İŞ ZEKASI VE VERİ MADENCİLİĞİ

Şadi Evren

İçindekiler

1. Giriş ve Kitabın Kullanılışı

Bu bölümün amacı kitapta kullanılan sembollerin açıklanması ve kitap hakkında tanıtıcı bilgi vermektir.

1. Giriş ve Kitabın Kullanılışı

Bu kitabın amacı, veri madenciliği (data mining) çalışmalarının iş zekası (business intelligence) konusunda kullanımı hakkında bilgi vermektir. Kitap, veri madenciliği konusundaki temel algoritma ve metotları açıklayacak ve ardından bu metotların iş zekası uygulamalarındaki kullanımını okuyucuya kazandırmaya çalışacaktır. Kitabın okunmasından önce temel istatistik ve yönetim bilgilerinin mevcut olması tavsiye edilir.

1.1. Kitabın konulara göre bölüm ve alt bölümlere ayrılışı

Bu kitap iki ana bölümden oluşmaktadır. İlk bölümde, veri madenciliğinin temelleri ve iş zekası uygulamalarında kullanılması gereken karakteristik metot

ve algoritmalara yer verilmiştir. İkinci kısımda ise, bu algoritma ve metotların, iş zekası uygulaması olarak kullanılması, örnekler üzerinden gösterilmiştir. Kitaptaki uygulamalar, Weka isimli, açık kaynak kodlu ve JAVA dilinde geliştirilmiş olan uygulama üzerinden açıklamaktadır. Weka programı ücretsiz olarak edinilebilir ve kitapta indirilmesi ve kurulması ile ilgili bir bölüm bulunmaktadır. Kitapta, genel olarak izlenen sıralama aşağıdaki şekildedir.

1. Veri madenciliği konusunun açıklandığı ve teorik anlatımın bulunduğu ilk alt bölüm.
2. Basit bir örnek üzerinde yöntemin uygulanması.
3. Bir iş zekası problemi üzerinde uyarlanması.
4. Weka kullanılarak iş zekası probleminin çözümü.

Genel olarak her bölümde yukarıdaki yapı izlenmesine karşılık, bazı giriş seviyesi bölümlerde bu yapının dışına çıkılmıştır. Kitabın başlangıcında, okuyucunun ihtiyacı olabilecek temel istatistik bilgilerine yer verilmekte ve istatistik konusunda genel bir tekrar yapılmaktadır. Ancak okuyucunun istatistik konusunda alt yapısı olması tavsiye edilmektedir.

1.2. Kitabın dili ve terminoloji

Kitap, şimdiye kadar anlaşılacağı üzere, Türkçe olarak neşredilmiştir. Ancak gelişen iletişim teknolojileri ve bilhassa İnternet sayesinde İngilizcenin rolü Türk dilinde de hissedilmektedir. Okuyucunun araştırma yapabilmesi ve aynı terimin Türkçeye farklı şekillerde çevrilmesinden kaynaklanan karmaşanın giderilmesi için bu kitapta Türkçe terminolojinin yanında İngilizce terimler parantez içerisinde verilecektir. Örneğin:

"Yönetim Bilişim Sistemleri (Management Information Systems)" şeklinde.

Kitapta bulunan terminoloji hakkında ansiklopedik bilgiye ihtiyaç duyulması halinde yine kitabın yazarı Şadi Evren ŞEKER tarafından hazırlanan www.bilgisayarkavramlari.com adresinden yararlanılabilir. Kitap konusu dışında kalan kavramlara kitapta değinilmeyecektir, dolayısıyla okuyucu bu konudaki araştırmalarını verilen bu adresten karşılayabilir.

Ayrıca bu kitapta konusu geldikçe önemli olan noktalar aşağıdaki şekilde:

ÖRNEK UYARI KUTUSU

!

Uyarı kutularında anlatılacaktır. Bu anlatılanlar ya yaşanan bir tecrübeye dayalı uyarılar olacak ya da konu için büyük öneme sahip noktalar olacaktır. Okuyucunun bu uyarı kutularını dikkate alması tavsiye edilir.

Ayrıca bilgilendirme kutuları da aşağıdaki şekilde kitap içerisinde yer bulmaktadır.

ÖRNEK UYARI KUTUSU

?

Bu kutuların içerisinde, genel olarak anlatılan konuyu açıklayan bilgiler bulunmaktadır.

1.3. Kitaptaki örnek ve kodların kullanılması

Kitapta bulunan örnek kodların kullanılması için öncelikle bu kodların kullanılabildiği Weka yazılımına ihtiyaç duyulmaktadır. Kitabın EK 1 ekinde ilgili yazılım ortamının kurulumu ile ilgili resimli anlatım bulunmaktadır. Okuyucu bu bölümden faydalanarak örnekleri çalıştırabileceği ortamı kurabilir. Ayrıca Java dili üzerinde geliştirme yapacak kişiler için, bu kitabın kapsamında bir programlama bilgisi sunulmayacaktır. Yazılım geliştirmeyi hedefleyen kişiler, ilgili yazılım ortamını kurmak ve basit birer uygulama geliştirmek için EK 2 başlıklı ekte bulunan yardımcı yazıdan faydalanabilirler.

1.4. Konu sonu alıştırmaları

Kitabın sonunda bulunan çalışma soruları, kitabın bir ders kitabı olarak kullanılması ihtimaline karşı eklenmiştir. Okuyucu konuları okuyarak öğrendiği bilgileri bu alıştırmalar ile pekiştirerek geliştirebilir.

Ayrıca ders kitabı olarak okutulduğu ortamlarda ödev olarak da bu sorulardan istifade edilebilir. Konu sonunda bulunan soruların çözümleri kitabı okutmaya karar veren hocalara, talep edilmesi durumunda verilecektir. Bunun için www.sadievrenseker.com/kitap adresinde bulunan hoca formunun doldurulması gerekmektedir.

Konu sonu sorularının yanlarında bulunan işaretlerin anlamları şu şekilde sıralanabilir:

* Diğer sorulara göre zor sorular

A Okuyucunun kitaptaki bilgilere ilave olarak araştırma yapması gereken sorular

J Java dilinde kodlanabilir sorular

W Çözmek için Weka'nın gerektiği sorular

2. Temel Kavramlar

Bu bölümde, temel iş zekası kavramlarına giriş yapılacak ve basit bir örnek üzerinden iş zekası sürecinin adımları gösterilmeye çalışılacaktır.

2. Veri Madenciliğine Giriş

Veri madenciliğine giriş konusunda, öncelikle veri (data) kavramından bahsedilecek, ardından bu verinin işlenme yöntemleri hakkında bilgi verilecektir. Verinin işlenmesi yöntemlerinden birisi olan veri madenciliği kavramı açıklandıktan sonra çeşitli veri madenciliği yöntemlerine giriş mahiyetinde bilgilendirme yapılacaktır. Bu yöntemlerin her birisi daha sonra birer bölüm olarak ele alınacaktır.

2.1. Veri Nedir?

İngilizcede terim olarak yerleşmiş ve çoğu çalışma için hayatî öneme sahip kelimeleri açıklayarak başlayalım. Seviyelerine göre 4 kelime yani "Data", "Information", "Knowledge" ve "Wisdom" kelimelerini, Türkçede çoğu zaman sadece bilgi ile karşılayan tercümeler bulunuyor. Ancak bu 4 kelime de farklı anlamlara sahip

ve aralarındaki farklar burada anlatılacaktır. Öncelikle İngilizcede bulunan ve Türkçede aynı kelimeyle karşılanan 4 kelime için 4 farklı karşılık bulalım. Bunlar aşağıda verilmiştir:

- Data: Veri
- Information: Malumat
- Knowledge: Bilgi
- Wisdom: İrfan, bilgelik

Yukarıdaki kelimeler sırayla verilmiştir ve bu sırada verinin işlenmesi ve işlendikçe daha değerli sonuçlara erişilmesi mümkündür. Yani basitçe irfan, bilgiden; bilgi, malumattan; malumat ise veriden işlenerek çıkarılır ve en ham hal olan veri en değersiz ve çoğu zaman en az işe yarayan şeyken irfan en değerli ve en çok işe yarayan sonuç hâline gelmektedir.

Veri, tanım itibariyle, herhangi bir işlemeye tabi tutulmadan, gözlem veya ölçüm yöntemleri ile ortamdan elde edilen her türlü değerdir. Örneğin saatin kaç olduğu, ortamın anlık basıncı, bir firmadaki bir çalışanın ismi, maaşı veya yaşı gibi değerlere veri denilir.

Malumat (information) ise bu verinin işlenmiş halidir. Örneğin bir arabanın hızını veri olarak kabul edebiliriz. Bu hızın ne anlama geldiği, örneğin tehlikeli bir hız veya varılacak yere geç kalmaya sebep olacak bir hız oluşu malumat olarak kabul edilebilir. Bu hızın arttırılması, azaltılması, geçmiş tecrübeler ile kıyaslanması gibi eylemlerin sonucu ve yolu ise bilgi olarak kabul edilebilir. Şoförlük bilgisi, tecrübe ve melekesi ise irfan olarak kabul edilebilir. Yani örneğin Şoför Ali

denildiğinde, Ali'nin şoförlük bilgisine sahip olduğu kabul edilir. Yukarıdaki örnekte bütün kelimeler tek bir örnek üzerinde gösterilmiştir. Elbette şoförlük bilgisinin altında hız dışında sayısız farklı veri, ve bu verilerin sunduğu malumatlar ve bu malumatlardan elde edilen bilgiler bulunmaktadır. Dolayısıyla bilgelikten veriye kadar giden yol belirsiz (nondeterministic) bir yoldur. Bu yolun tersi de ne yazık ki belirsizdir çünkü hız verisi şoförlük için kullanılabileceği gibi örneğin muavinlik, yolculuk, işletmecilik (otobüs işletmesi) gibi farklı irfanlar için de anlamlı olabilir. Dolayısıyla bu kavramları doğrusal bir yapıya oturtmak yanlış olacaktır.

2.2. Veri Tabanı Nedir?

Herhangi bir çalışma alanında, çok sayıdaki veri kaynaklarından akan veriler düzgün bir şekilde tutulmalı ve bu verilere istenildiği zaman kolayca erişilebilmelidir. Örneğin bir işletmedeki satış kayıtlarının tutulması, bu satışlarda bulunan ürün detaylarının, satış fiyatlarının ve satış zamanlarının tutulması veri tabanlarının (database) görevidir. Veri tabanı kavramı genel bir kavram olmakla birlikte, bu kavramı gerçekleştiren yazılımlara veri tabanı yönetim sistemleri (database management systems) ismi verilir. Veri tabanı yönetim sistemleri, verinin saklanması dışında çeşitli farklı görevleri de icra ederler.

Kitabın konusu dışında olmasına karşılık, giriş mahiyetinde veri tabanı yönetim sistemleri için şunları söyleyebiliriz:

Veri tabanı yönetim sistemi terimi (database management system), tam olarak bir veri tabanını ve bu veri tabanı üzerindeki yönetimle ilgili bütün yazılımları kapsamaktadır.

Veri tabanlarını iki seviyeye ayırmak mümkündür:

- Mantıksal Katman (Logical Layer)
- Fiziksel Katman (Physical Layer)

Buna göre mantıksal katman olan ve izafi olarak üstte ifade edilen katman, insan düşüncesine daha yakın ve düşünmesi ve kullanması biz insanlar için daha rahat olan katmandır. Fiziksel katman ile ifade edilen katman ise bilgisayarın verileri nasıl tuttuğu gibi daha somut şeylerle ilgilenirken, insanın günlük düşüncesinden biraz daha uzak, bilgisayarlar tarafından daha kolay anlaşılabilir bir yapıya sahiptir. Örneğin dosya yönetimi (file organisation) konusunda yapılan çalışmalar bu gruba girmektedir.

Daha net bir anlatımla elimizde saklanması istenen bir veri vardır. Bu veriyi mantıksal katmanda tablolara, kolonlara bölerken, fiziksel katmanda disk üzerindeki bloklardan, segmentlerden bahsediyoruz demektir.

İşte veritabanı yönetim sistemi ile kastedilen sistem bu iki katmanı çekirdek olarak içermelidir. Veri tabanı yönetim sistemleri bunun üzerinde kullanıcı yönetimi, sistem yedekleme ve geri yükleme, performans görüntüleme ve iyileştirme, veri tabanını dağıtık çalıştırma gibi pek çok ilave yazılımı barındıran sistemlerdir.

Kısaca sağladıkları faydaları aşağıdaki şekilde sıralayabiliriz:

- Veri Bağımsızlığı (Program ve verinin ayrılması, Data Independence)
- Verimli veri erişilebilirliği (Verinin en hızlı erişilebilir şekilde saklanması, Efficient Data Access)
- Veri bütünlüğü (Pek çok parçadan oluşan verinin tek elden yönetilerek silme ve güncellemelerin takibinin mümkün olması, Data Integrity)
- Veri Güvenliği (Verinin şifrelenebilmesi ve kullanıcı bazında kontrol edilebilmesi, Security)
- Veri Yönetimi (Data Administration)
- Eşzamanlı Erişim (Aynı anda veriye pek çok farklı program ile erişebilme imkanı, Concurrent Access)
- Sistem kurtarması (Sistemdeki yedekleme ve kurtarma özellikleri, crash recovery)
- Daha hızlı program geliştirme imkanı (Reduced application development time)

Ve yukarıdaki bu özelliklerin dışında pek çok faydaları sağlamaktadır.

2.3. Veri Madenciliği (Data Mining) Nedir?

Kitabın tamamının bu soru etrafında şekillendiğini söyleyebilecek olmamıza karşılık, bu aşamada genel bir tanım yapmaya çalışacağız. Veri madenciliği, çeşitli şekillerde ve çeşitli kaynaklardan toplanan verilerin üzerinde işlem yapılarak anlamlı bilgilerin çıkarılmasıdır.

Geniş anlamda, örneğin, bir firmadaki kişilerin, maaş, sigorta, KDV dökümlerinden (veri), bu firmadaki bir personelin ortalama maliyetinin bulunması (bilgi) gibi basit bir işlem bile veri madenciliği olarak kabul edilebilir. Ancak daha net bir tanımlama için, iş zekası

çalışmalarını anlamak gerekir. İş zekası yaklaşımına göre, veri madenciliği bu sürecin bir parçası olarak düşünülmelidir.

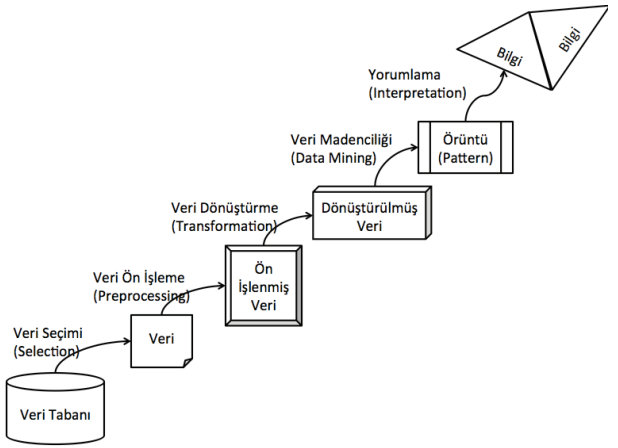
2.4. İş Zekası (Business Intelligence) Nedir?

İş zekası kavramının farklı açılardan farklı tanımları olmasına karşılık, genel anlamda, bir şirketteki veya organizasyondaki çeşitli seviyelerde ve çeşitli kaynaklarda bulunan verilerin işlenmesi ve karar süreçlerine etkili ve başarılı bir şekilde taşınmasıdır. Ayrıca, yönetim bilişim sistemleri açısından, en kısa tanımıyla, veri tabanından bilgiye (knowledge) ulaşmayı amaçlayan süreçtir. Literatürde KDD ismi verilen ve "Knowledge Discovery in Databases" kelimelerinin baş harflerinden oluşan ve Türkçeye Veri Tabanlarında Bilgi Keşfi olarak çevirebileceğimiz ve kısaca VTBK olarak ifade edebileceğimiz bir işlem olarak tanımlamak mümkündür [1].

Buna göre bir VTBK süreci aşağıdaki adımlardan oluşur denilebilir:

- Veri Seçimi (Selection)
- Veri Ön İşleme (Preprocessing)
- Veri Dönüştürme (Transformation)
- Veri Madenciliği (Data Mining)
- Yorumlama (Interpretation)

(1) Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth: The KDD Process for Extracting Useful Knowledge from Volumes of Data. Commun. ACM 39(11): 27-34 (1996)



Sırasıyla bütün işlemler, veri tabanından başlayarak, bilgiye (knowledge) ulaşana kadar uygulanır.

Veri tabanında uygulanan basit seçme işlemi, ilgili bilgi çıkarımı için gereken veriler ile sınırlıdır. Yani bir firmanın çok sayıdaki verisinden sadece ilgili veriler seçilerek işleme başlanır. Bu seçme işlemi sonucunda elde edilen sonuç aynı zamanda veri ambarı (data warehouse) olarak da isimlendirilen özel bir teknoloji üzerinde tutulabilir. Buradaki amaç veri seçim işlemini sürekli, hızlı ve güncel tutabilmektir. Ancak VTBK sürecinde bir veri ambarı bulunması şartı yoktur. Amacına uygun olarak seçilmiş olan veri, sürecin devamı için yeterlidir.

Ardından veri üzerinde ön işleme yapılır. Bu ön işleme süreci, çoğu zaman kirli verinin (gürültülü veri, noisy data) çözülmesi işlemidir. Örneğin firmada çalışan kişilerin maaşları üzerinden bir işlem yapılmak istenir ancak her çalışanın maaşı kayıtlı değildir. Bu

durumda eksik olan kişiler için bir çözüm üretilmesi gerekir. Veya bazı durumlarda verilerde çelişik durumlar olabilir. Mesela bir hastanın cinsiyeti erkek ancak sadece hamilelikte kullanılan ilaçlar kullanıyor olması veya maaşın eksi değer taşıması gibi çelişik durumlar ön işleme kısmında çözülmelidir. Burada "çözülme" kelimesi özellikle kullanılmıştır çünkü genel kanının aksine silmek her zaman için en iyi çözüm olmayabilir.

Üçüncü aşamada, verinin dönüştürülmesi (transformation) işlemi devreye girer. Bir önceki aşamada temizlenmiş olan veri üzerinde çeşitli dönüşüm işlemleri uygulanarak verinin veri madenciliğine uygun hale getirilmesi amaçlanır. Örneğin çok büyük miktardaki verinin işlenmesi, mevcut donanım ve teknolojik imkanlarla sağlanamıyorsa, bu verinin küçültülmesi, hatta kayıpsız küçültülmesi veya verinin üzerinden istenilen özellik çıkarımları bu aşamada yapılır.

Veri madenciliği (data mining) aşamasında ise bu kitapta detaylıca anlatılacak olan çeşitli veri madenciliği yöntemleri kullanılarak, bu aşamaya kadar gelen veri üzerinden, yorumlamaya uygun, anlamlı sonuçları olan ve çıkarılmak istenen bilgiye hizmet edecek işlemler yapılır.

Son olarak yorumlama aşamasında, veri madenciliği aşamasından çıkarılan örüntünün (pattern) yorumu yapılarak bu çıkarılan örüntünün anlamı sorgulanır. Bazı durumlarda çıkarılan örüntü anlamlı olmayabilir. Örneğin bir markette, sodalı içecek alan kişilerin çoğunun patates cipsi alması anlamlı bir sonuç olurken,

sodalı içecek alanların çoğunun ayakkabı da alması sorgulanabilir. Böyle bir sonuç çıkıyorsa, geri dönüp daha önceki adımların tekrar gözden geçirilmesi gerekir. Ancak tam da bu aşamada uyarmak gerekir ki, iş zekası uygulamalarında her zaman hata payı bulunmaktadır.

2.5. Örnek Çalışma

Örnek bir senaryo olarak, kurgusal bir firmadaki personel bilgilerini ele alacağız. Henüz iş zekası kavramına aşina olmadığımız için ve anlaşılması kolay olması amacı ile, az sayıdaki kayıt üzerinden genel bir iş zekası sürecini göstermek amaçlanmaktadır. Bu amaçla aşağıdaki tabloda gösterilmiş olan veri tabanı kayıtlarını ele alalım:

İsim	Çalışma Süresi	Maaş	Mezuniyet	Departman
Elif Şeker	3	3000	Bilgisayar Müh.	Bilgisayar
Ali Demir	5	4000	İşletme	Muhasebe
Cem Yıldız	4	5000	Bilgisayar Müh.	Bilgisayar
Ahmet Yılmaz	3	2500	İşletme	——
—	–	1000	İşletme	Muhasebe
Veli Demir	4	2500	İşletme	Yönetim

Yukarıdaki kayıtlara bakıldığında, bir firmada çalışan kişilerin isimleri, firmadaki çalışma süreleri, maaşları, mezuniyetleri ve çalıştıkları departman bilgileri görülmektedir. Amacımız yeni alacağımız personelin mezun olduğu bölüme göre firma için değerini bulmak

olsun. Yani, 'Yeni mezun birisi hangi bölümden gelirse firma için ne kadar değerli olur?' sorusunu soruyoruz. Öncelikle iş zekası sürecini işletelim ve seçim aşaması (selection) ile başlayalım:

Çalışma Süresi	Maaş	Mezuniyet
3	3000	Bilgisayar Müh.
5	4000	İşletme
4	5000	Bilgisayar Müh.
3	2500	İşletme
–	1000	İşletme
4	2500	İşletme

Bizim amacımız için bir anlam ifade etmeyen isim ve bölüm bilgilerini sildik. Amacımız, mezuniyet bilgisine göre personelin değerliliğini bulmak. İkinci aşama olan ön işleme aşamasına geçiyoruz. Buna göre yukarıdaki tabloda çalışma süresi belli olmayan ve bizim için veri madenciliği aşamasında sorun oluşturabilecek bir kayıt bulunuyor. Ön işleme aşamasında bu kaydı temizleyelim.

Çalışma Süresi	Maaş	Mezuniyet
3	3000	Bilgisayar Müh.
5	4000	İşletme
4	5000	Bilgisayar Müh.
3	2500	İşletme
4	2500	İşletme

Tablomuzun son halinde ön işlemeden çıkarılmış ve dönüştürülmeye hazır verimiz bulunuyor. Şimdi veriyi dönüştürme aşamasına geçebiliriz. Buna göre veri üzerinde birleştirme (aggregation) uygulamak isteyelim ve yukarıdaki tabloda bulunan mezuniyet bilgilerine göre tablodaki diğer alanların ortalama değerlerini alalım:

Ortalama Çalışma Süresi	Ortalama Maaş	Mezuniyet
3,5	4000	Bilgisayar Müh.
4	3000	İşletme

Tablomuzun son halinde, her mezuniyet alanı için ortalama çalışma süresi ve ortalama maaş hesaplaması yapılmıştır.

İş zekası sürecinin 4. aşaması olan veri madenciliği aşamasına geçebiliriz. Bu aşamada henüz yeni olduğumuz basit bir model kullanarak bir kişinin maaşının yüksek olmasının firma için değerli olduğu ve ortalama çalışma süresinin uzun olmasının da firma için değerli olduğu kabulünü yapalım. Bu değerler yoruma açıktır ancak başlangıç seviyesinde basit bir model kurma amacında olduğumuz için problemi basitleştirelim ve bu kabullere göre ortalama çalışma süresi ile ortalama maaşın çarpımının bir personelin firma için değerini belirlediğini düşünelim. Bu durumda veri madenciliği işleminin sonucu aşağıdaki şekilde olacaktır:

Ortalama Personel Değeri	Mezuniyet
14000	Bilgisayar Müh.
12000	İşletme

Yukarıdaki son haliyle, firmadaki iki farklı mezuniyet alanından gelen kişilerin ortalama personel değerlerini (veya maliyetlerini) çıkarmış olduk. Bu değerler yorumlandıktan ve değerlendirildikten sonra bundan sonraki işlemlerde kullanılabilir. Örneğin firma her aldığı yeni personelin mezuniyetine bakarak, firma için maliyetini hesaplayabilir.

2.6. Sorular

1. Bulunduğunuz kurumda veya örnek olarak üzerinde çalışabileceğiniz bir kurumda veri tabanı bulunup bulunmadığını, bulunuyorsa tutulan verilerin neler olduğunu, veri tabanı bulunmuyorsa verilerin nasıl tutulduğunu sorgulayınız.
2. Kurumdaki verilerden alınan ve bilgiye dönüşen işlemleri bulunuz. Mevcut işlemler dışında ilave olarak çıkarılabilecek bilgileri listeleyiniz.
3. Çıkarmış olduğunuz bilgilerin listesini önem sırasına sokunuz ve kurum için neden önemli olduklarını belirleyiniz.

3. WEKA

Bu bölümde, bundan sonraki bölümlerde kullanılacak olan ve veri madenciliği metot ve algoritmalarının çoğunu barındıran WEKA programına genel bir giriş yapılması hedeflenmektedir.

3. Weka

Weka esas itibariyle, bir makine öğrenmesi (machine learning) aracı olarak, Yeni Zelanda'da bulunan University of Waikato'daki bir grup araştırmacı tarafından geliştirilmiştir.

Günümüzde, üzerinde yapılan geliştirmeler devam etmektedir. Ayrıca çok sayıdaki veri madenciliği, makine öğrenmesi ve iş zekası uzmanı tarafından da bir araç olarak kullanılmaktadır.

Weka'nın bir diğer avantajı da JAVA dilinde geliştirilmiş olmasıdır. Bu sayede yazılım projelerine kolaylıkla entegre edilebilmekte ve JAVA dilinde yazılan herhangi bir projeye kolayca dahil edilebilmektedir.

Weka her ne kadar makine öğrenmesi amacıyla geliştirilmiş olsa da, bazı veri madenciliği işlemlerini de içermektedir. Örneğin veri işleme, kümeleme (clustering), sınıflandırma (classification), ilkelleme (regression), veri ön işleme (data preprocessing), özellik

seçimi (feature selection) gibi işlemler WEKA tarafından desteklenmektedir.

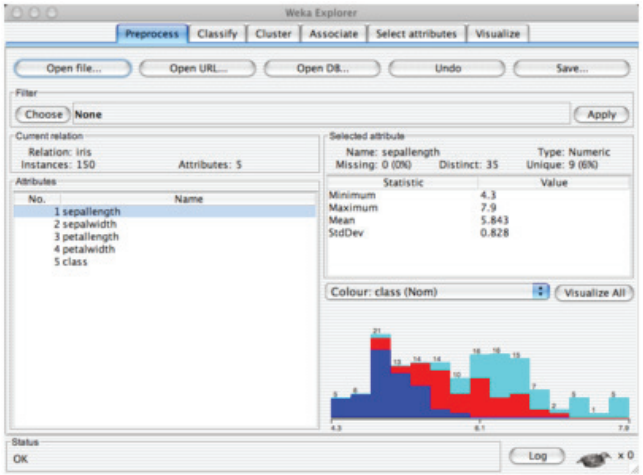
Bu bölümde, Weka'nın genel olarak arayüzleri ve ilerleyen bölümlerde sıkça kullanacağımız, bilgi akışı (knowledge flow) arayüzü üzerinden bir örnek açıklanmaya çalışılacaktır. Ayrıca Weka'nın kurulumu için EK 1'de yer alan kurulum kılavuzundan faydalanılabilir.

3.1. Weka'nın Ekranları

Aşağıda bu amaçla yazılan ekranlardan bazıları gösterilecektir. Bu ekranları, okuyucu hızlı bir şekilde deneyerek de dolaşabilir. Bir sonraki bölümde Weka üzerinde basit bir uygulama yapılacaktır, okuyucu isterse bu bölüme atlayabilir. Weka ilk çalıştırıldığında, aşağıdaki şekilde bir ekran belirecektir. Bu ekrandaki en üstte bulunan "Explorer" düğmesine tıklayarak aşağıda açıklamaları bulunan ekranlara erişim sağlanabilir:

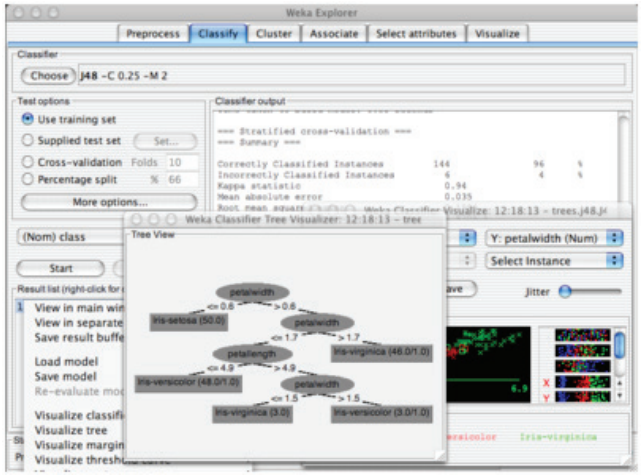
Ön işleme paneli (Preprocess Panel)

Bilgi dolaşıcısının (knowledge explorer) başlangıç noktası önişleme panelidir. Bu panelde veri kümeleri (dataset) yüklenebilir veya WEKA içerisinde bulunan filtreler ile veriler üzerinde işlemler yapılarak veri işlenebilir.



Sınıflandırıcı Paneli (Classifier Panel

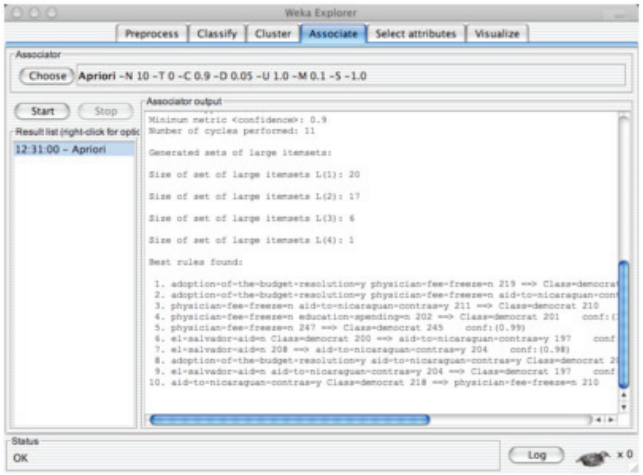
WEKA içerisinde yüklü olan sınıflandırma algoritmalarından herhangi birisini kullanarak mevcut veri kümesi üzerinde bu ekran marifetiyle sınıflandırma yapılabilir. Ayrıca bu ekranda test ve sağlama (validation) için ayrı kümeler kullanmak da mümkündür. Sınıflandırma hataları ayrı bir ekranda açılır ve şayet sınıflandırma algoritması bir karar ağacı (decision tree) oluşturursa bu da ayrıca bir ekranda görüntülenir.



Sınıflamaya benzer şekilde kümeleme (clustering) için kullanılan ekrandır ve benzer şekilde görselleştirme arayüzü bulunmaktadır.

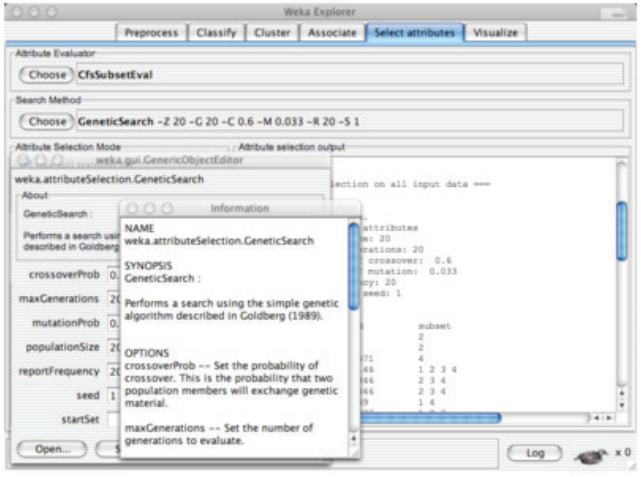
Sınıflandırıcı Paneli (Classifier Panel)

Weka içerisinde tanımlı birleştirme algoritmaları kullanılarak, mevcut veri kümesi üzerinde veri madenciliği (data mining) işlemi yapılmasını sağlar.



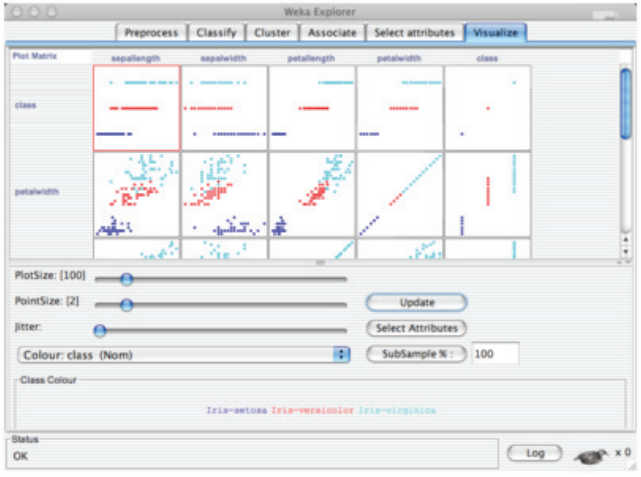
Seçim Özellikleri Paneli (Select Attributes Panel)

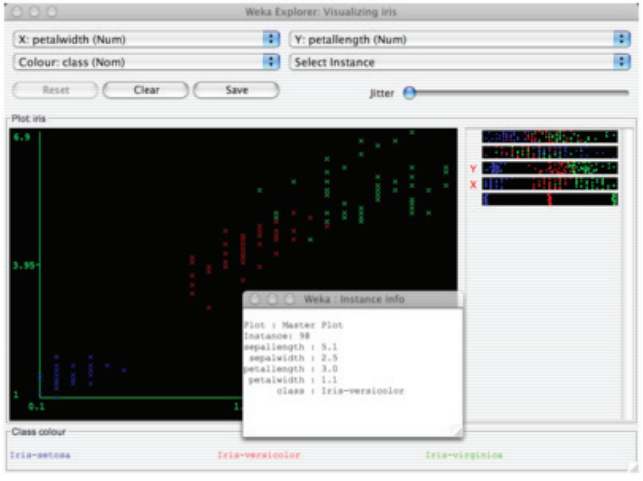
Veri kümesi üzerinde yapılan seçme ve işleme özelliklerini ayarlamaya yarar. Şayet seçme şemalarından birisi veriyi dönüştürüyorsa, dönüşmüş veri görselleştirme ekranında görülebilir.



Görselleme Paneli (Visualize Panel)

Bu panelde veri kümesi üzerinden bir çizim gösterilebilmektedir. Hücrelerin ve noktaların boyutları ekranın alt tarafındaki panelden ayarlanabilir. Seçim özellikleri ekranından, matris üzerindeki hücre sayısı değiştirilebilir. Ayrıca çok büyük veri kümeleri ile çalışılırken, işlem kolaylığı olması açısından sadece alt örneklem uzayının kullanılması da mümkündür.

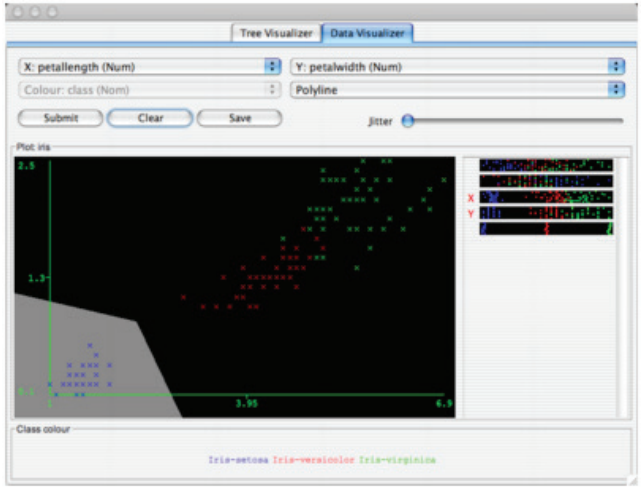


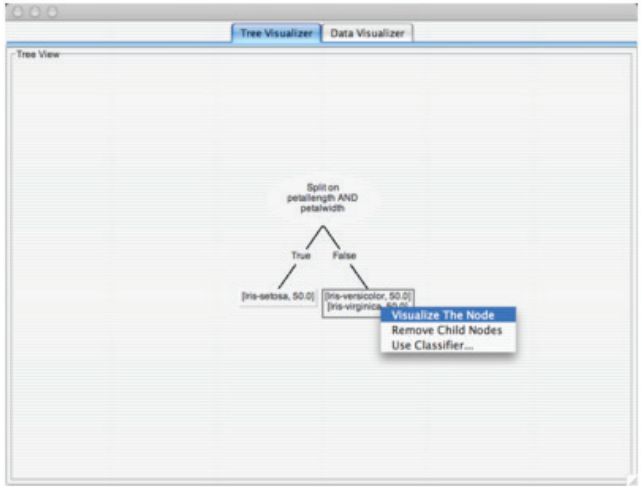


Etkileşimli Karar Ağacı İnşası
(Interactive decision tree construction)

WEKA içerisindeki bu araç ile çift alternatifli (bivariate) bölünmeleri ve bu bölünmeler üzerinde bir

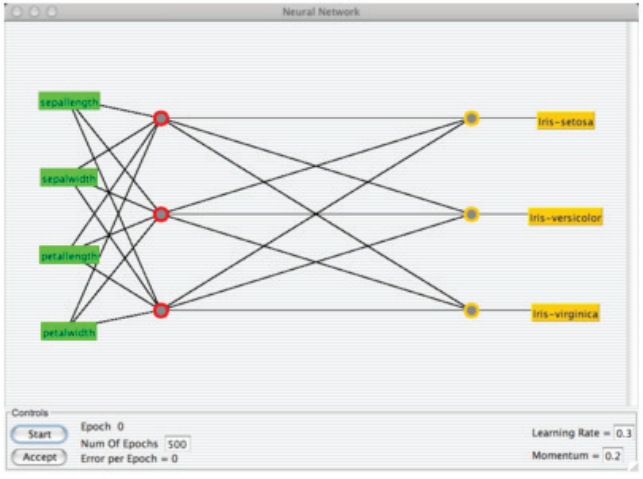
ağaç yapısını etkileşimli olarak inşa etmek mümkündür. Ayrıca inşa edilen bu ağacın yeniden değerlendirilmesi veya değiştirilmesi de mümkündür.





Yapay Sinir Ağı (Neural Network GUI)

WEKA içerisinde bulunan yapay sinir ağı (neural network) arayüzüdür. Bu arayüz marifetiyle **çok seviyeli perseptron (multi layer perceptron)** ve eğitimi kontrol eden parametrelerin girilmesi mümkündür.



3.2. Weka Bilgi Akış Ekranı (Knowledge Flow)

Knowledge flow ekranının amacı, Weka içerisinde bulunan çok sayıdaki kütüphaneye görsel bir ortamdan erişebilmektir. Bu bölümde, kitabın geri kalanından bağımsız olarak tek başına kullanılabilmesi için, bilgi akış ekranında baştan sona bir tasarım yapılacaktır. Ancak bu sırada kullanılan özellikler daha sonra kitabın geri kalan kısmında açıklanacaktır.

Amaç: Bu örnekte, Weka'nın kurulumu ile birlikte gelen, hazır veri kümelerinden (data set) cpu.arff dosya-

sını kullanarak sınıflandırma (classification) yapmaktır.

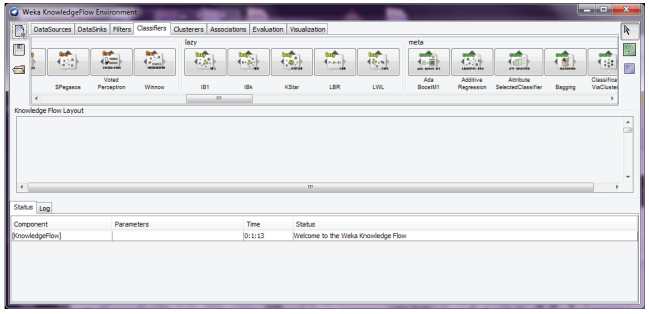
Bu yazı aşağıdaki adımlardan oluşmaktadır:

1. Adım: Weka'nın çalıştırılması ve knowldge flow ekranına erişim
2. Adım: ArffLoader bileşeni ile bir arff dosyasının yüklenmesi
3. Adım: Yüklenen veri kümesi üzerinden hedef sınıfın belirlenmesi
4. Adım: Veri kümesinin test ve eğitim (test and training) kümelerine ayrıştırılması
5. Adım: Sınıflandırıcının eklenmesi
6. Adım: Sınıflandırıcının sonuçlarının ve hata miktarının değerlendirilmesi
7. Adım: Sonuçların gösterilmesi.

3.2.1. 1. Adım: Weka'nın Çalıştırılması ve Knowledge Flow Ekranına Erişim

Weka ilk defa çalıştırıldığında, aşağıdakine benzer bir ekran açılacaktır.

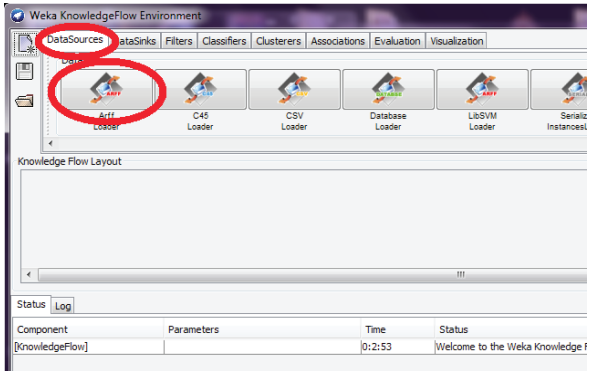
Yukarıdaki görüntüde de gösterildiği üzere knowledge flow düğmesine tıklayarak ekranımızı açıyoruz. Aşağıdaki gibi bir ekran belirecektir:



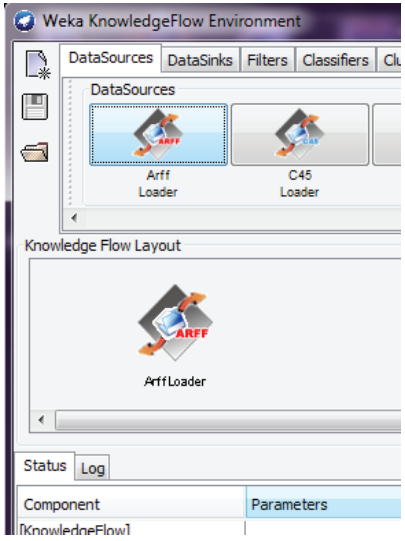
Bu ekranda, ekranın üst kısmında bulunan araçları kullanarak, veri akış düzeneği hazırlayacağız ve bu düzenek üzerinden bir sınıflandırma çalıştıracağız.

3.2.2. 2. Adım: ArffLoader Bileşeni İle Bir Arff Dosyasının Yüklenmesi

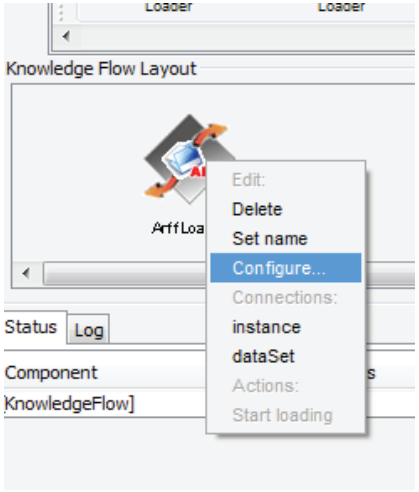
İlk adım olarak veri dosyamızı, ortama yüklemek için gerekli olan aracımızı ekrana ekleyelim. Ekleyeceğimiz araç, en üstte bulunan gruplardan, DataSources grubu altındaki "Arff Loader" aracıdır.



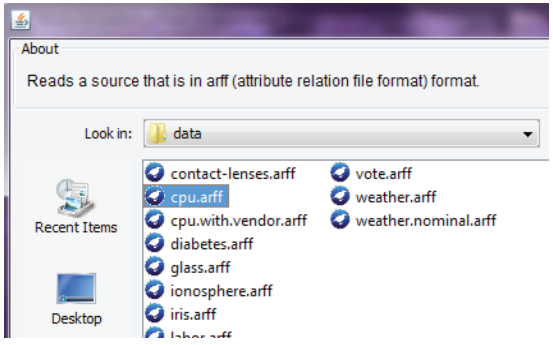
Ekranımıza eklemek için basitçe bu aracın üzerine tıkladıktan sonra fare imlecimiz büyük bir artıya dönüşecektir. Ardından ekran içerisinde bir yere tıklanınca, ArffLoader aracının logosunun bir kopyası çalışma ekranına eklenecektir.



İkinci adımda ArffLoader aracının istediğimiz veri dosyasını yüklemesi için ister çift tıklayarak ister de sağ tıkladığımızda açılan menüden, "configure" seçeneğini seçerek dosya yükleme diyaloğunu açalım:



Açılan diyalog, genelde Weka'nın kurulum dizinini doğrudan işaret edecektir ve bu dizinde, kurulum ile birlikte gelen data klasörü altında cpu.arff dosyası bulunacaktır.



Dosyayı işaretleyip onaylayalım.

İkinci adımda, yüklediğimiz dosyanın bir sınıf seçicisi tarafından işlenmesi gerekmektedir. Buradaki amaç, dosya içerisindeki çok sayıdaki kolondan, hangisinin sınıflandırmada kullanılacak olan hedef sınıf olduğunu belirtmektir. Bunu daha iyi anlatabilmek için arff dosyasının içeriğini göstermek istiyorum:

```
1  %
2  % As used by Kilpatrick, D. & Cameron-Jones, M. (1998). Numeric prediction
3  % using instance-based learning with encoding length selection. In Progres
4  % in Connectionist-Based Information Systems. Singapore: Springer-Verlag.
5  %
6  % Deleted "vendor" attribute to make data consistent with with what we
7  % used in the data mining book.
8  %
9  @relation 'cpu'
10 @attribute MYCT real
11 @attribute MMIN real
12 @attribute MMAX real
13 @attribute CACH real
14 @attribute CHMIN real
15 @attribute CHMAX real
16 @attribute class real
17 % www.bilgisayarkavramlari.com
18 @data
19 125,256,6000,256,16,128,198
20 29,8000,32000,32,8,32,269
21 29,8000,32000,32,8,32,220
```

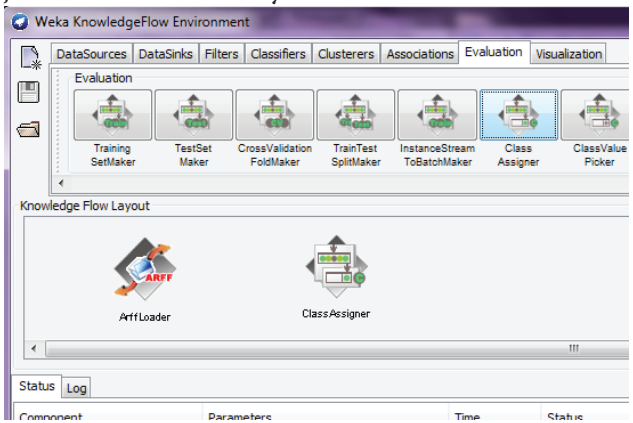
Dosyanın 9. satırından itibaren dosya içeriği başlamaktadır. Bu dosyanın 9. satırında bulunan relation komutu ile kurulmak istenen ilişkiye bir isim verilmiştir. Ardından gelen attribute satırları her kolonun ismidir. Sırasıyla MYCT, MMIN, MMAX şeklinde gitmektedir. Dikkat edileceği üzere 7 kolon ismi bulunmaktadır. Ardından 18. satırda başlayan data komutu artık bundan sonraki her satırın bir kayıt olduğunu (instance) belirtmektedir. Gerçekten de 19. satırdan itibaren her satırda yeni bir kayıt olup her kayıtta da 7 kolon bu-

lunmaktadır. Kolonlar arff dosya yapısında virgül ile ayrılmaktadır. Örneğin veri kümemizin (data set) 3. kaydının CHMIN değeri 8 olarak görülmektedir (21. satırın 5. kolonu. Hazır bahsediyorken, arff dosyasının kolon komutlarını da açıklamak istiyorum "@attribute MYCT real" şeklinde geçen satırın anlamı, bu kolondaki verilerin birer reel sayı olduğudur. Veri kümesindeki değerlere göre farklı tipler verilebilir.)

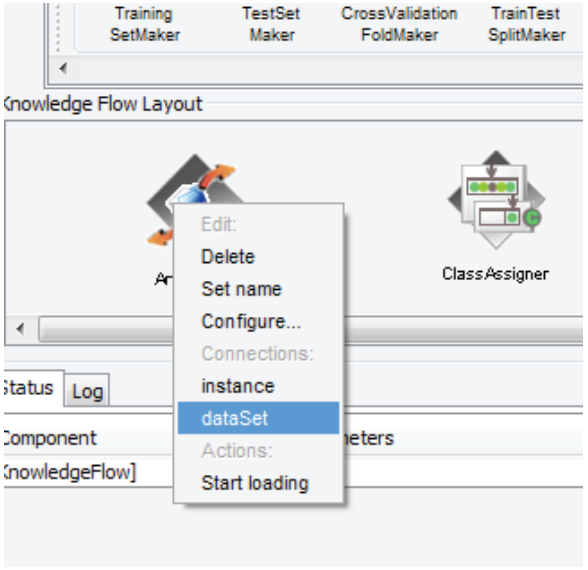
Bu ufak bilgilendirmeden sonra konumuza dönelim, ClassAssigner ile amaçlanan bu 7 sınıftan hangisini hedef sınıf seçtiğimizdir, yani 'Diğer 6 sınıftaki verilerle 7. **sınıftaki verileri doğru tahmin (prediction) edebilecek miyiz**, (ilişkiyi (correlation)) doğru kurabilecek miyiz?' **ş**eklinde soruyu tefsir etmek mümkündür.

3.2.3. 3. Adım: Yüklenen Veri Kümesi Üzerinden Hedef Sınıfın Belirlenmesi

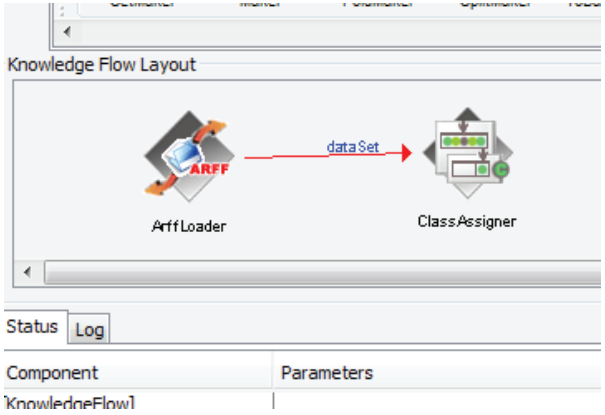
Evaluation grubu altındaki Class Assigner bileşenini seçerek ekranımıza ekliyoruz:



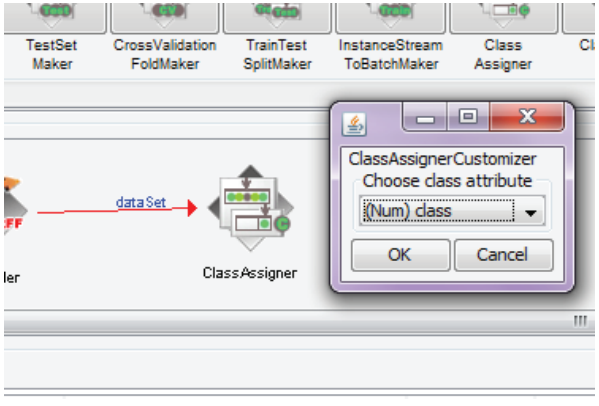
Sırada, eklediğimiz bu iki bileşeni birbirine bağlamak var. Bunun için, Weka'da her zaman, verinin kaynağından hedefine doğru gitmemiz gerekiyor. ArffLoader üzerinde sağ tıklayarak açılan menüdeki connections seçeneği altındaki dataSet seçeneğini seçeceğiz. Burada connections pasif görülecektir, aslında başlık olduğu için pasif görülüyor, **kısaca**, siz açılan menüden dataSet seçeneğini seçebilirsiniz.



Seçimin ardından, şimdiye kadar eklediğimiz bileşenlerin 4 köşesinde birer mavi kutucuk belirecektir. Bu bağlama haline geçtiğimizi ifade ediyor ve bağlayacağımız bileşen olan ClassAssigner üzerine fare ile giderek tıklıyoruz.



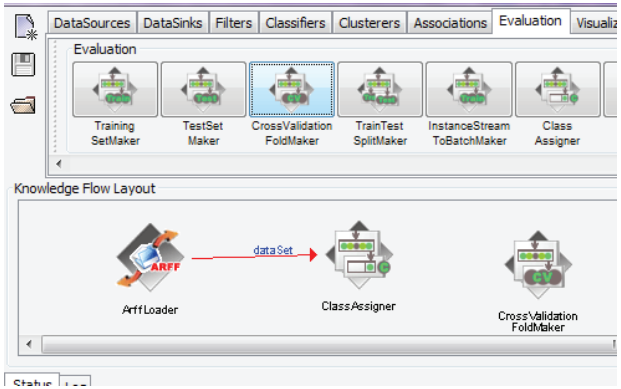
Aşağıdaki şekilde iki bileşen birbirine bağlanmış oluyor. Artık yüklenen arff dosyası önce ClassAssigner bileşenine gelecek ve burada işlenecek demiş oluyoruz. Şimdi ClassAssigner bileşenimizi ayarlayabiliriz. ClassAssigner bileşenine çift tıklayarak veya sağ tuşla tıkladığımızda açılan menüden configure seçeneğini seçerek aşağıdaki ayar ekranını açıyoruz:



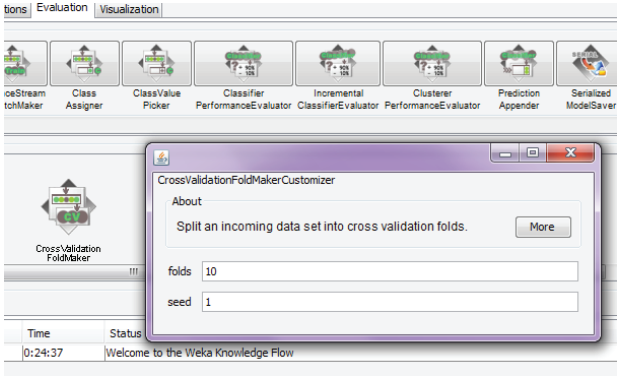
Bu ekranda, basitçe bizim veri kümemizde bulunan kolon isimleri görülecektir. Bunlardan hedefimiz olan kolonu seçiyoruz ki zaten seçili olarak gelen class bizim hedefimiz olacaktır. OK diyerek diyaloğu onaylıyoruz.

3.2.4. 4. Adım: Veri Kümesinin Test Ve Eğitim (Test And Training) Kümelerine Ayrıştırılması

Sırada CrossValidationFoldMaker bileşenini yerleştirmek var. Yine evaluation grubu altında bulunan bileşeni, ekranımızdaki boş bir alana yerleştiriyoruz:

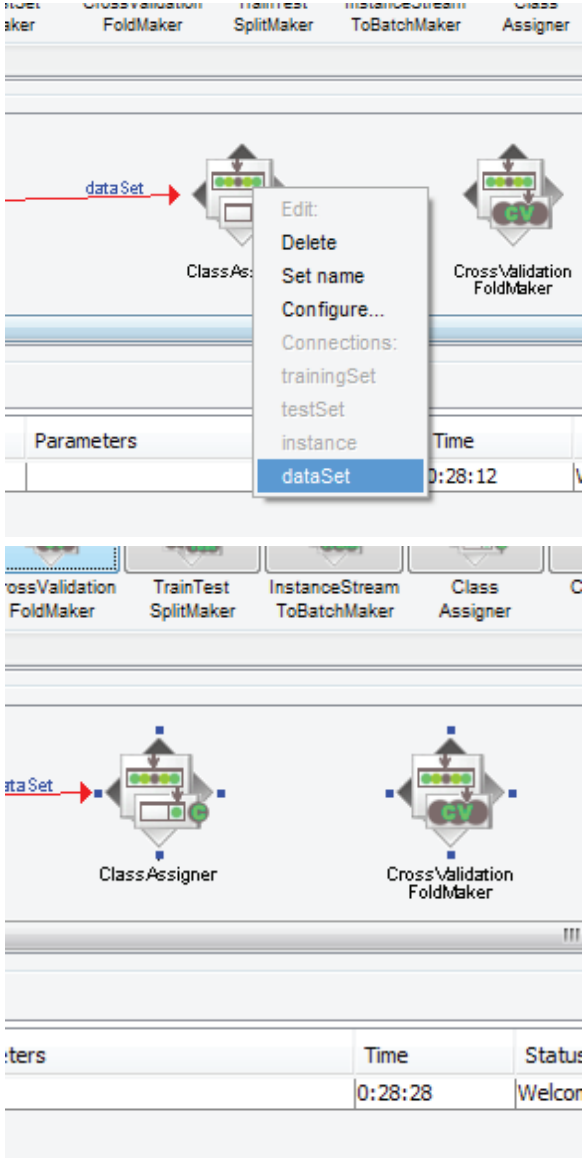


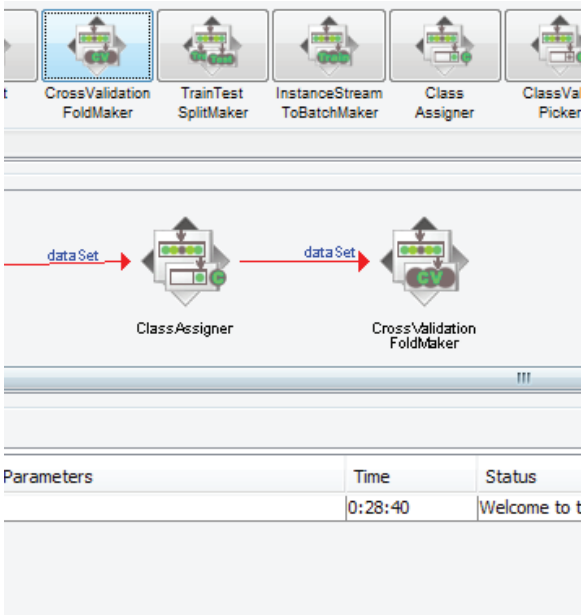
Bu bileşenin amacı, test ve eğitim kümeleri (test and training set) oluştururken kullanacağımız yöntemi "cross validation fold" olarak ayarlamaktır. Çift tıkladığımızda veya sağ tuşla tıkladığımızda açılan ekrandan configure seçeneğini seçtiğimizde açılan ayar ekranına bakacak olursak:



Kendi ayarı 10 ve 1 olarak ayarlı gelecektir. Bunun anlamı, veri kümemizi 10 eşit parçaya bölüp her seferinde 9 parçayı eğitim ve 1 parçayı test amacıyla kullanacak olmasıdır. Bu işlem 10 kere tekrarlanacak ve her seferinde test için ayrılan parça bir kaydırılacak, neticede de her parça 1 kere test için kullanılmış ve 10 kere eğitim için kullanılmış olacaktır. Sonuçta 10 çalışmadan çıkan neticelerin ortalaması alınacak ve başarı buna göre ölçülecektir. (Bu yazı Weka'da araçların kullanımı için hazırlandığından bir paragraf kısa açıklama ile iktifa ediyor ve konuya devam ediyorum).

Açılan diyaloğu gördükten sonra kapatıp, ClassAssigner bileşeni ile bağlayabiliriz. Yine bağlantıya kaynaktan başlayacak ve ClassAssigner üzerine sağ tuşla tıkladıktan sonra açılan pencereden dataSet seçeneğini seçecek (connections altında) ve bileşenlerimizin köşelerinde mavi kutular çıktıktan sonra CrossValidationFoldMaker bileşenine tıklayacağız:

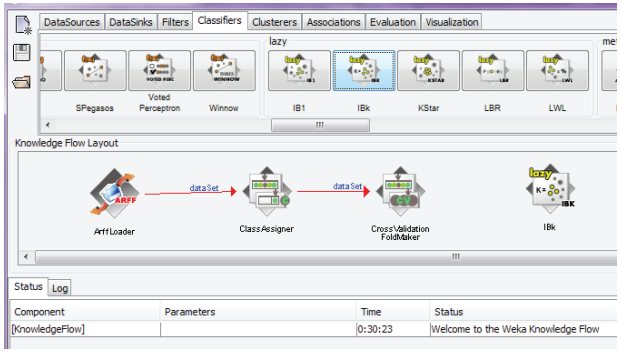




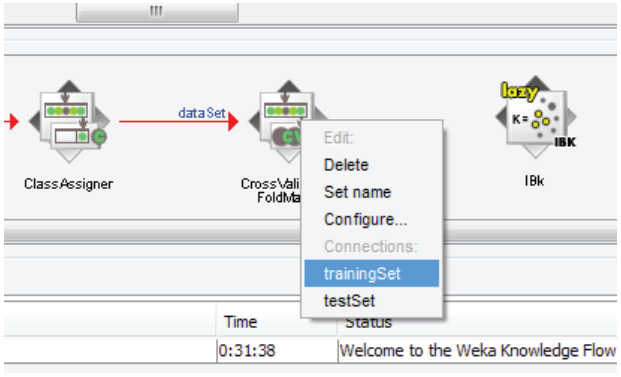
Neticede iki bileşen birbirine bağlanmış olacak ve aralarındaki ilişki dataSet olarak atanmış olacaktır.

3.2.5. 5. Adım: Sınıflandırıcının Eklenmesi

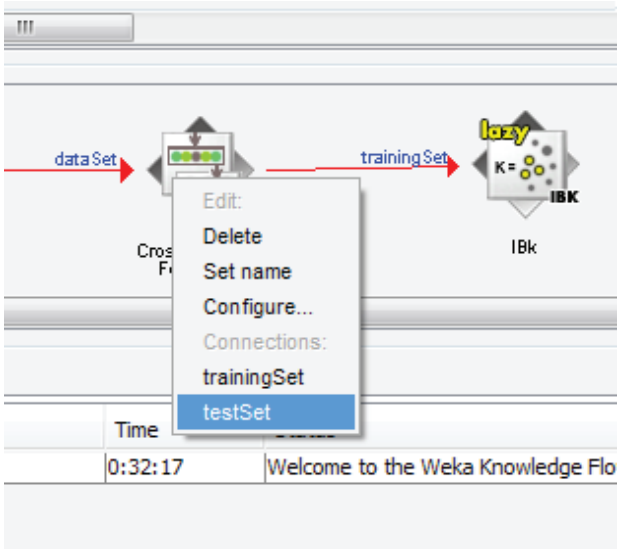
Sırada sınıflandırıcımızı (classifier) seçmek var. Bu örnek kapsamında K-NN (k nearest neighborhood, en yakın k komşu) algoritmasını kullanmak istiyor olalım. Basitçe Classifiers grubu altında bulunan IBK bileşenini ekranımızda boş bir alana yerleştiriyoruz.



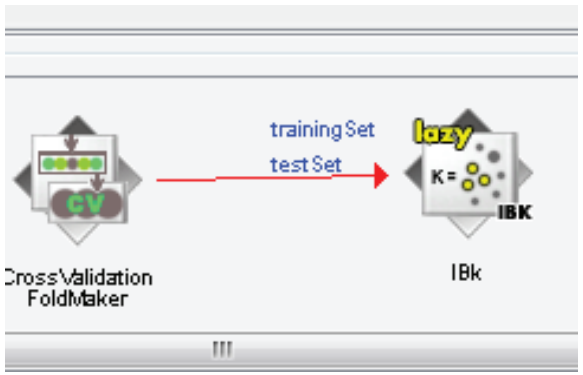
Ufak bir not. IB1 **sınıflandırıcısı K=1 için çalışan knn algoritmasıdır** ve IBk ise k değerinin atanabildiği sınıflandırıcıdır. Hemen bağlantı için CrossValidation-FoldMaker bileşenine sağ tuşla tıklayıp açılan menüden önce traininSet seçip Ibk ile bağlayalım:



Ardından aynı işlemi test için tekrar edelim:

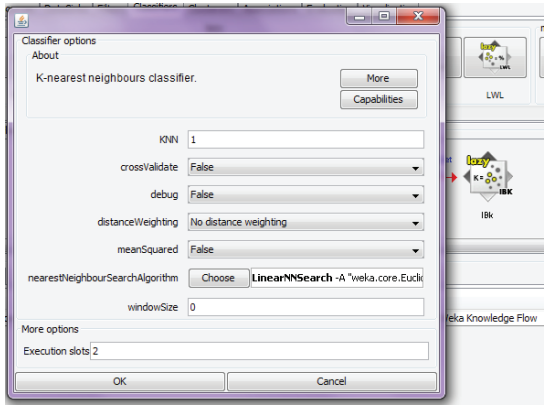


Neticede iki bağlantı da (testSet ve trainingSet) iki bileşen arasındaki bağlantıda yerini almış olmalıdır.



Bunun anlamı, ilgili sınıflandırıcımız olan IBk sınıflandırıcımızı, CrossValidationFoldMaker bileşeninin hem eğitim hem de test amacıyla besleyecek olmasıdır. Yani crossValidationFoldMaker bileşeni ürettiği hem test hem de train kümelerini IBk sınıflandırıcısına yollayacaktır.

Şimdi IBk bileşenine çift tıklayarak veya sağ tuşla tıklanınca açılan menüden configure seçeneğini seçtiğimizde açılan ekrandan ayarlarını yapalım:

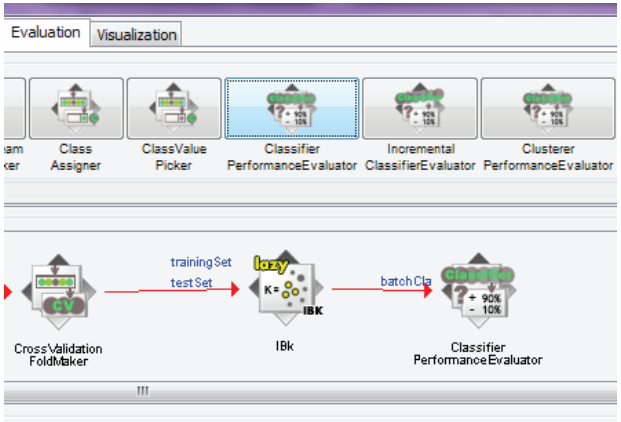


İlk açılışta gelen ayarlardan KNN değerini 3 yapalım. Bunun anlamı en yakın 3 komşuyu dikkate alacağıdır. Diğer ayarlarının detayına, konumuz dışında olduğu için, girmeden yazı kapsamında atlıyorum. KNN değerini 3 yaptıktan sonra OK düğmesine basarak diyaloğu onaylıyoruz.

Buraya kadar olan aşamalarda verimizi yükledik, hedef sınıfımızı belirledik, veri kümemizi eğitim ve test verileri olarak böldük ve sınıflandırıcımıza vererek sınıflandırma işlemini yaptık.

3.2.6. 6. Adım: Sınıflandırıcının Sonuçlarının Ve Hata Miktarının Değerlendirilmesi

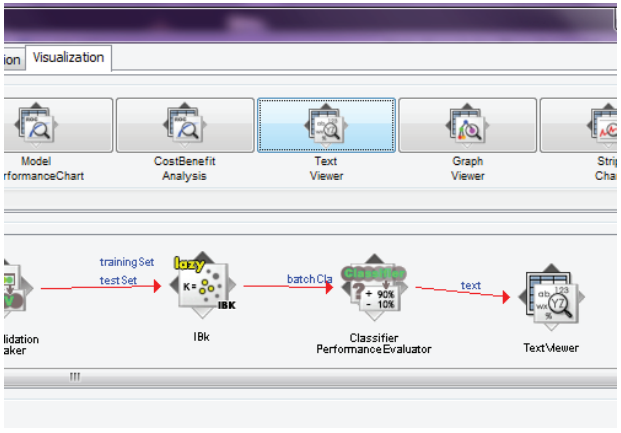
Şimdi sırada sınıflandırıcımızın çıkarmış olduğu verileri değerlendirmek var. Evet sınıflandırıcımız çalışacak ama ne kadar başarılı bir sonuç çıkaracak? Bunu değerlendirmek için classifierPerformanceEvaluator bileşenini, Evaluation grubundan seçerek ekranımıza ekliyoruz ve batchClassifierg seçeneği ile sınıflandırıcımız olan IBk bileşenini, ClassifierPerformanceEvaluator bileşenine ekliyoruz.



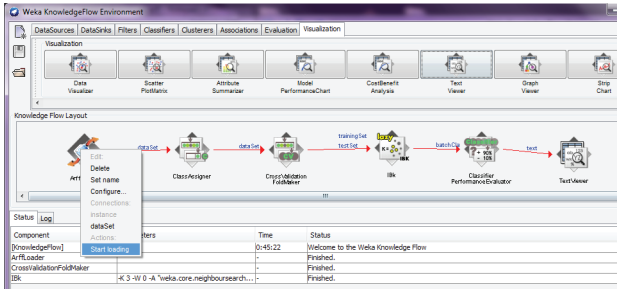
Artık sınıflandırıcımızdan çıkan sonuçları da değerlendirebilecek durumdayız. Geriye bu sonuçları görüntülemek kalıyor.

3.2.7 7. Adım: Sonuçların Gösterilmesi

Weka bize yine çeşitli alternatifler sunuyor. Biz basitçe Visualization grubu altında bulunan TextViewer seçeneğini ekranımıza yerleştirerek, sınıflandırıcımızın performans değerlendirmesi çıktısını bu yeni bileşene text seçeneği ile bağlıyoruz:



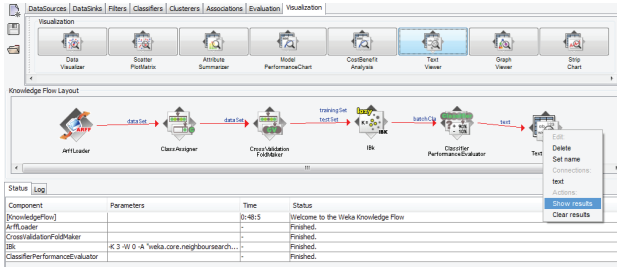
Artık düzeneğimiz (system) hazır. Yapacağımız son şey, düzeneğimizi çalıştırarak sonuçları görmek. Bunun için düzeneğin en başındaki arffLoader üzerinde sağ tuşla tıklayarak, açılan menüden startLoading seçeneğini seçmemiz yeterlidir.



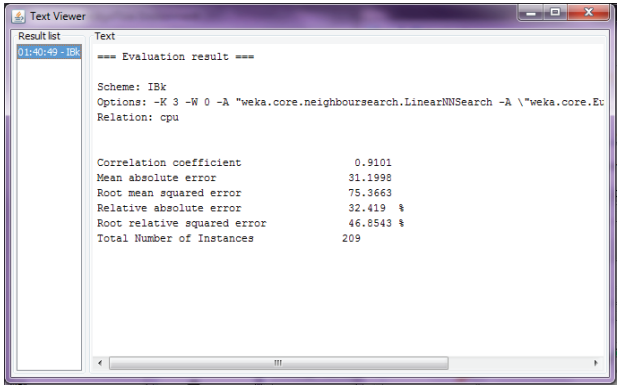
Seçimin ardından, ekranın en altında bulunan status ekranında çalışma ile ilgili bilgiler belirecektir. Ne kadar sürdüğü (mesela yukarıdaki ekran görüntüsünde 45 dakikadır bu yazıyı yazıyor olduğum için ekranın 45.22 dakikadır açık

olduğunu gösteriyor diğer adımlar gösterilmeyecek kadar kısa sürdüğünden onlarda bir zaman bilgisi belirmiyor) geçilen adımlar ve neticeleri gösterilmektedir.

Sonuç olarak sınıflandırıcımızın başarısını görmek için son adımda eklediğimiz TextViewer üzerinde sağ tuşa tıklıyor ve showresults diyoruz:

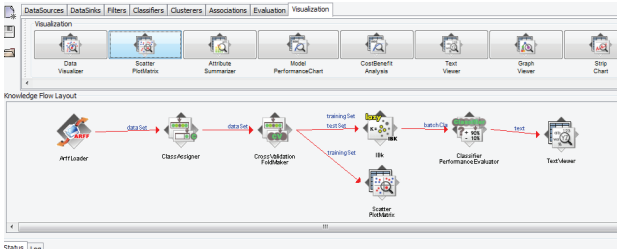


Yeni bir pencere açılıyor ve bu pencerede, sınıflandırıcımızın başarısını gösteren neticeler beliriyor:

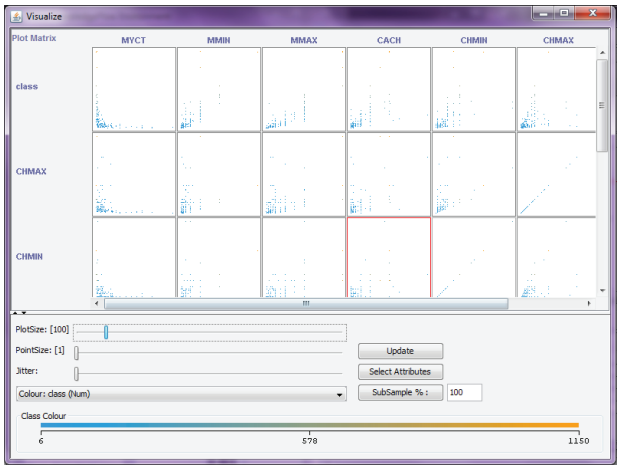


Burada, Weka'nın kullandığı hata ölçüm değerlerine göre sınıflandırıcımızın başarısı gösterilmekte.

Örneğimiz burada bitiyor ve basitçe Weka'nın bir sınıflandırma işlemini kullanacak kadar bilgiyi bu yazı kapsamında verdiğimi umuyorum. Ancak ufak bir not daha eklemek istiyorum. Çok sık kullanıldığı için scatterPlotMatrix bileşenini kullanarak eğitim verisini görsel olarak göstermek için aşağıdaki eklentiyi yaptım:



Yine ArffLoader bileşenine sağ tuşla tıklayarak startLoading dedikten sonra ScatterPlotMatrix üzerinde sağ tuşla tıklayıp showplot dersek aşağıdakine benzer şekilde eğitim kümemizdeki verilerin görsel değerlerini görebiliriz.



Ayrıca yine bir dipnot olarak, DataSink grubu altındaki bileşenler, çıkan sonuçları veri olarak, dosya, arff dosyası, veri tabanı gibi ortamlara yazmak için kullanılmaktadır. Yazıyı daha karmaşık ve okunması zor hale getirmemek için yazının içerisinde bileşen detaylarına veya çok sayıdaki bileşenlerin her birine değinemiyorum ancak ilgilendiğiniz bir bileşen olursa, yazının altındaki yorum kısmını kullanarak sorabilirsiniz, gelen sorulara cevap vermeye çalışacağım.

3.3. Sorular

1. Weka'nın kurulumu ile gelen iris veri kümesi, çiçek yaprakları üzerinde yapılan bir çalışmayı içermektedir.
2. Kurumdaki verilerden alınan ve bilgiye dönüşen işlemleri bulunuz. Mevcut işlemler dışında ilave olarak çıkarılabilecek bilgileri listeleyiniz.
3. Çıkarmış olduğunuz bilgilerin listesini önem sırasına sokunuz ve kurum için neden önemli olduklarını belirleyiniz.

4. Veri Ön İşleme

(Data Pre-Processing)

Bu bölümde, VTBK (KDD) akışının ikinci aşaması olan veri ön işleme aşamasının nasıl yapılabileceği açıklanacaktır.

4. Veri Ön İşleme (Data Pre-Processing)

Bu bölümün amacı, iş zekasının temelini oluşturan VTBK (veri tabanı bilgi keşfi, Knowledge Data Discovery, KDD) aşamalarının ikincisi olan veri ön işleme aşamasını açıklamaktır.

4.1. Veri Ön İşlemesinin Amacı

Veri ön işlemenin temel olarak amacı, gürültülü verilerin (noisy data) düzeltilmesidir. Gürültülü veri aşağıdaki üç şekilde olabilir:

1. Eksik veri: Veri tabanında, çeşitli sebeplerle bazı verilerin eksik olması durumudur. Örneğin çalışanların yaşlarının üzerinden işlem yapılmak istendiği bir sırada, her çalışanın doğum tarihinin veri tabanında yer almaması gibi.
2. Hatalı veri: Veri tabanındaki bazı verilerin tanımı ile

çelişmesi durumudur. Örneğin maaş hiçbir zaman eksi olamaz. Şayet bir çalışanın maaşı veri tabanında eksi değere sahipse bu durumda bu verinin hatalı olduğu söylenebilir.

3. Uyuşma sorunu: Veri tabanındaki birden fazla alandan gelen verilerin birbiri ile uyuşmaması durumudur. Örneğin bir kişinin doğum yılı 2000 olup, yaşı 75 olamaz.

Yukarıdaki bu gürültülü verilerin giderilmesi için çeşitli yöntemler kullanılmaktadır. Bu yöntemlerin tamamı, veri ön işlemesi olarak kabul edilir. Bu bölüm kapsamında bu çözüm yöntemlerine yer verilecektir.

4.2. Veri Neden Kirlenir?

Veri kaynağının kirli olmasının çeşitli sebepleri olabilir. Verinin toplanırken bazı verilerin eksik olması, eksik verinin temel sebebidir. Genelde verinin birden fazla kaynaktan toplanması durumlarında kullanılan veri şablonları arasında uyuşmazlık olması, bazı veri kaynaklarına kısıtlı erişim, verinin taşınması veya işlenmesi sırasında bozulması gibi sebeplerden dolayı eksik veri olabilir.

Verinin toplanması aşamasında karşılaşılan diğer bir durum da kayıt tekrarlarıdır. Aynı kayıt, birden fazla kaynaktan gelmiş ve veri kaynağında tekrar etmiş olabilir. Bu durum da ayrıca ele alınması ve temizlenmesi gereken bir durumdur.

4.3. Veri Ön İşlemesinin Önemi

Veri ön işlemesi, bundan sonraki aşamaların başarısını doğrudan etkileyen bir adımdır. Şayet herhangi bir verinin eksik olarak veya hatalı olarak veri madenciliği aşamasına geçirilmesi söz konusu olursa, bu durumda sonuçlar beklenmeyen şekilde hatalı olabilir.

4.4. Veri Ön İşleme Yöntemleri

Yukarıda sıralanan 3 farklı yöntem için farklı veri ön işlemesi yöntemleri bulunur. Sırasıyla bu durumlar açıklanacaktır. Birincisi verinin eksik olması durumudur ve bu durumda Zan (imputation) yöntemi kullanılır.

4.5. Zan (Imputation) Yöntemi

Veri ön işleme aşamalarından birisi de, eksik verilerle karşılaşılması halinde bir çözüm olarak bu eksik verilerin Zan edilmesi (yerine uygun verilerin üretilmesi, imputation) yöntemidir.

Zan, sözlükte olmayan bir şeyin yüklenmesi anlamındadır. Örneğin olmayan bir suçun birisine yüklenmesine zan altında bırakmak denilebilir. Bu anlamda bir veri kümesi (data set) üzerinde çalışılırken bazı sebeplerden dolayı verilerin eksik olması halinde bu verilerin uygun başka sayılarla tamamlanması sağlanabilir.

Genelde verinin hatalı okunması, veri kaynağında yaşanan bozulma gibi sorunlar veya bazı verilere erişim zorluğu eksik verilere sebep olabilmektedir.

Bu verilerin eksik olması durumu genelde sorunlara

sebep olur. Örneğin çoğu hazır istatistik paketleri (SAS, SPSS, Weka veya r-project gibi) bu tip durumlarda sorunlar yaşamaktadır. Gerçi bu paketlerin ücretli ve gelişmiş olanlarının çoğunda (SAS veya SPSS gibi) Zan modülleri (imputation) bulunmakta ve bu işi otomatik olarak yapabilmektedirler ancak bu yazı kapsamında gerek bu modüllerin nasıl çalıştığını anlamaya, gerekse bu işlemi elle yapmak istediğimizde nasıl müdahale etmemiz gerektiğini açıklamaya çalışacağız.

Zan yöntemleri duruma ve beklentilerimize göre çeşitlilik arz eder ve halen üzerinde çalışılmaktadır. Bunlardan çok bilinen bazılarını saymamız gerekirse:

- Sıcak deste (hot deck)
- Soğuk deste (cold deck)
- Liste boyunca silme (listwise deletion)
- Eşlerin silinmesi (pairwise deletion)
- Ortalama Zan (mean imputation)
- İlkelleme töhmeti (regression imputation)
- Son gözlemin taşınması (last observation carried forward)
- Olasılıksal Zan (stochastic imputation)
- Çoklu Zan (Multiple imputation)

Yukarıdaki bu kavramları kısaca açıklamaya çalışalım. Öncelikle sıcak deste ve soğuk deste (hot deck, cold deck) algoritmaları için ne yazık ki tam bir mutabakat olmadığını ve çeşitli versiyonları bulunduğunu belirtmek gerekir. Bu yüzden bu iki algoritmayı atlayacağım ancak hot deck (sıcak deste) için kısaca, rastgele bir verinin seçilerek eksik olan veri yerine Zan edildiğini söylememiz yeterli olacaktır. Bu rastgele seçim ise oldukça tartışmalı bir konu. Diğer metotları kısaca açıklayacağım.

4.5.1. Liste boyunca silme (listwise deletion)

Örneğin bir veri tabanından okunan tabloda bazı değerlerin eksik olduğunu düşünelim:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—	–	Muhasebe
Veli Demir	43	Yönetim

Yukarıdaki tabloda sondan ikinci ve üçüncü satırlarda kayıp bilgi bulunmaktadır. Bu durumda, örneğin çalışanların yaş ortalamalarının alınması istendiğinde hata ile karşılaşılacaktır.

Liste boyunca silme yaklaşımında (listwise deletion) bu kayıp veri içeren satırların tamamı tablodan temizlenir ve tablo aşağıdaki hale getirilerek işlem sonuçlandırılır.

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Veli Demir	43	Yönetim

Artık bu tablo üzerinde istenilen işlemler yapılabilir.

4.5.2. Eşlerin silinmesi (Pairwise deletion)

Bu yöntemde bütün tablonun temizlenmesi yerine gerekli olan işlem sırasındaki eksik veriler temizlenir. Örneğin bir önceki tabloya geri dönecek olursak:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—	–	Muhasebe
Veli Demir	43	Yönetim

Bu tablo üzerinde çalışanların yaşının ortalaması istenmişti. Bu durumda, bu soruya özel olarak sadece sondan ikinci satırın silinmesi yeterlidir:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
Veli Demir	43	Yönetim

Artık istenen işlem, yani yaşların ortalaması çalıştırılabilir.

Bu yöntemde farklı bir işlem yapılmak istendiğinde, örneğin çalışanların kısımlara göre dağılım grafiği istendiğinde ise işlem yapacağımız tablo aşağıdaki şekilde olacaktır:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
—	–	Muhasebe
Veli Demir	43	Yönetim

Görüldüğü üzere sadece o işlem için problem çıkaran satır silinmiş, diğer satırlar farklı kolonlarında eksik veri bulunmasına rağmen saklanmıştır.

Bu yöntemin menfi yanı, her işlem için farklı veri kümesi üretiliyor olması ve hatta her işlem sonucunda farklı sayıda veri ele alınıyor olmasıdır. İşlemler sonucunda verilerin karşılaştırılma güçlüğü ortaya çıkabilir.

4.5.3. Ortalama Zan (Mean imputation)

Bu yöntemde, bir eksik verinin Zannı sırasında ortalama değer hesaplanır. Örneğin yine aynı tablo üzerinden Zan yaklaşımını izah edelim:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—	–	Muhasebe
Veli Demir	43	Yönetim

Şayet bu tablodaki ortalama yaş için eksik olan satırların Zannı söz konusuysa bu durumda satırların silinmesi yerine veri üretilir. Örneğimizdeki yaşların ortalaması:

$$ortalama\ yaş = \frac{33+23+26+33+43}{5} = 31.6$$

olarak bulunur. Yaşın ondalıklı sayı olması mümkün olmadığı için 32 olarak yuvarlanarak kabul edilebilir ve veri kümemiz aşağıdaki şeklini alır:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—	32	Muhasebe
Veli Demir	43	Yönetim

Bu işlemin ardından artık istenen veri madenciliği yöntemleri uygulanabilir.

Bu yöntemin dezavantajı ise, çok büyük verilerde uygulanma zorluğudur (örneğin güncel problemlerin artık haritalama-indirgeme (map reduce) ortamlarında işlendiği düşünülürse, böyle bir ortamda kullanılamaz). Ayrıca işlenmesi için bütün verinin hafızaya yüklenip hesaplama yapılması zorluğu da bulunmaktadır.

Son olarak veri eksikliğinin çok fazla olduğu durumlarda üretilen verilerin sağlığı problem çıkarmaktadır.

4.5.4. İlkelleme Zannı (Regression Imputation):

Bu yöntemde, mevcut veriler üzerinden fonksiyonel bir ilkelleme yapılır (örneğin doğrusal ilkelleme (linear regression)) ve ardından ilkellenen fonksiyondan üretim yapılarak eksik veri doldurulur.

4.5.5. Son Gözlemin Taşınması (Last Observation Carried Forward):

Bu yöntemde, veri kümesindeki eksik veriler, kendilerinden bir önceki veri kopyalanmak marifetiyle Zan edilirler. Örneğimize geri dönecek olursak:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—	–	Muhasebe
Veli Demir	43	Yönetim

Veri kümesindeki verilerin son gözlemin taşınması yöntemiyle Zan edilmiş hali aşağıdaki şekildedir:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	Bilgisayar
Ahmet Yılmaz	33	Muhasebe
Veli Demir	43	Yönetim

Artık bu veri kümesi üzerinde veri madenciliği yöntemleri uygulanabilir.

Bu yöntemin dezavantajı ise bazı marjinal noktaların çoğaltılması olarak görülebilir. Örneğin yaş sütununda marjinal olarak 70 yaşında bir çalışan varsa bu kişinin kopyalanması, veri madenciliği sonuçlarını menfi etkileyebilir.

Son iki yöntemimiz olan istatistiksel Zan (stochastic imputation) ve çoklu Zan (multiple imputation) için şunları söylememiz yeterli olacaktır.

Öncelikle mevcut veri kümesi üzerinden bir istatistiksel dağılım elde edilir (örneğin ilkelleme (regression) burada kullanılabilir) ardından bu dağılım sayesinde eksik verileri dolduran bir bağlantı kullanılır. Çoklu Zanda ise bu işlem birden fazla kere yapılarak her bir veri kümesi daha sonra kullanılmak üzere saklanır. Ayrıca her veri kümesinin hata miktarı hesaplanır. Ardından bu veri kümelerinin ortalama değerleri alınarak nihai veri kümesi elde edilir. Buradaki tek ayrıntı, standart hatanın (standard error) hesaplanması sırasında iki veri kümesinin birleştirilmesi için iki standart hata miktarının toplanıp kareköklerinin alınmasıdır.

Kısacası çoklu Zan, daha iyi sonuç elde etmek için istatistiksel Zannın birden fazla kere çalıştırılması olarak düşünülebilir.

4.6. Hatalı Verilerin Düzenlenmesi

Veri ön işlemesi sırasında karşılaşılan bir diğer durum da verilerin hatalı olmasıdır. Örneğin bir kişinin yaşının eksi değere sahip olması gibi durumlarda verinin hatalı olduğunu söyleyebiliriz. Hatalı verilerin tespit edilmesi halinde düzeltmek için kullanılan dört temel yöntem bulunmaktadır. Bunlar aşağıdaki şekilde sıralanabilir:

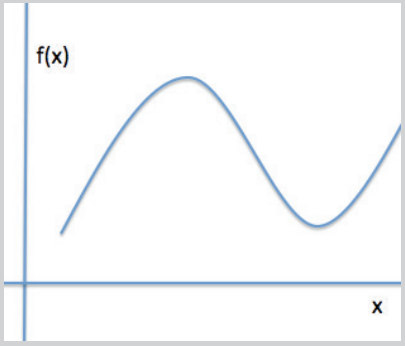
- Kutulama (Binning)
- Kümeleme (Clustering)
- İlkelleme (Regression)
- İnsan müdahalesi (Human Inspection)

4.6.1. Kutulama (Binning)

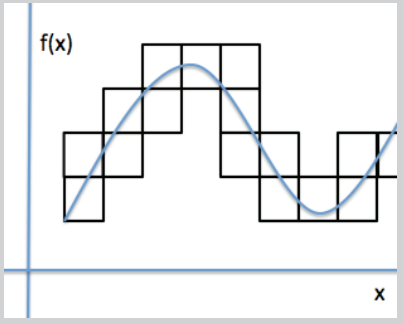
Bu yöntemde veri sıralanır ve ardından sıralanan veri üzerinde eşit boyutlarda gruplar şeklinde kutulama yapılır. Kutulama, çalışma mantığı açısından veri adetleme (data quantization) olarak düşünülebilir.

Veri Adetleme *(Data Quantization)*
Bu yöntem, dijital sinyal işleme sırasında kullanılan ve sürekli (continous) verilerin üzerinde kesikli (discrete) seriler elde etmeye yarayan bir yöntemdir. Bu açıdan, sürekli sinyallerin kesik sinyallere çevrilmesi açısından sıklıkla kullanılan

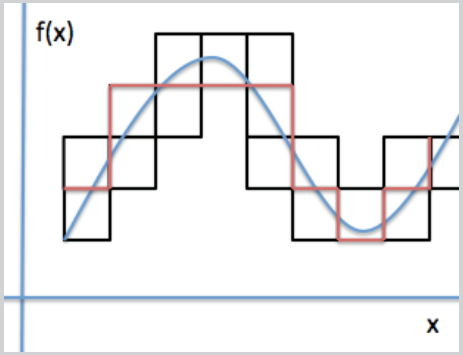
yöntemlerden birisidir. Yöntem basitçe, verinin tanımlı olduğu alanların sayı parçalarına bölünmesi (quant) ve ardından bu parçaların tekdüze olarak işlenmesi mantığı ile çalışır. Örneğin sürekli bir veri olarak aşağıdaki şekilde verilen fonksiyon grafiğini ele alalım:



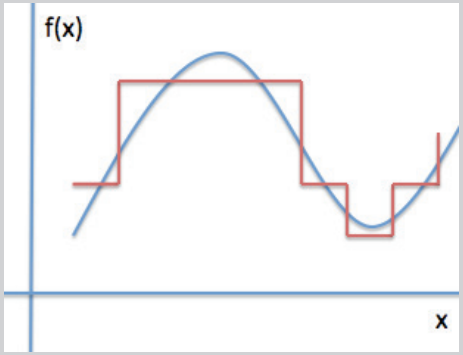
Yukarıdaki fonksiyon grafiğinin eşit boyutlardaki kutulara konulmuş hali aşağıdaki şekildedir:



Yukarıdaki şekilde kutulanan grafik üzerinde, fonksiyonun değerlerinin kutunun alt veya üst bölgesinde yoğun olmasına göre, kutu sınırlarında değerlerinin tanımlı hale getirilmesini istersek, aşağıdaki gibi bir sonuç elde ederiz:

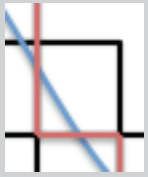


Yeni şekilde çizilen kutuların sınırlarını tek başına ele alacak olursak aşağıdaki gibi bir fonksiyon grafiği elde ederiz:



Görüldüğü üzere, sürekli bir fonksiyon ayrık bir fonksiyona dönüştürülmüştür. Burada kullanılan yöntem, her kutu içerisindeki fonksiyon değerinin sayılması ve ardından sigmoid fonksiyonuna tabi tutulmasıdır. Daha basit bir ifadeyle, kutunun yarısından fazlasının dolu olması halinde kutunun tavan değeri, kutunun yarısının altında olması halinde ise kutunun taban değerinin döndürülmesidir.

Daha net anlaşılması açısından örnek bir sayı parçası (quant) durumunu aşağıdaki şekilde temsil etmeye çalışalım:



Yukarıdaki kutuda görüldüğü üzere, kutunun azınlık kısmı fonksiyonun grafiği altında kalmaktadır. Bu durumda kutunun tavan değeri alınmıştır.

Kutulama işlemi sırasında, sayı parçalarına bölünme işleminde (quantization) olduğu gibi veri kümesindeki değerler kutulara bölünmekte ve her kutu için tek bir değer elde edilmektedir. Yani kutuların içerisindeki değer farklılıkları yok edilmekte ve bu aralıkta bulunan çok sayıdaki veriden tek bir veriye indirgeme yapılmaktadır.

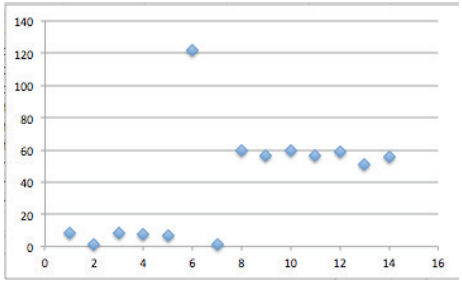
Bu anlamda kutulama işlemini tek yönlü bir fonksiyon (one way function) olarak görmek mümkündür. Ayrıca ileride değineceğimiz bir özellik olan özetleme (feature hashing) fonksiyonu olarak da düşünmek mümkündür.

Kutulama işlemi sayesinde, hatalı olan verilerin giderilmesi mümkün olmaktadır. Örneğin kutulama yapıldıktan sonra, kutu içerisinde kalan hatalı, eksik veya çelişkili veriler için, kutunun içindeki verilerin ortalama değeri atanabilir. Benzer şekilde kutunun ortanca değeri veya kutunun sınırlarındaki değerlerin ortalamasının atanması gibi yöntemler izlenebilir.

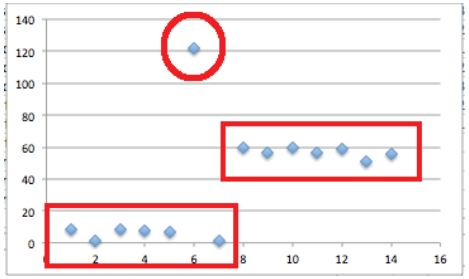
4.6.2. Kümeleme (Clustering)

Bu yöntemde, basitçe bir veri kümesinin elemanları, yakın elemanlardan oluşan kümelere (bölüt, topak, cluster) ayrılarak herhangi bir kümeye dahil olamayan elemanlar elenir. Bu elemanlara literatürde 'dışlanan elemanlar' (outlier) ismi verilir.

Daha iyi anlaşılabilmesi için, örnek bir veri kümesini görselleştirelim:



Yukarıdaki şekilde verilen veri kümesi değerlerini kendi içinde yakın değerler olarak öbekleyecek olursak, aşağıdaki şekilde iki öbek elde etmek mümkün olacaktır.



Şekilde yuvarlak içerisine alınan değer, herhangi bir öbeğe yakın olmadığı için dışlanan değer (outlier) olarak düşünülebilir.

Dışlanan değerlerin bulunması için, değerler arasındaki mesafelerin ölçülmesi ve diğer değerlere uzak mesafede bir değer bulunduğunda dışlanması gerekmektedir.

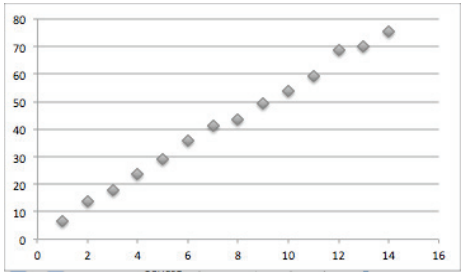
Genelde dışarıda kalan ve herhangi bir kümeye (cluster) dahil olmayan verilerin hatalı veri (noisy data) olduğunu söyleyebiliriz, ancak bu durum her zaman geçerli değildir ve dikkat gerektiren bir eleme işlemidir.

Örneğin bir firmadaki çalışan kişilerin yaşlarını çizdirdiğimiz bir örnekte 120 yaşında bir kişinin bulunması çok büyük ihtimalle bir hataya delalettir.

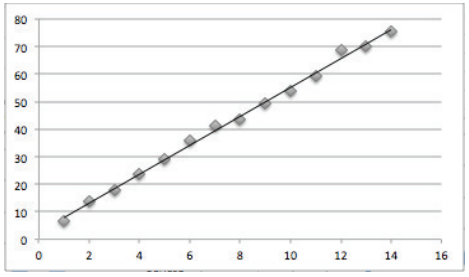
4.6.3. İlkelleme (Regression)

Basitçe bir veri kümesindeki veriler kullanılarak bu veriler üzerinden analitik bir fonksiyon elde etmeyi amaçlar. Doğrusal (linear), çok terimli (polynomial) veya logaritmik ilkelleme (regression) gibi çeşitleri vardır. Veri ön işleme kısmında basit olması açısından doğrusal ilkelleme konusundan bahsedelim.

Bu yazının amacı, doğrusal ilkelleme yöntemini (linear regression) açıklamaktır. Basitçe bir veri kümesinin iki boyutlu bir uzayda dağıldığını düşünelim.



Veri kümemizdeki değerlerin iki boyutlu uzayda, yukarıdaki şekilde gösterildiğini kabul edelim. Şimdi doğrusal ilkelleme ile amaçlanan, bu noktaların tamamına en yakın geçen doğruyu elde etmektir. Örneğin aşağıdaki şekilde olabilir:



Her doğrunun bir formülü olduğu gibi bu doğrunun da karakteristik bir şekilde

y= ax + b denklemine uygun bir formülü, daha doğrusu bir (a,b) ikilisi bulunacaktır.

İşte doğrusal ilkelleme, çok sayıdaki karmaşık hesaplama ve ölçümlere dayalı verinin basit bir doğruya indirgenmesi (ilkellenmesi) olarak düşünülebilir. Buradaki ama, veri kümesindeki verilerin değerlerinden böyle bir doğru denklemini elde edebilmektir.

Ardından bu doğru çeşitli amaçlar için kullanılabilir. Örneğin herhangi bir x değeri için y değerinin bulunması artık doğru denklemi ile mümkün olacağından, eksik verilerin Zannı (imputation) veya bütün veri kümesinin tutulması yerine basitçe sadece doğrunun tutulması ve işlenmesi veya kesitirim (estimation) ve öngörü (forecasting) problemlerinin çözümünde (örneğin gelecek seneki satış oranları veya deprem tahminleri gibi problemler) ve daha pek çok veri madenciliği probleminde kullanılabilir.

Buraya kadar doğrusal ilkelleme (linear regression) konusuna hızlı bir giriş yaptıktan sonra veri kümesinin nasıl ilkellendiğini açıklayabiliriz.

Aslında doğrusal ilkelleme verilen n adet nokta için, ki bu noktaları {yi,xi} çifti olarak yazabiliriz, i = 0,1,2 … n olmak şartıyla.

y =ax+b şeklindeki karakteristik denklemde en az hata miktarına sahip a, b ikilisini bulmak istiyoruz.

Hata miktarını ise noktaların bu doğruya olan mesafesi olarak düşünürsek formülümüz aşağıdaki şekilde olacaktır.

$$asg\ H(a,b)\ için\ H(a,b)=\sum_{i=1}^{n} h_i^2=\sum_{i=1}^{n}(y_i-a-bx_i)^2$$

Burada H ile gösterilen fonksiyon, a, b ikilisi için bütün noktaların uzaklığının ölçüldüğü hata miktarıdır. Elbette bu bir çift fonksiyon olmalıdır yani eksi hata olmaz. Dolayısıyla kare alınmıştır. Her nokta için ayrı ayrı hatanın hesabı ise, y-a-bx değeri olarak hesaplanabilir.

İşte amacımız bu hata değerini en asgari seviyeye indirmektir.

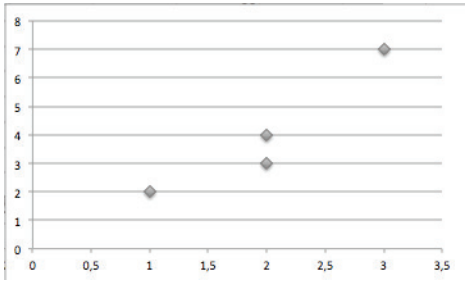
Örnek:

Konuyu bir örnek üzerinden açıklamaya çalışalım.

Örneğin 4 nokta koordinatı aşağıdaki şekilde verilmiş olsun.

(1, 2)
(2, 3)
(3, 7)
(2, 4)

Konuyu daha iyi anlayabilmek için noktaları koordinat sisteminde gösterelim:



Şimdi bütün bu noktalara en yakın geçen doğru denklemini bulmaya çalışalım. Bu işlem sırasında bir arama algoritması üzerinden a ve b ikilisini 0 ile 3 değerleri arasında arttırarak deneyebiliriz.

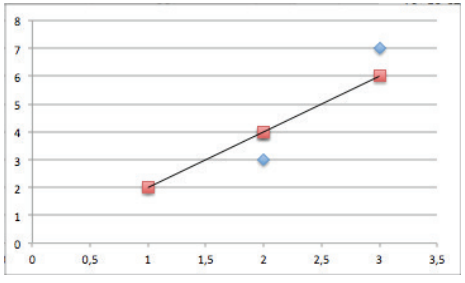
a=0 ve b=0 olarak başlayalım. İlk hata değerimiz için y=2 ve x=1 değerleri ile a ve b değerlerini denklemde yerine yazıyoruz.

$$h^2 = (y_i - a - bx_i)^2 = (2 - 0 - 0\,1)^2 = 4 \rightarrow h = \sqrt[2]{4} \rightarrow h = 2$$

olarak bulunacaktır. Ardından aynı nokta için a=1 , b=0 denemesini yapalım:

$$h^2 = (y_i - a - bx_i)^2 = (2 - 1 - 0\,1)^2 = 1 \rightarrow h = \sqrt[2]{1} \rightarrow h = 1$$

Demek ki a=1 b=0 noktası, daha düşük bir hata vermektedir. Bu şekilde a ve b değerleri sırasıyla arttırılarak verilen aralıktaki bütün a ve b değerleri denenir. Ardından veri kümemizdeki ikinci nokta için bütün a ve b değerleri denenir ve bütün noktalar için bütün a ve b ihtimalleri denendikten sonra, bütün noktalar için en düşük hata değerini veren a ve b ikilisi alınır. Örneğimizdeki hesaplamalar neticesinde a = 1 ve b = 2 değeri en düşük hata değerine sahiptir. Bu durumda eğimin 2 olduğu bir doğru çizilmesi bize sonucu verecektir.



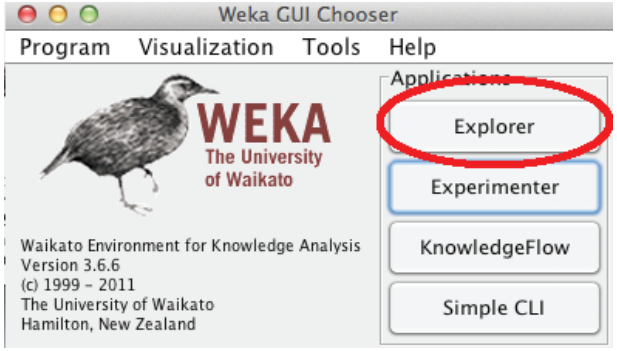
Çok kabaca, yukarıdaki şekilde y=2x doğrusu elde edilmiştir. Demek ki eksik olan bir veri yerine y=2x hesabının neticesinin konulması veya y=5, x=75 gibi bu doğruya çok uzak değerlerin dışlanması (outlier) artık mümkündür.

4.7. Weka ile Veri Ön İşlemesi

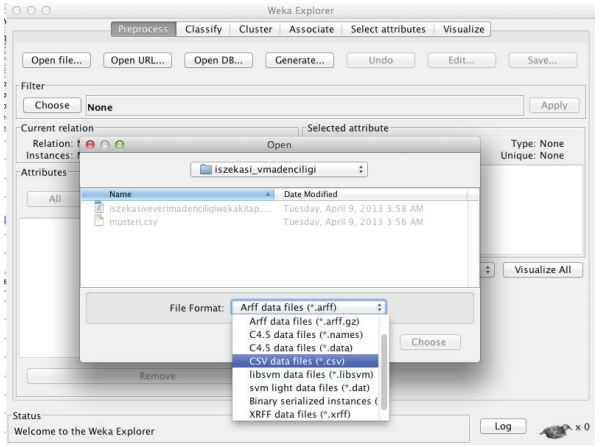
Bu bölümde örnek bir veri kümesi üzerinden Weka'yı kullanarak veri ön işlemesi yapılacaktır.

Öncelikle bu uygulamada kullanacağımız ve kitabın ekinde gelen CD içerisinde bulunan müşteri.csv dosyasını Weka ile açalım. Bu dosya bir bankanın müşterilerinin bilgilerini tutmaktadır.

Veri dosyamızı yüklemek için Weka'yı açtıktan sonra ilk seçenek olan "Explorer" ekranına giriyoruz:



Ardından açılan ekranda "Open File" seçeneğini seçerek dosyamızı açıyoruz.



Açma işlemi sırasında dikkat edilmesi gereken önemli bir nokta, açacağımız dosyanın bir CSV dosyası olmasıdır. Weka'nın kendi dosya yapısı olan arff dosya seçeneği seçili gelecektir. Bunu CSV olarak değiştirdiğimizde müşteri.csv dosyasını seçmemize izin verecektir.

CSV Dosyaları

CSV dosya yapısı, İngilizcedeki "*Comma Seperated Values*" isimlerinin baş harflerinden oluşmaktadır ve virgül ile ayrılmış değerler anlamına gelmektedir. Bu dosya tipi, veriyi tutmak için kullanılan en basit yapılardandır. Her kayıt yeni bir satırda ve her kayıttaki her değer yeni bir kolonda gösterilmektedir. Ancak kolon yapısının metin dosyalarında (text file) tutulması güç olduğu için kolon ayıracı olarak virgül kullanılmıştır. Örneğin aşağıdaki gibi bir tabloyu ele alalım:

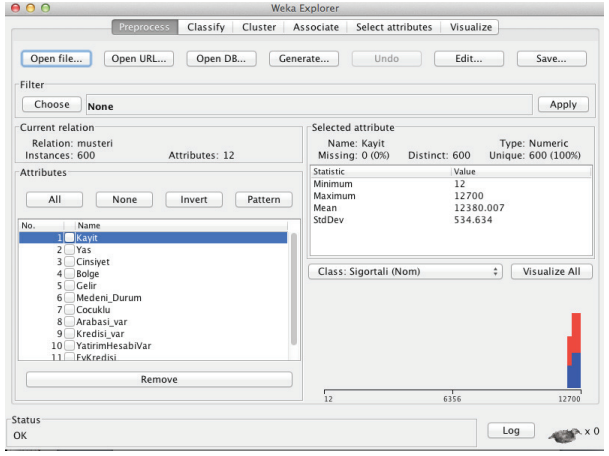
İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—-	-	Muhasebe
Veli Demir	43	Yönetim

Bu tablonun CSV karşılığı, aşağıdaki şekilde bir metin dosyası olacaktır.

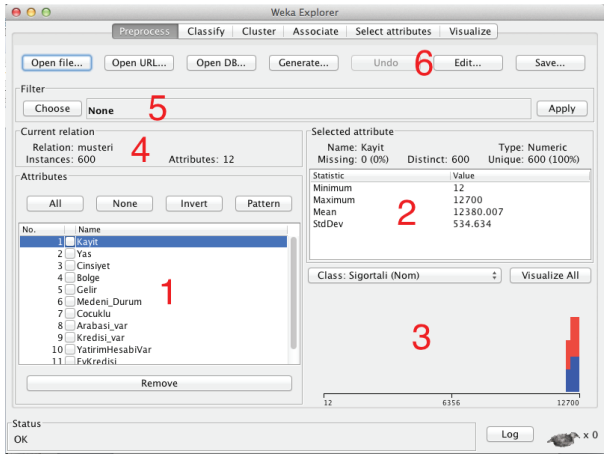
```
İsim, yaş, kısım
Şadi Evren ŞEKER,33,Bilgisayar
Ali Demir,23,Muhasebe
Cem Yıldız,26,Bilgisayar
Ahmet Yılmaz,33,
,,Muhasebe
Veli Demir,43,Yönetim
```

Ancak bazı durumlarda, CSV dosyasında bulunan yazıların içerisinde özel karakterlerin kullanılmasından dolayı, yazı tipindeki değerlerin tırnak işaretleri arasında kullanılması daha doğrudur. Örneğin yazının içerisinde geçen virgüller, ayıraç olan virgüller ile karışabilir. Tırnak işaretleri kullanılarak bu ve benzeri karışıklıklar giderilmiş olur.

Musteri.csv dosyası yüklendikten sonra, aşağıdakine benzer bir şekilde Weka'nın veri ön işleme ekranı belirecektir.



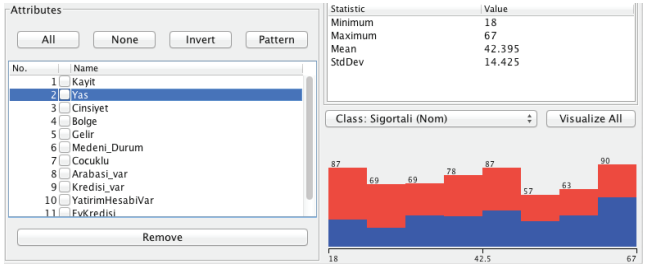
Veri ön işleme ekranı bir kaç farklı bölümden oluşmaktadır. Bu alanlar aşağıdaki şekilde sayılar ile işaretlenmiş ve açıklanmıştır:



1 Numaralı alanda veri kümemizde bulunan özellikler (tablo olarak düşünülürse, kolon başlıkları) yer almaktadır. CSV dosyasını ayrıca bir editör ile (örneğin wordpad) açıp içine bakmanızı tavsiye ederim. Bu dosyanın ilk satırında bulunan kolon başlıkları olduğu gibi 1 numaralı alanda sıralanmıştır.

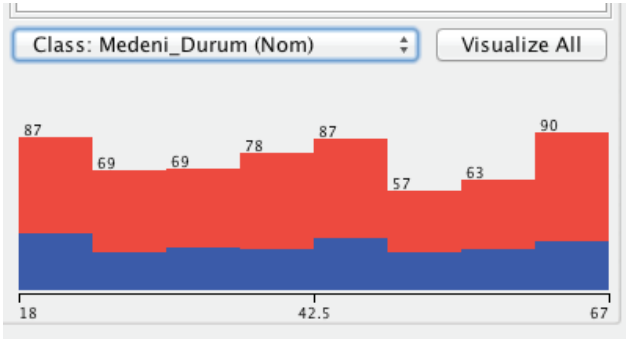
2 Numaralı alanda ise şu anda seçilmiş olan kolon ile ilgili istatistiksel veriler bulunmaktadır. Minimum, maximum, ortalama ve standart sapma değerleri yer almaktadır. Seçilen alanın özelliklerine göre buradaki istatistiksel bilgiler değişmektedir.

3 Numaralı alan ise 1 numaralı alanda seçilen kolonun görüntülenmiş (visualize) halidir. Örneğin 1 numaralı özellikler listesinde "Yas" seçeneğini seçersek aşağıdaki şekilde 3 numaralı alanın değiştiğini görebiliriz:



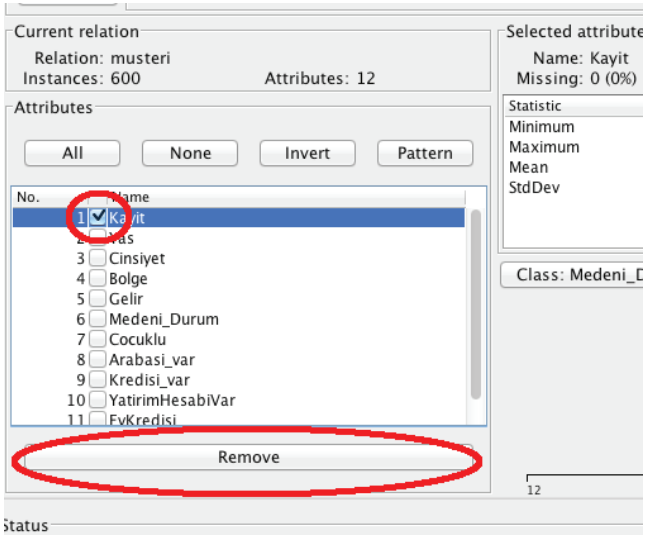
Yeni haliyle, 18 – 67 yaşları arasındaki kişilerin sayılarını göstermektedir (histogram). Ayrıca 3 numaralı alanda bulunan "Class: Sigortali" seçeneği de iki farklı alanı karşılaştırarak aynı grafikte görme imkanı sağlar. Şu anda seçili olan sınıf "sigortali" özelliğidir. Bunun sebebi veri kümesindeki son kolon olmasıdır. Genelde Weka, son kolonu, çalışmalarındaki hedef kolon olarak belirler, bunu elbette değiştirebiliriz ancak ilk seçimde gelen özellik bu şekildedir. Grafikte de sigortalı durumu evet (E) ve hayır (H) olan kişiler kırmızı ve mavi renklerle gösterilmektedir.

Örneğin bu "Class: Sigortali (Nom)" seçeneğine tıklayarak açılan menüden "medeni_durum" seçeneğini seçecek olursak aşağıdaki şekilde evli ve bekar dağılımı gösterilecektir:



Veri ön işleme sırasında hedefimize uygun olmayan verilerin işlemden çıkarılması mümkündür. Bu sayede gereksiz veriler ile uğraşılmayacak, hem vakit hem de bilgisayarın hafızasında yer kazanılacaktır.

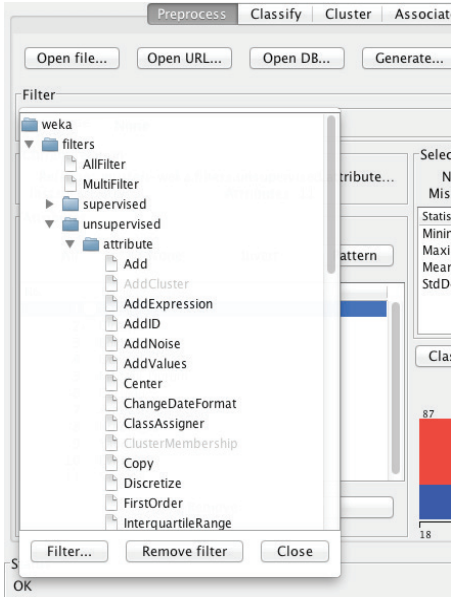
Örneğin müşteri numaralarının tutulduğu "Kayıt" özelliğini çalışmamızdan çıkarmak isteyelim. Öncelikle bu özelliğin yanındaki kutucuğu işaretliyoruz ve ardından listenin altındaki "Remove" düğmesine tıklıyoruz.



Kayıt listeden çıkarıldıktan sonra, kalan elemanlar ile çalışmaya devam edilebilir. Bu yaptığımız veriyi çıkarma işlemi, Weka'nın veri ön işleme ekranının bir özelliğidir. Daha ileride de detaylıca anlatılacağı üzere, Weka'yı kullanarak yaptığımız her işlem, aslında bir Java kodunu çağırmaktadır ve arka planda bir Weka kütüphanesi çalışmaktadır. Bu çalışan kütüphaneleri doğrudan çağırma imkanımız vardır.

Örneğin bir özelliği çıkarmak için Weka'nın filtreleme kütüphanesini açıp buradaki "Remove" fonksiyonunu çalıştırabiliriz.

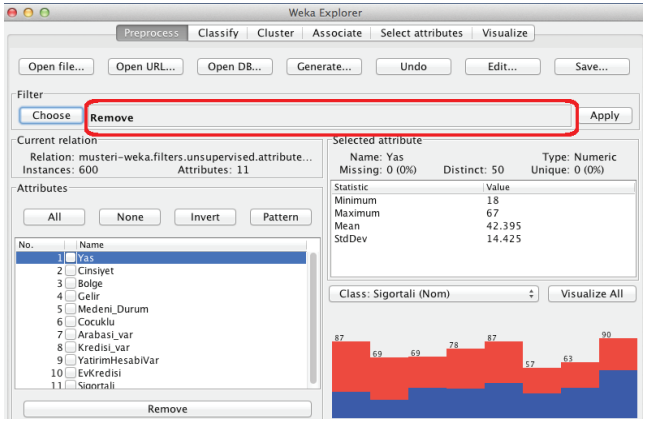
Yukarıdaki kaldırma işlemi, alternatif olarak aşağıdaki şekilde ekranın 5 numaralı alanında bulunan Filter bölümündeki "Choose" düğmesine basılarak da yapılabilirdi.



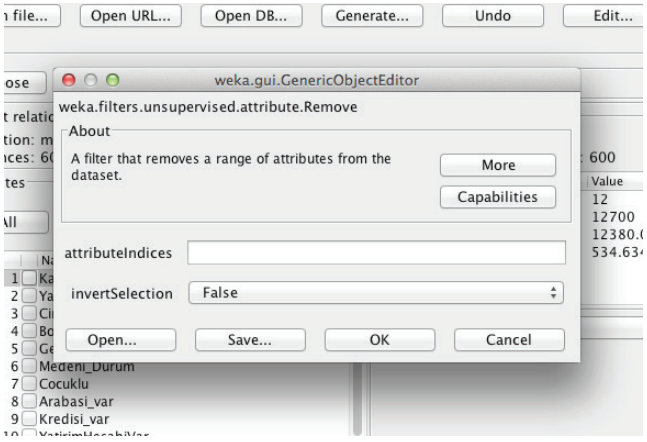
Yukarıdaki şekilde gösterildiği üzere, "Choose" düğmesine basıldığında açılan listeden, "filters" altındaki "unsupervised" altındaki "attribute" altındaki "remove" filtresini seçmemiz gerekir. Bu filtre "attribute" altında seçildiği için bir özelliği (kolonu) komple işlemden kaldıracaktır.

Seçim işlemi tamamlandıktan sonra hangi özelliğin kaldırılacağının da parametre olarak filtreye verilmesi gerekir.

Seçim işleminin ardından aşağıdaki şekilde ekranda 5 numaralı filtre alanında "remove" belirecektir. Bu alanda herhangi bir yere (kırmızı kutu içerisine alınan alanda) tıklayarak filtrenin özelliklerini açabiliriz:

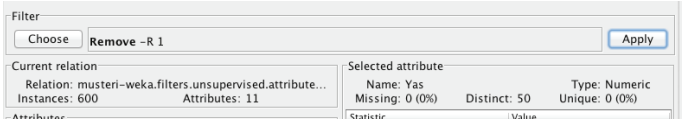


Açılan filtre özellikleri aşağıda gösterilmiştir:



Bu diyalogda bulunan "attributeIndices" seçeneği hangi özelliklerin kaldırılacağı bilgisini alır. Bizim örneğimizde ilk özelliğin kaldırılmasını istiyoruz.

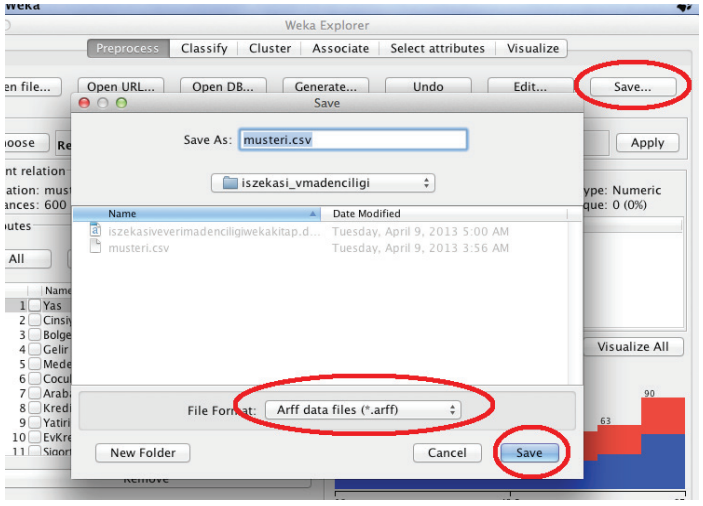
Dolayısıyla buradaki kutuya 1 yazıp "ok" düğmesine tıkladığımızda, Weka'nın veri ön işleme ekranına geri döneceğiz. Bu ekrandaki filtrenin yanına parametre olarak "–R 1" geldiğini görebilirsiniz.



Ardından ekranın en sağında bulunan "Apply" düğmesine tıklandığında filtre uygulanacaktır.

Hemen bu aşamada, daha ileride çok işimize yarayacak olan 4 numaralı alandaki "Current Relation" alanından bahsetmek istiyorum. Burada şu anda kullanılan Weka kütüphanesi ve fonksiyonunun detayları yer almaktadır. Ayrıca filtre uygulandıktan sonra, veri kümemizde kalan kayıt ve özellik sayıları da bu alanda bulunmaktadır.

Daha önce, Weka'nın kullandığı dosya uzantısının "ARFF" olduğundan bahsetmiştik. Şimdi mevcut çalışmamız olan CSV dosyasını bir .arff dosyası olarak kaydetmeyi deneyelim. Bunun için ekranın 6 numaralı alanında bulunan düğmelerden, "Save" düğmesini kullanacağız.



Save düğmesine tıklandığında, kayıt işlemi için yukarıdaki şekilde bir diyalog açılacaktır. Açılan diyalogta kaydetmek istediğimiz dosya yapısı olarak "File Format", arff seçeneğini seçip, dosyamıza bir isim verip diyalogdaki "Save" düğmesi ile kayıt ediyoruz.

Kaydedilen dosyanın içeriğini, herhangi bir editör (örneğin wordpad) ile açıp bakarsak aşağıdaki şekilde görülecektir:


```
1  @relation musteri-weka.filters.unsupervised.attribute
2
3  @attribute Yas numeric
4  @attribute Cinsiyet {Kadin,Erkek}
5  @attribute Bolge {sehir_Merkezi,Kirsal,Koy,sehir_Disi
6  @attribute Gelir numeric
7  @attribute Medeni_Durum {H,E}
8  @attribute Cocuklu numeric
9  @attribute Arabasi_var {H,E}
10 @attribute Kredisi_var {H,E}
11 @attribute YatirimHesabiVar {H,E}
12 @attribute EvKredisi {H,E}
13 @attribute Sigortali {E,H}
14
15 @data
16 48,Kadin,sehir_Merkezi,17546,H,1,H,H,H,E
17 40,Erkek,Kirsal,30085.1,E,3,E,H,E,E,H
18 51,Kadin,sehir_Merkezi,16575.4,E,0,E,E,E,H,H
19 23,Kadin,Kirsal,20375.4,E,3,H,H,E,H,H
20 57,Kadin,Koy,50576.3,E,0,H,E,H,H,H
21 57,Kadin,Kirsal,37869.6,E,2,H,E,E,H,E
22 22,Erkek,Koy,8877.07,H,0,H,H,E,H,E
23 58,Erkek,Kirsal,24946.6,E,0,E,E,E,H,H
24 37,Kadin,sehir_Disi,25304.3,E,2,E,H,H,H,H
25 54,Erkek,Kirsal,24212.1,E,2,E,E,E,H,H
26 66,Kadin,Kirsal,59803.9,E,0,H,E,E,H,H
27 52,Kadin,sehir_Merkezi,26658.8,H,0,E,E,E,E,H
28 44,Kadin,Kirsal,15735.8,E,1,H,E,E,E,E
29 66,Kadin,Kirsal,55204.7,E,1,E,E,E,E,E
30 36,Erkek,Koy,19474.6,E,0,H,E,E,E,H
31 38,Kadin,sehir_Merkezi,22342.1,E,0,E,E,E,E,H
32 37,Kadin,Kirsal,17729.8,E,2,H,H,H,E,H
33 46,Kadin,sehir_Disi,41016,E,0,H,E,H,E,H
```

Dikkat edileceği üzere, ilk satırda kullandığımız filtre bulunmaktadır. 3 ile 13. satırlar arasında, veri kümemizdeki her özelliğin ismi birer satırda yazılmıştır. Weka bu özellikleri daha sonraki aşamalarda kullanmaktadır. Ayrıca her özelliğin tipi de yanında yer alacaktır. Örneğin 6. satırda yer alan Gelir özelliğine dikkat edecek olursak "nümerik" olarak atanmıştır. Bunun anlamı bu özelliğin sayısal bir değer olduğudur. Benzer şekilde 4. satırdaki "Cinsiyet" özelliği ise iki ihtimalden birisi olabilir {Ka-

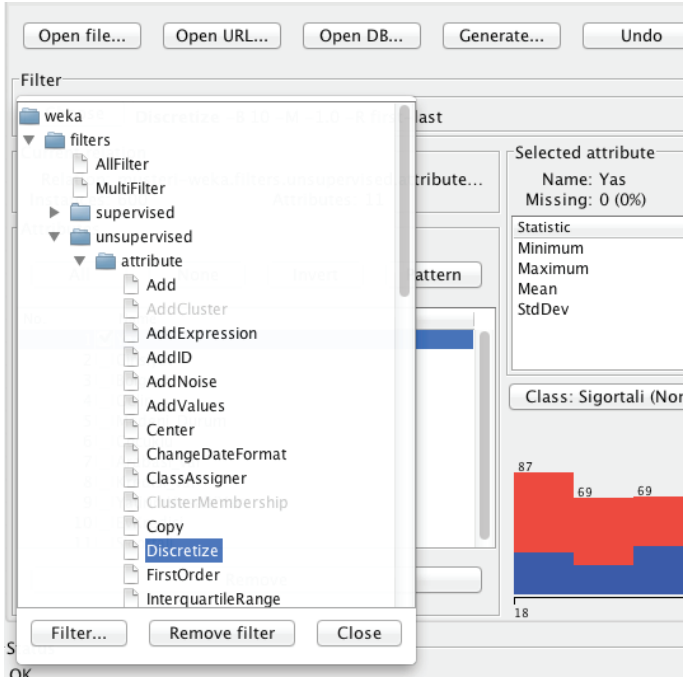
din, Erkek} kümesinin elemanlarından birisini alabilir şeklinde atanmıştır. Weka bizim için veriye bakarak bu özellikleri ve tipleri otomatik olarak çıkarmaktadır.

15. satırdan sonra, verinin başladığı "@data" etiketi ile belirtilmiştir ve her satırda, CSV dosyasına benzer şekilde virgül ile ayrılmış alanlarda veri kümesinin o satırının ilgili değerleri bulunmaktadır.

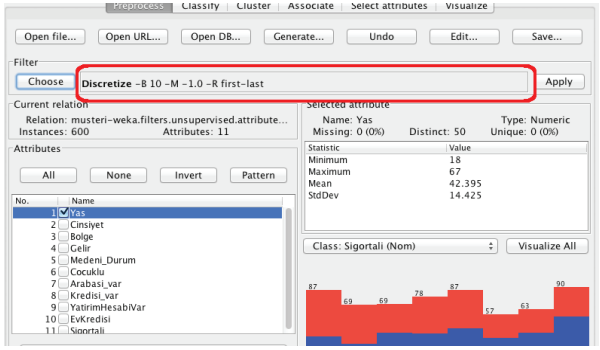
4.8. Weka ile Ayrık Veri Dönüşümü (Discretize)

Amacımız, basitçe Weka'yı kullanarak daha önce kutulama (binning) ve veri adetleme (Data Quantization) olarak açıklanan filtreyi uygulamak. Daha ileride göreceğimiz üzere sınıflandırma algoritmaları (classification) sayısal veriler üzerinden değil sınıflar üzerinden çalışabilir. Örneğin veri kümemizdeki "çocuklu" özelliğinde 0 değeri müşterinin çocuğu olmadığını belirtirken, 1, 2 veya 3 gibi değerler de müşterinin kaç çocuğu olduğunu belirtmektedir. Bu değerler sayısal olduğu için, bir önceki aşamada Weka tarafından bu özellik nümerik olarak atanmıştır. Ancak biz biliyoruz ki bu özellik sürekli bir değer değil, aksine kesik (discrete) bir değerdir. O halde yaş verisini kesikli hale getirmek isteyelim (discretize).

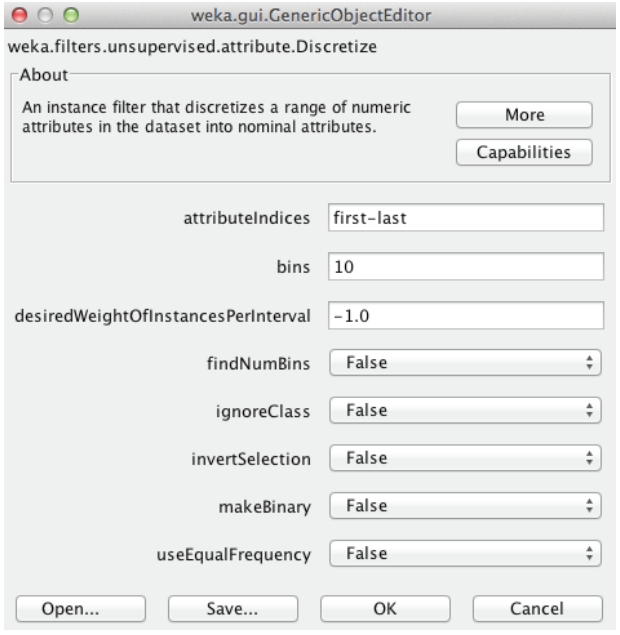
Öncelikle yüklü olan veri kümemizdeki yaş özelliği seçilir. Ardından filtreler listesinden, unsupervised altındaki attributes altındaki "discretize" seçeneği seçilir.



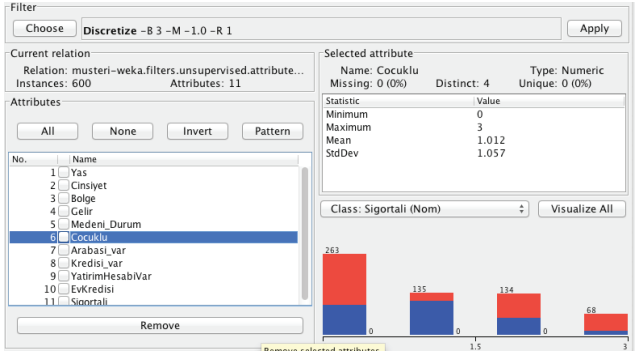
Bu filtrenin amacı verilen aralıklardaki değerlere göre kesik (discrete) dönüşümü yapmaktır. Ardından filtre alanına tıklanarak açılan filtre özelliklerinde ayarlamaya girilir:



Yukarıdaki işaretli kutu içerisindeki alana tıklanarak filtrenin ayar diyaloğu açılır.



Açılan diyalogta, attributeIndices kısmı, her bir kutunun (bin) genişliğini vermektedir. Biz kutu genişliği olarak 1 seçelim. Bu sayede her çocuk sayısı için ayrı bir sınıf oluşturulacaktır. Kutu sayısı ise bins olarak geçen alana yazılır. Bizim için en fazla 3 çocuğu olan müşteriler olabileceğini düşünerek buraya da 3 yazalım. Son olarak OK düğmesine basarak filtreyi uygulayalım:



Filtreyi uygulamak için, filtre alanındaki "Apply" düğmesine basılması yeterlidir. Yukarıdaki şekilde görüldüğü üzere, 4 kolonda bütün bilgiler toplanmıştır. Sırasıyla hiç çocuğu olmayanlar, 1, 2 ve 3 çocuğu olanlar ayrı kolonlarda toplanmış ve sayıları gösterilmiştir. Bu uygulama, çocuk sayısı için çok fazla bir etki oluşturmaz çünkü çocuk sayısı zaten kesikli bir veri olup tam da verilen filtreye uygun özelliktedir. Ancak aynı filtre yaş özelliğine uygulanırsa bu durumda aşağıdaki şekilde bir değişim gözlenebilir:

```
@attribute Yas {'\'(-inf-34.333333]\'','\'(34.333333-50.666667]\'','\'(50.666667-inf)\''}
@attribute Cinsiyet {Kadin,Erkek}
@attribute Bolge {sehir_Merkezi,Kirsal,Koy,sehir_Disi}
@attribute Gelir numeric
@attribute Medeni_Durum {H,E}
@attribute Cocuklu numeric
@attribute Arabasi_var {H,E}
@attribute Kredisi_var {H,E}
@attribute YatirimHesabiVar {H,E}
@attribute EvKredisi {H,E}
@attribute Sigortali {E,H}

@data
'\'(34.333333-50.666667]\'',Kadin,sehir_Merkezi,17546,H,1,H,H,H,H,E
'\'(34.333333-50.666667]\'',Erkek,Kirsal,30085.1,E,3,E,H,E,E,H
'\'(50.666667-inf)\'',Kadin,sehir_Merkezi,16575.4,E,0,E,E,E,H,H
'\'(-inf-34.333333]\'',Kadin,Kirsal,20375.4,E,3,H,H,E,H,H
'\'(50.666667-inf)\'',Kadin,Koy,50576.3,E,0,H,E,H,H,H
'\'(50.666667-inf)\'',Kadin,Kirsal,37869.6,E,2,H,E,E,H,E
'\'(-inf-34.333333]\'',Erkek,Koy,8877.07,H,0,H,H,E,H,E
```

Görüldüğü üzere, arff dosyasında yaş alanı 3 parçaya bölünmüş, eksi sonsuzdan 34.33'e kadar olan sayılar bir grupta, 34.33'ten 50.66'ya kadar olan yaşlar diğer bir grupta ve 50.66 üzeri yaşlar ise diğer bir grupta toplanmıştır. Burada dikkat edilecek bir nokta, normal parantez kullanılan değerin dahil olmadığı ancak köşeli parantezin olduğu taraftaki değerin ilgili alana dahil olduğudur.

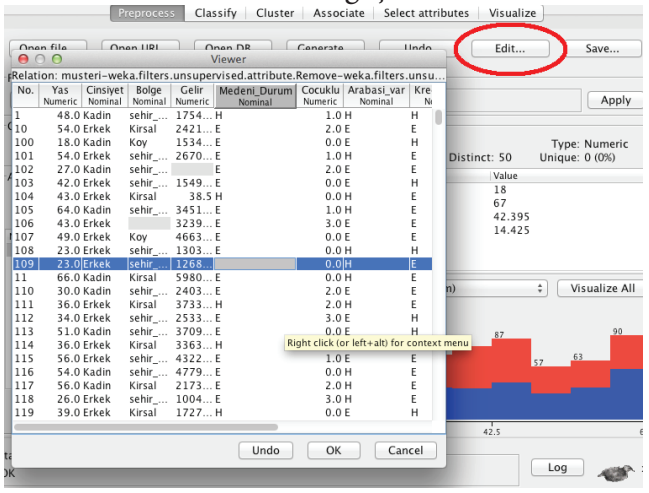
Veri kısmında ise bu değişiklik uygulanmış ve bütün verilerde ilgili alana düşen değerler bu 3 sınıftan birisi ile değiştirilmiştir.

Neticede veri kümesindeki yaş alanı 3 parçaya bölünerek veri kümesindeki her satır bu 3 kutudan en uygun olanına yerleştirilmiş ve 3 grup dışında 4. bir gruba izin verilmemiştir.

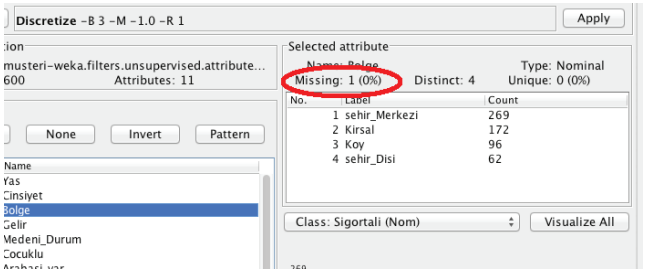
4.9. Weka ile Eksik Verilerin Ön İşlenmesi

Yeni bir örnek olarak veri kümemizdeki bazı değerlerin eksik olması halinde Weka ile nasıl işleyeceğimizi görelim. Bunun için veri kümemizin bulunduğu arff dosyamızı yeniden açalım ve Edit düğmesine tıklayarak

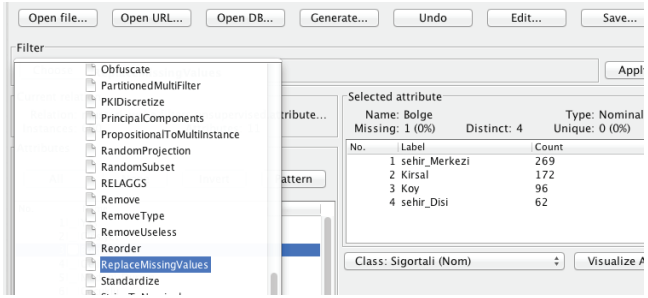
verileri düzenleme ekranına geçelim:



Silmek istediğimiz hücreye tıklayarak değiştirme imkanımız bulunmaktadır. Bu değiştirme işlemi sonucunda veri kümesinden bazı değerleri rastgele olarak silelim. Şimdi artık veri kümemizde eksik veri gösterilecektir. Bunu selected attribute alanındaki "missint value" değerinden takip edebiliriz.



Seçilen veri kümesi özelliğinden ne kadarının eksik olduğu sayılarak bu alanda gösterilir. Amacımız bu eksik değerlerden veri kümemizi temizlemek olsun. Bu amaçla yine bir veri ön işleme aşaması olan ReplaceMissingValues filtresini kullanacağız.



Filtre seçildikten ve Apply düğmesine basılıp uygulandıktan sonra eksik olan verinin doldurulduğu görülebilir. Burada en yoğun olan Şehir Merkezi değeri otomatik olarak bu alana doldurulmuştur.

5. Veri Dönüştürme (Data Transformation)

Bu bölümde, VTBK (KDD) akışının ikinci aşaması olan veri ön işleme aşamasının nasıl yapılabileceği açıklanacaktır.

5. Veri Dönüştürme (Data Transformation)

Veri dönüşümü aşaması, verinin işlenmesi ve üzerinde veri madenciliği yapılmasına uygun hale getirilmesi için uygulanan yöntemdir. Örneğin verinin boyutlarının çıkarılması, verinin anlamlı aralıklara indirgenmesi, verinin üzerinde genelleme yapılması ve hatta verilen veri üzerinden yeni özellikler çıkarılması bu aşamada ele alınır. Temel olarak veri dönüştürme işlemi sırasında yapılan işlemler aşağıdaki şekilde sıralanabilir:

- Düzleştirme (Smoothing)
- Birleştirme (Aggregation)
- Genelleme (Generalization)
- Normalleştirme (Normalization)
- Özellik İnşası (Attribute Construction)

Yukarıdaki bu işlemlerin her birisi alt başlık olarak bu bölüm altında incelenecektir. Ancak hepsinin genel olarak amacının, verinin işlenmesi ve daha anlamlı sonuçlar çıkarılması olduğu unutulmamalıdır.

5.1. Düzleştirme (Smoothing)

Düzleştirme işleminin amacı, veri üzerinde belirli yapıları yakalamak ve bu yapıları veri üzerinde uygulamaktır. Bu sayede veride bulunan gürültülerin giderilmesi hedeflenir. Düzleştirme için çok sayıda farklı metot kullanılabilir, genelde sinyal işleme ve filtreleme yöntemlerinin kullanıldığı düzleştirme işlemlerinden bazılarına, fikir vermesi açısından bu kitapta yer verilmiştir. Diğer düzleştirmede kullanılan sinyal işleme fonksiyonları daha detaylı olarak sinyal işleme kitaplarından öğrenilebilir. Bu bölümde, IMKB[2] verileri üzerinden düzleştirme işlemlerinin nasıl uygulandığı gösterilecektir.

Yukarıdaki şekilde, Ocak 2010 ile Ocak 2012 arasındaki IMKB kapanış değerleri verilmiştir. Bu bölümde anlatılan düzleştirme işlemleri, yukarıdaki borsa değerleri üzerinde çalıştırılmıştır. Okuyucunun takip edebilmesi için, işlenmemiş ham veri yukarıda verilmiştir.

(2) IMKB ismi Nisan 2013 itibariyle BIST olarak değiştirilmiştir ancak alınan değerler IMKB zamanından olduğu için kitap kapsamında IMKB ifadesinin kullanılması daha uygun görülmüştür.

5.1.1. Rassal Yürüme (Random Walk)

Bu yöntemde, verinin üzerinde tanımlı olan bir yürüme uzunluğu (yol, path) içinde kalan veri değerleri üzerinde değişim analizi yapılır. Örneğin yürüme uzunluğu 10 olan bir analizde, verinin üzerindeki ilk 10 değer alınarak bu yoldaki rassal olaylar incelenir ve bu 10 uzunluğundaki yol için tek bir değer döndürülür.

Örneğin borsa verileri için son 10 kapanışın artma veya azalma yönünde olduğuna bakılır. Bu değerlerin rassal olarak artı veya eksi yönde olduğu kabulüne dayanılarak artı değerler için +1 ve eksi değerler için -1 dönüşümü yapılır. Ardından aşağıdaki formülde verildiği üzere n=10 için 10 uzunluğundaki bir yolun hesabı yapılır.

$$s_n = \sum_{j=1}^{n} Z_j$$

Yukarıdaki formülde n değeri değiştirilerek farklı uzunluktaki yollar için rassal yürüme analizi yapmak mümkündür.

Rassal yürüme yöntemi, verinin daha geniş bir aralıktan toplanmasını hedeflemekte, bu sayede ani artma ve azalma değerlerini düzleştirmektedir.

5.1.2 Hareketli Ortalama (Moving Average)

Bu yöntemde, yine tanımlı bir alan için ortalama değer alınır. Örneğin aralık uzunluğumuz 10 olsun. Veri kümesinden ilk 10 değer (1-10 arasındaki değerler) alınarak ortalamaları hesaplanır ve bulunan değer

ilk değer olur. Ardından ikinci 10 değer alınır (2-11 arasındaki değerler) ve yeni bir ortalama hesaplanarak ikinci değer olarak atanır. İşlem, aralığın birer birer kaydırılmasıyla, veri kümesinin sonuna kadar devam eder.

$$SMA_{t,n} = \sum_{i=t-n}^{t} \frac{P_i}{n}$$

Yukarıda verilen hareketli ortalamanın özel bir şekli olan standart hareketli ortalama (Standard moving average) değeridir. Buna göre, verilen uzunluktaki (n) değerlerin (P) ortalaması hesaplanır.

Diğer bir hareketli ortalama hesaplama yöntemi ise ağırlıklı hareketli ortalama (weighted moving average) yöntemidir.

$$WMA_{t,n} = \sum_{i=t-n}^{t} \frac{(n+i-t)P_i}{n}$$

Bu yöntemde, hesaplanan her değer için bir katsayı ile çarpılma söz konusudur. Örneğin 10 günlük bir aralık hesaplanırken, son gün daha önemli olduğu için 10 ile, bir önceki gün 9 ile, bir önceki gün 8 ile, böylece 10 gün önceki değer ise 1 ile çarpılarak toplanır ve 1'den 10'a kadar sayıların toplamı olan 50 değerine bölünür. Bu sayede günlerin ağırlıkları farklı katsayılar ile çarpılmış ve farklı önemler verilmiş olur.

Kitap kapsamında anlatılacak olan son ağırlıklı ortalama hesaplama yöntemi ise üstel ağırlıklı hareketli ortalama hesabıdır.

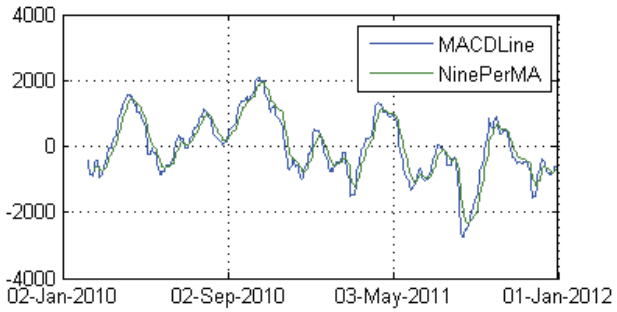
$$EMA_{t,n} = \sum_{i=t-n}^{t} \frac{(1-\alpha)^{n+i-t}P_i}{n}$$

Bu yöntemde kullanılan katsayı değerleri üstel olarak azalmaktadır. Formülde de gösterildiği üzere, verilen α α değeri için toplamda 1 olacak değerler zamana yayılır ve katsayı olarak günlük değerlerin başına yazılır. Bu sayede ağırlıklı ortalamaya göre yakın günlere çok daha fazla önem verilirken uzak günlere daha düşük değerde önem verilmiş olur.

Yukarıda açıklanan hareketli ortalama değerlerinin üzerinden, hareketli ortalamanın yakınsama veya ıraksama durumuna göre de analiz yapmak mümkündür. Buna da literatürde MACD (Moving average convergence / divergence) ismi verilir ve Türkçede 'hareketli ortalama yakınsama ıraksama' olarak çevirmek mümkündür.

$$MACD = EM_{kısa} - EMA_{uzun}$$

Basitçe bir kısa dönem bir de uzun dönemdeki üstel hareketli ortalama değerlerinin farkı olarak düşünülebilir.



Yukarıdaki şekilde, MACD çizgisi, 10 uzunluğundaki bir aralık için hesaplanmıştır. Ayrıca farkın karşılaştırılabilmesi için 9 uzunluğundaki bir alan üzerinden hesaplanan hareketli ortalama değeri de NinePerMA ismi ile gösterilmiştir.

5.1.3. Göreceli Güç Indeksi (Relative Strength Index)

Göreceli Güç İndeksi (RSI), üstel hareketli ortalama hesabının üzerine kurulmuş bir hesaplama yöntemidir. Konuyu daha iyi anlamak için borsa verisine dönecek olursak, iki üst üste kapanış değeri için üç farklı ihtimalden bahsedilebilir. $K_{şimdi}$ ve $K_{önceki}$ şeklinde iki ardışık kapanış değerini ele alacak olursak, RSI değeri aşağıdaki şekilde hesaplanır ve 3 farklı durum için bulunur.

$$RSI = 100 - \frac{100}{1 + RS}$$

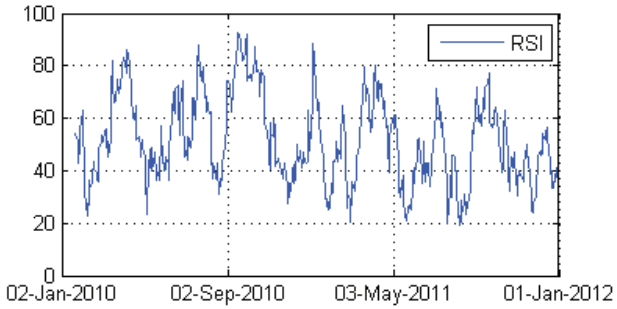
Bu formüldeki RS değeri için aşağıdaki hesaplama yapılır.

$$RS = \frac{EMA_{U,n}}{EMA_{D,n}}$$

Yukarıdaki RS formülü için verilen U ve D değerleri de aşağıdaki 3 farklı durum için ayrı ayrı hesaplanır.

$$\begin{cases} U = K_{şimdi} - K_{önceki}, D = 0, if\ K_{şimdi} > K_{önceki} \\ U = 0, D = K_{önceki} - K_{şimdi}, if K_{şimdi} < K_{önceki} \\ U = 0, D = 0, if\ K_{şimdi} = K_{önceki} \end{cases}$$

Yukarıdaki hesaplamaya göre iki değer, yani U ve D değerleri denklemde yerine yazılır ve sonuç değerleri indeks üzerinde işaretlenir.



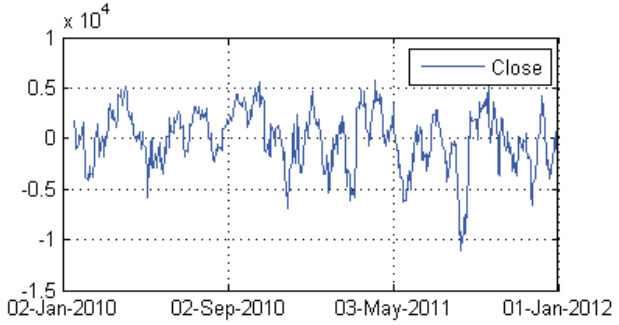
IMKB üzerinde uyarlanmış hali, yukarıdaki şekilde verilmiştir.

5.1.4. Momentum ve Değişim Oranı (Rate of Change)

Bu iki yöntemin aynı başlıkta toplanmasının sebebi aslında ikisinin hesaplanmasının birbirine çok benzer olmasıdır. Değişim oranının hesaplanması için öncelikle momentum hesaplanır.

$$momentum_n = K_{şimdi} - K_{şimdi-n}$$

Verilen n uzunluğundaki bir momentum değeri için kapanış değerleri arasındaki n günlük fark hesaplanır. Buradaki n, parametrik olarak gelen bir değerdir.



Yukarıdaki şekilde, n=10 için hesaplanmış momentum değerleri gösterilmektedir.

Değişim oranı (rate of change) ise aşağıdaki formül ile hesaplanabilir.

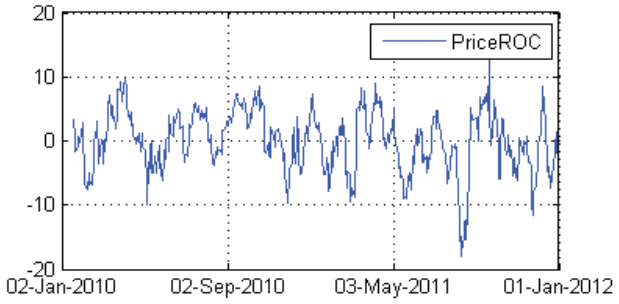
$$rate\ of\ change_n = \frac{momentum_n}{K_{şimdi-n}}$$

Formülden de anlaşılacağı üzere, verilen n uzunluğu için momentum değerinin, son n gündeki değişime oranı hesaplanmaktadır.

Yukarıdaki formülde, momentum değeri bir önceki formülden yerine konulacak olursa aşağıdaki formül de elde edilebilir:

$$rate\ of\ change_n = \frac{K_{şimdi} - K_{şimdi-n}}{K_{şimdi-n}}$$

Değişim oranının IMKB kapanış değerleri üzerinde uyarlanmış hali ise aşağıda verilmiştir.



Dikkat edilirse, momentum grafiğinden çok az farklı bir grafik elde edilmiştir.

5.1.5. Bollinger Bantı (Bollinger Bant)

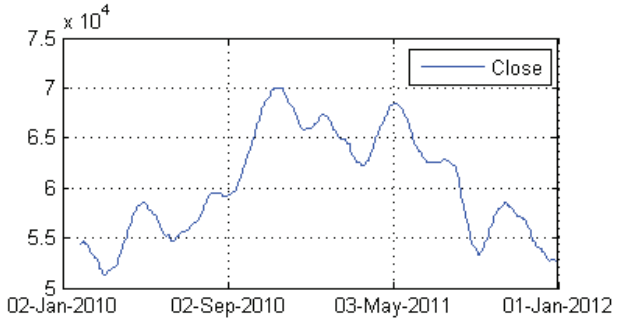
Bollinger bandı, üst, alt ve orta kanallar üzerinden düzleştirilmiş veri elde etmeyi amaçlar. Bu üç değerin hesaplanması aşağıdaki şekilde yapılabilir:

$$BB_{orta,n} = MA_n$$

$$BB_{üst,n} = \sum_{i=t-n}^{t} 1, if\ MA_n > K\sigma_n$$

$$BB_{alt,n} = \sum_{i=t-n}^{t} 1, if\ MA_n < K\sigma_n$$

Bu hesaplamada kullanılan K değeri bir sabittir.



Yukarıdaki şekilde K = 20 olarak hesaba dahil edilmiş ve sadece orta bollinger bandı çizilmiştir.

5.1.6. İvme (Acceleration) ve Fark (Difference)

Bu iki yöntem aslında oldukça basit bir mantık üzerine kuruludur. Fark hesaplaması sırasında mevcut fiyat ile n gün önceki fiyat arasındaki fark hesaplanır. Bu fark değeri indeks değeri olarak işleme alınır.

$$Fark_n = K_{şimdi} - K_{şimdi-n}$$

İvme değeri ise, hareketli ortalama yakınsak / ıraksak değeri ile verinin histogramı arasındaki farktır.

$$Acc = diff(MACD_n, Histogram)$$

Histogram *(Tekrar Dağılımı)*

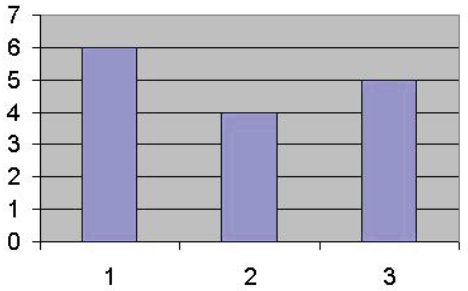
Tekrarlı sayıları içeren bir dizideki her sayının tekrar miktarını veren dağılımdır. Ayrıca bu dağılım kullanılarak çizilen grafiğe de histogram ismi verilmektedir. Örneğin 1'den 3'e kadar sayıların olabildiği aşağıdaki seriyi ele alalım:

1,3,1,2,3,2,1,1,1,2,2,3,1,3,3

Yukarıdaki 15 sayıda, sadece 1,2 ve 3 değerleri bulunmaktadır. Bu sayıların histogramı:

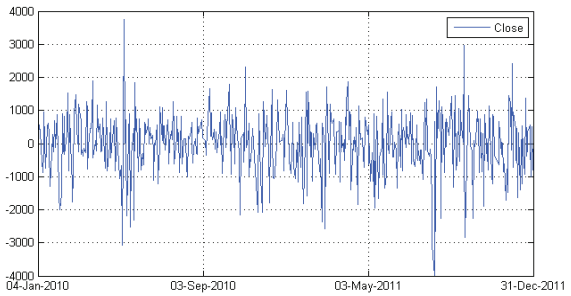
1 -> 6
2 -> 4
3 -> 5

adet olarak gösterilir. Yani seride kaç kere geçtiklerinin sayısıdır. Grafik olarak gösterilmek istenirse de aşağıdaki şekilde bir grafik çizilebilir.

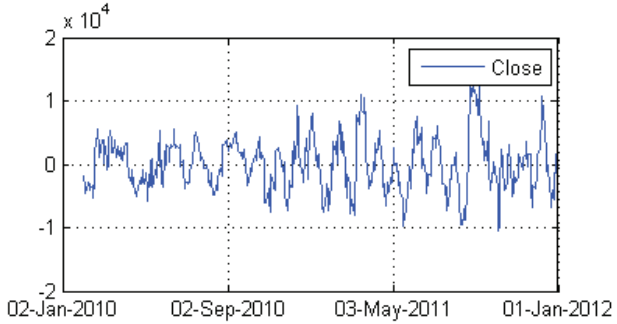


Dolayısıyla bir veri kümesindeki histogram değeri o veri kümesindeki tekrarlı değerlerin sayılarıdır.

Yukarıdaki formüllere göre hesaplanan ivme grafiği aşağıda verilmiştir:



Ayrıca fark grafiği de aşağıdaki şekilde çizilebilir:

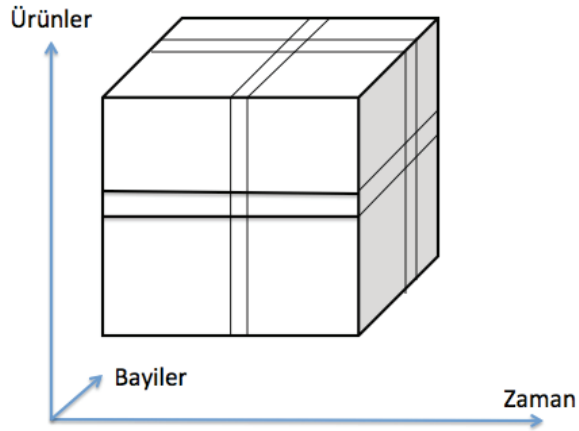


5.2. Veri Birleştirme (Data Aggregation)

Veri birleştirmesi işlemi, birden fazla veri kaynağından gelen farklı verilerin ilişkilendirilmesi işlemidir. Buradaki ilişkilendirme aslında birden fazla veri kaynağının ortak alanda kullanılmasıdır. Örneğin bir kişinin kilosu ve boyu iki farklı boyut olarak aynı yapı üzerinde birleştirilebilir.

Verilerin farklı boyutlarda birleştirilmesi ve 3 boyutlu veri yapıları elde etme işlemine ayrıca veri küpü (data cube) ismi verilmektedir.

Örneğin aşağıdaki şekilde, bir firmanın farklı zamanlarda (zaman), ürün ve bayilerinin satış bilgileri tutulmuştur.



Yukarıdaki örnekte, 3 boyutlu bir küp üzerinde seçilen bir bayinin verilen bir zamandaki herhangi bir ürün fiyatını görmemiz mümkündür. Buna göre küp üzerinde dilimler alınarak önce 2 boyutlu tabloya indirmek, ardından da bu tablo üzerinden seçim yapmak mümkündür.

Yukarıdaki örnekte gösterilen çok boyutlu veriler, veri tabanı üzerinde yapılan işlemler üzerinden çıkarılabilir. Bu işlemleri (transactions) takip etmek için veri tabanlarının çoğunda uygun teknoloji bulunmaktadır. Genelde bu sistemlere hareket işlem sistemleri (HİS, transaction processing systems, TPS) ismi verilmektedir.

5.2.1. HİS (Hareket İşlem Sistemleri, TPS, Transaction Processing Systems)

HİS sistemlerinin veri tabanı teorisinden gelen temel görevleri, kullanıcıların verilere kolay ulaşmasını sağlamak ve veri bütünlüğünü sağlamak olarak sayılabilir.

Bu amaçla bir HİS sisteminde aşağıdaki dört özelliğin bulunması beklenir. Bu özellikler, İngilizce baş harfleri birleştirilerek kısaca ACID olarak anılır ve atomluluk (atomicity), tutarlılık (consistency), yalıtımlılık (isolation) ve devamlılık (durability) kelimelerinden oluşur.

5.2.1.1. Atomluluk (Atomicity)

Latince bölünemez anlamına gelen atom kökünden üretilen bu kelime, bilgisayar bilimlerinde çeşitli alanlarda bir bilginin veya bir varlığın bölünemediğini ifade eder.

Örneğin programlama dillerinde bir dilin atomik (bölünemez) en küçük üyesi bu anlama gelmektedir. Mesela C dilinde her satır (statement) atomik (bölünemez) bir varlıktır.

Benzer şekilde bir verinin bölünemezliğini ifade etmek için de veri tabanı, veri güvenliği veya veri iletimi konularında kullanılabilir.

Örneğin veri tabanında bir işlemin (transaction) tamamlanmasının bölünemez olması gerekir. Yani basit bir örnekle bir para transferi bir hesabın değerinin artması ve diğer hesabın değerinin azalmasıdır (havale yapılan kaynak hesaptan havale yapılan hedef hesaba doğru paranın yer değiştirmesi) bu sıradaki işlemlerin

bölünmeden tamamlanması (atomik olması) gerekir ve bir hesaptan para eksildikten sonra, diğer hesaba para eklenmeden araya başka işlem giremez.

Örneğin bu işlem sırasında bir hesaptan para azaltıldıktan sonra diğer hesabın arttırılmadan işlemin yarıda kesilmesi durumunda paranın kaybolması söz konusu olabilir.

5.2.1.2. Tutarlılık (Consistency)

Bu özelliğin amacı, HİS üzerinde yapılan işlemler ile verinin bütünlüğünü sağlamaktır.

Örneğin veri tabanında çalışanların çalıştıkları departmanları tuttuğumuz bir tablo olsun.

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	Yazılım
Ayşe Er	21	Muhasebe
Veli Demir	43	Yönetim

Ayrıca departman tablosunda da bütün departmanları tutalım.

Kısım	Dahili	Yöneticisi
Bilgisayar	111	Ayşe Er
Muhasebe	112	Ahmet Demir

Yukarıdaki iki tablo arasında çok sayıda tutarsızlık (inconsistency) bulunmaktadır. Örneğin "Veli Demir"'in çalıştığı departman olan "Yönetim" departmanı bulunmamaktadır. Benzer şekilde "Muhasebe" departmanının yöneticisi "Ahmet Demir" diye bir çalışan yoktur. Ayrıca, "Bilgisayar" kısmının yöneticisi olan "Ayşe Er" bu kısımda değil "Muhasebe"de çalışmaktadır.

Yukarıdaki bu tespitlerin veri tabanında olmasını istiyor da olabiliriz. Ancak istenmemesi durumunda, yukarıdaki şartların sağlanması için HİS üzerinde bazı tanımlamalar yapılması gerekir. Bunlar aslında HİS üzerindeki kısıtlardır. Örneğin yeni bir çalışan eklenirken, kısım bilgisine sadece mevcut kısımlardan birisinin yazılabilmesi, HİS üzerinde bir işlemin kısıtlanması anlamına gelmektedir.

5.2.1.3. Yalıtılmışlık (Isolation)

Bu özellik, bir HİS sisteminin aynı anda birden fazla işlemi birbirini etkilemeden çalıştırabilmesi anlamına gelmektedir. Örneğin bir banka üzerinde yapılan çok sayıdaki işlemin birbirini beklemeden çalışması gerekir. Diyelim ki Ankara'dan yapılan bir işlemin İstanbul'daki bütün işlemlerin bitmesini beklemesi oldukça başarısız bir tasarım olacaktır. Ancak bazen bütün işlemler diğer işlemlerden yalıtılmış olamaz. Mesela aynı kişinin hesabına para yatırma ve çekme işlemleri yapılırken, kişinin hesabının eksi olamayacağı düşünülürse, önce yatırma sonra çekme işleminin yapılması gerekir. Bu durumda ardışık işlemlerin tespit edilmesi gerekir.

5.2.1.4. Devamlılık (Durability)

Bu özelliğin amacı, sistemin beklenmeyen durumlarda bile devamlılığının sağlanmasıdır. Örneğin sistemin çökmesi halinde HİS üzerindeki kayıtlar kullanılarak bir veri tabanı sisteminin geri getirilmesi (recovery) mümkündür. Aynı zamanda bu sistemin üzerindeki tamamlanmış işlemlerin tamamının geri getirilmiş olması ve yarım kalan işlemlerin ise sistemde bulunmaması beklenir.

5.2.2. OLTP (ÇİHİ)

HİS üzerinde yapılan gelişmeler sonucunda canlı (online) bir yaklaşım getirilmesi ile ortaya çıkan sistemlerdir. Çevrim İçi Hareket İzleme (ÇİHİ) anlamına gelen İngilizce "online transaction processing" kelimelerinin baş harflerinden oluşur.

Amaç 7 gün 24 saat erişilebilen bir hareket işleme sistemi bulundurmaktır. Ayrıca hareket işlem sistemlerinde karşılaşılan problemlerin anlık olarak çözülmesi ve sistemin sürekli işleme hazır tutulması amaçlanır. Bu amaç doğal olarak ilave maliyet getirmekte ve sistemin bakım ve işletim maliyetlerinin HİS'e göre daha yüksek olması anlamına gelmektedir.

Temel olarak bir OLTP (ÇİHİ) sistem ile TPS (HİS) arasındaki fark, OTLP tarafından anlık (online) olarak yapılan işlemlerin, TPS tarafından yığın (batch) işlem olarak yapılmasıdır. Bu anlamda bir yığın (batch) işlemin daha basit geliştirildiği, kodlandığı ve yürütüldüğünü söylemek mümkündür. Ancak işlemlerin yığınlar halinde tutulması herhangi bir geri alma durumunda bütün işlemlerin baştan itibaren işlenmesi ve bunun da

çok uzun zamana mal olması anlamına gelir.

Bu fark doğal olarak iş akışında da bir farklılığı beraberinde getirir. Örneğin HİS (TPS) sistemlerindeki temel işlem akışı yaklaşımı aşağıdaki adımlardan oluşur:

- Verinin Yakalanması (Data Capture)
- Verinin Doğrulanması (Data Validation)
- Verinin İşlenmesi (Data Processing)
- İşlemin Arşivlenmesi (Archieve Transaction)
- Raporların ve Dokümanların oluşturulması

Buna karşılık ÇİHİ (OLTP) sistemlerinde yukarıdaki adımların sırası beklenmeksizin anlık olarak işlemlerden herhangi birisi işletilebilir.

5.2.3. OLAP (ÇİAİ)

İngilizce çevrim içi analitik işlem anlamına gelen Online Analytical Processing (OLAP) sistemleri, temel olarak analiz seviyesinde çalışmak için tasarlanmıştır. Yani teknik alt yapının ve işlem hareketliliğinin OLTP (ÇİHİ) sistemi tarafından tutulduğu bir ortamda analitik işlemler için daha üst katmanda OLAP (ÇİAİ) çalışabilir. Diğer bir deyişle, iş analizi ve iş süreçlerinin takibi için ilave bir katman olarak OLAP (ÇİAİ) sistemleri geliştirilmiştir.

Buradaki amaç analiz işlemlerinin anlık olarak (online) yapılıyor olmasıdır. İş analizleri genelde karmaşık sorgulamalar üzerine kuruludur. Örneğin her yılın ilk ayında bir şubedeki bir ürünün maliyet fiyatı ve satışı arasındaki farkların getirilmesi gibi karmaşık sorgular analiz aşamasında beklenebilir. Hareket işlem sistemlerinde ise sorgular genelde basit düzeydedir. OLAP (ÇİAİ) bu karmaşık sorguları, daha altta çalışan OLTP

(ÇİHİ) katmanına dönüştürmek ve gerekirse sürekli güncellemeleri almakla sorumludur.

Sorguların karmaşık hale gelmesi, işlemlerin ve çalışmanın da uzaması anlamına gelmektedir. Bu yüzden bazı iş yerlerinde OLAP (ÇİAİ) sistemleri için ilave donanım hatta ilave personel istihdamı bile yapılmaktadır.

OLTP (ÇİHİ) sistemlerinin aksine, OLAP (ÇİAİ) sistemleri çok sayıdaki farklı veri kaynağından verileri toparlayarak işlemek zorunda kaldığı için, hafıza ihtiyaçları işledikleri veriye göre çok daha fazladır.

İşlemlerindeki karmaşıklıklar göz önüne alındığında, OLAP (ÇİAİ) sistemleri çoğu zaman yedeklenmeden sadece istenen rapor ve doküman detaylarını tutarlar. Zaten sistemin bir şekilde bozulması durumunda yeniden kurulması ve eski tutulan verilerin geri getirilmesinin çoğu zaman anlamı olmayacaktır. Sistem canlı (online) çalıştığı için gerekli olan kural ve sorguların yeniden yüklenmesi yeterli olacaktır.

5.3. Normalleştirme (Normalization)

İstatistiksel normalleştirme, özellikle, veri madenciliği (data mining) gibi bilgisayar bilimlerinin istatistiksel veri işleme alanlarında kullanılan bir yöntemdir. Yöntemin amacı, veriler arasında farklılığın çok fazla olduğu durumlarda verileri tek bir düzen içerisinde ele almaktır.

Diğer bir kullanılışı ise farklı ölçekleme sisteminde bulunan verilerin birbiri ile karşılaştırılabilmesidir. Buradaki amaç, matematiksel fonksiyonlar kullanarak, farklı sistemlerde bulunan verileri ortak bir sisteme

taşımak ve karşılaştırılabilir hale getirmektir.

Bu anlamda kullanılan normalleştirme fonksiyonları (normalization functions) aşağıda açıklanmıştır.

5.3.1. Asgari – Azami Normalleştirmesi (Min-Max Normalisation)

Bu yöntemde, bir grup verinin içerisindeki en büyük ve en küçük değerler ele alınır. Diğer bütün veriler bu değerlere göre normalleştirilir. Buradaki amaç en küçük değeri 0 ve en büyük değeri 1 olacak şekilde normalleştirmek ve diğer bütün verileri bu 0-1 aralığına yaymaktır.

Yukarıda verilen formüle göre her değerin normalleştirilmiş değeri hesaplanır. Bu hesaplamanın çalışmasını bir örnek sayı dizisi üzerinde gösterelim. Örneğin sayı dizimi aşağıdaki şekilde verilmiş olsun:

5,8,9,11,20,22,24,25,27,29

Yukarıdaki sayıların normalleştirilmiş halleri aşağıda verilmiştir:

5	0
8	0,125
9	0,166666667
11	0,25
20	0,625
22	0,708333333
24	0,791666667
25	0,833333333
29	1

Yukarıda görüldüğü üzere en küçük sayı 0 ve en büyük sayı 1 olarak normalleştirilmiştir. Bu sayılar dışındaki

sayılara ise yer aldıkları aralığa göre değerler atanır. Örneğin sayı dizimi aşağıdaki şekilde olsaydı:

10,15,16,17,50,51,55,56,60

Bu durumda normalleştirilmiş değerler aşağıdaki şekilde olacaktı:

10	0
15	0,1
16	0,12
17	0,14
50	0,8
51	0,82
55	0,9
56	0,92
60	1

Görüldüğü üzere bu yeni normalleştirme sonucunda da sayılarımız 0 ile 1 arasında değerler aldılar. Şimdi bu iki farklı sayı grubunu karşılaştırabiliriz. Örneğin ilk sayı dizimizde bulunan 25, normalleştirme sonucunda 0,83 değerini almıştır. İkinci sayı grubumuzda bulunan 51 ise, normalleştirilmiş değer olarak 0,82 değerini almıştır. Bu duruma ilk sayı dizisinde bulunan 25'in sayılar arasındaki konumu, ikinci dizide bulunan 51 ile yakındır denilebilir (en azından ikinci dizide bulunan 17 sayısına göre daha yakındır çünkü ikinci dizideki 17 sayısı 0,14 gibi çok daha farklı bir değerdir).

Örnekte de görüldüğü üzere iki farklı dizide bulunan sayılar birbiri ile karşılaştırılmak istendiğinde, ortak bir sayı sistemine çevrilerek içinde bulundukları diğer

sayılara göre göreceli olarak bir değer atanmış ve karşılaştırılmıştır. İşte normalleştirme tam olarak budur.

5.3.2. Standart Skor (Standard Score):

Diğer bir normalleştirme yöntemidir. Bir önceki yöntemde, sayılar en yüksek ve en düşük değerlere göre normalleştirilmişti. Bu yöntemde ise ortalama değer (mean value) ve standart sapma (standard deviation) değerleri göz önüne alınır. Sistemde kullanılan standart sapmaya atfen, standart skor (standard score) olarak da isimlendirilir. Oldukça popüler normalleştirme yöntemlerinden birisidir. Formülü aşağıdaki şekilde yazılabilir:

Yukarıdaki denklemden de anlaşılacağı üzere, bir değerin normalleştirilmesi sırasında, ortalama değere (μ) olan uzaklığı alınarak standart sapma değerine (σ) bölünür. Konunun anlaşılması için, bir önceki örnekte kullandığımız sayı dizisini yeniden işlemeye çalışalım:

5	-1,367526918
8	-1,025645188
9	-0,911684612
11	-0,683763459
20	0,341881729
22	0,569802882
24	0,797724035
25	0,911684612
29	1,367526918

Yukarıdaki değerleri inceleyecek olursak, 5 ve 29 sayılarının normalleştirme sonucunun mutlak değeri

aynı çıkmıştır. Bunun sebebi, iki sayının ortalama değer olan 17'ye olan uzaklıklarının eşit olmasıdır (5-17 = -12 ve 29-17 = 12) sadece yönleri farklıdır ve bu durum normalleştirme sonucunda görülmüştür.

Gelelim ikinci dizimizin normalleştirme sonucuna:

10	-1,251528679
15	-1,016867051
16	-0,969934726
17	-0,9230024
50	0,625764339
51	0,672696665
55	0,860425967
56	0,907358292
60	1,095087594

Bir önceki normalleştirme sonucunda en büyük ve en küçük sayıların sadece yönü farklıyken bu sefer değerlerde farklılık oldu. Bunun sebebi iki sayının ortalama değere olan uzaklığının farklılaşmasıdır. Bu ikinci dizideki ortalama değer 36.667 olduğu için iki değer farklı olarak normalleştirilmiştir.

Bir önceki normalleştirme olan asgari / azami normalleştirmede bu fark görülememekteydi. Diğer bir deyişle asgari / azami normalleştirme, en büyük sayıyı hep 1 ve en küçük sayıyı hep 0 olarak kabul edip, diğer sayıları bu 0-1 aralığına yerleştirmekteydi, z-skor normalleştirmesinde ise sayılar ortalama değere göre uzaklıklarına göre normalleştirilmektedir. Ayrıca standart sapmaya bölünerek, sayılar arasındaki hareketlilik

(değişim hızı) ortalamaya olan uzaklığı normalize etmektedir.

6. Veri Madenciliği (Data Mining)

Bu bölümde, veri madenciliği (data mining) kavramlarına genel bir giriş yapılacak ayrıca kullanılan sınıflandırma ve kümeleme algoritmalarının detayları açıklanacaktır.

6. Veri Madenciliği (Data Mining)

Bu aşamada, veri üzerinde klasik madencilik uygulamalarının nasıl yapıldığı açıklanacaktır. En klasik veri madenciliği problemleri aşağıda sıralanmıştır:

- Sınıflandırma (Classification)
- Kümeleme (Clustering)
- Kestirim (Prediction)
- İlişkilendirme (Correlation)
- Hata Oranları (Error Rates)

6.1. Sınıflandırma

Sınıflandırma kavramı, basitçe bir veri kümesi (data set) üzerinde tanımlı olan çeşitli sınıflar arasında veriyi dağıtmaktır. Sınıflandırma algoritmaları, verilen eğitim kümesinden bu dağılım şeklini öğrenirler ve daha sonra sınıfının belirli olmadığı test verileri geldiğinde doğru şekilde sınıflandırmaya çalışırlar.

Veri kümesi üzerinde verilen bu sınıfları belirten değerlere etiket (label) ismi verilir ve gerek eğitim gerekse test sırasında verinin sınıfının belirlenmesi için kullanılırlar.

Konunun daha kolay anlaşılabilmesi için bir örnek üzerinden anlatmaya çalışalım.

Örneğin aşağıdaki şekilde öğrencilerden toplanan bir veri kümemiz bulunsun.

Yaş	Boy	Kilo	Cinsiyet
20	175	70	Erkek
21	179	80	Erkek
19	162	50	Kız
22	169	55	Kız
20	183	90	Erkek
19	181	75	Erkek
21	171	57	Kız

Örneğin problemimizi şu şekilde tanımlayalım. Bir sınıflandırıcı yöntemimiz, yukarıdaki kümeye bakarak verilen yaş, boy ve kilo değerlerine göre bir öğrencinin cinsiyetini öğrenecek olsun. Yani yukarıdaki veri kümesini bir eğitim kümesi olarak kullanacağız. Ardından gelen yeni bir kayıt için, yaş, boy ve kilo değerleri verildiğinde, sınıflandırıcımız cinsiyetini otomatik olarak tahmin edecek (prediction).

Çok sayıdaki sınıflandırma algoritmalarından basit birini seçelim. Diyelim ki sınıflandırma algoritmamız verilen etiketteki değerlerin ortalamasını alacak ve bu ortalama değer öğrendiği değer olacak. Ardından gelen

test değerleri için bulmuş olduğu ortalamaya uzaklığına bakacak ve kime yakınsa o etiketten kabul edecek.

Yukarıdaki veri kümesini iki sınıf için ikiye bölelim ve ortalama değerlerini alalım:

Erkekler için öğrenme işlemimiz aşağıda verilmiştir:

Yaş	Boy	Kilo	Cinsiyet
20	175	70	Erkek
21	179	80	Erkek
20	183	90	Erkek
19	181	75	Erkek
20	179,5	78,75	Ortalama

Aynı işlemin kızlar için olanı aşağıdaki şekildedir:

Yaş	Boy	Kilo	Cinsiyet
21	171	57	Kız
19	162	50	Kız
22	169	55	Kız
20,66666667	167,3333333	54	Ortalama

Sonuç olarak algoritmamız erkekler için (20,179.5, 78.5) değerlerini öğrenirken, kızlar için (20.66, 167.33, 54) değerlerini öğreniyor. Diyelim ki yeni gelen ve test edilmesini istediğimiz değer de, yaş: 21, boy: 165, kilo 60 değerlerinde bir kişi olsun. Şimdi algoritmamız öğrendiği değerlere göre bu yeni gelen kişinin cinsiyetini tahmin etmeye çalışacak. Basitçe her değere olan mesafeyi hesaplayacak (burada da çok farklı mesafe hesap-

lama algoritmaları olmasına karşın, biz yine, amacımız temel kavramlar olduğu için, konuyu basit tutarak gauss mesafesini (gaussian distance) kullanalım).

Algoritmamızın Erkek tanımından öğrendiği değerler ile yeni gelen kişinin mesafesini hesaplayalım:

$$\sqrt{((20-21)^2+(179.5-165)^2+(78.75-60)^2)}=23.72$$

Benzer şekilde kızlar için öğrendiğimiz değere olan mesafesini hesaplamaya çalışalım.

$$\sqrt{((20.66-21)^2+(167.33-165)^2+(54-60)^2)}=6.44$$

Buna göre algoritmamızın verdiği değer, erkeklere olan mesafesinin 23.72 olduğu ve kızlara olan mesafesinin 6.44 olduğudur. Demek ki algoritmamız yeni gelen kişiyi kız sınıfından tanımlamıştır.

İşte sınıflandırma algoritmalarının çalışması, yukarıda da anlatıldığı gibi en basit anlamıyla 2 aşamadan oluşur.

- Eğitim verisi üzerinden öğrenme
- Öğrenilen değerlerle test verisi üzerinde sınıflandırma

Ancak veri madenciliği ve iş zekası çalışmalarında sınıflandırma sadece çalışmanın bir türü ve ufak bir parçası olmaktadır.

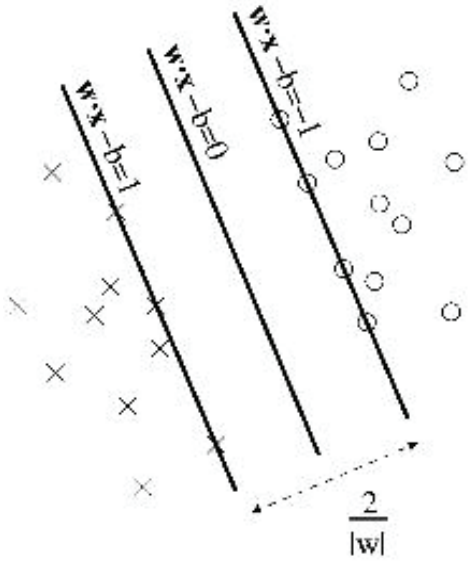
Sınıflandırma için kullanılan bazı metotlar alt bölümler olarak sunulacaktır.

6.2. Destekçi Vektör Makineleri (Support Vector Machine, SVM)

Sınıflandırma (Classification) konusunda kullanılan

oldukça etkili ve basit yöntemlerden birisidir. Sınıflandırma için bir düzlemde bulunan iki grup arasında bir sınır çizilerek iki grubu ayırmak mümkündür. Bu sınırın çizileceği yer ise iki grubun da üyelerine en uzak olan yer olmalıdır. İşte SVM bu sınırın nasıl çizileceğini belirler.

Bu işlemin yapılması için iki gruba da yakın ve birbirine paralel iki sınır çizgisi çizilir ve bu sınır çizgileri birbirine yaklaştırılarak ortak sınır çizgisi üretilir. Örneğin aşağıdaki şekildeki iki grubu ele alalım:



Bu şekilde iki grup iki boyutlu bir düzlem üzerinde gösterilmiştir. Bu düzlemi ve boyutları birer özellik olarak düşünmek mümkündür. Yani basit anlamda sisteme giren her girdinin (input) bir özellik çıkarımı (feature

extraction) yapılmış ve sonuçta bu iki boyutlu düzlemde her girdiyi gösteren farklı bir nokta elde edilmiştir. Bu noktaların sınıflandırılması demek, çıkarılmış olan özelliklere göre girdilerin sınıflanması demektir.

Yukarıda her iki sınıf arasında oluşan aralığa tolerans (offset) demek mümkündür. Bu düzlemdeki her bir noktanın tanımı aşağıdaki gösterim ile yapılabilir:

$$\mathcal{D} = \{(\mathbf{x}_i, c_i)|\mathbf{x}_i \in \mathbb{R}^p, c_i \in \{-1, 1\}\}_{i=1}^n$$

Yukarıdaki gösterimi şu şekilde okumak mümkündür. Her x,c ikilisi için X, vektör uzayımızdaki bir nokta, ve c ise bu noktanın -1 veya +1 olduğunu gösteren değeridir. Bu noktalar kümesi i= 1'den n'e kadar gitmektedir.

Yani bu gösterim bir önceki şekilde olan noktaları ifade etmektedir.

Bu gösterimin bir aşırı düzlem (hyperplane) üzerinde olduğunu düşünürsek. Bu gösterimdeki her noktanın:

wx – b = 0

denklemi ile ifade edilmesi mümkündür. Buradaki w aşırıdüzleme dik olan normal vektörü ve x noktanın değişen parametresi ve b ise kayma oranıdır. Bu denklemi klasik ax+b doğru denklemine benzetmek mümkündür.

Yine yukarıdaki denkleme göre b/||w|| değeri bize iki grup arasındaki mesafe farkını verir. Bu mesafe farkına daha önce tolerans (offset) ismini de vermiştik. Bu mesafe farkı denklemine göre mesafeyi en yüksek değere çıkarmak için yukarıdaki ilk şekilde gösterilen 0, -1 ve +1 değerlerine sahip 3 doğruyu veren denklemde 2/||w|| formülü kullanılmıştır. Yani doğrular arası mesafe 2 birim olarak belirlenmiştir.

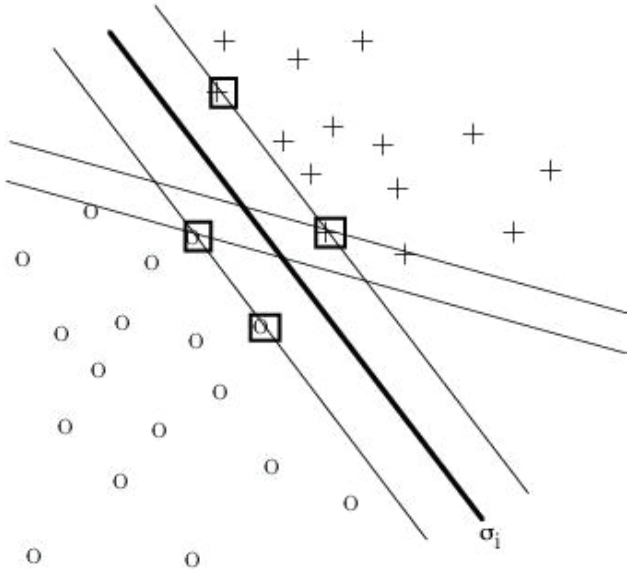
Bu denkleme göre elde edilen iki doğru denklemi:

wx – b = -1

wx + b = 1

olarak bulunmuştur. Aslında bu denklemler doğruların kaydırılması sonucunda elde edilen en yüksek değerlerin bulunması işleminin bir sonucudur. Aynı zamanda bu denklemlerle problemin doğrusal ayrılabilir (linearly seperable) olduğu da kabul edilmiş olur.

Tahmin edileceği üzere iki grup arasındaki aşırıdüzlemin (hyperplane) tek yönlü olması mümkün değildir. Aşağıda bu duruma bir örnek gösterilmiştir:

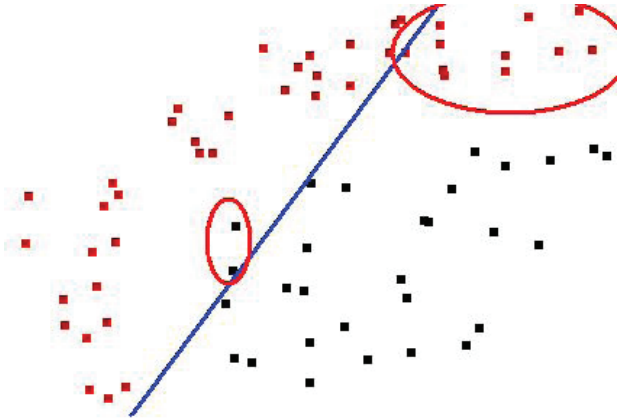


Yukarıdaki şekilde iki farklı hiperdüzlem (aşırıdüzlem) olasılığı bulunmasına karşılık SVM yönteminde

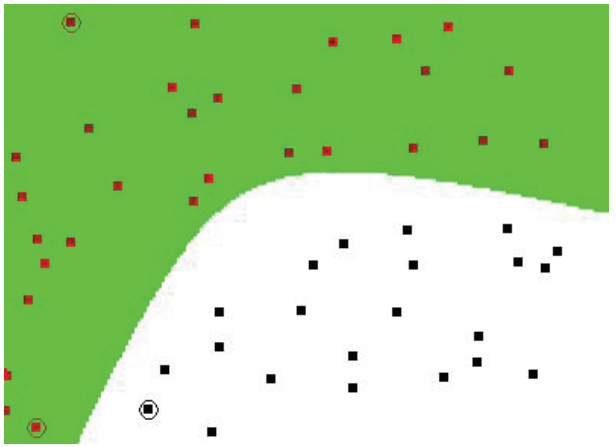
bu olasılıklardan en büyük toleransa (offset) sahip olanı alınır.

6.3. Doğrusal Olmayan Destekçi Vektör Makineleri

Destekçi vektör makinelerinde (Support vector machines) ayrım bir aşırıdüzlem (hyperplane) üzerinde iki grubu ayıran bir doğru ve bu doğruya bağlı bir tolerans (offset) ile yapılmaktadır. Ancak bu durum her zaman beklendiği kadar iyi sonuç vermeyebilir. Örneğin aşağıdaki şekilde doğrusal bir ayrım yapılması durumunda yanlış sınıflandırma yapılan örnekler (misclassification) biraz daha dikkatli ve doğrusal olmayan ikinci şekildeki gibi ayrılırlarsa tam olarak sınıflandırılabileceklerdir.



Yukarıdaki şekilde verilen örnekler aşağıdaki şekilde doğrusal olmayan bir yolla sınıflandırılabilirler:



Bu sınıflandırmanın yapılabilmesi için çekirdek hilesi (kernel trick) adı verilen bir yöntem kullanılabilir.

6.4. Naif Bayes (Naive Bayes) Sınıflandırıcısı

Herhangi bir sınıflandırma probleminde olduğu gibi, amacımız birden fazla özelliği taşıyan bir yöney (vektör) kullanarak verilen bilgilerden bir eğitim oluşturmak ve bu eğitim neticesinde gelen yeni verileri doğru bir şekilde sınıflandırmaktır.

Sınıflandırma işlemi sırasında nasıl bir yol izlendiğini bir örnek üzerinden açıklamaya çalışalım.

Örneğimiz, aşağıda verilen tablodaki verilerden bir sınıflandırıcı eğitimi olsun (*train*).

Kısım	Maaş	Yaş	İş Tecrübesi
Yazılım	3000	26	4
Muhasebe	1500	22	2
Yazılım	5000	30	9
Muhasebe	2000	30	7
Muhasebe	500	18	3
Yazılım	2000	20	2
Yazılım	7000	29	5
Muhasebe	6000	45	15

Yukarıda, sadece muhasebe ve yazılım kısımlarında çalışan kişilerin maaş, yaş ve iş tecrübelerini içeren temsili bir tablo verilmiştir. Buna göre aşağıdaki şekilde bize bir bilgi verilse:

Maaş: 3000, Yaş: 30, Tecrübe: 5 yıl

Bu kişinin hangi kısımda çalıştığını acaba bulabilir miyiz?

Öncelikle eğitim ile işe başlayalım, sonra da testimizi yaparız. Basitçe veri kümemizdeki (data set) Yazılım ve Muhasebe kısımlarının ortalama ve varyans değerlerini hesaplıyoruz. Bu değerler aşağıdaki şekildedir:

Muhasebe	2500	28,75	6,75
Yazılım	4250	26,25	5
Muhasebe	5833333,333	142,25	34,91666667
Yazılım	4916666,667	20,25	8,666666667

Yukarıdaki ilk iki satır ortalama ve ikinci iki satır ise varyans değerleridir. Bu değerleri basitçe Excel ile hesapladık.

Varyans kavramı aslında olasılıktaki birkaç konuyu birlikte bilmeyi gerektiriyor. Basitçe tanımlamak gerekirse kare sapması (squared deviation) olarak tanımlanabilir. Basit bir örnek üzerinden konuyu anlatmaya çalışalım. Öncelikle sayısal olarak ifade edilebilecek bir rastsal süreç (stochastic process) bulmamız gerekiyor. Burada zar atma örneği oldukça iyi bir örnektir ve 6 yüzü olan zarın her yüzü için bir değer belirleyebiliriz. Buna göre zar atma işleminden sonra gelebilecek sayılar 1, 2, 3, 4, 5, 6 olabilir.

Şimdi beklenen değeri (Excepted value) hesaplamak için aşağıdaki şekilde önce bütün yüzlerin alabileceği değerleri toplayalım:

1+2+3+4+5+6 = 21 bu değeri 6 farklı ihtimale bölelim: 21 / 6 = 3.5 değeri zar atma işleminin (process) çok fazla tekrarlaması sonucunda elde edilmesi beklenen değerdir. Örneğin bir milyon kere zar atsak ve gelen değerleri toplayıp attığımız zar sayısına bölsek, 3.5 değeri çıkmasını bekleriz.

Şimdi bu zar olayının mutlak sapmasını (absolute deviation) hesaplayalım. Bunun için her değerin ayrı ayrı beklenen değere (expected value) olan uzaklığını hesaplamamız gerekiyor (sapmasını).

|1-3.5| + |2-3.5| +|3-3.5| +|4-3.5| +|5-3.5| +|6-3.5| = 2.5 + 1.5 + 0.5 + 0.5 + 1.5 + 2.5 = 9 olarak bulunur. Burada mutlak değer alma sebebimiz

yaptığımız hesabın mutlak sapma hesabı olmasıdır. Mutlak sapma hesabını tamamlayabilmek için son adımda yine hesabını yaptığımız 6 ihtimale bölmemiz gerekir: 9 / 6 = 1.5 olarak mutlak sapmayı hesaplamış oluruz.

Benzer şekilde kare sapmasını (squared deviation) hesaplamak istersek yukarıda bulduğumuz değerlerin karelerinin 6 ihtimale bölünmesi gerekir:

2.5 2 + 1.5 2 + 0.5 2 + 0.5 2 + 1.5 2 + 2.5 2 = 17.5 / 6 = 2.9 yaklaşık değeri hesaplanır.

Bu hesaplanan değer (kare sapması, squared deviation) aslında literatürde varyans (variance) olarak geçen kavramdır.

Yukarıdaki örnek üzerinden basitleştirerek anlattığımız kavramı aşağıdaki şekilde tanımlamamız mümkündür.

Var(X) = E[(X-μ) 2].

Yukarıdaki bu formülde görülen E beklendik değeri (Expected value), X her bir rastsal sürecin (random process) değerini ve μ ise ortalama değeri belirtmektedir. Aslında bu formülde yazılan, yukarıdaki örnekte hesaplanan değerlerden farklı değildir.

Bu eşitlik aşağıdaki adımlarla açılabilir:

Var(X) = E[X2 – 2μX + μ2]

Var(X) = E[X2] – 2μE[X] +μ2

Var(X) = E[X2] – 2μ2 + μ2

Var(X) = E[X2] – μ2

Var(X) = E[X2] – (E[X])2

Ayrıca standart sapma (standart deviation) kavramı, varyansın karekköküdür (veya varyans, standart sapmanın karesidir)

Şimdi beklenen değeri hesaplayacağız. Yani gelen test verimizin Muhasebe kısmında birisine ait olması veya Yazılım kısmından birisine ait olması için beklenen durum hesabı yapacağız.

Hesaplamaya geçmeden önce yazının konusu olan naive bayes kavramını hızlıca açıklayalım. Aslında naive bayes sınıflandırıcısı basitçe bütün koşullu olasılıkların çarpımıdır.

Aşağıdaki şekilde gösterilebilir:

$$sınıflandırma(s_1, s_2, \ldots, s_n) = azami_c\, p(K=k) \prod_{i=1}^{n} p(S_i = s_i | K = k)$$

Bu formülde görüleceği üzere s1'den sn'e kadar olan sınıflar arasından bir seçim yapılırken aslında bu sınıfın olasılık değeri ve bu sınıfları yerine getiren k koşulları için çarpımından bir farkı yoktur.

Yani diğer bir deyişle her sınıfın bir koşullu olasılık değeri vardır ve biz sınıflardan hangisine ait olduğunu bulmak için bu koşullu olasılık değerlerini çarparız.

Bu durumda bizim formülümüz aşağıdaki şekildedir denilebilir:

$$beklenti(Yazılım) = \frac{P(Yazılım)\, p(maaş|yazılım)\, p(Yaş|yazılım)\, p(iş\ tecrübesi|yazılım)}{normalleştirme}$$

Yani bir kişinin yazılım kısmında olduğunu anlamak için öncelikle yazılım kısmında olan kişilerin oranını buluyoruz, P(Yazılım), ardından yazılım kısmındaki kişiler için verilen maaş, yaş ve iş tecrübesine göre koşullu olasılıklarını bulup bunu normalleştirme değerine bölüyoruz.

Benzer şekilde muhasebe kısmındaki kişilere ait beklenti de aşağıdaki gibi hesaplanabilir.

$$beklenti\,(Muhasebe) = \frac{P(Muhasebe)\,p(maaş|Muhasebe)\,p(Yaş|Muhasebe)\,p(iş\ tecrübesi|Muhasebe)}{normalleştirme}$$

Buradaki fark, verilen bilgilerin muhasebe kısmındaki kişilere göre koşullu olasılığının alınmasıdır.

Normalleştirme değeri ise sistemimizde bulunan bütün ihtimalleri içeren ve beklenti değerlerini normalleştiren değerdir. Aşağıdaki şekilde hesaplanabilir:

normalleştirme = P(Muhasebe) p(maaş | Muhasebe) p (Yaş | Muhasebe) p (iş tecrübesi | Muhasebe) + {P(Yazılım) p(maaş | yazılım) p (Yaş | yazılım) p (iş tecrübesi | yazılım)

Şimdi iki sınıfımız olduğu ve aslında ikisi arasında seçim yapacağımız için, iki sınıfı da bölen ve pozitif çıkmasını beklediğimiz normalleştirme değerini göz ardı edebiliriz.

Sırasıyla olasılıkları hesaplayalım:

P(Yazılım) = 8 kişiden 4'ü yazılım kısmında

P(Yazılım) = 0.5

Benzer şekilde,

P(Muhasebe) = 0.5

olarak buluruz. Ardından koşullu olasılık değerlerini hesaplayacağız. Bu aşamada gauss dağılımını kullanmak isteyelim (farklı dağılımlar kullanılabilir ve başarı bu dağılıma göre değişebilir) ve dağılım fonksiyonunu hatırlayalım:

$$p(maaş|yazılım) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

varyans, maaş ve ortalama değerlerini yazacak olursak:

$$\frac{1}{\sqrt{2\pi 4.91E6^2}}\exp\left(\frac{-(3000-4250)^2}{24.91E6^2}\right) = \frac{1}{1.46E7}\exp\frac{(1250^2)}{4.82E13} = 6.84E\text{-}8$$

Yani sonuç olarak 0.0000000687'e yakın bir değer bulunuyor. Bunun anlamı, verilen 3000 lira maaş ile çalışan kişinin gauss dağılımında (dağılımın toplam alanının 1 olduğunu hatırlayınız), yazılım kısmında çalışan birisi olması ihtimalinin 6.87E-7 olduğudur. Daha doğru bir ifadeyle, bir kişinin yazılımcı olduğunu kabul edersek (verilen koşul) bu kişinin 3000 lira maaş alma durumunu hesaplamış olduk. Şimdi aynı maaş değerine sahip kişinin muhasebe kısmında çalışma olasılığını hesaplayalım.

$$\frac{1}{\sqrt{2\pi 5.83E6^2}}\exp\left(\frac{-(3000-2500)^2}{25.83E6^2}\right) = \frac{1}{4{,}91E7}\exp\frac{(-500^2)}{6{,}79E13} = 8{,}11E\text{-}8$$

Yukarıdaki ikinci hesaplamanın sonucuna bakarak aslında bu maaşın, yazılım kısmındaki kişilere daha uygun olduğunu söyleyebiliriz.

Benzer hesaplamaları diğer koşullu olasılık durumları için de yaparsak aşağıdaki gibi bir tablo elde edebiliriz:

	Maaş	Yaş	Tecrübe
Muhasebe	6,84074E-08	0,002805118	0,011414151
Yazılım	8,11614E-08	0,019705849	0,046043474

Sonuçta naive bayes yöntemine göre bu verilen olasılıkların çarpımlarını alacağız:

$$beklenti\,(Yazılım)=\frac{P(Yazılım)\,p(maaş|yazılım)\,p(Yaş|yazılım)\,p(iş\ tecrübesi|yazılım)}{normalleştirme}$$

olduğunu hatırlayalım. Bu durumda beklenti(yazılım) aşağıdaki şekilde yazılabilir:

$$beklenti\,(Yazılım)=\frac{0.5\,x\,6{,}84\text{E-}8\,x\,0.0028\,x\,0{,}0114}{normalleştirme}=1{,}9\text{E-}12$$

Burada normalleştirme değeri daha önce de belirtildiği üzere hesaba katılmamıştır. Benzer şekilde muhasebe kısmına ait beklenti de aşağıdaki şekilde hesaplanabilir.

$$beklenti\,(Muhasebe)=\frac{0.5\,x\,8{,}11\text{E-}8\,x\,0.0019\,x\,0{,}046}{normalleştirme}=3{,}68\text{E-}11$$

Görüldüğü üzere muhasebe beklentisi, yazılım beklentisinin yaklaşık 20 misli daha yüksek çıkmıştır. Demek ki naive bayes sınıflandırmasına göre bu kişinin muhasebe kısmında çalıştığını söyleyebiliriz, en azından beklentimiz bu yönde olur.

Naif Bayes Metin Sınıflandırıcısı (Naive Bayes Text Classifier)

Oldukça basit ve etkili bir metin madenciliği yöntemi olan naive bayes sınıflandırıcısını anlamak için bir örnek kullanalım.

Örneğin iki metin aşağıdaki şekilde verilmiş olsun:

Metin 1: Java bazı bilgisayar mühendisliği bölümlerinde eğitimi verilen bir programlama dilidir.

Metin 2: Bazı bilgisayar mühendisliği projelerinde Java programlama dili kullanılmaktadır.

Bu metinlerin, kavram metin masfufu (term document matrix) daha önceki bir yazıda gösterilmiştir. Şimdi amacımız yeni gelen aşağıdaki metnin hangisine daha yakın olduğunu bulmak olsun.

Metin 3: Bilgisayar kavramları eğitimi

Problemi çözmeye başlamadan önce Naive Bayes yönteminin klasik formülünü hatırlayalım:

$$P(A|B)=\frac{p(A\cap B)}{p(B)}$$

Bu klasik formülde bir koşullu olasılığın, nasıl küme olasılıklarına indirgendiği gösterilmektedir.

Metin madenciliğinde (text mining) önemli konulardan birisi de metin içerisinden özelliklerin nasıl çıkarılacağıdır. Bu yazı kapsamında, konuyu dağıtmamak için en basit yöntem olarak kelime seviyesinde çalışacağız. Yani bir kelimenin bir sınıfı belirlemede kullanılmasını hedefliyoruz.

Yine olayı basitleştirmek için yukarıdaki bayes

formülünde olduğu üzere 2 sınıf var olduğunu kabul edeceğiz.

Bu durumda herhangi bir kelimenin herhangi bir sınıfta bulunma olasılığı aşağıdaki şekilde gösterilebilir:

$$P(k_i|S)$$

Yani diğer bir deyişle S sınıfı gerçekleşen olaysa, ki kelimesinin bu olayın içinde yer alma olasılığıdır.

Bu durumda naive bayes sınıflandırma yöntemini kullanırsak (lütfen ilgili yazı için tıklayın) kelime bazlı sınıflandırıcımızın aşağıdaki şekilde olduğunu söyleyebiliriz.

$$P(X|S)=\prod_i p(k_i|S)$$

Yani diğer bir deyişle, sistemimizde sınıflandırılmak istenen X metninin S sınıfında olma olasılığı, bu metindeki bütün kelimelerin S sınıfında olma olasılıklarının çarpımıdır.

Neticede elimizdeki bütün sınıflar için metin denenerek en yüksek olasılığa sahip sınıfa aittir denilebilir.

Örneğimize dönecek olursak metinlerdeki kelimeleri listeleyelim:

Terimler \ Metinler	Metin 1	Metin 2
Java	1	1
Bazı	1	1
Bilgisayar	1	1
Mühendisliği	1	1
Bölümlerinde	1	0
Eğitimi	1	0
Verilen	1	0
Bir	1	0
Programlama	1	0
Dilidir	1	0
Projelerinde	0	1
Dili	0	1
Kullanılmaktadır	0	1

Şimdi bu listeye göre Metin 1'de 9 kelime bulunuyor ve hepsinin frekansı (sıklığı) aynı, dolayısıyla bütün kelimeler için

$$p(k_i|M_1)=\frac{1}{9}$$

oranındadır denilebilir. Aynı durum Metin 2 için de geçerlidir ve 7 kelimenin tamamı için:

$$p(k_i|M_2)=\frac{1}{7}$$

değerindedir denilebilir. Neticede yeni gelen ve sınıflandırılmak istenen Metin 3 için bütün kelimelerin değerlerine bakacağız.

M3 = {bilgisayar, kavramları, eğitimi}

$$p(bilgisayar|M_1)=\frac{1}{9},$$

$$p(kavramları|M_1)=0,$$

$$p(eğitimi|M_1)=\frac{1}{9}$$

Burada sık karşılaşılan bir problem olan 0 durumunu düzeltmek için iki yöntemden birisi kullanılır, ya normalleştirme yapılarak bu değer sistemden çıkarılır, ya da çok düşük bir değer konulur. Biz normalleştirme yapalım:

$p(M_3|M_1)=\sum_{i=1}^{3} p\frac{(k_i|M_1)}{3}$ şeklinde ortalama alıyoruz.

$\frac{\frac{1}{9}+0+\frac{1}{9}}{3}=\frac{2}{27}=0.074$ olarak bulunur.

Şimdi gelelim ikinci metin için aynı hesabı yapmaya:

$$p(bilgisayar|M_2)=\frac{1}{7},$$

$$p(kavramları|M_2)=0,$$

$$p(eğitimi|M_2)=0$$

Bu durumda normalleştirilmiş değer:

$$\frac{\frac{1}{7}+0+0}{3}=\frac{1}{21}=0.0476$$

olarak bulunur. Demek ki yeni gelen Metin 3, Metin 1'e daha yakındır denilebilir. Sınıflandırmamız neticesinde bir sınıf seçilmesi istenseydi, Metin 1'in bulunduğu sınıfta olduğu söylenebilirdi.

6.5. Karar Ağacı Öğrenmesi (Decision Tree Learning)

Karar ağacı öğrenmesi (decision tree learning) yöntemi, makine öğrenmesi (machine learning) konularından birisidir. Literatürde karar ağacı öğrenmesinin alt yöntemleri olarak kabul edilebilecek sınıflandırma ağacı (classification tree) veya ilkelleştirme ağacı (regression tree, tahmin ağacı) gibi uygulamaları vardır.

Karar ağacı öğrenmesinde, bir ağaç yapısı oluşturularak ağacın yaprakları seviyesinde sınıf etiketleri ve bu yapraklara giden ve başlangıçtan çıkan kollar ile de özellikler üzerindeki işlemeler ifade edilmektedir.

Bu yazı karar ağacı olarak geçen (decision tree) konusunu değil, veri madenciliği (data mining) ve makine öğrenmesi (machine learning) konularında geçen karar ağacı öğrenmesi ile ilgilidir.

Karar ağacı öğrenmesi sırasında, öğrenilen bilgi bir ağaç üzerinde modellenir. Bu ağacın bütün iç düğümleri (interior nodes) birer girdiyi ifade eder. Örneğin aşağıda örnek bir öğrenilmiş ağaç gösterilmiştir:

Yukarıdaki şekil, kalp krizi geçiren kişilerin çeşitli özellikleri ele alınarak veriler üzerinden öğrenilmesi durumunda hayali olarak elde edilecek sonuçlardır. Diğer bir deyişle, iç düğümlerde (dikdörtgenler) çeşitli girdilere göre ağacın dallanması söz konusudur. Yapraklarda ise sonuç olarak elde edilen değerler gösterilmiştir.

Karar ağacı öğrenmesinde, ağacın öğrenilmesi sırasında, üzerinde eğitim yapılan küme, çeşitli özelliklere göre alt kümelere bölünür, bu işlem, özyineli olarak (recursive) tekrarlanır ve tekrarlama işleminin tahmin üzerinde bir etkisi kalmayana kadar sürer. Bu işleme özyineli parçalama (recursive partitioning) ismi verilir.

Genelde veri madenciliği sırasında verinin gelme şekli aşağıdaki gibidir:

(x,Y) = (x1,x2,x3,x4,…, Y)

Bu gösterime göre x1'den xn'e kadar olan değerler sistemin girdileri iken, Y değeri sistemin çıktısı olarak

elde edilmesi istenen değerdir. Örneğin yukarıdaki şekilde, kalp krizi geçirme oranı Y değeri olarak, "ailesinde kalp krizi geçirenlerin olup olmaması", "kişinin yaşı", "alkol kullanıp kullanmaması", "sigara içip içmemesi" gibi bilgiler ise sistemin girdileri olan x değerleri olarak düşünülebilir.

Veri madenciliğinde karar ağacı öğrenmesi (decision tree learning) iki temel amaç için kullanılır. Bu amaçlar ve karar ağacı öğrenmesinin bu amaca yönelik özel isimleri aşağıdaki şekildedir:

- Sınıflandırma problemleri: Sınıflandırma ağaçları (Classification Tree): Bir kişinin harcamalarından eğitim düzeyinin tahmini gibi, hedef kümeyi çeşitli sınıflardan birisine yerleştirmeyi amaçlayan ve sınıf tanımı yapan problemler.
- İlkelleme problemleri: İlkelleme ağaçları (Regression Trees): Sonuçta bir sınıf yerine sayısal bir değer döndüren veri madenciliği problemleri: Örneğin yukarıda verilen kalp krizi geçirme ihtimalleri gibi.

Ayrıca yukarıdaki iki terimi de içine alan bir çatı terim olarak CART (Classification and Regression Tree) terimi de kullanılmaktadır.

6.5.1. Karar ağacı öğrenme algoritmaları

Bu ağaçların çalıştırılması sırasında aşağıdaki bazı algoritmalardan yararlanılabilir (İnşallah vakit bulursam bunları da ayrı birer yazı halinde site üzerinden yayınlayacağım):

- Rastgele Orman (Random Forest): Sınıflandırma işlemi sırasında birden fazla karar ağacı kullanılarak sınıflandırma değerinin yükseltilmesi hedeflenir.
- Hızlandırılmış Ağaçlar (Boosted Trees): Hem sınıflandırma (classification) hem de ilkelleme (regression) problemleri için kullanılabilen bir algoritmadır.
- Döndürme Ağacı (Rotation Forest): Rastgele ağaca benzer şekilde birden fazla ağaç kullanılmaktadır ancak her ağaç, önce farklı bileşen analizi (Pricipal Component Analysis, PCA) kullanılarak eğitilmektedir. Bu eğitim için veri kümesinin rastgele seçilmiş bir alt kümesi kullanılmaktadır (sarnıçlama yöntemiyle).
- Ayrıca aşağıdaki algoritmalar da karar ağacı öğrenmesinde kullanılmaktadır:
- ID3 algoritması
- C4.5 algoritması
- Chi-Kare Otomatik İlişki Tarayıcısı (Chi-Square Automatic Interaction Detector, CHAID): Birden fazla seviyeye bölme işlemine izin veren bir sınıflandırma algoritmasıdır.
- MARS: Sayısal verilerin daha iyi işlenebilmesi için karar ağaçlarını iyileştiren bir yaklaşımdır.

6.5.2. Karar ağacı öğrenme algoritmalarının avantajları

Anlaşılması ve yorumlanması basittir. Basit bir açıklama ile karar ağacı öğrenmesinin neticelerini insanlar anlayabilirler.

- Hızlı veri ön işlemesi gerektirir: Çoğu alternatif tekniklere göre çok az bir işleme ile veri kullanılabilir hale gelir. Ön işleme aşaması diğer alternatiflerine göre daha kısa ve basittir.
- Hem sayısal hem de sınıfsal verilerin işlenmesi için kullanılabilir: Çoğu makine öğrenmesi algoritması, ya sayısal uygulamalar ya da sınıflandırma problemleri için kullanışlıdır. Karar ağacı öğrenmeleri ise iki alanda da kullanılabilir.
- Beyaz kutu modelini kullanır: Yazılım mühendisliğinin (software engineering) bir yaklaşımı olan beyaz kutu modelinde, her adım görüntülenebilir ve yorumlanabilirdir. Yine bir yazılım mühendisliği uygulaması olan kara kutu yaklaşımı (black box approach) ise makine öğrenmelerinde daha çok yapay sinir ağları (artificial neural network) tarafından karşılanır. Bu yöntemde girdi ve çıktı yorumlanabilirken sistemin iç dinamiklerinin her adımda gözlemlenmesi ve yorumlanması mümkün değildir.
- Düşük hesaplama karmaşıklığına (computational complexity) sahiptir: Basit ve hızlı olmasından dolayı yüksek miktardaki veriyi kısa sürede işleyebilir ve alternatifi yöntemlere göre veri miktarı arttığında daha da tercih edilebilir olur.

6.5.3. Yöntemin Kısıtları

İyileştirilmiş (optimum) bir karar ağacı öğrenme algoritması çoğu zaman NP-Tam (NP-Complete) karmaşıklığa sahiptir. Bu karmaşıklığından sakınıldığı için çoğu karar ağacı öğrenme algoritması, sezgisel algoritmalar (heuristic algorithms) veya açgözlü yaklaşımı (greedy approach) kullanmaktadır. Genelde bu uygulamalar, hedefe yönelik olarak, yerel olarak iyileştirilmiş (local optimum) bulmada başarılı olurken, genel iyileştirmeyi (global optimum) kaçırmaktadır.

Karar ağacı öğrenmesi ile çalışılırken kolaylıkla yapılacak bir hata, ağacı özel amaçlara yönelik olarak değiştirirken, bütün verinin özelliklerini modelleyemeyen bir ağaç ile karşılaşılabilir. Bu durum makine öğrenmesi konusunda aşırı uyum (overfitting) olarak geçmektedir. Bu durumu engellemek için budama (prunning) işlemleri yapılmaktadır.

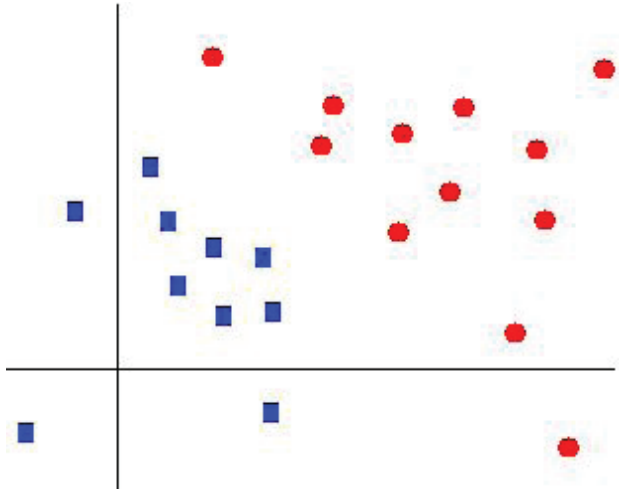
Makine öğrenmesi konularında meşhur olan ve çoğu algoritma için problem olan karmaşık konular (örneğin özel veya problemi (XOR problem), teklik problemi (parity problem) veya çoklayıcı problemi (multiplexer) problem gibi) karar ağacı öğrenmesi tarafından da modellenmesi ve çözümlenmesi zor konulardır. Bu problemlerin çözümü ya problemin tanım alanının yeniden tanımlanması (ki bu işleme haberselleştirme (propositionalisation) ismi verilir) veya algoritmanın daha açılayıcı bir gösterime dönüştürülmesi sayesinde yapılmaktadır.

Birden fazla seviyeden oluşan sınıfsal verilerin genelde diğer verilerden yüksek sonuç vermesi söz konusudur.

6.6. K- En Yakın Komşu (K-Nearest Neighborhood, KNN)

Sınıflandırmada (classification) kullanılan bu algoritmaya göre sınıflandırma sırasında çıkarılan özelliklerden (feature extraction), sınıflandırılmak istenen yeni bireyin daha önceki bireylerden k tanesine yakınlığına bakılmasıdır.

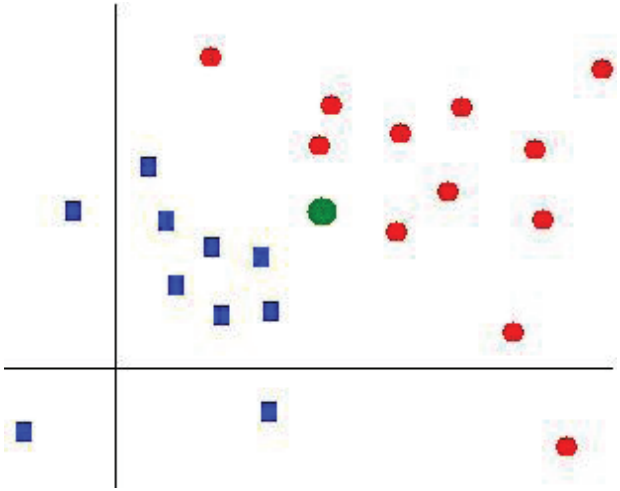
Örneğin k = 3 için yeni bir eleman sınıflandırılmak istensin, bu durumda eski sınıflandırılmış elemanlardan en yakın 3 tanesi alınır. Bu elamanlar hangi sınıfa dahilse, yeni eleman da o sınıfa dahil edilir. Mesafe hesabından genelde öklit mesafesi (euclid distance) kullanılabilir.



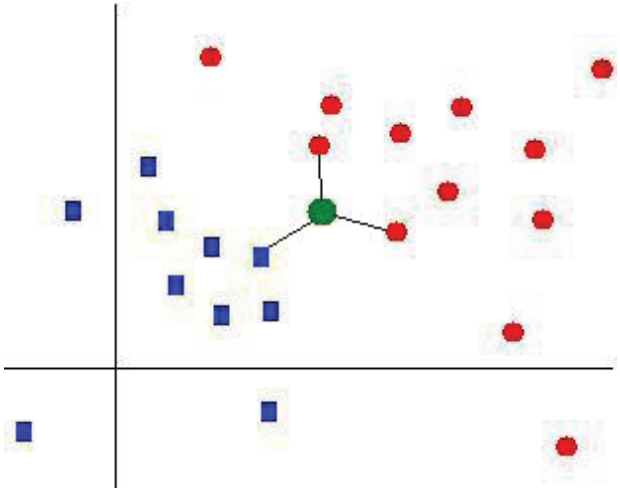
örneğin yukarıda verilen ve özelliklerine göre 2 boyutlu koordinat sistemine yerleştirilmiş olan örnekleri ele alalım. Bu örneklerin birbirinden ayrılması doğrusal

ayrıştırma (linear discrimination) problemidir ve buradaki yöntemlerle çözülür.

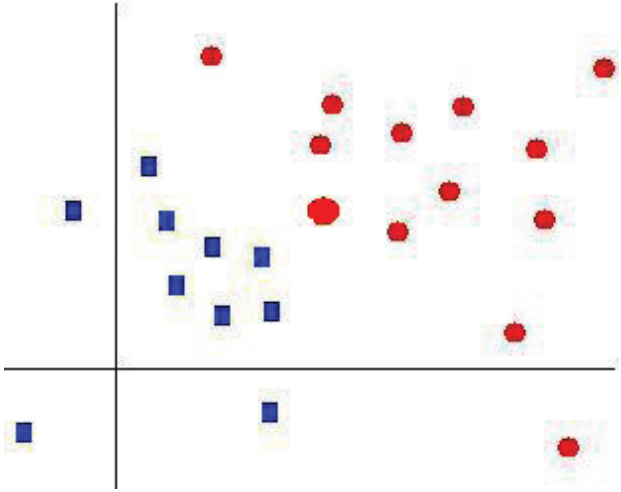
KNN yöntemine göre aşağıdaki şekilde yeni bir üyenin geldiğini düşünelim:



Yukarıdaki bu yeni gelen üyenin en yakın olduğu 3 üyeyi (3 nearest neighbors) tespit edelim.



En yakın 3 üyenin iki tanesi kırmızı yuvarlak üyeler olduğuna göre, yeni üyemizi bu şekilde sınıflandırabiliriz:



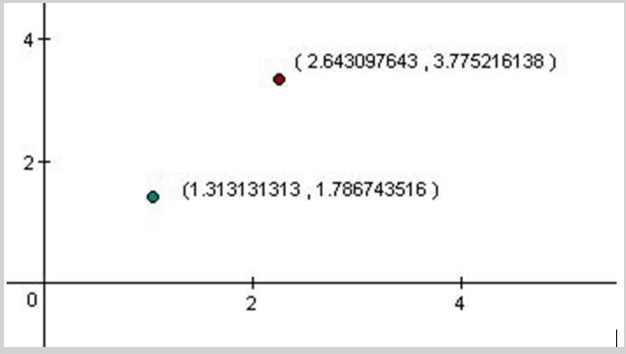
Öklit Mesafesi *(Euclidean Distance):*

Matematikte pisagor bağlantısı kullanılarak bulunan iki nokta arasındaki mesafe ölçüm birimidir. Buna göre iki boyutlu düzlemde iki nokta arasındaki mesafe basitçe iki noktanın x ve y koordinatlarının ayrı ayrı farklarının hipotenüsüne eşittir. Örneğin birinci nokta p(x,y) olsun (p noktasının koordinatları x,y) ve ikinci nokta q(s,t) (yani q noktasının koordinatları s,t olsun) bu durumda iki nokta arasındaki mesafe:

$$\sqrt{(x-s)^2+(y-r)^2}$$

olarak ifade edilebilir.

Örnek:



yukarıdaki resimde kartezyen uzay (cartesian space) üzerinde tanımlı iki nokta verilmiştir. Bu noktalar arasındaki öklid uzaklığı aşağıdaki formüldeki şekilde hesaplanabilir:

$$\sqrt{(2.643097643-1.313131313)^2-(3.775216138-1.786743516)^2}$$

Şayet uzay 3 boyutlu olursa bu defa öklit mesafesi noktaların 3 boyuttaki koordinat değerlerinin farklarının karelerinin toplamlarının karekökü olarak hesaplanır.

6.7. Kümeleme (Clustering)

Kümeleme problemleri, genel olarak çıkarılmış olan özellikler kullanılarak bir veri setindeki değerlerin (instance) farklı kümeler arasında paylaşılmasıdır. Sınıflandırma ile arasındaki fark, bir veri kümesindeki verilerin arasında ilişki olup olmadığının sorgulanmasıdır.

Örneğin bir marketten alışveriş yapan kişileri ele alalım. Bir müşterinin cinsiyetinin, çocuğunun olup olmadığının veya sigara içip içmediğinin belirlenmesi bir sınıflandırma problemidir. Burada müşteri, mevcut sınıflardan birisine konulmaya çalışılır.

Buna karşılık, müşterinin gelir düzeyinin tespiti bir kümeleme problemidir. Müşteri A gelir düzeyinde de olabilir B düzeyinde de olabilir veya bu ikisinin ortasında bir yerlerde de olabilir. Genelde kümeleme problemlerinde sürekli sonuçlar üzerinden çalışılır. Buna karşılık sınıflandırma problemlerinde değerler sabit sınıflar arasında dağıtılmıştır ve sınıflar arasındaki hatlar kesindir.

6.7.1. K-Ortalama Kümelemesi (K-means Clustering)

Kümeleme (clustering), kullanılan algoritmalardan birisidir. Amaç özellik çıkarımı (Feature extraction)

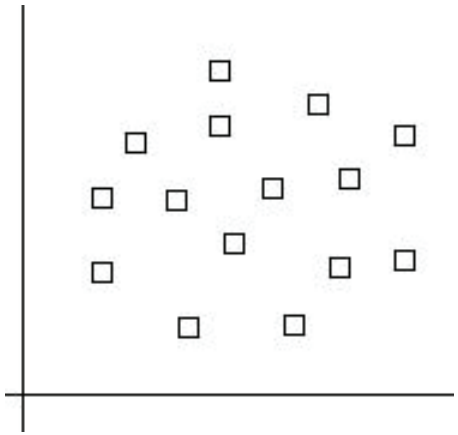
yapılmış bir grup verinin birden fazla sınıfa göre doğru sınıflandırılmasıdır.

Kullanılan matematiksel yöntem her sınıf için merkez belirlenen noktaya uzaklığa (aynı zamanda bu hata miktarıdır) göre yeni sınıfların yerleştirilmesidir.

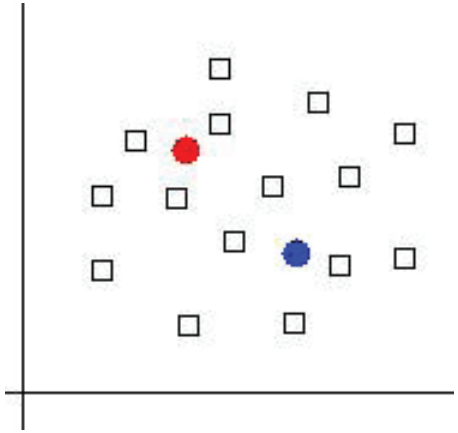
Algoritma temel olarak 4 aşamadan oluşur:

1. Sınıf merkezlerinin belirlenmesi
2. Merkez dışındaki örneklerin mesafelerine göre sınıflandırılması
3. Yapılan sınıflandırmaya göre yeni merkezlerin belirlenmesi (veya eski merkezlerin yeni merkeze kaydırılması)
4. Kararlı hale (stable state) gelinene kadar 2. ve 3. adımların tekrarlanması

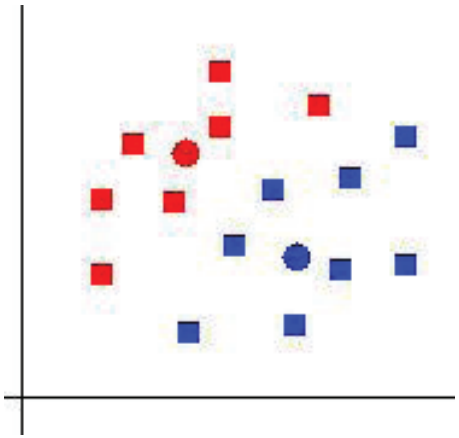
Çalışmayı daha net anlamak için aşağıdaki örnek uzaya dağılmış olan örnekleri inceleyelim:



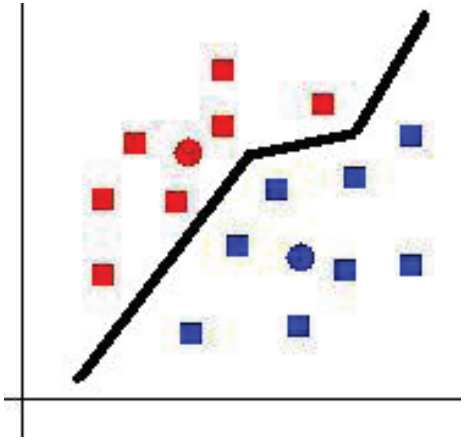
Yukarıda verilen ve uzayda koordinatları kodlanmış olan örnekler için iki adet hedef küme tanımlıyoruz. (iki sınıf ve bu sınıfların karakterlerini tanımlıyoruz).



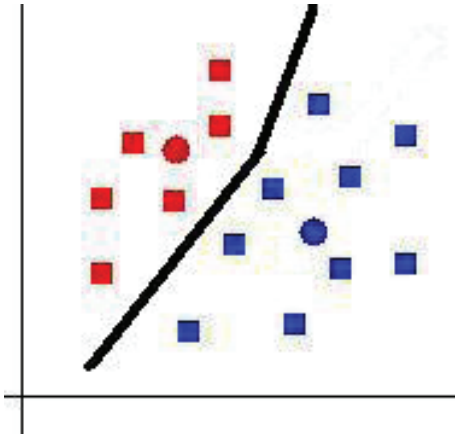
Bu sınıf tanımlarına uzaklıklarına göre (örneğin öklit mesafesi (euclid distance)) bütün örneklerimizi sınıflandırıyoruz. (hangi renge daha yakınsa).



Oluşan sınıfları ayıran bir hat, aşağıdaki şekilde çizilebilir:



Daha önceden sınıflandırdığımız örneklerin merkezlerini buluyoruz. (yuvarlak ile gösterilen ve sınıf karakteristiğini temsil eden ilk örneklerin yerini değiştirmek olarak da düşünülebilir).



Merkezleri hareket ettirdikten sonra örneklerden bazıları yeni merkezlere daha yakın olabilir. Buna göre örnek kümelerimizin sınıflandırılmasını güncelliyoruz.

Yukarıda son hali gösterilen k-means algoritmasında yeni merkezler ve her örneğin hangi sınıfa girdiği bulunmuştur.

6.8. Hata Oranları (Error Rates)

Şimdiye kadar anlatılan veri madenciliği yöntemlerinin tamamında, sistemin başarılı olabilmesi için, eğitim sırasında hatayı hesaplayabilmesi ve hesapladığı hataya göre kendisini eğitebilmesi gerekmektedir. Bu anlamda hatanın doğru hesaplanması oldukça önemli bir rol oynar. Bu bölümde, hataların nasıl hesaplanabileceğini göreceğiz.

6.8.1 Kare Ortalamalarının Karekökü (Root Mean Square Error)

Bilgisayar bilimlerinde çeşitli istatistik ve hesaplama alanlarında kullanılan bir formüldür. Basitçe üç aşamadan

oluşur:

Değerlerin karelerini al (square)

Kareleri alınan bu değerlerin ortalamasını al (mean)

Bu ortalama değerinin karekökünü al (root)

Örneğin aşağıdaki sayılar için bu işlemi deneyelim:

-2, 5, -8, 9, -4

kareleri: 4, 25, 64, 81, 16

Ortalamaları : (4+25 +64+81+16)/5 = 190 / 5 = 38

Karekökü = kök (38) = 6.16 olarak bulunur.

Yukarıdaki örnekten de anlaşılacağı üzere, sisteme giren değerlerin kareleri alındığı için sonuç her zaman pozitif çıkmaktadır.

6.8.2. Tip 1 ve Tip 2 Hata Oranları

Konu aslında, bir tahmin ve gerçekleşen durum arasında yaşanmaktadır ve istatistikte bulunan null hypothesis (yokluk hipotezi) üzerine kuruludur.

Yazının istatistik ile ilgili kısmına geçmeden önce biraz felsefe bilgimizi tazeleyerek yokluk hipotezinden bahsetmek istiyorum.

Basitçe, olmayana ergi (proof by contradiction, burhan-ı mütenakis) yapısında bir ispat yöntemidir. İspatlanmak istenen istatistiksel olgu bir hipotez olarak ortaya atılır ve buna alternatif hipotez (alterantive hypothesis) ismi verilir. Ardından bu alternatif hipotezin tam tersini gösteren ve yokluğu ispat edilmesi amaçlanan yokluk hipotezi (null hypothesis) konulur.

Bu durumu bir örnek üzerinden açıklayalım. Örneğin veri tabanı teorisinde sıkça verilen ve üretilebilir veri konusunda geçen iki alanı ele alalım. Diyelim ki bir kişinin yaşı ve doğum günü arasında ilişki olduğunu, bu iki değerin birbirine bağlı olduğunu iddia ediyoruz. Bu bizim alternatif hipotezimiz olmuş oluyor.

Bu durumda ilişki olmadığını göstermek de yokluk hipotezimiz olur.

Bu iki hipotezi aşağıdaki şekilde de gösterebiliriz:

Alternatif Hipotez: $Doğum\ Günü\ Sayısı = a\ Yaş + b$

Burada ifade edilen değer, doğum günleri ile yaş arasında doğrusal bir bağlantı olduğudur (linear, affine). Şimdi yokluk hipotezimizi yazalım (null hypothesis)

$Doğum\ günü\ sayısı\ artarken\ yaşın\ değişmemesi\ (veya\ tam\ tersi)$

Yani iki artış arasında bir bağlantı olmaması.

Ardından yapılan çok sayıdaki deneme ile görüyorsunuz ki yaş arttıkça doğum günü kutlamalarının sayısı artıyor veya doğum günü kutlamalarının sayısı arttıkça yaş ilerliyor.

Ardından diyebiliriz ki yokluk hipotezi yoktur (nullify).

Şayet yokluk hipotezinin yok olduğunu gösterebilirsek (istatistiksel olarak) o zaman alternatif hipotezimizin doğru olduğunu söyleyebiliriz. Yani doğum günü kutlaması ve yaş arasında pozitif bağlantı vardır denilebilir.

Bu konu aslında felsefede pek çok konuyu beraberinde getirmektedir. Örneğin Karl Popper'ın iddialarından birisi gerçekte alternatif ve yokluk hipotezlerinin birbirinin tersi olmasının gösterilmesinin zor olduğudur.

Ancak biz konumuzla ilgili olan bu kısmı ile iktifa

edeceğiz. Şimdi gelelim tip 1 ve tip 2 hata miktarlarına.

Öncelikle rastgele bir süreçten bahsediyoruz demektir (random process) ve bu süreçte beklenen ve gerçekleşen olaylar bulunuyor demektir. Aslında tam olarak yokluk hipotezimizin yok olduğunu ispatlamaya çalıştığımızı düşünelim:

	Beklenen (Expected)		
Gerçekleşen (Measured)		Müspet	Menfi
	Doğru	TP (DMü)	TN (DMe)
	Yanlış	FP (YMü)	FN (YMe)

Beklentimiz müspet (pozitif) veya menfi (negatif) yönde olabilir. Beklentimize bağlı olarak gerçekleşen olay (veya ölçümlerimiz) da doğru veya yanlış olabilir.

Bu durumu bir örnek üzerinden açıklayalım. Örneğin bir e-posta koruma sistemi yazıyoruz ve sistemdeki izinsiz gönderileri (spam mail) tespit etmek istiyoruz. Yazılımımız bu gönderileri tespit edip engelleyecek. Yazılımı hazırlayıp bir süre test ettikten sonra aşağıdaki gibi bir tablo hazırlanabilir.

	İstenmeyen Tahmini		
Gerçekten İstenmeyen		İstenmeyen	İstenen
	Doğru	TP (DMü)	TN (DMe)
	Yanlış	FP (YMü)	FN (YMe)

Buna göre örneğin bizim istenmeyen posta olarak sınıflandırdığımız ve gerçekte istenen postalar TP (True Positive), bizim istenen olarak sınıflandırdığımız ve gerçekte de istenen olanlar TN (True Negative), bizim istenmeyen diye sınıflandırdığımız ve gerçekte de istenmeyen olanlar FP (False Positive) ve son olarak bizim istenen diye sınıflandırdığımız ancak gerçekte istenmeyen olanlar da FN (False Negative) olarak isimlendirilebilir.

Bunlardan FN (False Negative) Tip 2 hata oranı (Type 2 error rate) ve FP (False Positive) ise Tip 1 hata oranı (Type 1 error rate) olarak isimlendirilir.

6.8.3. F Ölçümü (F-Measure)

Esas olarak istatistik biliminin bir skorlama kavramı olan ve literatürde f1 skorlama, f-skorlama, f-ölçümü (f-measure) olarak geçen kavram, bilgisayar bilimlerinde özellikle veri çıkarımı (information extraction) ve veri getirimi (information retrieval) konularında kullanılmaktadır.

Bilgi çıkarımı konusunda, kesinlik (precision) ve hassasiyet (recall, yakalama oranı) kavramları üzerinden hesaplanır.

Buradaki kesinlik kavramı genelde p harfi ile gösterilir ve getirilen bilgideki doğru sonuçların, getirilen bilginin tamamına oranı olarak hesaplanır.

$$Kesinlik(Precision) = \frac{\{ilgili\ getirim\} \cap \{bütün\ veri\ çıkarımı\}}{\{bütün\ veri\ çıkarımı\}}$$

Hassasiyet kavramı da genelde r harfi ile gösterilir ve getirilen doğru sonuçların, getirilmesi gereken doğru sonuçlara oranı ile hesaplanır.

$$Hassasiyet(Recall) = \frac{\{ilgili\ getirim\} \cap \{bütün\ veri\ çıkarımı\}}{\{ilgili\ veri\ çıkarımı\}}$$

Yukarıdaki bu tanımlar ışığında, F_1 skoru, bu değer-

lerin harmonik ortalamasıdır (harmonic mean):

$$F_1 Skoru = 2\,\frac{Kesinlik\,.\,Hassasiyet}{Kesinlik + Hassasiyet} = 2\,\frac{pr}{p+r}$$

F skorunun bir β değerine bağlanması da mümkündür. Bu durumda, F_1 skoru yerine F_β skoru terimi kullanılır.

$$F_\beta Skoru = (1+\beta 2)\,\frac{Kesinlik\,.\,Hassasiyet}{\beta^2\,.\,Kesinlik + Hassasiyet} = (1+\beta 2)\,\frac{pr}{\beta^2\,.\,p+r}$$

Buradaki β değeri, kesinlik ve hassasiyet arasındaki dengeyi belirler. Örneğin β = 2 için bulunan değer hassasiyeti iki misli etkili kılarken, β = 0.5 için bulunan değer kesinlik etkisini iki misline çıkarır. Unutulmaması gereken bir durum β değerinin pozitif tam sayı olduğudur.

7. WEKA ile Veri Madenciliği

Bu bölümde, bir önceki bölümde anlatılan veri madenciliği tekniklerinin WEKA yazılımı ile nasıl yapıldığı açıklanacaktır.

7. WEKA ile Veri Madenciliği

WEKA yazılımı, her ne kadar makine öğrenmesi (machine learning) için geliştirilmiş bir araç olsa da, temel veri madenciliği adımlarını kapsar. Kitabın daha önceki bölümlerinde veri madenciliğine kadar olan süreçte en önemli adımlardan birisi olan veri ön işleme (data preprocessing) kısmında WEKA'nın nasıl kullanıldığını açıklamıştık (bkz. 4. Bölüm). Bu aşamadan sonraki KDD adımı, yani veri dönüştürme (data transformation) adımının her kurumda farklı uygulamaları olabilir.

7.1. WEKA ile SVM

WEKA üzerinde genel olarak bir işlem yapmak için aşağıdaki adımlar izlenebilir:

- Eğitim için veri kümesi hazırlanması
- Weka yazılımının yüklenmesi
- Eğitim setinin Weka içerisinden yüklenmesi

- Weka içerisinde istenen modülün yüklenmesi
- İlgili modül için parametrelerin ayarlanması
- Test için ayarların yapılması
- Çıktıların seçilmesi
- Sonuçlar
- Tahmin Bilgileri

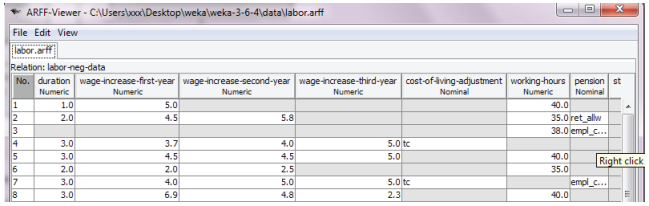
Yukarıdaki adımlara göre, öncelikle, işlenmek için hazır bir veri kümemiz bulunması gerekiyor.

Yazımızdaki amaç konuyu öğrenmek olduğu için, WEKA kurulumu ile gelen örnek veri kümelerinden birisini, bu yazı kapsamında kullanacağız.

1. **Adım:** WEKA'yı açıyoruz:

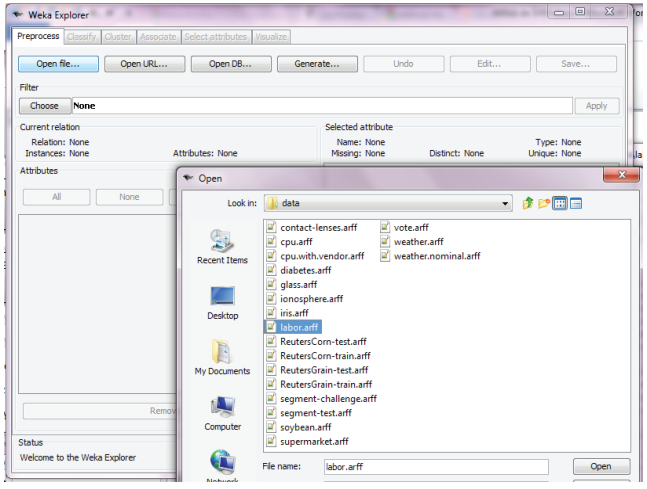
2. Adım: Önce veri kümemize bir göz atalım. WEKA'nın kurulumu ile birlikte gelen data dizini altındaki, labor.arff dosyasını görmek için, açılan ilk weka

ekranında (yukarıdaki resimde temsil edilen), "Experimenter" düğmesine basıyoruz. Açılan ekranda, File / Open seçeneklerini menülerden seçtikten sonra, dosya seçicisinden "labor.arff" dosyasını işaretleyip açıyoruz:

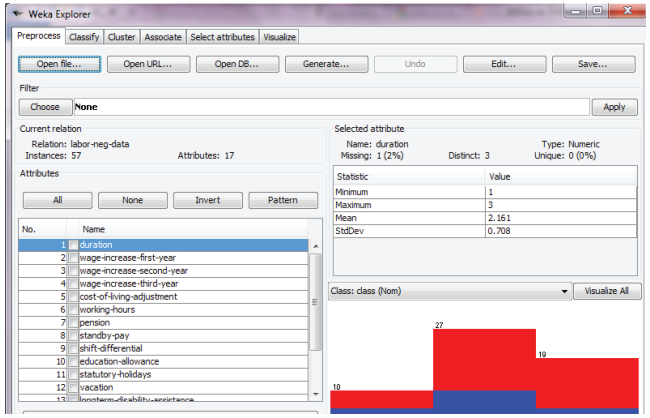


Kullanacağımız örnek veri kümesi yukarıdaki şekildedir. Hızlı bir bakışla, verilerin çalışan verileri olduğunu ve her çalışanın maaş, çalışma süreleri, sosyal güvenliğine ait bilgilerin yer aldığı bir veri kümesi olduğunu söyleyebiliriz. Şimdiki adımımız bu veri kümesi üzerinden SVM algoritmasını çalıştırmak olacak.

3. Adım: WEKA'nın ilk açılan ana ekranına dönerek, bu sefer "Explorer" düğmesine basıyoruz ve WEKA içerisinde yer alan fonksiyonları kullanacağımız ekran açılıyor.

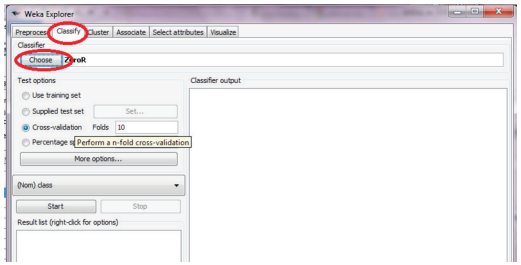


Yukarıdaki açılan Weka Explorer ekranında bulunan ilk sekme olan Preprocess sekmesinde (ki ilk açıldığında bu sekme açık gelir), bulunan "Open File" düğmesine tıklayarak dosya seçicimizi açıyoruz ve yine Weka'nın kurulumuyla gelen data dizini altında bulunan "labor.arff" dosyasını seçerek açıyoruz.

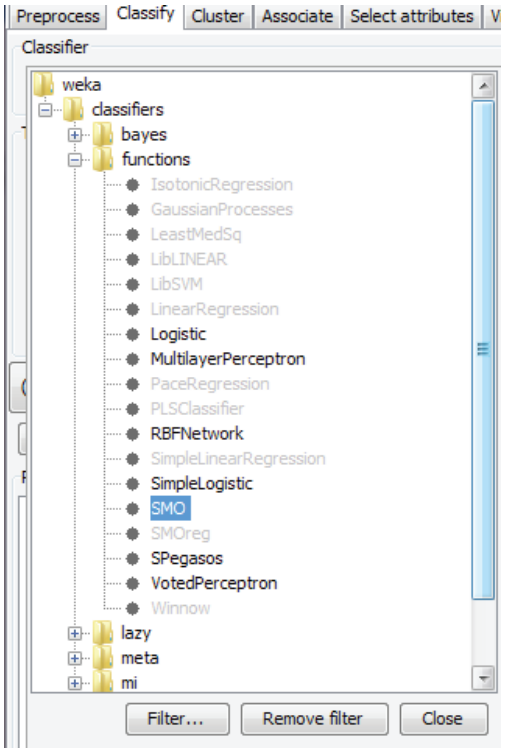


Ekranda, seçilen bu dosyanın ön işleme sonuçları görülüyor. Buna göre sol taraftaki listede, ilk başta görüntülediğimiz arff dosyasının her bir kolonu yer almakta olup, her kolona tıklandığında, bu kolon ile ilgili veri detayları sağ tarafta çubuk grafik olarak görülmektedir.

4. Adım: WEKA'nın sınıflandırma işlemi için ilgili modülün yüklenmesine geçeceğiz. Bu işlem için, ekranın üzerinde bulunan "Classify" sekmesine geçiyoruz.



Sekmeye geçtikten sonra "Choose" düğmesine tıklayarak kullanacağımız modülü seçeceğiz. Burada WEKA içerisinde yüklü olan ve konu ile ilgisi olan 3 ayrı modülden bahsetmekte yarar vardır:



Listede görülen classifiers klasörü altındaki functions elemanları sınıflandırma için kullanılan fonksiyonlardır. Tam bu noktada 3 ayrı alternatiften bahsedelim:

SMO

Yani veriyi birden fazla kümeye bölmeye yarar. Listede bulunan SMO, SVM benzeri bir yapıdadır.

Sequential Minimal Optimisation kelimelerinin baş harflerinden oluşan SMO, esas itibariyle Suppor Vector kullanan bir algoritmadır. Algoritmanın detayı için WEKA dokümantasyonunda da yer alan aşağıdaki makalelere bakılabilir:

- J. Platt: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schoelkopf and C. Burges and A. Smola, editors, Advances in Kernel Methods – Support Vector Learning, 1998.
- S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy (2001). Improvements to Platt's SMO Algorithm for SVM Classifier Design. Neural Computation. 13(3):637-649.
- Trevor Hastie, Robert Tibshirani: Classification by Pairwise Coupling. In: Advances in Neural Information Processing Systems, 1998.

LibSVM

SVM sınıflandırıcısının en önemli kütüphanesi, bir doktora tezi sırasında ortaya çıkmış olan ve libSVM olarak bilinen açık kaynak kodlu kütüphanedir. Bu kütüphane, ne yazık ki WEKA kurulumu ile gelememektedir. Ancak bu kütüphaneyi WEKA altında kullanmanın iki yolu bulunur. İlki, kütüphaneyi WEKA'ya tanıtmaktır.

Bunun için, yukarıdaki temsili resmi verilen listede bulunan libSVM kullanılabilir. Öncelikle kütüphane, aşağıdaki adresten indirilmelidir:

http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Ardından weka'nın paket yöneticisi tarafından kurulmalıdır. Weka'nın paket yöneticisi ise WEKA'nın classpath tanımının yapıldığı bir konsolda aşağıdaki komutu vererek çalıştırılabilir:

java weka.core.WekaPackageManager

Diğer bir yöntem ise, indirilen libsvm kütüphanesinin WEKA classpath'ine eklenmesidir. İlgili ayarlar yapıldıktan sonra 4. adımda bahsettiğimiz sınıflandırmalardan birisi olarak libSVM kullanılabilir.

LibSVM kullanımı için ikinci bir yöntem ise, LibSVM ve WEKA programlarını aynı anda içeren wlsvm kütüphanesini kullanmaktır. Bu kütüphaneye de aşağıdaki adresten erişilebilir:

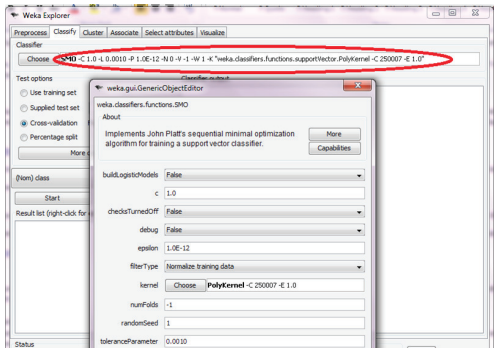
http://www.cs.iastate.edu/~yasser/wlsvm/

SMOreg

SVM ile ilgili, weka içerisinde bulunan 3. bir seçenek de SMOreg seçeneğidir. Bu seçenek, SVM üzerinde Regression (ilkelleme, regrezizasyon) uygulamaya yarayan kütüphanedir. Bu kütüphane için de detaylı bilgi aşağıdaki kaynaklardan edinilebilir:

- Alex J. Smola, Bernhard Scholkopf (1998). *A Tutorial on Support Vector Regression*. NeuroCOLT2 Technical Report Series – NC2-TR-1998-030.
- S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy, *Improvements to SMO Algorithm for SVM Regression*. Technical Report CD-99-16, Control Division Dept of Mechanical and Production Engineering, National University of Singapore.

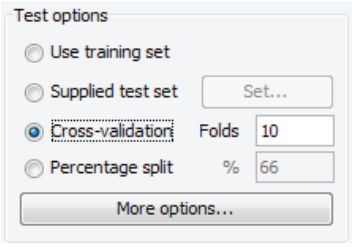
5. Adımda, Yukarıdaki seçeneklerden birisi seçildikten sonra, seçilen modüle göre ayarlama yapılması gerekmektedir. Örneğin SMO kütüphanesi üzerinden ilerleyecek olalım. Bu modülün parametre ayarlarını, ObjectEditor ile yapabiliriz. ObjectEditor'ü açmak için ekranda bulunan ve fonksiyonun parametrelerini içeren metin kutusuna tıklamak yeterlidir:



Açılan bu ekranda ilgili parametre ayarları yapıldıktan sonra, sınıflandırma fonksiyonunun çalışması için, WEKA Explorer ekranında bulunan "Start" düğmesine

basmak yeterli olacaktır. Sonuçlar ve sonuçların değerlendirmeleri (hata miktarları gibi), sonuç ekranında görülecektir. Buradaki sonucu etkileyen diğer bir ayar da, kullanım sırasında eğitim ve test için izlenecek stratejidir.

WEKA doğrudan, 4 farklı özellik desteklemektedir. Bunlar sırasıyla aşağıda anlatılmıştır:



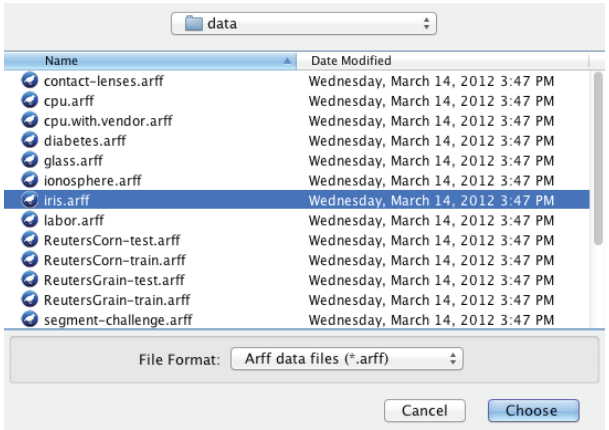
Training set seçeneği, seçilen verinin sadece eğitim amaçlı kullanılmasını sağlar. Bu seçenekte önce eğitim yapılıp ardından ikinci bir küme, test amaçlı kullanılabilir. Bu seçenek de yine yukarıdaki ekranda bulunan "Supplied test set" seçeneği ile gösterilen küme üzerinde çalışır.

"Cross-validation" seçeneği, veri kümesini belirtilen (folds sayısı olarak belirtilen) sayıda kümeye böler. Alt kümelerden birisini eğitim kümesi olarak kabul ederek sistemi eğitir. Ardından bu eğitim sonucunu diğer bir alt küme üzerinde sınar (ki bu kümeye sınama kümesi (validating set) veya test kümesi (test set) ismi verilir). Bu işlemi belirtilen küme sayısı kadar tekrarlayarak sistemi iyileştirmeye çalışır.

Son seçenek olan "percentage split" ise, verilen oranda kümeyi ikiye böler ve ilk kümeyi eğitim, ikinci kümeyi ise test amaçlı kullanır. Örneğin %66 bölümün anlamı, kümenin ilk %66 kısmının eğitim, sonraki %34 kısmının ise test amaçlı kullanılacağıdır.

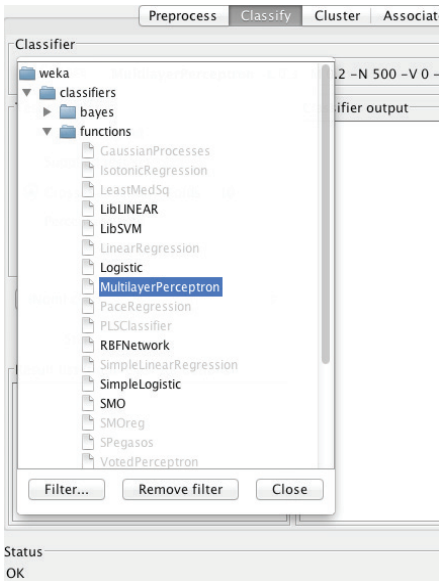
7.2. Weka ile Yapay Sinir Ağları (Artificial Neural Networks)

Bu bölümde, Weka kullanılarak basit bir sınıflandırma problemini, yapay sinir ağı ile çözmeye çalışacağız. Weka'nın kurulumu ile gelen örnek dosyalardan çiçek yaprakları üzerinden toplanmış olan iris.arff dosyasını yükleyerek başlayalım.

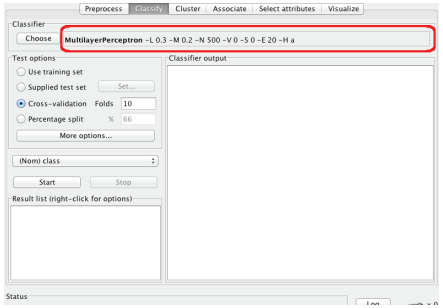


Veri yüklendikten sonra sınıflandırma amacıyla, Classifier bölümüne geçip yapay sinir ağı algoritmasını seçmemiz gerekiyor. Bunun için Classifier grubuna

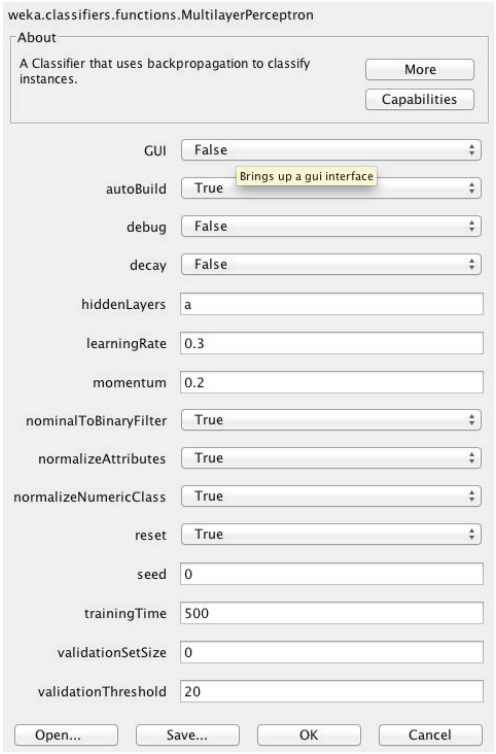
geçip "Choose" düğmesine basarak çıkan menüdeki functions klasörü altındaki MultilayerPerceptron seçeneğini seçiyoruz.



Seçimin ardından, ayarları yapmak için yüklü olan algoritmanın üzerinde herhangi bir yere tıklıyoruz:



Açılan ayarlar diyaloğu aşağıdaki şekildedir:



Bu diyalogdaki özelliklerin açıklaması aşağıda verilmiştir.

GUI: İngilizcedeki Graphical User Interface kelimelerinin baş harflerinden oluşup, kullandığımız yapay sinir ağının (ANN) görsel olarak gösterilip gösterilmeyeceğini belirleyen ayardır. Uygulamamız için bu ayarı True olarak değiştirelim. Bu sayede yapay sinir ağımız çalışırken bir yandan da ekranda ağın yapısı gösterilecektir.

İkinci seçenek olan autoBuild seçeneği, Weka'nın bizim için otomatik olarak bir yapay sinir ağı oluşturup oluşturmayacağını sormaktadır. Şayet kullanıcı kendisi elle bir yapay sinir ağı çizmek istiyorsa, bu seçeneği false olarak işaretlemelidir. Biz de detaylı bir tasarım yapabilmek için bu seçeneği false olarak işaretleyelim.

Üçüncü satırda bulunan debug seçeneği, işlemlerin çalışırken detaylı bir şekilde raporlanması ve tam olarak oluşturulan yapay sinir ağının bütün detaylarının gösterilmesini sağlar.

Dördüncü sıradaki decay (çürüme) seçeneği ise daha çok yapay sinir ağı bilgisi ile ilgili bir durumdur. Bu ayar, basitçe öğrenme oranının zaman içerisinde azalmasını (çürümesini) ifade eder.

Gizli katman sayısını belirten beşinci ayarımızda, a ifadesi otomatik olarak belirleneceği anlamına gelmektedir. Kullanıcı isterse buraya herhangi bir sayı yazarak kaç adet gizli katman istediğini belirtebilir. Buraya yazılacak 0 (sıfır) anlam olarak hiç gizli katman

bulunmadığı anlamına gelirken birden fazla sıfır (00 gibi) yazılması veya eksi ve ondalıklı sayıların yazılması bir hata olarak kabul edilmektedir. Otomatik olarak weka'nın atamasını sağlayan a değeri ise veri kümemizdeki özellik sayısı ve sınıf sayısının ortalamasıdır (ikisinin toplanıp ikiye bölümünden elde edilen değer). Ayrıca o harfi ile sınıf sayısını, i harfi ile özellik sayısını veya t harfi ile de özellik ve sınıf sayılarının toplamını parametre olarak vermemiz mümkündür.

Ardından gelen learningRate ve Momentum değerleri, yapay sinir ağlarında kullanılan ve öğrenme hızını belirleyen parametrelerdir.

Ardından gelen 3 satırdaki değerler, yine weka içerisinde olan filtreleri ifade etmektedir. Buna göre weka içerisinde bulunan nominaltobinary filtresi, bütün değerleri ikilik tabandaki (binary) 1 ve 0 değerlerine çevirmektedir. Basitçe her değer için 1 ve 0 değer karşılığı dönüşümü yapılır. Normalize Attributes ve Numeric Class seçenekleri ise veri kümesinin özellik ve sınıf bazında normalize edilmesini sağlar. Bu filtreler, veri üzerinde, sınıflandırma aşamasından önce de kullanılabilir.

Reset seçeneği, çalışma sırasında, yapay sinir ağı üzerindeki ağırlık değerlerinin hesaplanamaması (NaN değer üretmesi) gibi durumlarda, bütün öğrenme değerlerini (learning rate, momentum v.b.) orijinal haline geri getirerek baştan başlamasını sağlar. Bu sayede hesaplanamayan durumların önüne geçilmiş olunur.

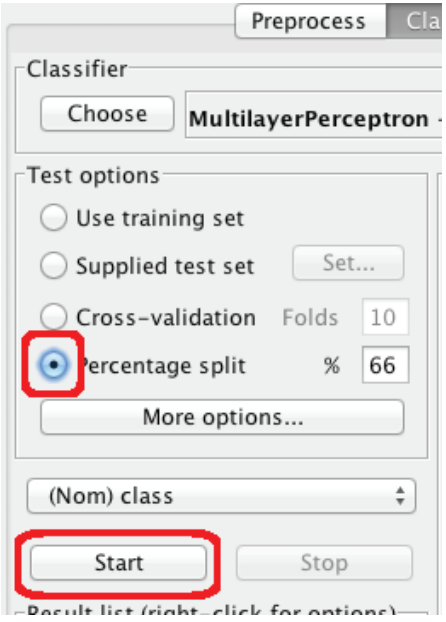
Ağın oluşturulması sırasında kullanılan rastgele

sayı üretecinin başlangıç beslemesi için seed değerine kullanıcının elle bir değer ataması mümkündür.

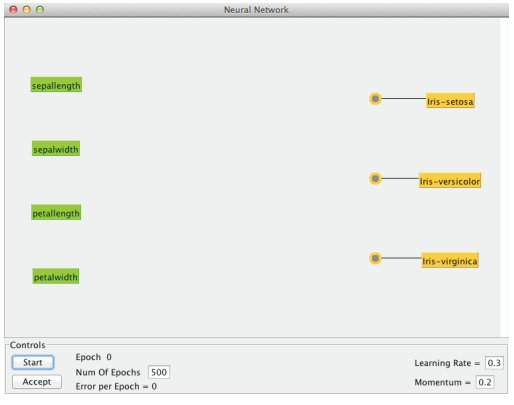
Trainingtime değeri ile öğrenme için geçecek olan adımlar (epoch) belirlenir. Buna göre ağdaki neuronların ve sinapsislerin değer güncellemesi yapılır.

Eğitim işlemi tamamlandıktan sonra test işlemi için kullanılacak olan değerler son iki satırda belirtilmiştir. Sınama (Validation) için yapılacak eşik değer ayarı bir nöronun ateşleme değeri olurken, sınama (validation) için kullanılacak veri kümesinin boyutu da validationSetSize parametresi ile ayarlanır.

Ayarlarımızı kaydettikten sonra sistemimizi çalıştırmak için önce bu diyalogdaki değerleri onaylıyoruz, ardından da Weka'nın ana ekranındaki ayarlara geçiyoruz. Weka'nın ana ekranında son olarak percentage split seçeneğini seçip 66% olarak değeri değiştirmeden bırakalım ve sistemin çalışmaya başlaması için start düğmesine basalım.



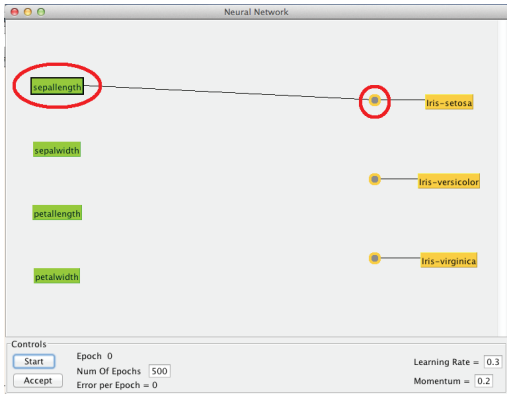
Start düğmesine basılmasının ardından, yapay sinir ağımızı tasarlayabilmemiz için Weka tarafından bir ekran açılacaktır.



Bir önceki ayarlar ekranında verdiğimiz değerler burada da son kez ayarlayabilmemiz için ekranda gösterilmiştir. Örneğin Num Of Epochs (kaç çağ boyunca sinir ağımızı eğiteceğimiz) veya öğrenme oranı (learning rate) veya momentum değerleri daha önce girdiğimiz şekilde ekranda belirmiştir ve değerleri değiştirilebilir.

Ekranın sol tarafında veri kümemizdeki özellikler (attributes) belirirken, sağ tarafta da veri kümemizdeki 3 adet sınıf gösterilmiştir. Bu özellikler ile sınıflar arasındaki bağlantıyı elle çizmemiz mümkündür.

Örneğimizde, soldaki yeşil girdiler ile çalışmaya başlayalım ve deneme amaçlı olarak girdi ve çıktı katmanları arasında bir ilişki kuralım. Örneğin "sepallength" girdisi ile "iris-setosa" arasında ilişki kurmak için öncelikle yeşil girdi değerine, yani "sepallength" tıklayıp, ardından "iris-setosa" düğümüne (yuvarlak olan çıktı katmanı) tıklıyoruz.



Yukarıda, tıklanacak olan düğümler, çember içerisine alınmıştır. Tıklamanın ardından bu iki düğüm arasında doğrudan bağlantı kurulur.

Diğer bir denemeyi de "sepalwidth" ve "iris-versicolor" arasında kurmak isteyelim. Ancak bu sefer araya ilave bir nöron eklemek istiyor olalım. Bu durumda önce boş alana bir kere tıklanarak yeni bir nöron eklenir.

Yapay sinir ağımızı çizdikten sonra ekranın altında bulunan "Start" düğmesine basılarak yapay sinir ağının öğrenmesi başlatılabilir.

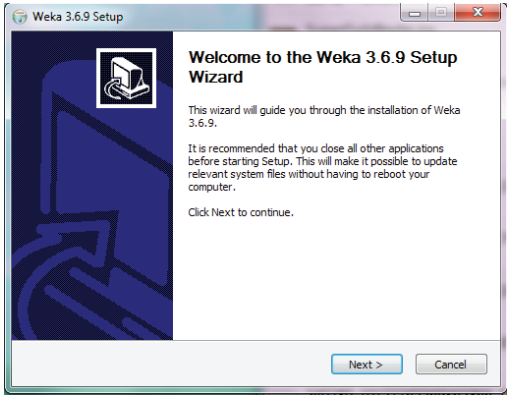
EK 1
WEKA'nın Kurulması

Weka yazılımı ücretsiz olarak, Weka'nın web sayfasından indirilebilir[3]. İndirme işlemi sırasında Weka, çeşitli mimari ve işletim sistemleri arasından seçim yapma imkanı sunmaktadır. Kullanmakta olduğumuz işletim sistemi ve bilgisayarın mimarisine göre (32 veya 64 bit) ilgili bağlantıyı seçip bilgisayara indiriyor ve kuruyoruz. Burada önemli bir nokta, Weka'nın çalışması için, bilgisayarda bir Java Sanal Makinesi (Java Virtual Machine, kısaca JVM) kurulu olması gerektiğidir. Çoğu ortamda bu kurulum zaten bulunduğu için, tercihe bağlı olarak JVM içermeyen bir bağlantı da indirilebilir.

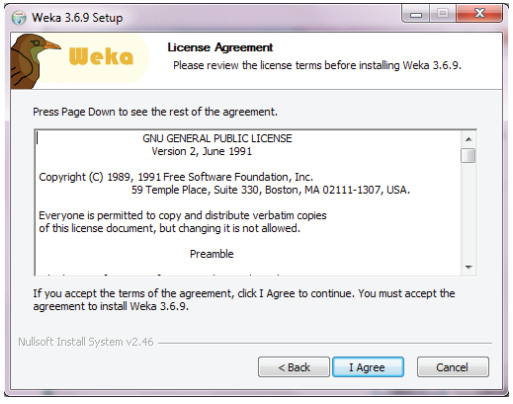
İndirme işlemi tamamlandıktan sonra, indirilen kurulum dosyası çalıştırılarak kurulum aşamasına geçilebilir. Bu kitap kapsamında sadece kurulum için, daha sık kullanılmasından dolayı Windows ortamındaki kurulum aşamaları açıklanacaktır. Ancak Weka kurulduktan sonra, Java teknolojisi sayesinde, bütün ortamlarda aynı şekilde çalışmaktadır.

Weka'nın adım adım kurulumu aşağıda anlatılmıştır. İndirdikten sonra tıklanan kurulum dosyası aşağıdaki ekran ile kurulum asistanını başlatır:

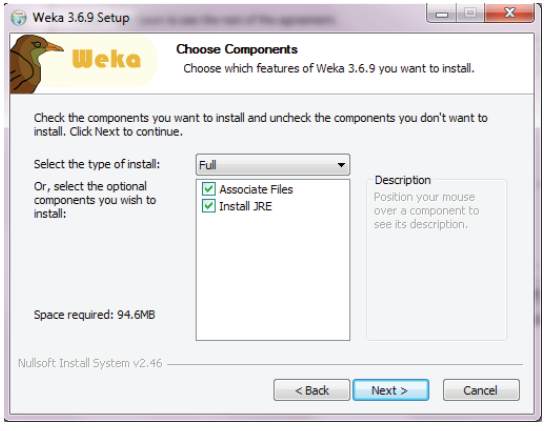
(3) Weka'nın indirilebileceği sayfa (Nisan 2013 itibariyle aktif indirme sayfasıdır): http://www.cs.waikato.ac.nz/ml/weka/downloading.html



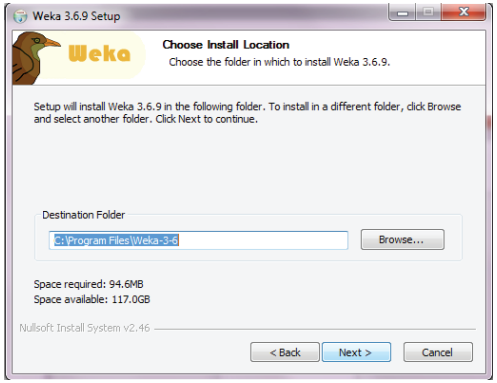
Yukarıdaki ilk kurulum ekranından sonra çıkan bütün diyalog ekranlarında "Next" düğmesine basılarak kurulum yapılabilir. Herhangi bir hata çıkmazsa kurulum bu şekilde tamamlanacaktır, ancak aşağıda kurulumun detayları hakkında bilgi verilecektir.



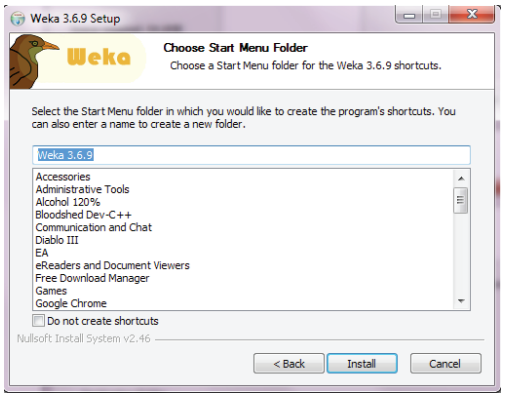
Yukarıdaki ikinci ekranda lisans sözleşmesi gösterilmektedir. Weka açık kaynak kodlu bir uygulama olup GPL (GNU, Public License) kapsamında bir son kullanıcı sözleşmesi içermektedir. Bu sözleşmeye göre, özetle, kod üzerinde değişiklik yapmaktan kodun farklı projelere dahil edilmesine kadar çok sayıda imkan kullanıcıya sunulmaktadır. Ancak en önemli ve neredeyse tek şartı, kullanılan bu kod hakkında yeni projelerde bilgi verilmesi ve Weka'ya atıfta bulunulması gerekliliğidir. Ayrıca geliştirilen yeni yazılımların da GPL kapsamında olması gerekir.



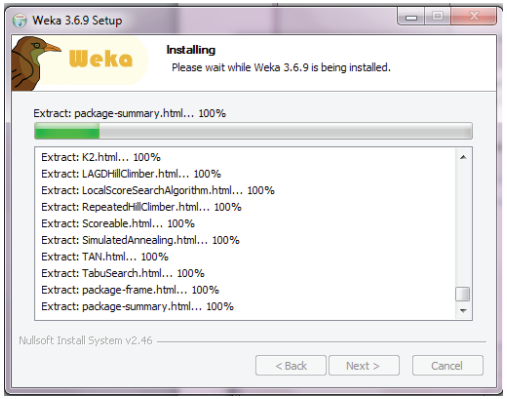
Yukarıdaki ekranda, kurulacak olan paketler seçilir. Bu kurulum adımları sırasında JRE içeren paket seçildiğinden iki alternatif de gösterilmiştir. İlk seçenek arff uzantılı dosyaların işletim sistemi tarafından Weka ile ilişkilendirilmesi, ikinci seçenek ise JRE kurulumunun da istendiği anlamına gelmektedir.



Bilgisayarda kurulması istenen konum yukarıdaki ekranda belirtilir.



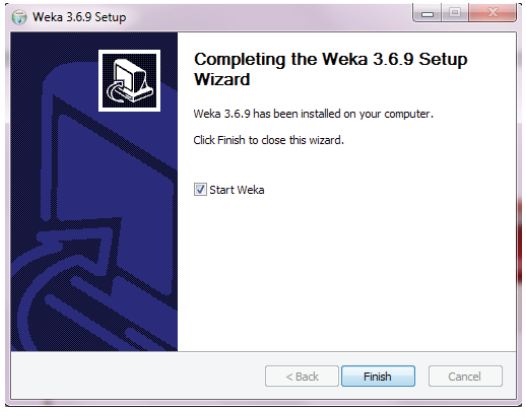
Bilgisayardaki başlat menüsü altında yerleştirileceği isim seçilir.



Bütün adımlar onaylandıktan sonra kurulum paketleri açılır.

Kurulum paketlerinin açılımının ardından, şayet seçildiyse JRE kurulumuna geçilir.

JRE kurulumu, Weka kurulumunun dışında ilave olarak kurulur ve kurulum tamamlandıktan sonra Weka'nın kurulumu devam eder.



Weka'nın kurulumu bittikten sonra, Weka'nn ilk defa başlatılması için son ekran yukarıdaki şekilde görüntülenir.

EK 2
WEKA üzerinde yazılım geliştirilmesi

WEKA, geliştiriliş şekli itibariyle JAVA dili üzerinde yazılmış açık kaynak kodlu bir yazılımdır. Bunun anlamı, WEKA içerisinde bulunan ve bu bölüme kadar ara yüzleri kullanarak eriştiğimiz bütün işlemlere, JAVA dilinden yapılacak basit nesne tanımlama ve metot çağırmaları ile ulaşabileceğimizdir. Kitabın bu kısmı, daha çok kendi yazılımını geliştiren ve yazılım tecrübesi bulunan iş zekası ve veri madenciliği konusundaki araştırmacı ve çalışanlara göre hazırlanmıştır. Şayet temel, nesne yönelimli programlama ve JAVA dili bilginiz bulunmuyorsa, bu bölümü okumadan önce bu ön bilgilerin tamamlanması tavsiye edilir.

Basit bir WEKA kodu

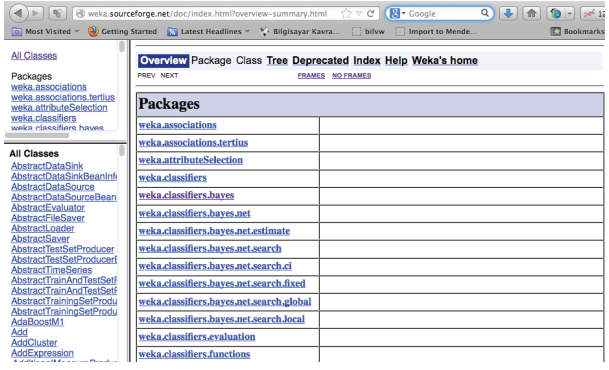
Weka ile yazılım geliştirme işlemine basit bir kod ile başlayacağız. Amacımız bir CSV dosyasını ARFF dosyasına çevirmek olsun. Bunun için daha önce anlatılan ve Weka içerisinde bulunan CSV Loader ve ARFF Saver sınıflarını kullanacağız (bkz. 3.3. Weka Bilgi Akış Ekranı)

Weka üzerinde bu kütüphaneler Weka'yı bilgisayarımıza indirdiğimiz anda gelmektedir. Weka kurulumu sırasında gelen weka.jar dosyası bütün bu kodları içinde barındırmaktadır. Bu kodda kullanacağımız kütüphanelere aşağıdaki import satırları ile ulaşmak mümkündür.

```
import weka.core.Instances;
import  weka.core.converters.Arf-
fSaver;
import weka.core.converters.CSVLo-
ader;
```

Yukarıdaki import satırlarının tam olarak yolunu bulabilmek için Weka'nın sitesinde bulunan API kullanılabilir.

Weka, açık kaynak kodlu bir proje olduğu için bütün doküman bilgilerine ve son sürümüne weka.sourceforge.net adresinden erişmek mümkündür. Nitekim API adresine de "http://weka.sourceforge.net/doc/overview-summary.html" yazılarak ulaşılabilir.



Klasik javadocs ile üretilmiş sayfalardan veya Java API'sinden alışık olunacağı üzere sınıflar, paketler ve her sınıfın altındaki bilgilere bu ara yüzden erişilebilir.

Bizim kullanacağımız CSVLoader ve ARFFSaver sınıfları ise import satırlarında da görüldüğü üzere weka.core.converter altında bulunmaktadır. Buradan, kullanacağımız sınıf hakkında daha detaylı bilgiye erişebiliriz.

Kodumuz 3 bölümden oluşacaktır:

- Girdi dosyasının yüklenmesi
- CSV yapısının ARFF yapısına çevrilmesi
- ARFF dosyasının kaydedilmesi

Yukarıdaki bu yapıya göre dosyaların yüklenmesi ve kaydedilmesi, temel Java bilgileri ile çözülebilir. Bu aşamada yeni ve önemli olan bilgi, aradaki dönüşüm aşamasıdır.

```
CSVLoader loader = new CSVLoader();
Loader.setSource(sourceFile);
Instances data = loader.getDataSet();

// save ARFF
ArffSaver saver = new ArffSaver();
saver.setInstances(data);
saver.setFile(new File(newFile));
saver.setDestination(new File(newFile));
saver.writeBatch();
```

Yukarıdaki kodda bu dönüşüm işlemi görülmektedir. CSVLoader kullanılarak bir loader objesi oluşturulmuş, ardından giriş dosyasından bu kayıtlar okunarak yine bir Weka sınıfı olan Instances sınıfından data objesi tanımlanmış. Bu data objesine CSVLoader nesnesinin veri kümesi döndürülmüştür.

Kodun ikinci kısmında ise bir ArffSaver sınıfı

oluşturulmuş ve bu sınıfın içerisine daha önce oluşturduğumuz data nesnesi parametre olarak verilmiş, ardından dosya ismi, dosya konumu saver nesnesine verilerek writeBatch() metodu ile dosyaya kayıt gerçekleştirilmiştir.

Yukarıdaki kritik kısmı alıntılanan kodun tamamı aşağıda verilmiştir.

```
   import weka.core.Instances;
   import weka.core.converters.ArffSaver;
   import weka.core.converters.CSVLoader;

   import java.io.*;


   public class csvarff {

     /**
      * takes 2 arguments:
      * - CSV input file
      * - ARFF output file
      */
     public static void main(String[] args) throws Ex-
ception {

       // load CSV
       CSVLoader loader = new CSVLoader();
       loader.setSource(new File("girdi.csv"));
       Instances data = loader.getDataSet();

       // save ARFF
       ArffSaver saver = new ArffSaver();
       saver.setInstances(data);
       saver.setFile(new File("cikti.arff"));
       saver.setDestination(new File("cikti.arff"));
       saver.writeBatch();
     }
   }
```