İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

VERİ MADENCİLİĞİNİN BULANIK UZMAN SİSTEMLERDE KULLANIMI

YÜKSEK LİSANS TEZİ
Müh. Alper KÜÇÜKURAL
(509001452)

Tezin Enstitüye Verildiği Tarih : 26 Nisan 2004
Tezin Savunulduğu Tarih : 21 Mayıs 2004

Tez Danışmanı :	**Prof.Dr. Gazanfer ÜNAL**
Diğer Jüri Üyeleri	**Yrd.Doç.Dr. Ali ERCENGİZ**
	Doç.Dr. Şakir KOCABAŞ

MAYIS 2004

ÖNSÖZ

Bana vermiş oldukları sonsuz sevgi, emek ve destek için anne, babama ve kardeşlerime teşekkürlerimi sunuyorum.

Sevgili dostlarıma ve iş arkadaşlarıma yardımlarını esirgemedikleri ve bu aşamada gösterdikleri anlayış için teşekkür ederim.

Yüksek lisans öğrenimim boyunca desteğini esirgemeyen, yeni yöntemleri gösteren ve hep yeniliklerin peşinden gitmemiz konusunda bizleri yönlendiren Sayın Prof.Dr.Gazanfer ÜNAL'a şükranlarımı sunarım.

Nisan 2004 Alper KÜÇÜKURAL

İÇİNDEKİLER

TABLO LİSTESİ **Sayfa No**

ŞEKİL LİSTESİ

Sayfa No

VERİ MADENCİLİĞİNİN BULANIK UZMAN SİSTEMLERDE KULLANIMI

ÖZET

Dünya üzerinde veri miktarının giderek artması ve kullanılamayacak boyutlara gelmesi verinin indirgenmesi, örneklenmesi ve veriden bilgiye ulaşma sürecinin önünü açmıştır. Veriler çeşitli metotlarla incelendikçe birbiri ile ilişkisi olmayan bazı alanların aslında birbirlerini dolaylı olarak etkiledikleri görülmüştür. Bu metotlarla verilerin içerisinde gizlenmiş pek çok değerli bilgiye ulaşılabilir. Bu yaklaşım veri madenciliğini gündeme getirmiştir.

Diğer taraftan uzman sistemler geniş bir kullanım alanı bulmuştur. Buna rağmen uzman sistemlerin dezavantajlarından biri kural tabanının uzmanlar tarafından hazırlanması ve yaratılmış olan kural tabanının doğruluk derecesinin ise uzman bilgisine bağlı olması idi. Bu yüksek lisans tez çalışmasında veri madenciliği yöntemlerini kullanarak, bulanık ilişkiler yardımı ile kural tabanları oluşturulması ve bu oluşturulan kural tabanının bir uzman sistem tarafından kullanılması hedeflenmiştir. Oluşturulan kurallara örnek olarak;

Eğer a = x ve b = y ise c = z

gösterilebilir. Geliştirilen yazılımın sınıflama ve kural çıkarım algoritması S.Wesley Chanchien ve Tzu-Chuen Lu tarafından 2001 yılında önerilen algoritmalara göre tasarlanmıştır. Bu uygulamada kullanılan sınıflama algoritması Kohonen tarafından önerilen SOM(Self Organizing Map) algoritmasıdır. Kural çıkarımı için Rough küme teorisi kullanılmıştır. Bulanık uzman sistem tarafında Edward S. Sazonov'un bulanık uzman sistem algoritmaları uygulanmıştır.

Geliştirilen uygulama java platformunda yazılmıştır. Veri tabanı olarak Oracle, Access ve SQLite denenmiş, proje, taşınılabilirliği ve hızı göz önünde bulundurulduğunda SQLite veritabanı ile sunulmuştur. Geliştirme ortamı olarak Oracle JDeveloper 10g kullanılmıştır.

Birinci bölümde, veri madenciliği, veri ambarları ve veri madenciliğinin kullanım alanları hakkında bilgi verilmiştir.

İkinci bölümde, uygulamada kullanılmış olan bulanık küme ve bulanık mantık kavramları hakkında bilgiler sunulmuştur.

Üçüncü bölümde, uzman sistemler, uzman sistemlerin kullanım alanları, bulanık uzman sistemler ile ilgili bilgi verilmiştir.

Dördüncü bölümde, veri madenciliği aracında kullanılan algoritmalara değinilmiştir.

Beşinci bölümde ise, uygulama tanıtılmıştır. Burada veri madenciliği aracı ve bulanık uzman sistem ile ilgili ayrıntılı bilgiler verilmiştir.

Beşinci bölümde sonuçlar ve önerilere değinilmiştir.

USING DATA MINING IN FUZZY EXPERT SYSTEMS

SUMMARY

The rapid increase of the amount of data has made it unusable, however, it has also given rise to data simplification and sampling, which in turn has facilitated the transfer of data into information. Investigation of data through various methods shows that there are fields, which appear unrelated at first sight, but do in fact indirectly influence one another. Through the employment of miscellaneous methods it is possible to reach to essential but hidden information within the data. This approach has brought to the agenda data mining.

Although expert systems are being used wide range of fields, one of the disadvantages of the expert systems is the preparation of the rule base by the expert and the same expert knowledge is used to check the validity of this created rule base.

This master thesis utilises the method of data mining and aims the creation of a rule base by the help of fuzzy relations which is going to be used by an expert system. The following can be an example of the created rules:

If a = x and b = y then c = z

The classification and rule extraction of the produced software is designed according to the algorithms proposed by S.Wesley Chanchien and Tzu-Chuen Lu in 2001.

The classification algorithm used in this application is the SOM (Self Organizing Map) algorithm developed by Kohonen. For rule extraction the Rough set theory is applied. As far as the fuzzy expert system is concerned, Edward S. Sazonov's fuzzy expert system algorithms are utilised. The developed application is written in Java platform (language). Oracle, Access and SQLite have been tried out as databases, however by taking into consideration the portability and performance, the project is presented with SQLite database. As a development environment Oracle JDeveloper 10g is used.

The organisation of the material is as follows: the first chapter gives information about data mining, data warehouses and the fields of application of data mining. The second chapter deals with the fuzzy set and fuzzy logic concept that are used in this application. The third chapter elaborates on expert systems, the fields where expert systems are applied and fuzzy expert systems. The forth capther

introduces on data mining tool and their alghoritms. The fifth chapter introduces the application. Moreover, this chapter provides detailed information about the data mining instrument and the fuzzy expert system. The sixth, and the last, chapter reflects on the results and offers some remarks for the future applications.

1. GİRİŞ

Verilerin saklanması ve işlenen verilerin yeni veriler oluşturması sonucu bilişim sitemlerinde bulunan veri miktarı her geçen gün katlanarak artmaktadır. Verilerin bu denli fazla olması veriler üzerinde çalışmayı güçleştirmektedir. Verilerin etkin bir şekilde sorgulanabilmesi ve kullanılabilmesi için pek çok yöntem geliştirilmiştir. Veri madenciliği uygulamaları bu alanda geliştirilen yöntemlerden biridir.

Bilgisayarların depoladığı her türlü veri o firma tarafından değerlidir anlamı taşımaz. Bu verilerin değerlendirilmesi ve anlaşılır bir dilde özetlerinin çıkarılması gerekir. Veri madenciliği uygulamalarında yapılan örnekleme, sınıflandırma gibi işlemler veri yoğunluğunu azaltarak değerli bilgiye ulaşılmasını sağlar.

Bu yüksek lisans tez çalışmasında kullanılan iki alanın, veri madenciliği ve bulanık uzman sistem uygulamalarının birleştirilerek birbirlerini beslediği bir yapının kurulması hedeflenmiştir. Genel olarak, verilerin bulanıklaştırılmasında, bulanık uzman sistemin metotlarını kullanan ve kural çıkarılmasında veri madenciliği uygulamasının metotlarını kullanan, bu kurallar ile çıkarım yapan bir model kurulmuştur.

Geliştirilen yazılımın sınıflama ve kural çıkarım algoritması S.Wesley Chanchien ve Tzu-Chuen Lu tarafından 2001 yılında önerilen algoritmalara göre tasarlanmıştır. Bu uygulamada kullanılan sınıflama algoritması Kohonen tarafından önerilen SOM(Self Organizing Map) algoritmasıdır. Kural çıkarımı için Rough küme teorisi kullanılmıştır. Bulanık uzman sistem tarafında Edward S. Sazonov'un bulanık uzman sistem algoritmaları uygulanmıştır.

Bölüm birde, veri madenciliği, veri ambarları ve veri madenciliğinin kullanım alanları hakkında bilgi verilmiştir. İkinci bölümde, uygulamada kullanılmış olan bulanık küme ve bulanık mantık kavramları hakkında bilgiler sunulmuştur. Bölüm üçte, uzman sistemler, uzman sistemlerin kullanım alanları, bulanık uzman sistemler ile ilgili bilgi verilmiştir. Dördüncü bölümde ise, uygulama tanıtılmıştır. Burada veri madenciliği aracı ve bulanık uzman sistem ile ilgili ayrıntılı bilgiler verilmiştir. Son bölümde ise sonuçlar ve önerilere değinilmiştir.

1.1 Veri Madenciliği

Verilerin dijital ortamda saklanmaya başlanması ile birlikte, yeryüzündeki bilgi miktarı exponansiyel bir şekilde artmaktadır. Günümüzde veri tabanlarının sayısı da benzer şekilde artmaktadır. Veri miktarının bu derece artması veriye ulaşmayı ve veri değerlendirmeyi zorlaştırmıştır. Veri madenciliği uygulamaları ile işlenmemiş verilerin belirli süreçlerden ve algoritmalardan geçirilerek kullanılabilir hale getirilmesi sağlanmıştır.

Veri kendi başına değersizdir. İstenilen, amaçlar doğrusunda bilgiye ulaşmaktır. Verilerin bilgiye dönüştürülmesine veri analizi denir. Veri madenciliği büyük miktarda veri içinden gelecekle ilgili tahmin yapmamızı sağlayacak bağıntı ve kuralların bilgisayar programları kullanarak aranmasıdır.

Günümüzde oldukça yaygınlaşan elektronik ticaret ve online alışveriş sitelerinin artmasıyla birlikte, bu alanda müşteri alışkanlıklarını belirlemek ve müşteriye özel çözümleri sunmak üzerine pek çok veri madenciliği uygulaması geliştirilmiştir.

Veri madenciliği, eldeki verilerden önceden bilinmeyen veya tahmin edilmesi güç olan ancak potansiyel olarak kullanışlı bilginin çıkarılması olarak özetlenebilir. Bu da; veri özetleme, kümeleme, değişikliklerin analizi, sapmaların tespiti gibi teknik yaklaşımlar yardımıyla yapılır.

Veri madenciliği, verilerin içerisindeki desenlerin, ilişkilerin, değişimlerin, düzensizliklerin, kuralların ve istatistiksel olarak önemli olan yapıların belirlenmesidir. Veriler arasındaki ilişkiyi, kuralları ve özellikleri belirler. Amaç, daha önceden fark edilmemiş veri desenlerini tespit edebilmektir.

1.1.1 Veri Ambarları ve Veri Madenciliği

Bir veya birden fazla veritabanı, kullanıcının veri üzerinde daha rahat sorgulama yapabilmesi için bir araya getirilerek tekrar modellenir. Burada amaç kullanıcının ilgilendiği verileri veritabanından çıkartarak daha kolay sorgulanabilecek hale getirmektir.

Genel olarak günümüzde yaygın olarak kullanılmaya başlanan veri ambarları günlük kullanılan veri tabanlarının birleştirilmiş ve işlemeye daha uygun bir özetini saklamayı amaçlar.

Günlük veri tabanlarından istenen özet bilgi seçilerek ve gerekli önişlemeden sonra veri ambarında saklanır. Ardından amaç doğrultusunda gerekli veri, ambardan alınarak veri madenciliği çalışması için standart bir forma çevrilir.

Veri ambarında veri oluşturulduktan sonra bu verinin analizi yapılabilir. Bunun için OLAP (Online Analytical Processing) programları kullanılır. Bu programlar veritabanında bulunan alanları belirli bir model ele alınarak boyutlar oluşturulmasıdır. Bu birbirinden bağımsız olarak oluşturulan boyut kümelerine OLAP küpleri denir. Böylece boyut bazında gruplama, boyutlar arasındaki korelasyonları inceleme ve sonuçları grafik veya rapor olarak sunma olanağı sağlar.

Veri madenciliğinde amaç, kullanıcının bilgi çıkarma sürecinde katkısının olabildiğince az tutulması, işin olabildiğince otomatik olarak yapılabilmesidir. Çünkü OLAP programlarını kullanırken bulunabilecek sonuçlar kullanıcının sormayı düşündüğü sorgularla sınırlıdır. Veri madenciliğinde temel amaç birbirinden tamamen farklı görünen alanların birbiri arasındaki ilişkilerini bulmaktır. Örneğin "Matematik dersinden iyi derecede mezun olanlar Fizik dersinden de iyi derecede mezun olur" önermesini gerçeklemek için veri madenciliği uygulamalarına ihtiyaç duyarız. Veri madenciliği uygulamaları ile birbirinden tamamen ayrı ve uzak ilişkileri bulunan alanların aslında birbirlerini etkilediği görülmüştür. Örneğin bir süper markette çocuk bezi alan müşterilerin aynı zamanda bira aldığı saptanmış ve market raflarına biraları, çocuk bezlerinin yakınlarına yerleştirip, satış artışı sağlanmıştır.

1.1.2 Veri Madenciliğinde Kullanılan Teknikler

İstatistiksel Yöntemler:

Veri madenciliği çalışması esas olarak bir istatistik uygulamasıdır. Verilen bir örnek kümesine bir kestirici oturtmayı amaçlar. İstatistik literatüründe son elli yılda bu amaç için değişik teknikler önerilmiştir. Bu teknikler istatistik literatüründe çok boyutlu analiz (multivariate analysis) başlığı altında toplanır ve genelde verinin parametrik bir modelden (çoğunlukla çok boyutlu bir Gauss dağılımından) geldiğini varsayar.

Bu varsayım altında sınıflandırma (classification; discriminant analysis), regresyon, öbekleme (clustering), boyut azaltma (dimensionality reduction), hipotez testi, varyans analizi, bağıntı (association; dependency) kurma için teknikler istatistikte uzun yıllardır kullanılmaktadır.[3]

Bellek Tabanlı Yöntemler:

Bellek tabanlı veya örnek tabanlı bu yöntemler (memory-based, instance-based methods; case-based reasoning) istatistikte 1950'li yıllarda önerilmiş olmasına rağmen o yıllarda gerektirdiği hesaplama ve bellek yüzünden kullanılamamış ama günümüzde bilgisayarların ucuzlaması ve kapasitelerinin artmasıyla, özellikle de çok işlemcili sistemlerin yaygınlaşmasıyla, kullanılabilir olmuştur. Bu yönteme en iyi örnek en yakın k komşu algoritmasıdır (k-nearest neighbor)

Yapay Sinir Ağları:

1980'lerden sonra yaygınlaşan yapay sinir ağlarında (artificial neural networks) amaç fonksiyon birbirine bağlı basit işlemci ünitelerinden oluşan bir ağ üzerine dağıtılmıştır.Yapay sinir ağlarında kullanılan öğrenme algoritmaları veriden üniteler arasındaki bağlantı ağırlıklarını hesaplar. YSA istatistiksel yöntemler gibi veri hakkında parametrik bir model varsaymaz yani uygulama alanı daha geniştir, ve bellek tabanlı yöntemler kadar yüksek işlem ve bellek gerektirmez. Kohonen tarafından 1995 yılında ortaya atılan Self Organizing Map bir yapay sinir ağı uygulamasıdır.

Karar Ağaçları:

İstatistiksel yöntemlerde veya yapay sinir ağlarında veriden bir fonksiyon öğrenildikten sonra bu fonksiyonun insanlar tarafından anlaşılabilecek bir kural olarak yorumlanması zordur. Karar ağaçları ise veriden oluşturulduktan sonra yukarıdaki örnekte de olduğu gibi ağaç kökten yaprağa doğru inilerek kurallar (IF-THEN rules) yazılabilir. Bu şekilde kural çıkarma (rule extraction), veri madenciliği çalışmasının sonucunun gerçeklenmesini sağlar. Bu kurallar uygulama konusunda uzman bir kişiye gösterilerek sonucun anlamlı olup olmadığı denetlenebilir. Sonradan başka bir teknik kullanılacak bile olsa karar ağacı ile önce bir kısa çalışma yapmak, önemli değişkenler ve yaklaşık kurallar konusunda bize bilgi verir ve tavsiye edilir.

1.1.3 Örnek Veri Madenciliği Uygulama Alanları

Veri madenciliği veri tabanları kullanılan sektörlerde sıklıkla kullanılmaya başlandı. Bu sektörler arasında başlıca ilaç, tıp, otomotiv, finans, eğitim sektörlerini sayabiliriz.

Günümüzde giderek yaygınlaşan ve gelişen bir sektör haline gelen elektronik ticaret, bilgisayar ve internet kullanıcılarının vazgeçemedikleri bir alışkanlık haline gelmiştir. Bilgisayarlar aracılığıyla istenilen ürün veya ürünlere ulaşmak, sipariş vermek mümkün olmaktadır. Kapsam olarak oldukça genişlediği gibi internet mağazalarının müşteri sayıları ve alışveriş kapasiteleri de tahminlerin ötesinde artışlar göstermektedir.

Internet üzerinde alışveriş yapmak isteyenler için büyük kolaylıklar ve geniş imkanların sağlanmaya çalışıldığı günümüzde, sadece internet bağlantısı olan bir bilgisayar ile elektrikli ev aletlerinden kaset ve CD' ye, gıda maddelerinden otomobile kadar her şeyi satın almak mümkün hale gelmiştir. İnsanlara da daha kolay geldiği için, bazı elektronik ticaret siteleri oldukça rağbet görmektedir.

İşte bu rekabet içerisinde, ürünlerini internet üzerinden satışa sunan firmalar, farklı satış stratejileri ve sunum taktikleri geliştirerek birbirlerine üstünlük sağlamaya çalışmaktadırlar. Böyle bir ortamda veri madenciliğinin önemi daha da artmıştır. Çünkü benzer kalite ve fiyattaki ürünlerin satıldığı bir pazarda, mevcut müşterilerinin alışveriş alışkanlıklarını, ürünlerin satış grafikleri ve desenlerini ya da müşteri sınıflarının belirlenmesi ve bu sonuçlara göre karar mekanizmalarının çalıştırılması, ilgili ticari kuruluşun rekabet edebilmesi ve hayatta kalabilmesini sağlayacaktır.

Günümüzde, milyonlarca ticari faaliyet gerçekleşmekte ve bu faaliyetlerde milyarlarca müşteri yer almakta. Bu ticari işlemlerin sonucunda da büyük ölçeklerde veri toplanmaktadır. Yakın zaman öncesine kadar, bu veriler sadece stoklanmak üzere kaydedilmiş ve bir daha kullanılmamışlardır. Ancak, ticari rekabet arttıkça ve ürün satmak zorlaştıkça, bu verilere yeniden başvurulmuş ve bazı gizli özelliklerin ortaya çıkarılmasına uğraşılmıştır. Böylelikle veri madenciliği ortaya çıkmıştır.

Ticari işlemler sırasında, iş sahasında büyük riskler ve büyük fırsatlar yer almaktadır. Örneğin müşterinin fatura ücretini ödeyip ödeyemeyeceği kesin değildir ve satıcı firma bu riski göze almalıdır. Ya da bir müşterinin ne kadar para harcayacağı ya da bundan sonra ne alabileceği gibi konularda büyük fırsatlar çıkabilir

Örnek olarak, büyük bir süpermarketin en basit fatura kayıtları incelendiğinde, traş bıçağı alan müşterilerin %56 sının kalem pil de aldığı ortaya çıkmıştır. Buna dayanarak firma, traş bıçağı ve kalem pil reyonlarını bir araya getirmek suretiyle kalem pil satışlarını %14 arttırmıştır. Bu ve buna benzer örnekler her zaman karşımıza çıkacaktır. Ürünler ve satışları arasındaki bu ilişkilerin belirlenmesiyle, satış stratejileri değiştirilip kazancın artırılması mümkündür.

2. BULANIK KÜMELER VE BULANIK MANTIK

Klasik Mantık tüm önermelerin bir ve sıfır gibi iki değerden oluşan bir sistem üzerine kurulmuştur. Önermeler, insan düşünce sisteminin tersine, sadece iki değer ile adlandırılırlar; doğru/yanlış. Bulanık Mantık, doğru/yanlış, evet/hayır, olumlu/olumsuz gibi iki değerli ilişkilendirmelerin ötesinde, üyelik ilişkilerinin tanımlanabildiği çok değerli bir mantık sistemidir.

2.1 Bulanık Mantık

Bilgisayarlarda sıfır ve bir dizilerine indirgenmiş kesin gerçekler ve doğru yada yanlış olan önermeler kullanılır. İnsan beyni ise, "uzun boy", "yaşlı hız", "genç kız" gibi belirsizlik ve bulanık anlatım içeren yapıları anlamlandırabilir .

Bulanık mantık, insanın düşünürken kullandığı yapıları bilgisayarda modellemek için kullanılır Bulanık mantık, "sıcak" ya da "soğuk" gibi kavramlar kullanır ve bu sayede, hangi hızla çalışacağına ya da programlandığı bir aşamadan diğerine ne zaman geçeceğine kendisi karar veren havalandırma, çamaşır makinesi ve benzeri aygıtları yapabilmeleri için mühendislere yardımcı olur. Matematikçilerin elinde bir sistemin girdilerine yanıt verecek özel algoritmalar bulunmadığında, bulanık mantık belirsiz niceliklere başvuran kuralları kullanarak sistemi denetleyebilir. Bilinen hiçbir matematiksel model bir kamyonun yükleme yerinden park yerine gidişini, kamyonun hareket noktası rasgele seçilebiliyorsa yönetemez. Oysa gerek insan, gerekse bulanık mantık sistemleri "Kamyon biraz sola dönerse sende biraz sağa çevir" gibi pratik, ancak kesinlik taşımayan kurallar kullanarak bu doğrusal olmayan kılavuzluk işlemini gerçekleştirebilir.

Bulanık mantığın uygulama alanları bu tür kontrol sistemlerinin de ötesine uzanmaktadır. Geliştirilen son teoremler bulanık mantığın ilke olarak, ister mühendislik, ister fizik, ister biyoloji ya da ekonomi olsun, her türlü konuda sürekli sistemleri modellemek üzere kullanılabileceğini göstermektedir. Çoğu alanda, bulanık mantıklı sağduyu modellerinin standart matematik modellerinden daha yararlı ya da kesin sonuçlar verdiği görülmektedir.

Bulanık Teorinin Avantajları

- İnsan düşünme tarzına yakın olması,
- Uygulanışının matematiksel modele ihtiyaç duymaması,
- Yazılımın basit olması dolayısıyla ucuza mal olması.

Bulanık Teorinin Dezavantajları

- Uygulamada kullanılan kuralların oluşturulmasının uzmana bağlılığı,
- Üyelik fonksiyonlarının deneme – yanılma yolu ile bulunmasından dolayı uzun zaman alabilmesi,
- Kararlılık analizinin yapılışının zorluğu (benzeşim yapılabilir).

2.1.1 Bulanık Mantığın Kullanım Alanları

Bulanık mantık çok çeşitli alanlarda çözümler getirebilir. Bulanık mantık özellikle kontrol sistemlerinde yaygın olarak kullanılır. Aşağıda bulanık mantığın kullanıldığı bazı alanlar özet olarak verilmiştir.

• **Biyoloji ve Tıp Bilimi:** Bulanık mantık tabanlı teşhis sistemleri, kanser araştırmaları hareket bozuklukları analizi vb.

• **Karar Destek Sistemleri ve Yönetimi** Bulanık mantık tabanlı fabrika yeri seçme sistemleri, Bulanık mantık tabanlı ordu karar destek sistemleri, bulanık mantık tabanlı Pazar stratejileri belirleme ve karar destek sistemleri, vb.

• **Ekonomi ve Finans** Karmaşık Pazar analizleri, bulanık mantık tabanlı ticaret sistemleri, bulanık mantık tabanlı en iyi fiyat analizi, bulanık mantık tabanlı yatırım yönlendirme sistemleri, vb.

• **Çevre Bilimi** Bulanık mantık tabanlı hava tahmin sistemleri, bulanık mantık tabanlı su kalite kontrolü, vb.

• **Mühendislik ve Bilgisayar Bilimleri** Bulanık veritabanı sistemleri, bulanık mantık tabanlı deprem tahmin sistemleri, bulanık mantık tabanlı nükleer santral kontrol otomasyonu, bulanık mantık tabanlı bilgisayar ağ dizaynı, bulanık mantık tabanlı mimari dizayn uygulamaları, bulanık mantık tabanlı kontrol sistemleri, vb.

• **Operasyon Araştırmaları** Bulanık mantık tabanlı iş modelleme ve zaman çizelgesi oluşturma uygulamaları, bulanık mantık tabanlı kaynak planlama, vb.

• **Örüntü Algılama ve Sınıflandırma** Bulanık mantık tabanlı konuşma algılama, bulanık mantık tabanlı el yazısı algılama, bulanık mantık tabanlı yüz karakteristiği algılama, bulanık mantık tabanlı ordu idare analizi, bulanık resim araştırması, vb.

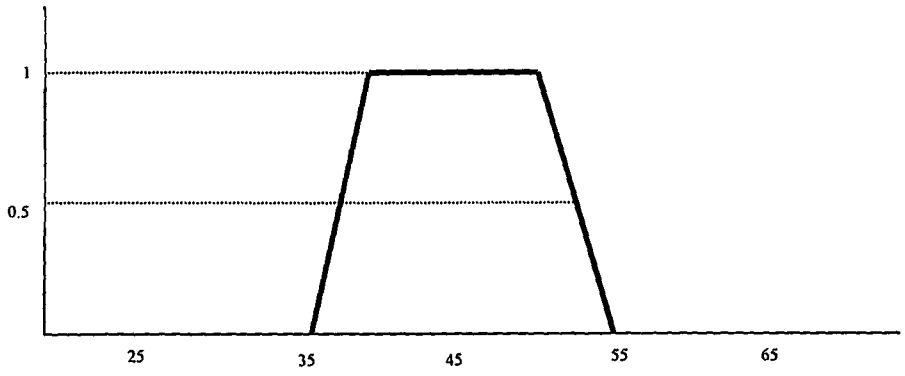
• **Psikoloji** Bulanık mantık tabanlı insan davranış analizi, bulanık mantık tabanlı suç araştırma ve önleme, vb.

• **Güvenilirlik ve Kalite Kontrol** Bulanık mantık tabanlı hata algılama sistemleri, üretim hattı gösterim sistemleri, vb.

Biz daha çok bulanık mantık tabanlı veri madenciliği ve uzman sistem uygulamaları üzerinde duracağız.

2.2 Bulanık Kümeler

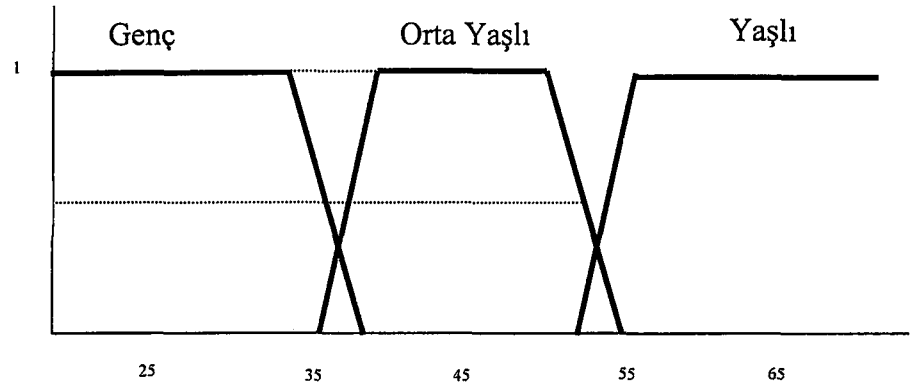
Bulanık teorinin merkez kavramı bulanık kümelerdir. Küme kavrama kulağa biraz matematiksel gelebilir ama aslında anlaşılması çok kolaydır. Örneğin "orta yaş" kavramını yakından inceleyerek yapıyı anlamaya çalışalım. "Orta yaş" denildiğinde her insanın kafasında bir şey canlanır ama kavramın sınırları belirsizdir, Herhangi birine göre 40 yaşını aşmış bir insan orta yaşlı olabileceği gibi, başka birine göre ise bu değer 45 olabilir. Kesin sınırlar söz konusu olmadığı için; kavram matematiksel olarak kolayca formülleştirilemez. Ama genel olarak 35 55 yaşları orta yaşlılık kavramının sınırları kabul edilir. Bu yapı grafik olarak şekil 2.1 deki gibi bir eğri olarak gösterilebilir. Bu eğriye "aitlik eğrisi" adı verilir ve bu kavram içinde hangi değerin hangi ağırlıkta olduğunu gösterir. [4]



Şekil 2.1 Bulanık küme için karakteristik fonksiyon

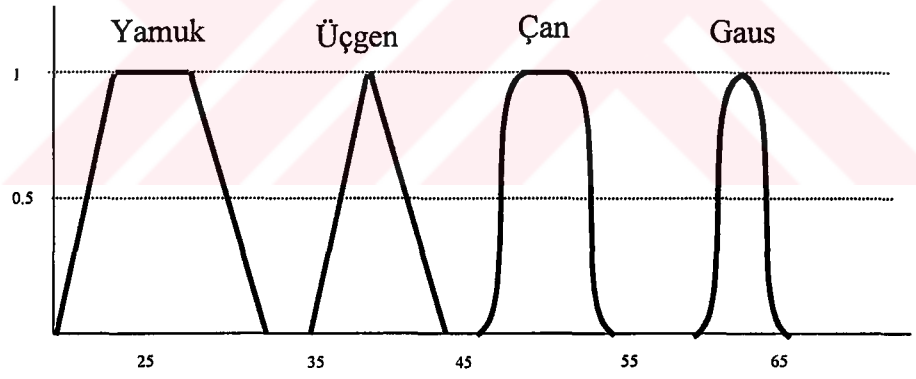
2.2.1 Üyelik Fonksiyonları:

Şimdi bir önceki bölümde anlattığımız orta yaş kavramının yanına genç ve yaşlı kavramlarını da ekleyelim, şekil 2.2 de kullanılan eğriler üyelik fonksiyonu olarak bilinir ve bu fonksiyonların değer kümesi [0,1] aralığındadır.



Şekil 2.2 Bulanık küme için üyelik fonksiyonları

Üyelik fonksiyonları çeşitli şekillerde olabilir. Şekil-2.2 'de kullanılan üyelik fonksiyonu yamuk(trapezoidal) tip üyelik fonksiyonudur. Şekil-2.3'de çeşitli şekillerde üyelik fonksiyonları verilmiştir.



Şekil 2.3 Üyelik fonksiyon tipleri

A bir bulanık küme olsun, ve bu A kümesinin, daha büyük bir E kümesinin (A'nın tanım uzayı) alt kümesi olduğunu düşünelim. A bulanık kümesi, her biri iki elemandan oluşan sıralı eleman çiftlerinden oluşur. Bu eleman çiftlerinin birincisi E'nin bir elemanı olan x,in A içerisindeki üyelik ağırlığı olarak bilinir. E'nin elemanları ile, bunların A içerisindeki üyelik ağırlıkları arasındaki ilişki üyelik fonksiyonu olarak bilinir. Bulanık kümeler üyelik fonksiyonları ile tanımlanır.

$E = [x_1, x_2, x_3, x_4]$ Bir E evrensel kümesi;

olsun.

Bu kümenin alt kümesi $A = \frac{0.1}{x_1} + \frac{0.4}{x_2} + \frac{0.7}{x_3} + \frac{1}{x_4}$ 'dir

Bu küme sadece ağırlıklarıyla gösterilebilir.

$\mu_A = [0.1, 0.4, 0.7, 1]$

İfadesine A'nın üyelik fonksiyonu denir.

2.2.2 Bulanık Küme Terimleri

Bulanık kümelerin başlıca terimleri aşağıda verilmiştir. Şekil 2.4 üzerinde bulanık küme terimleri gösterilmiştir. [2]

Çekirdek (core): Evrensel kümenin, $\underline{A}$ bulanık kümesine tam olarak üye olan elemanları kümesidir. Çekirdekteki elemanların bulanık kümeye üyelik dereceleri 1 'dir. Çekirdeği gösteren denklem 2.1'de verilmiştir.

$$core(\underline{A}) = \{x \in X \mid \mu_A(x) = 1\} \tag{2.1}$$

Sınır (boundry): Evrensel kümenin, $\underline{A}$ bulanık kümesine üye olan ancak üyelik değeri 1 'den küçük olan elemanları kümesidir. Sınırı boş olan bulanık kümeler belirgin küme olarak sınıflandırılırlar. Sınırı gösteren denklem 2.2 de verilmiştir.

$$boundry(\underline{A}) = \{x \in X \mid 0 < \mu_A(x) < 1\} \tag{2.2}$$

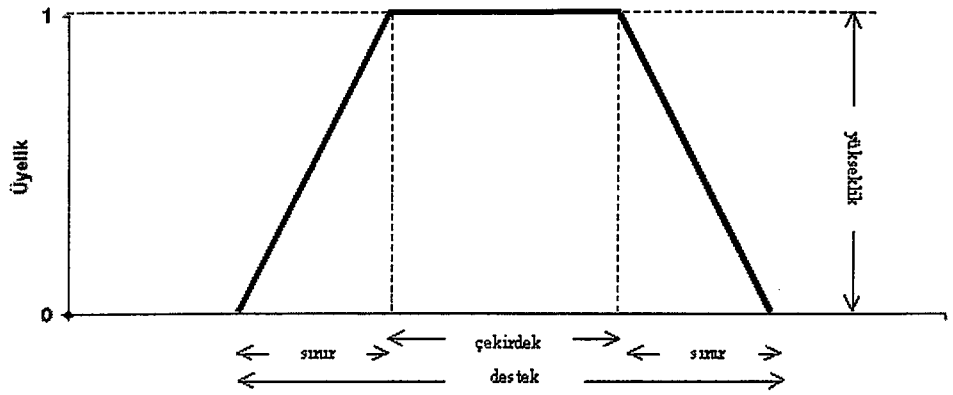
Destek (support): Evrensel kümenin, $\underline{A}$ bulanık kümesine üye olan elemanları kümesidir. Başka bir deyişle bulanık kümenin çekirdek ve sınırımın birleşimidir. Desteği gösteren denklem 2.3 de verilmiştir.

$$core(\underline{A}) = \{x \in X \mid \mu_A(x) > 0\} \tag{2.3}$$

Yükseklik (core): $\underline{A}$ bulanık kümesine üye olan elemanların en yüksek üyelik değeridir. Yüksekliği gösteren denklem 2.4'te verilmiştir.

$$height(\underline{A}) = \max(\mu_A(x)) \tag{2.4}$$

height($\underline{A}$) = 1 olan bulanık kümeler normal küme olarak adlandırılırlar. Fark edileceği gibi normal bir kümenin en az bir elemanı kümeye tam olarak üyedir.



Şekil 2.4 Bulanık küme terimleri

Kardinalite (cardinality): A bulanık kümesine üye olan elemanların üyelik değerleri toplamıdır. |A| ile gösterilir. Kardinaliteyi gösteren denklem 2.5'te verilmiştir.

$$|\underline{A}| = \sum \mu_A(x) \tag{2.5}$$

2.2.3 Bulanık Küme İşlemleri

A ve B , X çalışma evrenli iki bulanık küme olsun ve μ_A (x) ve μ_B (x) üyelik fonksiyonlarıyla

gösterelim. Burada x ∈ X 'dir.

Bulanık kümelere göre temel işlemler aşağıdaki gibi tanımlanır ;

Birleşme Özelliği:

Her x ∈ X için A ∪ B birleşiminin $\mu_{A \cup B}$ (x)üyelik fonksiyonu ;

$$\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} \tag{2.6}$$

Kesişme Özelliği:

Her x ∈ X için A∩B Kesişiminin $\mu_{A \cap B}$(x) üyelik fonksiyonu;

$$\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} \tag{2.7}$$

Tamlayan Özelliği:

Her $x \in X$ için A bulanık kümesinin tamlayanı olan $\mu_{A'}(x)$ üyelik fonksiyonu;

$$\mu_{A'}(x) = 1 - \mu_A(x) \tag{2.8}$$

Birleşim, kesişim ve tamlayan bulanık küme işlemleri boolean cebrindeki VE, VEYA ve DEGIL işlemlerine karşılık gelirler. Sıkça kullanılan bir diğer birleşim ve kesişim işlemi aşağıdaki gibidir;

$$\mu_{A\cup B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\} \tag{2.9}$$

$$\mu_{A\cap B}(x) = \mu_A(x) \cdot \mu_B(x) \tag{2.10}$$

Tablo-2.1 Bulanık küme özellikleri

a)	Birleşme	$\underline{A} \cup (\underline{B} \cup \underline{C}) = (\underline{A} \cup \underline{B}) \cup \underline{C}$
		$\underline{A} \cap (\underline{B} \cap \underline{C}) = (\underline{A} \cap \underline{B}) \cap \underline{C}$
b)	Değişme	$\underline{A} \cup \underline{B} = \underline{B} \cup \underline{A}$
		$\underline{A} \cap \underline{B} = \underline{B} \cap \underline{A}$
c)	Dağılma	$\underline{A} \cup (\underline{B} \cap \underline{C}) = (\underline{A} \cup \underline{B}) \cap (\underline{A} \cap \underline{C})$
		$\underline{A} \cap (\underline{B} \cup \underline{C}) = (\underline{A} \cap \underline{B}) \cup (\underline{A} \cap \underline{C})$
d)	De Morgan Kuralı	$\neg(\underline{A} \cup \underline{B}) = \neg\underline{A} \cap \neg\underline{B}$
		$\neg(\underline{A} \cap \underline{B}) = \neg\underline{A} \cup \neg\underline{B}$
e)	Geçişme	$\underline{A} \subseteq \underline{B}$ ve $\underline{B} \subseteq \underline{C}$ ise $\underline{A} \subseteq \underline{C}$

Bulanık Alt Küme:

A ve B X'de tanımlı iki bulanık küme ve sırasıyla $\mu_{\underline{A}}(x)$ ve $\mu_{\underline{B}}(x)$ bu kümelerin üyelik fonksiyonları olsun.

$$\forall x \in X, \mu_{\underline{A}}(x) < \mu_{\underline{A}}(x) \tag{2.11}$$

şartı sağlanıyorsa A,B 'nin bulanık alt kümesidir denir.

2.2.4 Bulanık Kümelerin Özellikleri

Geleneksel kümeler için geçerli olan özelliklerin bir çoğu bulanık kümeler için de geçerlidir. Temel bulanık küme özellikleri Tablo 2-1'de listelenmiştir.[3]

2.2.5 Dilsel Pekiştiriciler

Kullanılan doğal dilde "çok","az","çok az" gibi eklemeler ile önermeler üzerinde güçlendirici yada zayıflatıcı vurgular yapılır. Dilsel pekiştiriciler anlamı vurguladıkları gibi mevcut bulanık önerme ve kümelerden yeni bulanık önerme ve kümelerin türetilmesine de sebep olurlar. Örneğin uzun boylular bulanık kümesi üzerinde kullanılacak "çok" yada "çok çok" pekiştiricileri ile "çok uzun boylular" ve "çok uzun boylular" gibi iki bulanık küme daha elde etmiş oluruz.[4]

Doğal dil ile söylenen pekiştirici ifadeleri bulanık kümeler aracılığı ile matematiksel olarak ifade edebiliriz. Pekiştirici ifadenin bulanık küme üzerine uyarlanması üyelik fonksiyonunun bir pekiştirici fonksiyonla işleme sokulması ile olur. Dilsel pekiştiricilerin modele uygulanması ile doğruluk değerleri üzerine azaltıcı yada artıcı etki yapılmış olur. Kullanılan matematiksel fonksiyonun seçiminde doğal dil ile ifade edilmek istenen vurgu ile, doğruluk değeri arasındaki ilişkinin korunmasına dikkat edilmelidir.

Aşağıda bazı temel dilsel pekiştiriciler verilmiştir.[4]

a) **Çok:** Çok ile ilgili hesaplama yöntemi denklem 2.12'de verilmiştir

$$\underline{A}^2 = \sum_{i=1}^{n} \mu_A^2(x_i)/x_i \tag{2.12}$$

b) **Aşırı:** Aşırı ile ilgili hesaplama yöntemi denklem 2.13'te verilmiştir.

$$\underline{A}^3 = \sum_{i=1}^{n} \mu_A^3(x_i)/x_i \tag{2.13}$$

c) **Çok Çok:** Çok Çok ile hesaplama yöntemi denklem 2.14'te verilmiştir.

$$\underline{A}^4 = \sum_{i=1}^{n} \mu_A^4(x_i)/x_i \tag{2.14}$$

d) **Biraz:** Biraz ile ilgili hesaplama yöntemi denklem 2.15'te verilmiştir.

$$\underline{A}^{0.5} = \sum_{i=1}^{n} \mu_A^{0.5}(x_i)/x_i \tag{2.15}$$

3. UZMAN SİSTEMLER VE BULANIK UZMAN SİSTEMLER

Yapılacak veri madenciliği uygulamasının sonucunda elde edeceğimiz kural tabanı, çoğu uzman sistemler tarafından kullanılabildiğinden uzman sistemlerin anlatılması gerekmektedir.

Uzman sistemler için iki tanım verebiliriz. Uzman sistemler, belirli bir alanda uzmanlaşmış personel yada çalışma gruplarının iş süreçlerini takip eden yazılım sistemlerinin genelidir. Uzman sistemi insan uzmanlığına dayanarak, karar verme mekanizmalarına sahip bir bilgisayar sistemi olarak tanımlayabiliriz.

Bir uzman sistem basitçe konuyla ilgili bir uzmanın bilgisine sahip olması gerekir. Bu bilgi problemlerin çözümünde kullanılacaktır.

Geleneksel olarak bir program yapısı aşağıdaki gibidir.

algoritma + veri yapıları = program

bir uzman sistem ile bu tanımlar biraz değişiyor:

çıkarım sistemi + bilgi = uzman sistem

Tablo – 3.1 Uzman sistem nerelerde kullanılır

Alan	**Yüzde**
Üretim ve Operasyon Yönetimi	48%
Finans	17%
Bilgi Sistemleri	12%
Pazarlama	10%
Muhasebe ve Kontrol	5%
Uluslararası İlişkiler	3%
İnsan Kaynakları	2%
Diğerleri	2%

3.1 Uzman Sistemin Kullanım Alanları

Uzman sistemler çok çeşitli alanlarda kullanılır. Yapay zeka dünyasında en çok uygulama alanı bulan konulardan biridir. Aşağıdaki tablo uzman sistemlerin çeşitli alanlarda kullanım yüzdelerini göstermektedir.

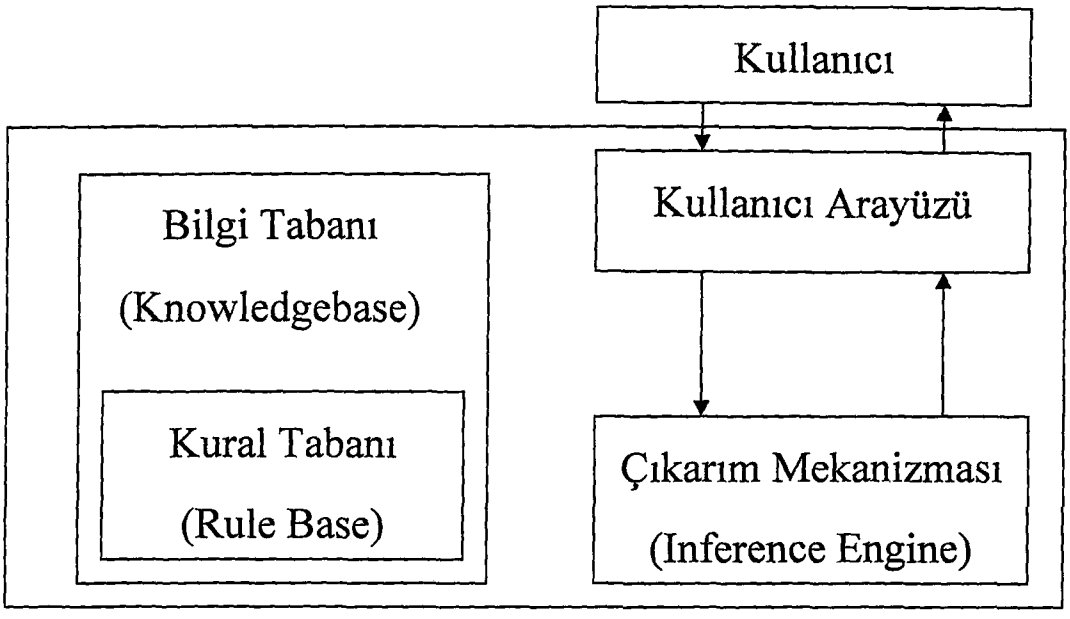
- Uzman sistemin kullanılmasının başlıca nedenleri:
- Kurumsal bilgi üretimine katkıda bulunur ve bilginin paylaşımını sağlar
- Nadir, pahalı ve olmayan bilgiye ulaşılmasını sağlar.
- Zaman kazandırır, Cevap verme süresinin kısaltır.
- Yeni işe başlayan çalışanları eğitir.
- Üretim performansını artırır. Verimlilik artışı sağlar
- Maliyetleri düşürür.

3.2 Bir Uzman Sistemin Bileşenleri

Uzman sistemler üç temel bileşeni vardır. Bunlar,

- Bilgi Tabanı (Knowledgebase)
- Kural Tabanı (Rulebase)
- Çıkarım Mekanizması (Inference Engine)
- Kullanıcı Arabirimi (User Interface)

olarak listelenebilir. Şekil 3.1'de tipik uzman sistem bileşenleri gösterilmiştir.



Şekil 3.1 Bir uzman sistemin bileşenleri

Bilgi Tabanı:

Bilgi tabanının en önemli özelliği kural tabanını tutuyor olmasıdır. Kurallar programların içerisinde tanımlanmaz. Programların içerisinde bu kuralları işleyecek mekanizmalar vardır. Kurallar ise veritabanında tutulur. Uzman sistemlerinde kullanılan kural tabanlarının tek bir standardı yoktur. Her uzman sistem kendi kural tabanını istediği tipte oluşturabilir ama temel olarak bir kural tabanı aşağıdaki bileşenlerden oluşur.

öncül(antecedent)=> netice(consequent)veya Eğer => ise

Biz kural tabanını oluştururken aşağıdaki gibi bir standart kullandık.

Eğer promosyon = 50 ve performans = 60 ise satis = 58

Çıkarım Sistemi:

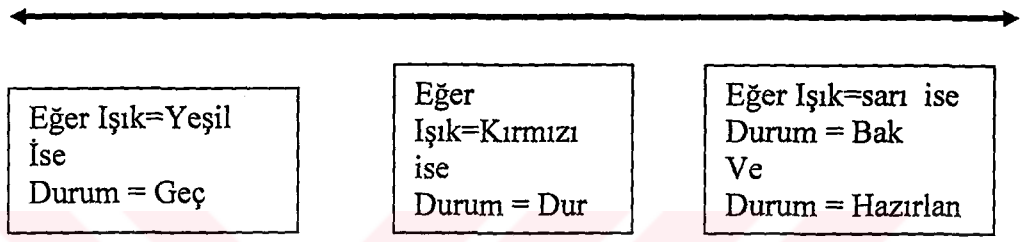
Çıkarım sistemi bilgi tabanında bulunan tüm kuralları tarar. Eğer kullanıcı arabirimi tarafından girilmiş olan gerçekleri sağlayan kuralı bulursa bu kuralı çalıştırır. Eğer netice tarafında bulunan bir değişken öncül tarafında da kullanılmış ise bu işlem kendini yineleyerek devam eder. Böylece birden fazla kural çalıştırılabilir.

Bir kural tabanında bulunan tüm kurallar aynı değerde olmayabilir. Bu kuralların birbirlerinden farklı değerlerde olduğunu ağırlıkları yardımıyla analarız. Her bir kurala, kural tabanını oluşturan uzmanlar veya sistemler tarafından, bir ağırlık

atanır. Bu ağırlık birden fazla kural çalıştırıldığında sonucun hesaplanması için kullanılır. Bu işleme kural birleştirme denir. (Combining)

Uzman istemlerde iki tip zincirleme yöntemi ile çıkarım yapılır. Bunlar ileri doğru(forward) zincirleme ve geri doğru(backward) zincirleme yöntemleridir. İleri doğru zincirleme yönteminde sonuca ulaşılmaya çalışılır. Geri doğru zincirleme yönteminde ise bir hipotezin sağlanıp sağlanmadığını ispat etmek için sonuçtan yola çıkarak kural taramaları yapılır.

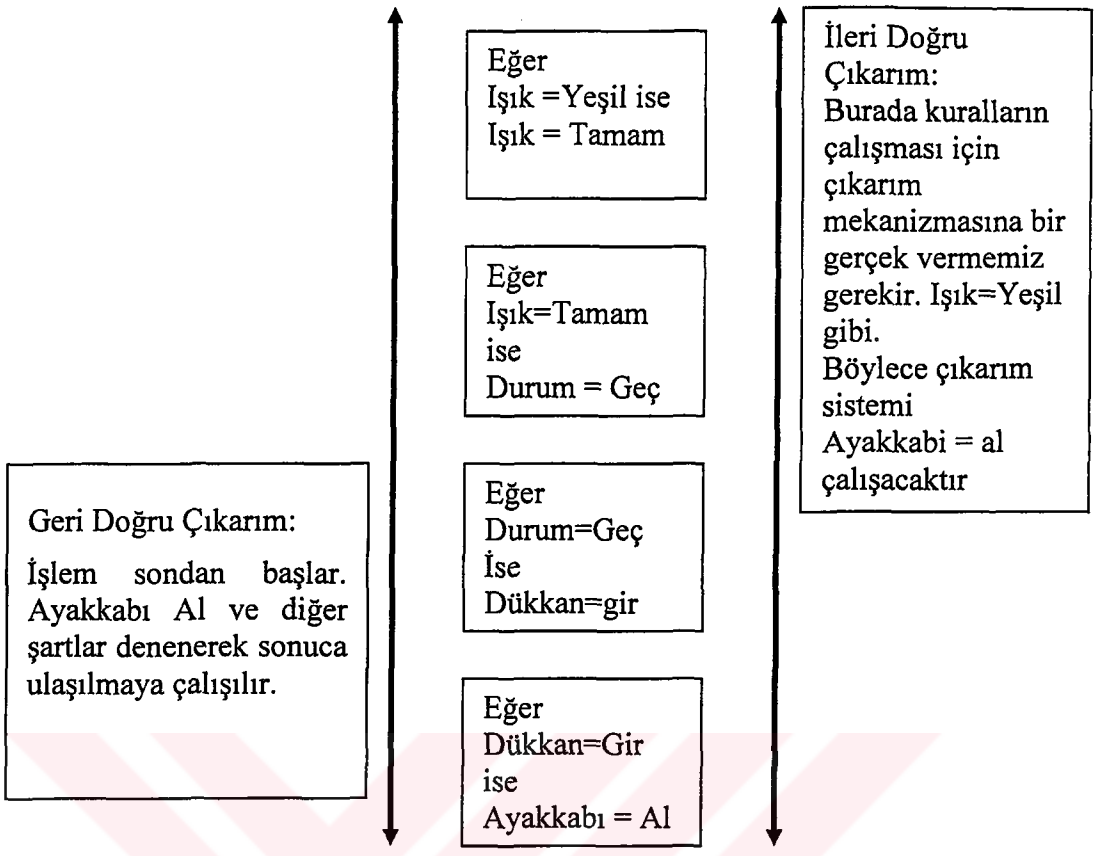
Bilgisayarlarda kullanılan çıkarım mekanizmaları insanlardaki tümden gelim, tüme varım vb. muhakeme çeşitlerine benzetilebilir.



Şekil – 3.2 Aynı Seviyede Çıkarım

Şekil – 3.2'de işe başlama zamanı önemlidir ve derin bir çıkarım yapmaya gerek kalmaz. Yapılacak harekete belli bir anda karar verilir ve uygulanır.

Şekil – 3.3'de ileri ve geri doğru çıkarım yapan sistemler anlatılmaktadır.



Şekil – 3.3 Farklı Seviyede Çıkarım

Belirsizlik

Belirsizlik ile uzman sistemlerde pek çok değişik sonuca varılabilir. Kurallara verilen Kesinlik faktörü bunlardan biridir. (Certainty Factor)

Kesinlik Faktörü: Bir kural doğruluk derecesini gösteren bir değer olarak düşünülebilir. Matematik olarak bir kuralın ağırlığı o kurala inanmamanın ölçüsüdür. (measure of disbelief)

Örnek olarak

Eğer Işık=Yeşil

ise

Durum=Geç kf=0.9

Bu kuralın söylediği; eğer ışık yeşil olursa ben %90 kesinlikte karşıya geçeceğim.

Kesinlik faktörlerinin belirli avantajları ve dezavantajları vardır. Hesaplanmaları ve sistemde belirsizliği modellemesi kolaydır. Ama bu değerlerin belirlemesi biraz geri beslemeli olarak yapılmalıdır.

Dempster-Shafer Teorisi:

Uzman sistemlerin belirsizlik faktörünün uygulanması için yöntemlerden biri Dempster-Shafer teorisidir. Örnek olarak bir sınıftaki insanların uyuması için inanç faktörü 0.7 ise bu 0.3 oranında bu sınıftakiler uyumayacak anlamını taşımaz.

Bayes Teoremi:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

Bayes teoremi E olayı olduğunda H olayının olma olasılığını verir. Bayes teoremi büyük sistemler için pratik değildir.

3.3 Bulanık Uzman Sistemler

Bulanık uzman sistemler uzman sistemlerde kullanılan kurallarda bulanık ifadelerin kullanılması sonucu ortaya çıkmıştır. Bulanık uzman sistemlerde veri tabanındaki değişkenler bir bulanıklaştırıcı yardımıyla bulanık değerlere dönüştürülürler.

Örneğin aşağıdaki kuralı düşünecek olursak;

Eğer boy>1.85 ise meslek=basketbolcu

önermesi;

Eğer boy=çok uzun ise meslek=basketbolcu

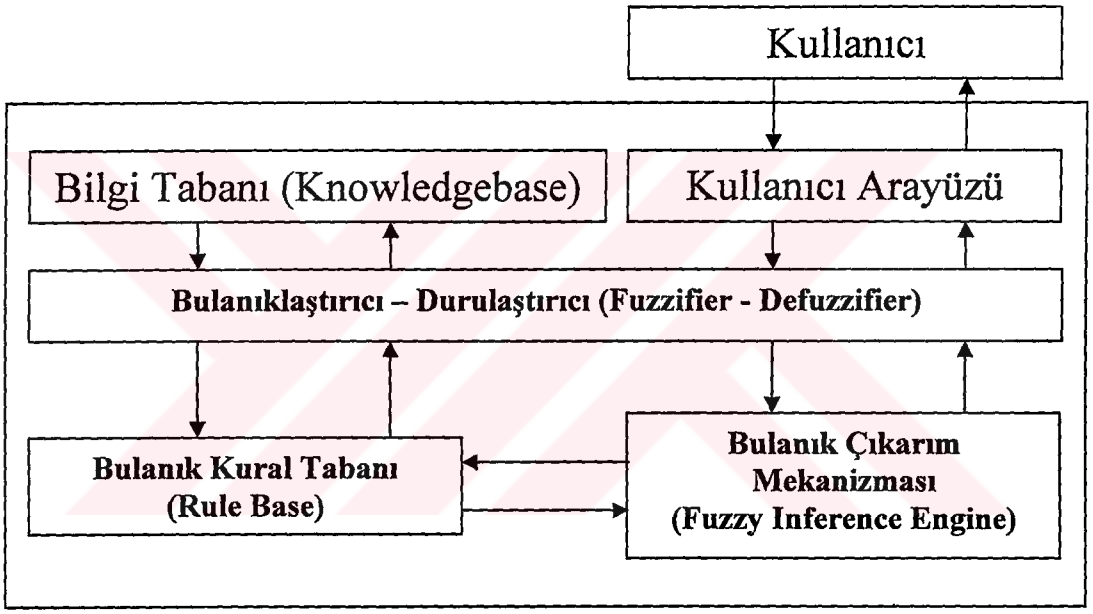
şekline dönüştürülebilir. 1.80'den uzun insanları bir üyelik fonksiyonu yardımı ile çok uzun olarak tanımlarsak yukarıdaki önermeye kolayca ulaşırız.

3.3.1 Bulanık Uzman Sistemlerin Bileşenleri

Bulanık uzman sistemler üç temel bileşeni vardır. Bunlar,

- Bilgi Tabanı (Knowledge Base)
- Bulanık Kural Tabanı (Fuzzy Rulebase)
- Bulanık Çıkarım Mekanizması (Fuzzy Inference Engine)
- Bulanıklaştırıcı – Durulaştırıcı (Fuzzifier-Defuzzifier)
- Kullanıcı Arabirimi (User Interface)

olarak listelenebilir. Şekil 3.4'te tipik uzman sistem bileşenleri gösterilmiştir.



Şekil -3.4 Bulanık Uzman Sistemin Bileşenleri

4. VERİ MADENCİLİĞİ ARACI

Veri madenciliği aracı ile bir veritabanı üzerinde belirlenmiş alanlar sınıflandırılarak bulanık uzman sistemde kullanılmak üzere bir kural tabanı çıkarması amaçlanmıştır. Veri madenciliği ile büyük veritabanlarından örneklemeler yaparak ve örüntüleri yakalayarak kurallar çıkarabiliriz. Veri madenciliği yaklaşımı kural çıkarımı, kümeleme, sınıflandırma, tahmin etme vb. üzerine kuruludur. Bir kural X ve Y bir kümenin elemanları olmak üzere X=>Y olarak gösterilir. Bu kural veritabanında X içeren kayıtlardan ve Y'yi içeren kayıtları vermeye yöneliktir. Agrawal, Imielinski & Swami 1993; Agrawal & Srikant, 1995; Anand, Patrick, Hughes & Bell, 1998; Bayardo, 1998; Goulbourne, Coenen & Leng 2000; Han & Fu, 1999; Houtsma & Swami, 1993; Mastsuzawa & Fukuda, 2000 yıllarında veri madenciliği hakkında bir çok algoritma ileri sürdüler.

Kümeleme heterojen verilerin daha homojen alt veri grupları haline getirmek olarak tanımlanır. Sınıflandırma yeni saptanmış veri grubunun daha önceden tanımlanmış veri grupları cinsinden gösterilmesi anlamını taşır. Tahmin etme ise bilinmeyen değişkenler üzerinden bir saptama yapmaktır. Pek çok sayıda kümeleme algoritması vardır, otomatik küme belirleme, K-means metod; agglomerative algoritması; divisive metodu; self-orginizing-maps bunlardan birkaçıdır. Bunlardan self-orginizing-map (SOM), yapay sinir ağları kullanarak veri içerisindeki örüntüleri yakalamaya çalışır ve bunları gruplar. SOM Kohonen tarafından 1990 yılında ileri sürüldü. Temel bir SOM ağı bir giriş katmanı ve çıkış katmanından oluşur. Bir SOM ağı veri kümeleri tarafından beslendiğinde, SOM hangi çıkış kümesinin hangi giriş kümesine ait olduğunu bilir.

Kümeleme tekniklerinden biri olan SOM farklı ve birbirinden tamamen ayrı olan verileri bir kümenin elemanları olacak şekilde bir araya toplar. Fakat SOM ağlarının geri dönüşünde sonuçları kesin olarak almak mümkün değildir. Sonuçları hesaplayabilmek için başka tekniklere ihtiyaç duyulur. Ha ve Park 1998 yılında karar ağaçları tabanlı bir sınıflandırma algoritması ileri sürmüşlerdir. Sonuçta çıkan rakamların anlamlandırılması biraz karmaşık olduğundan, biz her kümeyi anlatabilmek için küme ile ilgili kuralları çıkaracağız. Her kümeye bir kural karşılık getireceğiz. Bu tür sistemlerde bilgi kural tabanları ile gösterilir.

Rough küme teorisi Pawiak tarafından 1982'de ortaya atıldı. Bu teori aşağı ve yukarı kümeler yardımı ile yakınsama işlemleri yapar. Bu teori kümeler teorisinin bir uzantısı olarak görülebilir. Verinin tam olmadığı ve eksik olduğu durumlarda çalışan ve bu bilgiyi tamamlayan zeki bir sistem olarak düşünülebilir.

U boş olmayan, sonlu bir evrensel küme olsun. I, U kümesine denk bir bağıntı olsun. I(x) x elemanı içeren bir I bağıntı sınıfı olsun. Bu sınıf üzerinde iki temel işlem tanımlanır. I-lower (4.1) ve I-upper (4.2) yaklaşım metotları aşağıdaki gibi tanımlanır.

$$I_*=\{x \in U \mid I(x) \subseteq X\} \tag{4.1}$$

$$I^*=\{x \in U \mid I(x) \cap U \neq \Phi\} \tag{4.2}$$

Bir küme belirlenirken bu küme ile ilgili güvenilirlik fonksiyonu(Confidence Function) hesaplanır. CF denklem (4.3)'deki gibi tanımlanır.

$$CF(x) = \frac{Num(X \cap I(x))}{Num(I(x))} \tag{4.3}$$

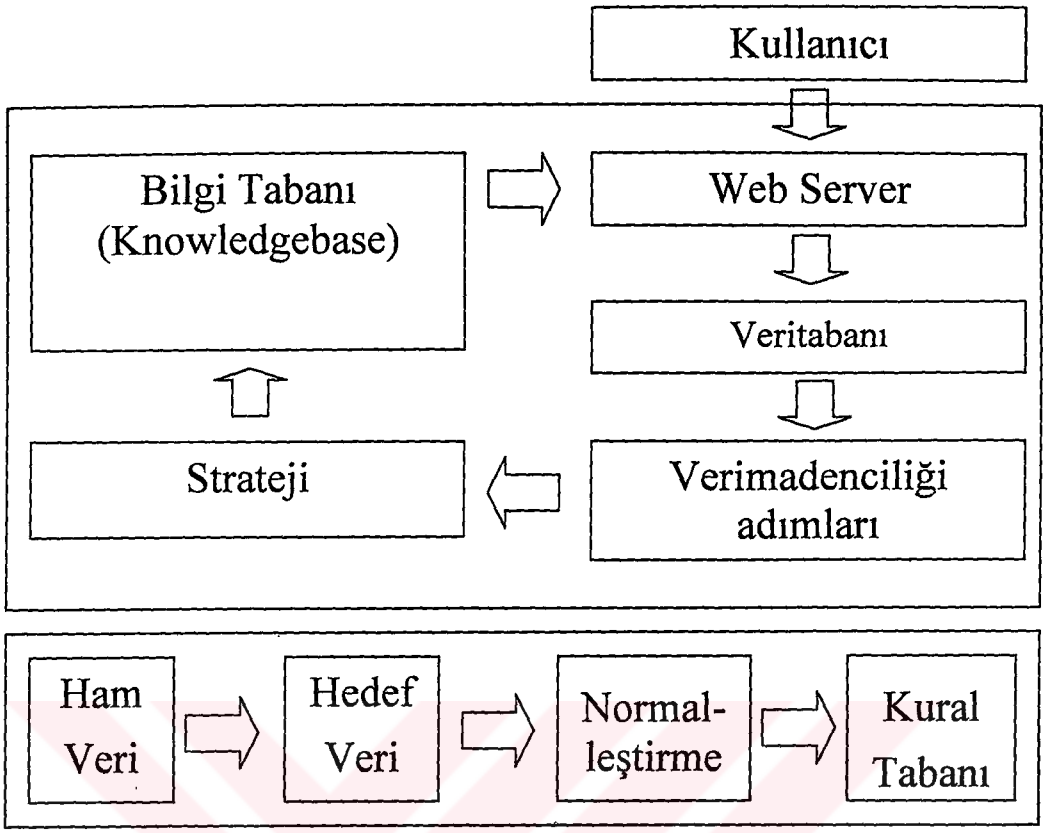
$CF(x) \in [0,1] Num(X \cap I(x))$ tüm değerler için eşzamanlı olarak hesaplanır. (4.1) ve (4.2)'de tanımlanan eşitlikler aşağıdaki gibi tekrar tanımlanır.

$$I_*=\{x \in U \mid CF(x) = 1\} \tag{4.4}$$

$$I^*=\{x \in U \mid CF(X) > 0\} \tag{4.5}$$

CF değeri x elemanının X kümesine I bağıntısı ile bağlılık derecesini göstermektedir. Bu değer bir x değerinin X kümesine olan aitlik değeridir.

Veri madenciliği aracı iki prosedürden oluşur. Birincisi, bir yapay sinir ağı uygulaması olan ve verileri sınıflandıran Self-Organizing-Map (SOM) bir diğeri ise rough küme teorisine dayanan kural çıkarım modülü.



Şekil-4.1 Veri Madenciliği Adımları

4.1 Veri Madenciliği Adımları

Veri madenciliği birkaç adımdan oluşur. Şekil-4.1'de gösterilen adımları kısaca anlatalım. Bu tür sistemlerde veritabanına verilere şekilde olduğu gibi kullanıcılar tarafından girilebilir ya da kontrol sistemlerinde olduğu gibi cihazlardan gelebilir. Biz burada web üzerinden verilerin oluşturulduğunu düşünelim. Bir elektronik ticaret sitesini ele alalım ve bu siteden alışveriş yapan müşteriler ile ilgili bilgileri veritabanında tutalım. Bu uygulamayı müşteri alışkanlıklarını ve ihtiyaçlarını öğrenebileceğimiz bir veri madenciliği uygulaması olarak düşünelim. veri madenciliği adımlarından sonra kurallar otomatik olarak çıkarılacaktır. Bu kurallar müşteriler tarafından girilen bilgiler yani gerçekler ve veritabanının veri madenciliği adımlarından geçirildikten sonra ortaya çıkmış örüntülerin veya oluşmuş bilgi tabanından çıkarılacaktır. Bu kurallar pazar stratejileri, promosyonlar ve hedef pazar seçimlerini belirlerken kullanılabilir.

Tablo – 4.1 Müşteri Kayıtları

Müşteri ID	Öğrenim (E)	İş(J)	GID
Mem^1	N	N	B
Mem^2	N	H	A
Mem^3	N	H	B
Mem^4	L	L	C
Mem^5	H	H	A
Mem^6	H	H	A
Mem^7	L	N	C

Tablo – 4.2 Sonuç Sınıfı

K	GID	X_k
1	A	$\{Mem^{(2)}, Mem^{(5)}, Mem^{(6)}\}$
2	B	$\{Mem^{(1)}, Mem^{(3)}\}$
3	C	$\{Mem^{(4)}, Mem^{(7)}\}$

Tablo – 4.3 Neden Sınıfı

A_{ij}		Y_{ij}
i=E	j=N	$Y_{EN} = \{Mem^{(1)}, Mem^{(2)}, Mem^{(3)}\}$
i=E	j=H	$Y_{EH} = \{Mem^{(5)}, Mem^{(6)}\}$
i=E	j=L	$Y_{EL} = \{Mem^{(4)}, Mem^{(7)}\}$
i=J	j=N	$Y_{JN} = \{Mem^{(1)}, Mem^{(7)}\}$
i=J	j=H	$Y_{JH} = \{Mem^{(2)}, Mem^{(3)}, Mem^{(5)}, Mem^{(6)}\}$
i=J	j=L	$Y_{JL} = \{Mem^{(4)}\}$

4.1.1 Adım – 1: Seçim ve Örnekleme

Bir veritabanı detaylı ve geçmişe dönük verilerden oluşur. Adım 1 hedeflenen veritabanı tablolarını, alanlarını ve özelliklerini seçmek, bunları veri madenciliği aracı için tekrar kaydetmektir. Bu adım dört aktivite içerir. Bir gerçek tablosunun yaratılması, neden ve sonuç alanlarının belirlenmesi, boyutların nişliklerinin belirlenmesi, veri filtrelenmesi.

1. Gerçek tablosunun belirlenmesi: Genelde, büyük veritabanlarında pek çok tablo vardır. Ama bunlardan göreceli olarak gerçek tabloları veri madenciliğine daha yakındır. Örneğin bir üretici tablosu, bir ürün tablosu, bir satış tablosu, ve bir de müşteri tablosu olsun. Bir gerçek tablosu bu dört tablodan alanlar içerir. Kullanıcı daha sonra bu gerçek tablosundan incelemek istediği alanları seçecektir.

2. Boyut seçimi: Veritabanındaki kayıtların analizi ve kural çıkarılması istendiğinde boyutlar, gerçek tablosundan seçilmelidir. Tüm boyutların çeşitli kombinasyonlarda incelenmesi analistlerin ilgisini çekmektedir. Örneğin bir yönetici müşterilerini sınıflandırmak isteyebilir.

3. Nitelik seçimi: Seçilen tablo genellikle pek çok özellik içerir. Ama tüm bu niteliklerin göz önünde tutulacağı anlamına gelmez. Bir yönetici için veya rapor alacak başka bir kullanıcı için farklı seviyede nitelik ağırlıklarını belirleyerek analiz yapılması sağlanabilir.

4. Veri Filtrelemesi: Büyük veri tabanlarından kural tabanı çıkarılması işlemi çok uzun zaman alabilir. Burada zamandan kazanmak için ve veritabanının büyüklüğüne göre veri örneklemesi veya veri filtrelemesi yapılabilir. Burada bazı boş alanlar alınmayarak hata ayıklaması yapılabilir. Veri örneklemesi çeşitli yöntemlerle yapılabilir. Bu yöntemlerden biri rasgele kayıt belirlemedir.

4.1.2 Adım – 2: Normalleştirme

Eğer gerekli ise veri kümelerini veri madenciliği aracına göndermeden önce normalleştirme işlemi yapılır. Biz burada, verileri 0,1 kapalı aralığında bir değere karşılık getirecek şekilde normalleştirme işlemi yaptık. Aşağıda bu iki verinin nasıl normalleştirildiği ile ilgili iki formül verdik. Denklem (4.6) normalleştirme işlemini göstermektedir.

1. Eğer alan sayısal bir değer ise;

$$sonuc\_\deg eri_{jk} = \frac{(\deg er_{jk} - \min(alan\_\deg eri_j))}{(\max(alan\_\deg eri_j) - \min(alan\_\deg eri_j))} \qquad (4.6)$$

$sonuc\_\deg eri_{jk}$ j numaralı alanın k numaralı kayıdı.

$\min(alan\_\deg eri_j)$ j numaralı alanın minimum değeri.

$\max(alan\_\deg eri_j)$ j numaralı alanın maximum değeri.

2. Eğer alan sayısal bir değer değil ise veri dönüşümü için bir yöntem modellenir. Örneğin alan karakter olarak tanımlanmış 'iş' ise bu alanı sayısal bir değere dönüştürmek gerekir. Bütün değerler aynı ölçü aralığında modellenmiş ise kural çıkarım aşamasında kullanılabilir.

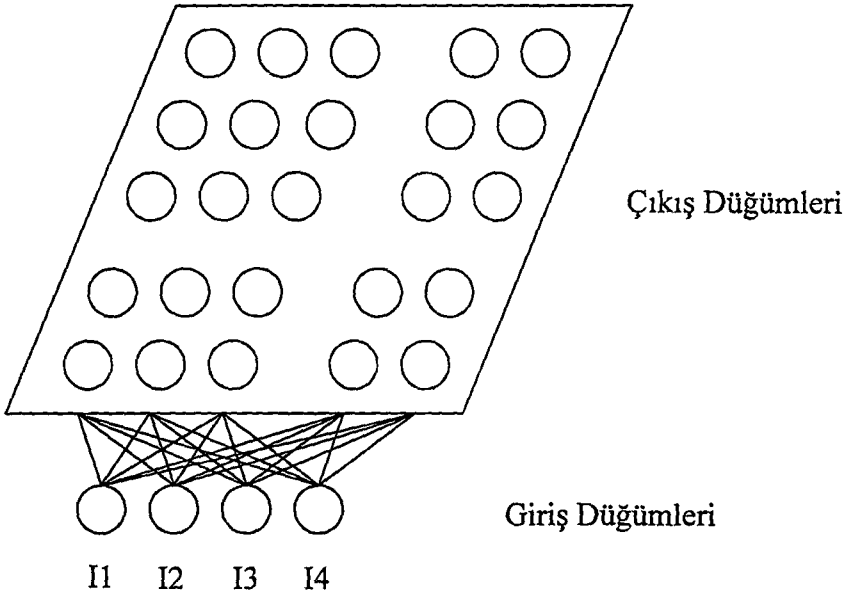
4.1.3 Adım – 3: Birliktelik Kuralları

Kural çıkarımı yapılmadan önce örneklenmiş veriler kümelenir. Bunun için burada yapay sinir ağları tabanlı kümeleme algoritması kullanılmıştır. Kümeleme algoritmasından sonra rough küme teorisi tabanlı kural çıkarım modülü kuralları çıkarır. Farklı alanların arasındaki benzer ilişkilerin karakteristiğinden sınıflar oluşturulur.

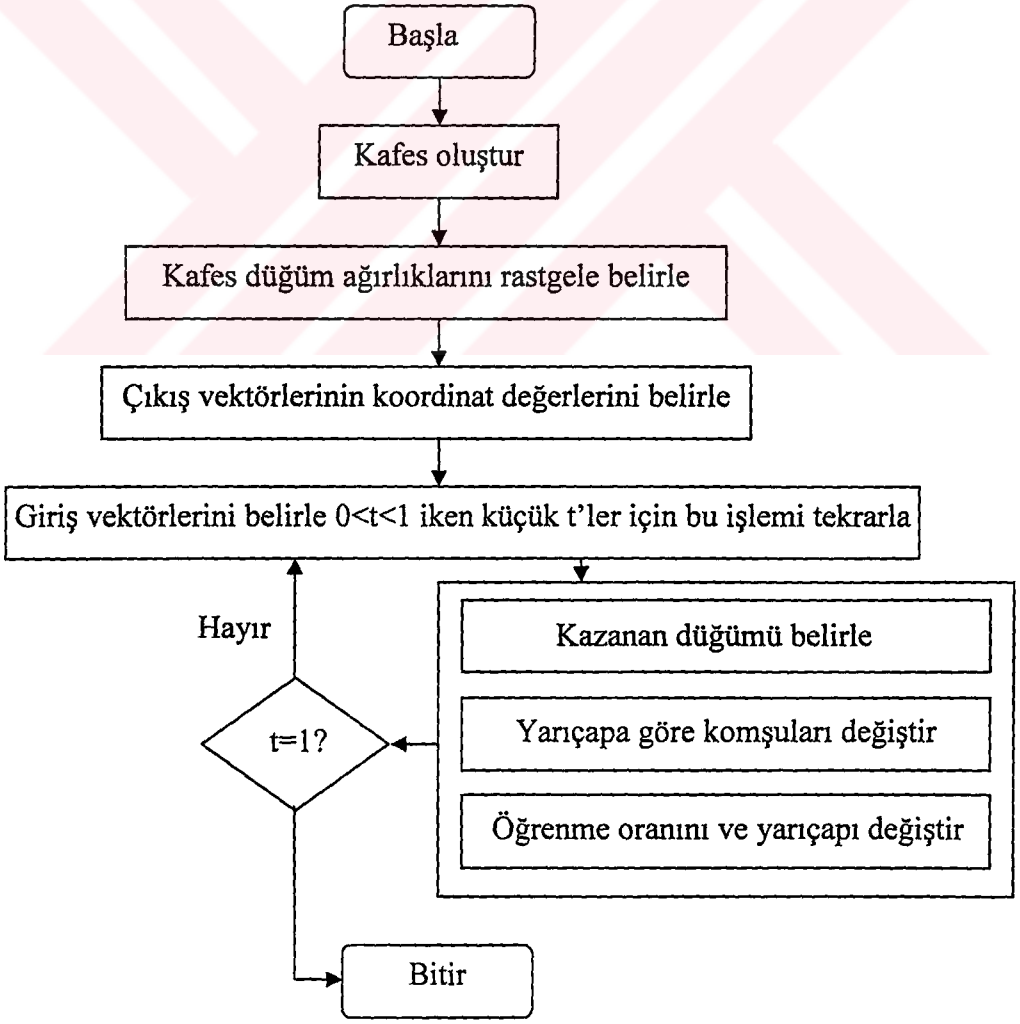
1. Kümeleme Modülü:

SOM 1980 yılında Kohonen tarafından ileri sürüldü. SOM rasgele başlangıç noktası olan, giriş, çıkış vektörleri bulunan bir topolojik haritadır. SOM sonucunda giriş vektörleri arasındaki doğal ilişkileri gösteren bir çıkış vektörü elde edilir. SOM ile giriş vektöründe kullanılmış alanlar kümelere ayrılır. Her farklı küme ile farklı sonuçlar çıkarabilecek kural tabanına ulaşabiliriz.

Bir SOM ağında giriş ve çıkış vektörleri birbirlerine tamamen bağlıdır. Her giriş düğümü çıkış düğümlerine ağırlıkları ile katkıda bulunurlar. Şekil 4.2 yapay sinir ağı yapısını göstermektedir. Oluşturulan sistemde farklı çıkış düğümlerine, öğrenme oranı, komşuluk yarıçapı atanır. Örneğin biz iki alan belirleyelim, Eğitim ve İş. Bu alanlardaki kayıtları normalleştirme işleminden sonra SOM giriş vektörleri olarak gönderelim. Bu vektörlerin networke gönderiliş sırası rasgele olarak belirlenir. Burada SOM'da bulunan kafes yapısı rasgele değerlerden oluşturulur. SOM ağı ilk önce kazanan düğümü belirleyecektir. Kazanan düğüm daha sonra kendi komşuluk yarıçapına göre kendi çevresinde bulunan komşularını değiştirir. SOM ağına tüm giriş vektörleri bittikten sonra bu işlem belli sayıda tekrarlanır. Yeterli sayıda bu işlem tekrarlandıktan sonra öğrenme işlemi tamamlanmış olur. Böylece ağ kendi ağırlıklarına göre çeşitli kümelere ayrılmış olur. Her bir veri kendi ilgili olduğu sınıfa yerleşmiş olur. Şekil 4.3 SOM öğrenme algoritmasını göstermektedir.



Şekil – 4.2 SOM yapay sinir ağı yapısı



Şekil – 4.3 SOM öğrenme algoritması

2. Kural Çıkarım Modülü:

Rough küme teorisi her homojen küme üzerinden bir kural çıkarabilir. Ayrıca her farklı küme arasındaki ilişkileri de belirleyebilir. Tablo 4.1 kümeleme üyelerini ve sonuçları göstermektedir.

Normalleştirilmiş ve sınıflara ayrılmış alanlardan GID, sınıf numarasını ve Mem^i müşteri numarasını gösteriyor. İlk önce rough küme teorisi kullanılarak her bir sınıfın karakteristiğine göre kurallar çıkarılır.

4.1.4 Adım – 4: Sınıfların Karakteristiği

1. Sonuç sınıfını oluştur. X_k sonuç sınıfını göstersin, k seçilen tablonun boyutu olsun. Tablo 4 sonuç sınıfını göstermektedir.

2. Neden sınıfını oluştur. Y_{ij} neden sınıfını göstersin. Bu nesnelerin kümesi belirli bir i alanına karşılık bir j değerlerlerinin kümesi olarak tanımlanır. Tablo-4.2 neden sınıfını gösterir.

3. Lower yaklaşım metodunu kullanalım. $A_{ij}^{X_k}, A_{ij}$ ile aynı özellikte olan tüm elemanların sınıfı olsun.

Örneğin GID=A ise

$X_1 = \{Mem^{(2)}, Mem^{(5)}, Mem^{(6)}\}$ (Tablo – 4.2)

$A_{EN*}^{X_1} = \{Mem^{(2)}\},$

$A_{EH*}^{X_1} = \{Mem^{(5)}, Mem^{(6)}\},$

$A_{EL*}^{X_1} = \{\Phi\}$

$A_{JN*}^{X_1} = \{\Phi\}$

$A_{JH*}^{X_1} = \{Mem^{(2)}, Mem^{(5)}, Mem^{(6)}\}$

$A_{JL*}^{X_1} = \{\Phi\}$

$A_{EH*}^{X_1}$ boş küme olmadığından lower yaklaşım kuralına göre

Kural1: Eğer Öğrenim=H ise GID=A

kuralı çıkar. Her lower yaklaşım metoduna göre bulunmuş kurallara güven %100 dür.

4. Upper yaklaşım metodunu kullanalım. $A_{ij}^{X_k}, A_{ij}$ ile aynı özellikte olan tün elemanların değil bazı elemanların sınıfı olsun. (4.2) ve (4.3) de verilen denklemleri kullanarak CF değerlerini hesaplayalım.

4.1. Upper yaklaşım metodunu kullanalım. GID=A için Y_{EN}, Y_{JH}, X_1 tarafından içerilir. Bunun anlamı neden sınıfında Öğrenim=N ve İş=H olan kayıt eleman sayısı, sonuç sınıfı X_1'de olandan daha fazla. X_1 için upper yaklaşım metoduna göre iki kural çıkarıyoruz. Şekil 4.6 A kümesi için kural çıkarımını göstermektedir. Bunlar;

Kural2: Eğer Öğrenim=N ise GID=A

Kural3: Eğer İş=H ise GID=A

4.2. Her upper yaklaşım kuralının CF değerleri hesaplanır. Bu değeri hesaplamak için eşitlik (4.3) kullanılır. Sistem analistleri CF için bir threshhold(minimum CF değeri) değeri belirlerler. Örneğin Kural2: Eğer Öğrenim=N ise GID=A, bu kuralın kesinlik değeri 1/3' tür. Şekil 4.5 X_1 için EN sınıfında kural bulunmasını göstermektedir.

$$CF(A_{EN}^{*X_1}) = \left(\frac{1(i.e., Mem^{(2)})}{3(i.e., Mem^{(1)}, Mem^{(2)}, Mem^{(2)})} \right) = \frac{1}{3} \approx 0.33$$

Kural3: Eğer Job=H ise GID=A, bu kuralın kesinlik değeri ¾ 'tür.

$$CF(A_{JH}^{*X_1}) = \left(\frac{3(i.e., Mem^{(2)}, Mem^{(5)}, Mem^{(6)})}{4(i.e., Mem^{(2)}, Mem^{(3)}, Mem^{(5)}, Mem^{(6)})} \right) = \frac{3}{4} \approx 0.75$$

Eğer minimum CF değeri olarak 0.75 alınırsa sadece Kural3 geçerli olacaktır.

5. Yeni bir birleşim kuralı tanımayalım. Yukarıda sadece bir alanın bulunduğu kurallar ürettik. Bununla beraber veritabanındaki bağıntılar bir veya birden fazla alanın bulunduğu kurallar içerebilir. Eğer iki kural aynı sonucu üretiyor ise bu kuralları birleştirebiliriz. Bu birleşim aşağıdaki gibi yapılır.

$$CF(A_{EN,JH}^{X_1}) = Min\left(\frac{1}{3},\frac{1}{4}\right) = \frac{1}{4} = 0.25$$

Bu kural aşağıdaki gibi tanımlanır.

$$CF(x) = Min\left[\frac{Num(X_i \cap I(x))}{Num(I(x))},\frac{Num(X_j \cap I(x))}{Num(I(x))},...,\frac{Num(X_i \cap I(x))}{Num(I(x))}\right] \quad (4.7)$$

$CF(x) \in [0,1]$ iken.

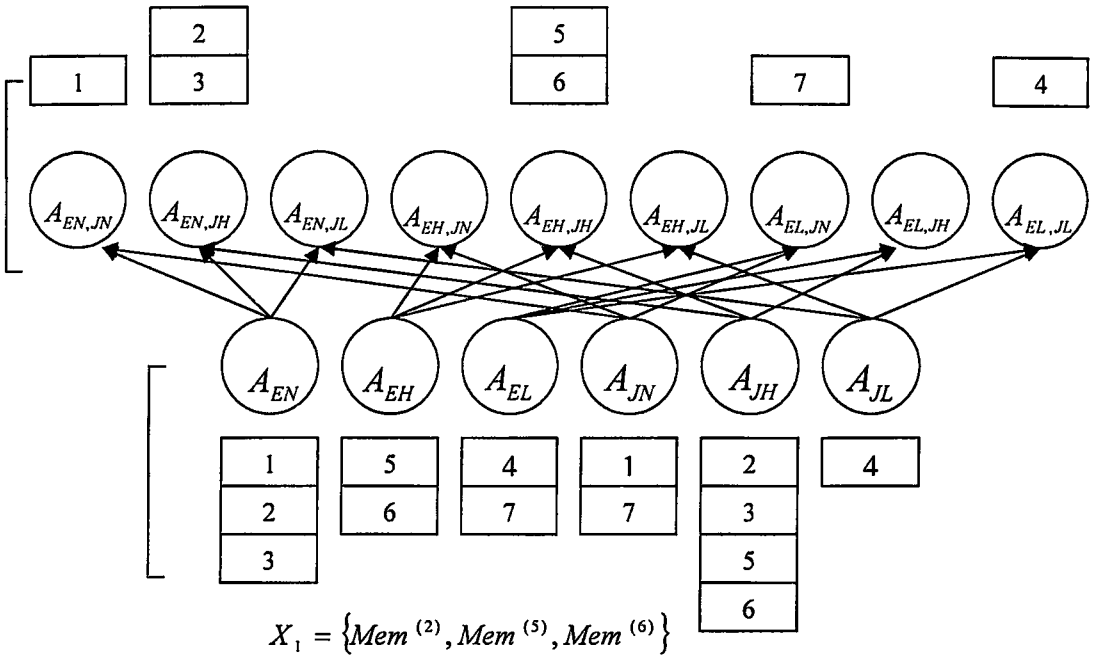
Yeni bir birleşim kuralı yaratılmış oldu. Şekil 4.4 iki sınıfın birleştirilmesini göstermektedir.

Eğer Öğrenim=N ve İş=H ise GID=A CF=(0.25)

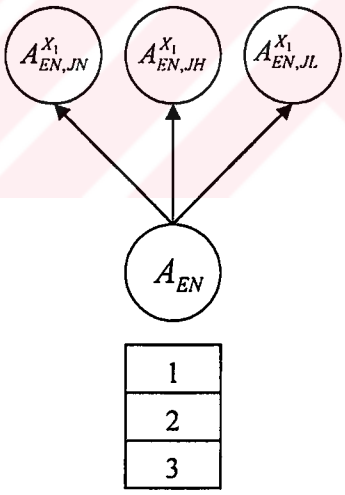
$A_{EN,JH}^{X_1}$'nin kesinlik değeri $A_{EN}^{X_1}$ 'den daha küçüktür.

6. Her bir kümenin karakteristiğini açıklamaya çalışalım. Aynı sonuç sınıfının Lower yaklaşım ve Upper yaklaşım kuralları ve birleştirilmiş kuralları kendi sınıfının karakteristiğini anlatır. Örneğin Kural1 ve Kural3 A sınıfını karakterize etsin. A sınıfının üyeleri aşağıdaki karakterdedir. Sınıf A'da 100% öğrenimi yüksek olanlar. A sınıfına ¼ üyelik değerleri vardır. Öğrenim seviyeleri normal ve işleri iyi olanların ise A sınıfına ¾ üyelik değerleri vardır.

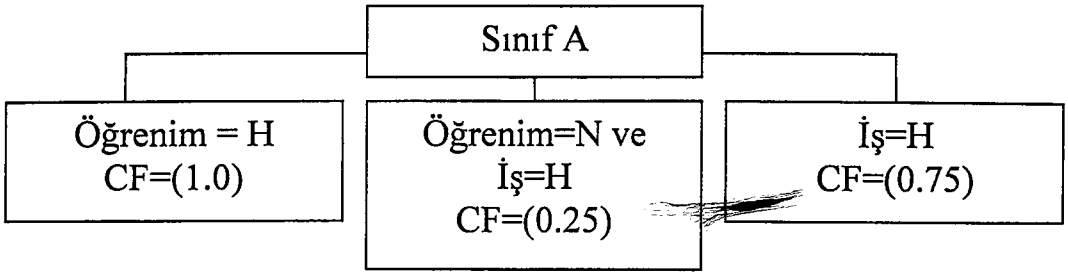
7. 3. adımdan itibaren aynı işlemler X_2 için tekrarlanarak yeni kurallar bulunur. Ve diğer kümeler için kurallar oluşturulmuş olur



Şekil – 4.4 İki sınıfın birleştirilmesi.



Şekil – 4.5 X_1 için EN sınıfında kural bulunması



Şekil – 4.6 Sınıf A için kural çıkarımı

4.1.5 Adım – 5: Farklı Sınıfların Birleştirilmesi

Rough küme teorisi ile her sınıfın karakteristikleri incelenebildiği gibi iki farklı sınıf arasındaki ilişkilerde incelenebilir. İki farklı sınıf arasındaki ilişkileri inceleyen prosedür 2. adımda tanımlandığı gibi yapılır.

5. GELİŞTİRİLEN UYGULAMA

Uygulama 2 ana modülden oluşmaktadır.

1. Veri Madenciliği Modülü
2. Bulanık Modül

5.1 Veri Madenciliği Modülü

Bu modül yedi bölümden oluşmaktadır. Bunlar, clusterData; Sınıflama ve ilk değerlerin tanımlanması için kullanılır. Vektor; java.util.Vector sınıfından türetilmiştir Uzaklık bulma fonksiyonu bu sınıftadır. Kafes; Düğümlerden oluşur ve bir matris gibi düşünülebilir. Rasgele olarak üretilir. Dugum; Her bir düğümün nesne olarak tanımlanması için kullanılır. Egitmen; Kafese giriş vektörlerini tanımlanmış bir işlem adedi kadar göndererek kafesin öğrenmesini sağlar. KuralCikar; Kural çıkarım işleminin yapıldığı sınıftır. VTimer; Zamanlayıcı gereken yerlerde kullanılır. KafesSunucu; Kafesin bir Jframe olarak ekranda gösterilmesi için kullanılır.

5.1.1 ClusterData Sınıfı

```
Class clusterData java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Frame
                          |
                          +--javax.swing.JFrame
                                |
                                +--sommining.clusterData
```

Bu sınıf programın başlatıldığı sınıftır. Aşağıda bu sınıf tarafından yapılan önemli işlemler maddelenmiştir.

- Veri madenciliği ön tanımlar yapılır,
- kafesWidth ve kafesHeight bu sınıf içerisinde tanımlanır.

  private int kafesWidth =20;

  private int kafesHeight = 20;

  Bu çıkış düğümlerinin sayısını belirler.
- Bulanık Uzman Sistem ile ilgili ön tanımlar yapılır,
- Giriş vektörleri veritabanından alınır.

- Jframe ile ilgili tanımlar yapılır. Form nesnesi olarak JPanel, Jtext, Jlabel ve Jbutton kullanılmıştır.
- Basla buttonu ile işlem başlatılır.
- Durdur butonu ile işlem durdurulur.
- Kural Cikar butonu ile output ile üretilen vektör ağırlıkları kurallara dönüştürülür.
- Kullanılan iki text box'a gerçekleşen bilgiler girilir ve Kural Bul butonuna basıldığında ilgili kural, kural tabanından bulunmuş olur.

Programın main fonksiyonu aşağıdaki gibidir. Uygulama basla komutuyla çalıştırılmış olur.

```
public static void main(String args[]) {
      clusterData clusterUygulamasi = new clusterData();
      clusterUygulamasi.show();
      clusterUygulamasi.basla();
}
```

ClusterData sınıfında veritabanı bağlantısı ConnectDB fonksiyonu ile yapılır. Ve veritabanında bulunan ilgili sütunların öncelikle maximum ve minimum değerleri saptanır. Bu değerler normalleştirme işlemi sırasında kullanılacaktır.

```
public void connectDB () {
    String driverPrefixURL = "jdbc:odbc:";
    String dataSource = null;
    Vektor tempVek;

    timer.reset();
    try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      System.out.println("Loaded: " + timer.elapsed());
    } catch (Exception e) {
      System.out.println("JDBC/ODBC çağrılamadı.");
      return;
    }
    dataSource = "vt";
    try {
      timer.reset();
      Connection con =
DriverManager.getConnection(driverPrefixURL+dataSource);
      System.out.println("Connected.: " + timer.elapsed());
        ...
       {
```

```
        String mySQL = "SELECT Max("+d1Adi+"), Min("+d1Adi+"),
        Max("+d2Adi+"), Min("+d2Adi+"),  Max("+d3Adi+"),
        Min("+d3Adi+") FROM "+ tabloAdi +";";
          stnt = con.prepareStatement(mySQL);
          rs = stnt.executeQuery();
          if(rs.next())
          {
            d1Max= rs.getDouble(1);
            d1Min= rs.getDouble(2);
        //Burada sütunların maximum ve minimum değerleri
alınıyor.
             ...
          }
          stnt = null;
          rs = null;

        stnt = con.prepareStatement("SELECT "+d1Adi+",
        "+d2Adi+", "+d3Adi+" from " + tabloAdi);
          rs = stnt.executeQuery();
          girisVektorleri = new Vector();
          int i=0;
          timer.reset();
          while (rs.next())
            {
              tempVek = new Vektor();
        tempVek.addElement(new
        Double(normalize(rs.getDouble(1),d1Max,d1Min)));
        tempVek.addElement(new
        Double(normalize(rs.getDouble(2),d2Max,d2Min)));
        tempVek.addElement(new
        Double(normalize(rs.getDouble(3),d3Max,d3Min)));
        //Sütunların maximum ve minimum değerleri kullanılarak
        alanlardaki değerler
        //normalleştiriliyor
              girisVektorleri.addElement(tempVek);
              i=i+1;
            }
            System.out.println(timer.elapsed());
          }
    }
}
```

Veri tabanından çekilen değerler, girisVektorleri adlı vektor sınıfına yerleştirilir. Bu vektor daha sonra kafesin eğitilmesi sırasında ve CF'ler hesaplanırken kullanılacaktır. Tablo – 5.1 ClusterData metod özetini göstermektedir.

Normalleştirme işlemi aşağıdaki fonksiyon ile yapılır.

```
public static double normalize
      (double deger, double maxdeger, double mindeger)
        {
          return (maxdeger-deger)/(maxdeger-mindeger);
        }
```

Tablo – 5.1 ClusterData metod özeti

clusterData() metod özeti	
void	**basla**()
void	**connectDB**()
private void	**dilselDegEkle**(DilDegiskeni dd)
private void	**exitForm**(java.awt.event.WindowEvent evt)
java.lang.String	**getResult**()
private void	**initComponents**()
private void	**initFuzzySystem**()
static void	**main**(java.lang.String[] args)
static double	**normalize**(double deger, double maxdeger, double mindeger)
private void	**setValues**(double var1, double var2)

ClusterData algoritması:

Program Şekil – 5.1 teki adımlar ile açılır.

Bulanık Uzman Sistem ön değerlerinin oluşturulması (initFuzzySystem)

Bu fonksiyon yardımı ile bulanık uzman sistem içerisinde kullanılacak olan dil değişkenleri tanımlanır. Örneğin promosyon dil değişkeni için;

```
private DilDegiskeni promosyon;

promosyon  = new DilDegiskeni("Prom");
```

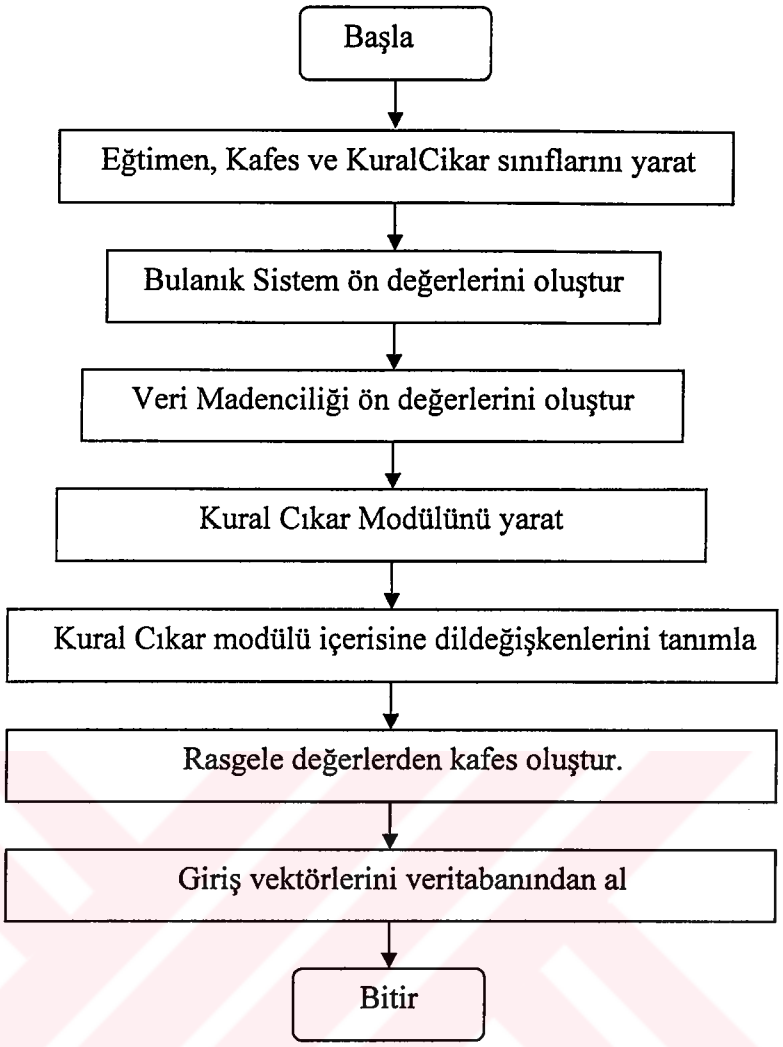
şeklinde tanımlama yapılır. Kullanılacak olan her dil değişkeni ayrı ayrı yukarıda olduğu gibi tanımlanmalıdır. Dil Değişkenleri veritabanındaki alanlar olarak düşünülmelidir. Veri tabanında bulunan değerlerin dildeki karşılıkları bu dil değişkenleri sayesinde programa öğretilir. Bir dil değişkeni yaratıldıktan sonra o dil değişkeni ile ilgili çeşitli aralıklarda dilsel değerler eklememiz gerekir. Burada kullanılacak üyelik fonksiyon yamuk(trapezoidal) tip üyelik fonksiyonu olarak belirlenmiştir.

```
promosyon.ekle("CokZayif", 0, 0, 15, 25);

promosyon.ekle("Zayif", 15, 25, 40, 50);

promosyon.ekle("Orta", 40, 50, 60, 70);

promosyon.ekle("Iyi", 60, 70, 80, 90);

promosyon.ekle("CokIyi", 80, 90, 100, 100);
```

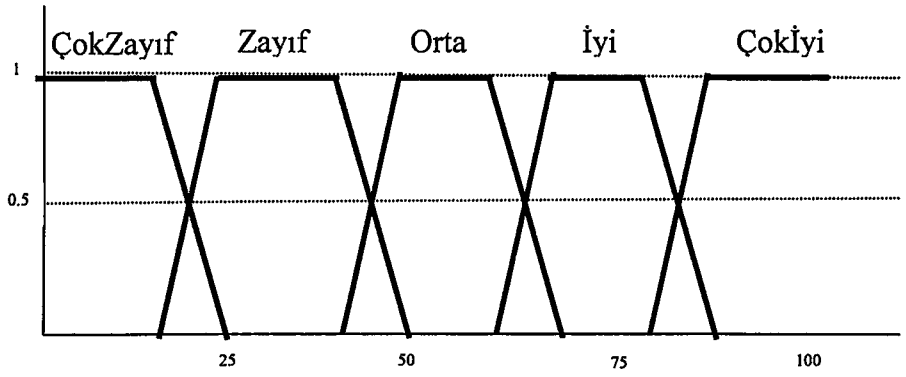
Burada ekle fonksiyonu aşağıdaki gibi tanımlanmıştır.

```
Dildegiskenadi.ekle(DilselDegerAdi, sol alt köşe değeri, sol üst köşe değeri,
sağ üst köşe değeri, sağ alt köşe değeri)
```

Yukarıdaki promosyon ile ilgili dil değişkeninin, dilsel değerlerine karşılık gelen grafik Şekil 5.2'te gösterilmiştir.



Şekil – 5.1 clusterData algoritması



Şekil -5.2 Promosyon Dil Değişkeni

tanımlanmış olan dil değişkeni bulanık uzman sistemde oluşturulmalıdır. Bu tanımla fonksiyonu yardımı ile yapılır.

fuzzyEngine.tanimla(promosyon);

Programda dilsel değişken olarak kullanılacak her alan bu aşamada tanımlanmalıdır.

Veri Madenciliği ön değerlerinin oluşturulması: (initComponents)

Bu aşamada Jframe yardımıyla ekranda görüntülenecek form üzerinde bulunan textbox, label, panel ve button gibi nesnelerin tanımlaması yapılır. Ayrıca kullanılacak olan nesnelerin boyutları yerleri fontları yine bu fonksiyon içerisinde tanımlanır. Ayrıca panel üzerine çizilecek olan kafes tanımları burada yapılır.

Öğrenme işleminin başlaması ve kuralların çıkarılması ileride anlatılacak. Önce Kafes, Düğüm ve Vektör sınıflarını inceleyelim.

5.1.2 Vektör Sınıfı

```
java.lang.Object
  |
  +--java.util.AbstractCollection
        |
        +--java.util.AbstractList
              |
              +--java.util.Vector
                    |
                    +--sommining.Vektor
```

Vektör sınıfı java.util.Vector sınıfından türetilmiştir. Buna ek olarak kafes sınıfında getBMU fonksiyonunda kullanılan uzaklık bulma fonksiyonu eklenmiştir.

Vektör sınıfında vektörler arasındaki en yakın uzaklığı bulmak için euclidean distance kullanılmıştır. Euclidean distance iki boyutta aşağıdaki gibi tanımlanır.

$A(x_1,y_1),B(x_2,y_2)$ iki nokta olsunlar. Bu iki nokta arasındaki uzaklık denklem 5.1 de verilmiştir.

$$dist((x_1,y_1),(x_2,y_2))=((x_2-x_1)^2+(y_2-y_1)^2)^{1/2} \quad (5.1)$$

Tablo – 5.2 Vektör metod özetini göstermektedir.

Tablo 5.2 Vektor metod özeti

Vektor() metod özeti	
double	**euclideanDist**(Vektor v2)

5.1.3 Kafes Sınıfı:

```
java.lang.Object
  |
  +--sommining.Kafes
```

Kafes her biri düğümlerden oluşan bir matris şeklinde düşünülebilir. Kafes içerisinde giriş vektörlerinin en yakın olduğu vektörü bulabilmek için getBMU adlı bir fonksiyon tanımlanmıştır. Burada euclidean distance olarak adlandırılan yöntem kullanılmıştır. Bu distance hesaplaması vektor sınıfı içerisinde bulunan bir fonksiyon yardımı ile yapılır. Fonksiyonun açıklaması vektör sınıfı içerisinde yapılacaktır. Kafes kullanılarak bir sınıf yaratıldığında kafes genişliğini ve yüksekliğini belirlemek gerekir. clusterData sınıfında kafesWidth ve kafesHeight olmak üzere iki değişken bu değerleri belirler.

Kafes kafes = new Kafes(kafesWidth, kafesHeight);

İfadesi kullanılarak kafes yaratılmış olur.

Kafeste bulunan en iyi vektörün(best matching unit) bulunması aşağıdaki fonksiyon ile yapılır. Tablo – 5.3 Kafes metod özetini göstermektedir.

```
public Dugum getBMU(Vektor girisVektor) {
      Dugum bmu = matrix[0][0];
      double bestDist = girisVektor.euclideanDist(bmu.getVektor());
      double curDist;
      for (int x=0; x<width; x++) {
            for (int y=0; y<height; y++) {
            curDist =
      girisVektor.euclideanDist(matrix[x][y].getVektor());
            if (curDist < bestDist) {
                  bmu = matrix[x][y];
                  bestDist = curDist;
                  }
            }
      }return bmu;}
```

Tablo – 5.3 Kafes metod özeti

Kafes() metod özeti	
Dugum	**getBMU**(Vektor girisVektor)
Dugum	**getDugum**(int x, int y)
int	**getHeight**()
double	**getIslemAdedi**()
int	**getWidth**()
void	**setIslemAdedi**(int islemAdet)

5.1.4 Düğüm Sınıfı:

```
java.lang.Object
  |
  +--sommining.Dugum
```

Kafes içerisinde çıkış vektörleri olarak tanımlanmış her bir yapı düğüm olarak tanımlanır. Düğümlerin oluşturduğu sınıfa çıkış vektörü olarak ta adlandırabiliriz. Burada dikkat edilmesi gereken giriş ve çıkış vektörlerinin aynı boyutta olmasının sağlanmasıdır. Biz yazılan uygulamada giriş ve çıkış vektörlerini üç boyutlu olarak tanımladık ve gösterimde bu vektörlerin renk karşılıklarını kullandık. Bu vektör boyutları kullanılacak olan alan sayısına göre değişiklikler gösterebilir. Bu sistem n boyutu destekleyecek şekilde dizayn edilebilir. Kafes ilk olarak oluşturulduğunda her bir düğüm vektörü rasgele olarak belirlenir. Biz bu uygulamada üç boyut kullandığımızdan her üç boyut için rasgele bir değer belirlenir. Bu boyutlar kafesSunucu sınıfı yardımı ile birer renge dönüştürülerek gösterilir. Bu değerler 0 ve 1 arasındadır. Vektör boyutları r,g,b değerlerine karşılık getirilir. Kafes oluşturulduğunda her bir düğüm rasgele olarak seçildiğinden birbirinden farklı renklerde düğümlerden oluşan bir kafes görüntüsü oluşacaktır.

Düğüm içerisinde gönderilen giriş vektörüne karşılık gelen en iyi düğüm bulunduğunda çevresindeki öğrenme oranı ve komşuluk yarıçapı değerlerine göre

diğer düğümleri değiştirecek bir fonksiyon tanımlanmıştır. Bu dugumleriDegistir fonksiyonudur. Tablo – 5.4 Dugum metod özetini göstermektedir.

Yeni değer aşağıdaki gibi belirlenir.

t+1 anındaki ağırlık vektörü;

ağrilik(t+1)= ağrilik(t)+öğrenmeOrani(t)*(eskiVektor-girisVektoru) (5.2)

Tablo – 5.4 Dugum metod özeti

Dugum() metod özeti	
double	**distanceTo**(Dugum n2)
void	**dugumleriDegistir**(Vektor girisVektoru, double ogrenmeOrani, double aradakiUzaklik)
Vektor	**getVektor**()
double	**getWeight**(int w)
int	**getX**()
int	**getY**()
void	**setWeight**(int w, double value)
void	**setX**(int xpos)
void	**setY**(int ypos)

Düğümlerin değiştirilmesi aşağıdaki fonksiyonla yapılır.

```
public void dugumleriDegistir(Vektor girisVektoru, double
ogrenmeOrani, double aradakiUzaklik)
      {
            double agirlikt, eskivektor;
            for (int w=0; w<agirlik.size(); w++) {
                  agirlikt =
                  ((Double)agirlik.elementAt(w)).doubleValue();
```

```
                    eskivektor =
                    ((Double)girisVektoru.elementAt(w)).doubleValue();
                    agirlikt += aradakiUzaklik * ogrenmeOrani *
                    (eskivektor - agirlikt);
                     if (agirlikt<0){return;}
                    agirlik.setElementAt(new Double(agirlikt), w);
                }
        }
```

5.1.5 Eğitmen Sınıfı:

```
java.lang.Object
  |
  +--sommining.Egitmen
```

Rasgele oluşturulan kafes sınıfına rasgele gönderilen vektörlerin en yakın uzaklıkta olanı bulan ve bu vektörün çevresindekileri de denklem (5.2) ye göre değiştiren bir sınıftır. Eğitmen sınıfında ISLEM_ADEDI adlı değişken öğrenme süreci için ayrılan maximum işlem adedini tutar. Burada tüm vektörler kafese gönderildikten sonra bir işlem tamamlanmış olur. Biz uygulamamızda bu değeri 100 olarak belirledik. Bu t için artım aralığının 1/100 olacağı anlamını taşır.

OGRENME_ORANI adlı değişken komşu ağırlıkları değiştirirken kullanılıyor (denklem (5.2)).

Uygulama threadli bir yapı üzerine oturtuldu ve uygulama başlatıldığında run fonksiyonu içerisindeki adımlar çalıştırılır. Burada kafes içerisinde, rasgele olarak seçilmiş giriş vektörüne en yakın uzaklıkta bulunan vektörün x ve y değerleri belirlenir. Daha sonra çevresindeki düğümler değiştirilerek öğrenme sağlanır. Tablo 5.5 Egitmen sınıfının metod özetini göstermektedir.

Thread içerisinde çalışan, öğrenme fonksiyonu aşağıdaki gibidir.

```
public void run() {
            int lw = kafes.getWidth();
            int lh = kafes.getHeight();
            int xbas, ybas, xend, yend;
            double uzaklik, aradakiUzaklik;
            KAFES_YARICAPI = Math.max(lw, lh)/2;
            ZAMAN_SABITI = ISLEM_ADEDI / Math.log(KAFES_YARICAPI);
            int islem = 0;
            double nbhYaricapi;
            Dugum bmu = null, temp = null;
            Vektor curInput = null;
```

```
double ogrenmeOrani = OGRENME_ORANI;
while (islem < ISLEM_ADEDI && calisiyor) {
      nbhYaricapi = getKomsuYaricapi(islem);
      for (int i=0; i<girisler.size(); i++) {
            curInput = (Vektor)girisler.elementAt(i);
            bmu = kafes.getBMU(curInput);
            xbas = (int)(bmu.getX() - nbhYaricapi - 1);
            ybas = (int)(bmu.getY() - nbhYaricapi - 1);
            xend = (int)(xbas + (nbhYaricapi * 2) + 1);
            yend = (int)(ybas + (nbhYaricapi * 2) + 1);
            if (xend > lw) xend = lw;
            if (xbas < 0) xbas = 0;
            if (yend > lh) yend = lh;
            if (ybas < 0) ybas = 0;
            for (int x=xbas; x<xend; x++) {
               for (int y=ybas; y<yend; y++) {
                 temp = kafes.getDugum(x,y);
                 uzaklik = bmu.distanceTo(temp);
                  if (uzaklik <= (nbhYaricapi *
                  nbhYaricapi)) {
                  aradakiUzaklik = getUzaklik(uzaklik,
                  nbhYaricapi);
                  temp.dugumleriDegistir(curInp,
                  ogrOrani, araUzaklik);
                  }
               }
            }
      }
      islem++;
      ogrenmeOrani = OGRENME_ORANI *
                  Math.exp(-(double)islem/ISLEM_ADEDI);
      sunucu.render(kafes, islem);
}
calisiyor = false;}
```

Tablo – 5.5 Eğitmen metod özeti

Egitmen() metod özeti	
void	**dur**()
private double	**getKomsuYaricapi**(double islem)
private double	**getUzaklik**(double uzaklikSq, double yaricap)
void	**run**()
void	**setTraining**(Kafes ogrenKafes, java.util.Vector girisDegerleri, KafesSunucu kafesSunucu)
void	**start**()

5.1.6 KuralCikar Sınıfı:

```
java.lang.Object
  |
  +--sommining.KuralCikar
```

Kural çıkarımının yapıldığı sınıftır. Kafes üzerinde öğrenme bittikten sonra sınıflandırma işlemi genel hatları ile bitmiş olur. Burada çıkan değerlerin bulanık istemdeki dil değerleri bulunarak sınıflara ayrılır. Burada kullanılan metod Rough küme teorisidir. Şekil 5.3 Veritabanından okunan her bir değere karşılık üretilmiş olan kuralların ağırlıkları hesaplanır. Bu ağırlıklar kural olarak öne sürülen ifadenin veritabanında ne kadar sıklıkla bulunduğu, veritabanının bu kuralı ne kadar desteklediği ile doğru orantılı olarak bulunur. Bu ağırlıklar denklem (4.3) kullanılarak bulunur.

Kural çıkarımında kullanılan CF'lerin hesabı aşağıdaki fonksiyon yardımı ile yapılır. Burada sınıflandırma işlemi bittikten sonra muhtemel sınıflar sinifVektorleri vektörünün içerisine atılır ve bu vektörlerin veritabanını ne kadar desteklediğinin saptanması için giriş vektörlerinin tutulduğu girisVektorlerine dönülür. Tablo – 5.6 KuralCikar sınıfının metod özetini göstermektedir.

```
private void CFHesapla()
  {
    Vector curSinif=null;
    Vector curGiris=null;
```

```
Vector tempVek=null;
if (sinifVektorleri != null && girisVektorleri != null)
{
  for (int i=0;i<sinifVektorleri.size();i++)
  {
    double sinifVektorSayisi=0;
    double d1DegSayisi=0;
    double d2DegSayisi=0;
    double d3DegSayisi=0;

    for (int j=0;j<girisVektorleri.size();j++)
    {
        curSinif = (Vector)sinifVektorleri.elementAt(i);
        curGiris = (Vector)girisVektorleri.elementAt(j);
        //Bir sınıf vektöründen giriş vektörlerinin içerisinde
        kaçtane olduğu bulunur.

        rDD.setGirisDegeri(Double.parseDouble(curGiris.elementAt
        (0).toString()));
        String sonucR=getBestDilDeg(rDD);

        gDD.setGirisDegeri(Double.parseDouble(curGiris.elementAt
        (1).toString()));
        String sonucG=getBestDilDeg(gDD);

        bDD.setGirisDegeri(Double.parseDouble(curGiris.elementAt
        (2).toString()));
        String sonucB=getBestDilDeg(bDD);

        if (curSinif.elementAt(0).equals(sonucR) &&
        curSinif.elementAt(1).equals(sonucG) &&
        curSinif.elementAt(2).equals(sonucB))
        {
              sinifVektorSayisi++;
        }
        if (curSinif.elementAt(0).equals(sonucR))
        {
              d1DegSayisi++;
        }
        if (curSinif.elementAt(1).equals(sonucG))
        {
```

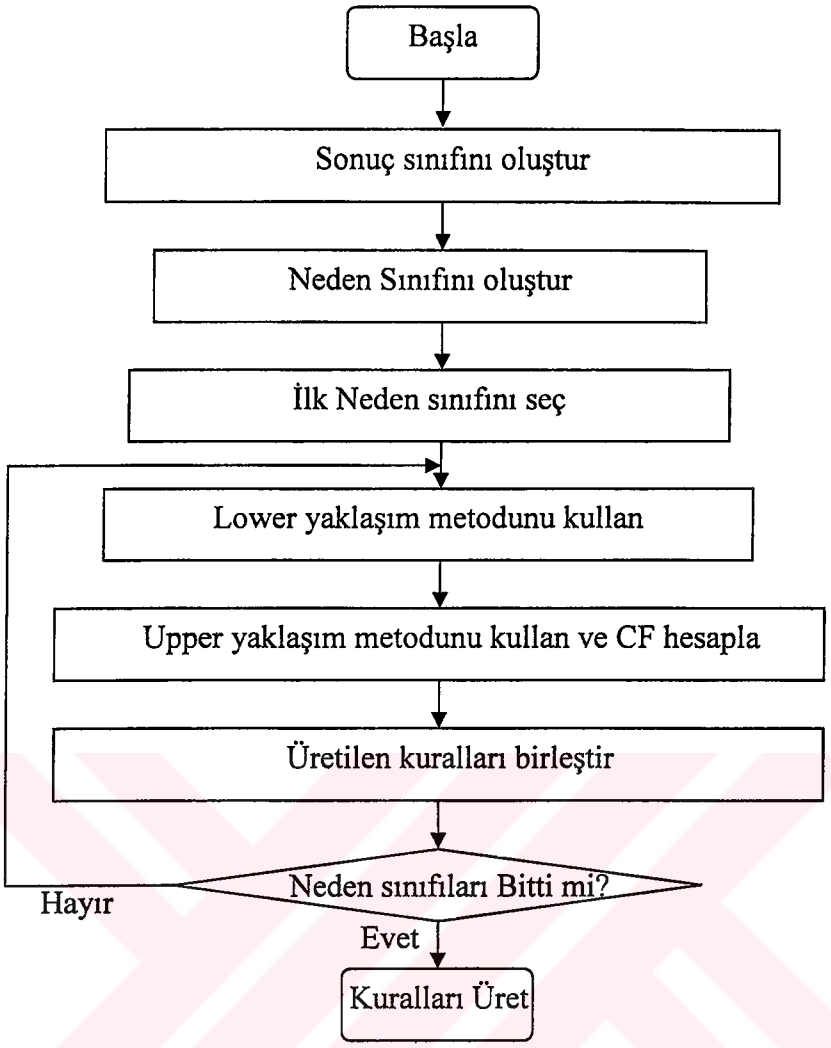
```
                d2DegSayisi++;
            }
            if (curSinif.elementAt(2).equals(sonucB))
            {
                d3DegSayisi++;
            }
        }
        tempVek = new Vector();
        tempVek.addElement(new Double(sinifVektorSayisi/d1DegSayisi));
        tempVek.addElement(new Double(sinifVektorSayisi/d2DegSayisi));
        tempVek.addElement(new Double(sinifVektorSayisi/d3DegSayisi));
        CFVektorleri.addElement(tempVek);
        }
    }
}
```

Tablo – 5.6 KuralCikar metod özeti

KuralCikar() metod özeti	
private void	**CFHesapla**()
java.lang.String	**getBestDilDeg**(DilDegiskeni DD)
void	**getKural**()
java.lang.String	**getKuralStr**()
java.lang.String	**getThreshold**()
void	**girisVekDilDeg**()
void	**kurallariOlustur**()
double	**rollDouble**(double number)
void	**setDilDegiskeniB**(DilDegiskeni bDDi)
void	**setDilDegiskeniG**(DilDegiskeni gDDi)

Tablo – 5.6 KuralCikar metod özeti (devamı)

void	**setDilDegiskeniR**(DilDegiskeni rDDi)
void	**setGirisVektor**(java.util.Vector girisVektor)
void	**setIsim**(java.lang.String rA, java.lang.String gA, java.lang.String bA)
void	**setKafes**(Kafes kaf)
void	**setThreshold**(double t)
void	**setValues**(double r, double g, double b)
private boolean	**sinifBul**(java.lang.String sonucR, java.lang.String sonucG, java.lang.String sonucB)
private void	**sinifEkle**(java.lang.String sonucR, java.lang.String sonucG, java.lang.String sonucB)
private void	**siniflaraEkle**(double r, double g, double b)



Şekil – 5.3 Rough küme teorisi uygulaması

5.1.7 VTimer Sınıfı;

```
java.lang.Object
  |
  +--sommining.VTimer
```

Zamanlayıcı kullanılması gereken yerlerde VTimer sınıfı kullanılır. Özellikle veritabanından connection alırken ve bir sorgulama yaparken geçen sürelerin saptanmasında kullanıldı. Tablo – 5.1 Vtimer sınıfının metod özetini göstermektedir.

Tablo – 5.7 VTimer metod özeti

VTimer() metod özeti	
`long`	**elapsed**()
`void`	**print**(java.lang.String s)
`void`	**reset**()

Tablo – 5.8 KafesSunucu metod özeti

KafesSunucu() metod özeti	
java.awt.image.BufferedImage	**getImage**()
boolean	**getIslemBitti**()
void	**kafesKaydet**(Kafes kaf)
void	**paint**(java.awt.Graphics g)
private void	**panelMouseMoved**(java.awt.event.MouseEvent evt)
void	**render**(Kafes kafes, int islem)
void	**setImage**(java.awt.image.BufferedImage bimg)
void	**setIslemBitti**(boolean islemDur)
void	**setKuralCikar**(KuralCikar kci)
void	**setKuralGoster**(boolean KuralGoster

5.1.8 KafesSunucu Sınıfı:

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--sommining.KafesSunucu
```

Kafesin ekranda gösterilmesini sağlayan sınıftır. Öğrenme esnasında değişen komşulukları yine bu sınıf ekranda günceller. Tablo – 5.8 KafesSunucu sınıfının metod özetini göstermektedir.

5.2 Bulanık Uzman Sistem:

Bu modül 10 sınıftan oluşur. BulanıkUzman; bulanık uzman sistemin çatısını oluşturur. kullanılacak olan kurallar dil değişkenleri bu sınıf üzerinden tanımlanır. BulanıkKural; bulanık kuralların çalıştırıldığı ve işlendiği sınıftır. BulanıkIfade; bulanık bir ifade bu sınıf içerisinde tanımlanır ve çalıştırılır. UyelikFonksiyonu; bu projede yamuk tip üyelik fonksiyonu kullanılmıştır. İstenilen üyelik fonksiyonu bu sınıf değiştirilerek kullanılabilir. DilDegiskeni; kullanılacak olan dil değişkenleri bu sınıf yardımı ile tanımlanır. BlokKurallar; kurallar tek tek değil blok halinde programa bu sınıf yardımı ile tanımlanır. Veri madenciliği programından üretilen kurallar bu sınıf ile uzman sisteme aktarılır. Op, OpBiraz, OpCok, OpDegil; çok, biraz, değil gibi dilsel pekiştiriciler bu sınıflar ile tanımlanır.

5.2.1 BulanıkUzman Sınıfı:

```
java.lang.Object
  |
  +--bulanik.BulanikUzman
```

veri madenciliği aracı içerisinde bulunan clusterData sınıfı bu sınıf yardımı ile uzman sistem ile haberleşir ve tanımlamalarını yapar. clusterData sınıfının bulanıkUzman sistemi kullanabilmesi için sırası ile Şekil 5.4'teki işlemler yapılır.

Şekil – 5.4 Bulanık uzman sistem algoritması

BulanıkUzman Sınıfında kural tanımlaması, blok halinde kural tanımlaması, dil değişkeni tanımlaması, bir bulanık ifadenin parse işlemi, giriş değerlerinin uzman sisteme aktarılması ve eğer bu değerler herhangi bir veya birden fazla kuralı gerçekliyorsa bu kuralların çalıştırılması.

Bir kuralın parse edilmesi:

Bir kural parse edilirken 2002 yılında Sazanov'un önerdiği sistem kullanıldı. Burada bir bulanık kuralda bulunabilecek her öğenin bir sayı karşılığı atandı. Aşağıda kod içerisinde bulunan ve parsing aşamasında kullanılacak olan tanımlamalar görülmektedir.

```
private final int EGER = 1;
private final int ISE = 2;
private final int ESITTIR = 3;
private final int VE = 4;
```

```
private final int VEYA = 5;
private final int SOLPARANTEZ = 6;
private final int SAGPARANTEZ = 7;
private final int ISLEMYOK = 8;
private final int CALISTIR = 9;
private final int OP = 10;
private final int TANIMSIZ = 11;
private final int DILDEGISKENI = 12;

private final int HAZIR = 50;
private final int DILDEGISKENI_OKUNDU = 51;
private final int ESITTIR_OKUNDU = 52;
private final int DEGIL_OKUNDU = 53;
private final int OP_OKUNDU = 54;
private final int EXCEPTION = 55;

private final int ISLEM_TAMAMLANDI = 100;

private final int CALISMA_TAMAMLANDI = 150;

private final int BASLA = 200;
private final int ISLEM = 201;
private final int CALISMA = 202;
```

Burada bir bulanık kural parse etme fonksiyonuna verildiğinde, bu bulanık ifade ilk önce boşluklarından ayrılarak bir vektöre yerleştiriliyor. Daha sonra vektöre bulunan her bir öğe incelenmeye başlanıyor. Sistemde bir kuralın hangi öğelerden oluştuğu biliniyor. Bizim kurduğumuz sistemde kurallar aşağıdaki gibi tanımlanmıştır.

Eger (Degisken1 = Op1 Deger1) ve (Degisken2 = Op2 Deger2) ise (Degisken3 = Op3 Deger3)

Burada her bir boşluğun önemi vardır. Öreğin kullanılan '=' işaretlerinden önce ve sonra birer boşluk olmalıdır.

Burada kullanılan ifadeler;

Degisken; Dilsel Değişkeni,

Deger; Degisken için tanımlanmış dilsel değerleri,

Op; sistemde tanımlı olan dilsel pekiştiricileri anlatmaktadır.

Parse işlemi başladığında ilk önce kuralın Eger ile başlayıp başlamadığı kontrol edilir. Eğer ise durum değişkeni ISLEM'e yani 201'e çekilir. Bu ISLEM'in başlatılacağı anlamını taşır. İşlem başlatıldıktan sonra DilDeğişkeni, parantezler, ve,veya, ise gibi değerler okunur. Bu değerler Alt durumları oluşturur. Eğer Alt

durum Dil Değişkeni ise bu aşamadan sonra neler gelebileceği bellidir. Bir dil değişkeninden sonra bir boşluk ve '=' olması gerekir. Bu detaylar kontrol edilir. Daha sonra bir operatör olabilir veya dil değeri gelebilir. İlk önce operatör olup olmadığı kontrol edilir. Eğer bir operatör yok ise dil değeri alınır. Örneğin; borsa = düşük gibi. Bu şekilde devam edilerek parse işlemi bitirilir. Parse işlemi bittiğinde kural, bulanık ifadelere ayrılmış ve değerleri hesaplanmış olur. Tablo – 5.9 BulanıkUzman metod özetini göstermektedir.

Tablo – 5.9 BulanıkUzman metod özeti

BulanikUzman() metod özeti	
void	**blokIsle**(BlokKurallar blok)
java.lang.String	**kuralCalistir**(java.lang.String kural)
boolean	**kuralGerceklendimi**()
void	**opEkle**(Op op)
private java.lang.String	**parseIfade**(java.util.StringTokenizer tokens, BulanikKural kural)
BulanikKural	**parseKural**(java.lang.String kural)
void	**reset**()
void	**tanimla**(BlokKurallar blok)
void	**tanimla**(DilDegiskeni fonksiyon)

5.2.2 BulanikKural Sınıfı

```
java.lang.Object
  |
  +--bulanik.BulanikKural
```

Girilen gerçek değerler BulanıkUzman ile BulanikKural sınıfına iletilir. Bu sınıf ile girilen gerçek değerlerine karşılık gelen kural bulunur. İlk önce kural dizisinden birinci kural alınır. Girilen değerin bulanık ifadeleri bulunur ve eğer bir numaralı

kural bu ifadeyi desteklemiyor ise diğer kurallara geçilir. Eğer hiçbir kural desteklemiyorsa herhangi bir kural çalıştırılmaz.

BulanıkUzman parse işlemi sırasında bir kuralların ise'den önceki kısımında bulunan ifadeleri solTaraf isimli; iseden sonraki ifadeleri sagTaraf isimli vektörlere atar. Gerçek(Fact) Sol tarafta bulunan bulanık ifadeler içerisinde aranır ve bu değerler dilsel değerlere dönüştürülerek ilgili ifadeye üyelik değerleri hesaplanır. Eğer girilen değerler herhangi bir kuralda sağlanıyorsa bu kural çalıştırılır ve sonuç değeri üretilir. Sonuç değeri sol taraftaki bulanık ifadelerin ve/veya ile birbirlerine bağlanmış olduğu durumlara göre; sol tarafta hesaplanmış değerlerin minimumu yada maximumu olacak şekilde üretilir. Tablo – 5.10 BulanikKural metod özetini göstermektedir.

Tablo – 5.10 BulanikKural metod özeti

BulanikKural() metod özeti	
java.lang.String	**getEtiket**()
double	**getIslemSonuc**()
java.util.Vector	**getSagTaraf**()
java.util.Vector	**getSolTaraf**()
boolean	**kuralGerceklendimi**()
void	**kuralIsle**()
java.lang.String	**kuralTextIsle**()
void	**setEtiket**(java.lang.String yEtiket)

5.2.3 BulanikIfade Sınıfı

```
java.lang.Object
  |
  +--bulanik.BulanikIfade
```

BulanıkKural sınıfı ile işletilen kuralların sol ve sağ tarafları bulanık ifadelerden oluşur. Her biri, birbirine ve/veya'lar ile bağlanabilen bu ifadeler program içerisinde BulanıkIfade sınıfı tarafından temsil edilir. Kuralların bulanık değerleri hesaplanırken, içerisinde bulunan bulanık ifadelerin değerleri hesaplanır ve daha sonra bunlar bulanık işlemler yardımı ile birleştirilir.

İfade işlenirken;

```
if(DD!=null && UF!=null)
        sonuc = DD.esittir(UF);
else
        throw new Hata(" - Dil değişkeni veya uyelik
        fonksiyonu tanımlanmamış: ");
...
for (Enumeration en = op.elements() ; en.hasMoreElements() ;)
{
        sonuc = ((Op)en.nextElement()).hesapla(sonuc);
}
```

olacak şekilde bir UyelikFonksiyonuna karşılık bir değer hesaplanır. Burada örneğin giriş DilDeğişkeni faiz olsun ve gerçek(fact) olarak giriş değeri 0.33 olsun bu değere karşılık. Uyelik Fonksiyonu 'dusuk' olarak bulunur ve 0.33 için uyelik derecesi 0.58 olacaktır. Bu değer 0.22 için 1'e daha da yaklaşacaktır.

İfade çalıştırılırken;

```
for (Enumeration en = op.elements() ; en.hasMoreElements() ;)
{
  gDeger = ((Op)en.nextElement()).hesapla(gDeger);
}
if(DD!=null && UF!=null)
{
        if(gDeger>0.0)
        {
        try{
                        DD.set(etiket, UF, gDeger);
        }
        catch(Exception e)
                {throw new Exception(e.getMessage());}
        }
}
```

```
else
{
        throw new Hata(" - Dil değişkeni veya uyelik
        fonksiyonu tanımlanmamış: ");
}
```

yapısı kullanılır. Artık kuralların sağ tarafları kullanılacaktır. Eğer bir kural çalıştırıldığında kural için hesaplanmış değer sıfırdan büyük ise bu kural gerçeklenmiş sayılacaktır ve bu sonuçta gözükecektir. Tablo – 5.11 BulanıkIfade metod özetini göstermektedir.

Tablo – 5.11 BulanıkIfade metod özeti

BulanikIfade() metod özeti	
java.lang.String	**getTextIfade**()
double	**ifadeCalistir**(double deger, java.lang.String etiket)
double	**ifadeIsle**()

5.2.4 UyelikFonksiyonu Sınıfı:

```
java.lang.Object
  |
  +--bulanik.UyelikFonksiyonu
```

Üyelik fonksiyonu olarak yamuk(trapezoidal) tip üyelik fonksiyonlarını seçtik. Burada bir üyelik fonksiyonu tanımlanmak istendiğinde fonksiyonu adı, yamuğun sırasıyla sol alt köşe değeri sol üst köşe değeri, sağ üst köşe değeri ve sağ alt köşe değerinin verilmesi gerekir. Böylece bir dil değişkeni için üyelik fonksiyonu tanımlanmış olur.

Bulanıklaştırma işlemi için aşağıdaki yapı kullanılmıştır.

```
public double  bulaniklastir(double deger)
{
      if(deger<dizi[0] || deger>dizi[3]) return 0;
      if(deger>=dizi[1] && deger<=dizi[2]) return 1;
      if(deger>=dizi[0] && deger<dizi[1])
            return (deger-dizi[0])/(dizi[1]-dizi[0]);
      if(deger>dizi[2] && deger<=dizi[3])
```

```
            return (dizi[3]-deger)/(dizi[3]-dizi[2]);
        return 0;
}
```

Burada eğer değer yamuk içerisinde bir noktada değil ise 0 döndürülür. Bu kuralın bu değer için gerçeklenmeyeceği anlamını taşır. Eğer değer yamuğun üst kenarında bir noktaya karşılık geliyorsa bu giriş değerini tam olarak karşılıyor anlamını taşır ve bu bulanık ifade için hesaplanan değer 1 olarak döndürülür. Eğer yamuğun diğer iki kenarında ise değer hesaplanarak döndürülür. Tablo – 5.12 UyelikFonksiyonu metod özeti gösterilmektedir.

Tablo – 5.12 UyelikFonksiyonu metod özeti

UyelikFonksiyonu() metod özeti	
double	**bulaniklastir**(double deger)
java.lang.String	**getAd**()
double[]	**getDizi**()
double[]	**ciz**(double from, double to, int size)

5.2.5 DilDegiskeni Sınıfı:

```
java.lang.Object
  |
  +--bulanik.DilDegiskeni
```

Dil değişkenleri veritabanında bulunan değerlerin dil karşılıklarını bulmak için kullanılır. Bunlar uzun, kısa, az, çok normal gibi kelimeler olabilir. Bir dildeğişkeni eklemek için aşağıdaki ifade kullanılır.

İlk önce bir dildeğişkeni tanımlanır. Ve eklenmiş olan bu dil değişkenine uyelik fonksiyonu eklenir.

```
        private DilDegiskeni promosyon;

        promosyon = new DilDegiskeni("Prom");
```

```
public void ekle(String ad,double bas, double sol_ust, double
sag_ust, double son)
{
```

```
        double [] dizi = {bas,sol_ust,sag_ust,son};
        UyelikFonksiyonu temp = new UyelikFonksiyonu(ad,dizi);
        tablo.put(ad,temp);
        if(bas<minDeger)  minDeger=bas;
        if(son>maxDeger)  maxDeger=son;
}
```

Yukarıda bulunan fonksiyonun kullanımı aşağıdaki gibi olur.

promosyon.ekle("Zayif", 0, 0, 15, 25);

Burada tanımlanmış bir dil değişkenine Zayif adlı Uyelik fonksiyonunun eklenmesi sağlanmıştır.

Durulaştırma işlemi ise aşağıdaki gibi yapılır.

```
public double durulastir() throws Exception
{
        int sonuc = durulastirma.size();
        ...
        double adim = Math.abs((maxDeger-minDeger)/100);
        ...
        for(int i=0; i<(sonuc-1); i+=3)
        {
                scaled =
                this.getUyelikFonsiyonuAdi((String)durulastirma.elementA
                t(i+1)).ciz(minDeger,maxDeger,100);
                double scale =
                ((Double)durulastirma.elementAt(i+2)).doubleValue();
                for(int j=0; j<100; j++)
                toplam[j]+=scaled[j]*scale;
        }
        ...
        for(int i=0; i<100; i++)
        {
                nominator+=(minDeger+adim*i)*toplam[i];
                denominator+=toplam[i];
        }
        return nominator/denominator;
}
```

Burada bulanık değerlere karşılık sayı değerlerinin getirilmesi sağlanmıştır. Tablo – 5.13 DilDegiskeni metod özeti gösterilmektedir.

Tablo – 5.13 DilDegiskeni metod özeti

DilDegiskeni() metod özeti	
double	**durulastir**()
void	**ekle**(java.lang.String ad, double bas, double sol_ust, double sag_ust, double son)
void	**ekle**(UyelikFonksiyonu uFonk)
double	**esittir**(java.lang.String adi)
java.lang.String	**getDildegiskeniAdi**()
java.util.Hashtable	**getTablo**()
UyelikFonksiyonu	**getUyelikFonsiyonuAdi**(java.lang.String adi)
void	**reset**()
void	**set**(java.lang.String etiket, java.lang.String adi, double deger)
void	**setDilDegiskeniAdi**(java.lang.String adi)
void	**setGirisDegeri**(double deger)

5.2.6 BlokKurallar Sınıfı

```
java.lang.Object
  |
  +--bulanik.BlokKurallar
```

BlokKurallar sınıfı, Veri Madenciliği aracı tarafından blok olarak üretilmiş olan kuralları birbirinden ayırarak vektörlere yerleştirir ve işlenmesini sağlar.

İlk önce blokIsle fonksiyonu çalışır. Herbir kuralı BulanıkKural sınıfı yardımı ile tanımlar ve ayrı ayrı parse edilmesini sağlar ve kuralın girilen değerlerle gerçeklenip gerçeklenmediğini denetler. Eğer herhangi bir kural gerçeklenmedi ise Exception'a düşerek programdan çıkılır. Tablo – 5.14 BlokKurallar metod özeti gösterilmektedir.

```
public void blokIsle() throws Exception
{
      kuralGerceklendi = false;
      for (Enumeration en = tumKurallar.elements() ;
      en.hasMoreElements() ;)
      {
          BulanikKural degKurallar = (BulanikKural)en.nextElement();
                try
                {
                  degKurallar.kuralIsle();
                  if(degKurallar.kuralGerceklendimi())
                        kuralGerceklendi = true;
                  }
                  catch(Exception e)
            {
                throw new Exception(e.getMessage());
            }
      }
}
```

Tablo – 5.14 BlokKurallar metod özeti

BlokKurallar() metod özeti	
void	**blokIsle**()
java.lang.String	**blokTextIsle**()
boolean	**kuralGerceklendimi**()
void	**parseBlok**()
void	**setBulanikUzman**(BulanikUzman myBulanikUzman)

5.2.7 Op Sınıfı

```
java.lang.Object
  |
  +--bulanik.Op
```

Op sınıfı soyut bir sınıf olarak tanımlanır. Cok, Biraz, Degil vb niteleyicileri programa tanıtmak için kullanılan sınıflar Op sınıfından türetilir. Tablo – 5.15 Op metod özeti gösterilmektedir.

Tablo – 5.15 Op metod özeti

Op() metod özeti	
java.lang.String	**getIsim**()
double	**hesapla**(double deger)

5.2.8 OpBiraz Sınıfı

```
java.lang.Object
  |
  +--bulanik.Op
        |
        +--bulanik.OpBiraz
```

OpBiraz sınıfı Op sınıfından türetilmiştir. Tablo – 5.16 OpBiraz metod özeti gösterilmektedir.

(5.3)

işlemi yapılır.

Tablo – 5.16 OpBiraz metod özeti

OpBiraz() metod özeti	
java.lang.String	**getIsim**()
double	**hesapla**(double deger)

5.2.9 OpCok Sınıfı

```
java.lang.Object
  |
  +--bulanik.Op
        |
        +--bulanik.OpCok
```

OpCok sınıfı Op sınıfından türetilmiştir. Tablo – 5.17 OpCok metod özeti gösterilmektedir.

$$\underline{A}^2 = \sum_{i=1}^{n} \mu_A^2(x_i) / x_i \tag{5.4}$$

işlemi yapılır.

Tablo – 5.17 OpCok metod özeti

OpCok() metod özeti	
java.lang.String	**getIsim**()
double	**hesapla**(double deger)

5.2.10 OpDeğil Sınıfı

```
java.lang.Object
  |
  +--bulanik.Op
        |
        +--bulanik.OpDegil
```

OpDegil sınıfı Op sınıfından türetilmiştir. Tablo – 5.18 OpDegil metod özeti gösterilmektedir.

$$\underline{A}^* = \sum_{i=1}^{n} (1 - \mu_A(x_i)) / x_i \tag{5.5}$$

işlemi yapılır.

Tablo – 5.18 OpDegil metod özeti

OpDegil() metod özeti	
java.lang.String	**getIsim**()
double	**hesapla**(double deger)

6. SONUÇLAR VE ÖNERİLER

Bu yüksek lisans tez çalışmasının konusu olan veri madenciliğinin bulanık uzman sistemlerde kullanılması uygulamasının geliştirme sürecinde S. Wesley Changchien, Tzu-Chuen Lu tarafından önerilen veri madencilği algoritmaları ve Edward S. Sazonov, Powsiri Klinkhachorn, Hota V.S. GangaRao, and Udaya B. Halabe tarafından öne sürülmüş olan bulanık uzman sistem algoritmaları temel alınmıştır.

Kullanılan örnek tablolar üzerinde yapılan sorgulamalar sonucunda oluşmuş olan kural tabanının tutarlı olduğu ve expert sistem tarafından tutarlı sonuçlar döndürüldüğü gözlenmiştir.

Program üç boyutu destekler nitelikte tasarlanmıştır ama kullanılan fonksiyonların bir çoğunda diğer boyutları destekleyecek şekilde tasarımlar yapılmıştır. Üç boyuttan fazla alanların incelenmesi gerektiğinde kodda değişikliğe gidilmesi gerekmektedir.

Yapılan uygulama büyük veritabanlarından örnekleme yapan algoritmaları içermez. Bu programın büyük çaptaki veritabanlarını destekleyebilmesi için verileri örnekleyerek azaltacak bir algoritma ile desteklenmesi gerekmektedir. Bu tür bir algoritma desteği olmadan programın kullanımı sonuçların alınmasını geciktirecektir.

Geliştirilen uygulama çok geniş bir alan üzerinde etkin bir şekilde uygulanabilir. Biz ekonomideki üç değişkeni inceledik ama pek çok alanda bu program kullanılarak çıkarımlar yapılabilir. Örnek olarak metoroloji, satış, deprem araştırmaları vb. Alanlar verilebilir.

Veri madenciliği aracının bir çıktısı olarak oluşturulan kuralş tabanı her program çalıştırıldığında yeniden oluşturulmaktadır. Bu kurallar veritabanına yazılarak yeni eklenen verilerin kuralları destekleyip desteklemediğine bakarak yeni kuralların üretilmesi sağlanabilir. Böylece her yeni veri geldiğinde kuraltabanının yeniden oluşturulmasına gerek kalmaz. Bu hem performans hem de zaman kaybını önleyecektir.

KAYNAKLAR

[1] **S.Wesley Changchien, Tzu-Chuen Lu**, 2001. Mining association procedure to support on-line recommendation by customers and products fragmentation, Taiwan

[2] **Klir, G.J. and Folger,T.A.** , 1988. Fuzzy Sets, Uncertainity and Information , Prentice Hall

[3] **Alpaydın, E.** ,2000. Zeki veri madenciliği, ham veriden altın bilgiye ulaşma yöntemleri , Bilişim Eğitim Semineri

[4] **Yonar, Y.B.** ,1999. Genel Amaçlı Bir Uzman Sistem, *Yüksek Lisans Tezi*, İ.T.Ü Fen Bilimleri Enstitüsü, İstanbul.

[5] **Edward S. Sazonov**, Powsiri Klinkhachorn, Hota V.S. GangaRao, and Udaya B. Halabe, 2002. Fuzzy logic expert system for automated damage detection from changes in strain energy mode shapes, Taylor & Francis Publishing, Volume 18, Number 1/2002, Pages 1 - 17.

EK - A

ID	Tarih	Doviz	Faiz	Borsa	ID	Tarih	Doviz	Faiz	Borsa
1	02.12.2003	1454169	27,84	16242	46	10.02.2004	1348606	24,16	17418,52
2	03.12.2003	1457619	27,84	16389,59	47	11.02.2004	1345300	24,16	18000,26
3	04.12.2003	1462160	27,84	16271,86	48	12.02.2004	1341164	24,16	18885,89
4	05.12.2003	1455116	27,84	16504,8	49	13.02.2004	1321265	24,16	19000,46
5	08.12.2003	1452649	27,84	16913,31	50	16.02.2004	1317509	24,16	19324,49
6	09.12.2003	1441914	27,84	16861	51	17.02.2004	1323113	24,16	19010,14
7	10.12.2003	1433525	27,84	16614,74	52	18.02.2004	1320252	24,16	19478,68
8	11.12.2003	1434632	27,84	16551,62	53	19.02.2004	1322274	24,16	18605,97
9	12.12.2003	1443481	27,84	16955,2	54	20.02.2004	1327253	24,16	18603,83
10	15.12.2003	1438623	27,84	17410	55	23.02.2004	1329280	24,16	18284,02
11	16.12.2003	1434879	27,84	17208,51	56	24.02.2004	1329990	24,16	18497,74
12	17.12.2003	1425887	27,84	17230,78	57	25.02.2004	1325906	24,16	18707,11
13	18.12.2003	1432007	27,84	17742,44	58	26.02.2004	1321065	24,16	18771,63
14	19.12.2003	1428638	27,84	18206,1	59	27.02.2004	1330297	24,16	18889,2
15	22.12.2003	1428217	27,84	18387,58	60	01.03.2004	1327679	22,6	18786,39
16	23.12.2003	1432911	27,84	18242,01	61	02.03.2004	1321167	22,6	19356,62
17	24.12.2003	1430911	27,84	18239,88	62	03.03.2004	1320596	22,6	19171,91
18	25.12.2003	1427843	27,84	17643,74	63	04.03.2004	1332769	22,6	19015,47
19	26.12.2003	1419346	27,84	17996,8	64	05.03.2004	1329357	22,6	19165,7
20	29.12.2003	1414344	27,84	17973,9	65	08.03.2004	1329601	22,6	19495,36
21	30.12.2003	1414467	27,84	18292,92	66	09.03.2004	1320780	22,6	19488,47
22	31.12.2003	1402567	27,84	18625,02	67	10.03.2004	1317540	22,6	19798,79
23	02.01.2004	1399998	26,06	19147,69	68	11.03.2004	1318973	22,6	19381,4
24	05.01.2004	1395983	26,06	19696,61	69	12.03.2004	1324223	22,6	19364,41
25	06.01.2004	1377990	26,06	19013,84	70	15.03.2004	1322177	22,6	19526,54
26	07.01.2004	1369962	26,06	19382,8	71	16.03.2004	1317509	22,6	19321,6
27	08.01.2004	1370241	26,06	19404,91	72	17.03.2004	1317569	22,6	19294,5
28	09.01.2004	1367887	26,06	19926,48	73	18.03.2004	1322213	22,6	19611,13
29	12.01.2004	1361470	26,06	19558,81	74	19.03.2004	1327673	22,6	20023,77
30	13.01.2004	1338697	26,06	19460,26	75	22.03.2004	1318535	22,6	20167,21
31	14.01.2004	1336082	26,06	18818,56	76	23.03.2004	1319323	22,6	20185,78
32	15.01.2004	1347561	26,06	18952,22	77	24.03.2004	1319571	22,6	20347,82
33	16.01.2004	1342679	26,06	18301,16	78	25.03.2004	1318793	22,6	20472,61
34	19.01.2004	1340468	26,06	17788,62	79	26.03.2004	1325489	22,6	20836,12
35	20.01.2004	1355695	26,06	18832,78	80	29.03.2004	1321958	22,6	20887
36	21.01.2004	1339430	26,06	18899,94	81	30.03.2004	1319591	22,6	20030,71
37	22.01.2004	1332689	26,06	18518,07	82	31.03.2004	1317611	22,6	20190,83
38	26.01.2004	1327332	26,06	18356,54	83	01.04.2004	1316538	21,6	20322,17
39	27.01.2004	1325741	26,06	17899,54	84	02.04.2004	1313062	21,6	20485,03
40	28.01.2004	1321924	26,06	17902,02	85	05.04.2004	1307617	21,6	20330,9
41	29.01.2004	1327445	26,06	17282,3	86	06.04.2004	1315617	21,6	20272,94
42	30.01.2004	1339619	26,06	17259,25	87	07.04.2004	1311386	21,6	20040,1
43	05.02.2004	1343450	24,16	17033,75	88	08.04.2004	1324256	21,6	19419,79
44	06.02.2004	1327696	24,16	16965,83	89	09.04.2004	1341950	21,6	19505,17
45	09.02.2004	1339229	24,16	17640,98	90	12.04.2004	1331045	21,6	19259,5
					91	13.04.2004	1350435	21,6	19259,5

EK – B

ID	Doviz	Faiz	Borsa	ID	Doviz	Faiz	Borsa
1	0.052	0.0	1.0	46	0.735	0.59	0.747
2	0.029	0.0	0.968	47	0.756	0.59	0.621
3	0.0	0.0	0.994	48	0.783	0.59	0.431
4	0.046	0.0	0.943	49	0.912	0.59	0.406
5	0.062	0.0	0.855	50	0.936	0.59	0.336
6	0.131	0.0	0.867	51	0.9	0.59	0.404
7	0.185	0.0	0.92	52	0.918	0.59	0.303
8	0.178	0.0	0.933	53	0.905	0.59	0.491
9	0.121	0.0	0.846	54	0.873	0.59	0.492
10	0.152	0.0	0.749	55	0.86	0.59	0.56
11	0.177	0.0	0.792	56	0.855	0.59	0.514
12	0.235	0.0	0.787	57	0.882	0.59	0.469
13	0.195	0.0	0.677	58	0.913	0.59	0.455
14	0.217	0.0	0.577	59	0.853	0.59	0.43
15	0.22	0.0	0.538	60	0.87	0.84	0.452
16	0.189	0.0	0.569	61	0.912	0.84	0.329
17	0.202	0.0	0.57	62	0.916	0.84	0.369
18	0.222	0.0	0.698	63	0.837	0.84	0.403
19	0.277	0.0	0.622	64	0.859	0.84	0.371
20	0.309	0.0	0.627	65	0.858	0.84	0.3
21	0.309	0.0	0.558	66	0.915	0.84	0.301
22	0.386	0.0	0.487	67	0.936	0.84	0.234
23	0.402	0.285	0.374	68	0.927	0.84	0.324
24	0.428	0.285	0.256	69	0.893	0.84	0.328
25	0.545	0.285	0.403	70	0.906	0.84	0.293
26	0.597	0.285	0.324	71	0.936	0.84	0.337
27	0.595	0.285	0.319	72	0.936	0.84	0.343
28	0.61	0.285	0.207	73	0.906	0.84	0.275
29	0.652	0.285	0.286	74	0.87	0.84	0.186
30	0.799	0.285	0.307	75	0.929	0.84	0.155
31	0.816	0.285	0.445	76	0.924	0.84	0.151
32	0.742	0.285	0.417	77	0.923	0.84	0.116
33	0.773	0.285	0.557	78	0.928	0.84	0.089
34	0.787	0.285	0.667	79	0.884	0.84	0.011
35	0.689	0.285	0.442	80	0.907	0.84	0.0
36	0.794	0.285	0.428	81	0.923	0.84	0.184
37	0.838	0.285	0.51	82	0.935	0.84	0.15
38	0.872	0.285	0.545	83	0.942	1.0	0.122
39	0.883	0.285	0.643	84	0.965	1.0	0.087
40	0.907	0.285	0.643	85	1.0	1.0	0.12
41	0.872	0.285	0.776	86	0.948	1.0	0.132
42	0.793	0.285	0.781	87	0.976	1.0	0.182
43	0.768	0.59	0.83	88	0.892	1.0	0.316
44	0.87	0.59	0.844	89	0.778	1.0	0.297
45	0.795	0.59	0.699	90	0.848	1.0	0.35
				91	0.723	1.0	0.35

EK – C

ID	Doviz	Faiz	Borsa
1	Dusuk	Dusuk	Yuksek
2	Dusuk	Dusuk	Yuksek
3	Dusuk	Dusuk	Yuksek
4	Dusuk	Dusuk	Yuksek
5	Dusuk	Dusuk	Yuksek
6	Dusuk	Dusuk	Yuksek
7	Dusuk	Dusuk	Yuksek
8	Dusuk	Dusuk	Yuksek
9	Dusuk	Dusuk	Yuksek
10	Dusuk	Dusuk	Yuksek
11	Dusuk	Dusuk	Yuksek
12	Dusuk	Dusuk	Yuksek
13	Dusuk	Dusuk	Yuksek
14	Dusuk	Dusuk	Orta
15	Dusuk	Dusuk	Orta
16	Dusuk	Dusuk	Orta
17	Dusuk	Dusuk	Orta
18	Dusuk	Dusuk	Yuksek
19	Dusuk	Dusuk	Orta
20	Dusuk	Dusuk	Orta
21	Dusuk	Dusuk	Orta
22	Orta	Dusuk	Orta
23	Orta	Dusuk	Orta
24	Orta	Dusuk	Dusuk
25	Orta	Dusuk	Orta
26	Orta	Dusuk	Dusuk
27	Orta	Dusuk	Dusuk
28	Orta	Dusuk	Dusuk
29	Yuksek	Dusuk	Dusuk
30	Yuksek	Dusuk	Dusuk
31	Yuksek	Dusuk	Orta
32	Yuksek	Dusuk	Orta
33	Yuksek	Dusuk	Orta
34	Yuksek	Dusuk	Yuksek
35	Yuksek	Dusuk	Orta
36	Yuksek	Dusuk	Orta
37	Yuksek	Dusuk	Orta
38	Yuksek	Dusuk	Orta
39	Yuksek	Dusuk	Orta
40	Yuksek	Dusuk	Orta
41	Yuksek	Dusuk	Yuksek
42	Yuksek	Dusuk	Yuksek
43	Yuksek	Orta	Yuksek
44	Yuksek	Orta	Yuksek
45	Yuksek	Orta	Yuksek

ID	Doviz	Faiz	Borsa
46	Yuksek	Orta	Yuksek
47	Yuksek	Orta	Orta
48	Yuksek	Orta	Orta
49	Yuksek	Orta	Orta
50	Yuksek	Orta	Dusuk
51	Yuksek	Orta	Orta
52	Yuksek	Orta	Dusuk
53	Yuksek	Orta	Orta
54	Yuksek	Orta	Orta
55	Yuksek	Orta	Orta
56	Yuksek	Orta	Orta
57	Yuksek	Orta	Orta
58	Yuksek	Orta	Orta
59	Yuksek	Orta	Orta
60	Yuksek	Yuksek	Orta
61	Yuksek	Yuksek	Dusuk
62	Yuksek	Yuksek	Orta
63	Yuksek	Yuksek	Orta
64	Yuksek	Yuksek	Orta
65	Yuksek	Yuksek	Dusuk
66	Yuksek	Yuksek	Dusuk
67	Yuksek	Yuksek	Dusuk
68	Yuksek	Yuksek	Dusuk
69	Yuksek	Yuksek	Dusuk
70	Yuksek	Yuksek	Dusuk
71	Yuksek	Yuksek	Dusuk
72	Yuksek	Yuksek	Dusuk
73	Yuksek	Yuksek	Dusuk
74	Yuksek	Yuksek	Dusuk
75	Yuksek	Yuksek	Dusuk
76	Yuksek	Yuksek	Dusuk
77	Yuksek	Yuksek	Dusuk
78	Yuksek	Yuksek	Dusuk
79	Yuksek	Yuksek	Dusuk
80	Yuksek	Yuksek	Dusuk
81	Yuksek	Yuksek	Dusuk
82	Yuksek	Yuksek	Dusuk
83	Yuksek	Yuksek	Dusuk
84	Yuksek	Yuksek	Dusuk
85	Yuksek	Yuksek	Dusuk
86	Yuksek	Yuksek	Dusuk
87	Yuksek	Yuksek	Dusuk
88	Yuksek	Yuksek	Dusuk
89	Yuksek	Yuksek	Dusuk
90	Yuksek	Yuksek	Dusuk
91	Yuksek	Yuksek	Dusuk

EK – D

1. Eger doviz=Yuksek(0.175) ve faiz=Orta(0.647) ise borsa=Orta(0.306)

2. Eger doviz=Yuksek(0.095) ve faiz=Yuksek(0.188) ise borsa=Orta(0.167)

3. Eger doviz=Yuksek(0.413) ve faiz=Yuksek(0.813) ise borsa=Dusuk(0.765)

4. Eger doviz=Yuksek(0.143) ve faiz=Dusuk(0.214) ise borsa=Orta(0.25)

5. Eger doviz=Yuksek(0.032) ve faiz=Orta(0.118) ise borsa=Dusuk(0.059)

6. Eger doviz=Orta(0.571) ve faiz=Dusuk(0.095) ise borsa=Dusuk(0.118)

7. Eger doviz=Orta(0.429) ve faiz=Dusuk(0.071) ise borsa=Orta(0.083)

8. Eger doviz=Yuksek(0.063) ve faiz=Orta(0.235) ise borsa=Yuksek(0.19)

9. Eger doviz=Orta(0.0) ve faiz=Orta(0.0) ise borsa=Dusuk(0.0)

10. Eger doviz=Orta(0.0) ve faiz=Orta(0.0) ise borsa=Orta(0.0)

11. Eger doviz=Dusuk(0.333) ve faiz=Dusuk(0.167) ise borsa=Orta(0.194)

12. Eger doviz=Dusuk(0.667) ve faiz=Dusuk(0.333) ise borsa=Yuksek(0.667)

13. Eger doviz=Yuksek(0.048) ve faiz=Dusuk(0.071) ise borsa=Yuksek(0.143)

14. Eger doviz=Orta(0.0) ve faiz=Dusuk(0.0) ise borsa=Yuksek(0.0)

ÖZGEÇMİŞ

Alper Küçükural 1977 yılında Gölcük'te doğdu. 1994 yılında İstanbul Tuzla Endüstri Meslek Lisesi Bilgisayar Blümünden mezun oldu. 1995 yılında İstanbul Teknik Üniversitesi Fen Edebiyat Fakültesi, Matematik Mühendisliği bölümünde lisans öğrenimine başladı. Matematik Mühendisliği bölümünden 2000 yılında mezun oldu, aynı yıl İstanbul Teknik Üniversitesi, Sistem Analizi programında yüksek lisans öğrenimine başladı. Halen, Koçsistem Bilgi ve İletişim Hiz.A.Ş. 'de Uygulama Geliştirme Danışmanı olarak görev yapmaktadır.