oluşan snapshot objesi, MyTrend ile oluşturulan trendler ile aynı tipte bir objedir. Dolayısıyla trendler üzerinde kullanılabilen fonksiyonlar snapshot için de çalışır.

Snapshot'ın X1, X2, Y1, Y2 gibi tanım parametreleri sabit kalacaktır. CurrentIndex ve CurrentValue parametreleri ise yeni barlar geldikçe güncellenir.

3. Indikatör üzerinden trend tanımlanabilmesi
Indikatör üzerinden trend tanımlayabilmek için aşağıdaki fonksiyonlar eklendi:

MyTrend(IIndicator indicator, int barCount, int refIndex, TrendType trendType, int lineIndex = 0, bool isAutoTrend = false): İndikatör çizgisinin verisi ile yükselen/düşen trend oluşturmak için kullanılır.

indicator: Trend için kullanılacak indikator
barCount: Bar sayısı
refIndex: Bitiş noktası referans indeksi
trendType: Yükselen / Düşen trend seçeneği
lineIndex: Indikator cizgisinin indeksi. Opsiyonel (default 0)
isAutoTrend: Trendi data ile birlikte güncelle. Opsiyonal (default false)

Örnek: *var trend = MyTrend(sma, 100, 1, TrendType.Increasing, 0, false):* Algoda tanımlı sma datasını kullanarak, bir bar önce biten ve 100 bar uzunluğunda bir yükselen trend oluşturmak için bu ifade kullanılabilir.

MyTrend(IIndicator indicator, DateTime startTime, DateTime endTime, TrendType trendType, int lineIndex = 0): İndikatör çizgisinin verisi ile tarih vererek yükselen/düşen trend oluşturmak için kullanılır. Tarihler, indikatörün oluşturuluğu sembol ve periyodun bar data tarihlerine denk gelmelidir.

İndicator: Trend için kullanılacak indikator
startTime: Trend başlangıç zamanı
endTime: Trend bitiş zaman
trendType: Yükselen / Düşen trend seçeneği
lineIndex: Indikator cizgisinin indeksi. Opsiyonel (default 0)

Örnek: *var trend = MyTrend(sma, new DateTime(2021, 1, 3), new DateTime(2021, 3, 3), TrendType.Increasing, 0):* ifade ile verilen iki tarih arasında, algoda tanımlı sma kullanılarak yükselen trend oluşturulabilir.

MyTrend(IIndicator indicator, int barCount, int refIndex, decimal startValue, decimal endValue): İndikatör üzerine verilen değerler ile trend oluşturmaya yarar. İndikatör yerine sembol ve periyot verilerek de aynı trend oluşturulabilir. İki durumda da verilen data serisinin trende etkisi, trendin indekslerini belirlemektir.

İndicator: Trend için kullanılacak indikator

barCount: Bar sayısı
refIndex: Bitiş noktası referans indeksi
startValue: Başlangıç noktasındaki trend değeri

Örnek: *var trend = MyTrend(sma, 100, 0, 15.2m, 17.4m, false):* 100 bar önce 15.2 değeri ile başlayıp, güncel barda 17.4 değerinde biten bir trend çizgisi önceden tanımlanmış bir sma indikatörü ile oluşturulur. Trend çizgisi 100 bar sayımı için, SMA indikatörünün endeksini, dolayısıyla periyotunu kullanmaktadır.

MyTrend(IIndicator indicator, DateTime startTime, decimal startValue, DateTime endTime, decimal endValue): İndikatör üzerine verilen değerler ve tarih aralığı ile trend oluşturmaya yarar. indikatörün oluşturuluğu sembol ve periyodun bar data tarihlerine denk gelmelidir. İndikatör yerine sembol ve periyot verilerek de aynı trend oluşturulabilir. İki durumda da verilen data serisinin trende etkisi, trendin indekslerini belirlemektir.

İndicator: Trend için kullanılacak indikator
startTime: Trend başlangıç zamanı
startValue: Başlangıç noktasındaki trend değeri
endTime: Trend bitiş zaman
endValue: Bitiş noktasındaki trend değeri

Örnek: 3 ocak 2021'de 15.2 değeri ile başlayıp, 3 Mart 2021'de 17.4 değerinde biten bir trend çizgisi önceden tanımlanmış bir sma indikatörü ile aşağıdaki gibi oluşturulabilir.
var trend = MyTrend(sma, new DateTime(2021, 1, 3), 15.2, new DateTime(2021, 3, 3), 17.4);

4. Trendler için cross fonksiyonları:

CrossAbove(ISymbolBarData symbolBarData, OHLCType ohlcType, ITrend trend): Bar datanın trendi yukarı kırıp kırmadığını kontrol eder

symbolBarData: Trendi kırma kontrolü yapılacak bar data
ohlcType: Ohlc Tipi
trend: Kırılma kontrolünün yapılacağı trend

Örnek: Kapanışın trendi yukarı kırıp kırmadığını kontrol etmek için aşağıdaki kod kullanılmalıdır.
var crossAbove = CrossAbove(barData, OHLCType.Close, myTrend);

CrossBelow(ISymbolBarData symbolBarData, OHLCType ohlcType, ITrend trend): Bar datanın trendi aşağı kırıp kırmadığını kontrol eder

symbolBarData: Trendi kırma kontrolü yapılacak bar data
ohlcType: Ohlc Tipi
trend: Kırılma kontrolünün yapılacağı trend

Örnek: Kapanışın trendi aşağı kırıp kırmadığını kontrol etmek için aşağıdaki kod kullanılmalıdır.
var crossBelow = CrossBelow(barData, OHLCType.Close, myTrend);

CrossAbove(IIndicator indicator, ITrend trend, int indicatorLineIndex = 0): Verilen indikatör çizgisinin trendi yukarı kırıp kırmadığını belirten bir boolean döner.

İndicator: Trendi kırma kontrolü yapılacak indikatör
Trend: Kırılma kontrolünün yapılacağı trend
indicatorLineIndex: İndikatörün çizgi indeksi. Opsiyonel (default 0)

Örnek: Bollinger down çizgisinin trendi yukarı kırıp kırmadığını kontrol etmek için şu ifade kullanılmalıdır.
var crossAbove = CrossAbove(bollinger, myTrend, 2);

CrossBelow(IIndicator indicator, ITrend trend, int indicatorLineIndex = 0): Verilen indikatör çizgisinin trendi aşağı kırıp kırmadığını belirten bir boolean döner.

İndicator: Trendi kırma kontrolü yapılacak indikatör
Trend: Kırılma kontrolünün yapılacağı trend
indicatorLineIndex: İndikatörün çizgi indeksi. Opsiyonel (default 0)

Örnek: Bollinger down çizgisinin trendi aşağı kırıp kırmadığını kontrol etmek için şu ifade kullanılmalıdır.
var crossBelow = CrossBelow(bollinger, myTrend, 2);

5. Diğer:

DisposeTrend(ITrend trend): Algo işleyişi esnasında yeni trendler oluşturulmasına imkan verilmesi ve bu trendlerin de arka planda kaydedilip güncellemelerinin yapılması, kullanıcıların sonsuz sayıda trendi algoya eklemesine imkan veriyor. Memory ve performans sorunu oluşturmamak için artık kullanılmayacak trendlerin silinmesi gerekiyor. Bu nedenle kullanımı biten trendler DisposeTrend fonksiyonu ile silinmelidir.

Versiyon 4.0.9

Bknz. GetPriceStepForBistViop

Bknz. RoundPriceStepBistViop

Bknz. HighestHighWithIndex

Bknz. LowestLowWithIndex

7. Strateji Çalıştırmadan Önce Dikkate Alınması Gereken Hususlar

➔ Stratejinizde kullandığınız sembolün hangi piyasaya ait olduğuna veya hangi hesapla çalıştırmak istediğinize dikkat ediniz. Stratejinin çalışacağı hesabı doğru seçmelisiniz. Aksi takdirde işlemlerinizin reddedilmesi durumu veya istemediğiniz hesaplarda emir gerçekleşmeleri ile karşılaşabilirsiniz.

➔ Emir göndermek istediğiniz sembolün ait olduğu piyasanın kurallarına dikkat ederek emir bilgilerini tanımlayınız. VİOP ve Hisse tarafı için farklı kurallar söz konusudur. Binance piyasasından bir sembolle strateji çalıştırılmadan önce, Binanace kurallarını inceleyiniz.

➔ Programı herhangi bir sebeple kapatıp tekrar açtığınızda, kapatılmadan önce çalışmakta olan stratejileriniz siz tekrar çalıştırmadıkça aktif olmayacaktır.

➔ Durdurulmuş / kapatılmış stratejilerin, çalıştırılmış stratejiler bölümünden tekrar devam etmesini sağlayabilirsiniz. Bununla beraber, stratejinizi devam et ile başlatsanız dahi, strateji içerisinde tutulan veriler kaybolacağı için, işlemlerin kaldığı yerden devam etmesinin garantisi yoktur. Stratejinizin yapısına göre bu durum değişecektir. Bu sebeple, stratejinizi devam ettirdiğiniz takdirde, nasıl çalışacağını ayrıca değerlendirmeli ve ona göre hareket etmelisiniz. Örnek: Eğer stratejinizde emir gönderimi sıralı ise son emir yönü tutulduğu için, strateji devam ettirildiğinde, sonraki ilk emrini durdurmadan önceki sıraya göre iletecektir.

➔ **Stratejinizin, istediğiniz gibi çalışıp çalışmadığını bir müddet gözlemleyiniz. Doğrudan ciddi miktarlarla kesinlikle çalıştırmayınız. Öncelikle test hesaplarında veya düşük miktarlarla gerçek hesaplarınızda çalıştırıp izlemenizi tavsiye ederiz.**

➔ Stratejinizin çalışması esnasında, internet bağlantısının kesilmesi durumunda olumsuz sonuçlar (sinyal kaçırma vb.) oluşabilir. Bu nedenle strateji çalıştırmadan önce internet bağlantınızı kontrol ediniz ve sağlıklı bir internet erişiminiz olduğundan emin olunuz.

➔ Strateji çalıştıracağınız bilgisayarın tarih, saat ve bölgesel ayarlarının Türkiye'ye göre ayarlanmış olduğundan emin olunuz.

➔ Program içinde, emir iletim ve gerçekleşme gibi bilgiler kurum tarafından gelir. Ve soket bağlantısı yoktur. Yani manuel veya otomatik güncelleme gereklidir. MatriksIQ'da çalıştırılan stratejilerde, gönderilen emir verileri veya portföyden çekilmek istenen veriler, portföyün son güncellenmesindeki

verilerden alınır. Portföyünüz güncellenmediği sürece, gerçekleşmeler görülmeyecektir. **Bu nedenle strateji çalıştırmadan önce portföy ayarlarından Ekranları Otomatik Güncelle seçeneğini işaretleyip 1dk. seçmenizi önermekteyiz. Seçilebilir minimum süre 1 dakikadır. Stratejileriniz emir gerçekleşmelerini anlık olarak göremeyecektir. Bu tip stratejiler kullanıyorsanız, lütfen bu durumu göz önünde bulundurunuz.**

➔ Programın kullanımı ve yukarıda bahsettiğimiz detaylarla ilgili muhtelif dokümanlar vardır. Emin olmadığınız konuları mutlaka eğitim birimimize sorunuz.

8. 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_0.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/hata alıyorum.**

A. İndikatorler liste/array gibi yapılandırılmıştır. Dolayisiyla *accBands.Lower* ifadesi, içerisinde bir çok değer taşıdığından, büyüktür/küçüktür operatörüyle kullanabileceğimiz bir ifade değildir. Eğer close objesine atadığımız değerin bir indikatör değerinden 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 altındaki 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 oluşturup, bu endeksin hareketli ortalamasını alabilir miyim?**

A. Evet! MatriksIQ Algo ile kendi oluşturduğunuz değişkenin hatta kendi indikatörünüzün bile hareketli ortalamasını oluşturabilir, bütün diğer indikatörlerin içerisinde de kullanabilirsiniz. Daha fazla bilgi için aşağıdaki örnek koda ve yorumlara 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)
                    {

                    }
            }
    }
}
```

9. Sık Rastlanan Hatalar

Q. **error CS0246: 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)
```

MATRİKS
BİLGİ DAĞITIM HİZMETLERİ

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