ALGOLAB KULLANICI EL KİTABI

İÇİNDEKİLER

Algolab Nedir ?

Algolab DenizBank ve Deniz Yatırım müşterilerinin algoritmik işlemlerde kullanması için geliştirilen platformdur.

Genel Kullanım

Algolab C# dilinde algoritmalar oluşturmak ve platform için geliştirilen Algo Engine metotlarıyla kullanıcılar için birçok özel fonksiyonu kullanma imkanı sunmaktadır. Piyasa verilerinin güncel kalması, işlemlerin kesintisiz ve güvenle çalışması için tamamen bulut servislerinde çalışmaktadır.

Algolab içerisinde bulunan hazır stratejilerden dilediğinizi kullanabilir, kodlama bilmiyorsanız sürükle bırak metotları ile stratejinizi oluşturup düzenleyebilir, C# fonksiyonlarına hakimseniz dilediğiniz algoritmayı kodlama editörü ile hazırlayabilirsiniz.

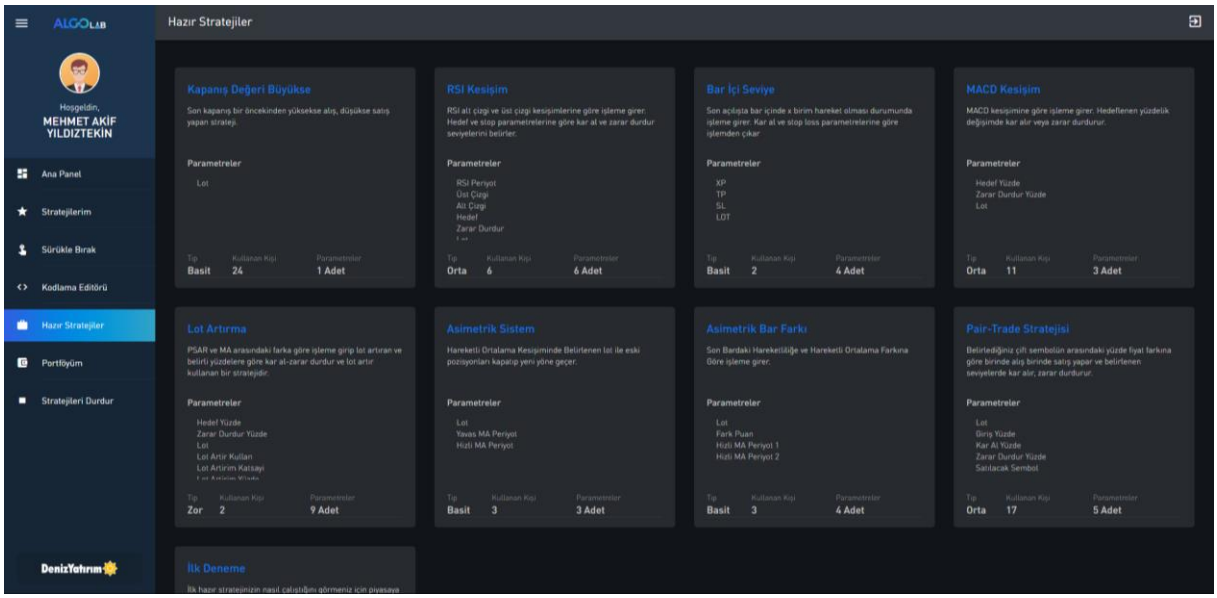
Geçmiş dönem verileri ile birden fazla periyotta stratejilerinin geçmiş performansını görebilir, piyasa şartlarına göre oluşabilecek slipaj ve komisyon maliyetlerini dahil ederek gerçeğe en yakın test sonucuna erişebilirsiniz.

Strateji Oluşturma

Kendinize uygun stratejiyi oluşturmak için 3 farklı yöntemden faydalanabilirsiniz:

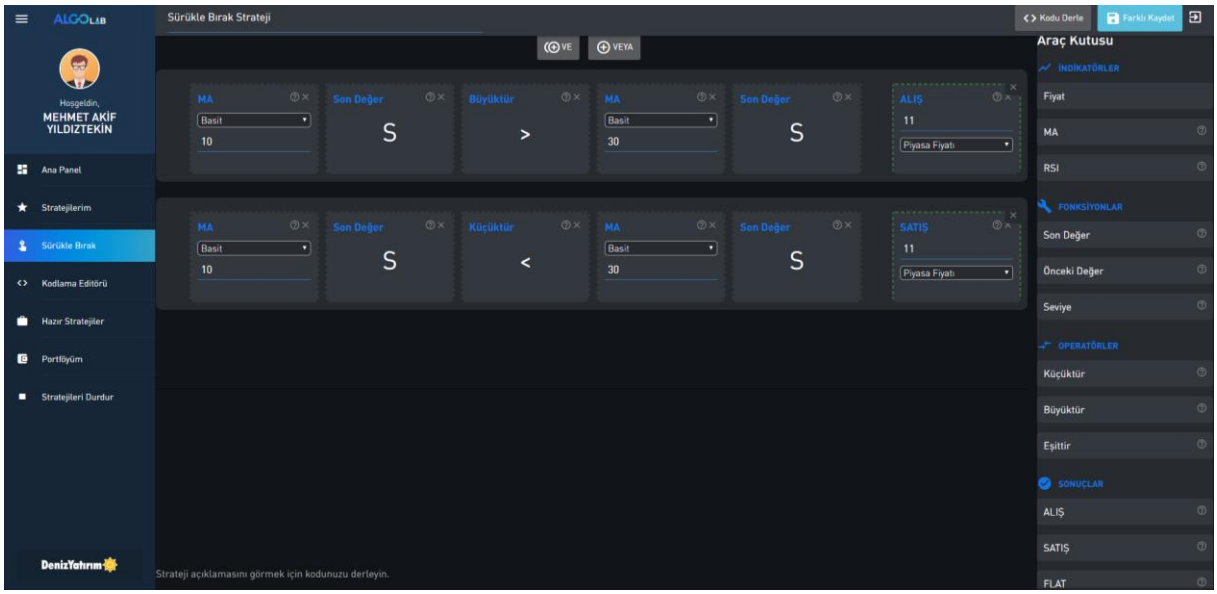
- Hazır Stratejiler
- Sürükle-Bırak
- Kodlama Editörü

Hazır Stratejiler :



Algolab' e giriş yaptıktan sonra menü kısmında Hazır Stratejiler bölümüne giriş yaparak kullanmak istediğiniz stratejiyi seçip ilgili parametreleri girdikten sonra stratejinizi kaydedip kullanmaya başlayabilirsiniz.

Sürükle-Bırak:



Kodlama bilginiz yeterli gelmiyor ya da hiç bilmiyorsanız Algolab kullanıcı menüsünden sürükle bırak ekranına giriş yapıp ilgili kutucuklara araç kutusundan karşılık gelen fonksiyonları sürükleyerek stratejinizi oluşturabilirsiniz.

Strateji Çalıştırma

Algolab' de algoritmalarınızı Stratejilerim ekranında Pasif ve Aktif olmak üzere iki menüde görebilmektesiniz. Pasif olan bir stratejiyi çalıştırmak ya da çalışan bir

stratejiyi yeni bir varlıkta çalıştırmak için herhangi bir stratejiyi seçip çalıştır butonuna basarak ilgili parametreleri girip başlatmanız yeterli olacaktır.

Strateji Durdurma

Herhangi bir stratejiyi seçtikten sonra açılan detay sayfasından durdur butonu ile ya da ayrıntılı bilgi ekranında yer alan durdur seçeneklerinden çalışan varlıklar da tek tek durdurabilirsiniz.

Ayrıca herhangi bir acil durumda ya da dilediğinizde ana ekran menüsünde yer alan stratejileri durdur seçeneği ile tüm sistemi durdurup algoritmanın açtığı pozisyonları da aktif fiyattan kapatmasını sağlayabilirsiniz.

Başvurular

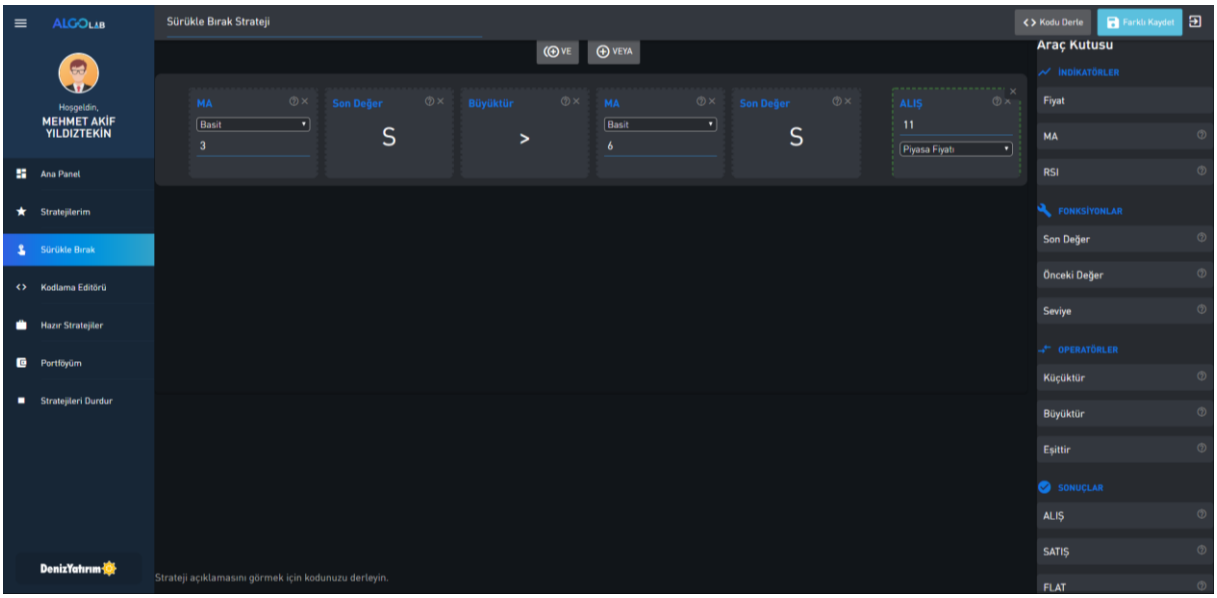
Algolab' de ekranın sağ üst menüsünde bulunan başvurular ekranından anlık veri izlemek için canlı data talebinde bulunabilir, algoritma çalıştırmak için yetki verilmesi için başvuru yapabilirsiniz.

Sürükle Bırak

Sürükle bırak modülü ile stratejilerinizi basit bir şekilde oluşturabilirsiniz. Sürükle bırak sayfasında sağ tarafta araç kutusunda indikatörler ve fonksiyonlar bulunmaktadır. Bu indikatör ve fonksiyonları sol tarafta bulunan kutucuklara sürükleyerek yerleştirebilirsiniz.

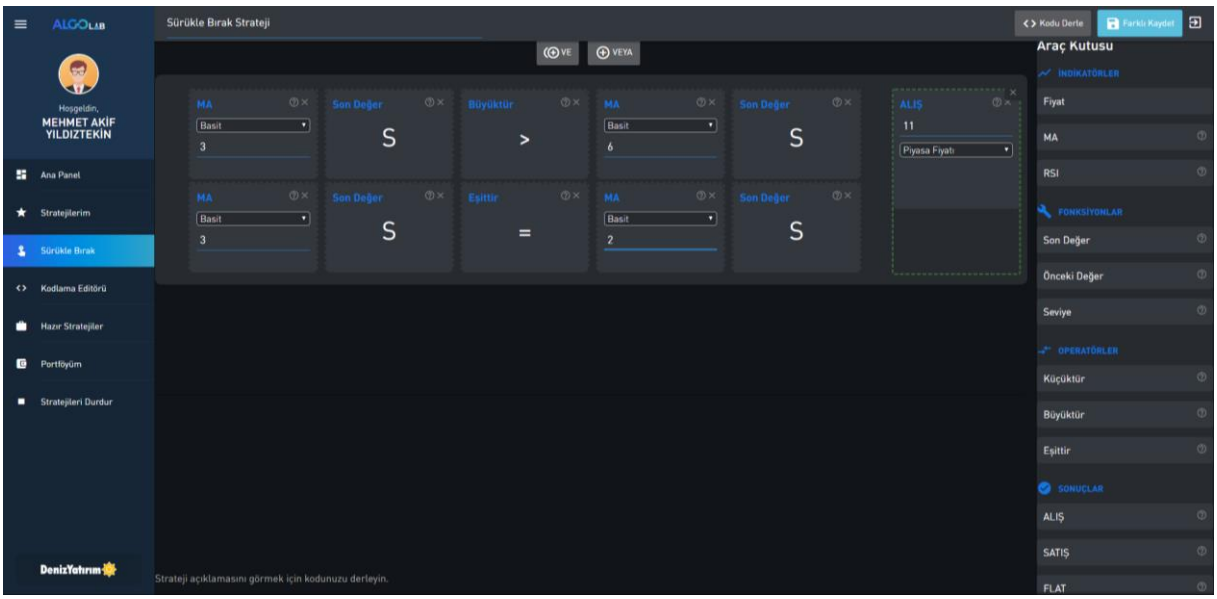
Sağ tarafta bulunan araç kutusunda indikatörler indikatör kısmına, fonksiyonlar fonksiyon kısmına, operatörler operatör kısmına ve sonuçlar sonuç kısmına fare ile sürükleyerek yerleştirilir. Kutular içerisinde bulunan parametreler girildikten sonra kodu derle butonuna basılır.

Örnek : Aşağıdaki örnekte basit 6 periyotluk hareketli ortalamanın(MA) son değeri, basit 3 periyotluk hareketli ortalamanın(MA) en düşük değerinden küçükse 7 lot satış emri aç anlamına gelir.



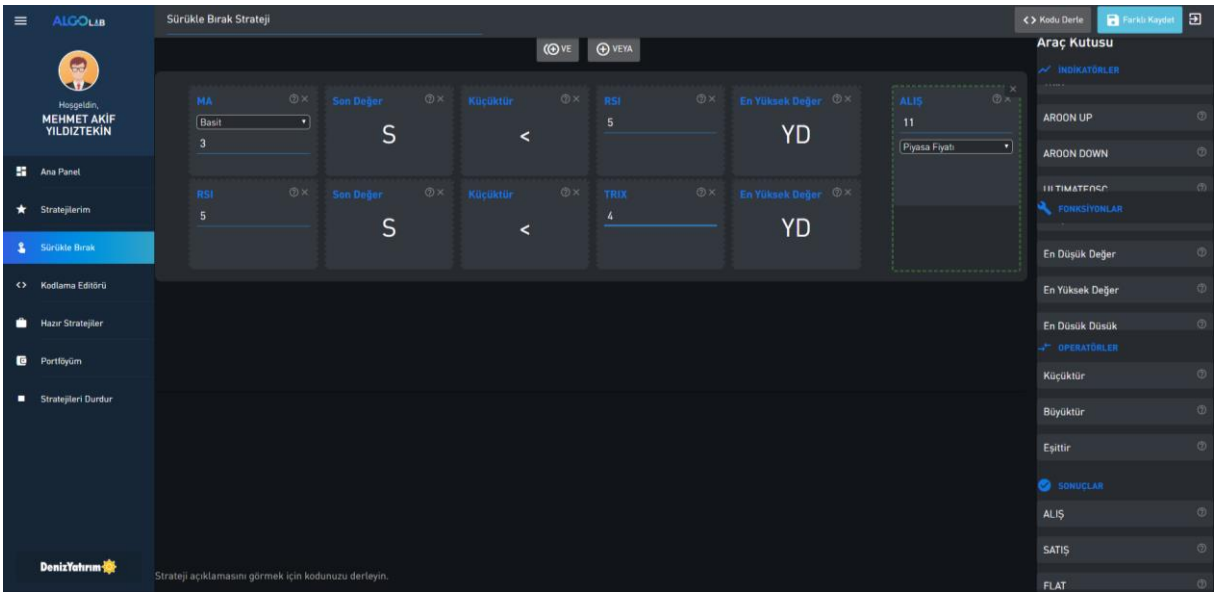
Örnek 2 : Basit 6 periyotluk hareketli ortalamanın(MA) son değeri, 3 periyotluk hareketli ortalamanın(MA) en düşük değerinden küçükse ve aynı zamanda 5 periyotluk hareketli ortalamanın 6 önceki değeri, 2 periyotluk hareketli ortlamanın(MA) ortalamasına eşit ise 7 lot satış emri aç.

Bu örnekte dikkat edilmesi gereken nokta "VE" bağlacıyla oluşturulmuş 2 koşulun birleşik olmasıdır. Bu iki koşul aynı anda gerçekleştiğinde satış işlemi açılacaktır.



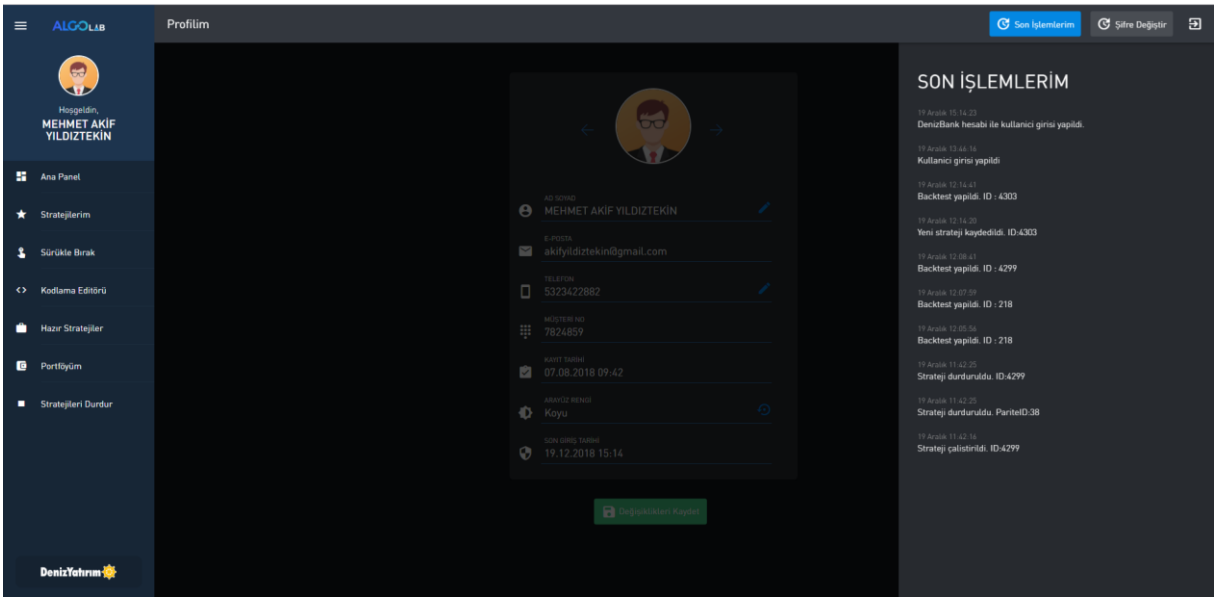
Örnek : Basit 3 periyotluk hareketli ortalamanın son değeri 5periyotluk RSI değerinin en yüksek değerinden küçükse 5 periyotluk alış aç veya 5 periyotluk RSI son değeri, 4 Periyotluk TRIX en yüksek değerinden küçükse 5 periyotluk satış aç anlamına gelir.

Bu örnekte dikkat edilmesi gereken nokta "VEYA" bağlacıyla oluşturulan koşullar birbirinden bağımsız sinyal veren ayrı koşullar olmasıdır. Yani iki koşuldan herhangi biri gerçekleşirse işlem açılacaktır.



Son İşlemlerim

Son işlemlerimde stratejilerinizin açtığı son emirleri ve son işlemlerinizi takip edebilirsiniz.



Algolab Engine Fonksiyonları

2.1 Engine Fonksiyonları

C# yazılım dilinin sizlere sağladığı sınırsız kodlama özgürlüğüne ek olarak algolab özelinde geliştirilmiş fonksiyonları hazır olarak kullanabilmenizi ve bazı hesaplamaları sayfalarca kod yazmadan tek bir döngü ile hesaplamanızı sağlar.

2.2 Engine Fonksiyonlar ve Kullanım Örnekleri:

2.2.1 Kök Alma Fonksiyonu

Bu fonksiyon verilen parametrenin kökünü alır ve "float" tipinde döndürür.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
var sayi = 4f;
float kok = Engine.Kok(sayi); // // kok değişkeninin değeri 2 sayısına eşitlenir.
```

Bu örnekte 4 sayısının karekökü hesaplanarak çıkacak olan değer hesaplanır. Dilerseniz aşağıdaki şekilde bu hesaplamaya bir fonksiyon daha ekleyerek kodlama editörünün sağ tarafında yer alan kısımda işlemin sonucunu da yazdırabilirsiniz.

Örnek :

```
var sayi = 4f;
float kok = Engine.Kok(sayi); // // kok değişkeninin değeri 2 sayısına eşitlenir.
Yazdir(kok.ToString());
```

2.2.2 Fiyat Getir Fonksiyonu

Engine.FiyatGetir fonksiyonu ilk parametre olarak mum barları, ikinci parametre olarak ise fiyatın tipini alarak geriye bir "float" tipinde sayı listesi döndürür.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
var kapanislar = Engine.FiyatGetir(barlar,"kapanis");
var acilislar = Engine.FiyatGetir(barlar,"open");
var düşükler = Engine.FiyatGetir(barlar,"düşük");
var yüksekler = Engine.FiyatGetir(barlar,"H");
```

Üstteki kod örneğindeki "tip" olarak tanımlanan parametrelerde, Türkçe kelimeler ve yabancı kelimeler kullanımının çoğunun çalıştığını göstermek amacıyla yazılmıştır.

2.2.3 Önceki Değer Fonksiyonu

Engine.OncekiDeger fonksiyonu ilk parametre olarak bir "float" tipinde sayı listesi, ikinci parametre olarak ise "int" değişken alır. Listenin sondan kaçıncı elemanının geri döneceğini(float tipinde) belirler.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
var rsi1 = Engine.RSI(barlar,14);
var rsi2 = Engine.RSI(barlar,5);

 if (Engine.OncekiDeger(rsi1,0) > Engine.OncekiDeger(rsi2,3))
 {
 // Burada belirlenen işlem gerçekleşir
 }
```

Yukarıdaki örnekte 14 periyotluk RSI indikatörünün son değeri(0) , 5 barlık RSI indikatörünün 3 önceki değerinden büyükse "if" bloğunun içindeki kodlar çalışacak demektir.

2.2.4 Ortalama Fonksiyonu

Engine.Ortalama fonksiyonu "float" tipinde sayı listesi olarak verilen parametrenin değerinin ortalamasını geri döndürür.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
List degiskenler = new List<float>() {2.5f, 7, 46 , -13.5f};
float ortalama = Engine.Ortalama(degiskenler);
```

Yukarıdaki örnekte ortalama fonksiyonu değiskenler listesinin ortalamasını verir. Başka bir Algolab kullanım örneği ise aşağıdaki şekildedir;

```
 var kapanislar = Engine.FiyatGetir(barlar,"kapanis");
 float ortalama = Engine.Ortalama(kapanislar);
```

Yukarıdaki örnekte ilk örnekten farklı olarak bütün barların ortalamasını verir.

2.2.5 Fiyat Listeleri Fonksiyonu

Engine.FiyatListeleri fonksiyonu fiyatların listesini tek bir dictionary üzerinden çağırmak için kullanılır. Fiyat listeleri fonksyonu belirlenen parametrenin üzerinden mum barların açılış, kapanış, yükseklik ve hacim değerlerini liste olarak kullanılmasına olanak sağlar. Tamamıyla kod efektifliğini artırmak amacıyla oluşturulmuştur.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
var fiyatlar = Engine.FiyatListeleri(barlar); // Değişkenin tanımlanması
 if (Engine.OncekiDeger(fiyatlar["kapanislar"]),0) ==
Engine.OncekiDeger(fiyatlar["acilislar"]),0))
 {
  // Burada belirlenen işlem gerçekleşir
 }
```

2.2.6 Son Fiyatlar Fonksiyonu

Engine.SonFiyatlar fonksiyonu fiyatların son değerlerini tek bir dictionary üzerinden çağırmak için kullanılır. Fiyat listeleri fonksiyonu ile benzerlik gösterir fakat son fiyatlarda liste olarak kullanılmaz sadece son barın açılış, kapanış, hacim, yükseklik değerlerini kullanılmasına olanak sağlar.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
float SonFiyat(List<Bar> liste)
var sonfiyat = Engine.SonFiyat(barlar);

if (sonfiyat > 8 && SonYon != "ALIS" )
{
  // Burada belirlenen işlem gerçekleşir
}
```

Yukarıdaki örnekte eğer sonfiyat değişkeninin değeri 8 değerinden büyükse if kod bloğu çalışır.

2.2.7 Us Alma Fonksiyonu

Engine.Us fonksiyonu parametre olarak verilen "float" deger parametresinin belirlenen katsayısı kadar üssünü alır.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
 float sayi1 = 3;
 float usKatsayi = 2;
 float sonuc = Engine.Us(sayi1,usKatsayi);
```

Yukarıdaki örnekte sonuç 9 olacaktır.

2.2.8 Standart Sapma Fonksiyonu

Engine.StandartSapma fonksiyonu parametre olarak verilen "float" tipinde sayı listesinin standart sapmasını döndürür.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
var kapanislar = Engine.FiyatGetir(barlar,"kapanis");
 var sapma = Engine.StandartSapma(kapanislar);
```

2.2.9 En Yüksek Değer Fonksiyonu

Engine.EnYuksekDeger fonksiyonu "float" tipinde sayı listesi şeklinde verilen parametrenin en yüksek değerini döndürür.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
var liste = new List<float> {1,2,7};
var enYuksek = Engine.EnYuksekDeger(liste);
```

Yukarıdaki örnekte sonuç 7 dönecektir.

2.2.10 En Düşük Değer Fonksiyonu

Engine.EnDusukDeger fonksiyonu "float" tipinde sayı listesi şeklinde verilen parametrenin en yüksek değerini döndürür.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
var liste = new List<float> {1,2,7};
var enDusuk = Engine.EnDusukDeger(liste);.
```

Yukarıdaki örnekte sonuç 1 dönecektir

2.2.11 Değer Getir Fonksiyonu

Engine.DegerGetir fonksiyonu "float" tipinde sayı listesi olarak verilen parametrenin en sondan başlayarak belirtilen "adet" kadarını bir "float" tipinde sayı listesine atarak geri döndürür.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
var dusukler = Engine.FiyatGetir(barlar,"dusuk");
var liste = Engine.DegerGetir(dusukler,5);
```

Yukarıdaki örnekte Sistemdeki bar değerlerinin düşük olanlarının son 5 tanesini "liste" değişkenine atmış oluyoruz.

2.2.12 Sinüs Fonksiyonu

Engine.Sinus fonksiyonu verilen parametrenin sinüs değerini geri döndürür.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
float sayi = 1;
float sonuc = Engine.Sinus(sayi);
```

Yukarıdaki örnekte sonuc değişkeninin değeri 0.0174524064373 sayısına eşitlenir.

2.2.13 Cosinus Fonksiyonu

Engine.Cosinus fonksiyonu verilen parametrenin kosinüs değerini geri döndürür.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
float sayi = 7;
float cos = Engine.Cosinus(sayi);
```

Yukarıdaki örnekte cos değişkeninin değeri 0.992546151641 sayısına eşitlenir.

2.2.14 Tanjant Fonksiyonu

Engine.Tanjant fonksiyonu verilen parametrenin tanjant değerini geri döndürür.

Algolab kullanım örneği ise aşağıdaki şekildedir:

```
var sayi = 2f;
float tan = Engine.Tanjant(sayi);
```

Yukarıdaki örnekte tan değişkeninin değeri 0.0349207694917 sayısına eşitlenir.

2.2.15 Emir Gönderme Fonksiyonu

EmirGonder fonksiyonunun 3 parametresi bulunmaktadır.

1.parametre olarak Sembol (string)

2.parametre olarak Yön (string)

3.parametre olarak Miktar (float)

EmirGonder fonksiyonu girilen parametreler ile piyasa aktif fiyattan emir göndermektedir.

```
EmirGonder(Sembol, "ALIS",5);
EmirGonder(Sembol,"SATIS",120);
float miktar = 500;
string yon = "ALIS";
string baskaSembol = GARAN.E.BIST;
EmirGonder(baskaSembol , yon, miktar);
```

1.örnekte üzerinden çalıştırılan Sembole 5 lotluk bir alış emri gönderir.

2.örnekte üzerinden çalıştırılan Sembole 120 lotluk bir satış emri gönderir.

3.örnekte miktar değişkeninin değeri 500 e eşit, yön alış yönüne ve sembol Garan sembolüne eşit durumda. Parametreleri EmirGonder fonksiyonuna bu şekilde de verebiliriz. Eğer Stratejinizi Garan sembolü üzerinden çalıştıracaksanız, EmirGonder fonksiyonunun ilk parametresine "Sembol" yazmanız yeterlidir. Eğer Üzerinde

çalıştırdığınız sembolden başka bir sembole emir iletmek isterseniz, o sembolün kodunu ilk parametre kısmına yazabilirsiniz.

İNDİKATÖRLER

STOCHRSI NEDİR ?

STOCHRSI indikatörü aşırı alım-satım bölgelerini belirlemek amacıyla oluşturulmuş bir göstergedir.

STOCHRSI indikatörün açılımı Stochastic Relative Strength Index'tir. RSI indikatörü ile STOCH indikatörlerini kullanarak hesaplanır. RSI indikatörünün gürültüsünü belli bir oranda azaltmak amacıyla oluşturulmuştur.

STOCHRSI NASIL KULLANILIR ?

İndikatörün değerinin yaygın olarak kullanılan 20 ve 80 aşırı alım-satım noktalarını kesmesi sinyal olarak kabul edilir. Eğer indikatörün değeri 20 seviyesine aşağıdan yukarı keserse "Alım", 80 değerini ise yukarıdan aşağı kesmesi "Satım" sinyali olarak değerlendirilebilir. 20-80 seviyeleri yaygın olarak 30-70 olarak da kullanılabilmektedir.

STOCHRSI KODLAMA EDİTÖRÜ ÖRNEĞİ

STOCHRSI indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.STOCHRSI fonksiyonu çağırılır.
Parametreleri : barlar (List<Bar>), periyot 1 (int) ,,periyot 2 (int) ,periyot 3 (int)

```
// Kullanım şekli : List<float>STOCHRSI(List<Bar> barlar, int prm1, int prm2, int prm3)

var srsi = Engine.STOCHRSI(barlar,14,7,3);

if (Engine.Kesisim(srsi,20,"yukari"))

{
// Burada belirlenen işlem gerçekleşir
}
```

DEMA NEDİR ?

DEMA indikatörü üssel hareketli ortalama ve üssel hareketli ortalama kullanan başka bir üssel hareketli ortalamanın oluşturduğu bir kombinasyondur.

DEMA indikatörünün açılımı "Double Exponential Moving Average"'dır. DEMA indikatörü yaygın olarak kullanılan üssel hareketli ortalamanın bir zaafiyeti olan gecikmelerin azaltılması amacıyla oluşturulmuştur.

DEMA NASIL KULLANILIR ?

İndikatörün değerinin, fiyatın altında veya üstünde seyretmesine ve ya başka bir hareketli ortalamayla kesişimleri sinyal oluşumu olarak yorumlanabilir. Örnek olarak "10 periyotluk basit hareketli ortalama", "20 periyotluk DEMA"'yı aşağıdan yukarı keserse "Alım" sinyali, aşağı keserse "Satım" sinyali olarak yorumlanabilir.

DEMA KODLAMA EDİTÖRÜ ÖRNEĞİ

DEMA indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.DEMA fonksiyonu çağırılır.
Parametreleri :
barlar (List<Bar>)
periyot (int)

```
// Kullanım şekli : List<float>DEMA(List<Bar> barlar, int prm)
 var dema = Engine.DEMA(barlar,8);
 if (Engine.SonDeger(dema,0) > Engine.SonDeger(dema,3))
 {
 // Burada belirlenen işlem gerçekleşir

 }
```

PARABOLIC SAR NEDİR ?

Parabolic SAR indikatörü J.Welles Wilder tarafından geliştirilmiştir. Pozisyon değişim noktalarını belirlemek için oluşturulmuştur. Ne zaman alım-satım yapılacağı ile ilgili bilgi verir.

İndikatörün adı Stop And Reverse kelimelerinin baş harflerinden gelmektedir. Çift yönlü piyasalarda genellikle trendi oluşmuş dönemlerde kullanılmak üzere geliştirilmiş Parabolic SAR indikatörü, fiyat grafiğinin üzerinde kesikli noktalardan oluşmaktadır.

PARABOLIC SAR NASIL KULLANILIR?

Kullanımında yaygın olarak, indikatör değerinin, fiyatın altında seyretmesinde "Alım", üstünde seyretmesinde ise "Satım" sinyali olarak yorumlanır.

PARABOLIC KODLAMA EDİTÖRÜ ÖRNEĞİ

SAR indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.SAR fonksiyonu çağırılır.
Parametreleri :
barlar (List<Bar>)
periyot 1 (float)
periyot 2 (int)

```
// Kullanım şekli : List<float>SAR(List<Bar> barlar, float prm1,int prm2)

 var psar = Engine.SAR(barlar,0.02f,1);

 if (Engine.SonDeger(psar,0) > Engine.SonDeger(psar,5))

 {
 // Burada belirlenen işlem gerçekleşir
 }
```

LINEARREG NEDİR ?

Linear regression indikatörü kullanılan dönem içerisindeki fiyat hareketlerinin oluşturduğu trende bağlı olarak kendini güncelleyen bir indikatördür.

Hesaplanışı matematikteki "En küçük kareler" metodununa dayanmaktadır. Önceki fiyatları dikkate alarak bir sonraki fiyatın hesaplanan değerini göstermektedir.

LINEARREG NASIL KULLANILIR ?

İndikatörün değerinin, fiyatın altında veya üstünde seyretmesine bakılarak trend yönü belli bir ölçüde tahmin edilebilir. Kullanımı açısından hareketli ortalamaya benzemektedir. Uzmanlar düşük periyot kullanan bir hareketli ortalamayla LINEARREG indikatörünün değerlerinin kesişimine göre sinyal üretmektedir. Eğer LINEARREG değeri hareketli ortalamadan yüksekse "Satım" sinyali veya trendin "Aşağı" devam edeceği, düşük olduğu durumda ise "Alım" sinyali veya trendin "Yukarı" yönlü devam edeceği yorumu yapılabilir.

LINEARREG KODLAMA EDİTÖRÜ ÖRNEĞİ

LINEARREG indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.LINEARREG fonksiyonu çağırılır.
Parametreleri :
fiyatlar (List<float>)

```
// Kullanım şekli : List<float>LINEARREGSLOPE(List<float> fiyatlar, int prm)
 var kapanislar = Engine.FiyatGetir(barlar,"kapanis");
 var lreg = Engine.LINEARREG(kapanislar,14);
 if (Engine.SonDeger(lreg,0) > 0)
 {
   // Burada belirlenen işlem gerçekleşir
 }
```

PPO NEDİR ?

PPO indikatörü fiyat üzerinden olulşan iki hareketli ortalaması arasındaki farkı gösteren bir indikatördür.

PPO indikatörünün açılımı "Price Oscillator"'dür. Değerleri fiyata göre değişkenlik gösterir.

PPO NASIL KULLANILIR ?

PPO indikatörünün genel kullanımında 0 seviyesi alım-satım sinyali için kullanılır. Eğer indikatör 0 seviyesini yukarıdan aşağıya doğru keserse "Satım", aşağıdan yukarıya doğru keserse ise "Satım" sinyalinin geldiği çıkarımı yapılabilir.

PPO KODLAMA EDİTÖRÜ ÖRNEĞİ

PPO indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.PPO fonksiyonu çağırılır.
Parametreleri :
barlar (List<Bar>)
periyot 1(int)
periyot 2(int)

```
// Kullanım şekli : List<float>PPO(List<Bar> barlar, int prm1, int prm2)

 var ppo = Engine.PPO(barlar,10,21);

 if (Engine.Kesisim(ppo,0,"yukari"))

 {
```

```
// Burada belirlenen işlem gerçekleşir
 }
```

TRIX NEDİR ?

TRIX indikatörü, 3 derece hareketli ortalama kullanan bir hareketli ortalama versiyonudur.

TRIX indikatörü hesaplanırken kapanış fiyatının hareketli ortalamasını alır. Bu aşamadan sonra hesaplanan hareketli ortalamanın da hareketli ortalamasını alarak, klasik bir hareketli ortalamaya göre fiyat değişimlerinden daha düşük bir ölçüde etkilenen bir indikatördür.

TRIX NASIL KULLANILIR ?

TRIX indikatörünü kullanırken 0 referans seviyesine kesişim durumuna ve TRIX indikatörünün üzerine atılan başka bir hareketli ortalama ile olan kesişimlerine bakılabilir. İlk yöntemde TRIX indikatörünün 0 seviyesini yukarıdan aşağıya doğru kestiğinde "Satım" , aşağıdan yukarıya kestiğinde ise "Alım" sinyalinin oluşumu yorumu yapılabilir. İkinci yöntemde ise TRIX ve hareketli ortalamadan, daha hızlı olanın yavaş olanı aşağıdan yukarıya doğru kesmesi "Alım", aksi durumda ise "Satım" sinyali yorumu yapılabilir.

TRIX KODLAMA EDİTÖRÜ ÖRNEĞİ

TRIX indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.TRIX fonksiyonu çağırılır.
Parametreleri :
barlar (List<Bar>)
periyot (int)

```
// Kullanım şekli : List<float>TRIX(List<Bar> barlar, int prm1)
 var trix = Engine.TRIX(barlar,10);
 if (Engine.Kesisim(ppo,0,"asagi"))
 {
 // Burada belirlenen işlem gerçekleşir

 }
```

ROC NEDİR ?

ROC indikatörü belirli bir vadedeki fiyat değişimini ölçmeye yarayan bir indikatördür.

ROC indikatörünün açılımı "Rate Of Change"'dir. Hesaplanırken belirtilen vade içerisinde meydana gelen fiyat değişiminin periyot kadar önceki fiyatına bölerek başlangıç-bitiş arasındaki fiyat yüzdesini hesaplar.

ROC NASIL KULLANILIR ?

Paritenin fiyatına göre değişkenlik göstermesiyle beraber +5 ve -5 aralıkları arasında salınım yapar. ROC'un değeri 0 referans seviyesini yukarıdan aşağıya kestiğinde "Satım", aşağıdan yukarıya kestiğinde ise "Alım" sinyali yorumu yapılabilir. Bununla beraber ROC'un limit seviyelerine(+5 ve -5) yakınlaşması ile, pozitif yön ise yukarı, negatif yön ise trendin aşağı doğru hızlıca gitmekte olduğu yorumu da yapılabilir.

ROC KODLAMA EDİTÖRÜ ÖRNEĞİ

ROC indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.ROC fonksiyonu çağırılır.
Parametreleri :
barlar (List<Bar>)
periyot (int)

```
// Kullanım şekli : List<float>ROC(List<Bar> barlar, int prm)

 var roc = Engine.ROC(barlar,10);
 if (Engine.Kesisim(roc,0,"asagi"))
 {
// Burada belirlenen işlem gerçekleşir
 }
```

CMO NEDİR ?

CMO indikatörü genel olarak aşırı alış-satış bölgelerini belirleyen bir indikatördür.

CMO indikatörünün açılımı "Chande Momentum Oscilator"dür. Belirli bir vade arasındaki fiyat değişimlerini hesaplayarak trendin yönünü ve gücünü ölçüp hesaplanan değerleri bir aralık arasında gösterir.

CMO NASIL KULLANILIR ?

CMO indikatörünün referans değerleri genel olarak +50 ve -50 olarak belirlenmiştir. +50 seviyesi aşırı alımın olduğunu ve kısa bir vadede paritenin düşme eğilimine girebileceği, -50 seviyesi ise aşırı satımın olduğunu ve kısa bir vadede paritenin yükselme eğilimine girebileceği hakkında fikir verir. Başka bir kullanım şeklinde ise CMO indikatörünün üzerine atılan bir hareketli ortalama ile gerçekleşen kesişimlerine de bakılabilir.

CMO KODLAMA EDİTÖRÜ ÖRNEĞİ

CMO indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.CMO fonksiyonu çağırılır.
Parametreleri :
barlar (List<Bar>)
periyot (int)

```
// Kullanım şekli : List<float>CMO(List<Bar> barlar, int prm)

 var cmo = Engine.CMO(barlar,14);
 if (Engine.Kesisim(roc,-50,"asagi"))
 {
 // Burada belirlenen işlem gerçekleşir
 }
```

AROON NEDİR ?

AROON indikatörü fiyatları analiz ederek bir trendin var olup olmadığını bulmayı amaçlayan bir indikatördür.

AROON indikatörü belirlenen periyot kadar önce oluşmuş bir tepe fiyatı ile aynı vadedeki dip fiyatı arasında geçen zamanı baz alarak oluşturulur. AROON indikatörünün "UP" ve "DOWN" olmak üzere iki çizgisi bulunur. AROON UP hesaplanan tepeden itibaren geçen zamanı, AROON DOWN ise hesaplanan dipten itibaren geçen zamanı belirtir.

AROON NASIL KULLANILIR ?

AROON indikatörünün referans değerleri 0 ile 100 olarak belirlenmiştir. AROON UP göstergesinin 100 seviyesine yaklaşması veya değmesi durumu ile AROON DOWN göstergesinin 0-30 bandında seyretmesi yukarı trendin güçlü olduğunu göstermektedir. Aynı şekilde AROON DOWN göstergesinin 100 seviyesine yaklaşması veya değmesi durumu ile AROON UP göstergesinin 0-30 bandında seyretmesi aşağı trendin güçlü olduğunu göstermektedir. Başka bir kullanım şeklinde ise bu iki çizginin kesişimleri yorumlanıp sinyal üretilir. Eğer UP çizgisi DOWN çizgisini kesip yukarı yönlü giderse "Alım" , aksi durumda ise "Satım" sinyali dikkate alınır.

AROON KODLAMA EDİTÖRÜ ÖRNEĞİ

AROON UP indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.AROONUP fonksiyonu çağırılır. AROON DOWN indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.AROONDOWN fonksiyonu çağırılır.
Parametreleri :

barlar (List<Bar>)
periyot (int)

```
// Kullanım şekli (UP): List<float>AROONUP(List<Bar> barlar, int prm)
 // Kullanım şekli (DOWN): List<float>AROONDOWN(List<Bar> barlar, int prm)

 var up = Engine.AROONUP(barlar,14);
 var down = Engine.AROONDOWN(barlar,14);
 if (Engine.Kesisim(up,down,"yukari"))
 {
 // Burada belirlenen işlem gerçekleşir
 }
```

MA NEDİR ?

MA indikatörü Dünya'nın en çok kullanılan bir trend takip indikatörüdür.

MA indikatörünün açılımı "Moving Average'dır. Bir çok hesaplama türü bulunur. "Basit","Üssel","Ağırlıklı" ortalama bunlardan bazılarıdır. Basit hesaplamalar yapmasına rağmen son derece kullanışlı bir indikatördür.

MA NASIL KULLANILIR ?

MA indikatörü kullanılan periyota göre çok değişkenlik göstermesine rağmen çoğu zaman fiyat ile benzer hareket etmektedir. Kullanımında bir çok varyasyon denenebilmektedir. Örneğin 20 periyotluk bir MA ile 5 periyotluk bir MA'nın kesişiminden, 10 periyotluk bir MA'nın fiyat ile kesişiminden veya iki adet MA'nın birbirlerden olan uzaklıkları belli bir değeri geçtiğinde sinyal üretilebilir.

KODLAMA EDİTÖRÜ ÖRNEĞİ

MA indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.MA fonksiyonu çağırılır.
Parametreleri :
barlar (List<Bar>) veya fiyatlar (List<float>)
tip (string)
periyot (int)

```
// Kullanım şekli : List<float>MA(List<Bar> barlar, string tip, int periyot)
 // Kullanım şekli : List<float>MA(List<float> fiyatlar, string tip, int periyot)

 var dusukler = Engine.FiyatGetir(barlar,"dusuk");
 List<float> rastgeleSayilar = new List<float>() {5,9,4,7,2,5,7,2,13,56,21,79,34};
 var ma1 = Engine.MA(barlar,"üssel",20);
 var ma2 = Engine.MA(dusukler,"basit",5);
 var maRastgele = Engine.MA(rastgeleSayilar,"agirlikli",3);

if (Engine.Kesisim(ma2,ma1,"yukari"))
{
 // Burada belirlenen işlem gerçekleşir

 }

if (Engine.OncekiDeger(maRastgele,2) > 10 &&
Engine.MutlakDeger(Engine.SonDeger(ma1) - Engine.OncekiDeger(ma2,3)) > 5)

{
// Burada belirlenen işlem gerçekleşir
}
```

Yukarıdaki örnekte ilk if bloğu için eğer ma2'nin son değeri, ma1'in son değerini aşağıdan yukarı doğru keserse belirlenen işlem çalışacaktır.

İkinci if bloğu için eğer maRastgele'nin 2 önceki değeri 10'dan büyük ise ve ma1'in son değeri ile ma2'nin 3 önceki değerinin birbirinden farkı 5'ten büyük ise belirlenen işlem çalışacaktır.

ULTIMATEOSC NEDİR ?

ULTIMATEOSC indikatörü aşırı alım-satım noktalarını belirlemeye yarayan bir indikatördür.

ULTIMATEOSC indikatörünün açılımı "Ultimate Oscillator"dır. İndikatörün değerleri 0 ile 100 arasında salınım yapar. RSI indikatörünün bir benzeridir.

ULTIMATEOSC NASIL KULLANILIR ?

3 adet parametre alan ULTIMATEOSC indikatörünün genel olarak kullanılan parametreleri 7,14 ve 28'dir. Bu parametreler indikatörün kısa-orta-uzun vade anlayışına göre belirlenmiştir. ULTIMATEOSC indikatörü bu periyotları kullanarak hesaplamalarını yapar. Genellikle aşırı alım-satım noktalarına göre işlem yapılması uygun görülmüştür.

KODLAMA EDİTÖRÜ ÖRNEĞİ

ULTIMATEOSC indikatörünü Algolab Kodlama Editöründe kullanabilmek için Engine.ULTIMATEOSC fonksiyonu çağırılır.
Parametreleri :
barlar (List<Bar>)
periyot 1(int)
periyot 2(int)
periyot 3(int)

```
// Kullanım şekli : List<float>ULTIMATEOSC(List<Bar> barlar, int prm1, int prm2, int prm3)

 var ult = Engine.ULTIMATEOSC(barlar,7,14,28);
 if (Engine.Kesisim(ult,30,"yukari"))
 {
 // Burada belirlenen işlem gerçekleşir
 }
```