İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TÜMEVARIM ÖĞRENME TEKNİKLERİNDEN C4.5'İN İNCELENMESİ

YÜKSEK LİSANS TEZİ

Müh. Savaş YILDIRIM

Anabilim Dalı: Fen Bilimleri Enstitüsü
Programı: Savunma Teknolojileri

Tez Danışmanı: Doç.Dr. Şakir KOCABAŞ

MAYIS 2003

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TÜMEVARIM ÖĞRENME TEKNİKLERİNDEN C4.5'İN İNCELENMESİ

YÜKSEK LİSANS TEZİ

Müh. Savaş YILDIRIM

Tezin Enstitüye Verildiği Tarih : 5 Mayıs 2003

Tez Danışmanı : **Doç.Dr. Şakir KOCABAŞ**

Diğer Jüri Üyeleri **Doç.Dr. Sedef KENT (İ.T.Ü.)**

Doç.Dr. Serhat ŞEKER (İ.T.Ü.)

MAYIS 2003

ÖNSÖZ

Bu tez çalışmamda her türlü desteği gösteren danışman hocam Doç.Dr. Şakir KOCABAŞ'a, Bilgi Üniversitesi Bilgisayar Bilimleri Bölümü Öğretim Görevlileri'ne ve Ahu Sıla BAYER'e teşekkürlerimi sunarım.

Mayıs 2003 Savaş YILDIRIM

İÇİNDEKİLER

ÖNSÖZ ii
KISALTMALAR iv
TABLO LİSTESİ v
ŞEKİL LİSTESİ vi
ÖZET viii
SUMMARY xi

1. GİRİŞ 1

1.1 Öğrenme Elemanları 1
1.2 Sınıflandırma 4
1.3 Karar Ağaçları ve Kural Türetme 5
1.4 Yapay Sinir Ağları 5

2. KARAR AĞAÇLARI 7

2.1 Karar Ağacı Oluşturma 9
2.1.1 Böl ve Elde Et (Divide and Conquer) 9
2.1.2 ID3 Algoritması 12
2.2 C4.5: Karar Ağacı Eğitme Algoritması 16
2.2.1 Bölünme-Dallanma bilgisi (Split Information) 16
2.2.2 Sayısal Özellikler 18
2.2.3 Kayıp Veriler 19
2.3 Ağacın Testi 28
2.4 Ağaç Budama Metotları 30
2.5 Sayısal Özellikler için bir iyileştirme 38

3. KARAR AĞACI OLUŞTURMA UYGULAMASI 42

3.1 Amaç 42
3.2 Java Programlama Dili 42
3.2.1 Çalışma Disiplini 43
3.2.2 Gelişim evreleri 45
3.2.3 Dilin Yapısı 45
3.2.4 Java'da Grafik ve Arayüz 54
3.3 Program 55
3.3.1 ARFF Format 56
3.3.2 Programın Birimleri 57
3.3.3 Programın Çalıştırılması 63

4. VERİ KÜMELERİ İLE ÇALIŞMA 66

4.1 Amaç 66
4.2 Veri Kümeleri 66
4.3 Sonuçlar 68

5. SONUÇ VE TARTIŞMA 75

KAYNAKLAR 78

ÖZGEÇMİŞ 79

KISALTMALAR

KA	:Karar Ağacı
YSA	:Yapay Sinir Ağları
JVM	:Java Virtual Machine
JSM	:Java Sanal Makinesi
ARFF	: Attribute Relation File Format

TABLO LİSTESİ **Sayfa No**

Tablo 2.1	Örnek bir olay kümesi.....	11
Tablo2.2	Tablo2.1'deki veri yığının kazanç oranlarının hesaplanması.....	18
Tablo 2.3	Hava durumunun tenis oynamaya müsait olup olmadığına ilişkin veri kümesi.....	22
Tablo 2.4	Tablo2.3'teki verilerin bilgi kazanç oranları.....	23
Tablo 2.5	Tablo2.3'teki 6. satır.....	23
Tablo 2.6	6.satırın hava durumu özelliğinin kayıp olması.....	24
Tablo 2.7	Tablo2.5'in 6. satırındaki hava durumu bilgisinin.....	24
Tablo2.8	Hava durumu=güneşli alt ağacının kapsadığı kümenin ağırlık bilgileriyle gösterilmesi.....	25
Tablo 2.9	kayıp verileri olan, sınıflandırılacak örnek bir olay.....	27
Tablo 2.10	Sınıflandırılacak yeni bir olay.....	28
Tablo 2.11	İkili bölme ve Üçlü bölmede bilgi kazançları.....	39
Tablo 2.12	Sıcaklık bilgisine göre düzenlenmiş tablo.....	40
Tablo 3.1	Java'nın gelişim evreleri.....	45
Tablo 3.2	İlkel tiplerin özelikleri.....	47
Tablo 3.3	İlkel tiplerin varsayılan değerleri.....	49
Tablo 4.1	Veri kümelerinden elde edilen karar ağaçlarının durumu.....	69
Tablo 4.2	Budama oranının artması ile ağacın verdiği tepki.....	70
Tablo 4.3	Eğitim kümesi ile test edilen Karar Ağaçlarının budama oranına göre doğruluk sonuçları (%).....	71
Tablo 4.4	Karar Ağaçlarının test kümesi ile test edilmesinin budama oranına göre sonuçları.....	72
Tablo 4.5	Tüm verilerdeki doğruluk oranlarının ortalaması.....	73
Tablo 4.6	Eğitim kümesi ile sağlama kümesinin budama oranlarına verdiği doğruluk oranları.....	73

ŞEKİL LİSTESİ **Sayfa No**

Şekil1.1 Simon'un tanımladığı öğrenme sistemi..... 2
Şekil 1.2 Öğrenmenin metotlara göre sınıflandırılması..... 3
Şekil 2.1 Koordinat sistemleriyle ifade edilen olay kümeleri..... 8
Şekil 2.2 Şekil 2.1'deki veri kümesi ile yaratılan ağaç..... 8
Şekil 2.3 Hunt'ın ağaç oluşturma metodu..... 10
Şekil 2.4 Tablo 2.1'in büyüklük sınıfına göre bölünmesi..... 11
Şekil 2.5 Şekil 2.4'deki ağacın bölünmüş kümeleri biçim özelliğine göre tekrar bölünmesi sonucu oluşan ağaç..... 12
Şekil 2.6 Şekil2.5'teki ağacın sınıflandırılmamış olaylarının renk özelliği ile sınıflandırılması..... 12
Şekil 2.7 ID3 Algoritması akış diyagramı..... 14
Şekil 2.8 Tablo2.1'in ID3 ile oluşturulmuş KA 16
Şekil 2.9 Tablo2.1'in etiket özelliğine göre bölümlenmesi..... 17
Şekil 2.10 Tablo2.3'teki veri kümesinin C4.5 ile yaratılması..... 23
Şekil 2.11 Kayıp veri göz önüne alınmadan oluşturulan alt ağaç..... 26
Şekil 2.12 Kayıp veri göz önüne alındığında oluşan alt ağaç..... 26
Şekil 2.13 Hata içeriği ile ağacın gösterilmesi..... 27
Şekil 2.14 Eğitim kümesi ile test kümesinin ağaç boyuna göre doğruluk performansı..... 31
Şekil 2.15 Ağaç budama..... 33
Şekil 2.16 Ağaç budama algoritmasına ilk yaklaşım..... 33
Şekil 2.17 Budamanın ağırlık bilgisi göz önüne alınarak yapılması..... 34
Şekil 2.18 Ağırlık bilgisi ile yapılan budama algoritması..... 35
Şekil 2.19 Ağaç yükseltme..... 36
Şekil 2.20 Dal budama..... 37
Şekil 3.1 Java Sanal Makinası..... 44
Şekil 3.2 Java kodunun geçirdiği evreler..... 44
Şekil 3.3 İlk java programı..... 44
Şekil 3.4 Java'da geçerlilik alanı..... 47
Şekil 3.5 Java'da yeni bir sınıf oluşturma..... 48
Şekil 3.6 Yeni bir sınıfın global değişkenlerini..... 48
Şekil 3.7 Geri dönüş tipinin belirlenmesi..... 50
Şekil 3.8 Geri dönüş tipine bir örnek..... 50
Şekil 3.9 Dizi yaratılması..... 51
Şekil 3.10 Dizilerin bellekte yaratılma biçimleri..... 51
Şekil 3.11 Bir java sınıfı..... 52
Şekil 3.12 Türetilen bir sınıf..... 52
Şekil 3.13 Sınıf Türetme..... 52
Şekil 3.14 Yeni bir sınıfta gizli kalıtım..... 53
Şekil 3.15 Uygulamanın çıktısı..... 53
Şekil 3.16 Grafiksel ara yüz örneği..... 55
Şekil 3.17 Programın aşamaları..... 57
Şekil 3.18 ARFF formatında hava durumu verisi..... 56
Şekil 3.19 Programın Birimleri ve Çalışma Düzeni..... 58
Şekil 3.20 Veri okuma birimi..... 59
Şekil 3.21 Ağaç oluşturma birimi..... 59
Şekil 3.22 Özellik değerlerini temsil eden birim..... 60
Şekil 3.23 Düğümleri temsil eden birim..... 60
Şekil 3.24 Düğümlerin rekürsif olarak birbirlerini göstermesi..... 61
Şekil 3.25 Bilgisayarda grafik oluşturma..... 61
Şekil 3.26 Ağacın veri yapılı halinden x-y koordinat verilerini oluşturma..... 62

Şekil 3.27 Ana program ekranı..... 63
Şekil 3.28 Veri kümesini bulan ara yüz..... 64
Şekil 3.29 Programın kullanıcı ara yüz görüntüsü..... 65
Şekil 4.1 Budama ve doğruluk testi..... 74

TÜMEVARIM ÖĞRENME TEKNİKLERİNDEN C4.5'İN İNCELENMESİ

ÖZET

Makine öğrenmesinin önemli tekniklerinden biri olan C4.5 (Karar Ağacı Eğitme Algoritması) ve eski uyarlaması ID3 ile farklılıkları bu tez kapsamında ele alınmıştır. Sadece C4.5 sisteminin getirdiği yaklaşımlar değil diğer yaklaşımlar da incelenmiştir. Tıp, Savunma, Ekonomi vb. bir çok alanda kullanılan karar destek sistemleri için önemli bir noktada duran karar ağacı oluşturma algoritmaları bu alanlarda çok faydalı işler görmektedir. Quinlan tarafından 1993'te C dilinde yazılan bu algoritma bu tez kapsamında Java programlama dilinin grafik desteği ile yazılmış ve bir çok deney yapılarak yaklaşımın zayıf ve güçlü yanları belirtilmiş ve bazı iyileştirmeler önerilmiştir.

Veri kümesindeki olaylar ilgili özelliklerine göre sorgulanır ve kurallar oluşturulur. Buradaki amaç birbirinden bağımsız duran veriler arasındaki ilişkiyi çözmektir. Her veri kümesinde gizlenmiş kurallar mevcuttur ve karar ağacı oluşturulurken bu kurallar ortaya çıkarılır. Karar ağacı oluşturmadaki en önemli aşama, veri kümesini bölerken soruların hangi sırayla sorulacağı aşamasıdır Bunun için her bir özelliğin gücü ya da bilgi kazancı, entropi formüllerine dayanarak hesap edilir. Kategorik verilere sahip özelliklerin bilgi kazancının hesap edilmesi sayısal verilere göre kısmen daha kolaydır. Sayısal özelliklerin bilgi kazancı sayısal aralığın belli bir noktadan ikiye bölünmesiyle hesap edilir. Bu çalışmada sayısal özelliklerin, ikiden fazla bölgeye bölünerek incelenmesi durumunda olumlu sonuçlar verdiği görülmüştür. Veri kümesi en güçlü özelliği içeren soru ile sorgulanarak bölünür ve bu işlem yinelenerek karar ağacı oluşturulur.

Yaprak oluşturma işlemi için belli koşulların oluşması beklenir. Koşulların oluştuğu ilk aşamada yaprak oluşturulur ve yaprağa ilişkili olduğu sınıfın içerik bilgisi eklenir. Yaprak oluşturmaya yönelik ilk yaklaşım şöyledir: Bölünen veri kümesindeki bütün olaylar aynı sınıfa ait ise bu noktada dur ve o sınıfın içeriğine sahip bir yaprak yarat. Yaprak yaratmak için koşulların nasıl belirleneceği önemli bir sorun olduğundan bu konu ayrıntılı olarak tartışılmıştır.

Önemli başlıklardan biri olan ağaç budama işlemi ikiye ayrılır: Ön budama (pre-pruning) ve sonradan budama (post-pruning). Bu tekniklerden ön budama ağaç oluşturulurken yapılır ve yaprak oluşturma koşullarına eklenir. Sonradan budamada, ağaç oluşturulduktan sonra her bir yaprağa, kapsadığı olay sayısına göre bir ağırlık bilgisi verilir ve bu bilgi kullanılarak hata tabanlı bir budama yapılır. Ancak C4.5'in budama oranını girdi olarak kabul etmesi bir eksikliktir. Bu yüzden ağacın doğruluk oranını maksimum yapabilecek budama oranını hesaplayan bir birimin oluşturulması gerekir.

Ağaç budama ilk bakışta ağacın performansını olumsuz etkiler gibi görünmektedir. Ancak ağaç bağımsız veriler tarafından test edildiğinde bunun tam tersi olduğu görülmektedir. Makine öğrenmesi alanında çalışan araştırmacılar tarafından oluşturulmuş veri kümeleri ile deneyler yapılmış ve verilerin hangi aralıkta budanması gerektiği hesap edilmiş ve önerilmiştir. Bunun yanında her ağacın kendine özgü budama oranı olduğu ve budamadaki en önemli unsurun yaprakların taşıdığı olay sayısı olduğu gözlemlenmiştir. Budama işleminde yaprak sayısından ziyade yaprakların taşıdığı yükler göz önüne alınmıştır.

Karar ağacının doğruluğunu hesaplamak için ilgili veri kümesi ikiye bölünür. Bu kümelerden biriyle ağaç oluşturulur ve diğeriyle ağaç test edilir. Böylelikle bu karar mekanizmasına ne kadar güvenileceği bilgisi elde edilmiş olur. Bu ikili bölme yönteminin yanında başka yöntemlerde araştırılmıştır. Çünkü veri kümesinin bölünecek kadar büyük olmaması karar ağacını test etmede önemli bir sorunlar yaratmaktadır.

Bu algoritmaların programlanmasında Java programla dilinin desteğine başvurulmuştur. Bu sebepten dolayı Java programlama dilinin yapısına ayrıntılı olarak değinilmiştir. Nesneye yönelik programlama dilinin getirdiği avantajlarla, ağaç yapısı çok kolay bir şekilde ifade edilmiştir. Buna ek olarak oluşturulan ağaçlar Java'nın grafik kütüphanesi yardımıyla ve grafik ara yüz desteği ile iki boyutlu uzayda dinamik olarak çizdirilmiştir. Veri kümesini seçme, ağacı oluşturma, budama, test etme gibi tüm aşamalar görsel hale getirilmiştir.

Sonuç aşamasında karar ağacının tüm başlıkları (Ağaç oluşturma, bilgi kazancının hesaplanması, sayısal özellikler, budama, sağlama vb.) değerlendirilmiştir. C4.5'in ID3'e getirdiği yeni yaklaşımlar ele alınmış, bu sistemin güçlü ve zayıf yanları belirtilmiş ve gerektiğinde bazı iyileştirmeler öne sürülmüştür.

RESEARCH ON C4.5 AS ONE OF THE INDUCTIVE LEARNING TECHNIQUES

SUMMARY

The thesis in hand deals with C4.5 (Decision Tree Construction Algorithm) as one of the most significant techniques of machine learning, and how it differs from its older version ID3. With this aim in mind, not only the approaches provided by C4.5 but also other approaches are examined. The decision tree algorithms are useful in a variety of spheres from defense to medicine or economics; and bear a vital importance for decision support systems in these areas. Written by Quinlan in 1993 in C programming language, this algorithm is written in this thesis with the graphic support of Java programming language and has been subject to many experiments that point out the strong and insufficient aspects of the approach. Some suggestions of improvement are also included.

The instances in data sets are inquired according to their attributes and the rules are created. The aim of this inquiry is to figure out the relationship between data that seem to be totally distinct from each other. In each of the data sets some rules are hidden inherently; and in the course of creating the decision tree, these rules are revealed. The most important stage in the creation of decision trees is the order of the questions to be asked during the process of dividing the data sets. For this reason, power and information gain of each attribute is calculated on the basis of formulas of entropy. Calculating the information gain of the nominal attributes is relevantly easier when compared to the continues attributes. The information gain of numerical attributes is calculated by dividing the numerical range into two parts from a particular point. In this thesis it has been observed that analyzing the numerical attributes by dividing them into more than two parts leads us to some positive results. The data set is inquired with the question that contains the most powerful attribute; and via the repetition of this process the decision tree is created.

For the process of leaf creation the appropriate conditions must occur. As soon as the conditions are proper, leaf is created and content information is added to the leaf about the class to which it is related to. The first approach to leaf creation is as follows: If all the instances in the divided data set belong to the same class, stop here and create a leaf that possesses the content of that class. The determination of

the proper conditions in leaf creation has been regarded as an issue of magnitude and thus has been discussed in detail.

Being one of the important titles, the process of pruning is analyzed in two parts: pre-pruning and post-pruning. Pre-pruning process is realized in the course of tree creation and is added to the leaf creation conditions. In post-pruning, after the tree is created each leaf is given weight information according to the number of instances that it contains. Next this information is used while pruning is done on error basis. However, the fact that C4.5 accepts the pruning rate as an input is a deficiency of this algorithm. Due to this fact it is necessary to create a unit that calculates the pruning rate necessary for the maximum accuracy rate.

At first sight tree pruning may seem to affect the performance of the tree negatively. In fact, testing the tree by using independent data that do not create the tree is enough to prove the opposite. Researchers in the field of machine learning have created data sets and used these sets in their experiments; coming up with the results of which range is the most appropriate for pruning the data. In addition, it has been observed that each tree has its own unique pruning rate and that the most important element in pruning is the number of instances that a leaf contains. In the process of pruning, rather than the number of leaves, the weight that the leaves bear is taken into consideration.

In order to calculate the accuracy of the decision tree, the related data set is divided into two. One of these sets serves the creation of the tree whereas the other is used to test it. As a result, we learn to what extent we can rely on this decision mechanism. In addition to this division technique, new techniques have been searched. That is simply because in the case of the data set not being big enough to be divided, vital problems occur while testing the decision tree.

The support of Java programming language has been beneficial in the programming of these algorithms. That is why the structure of Java programming language has been dealt in this thesis in detail. Thanks to the advantages of object-oriented programming, the tree structure was able to be defined much more easily. Furthermore, the trees that were created could be dynamically graphed in two-dimensional space with the help of Java's graphic library and graphic inter-face support. All the stages of data set selection, creating the tree, pruning and testing have been visualized.

In the conclusion part, all the titles of decision tree (creation of trees, calculation of information gain, numerical attributes, pruning, validation, etc.) have been evaluated. New approaches of C4.5 to its older version ID3 have been taken into consideration, the strong and insufficient aspects of the system have been identified and some suggestions of improvement have been offered.

1. GİRİŞ

Öğrenme yapay zeka'nın ilgilendiği önemli bir araştırma alanıdır ve bu konuda çok yönlü araştırmalar sürdürülmektedir. Zeki sistemlerin bilgilerini artırma ve yeteneklerini geliştirme süreçleri öğrenme olarak tanımlanır. Öğrenme süreci, sağduyu oluşturma, tasvir etme yeteneği, tanımlama yeteneği, teknik ve teorik bilgi kazanımı olarak tanımlanır.

Öğrenmenin tanımı bilim adamları tarafından farklı şekillerde dile getirilmiştir. **Simon** : "Benzer görevleri tekrarlarken, bir sistemdeki daha etkili olabilecek herhangi bir değişime öğrenme deriz. Bu değişim tersinmez olmalıdır. Ve ayrıca anlama ve öğrenme arasında sıkı bir bağ kuruludur. Diğer insanlardan öğrendiklerimizin çoğunu doğal dil yoluyla elde ederiz". Michalski: "Deneyimlerin gösterim biçimlerini oluşturma ve bunları değiştirmeye öğrenme denir." Bu tanımla Simon'un öğrenme için kullandığı "performans" ifadesi, yerini "deneyimlerin gösterilimi" ifadesine bırakmıştır [3].

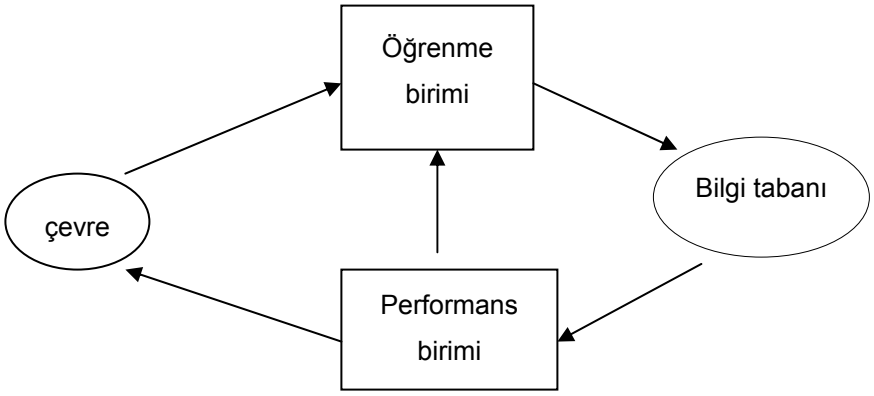
Öğrenme, anlama ve zeka birbirlerine sıkı sıkıya bağlı kavramlar olduğundan öğrenmenin tanımını yaparken biraz da zekadan bahsetmek gerekir. Lenat ve Feigenbaum zekayı, geniş arama alanı içinde gizlenmiş olan çözümü, hızlı bulabilen güç olarak tanımlıyor. Ancak Feigenbaum zekayı şu şekilde tekrar tanımlar: "Zeka arama yapmaktan ziyade belli bir amaç için gereken yararlı bilgileri kurabilme anlamına, yani "bilgi kurma" anlamına gelmektedir [3].

Fatmi ve Young zekayı fiziksel ve biyolojik açıdan değerlendirdiler. Zeka, daha önce düzensiz algılanan bir sistemi düzenli bir şekilde algılayabilme yeteneğidir. Bir başka deyişle zeka düzensiz bir sistemi düzenlemeyi başaran bir takım yönergeleri anlayabilme ve yaratabilme yeteneğidir [3].

1.1 Öğrenme Elemanları

Simon çok yararlı bir sistem tanımlar. Bu öğrenme sistemi şekil1.1de de görüldüğü gibi dört ana birimden oluşmaktadır: Çevre birimi, öğrenme birimi, bilgi tabanı ve performans birimi. Çevre birimi öğrenme birimine bazı bilgiler sunar. Öğrenme birimi ise

bu bilgiyi kullanarak bilgi tabanında bazı ilerlemeler yaratır. Performans birimi ise bu bilgiyi kullanarak kendi görevini gerçekleştirir. Bu işlemler sırasında kazanılan bilgi geri besleme olarak öğrenme birimine gönderilir. Ve bu sayede çevre birimi de değişime uğrar. Öğrenme sistemlerinin bu model üstünden birbirleriyle karşılaştırılmaları daha kolay olur. [3]

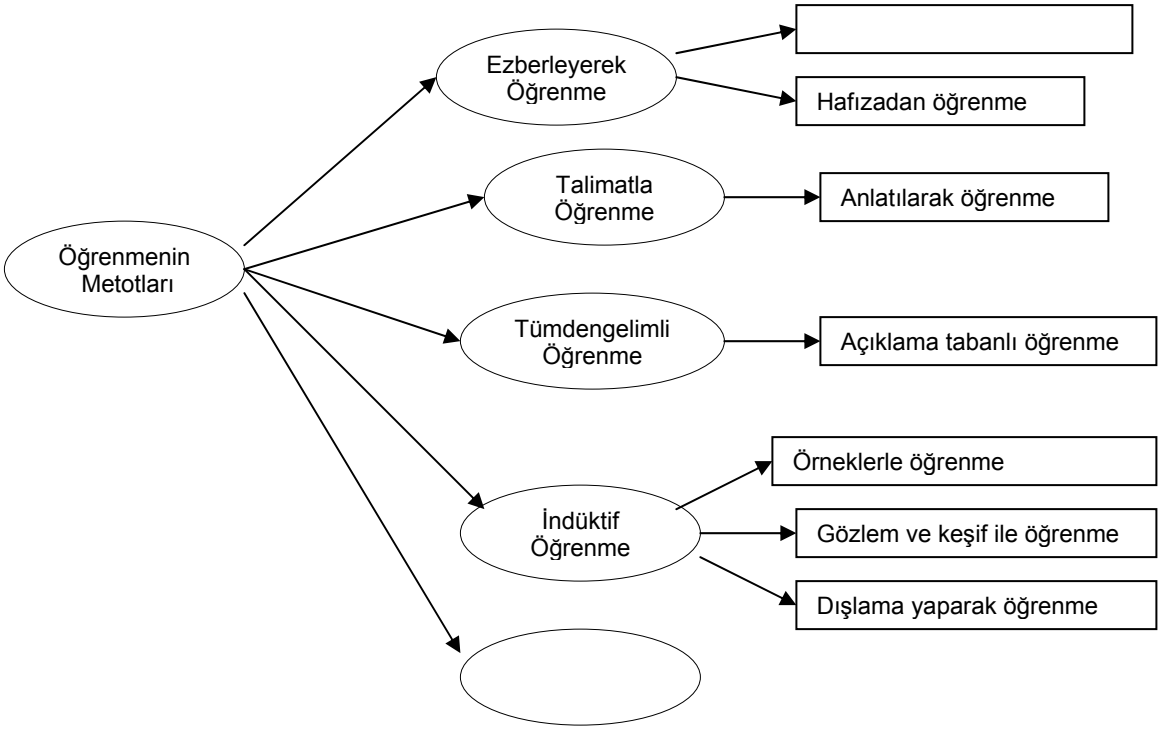


Şekil1.1 Simon'un tanımladığı öğrenme sistemi.

Öğrenmeyi sınıflandırmadaki en önemli kriter stratejik açıdan bakılırsa çıkarım mekanizmasıdır. Bazı sitemler çıkarım kullanırken, bazı sistemler ise kullanmamaktadır. Çıkarım kullanan sistemlerin performans kapasitelerinde her zaman bir artış olduğu görülmüştür.

Şekil 1.2 öğrenmenin genel stratejilerini göstermektedir.

- Ezberleme
- Talimatla Öğrenme
- Tümdengelimle Öğrenme
- İndüktif Öğrenme
- Analojik Öğrenme



Şekil 1.2 - Öğrenmenin metotlara göre sınıflandırılması

Ezberlemede, çevre birimi performans seviyesinde kesin bilgiler sağlar. Dolayısıyla herhangi bir çıkarıma gerek yoktur. İki türlü ezberleme vardır: Programlayarak ve hafızada tutarak. Anlatılarak öğrenmede, çevreden sağlanan bilgi genel ve soyuttur. Öğrenme birimi detayları tamamlayabilmek için bazı çıkarımlar yapar. Makine öğrenmesinin bir hedefi de tavsiye ve talimat kabul edebilen bir sistem kurabilmektir. Bu yetenek bilgiyi saklama ve efektif olarak kullanmayı mümkün kılmaktadır. Bununla ilgili yazılmış TEIRESIAS ve FOO isimli iki program vardır [3].

Tümdengelimli öğrenme stratejisi bilginin tekrar formüle edilmesini, derlenmesini ve makro operatörlerin yaratımını içerir. Tümdengelimli metotları kullanan hesaplamalı sistemlerden bazıları şunlardır: SOAR, LEX ve PET [3].

Efektif öğrenme gözlemlerden, keşiflerden ve örneklerden yararlanır. Örneklerle öğrenen sistemlerde bazı örnek kümeleri verilirken bunun hemen karşısına söz konusu kavramın karşıt örnek kümesi koyulur ve öğrenen sistem genel kavram tanımı içerir. Bu strateji benzerlik tabanlı öğrenme ve kavramsal sınıflandırma metotlarını içerir.

Genetik algoritmayla öğrenme de bu kategoriye dahil edilebilir. ID3 [3], C4.5 [8] ve CLUSTER/2 [3] sistemleri de tümevarım öğrenme sistemleri olarak bahsedilebilir.

Keşif ve gözlemlerden öğrenme aynı zamanda "unsupervised" (denetimsiz) öğrenme olarak ta adlandırılabilir. Bu teknik kalitatif ve kantitatif keşif, teori oluşturma,teori yenileme ve sınıflandırma gibi metotlar içerir. Keşif ile öğrenen sistemlerin bazıları şunlardır: AM, EURISKA, BACON, STAHL, GLAUBER, IDS, KEKADA, BR-3 ve ASTRA [3].

Analojik öğrenmede çevreden sağlanan bilgi sadece ve sadece benzer işi olan bir göreve bağlıdır. Bu teknik tümdengelimli ve tümevarım metotları kullanır.

Öğrenen sistemlerin sınıflandırılmaları o sistemlerin kazandığı bilgi tipine göre yapılırsa şöyle bir liste oluşur: Cebirsel parametreli sistemler (algılayıcılar, sinir ağları), karar ağaçları (basit uzman sistemler), biçimsel gramer, üretim kurallı sistemler (kural tabanlı uzman sistemler), çizge ve ağ, çerçeveler, şemalar ve çerçeve tabanlı sistemler.

1.2 Sınıflandırma

Dünyadaki bütün olgular insanlar tarafından yorumlanırken sınıflandırma en önemli bir faktör olur. Önce bir sınıf tanımı yapılır ve olgular bu sınıf tanımı altında kategorilere dahil edilir. Canlılar, hayvanlar ve bitkiler şeklinde ikiye sınıflandırılırken maddenin hali sıvı, katı ve gaz şeklinde sınıflandırılır. Bu aşamada yapılan ön tanımın ardından yeni bir olgunun hangi sınıfa ait olduğu kurallarla belirlenir. "Eğer bir madde içine konulduğu kabın şeklini alıyorsa sıvıdır" bir kuraldır. Bu aşamada en önemli olay sınıflandırma kurallarını oluşturmaktır. Bu kurallar oluştuktan sonra yeni bir olayın sınıflandırması mümkün olmaktadır. Sınıflandırma için şu örnekler sıralanmıştır:

- Bir hastanın kanser olup olmadığını sınıflandırma
- Başvuran kişiye ya da kuruma kredi verilip verilmeyeceğini sınıflandırma
- Tüm koşularının spor yapmak için uygun olup olmadığını sınıflandırma
- Toplumdaki bireyler sosyoekonomik yapısına göre hangi partiye oy verdiğini sınıflandırma

1.3 Karar Ağaçları ve Kural Türetme

Öğrenme elemanlarından karar ağaçları yöntemi en güçlü ve en yaygın sınıflandırma ve öngörü araçlarından birisidir. Ağaç yapılı yöntemlerin sık kullanılmasının nedeni ise yapay sinir ağlarının tersine ağaç yapılarının kuralları ifade edebilmesinden kaynaklanmaktadır. Oluşan kurallar çok sade ve nettir. Eğer Ağaç çok fazla büyük değilse her insanın kolaylıkla okuyup anlayacağı türden bir sınıflandırma gerçekleştirir. Sık kullanılan bazı oyunlarda, karar ağacı (KA) bir dizi soru sorup bunların cevapları doğrultusunda hareket ederek en kısa sürede sonuca gider. Bu yapı sorduğu sorulara aldığı cevapları toplayarak kurallar oluşturur. Buna paralel olarak sorulara aldığı cevaplardan diğer sorusunu belirler.

Sorulacak sorular ve bu sorulara gelebilecek cevapların yönlendirdiği başka soruların bulunduğu bir ağaç yapısı olarak adlandırılan karar ağaçları ile değerlendirme yaparken yeni gelen bir kayıt, ağacın kökünden girer. Kökte test edilen bu yeni kayıt test sonucuna göre bir alt düğüme gönderilir. Ağacın belirli bir yaprağına gelene kadar devam eder. Kökten her bir yaprağa giden tek bir yol vardır ki bu yol kuralı oluşturur. Ağaçtaki bir çok yaprak aynı sınıflandırmayı yapabilir ancak her bir yaprak farklı sebeplerden dolayı bu sınıflandırmayı yapar. Karar ağaçlarının güçlü olduğu noktalar şunlardır:

- Kolay anlaşılır sonuçlar üretir
- Çok fazla işlem yapmadan sonuca ulaşır
- Hem sayısal hem kategorik veriler için işlem yapabilir.
- Her programlama dilinde kolaylıkla ifade edilebilmektedirler.

Bölüm 2'de KA ayrıntılı olarak ele alınmıştır.

1.4 Yapay Sinir Ağları

Yapay sinir ağları (YSA) beynin bazı fonksiyonlarını ve özellikle öğrenme yöntemlerini benzetim yolu ile gerçekleştirmek için tasarlanır ve geleneksel yöntem ve bilgisayarların yetersiz kaldığı sınıflandırma, kümeleme, duyu-veri işleme, çok duyulu makine gibi alanlarda başarılı sonuçlar verir. Yapay sinir ağlarının özellikle tahmin problemlerinde kullanılabilmesi için çok fazla bilgi ile eğitilmesi gerekir. Ağların eğitimi için çeşitli algoritmalar geliştirilmiştir. Yapay sinir ağlarının temel yapısı, beyne, sıradan bir

bilgisayarınkinden daha çok benzemektedir. Yine de birimleri gerçek nöronlar kadar karmaşık değil ve ağların çoğunun yapısı, beyin kabuğundaki bağlantılarla karşılaştırıldığında büyük ölçüde basit kalmaktadır. Çünkü insan beyninde yaklaşık 10^{12} adet nöron vardır ve aradaki bağlantıları da göz önüne alırsak bunun günümüz teknolojisi ile sağlanmasının mümkün olmadığı görülür.

YSA'lar Giriş ve çıkışları olan kara kutular gibi düşünülebilirler. Bu kara kutunun işlevi, basitçe, matematiksel bir fonksiyonu temsil etmek şeklinde açıklanabilir. Ama aslında bu fonksiyonun tam olarak bir matematiksel karşılığının olması gerekmez. Zaten olsaydı YSA kullanmaya gerek kalmazdı.

Yapay sinir ağlarındaki her bir işlem birimi, basit anahtar görevi yapar ve şiddetine göre, gelen sinyalleri söndürür ya da iletir. Böylece sistem içindeki her birim belli bir yüke sahip olmuş olur. Her birim sinyalin gücüne göre açık ya da kapalı duruma geçerek basit bir tetikleyici görev üstlenir. Yükler, sistem içinde bir bütün teşkil ederek, karakterler arasında ilgi kurmayı sağlar. Yapay sinir ağları araştırmalarının odağındaki soru, yüklerin, sinyalleri nasıl değiştirmesi gerektiğidir. Bu noktada herhangi bir formdaki bilgi girişinin, ne tür bir çıkışa çevrileceği, değişik modellerde farklılık göstermektedir. Diğer önemli bir farklılık ise, verilerin sistemde depolanma şeklidir. Nöral bir tasarımda, bilgisayarda saklı olan bilgiyi, tüm sisteme yayılmış küçük yük birimlerinin birleşerek oluşturduğu bir bütün evre temsil etmektedir. Ortama yeni bir bilgi aktarıldığında ise, yerel büyük bir değişiklik yerine tüm sistemde küçük bir değişiklik yapılmaktadır. Günümüzde gelişmiş sinir ağı uygulamaları ya geleneksel bilgisayarlar üzerinde yazılım benzetimleri kullanılarak, veya özel donanım içeren bilgisayarlar kullanarak gerçekleştirilmektedir. Kredi risk değerlemesinden imza kontrolü, mevduat tahmini ve imalat kalite kontrolüne kadar uzanan uygulamalar yazılım paketlerinden faydalanılarak yapılmaktadır

Yapay sinir ağları veri madenciliğinde ve karar destek sitemlerinde başarılı bir performans göstermiştir. YSA sınıflandırma, demetleme ve kestirim amaçları ile kolaylıkla kullanılabilecek genel amaçlı ve güçlü araçlardır. Ekonomiden tıbba her alanda kullanılmaktadır.

2. KARAR AĞAÇLARI

Kavram öğrenme ya da tümevarım algoritmaları karar ağacı biçiminde sınıf tanımı yaratırlar. Bu karar ağacı pratik önemi büyük olan bir çok problemi çözebilmektedir. Karar ağacı verili bilgi yığınının daha düzenli ifade edilmesidir. Temel olarak üç birim vardır:

- Düğüm
- Dal
- Yaprak

Düğüm soruları ifade eder, dal bu soruların cevaplarını temsil eder ve yaprak ise kararın verildiği sınıfı ifade eder. Öyleyse ağacın ilk düğümü ile sorular sorulmaya başlar ve dalları olmayan düğümler ya da yapraklara gelene kadar bu sorular devam eder. Karar ağacının if-then yapıları gibi kurallarla ifade edilmesi ise son derece kolaydır. Bu noktada kaç tane yaprak varsa o kadar kural oluşur.

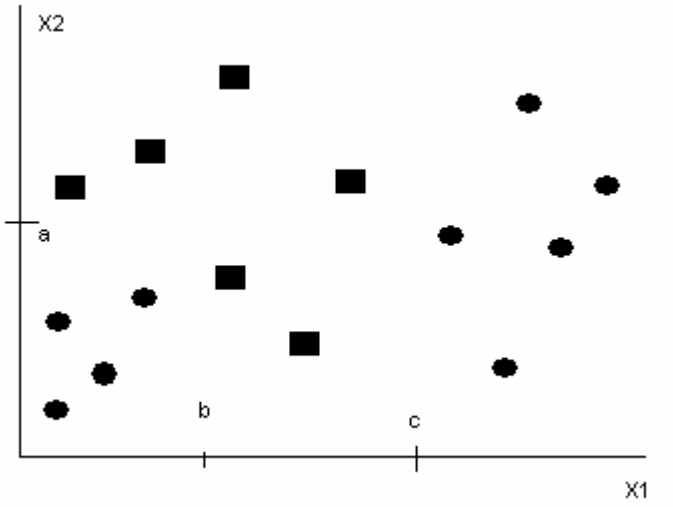
Verili deney sonuçlarından, gözlemlerden vb. yöntemlerle elde edilmiş bilgi yığınından böyle bir karar ağacı oluşturma ise en önemli noktalardan biridir. Bu algoritmalara geçmeden önce karar ağaçlarının türlerinden bahsedilebilir. Düğümlerin yapısı bakımından karar ağaçlarının üç tipi vardır:

- Tek değişkenli karar ağaçları
- Çok değişkenli karar ağaçları
- Melez karar ağaçları

Tek değişkenli karar ağaçlarında, düğümlerde sorulan sorular ilgili olayın bir tek değişkenine bakılarak yaratılır ve bu da aslında uzayı dikine bölmedir (Örnek: $X>5$). Çok değişkenli karar ağaçlarında, düğümlerdeki sorular bir çok değişkenin oluşturduğu uzayda ifade edilir (Örnek:$3X+5Y<7$). Melez ağaçlarda ise bu iki test biçimi de kullanılır. Ancak biri diğerinden iyi diyemeyiz; her iki yöntemin de birbirinden farklı avantajları var. Bu iki ağaç tipinde öğrenme metotları oldukça yaygındır.

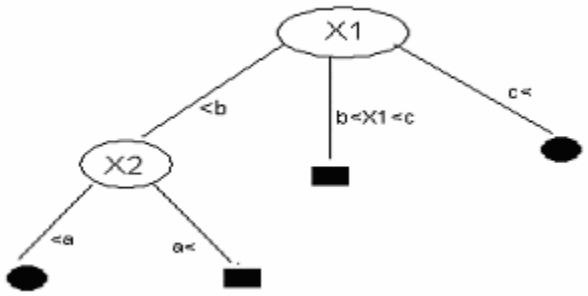
Karar ağacı oluşturma algoritmalarında olay kümesinin belirli özellikleri vardır. Bu özelliklerin aldığı değerler ya kategoriktir ya da sayısaldır (Örnek: hava(sıcak,ılık,soğuk): KATEGORİK ya da NOMİNAL, hava(72.7, 87.8): SAYISAL). Tüm özelliklerin aldığı değerler ve bu değerlere karşılık gelen sınıf bir vektör oluşturur.

Örnek olarak Şekil 2.1'de x1 ve x2 özelliklerine sahip bir olaylar kümesinin sonuçları vardır.



Şekil 2.1 - Koordinat sistemleriyle ifade edilen olay kümeleri

Bu sonuçlar kare ve daire olmak üzere ikili sınıftan ibarettir. Bu şekle göre Şekil2.2'deki ağaç yaratılır.



Şekil 2.2 - Şekil 2.1'deki veri kümesi ile yaratılan ağaç

Bu karar ağacı tek değişkenli bir karar ağacıdır. Burada olayların tek bir özelliği göz önüne alınarak testler yapılmıştır. Eğer düğümlerdeki sorular X1 ve X2'i aynı anda içerseydi çok değişkenli karar ağaçları yapısı oluşurdu.

Bu ağaçta aslında gizlenmiş dört adet kural vardır:

- X1<b ve X2<a → daire
- X1<b ve X2>a → kare
- b<X1<c → kare
- c<X1 → daire

Karar ağacı oluşturma algoritmaları çalıştıkları uzayın sınıf sayısına göre ise iki tipe ayrılır; birinci tip ikili sınıflandırıcı, ikinci tip ise çoklu sınıflandırıcı olarak isimlendirilir. Yukarıda verilen örnek ikili sınıflandırıcıları temsil eder. Bu iki tipe göre çeşitli ağaç oluşturma algoritmaları geliştirilmiştir. İkili sınıflandırıcıların en önemlisi ID3'dür [3]. Çoklu sınıflandırıcıların en önemli algoritması ise CART 'dir [1].

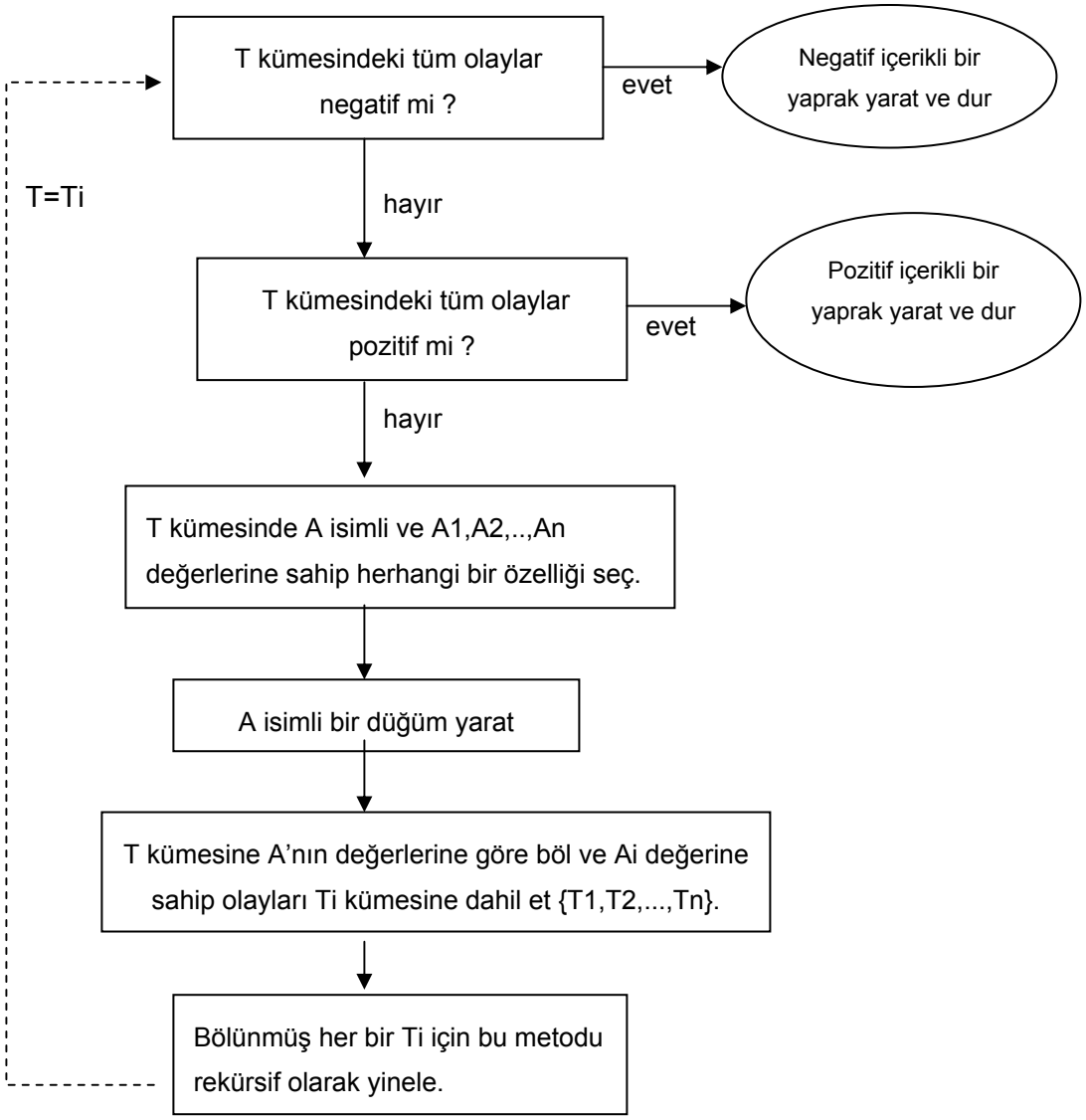
2.1 Karar Ağacı Oluşturma

Ağacın oluşturulmasına yönelik olarak çeşitli ağaç oluşturma metotları vardır. Ağacı oluşturmadaki en önemli kriter belli özelliklere göre toplanmış, güvenilir ve yeterli sayıda olay örneklerinin varlığıdır. Bu iki faktör ağaç oluşturmanın temelini oluşturur. Ağaç oluşturmadaki en önemli adım ise böl ve elde et aşamasıdır.

2.1.1 Böl ve Elde Et (Divide and Conquer)

Bu metot Hunt'ın uyguladığı bir metottur. Bu metotta örnek uzay T ve sınıflar pozitif ve negatif olsun. Bu durumda bir KA oluşturma Şekil2.3'deki gibi olacaktır.

Bu algoritma en temel ağaç yaratma algoritmasıdır. Bu algoritmanın enine boyuna incelenmesi ve geliştirilmesine yönelik olarak çeşitli çalışmalar vardır. Bunlardan en önemlileri tek değişkenli karar ağaçları için Quinlan'ın 1983'te geliştirdiği ID3 algoritması ve ardından yine Quinlan'ın geliştirdiği C4.5 algoritmasıdır. Çok değişkenli karar ağaçları için ise Breiman'ın geliştirdiği CART algoritması vardır.



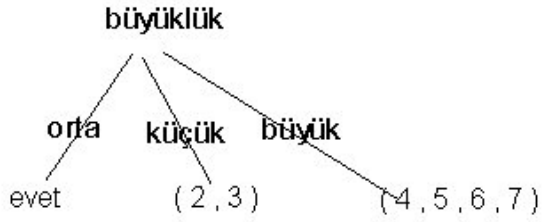
Şekil 2.3 - Hunt'ın ağaç oluşturma metodu

Örnek olarak bir maddenin bizim için uygun olup olmadığına bakan bir çalışma ele alınsın. Bu maddenin büyüklük, renk ve şekil gibi özellikleri olsun ve 7 adet örnek olay olsun. Bu örnekler evet-hayır olarak ikili sınıflandırılmış olarak Tablo2.1 'de gösterilmiştir.

Tablo 2.1- Örnek bir olay kümesi

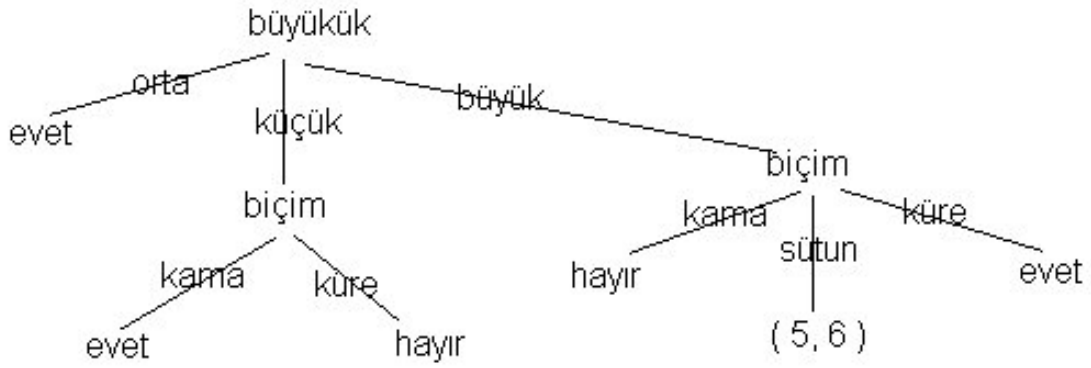
	Büyüklük	**Renk**	**Biçim**	**Sonuç**
1	Orta	Mavi	Tuğla	Evet
2	Küçük	Kırmızı	Kama	Hayır
3	Küçük	Kırmızı	Küre	Evet
4	Geniş	Kırmızı	Kama	Hayır
5	Geniş	Yeşil	Sütun	Evet
6	Geniş	Kırmızı	Sütun	Hayır
7	Geniş	Yeşil	küre	Evet

1 den 7'e kadar sıralanmış örnekler rasgele seçilen özelliği ile alt kümelere bölünsün. Şekil 2.4'de de gösterildiği gibi büyüklüğün muhtemel üç çeşit değeri olur ve üç tane dal oluşur.



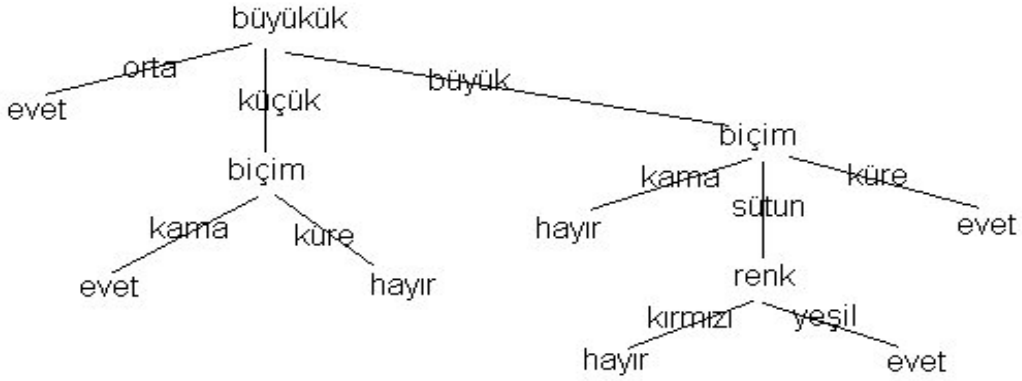
Şekil 2.4 - Tablo 2.1'in büyüklük sınıfına göre bölünmesi

Bu noktada büyüklük=küçük dalına ve büyüklük=büyük dalına yönelik olarak aynı işlem gerçekleştirilsin. Bölme işlemi yine rasgele seçilen biçim özelliğine göre yapılırsa Şekil2.5 'teki ağaç oluşur.



Şekil 2.5 - Şekil 2.4'deki ağacın bölünmüş kümeleri biçim özelliğine göre tekrar bölünmesi sonucu oluşan ağaç

Ağaçta sınıflandırılmamış iki olay kaldı. Bu yüzden bölümleme işlemine devam ediliyor. Bu aşamada yine rasgele olarak renk özelliğini seçilirse Şekil 2.6 'teki ağaç oluşacaktır.



Şekil 2.6 - Şekil2.5'teki ağacın sınıflandırılmamış olaylarının renk özelliği ile sınıflandırılması.

Bu durumda tüm olaylar sınıflandırıldı. Her bir aşamada sadece bir tek olay kaldığı için sonlandırıldı. Ancak, sadece bir tip sınıfa ait olaylar kalsaydı, bu durumda da sonlandırma işlemi gerçekleştirilecekti.

2.1.2 ID3 Algoritması

Yukarıda akış diyagramı gösterilen HUNT'ın meşhur algoritmasındaki en önemli eksiklik özelliklerin rasgele seçilmesidir. Oysa ki bu seçim sırasında bilgi kazancı en yüksek olan özellik dikkate alınırsa oluşturulan ağaç o kadar sade ve anlaşılır olacaktır. Buna

yönelik olarak Quinlan entropy kurallarını içeren bilgi teorisini kullanmıştır. Shannon ve Weaver'ın Bilgi Teorisinde temel olarak kaynak, mesaj ve alıcı vardır. Bu sistemde bilgi, mesaja bakılarak değil de, alıcıya bakılarak elde edilir. Alıcı mümkün olan mesaj uzayı bilgisine ve bu mesajların olasılıklarına sahiptir. Ağaçlardaki bazı düğümler ve bu düğümlerdeki kararlar anlamsız ve gereksiz olabilmektedir. Ancak bu tip düğümler de negatif–pozitif olay balanslarına sahiplerdir. İşte bu şekilde sınıflandırma yapılabilmektedir [3].

Örnek olarak X düğümünde 5 pozitif ve 3 negatif olay var. Bu noktada yapılacak bir sınıflandırmanın pozitif olasılığı 5/8'dir, negatif olasılığı 3/8'dir. İşte bu olasılıksal sınıflandırmayı türetme yeteneğinin anlamı şudur: Doğru olarak sınıflandırılmış bir örneğin söylediği mesajın bilgi içeriği artık hesaplanabilir.

Öyle ki bir tablonun sonuçları mesaj olsun ve mesajlar iki değere sahip olsunlar. Bu değerlerle birlikte p bilgisi pozitif olasılığını, q bilgisi negatif olasılığını gösterir. Bu iki değerin toplamı zaten 1 (p+q) olmak zorundadır. Doğru sınıflandırma veren bir mesajın bilgi içeriği

$$I(p,n) = -p \log_2 p - q \log_2 q \qquad (2.1)$$

şeklinde hesaplanır.

Bu formül genel bilgi içerik formülünün özel bir durumudur. Çünkü özel olarak iki olasılık mevcuttur: pozitif ve negatif.

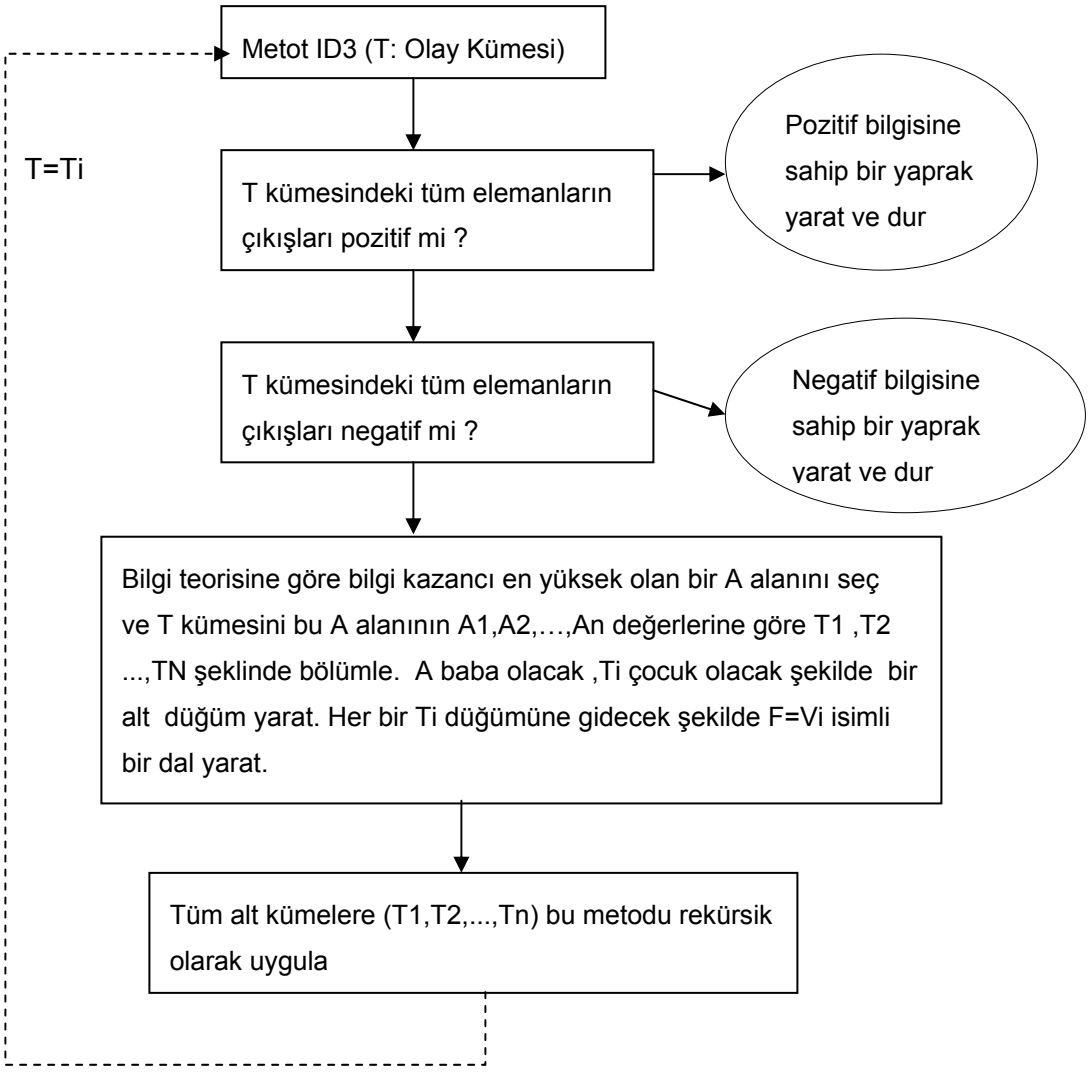
$\{A_1,A_2,...,A_n\}$ değerlerine sahip A özelliği ağacın bölünmesi için kullandığında,T kümesi $\{T_1,T_2,...,T_n\}$ şeklinde bölünecektir. Bu bölümleme de T kümesindeki A özelliğinin A_i olduğu bölgelere T_i densin. Bu kümedeki pozitif olayların sayısını p_i temsil etsin, negatif olayların sayısını n_i temsil etsin. Bu durumda T_i alt ağacı için beklenen bilgi gereksinimi ise $I(p_i,n_i)$ olur. T ağacı için beklenen bilgi gereksinimi tüm T_i ağaçlarının beklenen bilgi gereksinmelerinin ağırlıklı ortalamalarının toplamı olur ve.

$$E(A) = \sum_{i-1}^{n} \frac{pi + ni}{p + n} I(pi, ni) \qquad (2.2)$$

şeklinde hesaplanır. Dolayısı ile A özelliği üzerinden sağlanan bilgi kazancı

$$\text{Bilgi kazancı}(A) = I(p,n) - E(A) \qquad (2.3)$$

şeklinde ifade edilir. Şekil 2.7 - ID3 algoritmasının aşamalarını ifade etmektedir.



Şekil 2.7- ID3 Algoritması akış diyagramı

Bilgi gereksinimi ve bilgi kazancı ID3 algoritmaları için iki önemli kavramdır. Belirleyici bir sınıflandırma için bilgi ihtiyacı aslında doğru sınıflandırmayı sağlayan mesajın bilgi içeriğinden başka bir şey değildir. Buna yönelik olarak yaratmak istenilen karar ağaçlarının amacı doğru soruları sormasıdır. Ve sonunda öyle bir noktaya ulaşılmalı ki, bu noktanın karar için bilgi gereksinimi 0 olsun. İşte bu noktada ID3 algoritmasının yaptığı şey, ağacı doğru kurmaktır. Kurulu karar ağacının her seviyesinde geriye kalan bilgi gereksinimi (remaining information required) minimize edilir.

Bu bilgiler ışığında Tablo2.1'deki örnek ele alınsın. Bu olayların hepsi birden **evet** ya da hepsi birden **hayır** olamadıklarından bilgi kazancı en yüksek olan özellikten

başlayarak bölümleme işlemi gerçekleştirilir. Örnek uzayda 4 adet pozitif olay olduğunda bir olayın pozitif gelme olasılığı 4/7=0.57'dir. Negatif gelme olasılığı 3/7=0.43'dür. Bundan dolayı doğru bir sınıflandırma için gerekecek bilgi kazancı

$-(0.57 \times \log_2 0.57) - (0.43 \times \log_2 0.43) = 0.99$

olur.

Şimdi her bir özellik için bilgi gereksinmeleri hesaplansın. Büyüklük özelliği için küçük, orta ve büyük olmak üzere üç tip değer vardır. **Büyük** değeri kümenin {4,5,6,7} elemanlarını kapsamaktadır. Bu küme içersinde 2 **evet** ve 2 **hayır** sınıfı bulunduğundan ve p,n = 2/4=0.5 olduğundan gereken bilgi kazancı

$-(0.5 \times \log_2 0.5) - (0.5 \times \log_2 0.5) = 1$

olur.

Aynı işlem **küçük** değeri {2,3} elemanlarını içermektedir. Bu kümede 1 **evet** ve 1 **hayır** sınıfı vardır. Bu durumda gereken bilgi kazancı

$-(0.5 \times \log_2 0.5) - (0.5 \times \log_2 0.5) = 1$

olur. **Orta** değeri için hesap edildiğinde ilgili kümede sınıflardan sadece bir tanesi olduğundan sonuç 0 çıkar.

Şu anda büyüklük özelliği için beklenen bilgi gereksinimini hesap edilebilir. Bütün bilgi gereksinim sonuçları ilgili özellik değerlerinin orantısıyla çarpılarak toplanır ve

$(1 \times 4/7) + (1 \times 2/7) + (0 \times 1/7) = 0.86$

olur. Büyüklük özelliği için beklenen bilgi kazancı mevcut bilgi ihtiyacından beklenen bilgi ihtiyacı çıkarılarak hesaplanır;

$0.99 - 0.86 = 0.13$

Aynı işlemi renk ve biçim için yapılarak işlem tamamlanır. Bilgi kazancı **renk** için 0.52 ve **biçim** için 0.7 bulunur. Bu koşullar altında **biçim** özelliği en yüksek bilgi kazancına sahip özellik olur. Buna göre ağaç tekrar oluşturulduğunda Şekil 2.8'deki gibi daha sade bir şekil alır.



Şekil 2.8 - Tablo2.1'in ID3 ile oluşturulmuş KA

2.2 C4.5: Karar Ağacı Eğitme Algoritması

ID3 algoritmasında bazı eksiklikler ve sorunlar vardır. Bunlar aşağıda anlatılmaktadır. Bu sorunlar yine Quinlan'ın geliştirdiği C4.5 algoritmasıyla giderildi. C4.5 Algoritması ID3 algoritmasının bütün özelliklerini kendine miras alarak oluşturulmuş bir algoritmadır. Yukarıda bahsedilen tüm içeriğin üzerine yeni kavramlar eklenmiştir. Bölünme-Dağılma Bilgisi (Split-Info), özelliklerin kayıp değerleriyle baş edilmesi, sayısal özellik değerlerinin hesaplara katılması bu başlıklardan en önemlileridir. Bu başlıklardan biri olan Bölünme-dallanma bilgisi ile başlanabilir.

2.2.1 Bölünme-Dallanma bilgisi (Split Information)

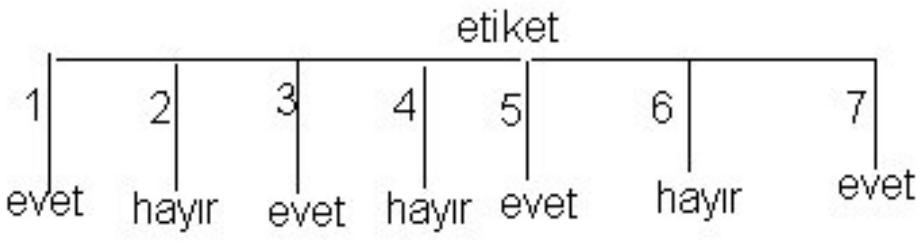
Bir kategorik özelliğin olası değer çeşitliliği ne kadar yüksek olursa o özelliğin bilgi kazancı gereksiz bir şekilde yüksek çıkar ve bu durum ağacın doğruluğunu kötü bir şekilde etkiler. Bu tip özellikler işe yaramadıkları gibi bilgi kazancı yüksek özelliklerin de önüne geçip veride gizlenmiş kuralların çıkarılmasına engel teşkil ederler. Yukarıda ele alınan örnekte değer çeşitliliği en fazla dörttü. Şimdi bu veriye çeşitliliği çok yüksek bir özellik eklenirse nasıl bir sonuç çıkacaktır? Öyle ki Tablo2.1'deki 1'den 7'e kadar verilmiş olan etiket numaraları bir özelliğe karşılık gelsin. Bu aşamada bu özelliğin bilgi kazancı hesap edilsin; 1:evet, 2:hayır, 3:evet,4:hayır,5:evet,6:hayır ve 7:evet şeklinde her bir özellik değeri için bir tane sonuç elde edilecektir. Bu durumda 1 değeri için gereken bilgi kazancı $-(1 \times \log_2 1/1) - (0 \times \log_2 0/1) = 0$ çıkacak ve aslında tüm değerleri için gereken bilgi kazancı 0 çıkacaktır. Etiket özelliği için ortaya çıkan bilgi kazancı ise

0 x (1/7) +0 x (1/7) +0 x (1/7) +0 x (1/7) +0 x (1/7) +0 x (1/7) +0 x (1/7) =0

olacaktır. Genel bilgi teorisi için bu sonucu mevcut bilgi gereksiniminden çıkarmak gerekecek. Bu durumda etiket özelliği için bilgi kazancı

0.99 – 0 = 0.99

olacaktır. Bu sonuç diğer sonuçlar arasındaki en yüksek sonuçtur ve buna göre bir ağaç oluşturulduğunda Şekil 2.9'daki ağaç yapısı oluşur.



Şekil 2.9 - Tablo2.1'in etiket özelliğine göre bölümlenmesi

Ancak bu ağaçtaki etiket numarası ile yapılan bir sınıflandırmanın hiçbir yararı yoktur. Yani bu kurallar çok gereksiz kurallardır. Bilgi kazancının yüksek çıkmasının sebebi özellik çeşitliliğinin yüksek olmasıdır. Bu gereksiz kural oluşması şu örnekle daha iyi açıklanabilir. İnsanlarda şeker hastalığının olup olmadığına karar veren bir karar ağacı oluşturulacaktır. Bu konuda şeker hastalarından bazı verilen toplanır. Ancak bu verilerin içinde hastaların etiket numaraları da bulunur. Bu durumda yukarıdaki örneğe paralel olarak hasta numarası en yüksek bilgi kazancına sahip olacaktır. Şeker hastalığının olup olmadığına etiket numarası ile karar verilmiş olunacaktır. İşte bu tip gereksiz bilgilerle başa çıkabilmek için Quinlan bölünme bilgisi kavramı ile algoritmasını güncelledi. Bu algoritma değer çeşitliliği fazla olan özelliklerin bilgi kazancını azaltarak algoritmanın gereksiz bazı çıkarımlar yapmasını engellemektedir.

Bu noktada bölünme bilgisi denilen yeni bir kavram ekleniyor bu algoritmaya. A bir özellik, A_i bu özelliğin değerleri, T_i A_i özelliğinin bu veride kaç kez tekrarlandığı ve T ise ele alınan olay sayısını temsil etsin. Bu durumda bölünme bilgisi

$$- \sum_{i=1}^{n} \frac{Ti}{T} \times \log_2 \left(\frac{|Ti|}{|T|} \right) \quad (2.4)$$

olarak ifade edilir. Bu bölünme bilgisi tüm özelliklerin bilgi kazanç formülüne bölen olarak eklenir ve bu kazanç oranı olarak ifade edilir. Bu durumda A özelliğinin kazanç oranı

kazanç_oranı=bilgi_kazancı(A) / bölünme_bilgisi(A) (2.5)

şeklinde hesap edilir. Yenilenen bu formüle göre Tablo2.1'deki veri test edilirse Tablo2.2'deki sonuç ortaya çıkar.

Tablo 2.2 - Tablo2.1'deki veri yığının kazanç oranlarının hesaplanması

Özellik ismi	Bilgi kazancı	Bölünme bilgisi	Kazanç oranı
Etiket	0.99	2.80	0.35
Büyüklük	0.13	1.38	0.09
Renk	0.52	1.38	0.37
Biçim	0.7	2.2	0.31

Bu tabloya bakıldığında bilgi kazancı en yüksek olan özellik 0.99 bilgi kazancı ile etiket özelliğidir. Ancak bilgi oranı dikkate alındığında en yüksek özellik 0.37 ağırlıkla renk özelliği olacaktır. Ancak bu tablo şunu da göstermektedir; etiket gibi yararsız özelliklerin bölünme bilgisi gibi bir formülle bilgi kazançları biraz törpülense bile yine de bu özellikler kayda değer bir bilgi oranına sahip oluyorlar. Bu sistemde de renk özelliğinden sonra ikinci sıraya yerleşmiş durumdadır. Dolayısıyla bu tip özelliklerin henüz üstesinden gelinememiştir.

2.2.2 Sayısal Özellikler

Veri kümesinde iki tip veri vardır; Nominal (kategorik) ve sayısal. Nominal daha önceki bölümlerde kullanılan veri tipleridir. Örnek olarak renk özelliği mavi, kırmızı ve sarı ile, büyüklük özelliği büyük, orta ve küçük ile ifade edilir. ID3 algoritması daha önce sadece nominal değerlere sahip veri tipleri ile işlemler yapabilirdi. Ancak bu

algoritmanın varisi olan C4.5 algoritması ise bu tip veriler için de bir yöntem geliştirmiştir.

İlk bakışta sayısal özelliklerle uğraşmak ve onların bilgi kazancını hesaplamak oldukça zor gelebilir. Ancak bu iş o kadar da zor değildir. Yapılması gereken iş sadece bu özelliğin sayısal değerleri arasında uygun eşik değerini bulmaktır. Bu eşik değeri bulunduktan sonra ikili bir bölünme ile veri kümesi bölünebilir; Bu eşik değerinden büyük veriler ile bu eşik değerinden küçük veriler. Bu anlamda algoritma çok sade bir şekilde açıklanabilir. Öncelikle tüm sayısal değerler küçükten büyüğe sıralanır.Bu sıra $\{v_1,v_2,...,v_m\}$ ile ifade edilsin. Bu durumda seçilen eşik değeri v_i ve v_{i+1} arasında olursa $\{v_1, v_2,...,v_i\}$ ile $\{v_{i+1}, v_{i+2},...,v_m\}$ gibi iki grup ortaya çıkar. Buradan da görülüyor ki m-1 adet eşik değeri seçilebilir. Bu seçim işlemi için olası bütün eşit değerleri

$$\frac{v_i + v_{i+1}}{2} \qquad (2.6)$$

formülü ile hesaplanır. Bu yapıyla sanki söz konusu özellik büyük-küçük değerleri olan nominal bir özelliktir. Bu anlayışla nominal değerlere uygulanan bilgi oranı formülü tüm eşik değerleri için uygulanır ve bilgi kazanımı en iyi olan eşik değeri söz konusu özelliğin eşiği olarak kabullenilir. Eğer en iyi eşik değeri ***e*** ise ve söz konusu özelliğin sayısal değerleri $\{v_1, v_2,...,v_n\}$ kümesi ile ifade ediliyorsa, bu kümedeki $v_i < e$ koşulunu sağlayan elemanlar küçük kategorisine ve $v_i >$ **e** koşulunu sağlayan elamanlar büyük kategorisine dahil edilir.

Böyle bir bölümleme işlemi bir çok veri üzerinde denendiğinde olumlu sonuçlar vermiştir. Ancak eksik olan yanı sadece ikili bir bölme işlemi gerçekleştirmesidir. Oysa ki bu tip bir bölümleme üçlü ya da daha fazla olursa veri yığını içersine gizlenmiş olan kuralları bulma olasılığı daha çok artar. Bu anlamda böyle bir çalışma bu tez içersinde denenmiştir ve sonuçlarıyla birlikte sonuç bölümünde açıklanmıştır.

2.2.3 Kayıp Veriler

Bir önceki bölümde gizli bir kabullenme vardır; söz konusu ID3 algoritması eksik olmayan bir veri yığınını dikkate alır. Fakat veride bazı bilgilerin bulunmaması tüm algoritmanın çalışmasını engeller ve yanlış çıkarımlara yol açar. Eğer gerçek uygulamalar hedef alınırsa, böyle bir sorunla her zaman karşılanacaktır ve bu durum her zaman için kaçınılmazdır. Veri, çeşitli sebeplerden dolayı eksik olabilir.

Verinin bazı özeliklerinin toplanması özgün koşullardan dolayı zor olabilir, veri formatına uymayan bir bilgi alınmış olabilir, veri tabanına aktarırken eksik girilmiş olabilir vb. bir çok sebepten dolayı veri bütünlüklü olmayabilir. Bu aşamada veri yığınını tamamlamaktan ziyade algoritmaya bazı yetenekli metotlar eklemek gerekir. Bu aşamada algoritmanın önünde üç problem durmaktadır.

a) Bilgi kazancı ve bilgi oranı değerleri olmayan özellikler için nasıl hesaplanır?

b) KA'nı oluştururken alt ağaç yaratma işlemi sırasında özellik değeri olmayan satırlar hangi alt ağaca eklenecektir?

c) KA oluşturduktan sonra, kayıp değerleri olan yeni bir olay sınıflandırılırken nasıl bir test yapılacaktır?

Bu üç soru çerçevesinde kayıp veriler sorunu enine boyuna tartışılabilir. Kayıp verilere genel yaklaşım bu verilerin bir şekilde tamamlanması yönündedir. Öncelikle verideki kayıplar tespit edilir. Ardından ilgili özelliğin kayıp değeri yerine o özelliğin en sık tekrarlanan değeri konur. Örnek olarak 100 satırlık bir veri yığınında renk isimli bir özellik olsun. Bu renk alanı 25 kırmızı, 40 mavi, 30 beyaz değerine sahip olsun ve bu alanın 5 değeri ise kayıp olsun. Eksik olan bu değerlerin kırmızı olma olasılığı 25/95 = %26, mavi olma olasılığı 40/95 = %42 ve beyaz olma olasılığı 30/95 = %31'dir. Bu sonuçlar dikkate alındığında kayıp bütün değerler olasılığı en yüksek olan mavi değeri ile ifade edilir. Ancak burada %60 bir hata oranı vardır ki bu oran çok yüksektir. Olasılık yöntemiyle tamamlanan bu tip veri tamamlama yöntemleri kaybı az olan veriler için çok sorun çıkarmamaktadır. Ancak çok fazla kaybı olan verilerde bu hata oranı katlanarak artar ve oluşan KA'nın güvenilirliği azalır. Bu sebepten C4.5 algoritması kayıp verilerle başa çıkmak üzere çeşitli yaklaşımlar sunmuştur.

2.2.3.1 Bilgi Kazancı

Bir önceki bölümde bilgi kazancı hesap etme formülleri eksik olmayan verilere göre yapılmıştır. Şimdi ise bu eksik olan verilerde dikkate alınarak formüller güncellenecektir. T çalışılan veri kümesi olsun ve genel bilgi kazancı bilgi(T) olsun. X bu kümenin herhangi bir özelliği olsun ve X özelliğinin bilgi kazancı ise $bilgi_X(T)$ olsun. Bilgi(T) eskisi gibi hesap edilir. Ancak $bilgi_X(T)$'yi hesap edilirken olmayan veriler bu kümeden çıkarılır. Olay sayısı n ile ifade edilirse ve bilinmeyen veriler b ile ifade edilirse X

özelliğin n-b adet eksik olmayan verileriyle sanki hiçbir veri eksik değilmiş gibi klasik formül uygulanır. Ardından eksik olmayan değerlerin toplam değerlere oranı

$$F= (n-b) / n \quad (2.7)$$

formülü ile hesaplanır. Bu durumda

$$F \bullet (\text{bilgi}(T) - \text{bilgi}_X(T)) \quad (2.8)$$

formülü ile bilgi kazancı hesaplanmış olur. Bu noktada bölünme bilgisini de hesaplara eklemek gerekecektir. Bölünme bilgisi hesabı sırasında olmayan verilere sanki X özelliğinin bir değeri gibi muamele edilir. Yani k adet çeşitliliğe sahip X özelliğinin çeşitliliği k+1 olur ve buna göre bölünme bilgisi standart formül ile hesap edilir.

2.2.3.2 Alt Ağaçlara Bölme

Ağaç bölme işlemi önceki işlemlerin aynısıdır. T veri yığınına bir X testi uygulandığında O_1, O_2,..., O_n çıktıları elde edilir. Ancak bilinmeyen verilere bu test uygulandığında bu çıktılar sonuç vermez. T veri yığınından bilinen bir O_i çıkışı T_i alt kümesini oluşturur ve O_i çıkışını yaratan olayların T_i kümesine ait olma olasılığı 1 ve diğer bütün alt kümelere ait olma olasılığı 0'dır. Örneğin renk özelliği test ediliyorsa ve renk özelliği %30 oranla kırmızı değerine sahipse $O_{renk=kırmızı}$ çıkışı ile oluşan $T_{renk=kırmızı}$ kümesi, T kümesinin %30'na sahip olur. T kümesindeki renk özelliği kırmızı olan satırların hepsinin $T_{renk=kırmızı}$ kümesine ait olma olasılığı 1 ve bunun dışındaki tüm T_i kümelerine ait olma olasılığı 0'dır.

Kayıp verilerle nasıl bir yöntem izlenecektir? Bu durumda kayıp verinin bulunduğu satır her bir T_i alt ağacında bulunma olasılığı hesaplanır. Buna göre bu satır her alt ağaca gönderilir. Ancak bu noktada o satırın o alt ağaçta olma olasılığı da eklenen satıra işlenir. Böylelikle bu satırın olma olasılığı her bir alt kümede 1'den küçük olacak ve her bir alt ağaç için oluşan olasılıklar toplamı 1'e eşit olacak. Ancak bir satırda birden fazla kayıp değer varsa bu olasılıklar çarpılacaktır. Eğer bir satırın ağırlığını w ile gösterirsek- ki başlangıç noktasında tüm w değerleri 1'dir- bir sonraki testten oluşacak ağırlık

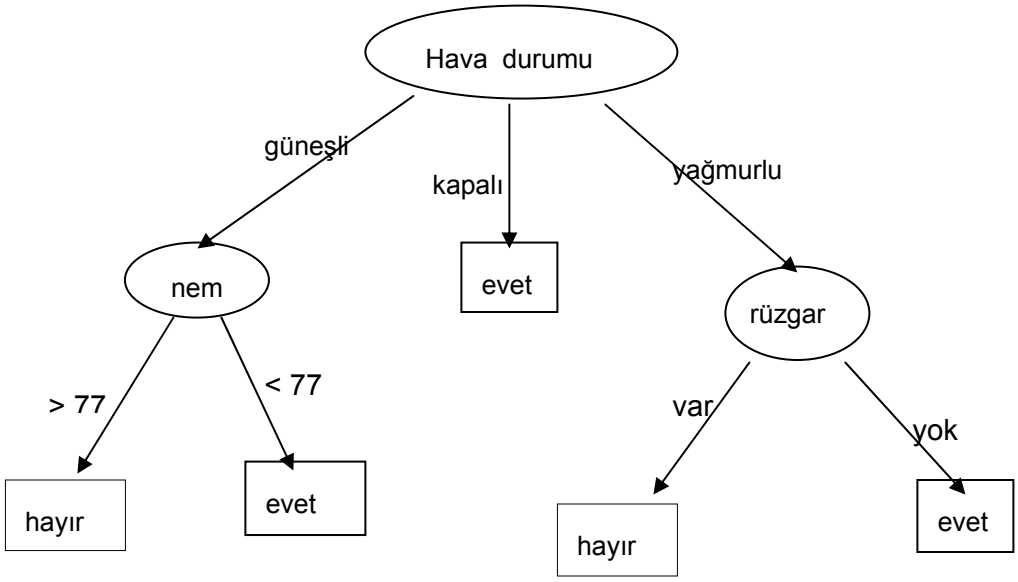
$$w_i = w \times O_i \text{ çıkışı olasılığı} \quad (2.9)$$

şeklinde bir formülle hesaplanacaktır. Tablo2.3'teki örnek bu durumu canlandıracaktır. Bu tablo koşulların tenis oynanmasına müsait olup olmadığını inceliyor.

Tablo 2.3 - Hava durumunun tenis oynamaya müsait olup olmadığına ilişkin veri kümesi

	Hava durumu	**Sıcaklık (°F)**	**Nem(%)**	**Rüzgar**	**Sonuç**
1	Güneşli	75	70	Var	Evet
2	Güneşli	80	90	Var	Hayır
3	Güneşli	85	85	Yok	Hayır
4	Güneşli	72	95	Yok	Hayır
5	Güneşli	69	70	Yok	Evet
6	Kapalı	72	90	Var	Evet
7	Kapalı	83	78	Yok	Evet
8	Kapalı	64	65	Var	Evet
9	Kapalı	81	75	Yok	Evet
10	Yağmurlu	71	80	Var	Hayır
11	Yağmurlu	65	70	Var	Hayır
12	Yağmurlu	75	80	Yok	Evet
13	Yağmurlu	68	80	Yok	Evet
14	Yağmurlu	70	96	Yok	Evet

C4.5 algoritmasının böyle bir veri yığınından çıkardığı karar ağacı Şekil2.10'daki gibidir. Karar ağacının ilk sorusu hava durumu oldu ve 5 yapraklı ya da 5 kurallı bir karar ağacı yaratıldı.



Şekil 2.10 - Tablo2.3'teki veri kümesinin C4.5 ile yaratılması.

Bu verideki özelliklerin bilgi kazançları tablodaki gibidir. Bu veri kümesindeki kazanç oranları Tablo2.4 'te verilmiştir

Tablo 2.4 -Tablo2.3'teki verilerin bilgi kazanç oranları

Özellik	**Bilgi kazanç oranı**
Hava durumu	0.156
Sıcaklık	0.01
Nem oranı	0.151
Rüzgar	0.04

Eğer ki veri yığınımızda bazı özelliklerin değerleri olmasaydı nasıl bir karar ağacı ortaya çıkacaktı.

Tablo 2.5 - Tablo2.3'teki 6. satır

6	Kapalı	72	90	Var	Evet

Tablo2.5'de ifade edilen 6. satırın hava durumu özelliği kapalı yerine bilinmiyor olsun ve

Tablo 2.6 - 6.satırın hava durumu özelliğinin kayıp olması

6	?	72	90	Var	Evet

Tablo 2.6'da ifade edildiği gibi "?" ile ifade edilsin. 14 satırlık bu veri tabanının artık başlangıç için değerleri bilinen ve sağlam 13 satırı ele alınacaktır. Bu durumda

Tablo 2.7 - Tablo2.5'in 6. satırındaki hava durumu bilgisinin kayıp olduğunda ortaya çıkan istatistik görüntüsü

Hava durumu	Evet	Hayır	Total
Güneşli	2	3	5
Kapalı	3	0	3
Yağmurlu	3	2	5
Toplam	8	5	13

Tablo2.7'deki gibi bir tablo elde edilir. Hava durumu özelliğinin bilgi kazançları öncellikle sanki 6. satır hiç yokmuş gibi hesap edilir.

$$\text{Bilgi}(T) = -\,8/13 \times \log_2 (8/13) - 5/13 \times \log_2(5/13) = 0.961$$

$$\text{Bilgi}_X(T) = 5/13 \times (-2/5 \times \log_2 (2/5) - 3/5 \times \log_2 (3/5))$$

$$+\,3/13 \times (-2/5 \times \log_2 (2/5) - 0/3 \times \log_2 (0/3))$$

$$+5/13 \times (-3/5 \times \log_2 (3/5) - 2/5 \times \log_2 (2/5)) = 0.747$$

$$\text{Bilgi kazancı} = 13/14 \times (0.961 - 0.747) = 0.199$$

Bu sonuç daha önce hesap edilen 0.246 'dan küçüktür. Çünkü veri yığınındaki verilerin bazıları bilinmiyor. Bölünme bilgisinin de hesap edilmesi gerekecek. Bölünme bilgisini hesap ederken eskiye göre yeni bir kategori oluşmuştur. Bu kategori "?"dir, yani kayıp veri kategorisidir. Bu değer güneşli, yağmurlu gibi değerlerin yanında kayıp değer olarak yer alacak. Bu durumda

$-5/14 \times \log_2 (5/14)$ → güneşli değeri için

$-3/14 \times \log_2 (3/14)$ → kapalı değeri için

-5/14 x $\log_2$ (5/14) → yağmurlu değeri için

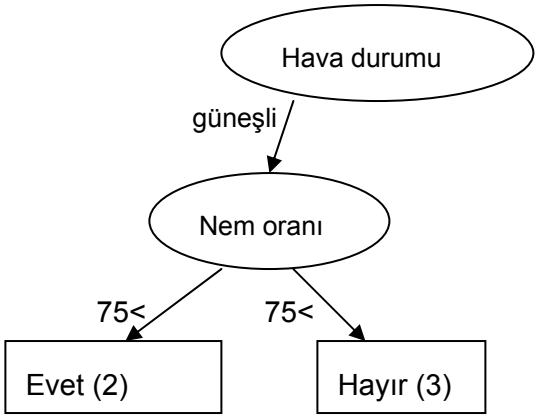
-1/14 x $\log_2$ (1/14) → ? değeri için (formüldeki 1 bilinmeyen sayısını temsil ediyor)

ile hesap edildiğinde daha önce 1.577 sonucunu veren bölünme bilgisi değeri 1.809 gibi bir değer alır. Bu sonucun daha yüksek çıkması çok yerindedir. Bu durumda bilgi oranı ise 0.156'dan 0.110'a düşer. 14 olay bu testler ile sınıflandırıldığı zaman bu olayların 13'ü hiç problemsiz ağaca yerleşirler. Ancak söz konusu 6.satır bu ağaçtaki yerini nasıl alacak ve hangi alt ağaca gönderilecek ? Bu aşamada bilinmeyen içerikli satır tüm alt ağaçlara ya da kümelere belli bir ağırlık bilgisiyle gönderilir. Çünkü tüm bu alt ağaçlara ait olma ihtimali vardır. Söz konusu satır hava durumu = güneşli, kapalı, yağmurlu 3 alt kümeye sırasıyla 5/13, 3/13, 5/13 ağırlık bilgileriyle gönderilir. Birinci alt ağaca gönderilme durumu canlandırılırsa Tablo2.8 elde edilir.

Tablo2.8 - Hava durumu=güneşli alt ağacının kapsadığı kümenin ağırlık bilgileriyle gösterilmesi

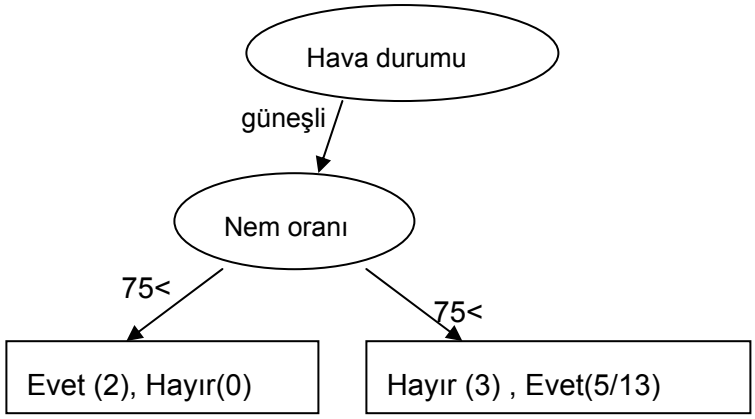
	Hava durumu	Sıcaklık (°F)	Nem(%)	Rüzgar	Sonuç	Ağırlık
1	Güneşli	75	70	Var	Evet	1
2	Güneşli	80	90	Var	Hayır	1
3	Güneşli	85	85	Yok	Hayır	1
4	Güneşli	72	95	Yok	Hayır	1
5	Güneşli	69	70	Yok	Evet	1
6	?	72	90	Var	Evet	5/13

Görüldüğü gibi 6. satır dışındaki bütün satırların ağırlıkları 1. Çünkü bu satırların bu alt ağaca ait olma olasılığı 1 olur. Bütün değerleri bilinen ilk ağaçta hava durumu = güneşli alt ağacı Şekil 2.11'deki gibidir.



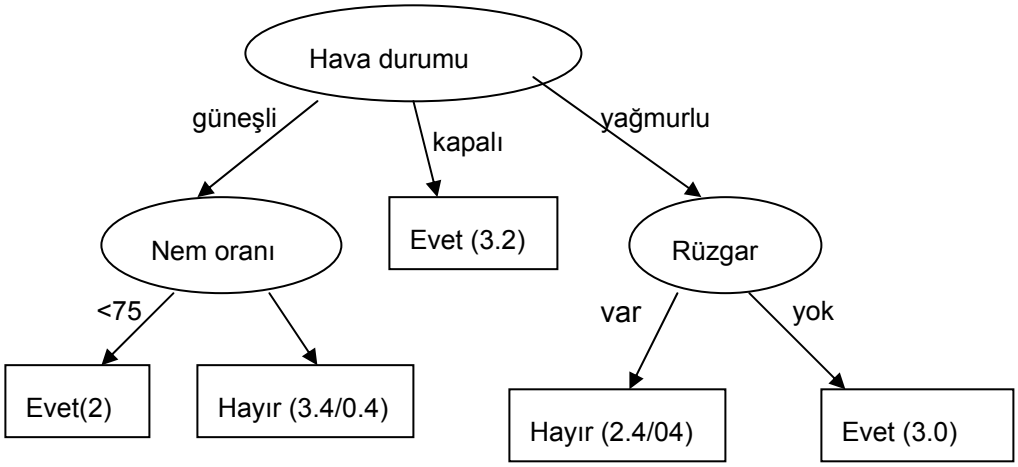
Şekil 2.11 - kayıp veri göz önüne alınmadan oluşturulan alt ağaç.

. Bu tablodaki parantez içindeki rakamlar o yaprağı oluşturan olay sayısını ifade ediyor. 6. satırının hava durumu "?" olduğunda bu alt ağaç şekil 2.12'deki



Şekil 2.12 - Kayıp veri göz önüne alındığında oluşan alt ağaç.

gibi ifade edilir. Bu alt ağaçta artık iki sınıf hakkında da bilgi bulunmaktadır. Bu sınıf bilgileri olasılık olarak ifade edilir. Ancak sonuç olarak olasılığı en yüksek sınıfla yaprak etiketlenir. Tüm alt ağaçlar bu şekilde oluşturulduğunda Şekil2.13'teki ağaç elde edilir.



Şekil 2.13 - Hata içeriği ile ağacın gösterilmesi

Bu ağaçtaki (N /E) ile ifade edilen yaprak bilgisinde N olay sayısını ve E ise hatalı hesaplanmış ve o yaprağın kabul ettiği sınıfın karşıt sınıfının olay sayısını vermektedir. Tabii ki bu noktada reel sonuçlar çıktığından sayı yerine miktar demek daha doğru olabilir.

2.2.3.3 Yeni bir olayı sınıflandırma

Yeni bir olayın alan değerlerinin bazıları bilinmiyorsa bu olayın sınıflandırılması başarılı olabilir ya da olmayabilir. Bu daha çok o olayın verisi kayıp özelliğinin ağacın üst düğümlerinde bulunup bulunmamasıyla ilgilidir. Öyle ki Tablo 2.9 'deki gibi bir satırda

Tablo 2.9 - kayıp verileri olan, sınıflandırılacak örnek bir olay.

Hava durumu	Sıcaklık (°F)	Nem(%)	Rüzgar	Sonuç
kapalı	?	?	?	

sıcaklık, nem, rüzgar alanları bilinmemesine karşın hava durumu bilgisi bilindiğinden hava durumu = kapalı → EVET kuralından bu olay EVET sınıfı alarak ağaçtan çıkmayı başarır. Ancak bütün durumlar böyle şanslı olamayabilirler. Buna örnek olarak

Tablo 2.10 - Sınıflandırılacak yeni bir olay.

Hava durumu	Sıcaklık (°F)	Nem(%)	Rüzgar	Sonuç
Güneşli	70	?	yok	

Tablo 2.10'daki olay verilebilir. Hava durumu bu yeni olayı birinci alt ağaca yönlendirir. Birinci alt ağaç bu olaya nem oranını sorduğu için bu durumun sınıflandırılma işlemi başarısız olacaktır. Çünkü nem oranı bilgisi yoktur. Bu durumda olasılık olarak bu olayın hangi sınıfa daha yatkın olduğunu hesaplamak gerekecektir.

- Eğer nem oranı 75% 'den küçükse bu olay Evet ile etiketlenecek.
- Eğer nem oranı 75% 'den büyükse bu olay 3/3.4 = 88% olasılıkla EVET ve 0.4/3.4 =12% olasılıkla HAYIR olacak.

Ağaç oluşturulduğu zaman bu iki çıkış sırasıyla 2 ve 3.4 olayı ifade etmekteydiler. Göreli ağırlıkları ise 2.0/5.4 ve 3.4/5.4 olacaktır. Bundan dolayı karşılıklı iki sınıfın bu olay bazındaki olasılıkları

EVET= 2.0/5.4 x 100% +3.4/5.4 x 12% =44 %

HAYIR = 3.4/5.4 x 88% =56%

şeklinde olacaktır. Bu noktadan sonra bu olay %44 bir hatayla HAYIR ile etiketlenebilir.

2.3 Ağacın Testi

Oluşturulan ağacın performansı o ağaçtaki hata oranı ölçülerek tespit edilir. Sınıflandırıcı yani söz konusu bu ağaçlar, her bir olay için bir sınıf tahmininde bulunurlar; bu tahmin doğruysa söz konusu ağaç başarılıdır; değilse başarısızdır. Hata oranı ise tüm veri kümesindeki hatalı sınıflandırılmış olayların tüm olaylara oranıdır. İşte bu sonuç o sınıflandırıcının performansına karşılık gelir.

Tabii ki bu noktada karar ağaçlarının ya da herhangi bir sınıflandırıcının başa çıkması gereken eski olaylardan ziyade yeni olaylardır. Yeni olaylar karşısında göstereceği performans o sınıflandırıcının doğruluğunu gösterir. Çünkü eski olayların hangi sınıftan

olduğu zaten bilinmektedir ve bunlar üzerinde yapılacak bir doğruluk oranı genelde yüksek çıkacaktır. Eğer ağaçta hiç bir budama işlemi ya da bir kestirim yapılmamışsa, genelde ağacın hata oranı 0%'a yakındır. Bu nedenle ağacın yaratılmasında kullanılan verilerle gerçekleştirilen hata oranı gelecekteki yeni verilere ilişkin hata oranına pek bir ışık tutmamaktadır. Dolayısıyla hata oranı tespiti yeni bir veri kümesi üzerinde yapılmalıdır. Ağacı yaratan verilerdeki hata oranı genelde o ağacı yaratırken kullanılan yer değiştirme işlemlerini ifade etmektedir. İşte bu hata oranı hiç güvenilir bir oran değildir. Önemli olan bu veriden bağımsız bazı veriler üzerinde bu testi uygulamaktır. Bu noktada söz konusu olan iki ayrı veri kümesi vardır.

- Eğitim Kümesi : Ağacı oluşturan veri kümesi
- Sağalama Kümesi : Ağacın güvenilirliğini-hata oranını tespit eden veri kümesi

Test veri kümesi tamamen bağımsız olmak zorundadır. Eğer tam bir genelleme yapılmak isteniyorsa bu nokta şüphesiz ki çok önemlidir. Örnek olarak İstanbul bankalarında hesabı olan müşterilerden oluşan bir karar ağacı, Ankara'da oluşan bir veri kümesini ile test edildiğinde sonuç olumlu çıkabilir. Ancak bu test Afyon, Çorum, Kars vb. iller ile yapıldığında sonuç aynı çıkmayabilir. Çünkü müşteri davranışı büyük şehirlerde küçük şehirlere göre farklı olacaktır.

Yani sonuç olarak bir karar ağacı yaratılmadan önce veri kümesi ikiye ayrılır: Eğitici küme ve sağlama kümesi. Bu noktadaki en önemli problem ikiye bölünecek kadar verinin bulunamamasıdır. Veri çok olduğu zaman bu işlem kolaylıkla yapılabilir. Ancak veri yetersizse, test için ayrılacak veri kümesini azaltmak gerekecektir. Buna yönelik olarak test kümesini de sonradan eğitici kümeye dahil edilmesi gibi bazı algoritmalar geliştirilmiştir. Fakat bu konu şu an bu tezin kapsamında değildir.

Test aşamasında çok kritik bir aşama vardır ki oda seçilen bu iki kümenin birbirine tamamen zıt olmasıdır. Bu şansızlığın ardından aslında doğru olan karar ağacına sanki tamamen yanlışmış muamelesi yapılabilir. Buna yönelik olarak çeşitli yöntemler geliştirilmiştir. Onlardan bir tanesi olan ikili-çapraz-sağlama **(cross-validation)** veri kümesini iki eşit parçaya böler; önce birini sonra diğerini eğitici küme olarak seçer ve bu işlemi aynı şekilde test kümesi için de yapar. Bu durumda ortaya çıkacak olan ortalama hata oranı daha sağlıklı sonuç verecektir. Ancak bazı çalışmalar bunun da yetersiz kaldığını göstermiştir. Bu amaçla bu ikili-sağlama işleminin yaptığı ikiye bölme yerine

ona bölme denenmiştir. Onlu-çapraz-sağlama işlemi önce tüm veri yığınını 10 eşit parçaya böler. Ardından her birini sırayla dışarı alır ve kalan 9 adet veri kümesinden bir eğitim seti oluşturarak karar ağacını oluştur. Ardından bu ağacı dışarıya aldığı veri kümesi ile test eder. Bu işlemi her biri için yapar ve bu işlemlerden çıkan 10 adet hata oranının ortalaması gerçek ortalama olarak kabul eder. Aslına bakılırsa 10 yerine 15 ya da 20 de seçilebilirdi. Bu anlamda özellikle 10 sayısının seçilmesinin çok büyük bir anlamı yok. Tek anlamı ne çok büyük ne de çok küçük.

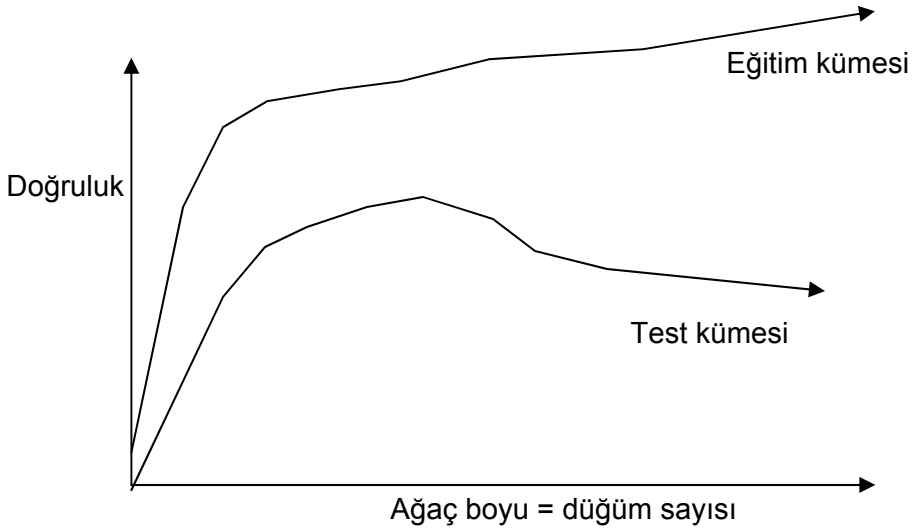
2.4 Ağaç Budama Metotları

Ağaç budama karar ağaçlarının en önemli konularından biridir. Veri yığınından oluşan ağaç çok büyük olabilir ve hiç bir anlam ifade etmeyebilir. Derinliği 20 olan bir karar ağacı 20 tane **if** (eğer) kuralından sonra bir sonuca ulaşıyor demektir. Ya da bir olaya 20 tane soru sorduktan sonra o olay hakkında konuşabiliyor demektir. Bu durum istenilen bir durum değildir. Basit bir veri yığınından ağacın çok büyük çıkmasına şişme (**overfitting)** denir. Overfitting her veri yığını için geçerlidir denemez ama olasıdır. Böyle bir sonuç çıkmasının iki ana nedeni olabilir. Birincisi veri yığını üzerinde gürültü vardır. Gürültü veri yığınının gereksiz kurallar türetmesine yol açar. Bu durumda ağaçtan çıkarılan kurallar aslında gürültüye aittir. İkinci olasılık ise seçilen veri kümesinin o olayı temsil edebilme yeteneğinin olmamasıdır. Buna yönelik olarak Quinlan'ın yaptığı bir çalışma, açıklayıcı olması bakımından verilebilir. Yapılan çalışmada 1 ve 0'lardan oluşan 500 satırlık iki veri kümesi rasgele yaratılır. Yani veriler arasında hiç bir kural gizli değildir. Sınıflar 75% ve 25% olmak üzere iki tanedir. Bu verilerden 119 düğümlük bir karar ağacı oluşur ve söz konusu test kümesinde yapılan testin sonucu 35% çıkar. Bu sonuç çok başarısızdır. Çünkü bu ağaçta oluşan tahmini hata oranının bile altındadır. Küme A ve B sınıflarını içeriyorsa ve A'nın oranı p ise

- A sınıfının B sınıfı olarak tahmin edilme olasılığı p x (1 - p) (2.10)
- B sınıfının A sınıfı olarak tahmin edilme olasılığı (p - 1) x p (2.11)

dir. İkisinin toplamı 2 x (1-p) x (p) 'dir. Söz konusu veri tabanı için tahmini hata oranı 2 x 0.75 x 0.25=0.37% sonucu bulunur. Karar ağaçları bu tahmini hata oranlarının çok altında bir performansa sahip olabilmelidirler. Oysaki rasgele seçilen bir veri yığının yarattığı ağacın hata oranı bu tahmini hata oranına çok yakındır. Bu durum gösteriyor ki

overfitting'in ikinci ve en büyük sorunu verinin söz konusu olayla ilgili olmaması ya da bir şey ifade etmemesidir.



Şekil 2.14 - Eğitim kümesi ile test kümesinin ağaç boyuna göre doğruluk performansı

Ağaç oluşturma algoritmaları her zaman overfitting oluştururlar. Oluşan ağacın çok büyük olması bu etkiyi artırır. Şekil2.14'te de görüldüğü gibi ağaç dengeli olabilmesi için belli bir büyüklüğün üstünde olmalıdır. Ancak bu büyüklük arttıkça test verisinin hata oranı yükselmekte ve ağacın doğruluğu azalmaktadır. Bu noktada ki en büyük işlem ise ağaç budama işlemidir. Şekilde de görüldüğü gibi zaten ağaç belli bir büyüklükten sonra doğruluğunu kaybetmeye başlıyor. Ağaç oluşturma algoritmaları eğitim kümesini tutarlı yapmaya çalışır. Eğitim setindeki en ufak detayları bile yorumlar. O veriye özgü durumları kurallaştırır. Bu özelleştirme işlemi sırasında ise çok fazla şişer. Ancak önemli olan ağacın eğitim kümesinin tüm verilerini tutarlı kılmak değil sağlama kümesine doğru cevap vermektir. Zaten şekil 2.14 şunu gösteriyor ki eğitim kümesini tutarlı kılmak için şişmeye başlayan ağaç, bu sırada sağlama kümesini ise tutarsızlaştırmaktadır. Dolayısı ile eğitim kümesi için hata oranını yükselten ağaç budama işlemi, aslında sadece eğitim kümesine özgü bazı kuralları temizlemektedir. Bu budama işlemi sanıldığının tersine sağlama verisini daha çok doğrulamaktadır. Böyle bir noktada budama işleminin önemi artmaktadır.

Yukarıda sayılan sebeplerden dolayı budama işlemi KA'larında önemli bir yer tutar. Ağacın daha sade görünmesi ve daha rahat yorumlanması diğer bir avantajdır. İki türlü ağaç budama vardır;

- Ön Budama (Pre-Pruning)
- Sonradan Budama (Post Pruning)

Ön budama işlemi aslında ağaç yaratılırken devreye girer. Bu aşamada ağaç oluşturulurken bir noktada ağacı durdurarak görevini yerine getirir. Örnek olarak bölünen kümeler özellikleri belli bir eşik değerinin üstünde değilse o noktada ağaç bölümleme işlemi durdurulur ve o kümedeki baskın sınıf yaprak olarak yaratılır. Böylelikle gereksiz alt ağaç oluşmasının önüne geçmiş olur. Bu ön budama çok önemlidir çünkü çalışma zamanında ağaca müdahale eder ve zamandan kazanır, bu sayede çok verimli bir yöntemdir. Ancak Breiman bu yöntemin çelişkili performanslar verdiğini savunmuştur [1]. Bu yöntem ağacı çok önceden durdurup, ağacın yeterli olgunluğa gelmesini engellemektedir. Bu tutarsızlık fark edildikten sonra ikinci bir teknik olan Post-Pruning yöntemi kullanılmıştır. Ancak ne var ki çok büyük veri yığınlarında verimli ve hızlı çalışma, tutarlı çalışmanın önüne geçmektedir. Bu sebeple ön budama yöntemi tamamen terkedilmiş bir yöntem değildir.

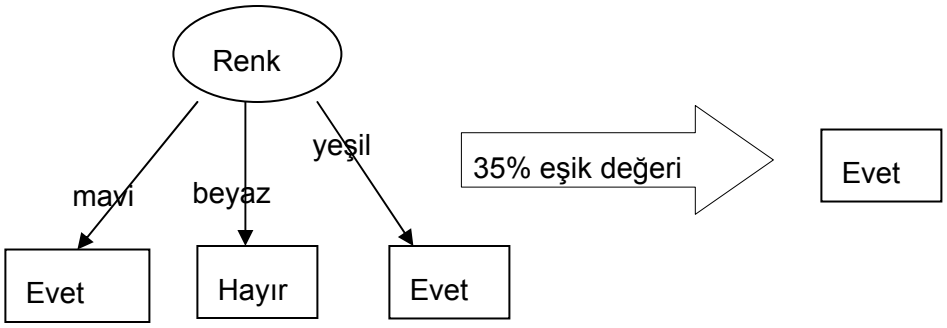
Post–pruning ağaç oluşturulduktan sonra devreye giren bir yöntemdir. Budanmamış T ağacını basitleştirerek T' ağacına çevirir. Budama yöntemi belirli alt ağaçların yerine o alt ağaçlarda en sık bulunan sınıfa sahip bir yaprak koyar. Bu yöntem daha öncede bahsedildiği gibi aslında eğitim kümesindeki hata oranını artırır. Ancak yeni bir olayın test edilmesindeki hata oranını düşürür. Bu da gösteriyor ki budama işlemi aslında hata yaratmaktan ziyade doğruluğu arttırmaktadır.

Bu yöntemdeki en önemli nokta ise budama işlemlerinin nasıl yapılacağıdır. Post-pruning altında üç tip budama metodu vardır.

- ✓ Alt ağaçları silme ve yaprak oluşturma
- ✓ Alt ağaçları yükseltme
- ✓ Dal kesme

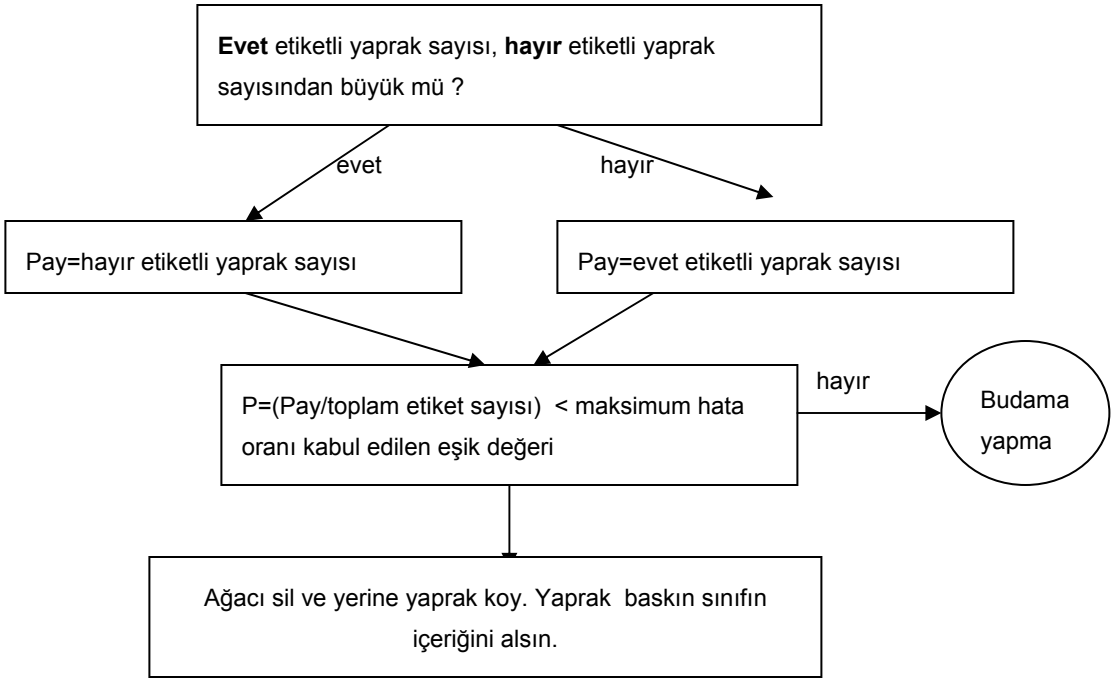
Alt ağaçları silme ve yaprak oluşturma

Alt ağaçları silmek için öncelikle söz konusu alt ağaçta baskın bir sınıfın var olması ve bu baskın olmayı ifade edecek bir eşik değerinin olması gereklidir.



Şekil 2.15 - Ağaç budama

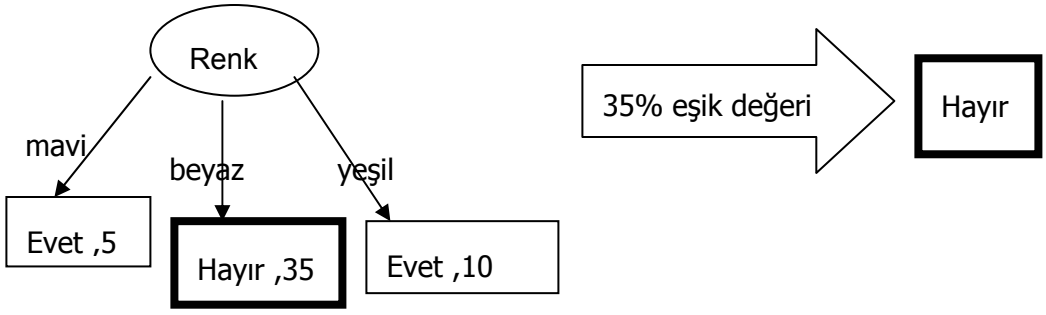
Şekil 2.15'te bir alt ağaç budaması gösterilmiştir. Böyle bir alt ağacı yaprak ile yer değiştirmek için ilk yaklaşım Şekil 2.16'deki gibi olur.



Şekil 2.16 - Ağaç budama algoritmasına ilk yaklaşım

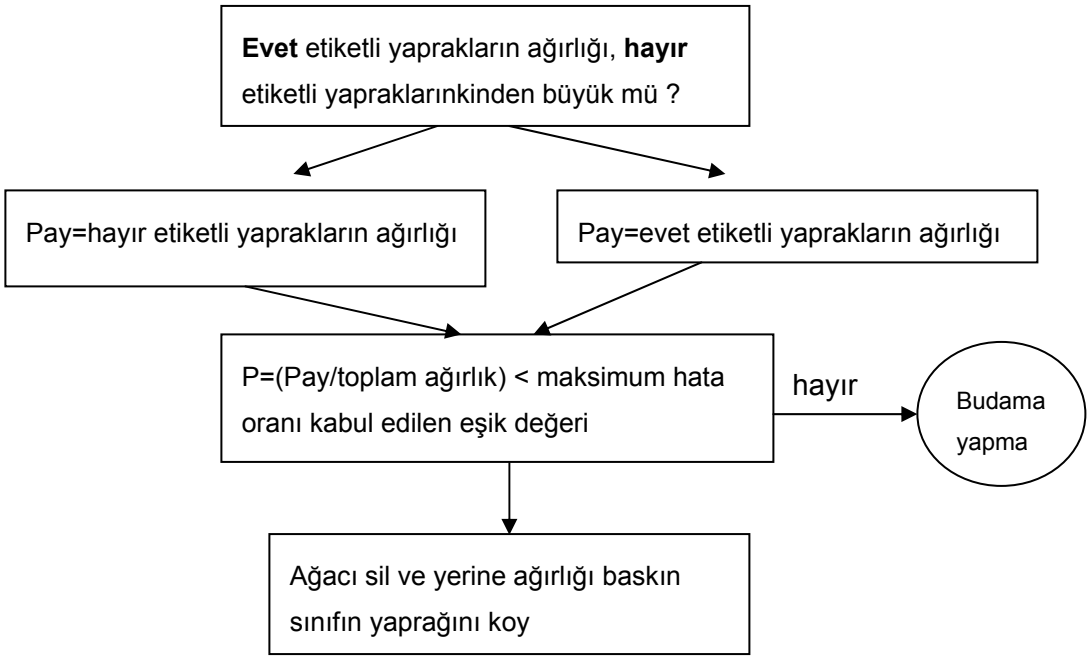
Bu alt ağacı kaldırıp yerine evet isimli yaprak konulmuştur. Bu alt ağaçta 2 adet **evet** sınıfı 1 adet **hayır** sınıfı vardır. Böyle bir budama işleminin yaratacağı yer değiştirme

hata oranı 1/3= 33% olur. Eğer hatalı budama oranımız 33% ‘ün üstünde ise bu budama gerçekleşir. Ağacımızı yarattıktan sonra böyle bir alt ağacı bu yöntemle budamak ilk bakışta mantıklı gelir. Ancak bu yöntemde hatalı bir durum vardır. O da yapraklara verilen değer bilgisiyle ilgilidir. Yani burada görülen 3 yaprak aynı değere mi sahiptir. Eğer sahipse yapılan budama işlemi hesaplandığı gibi 33% çıkacaktır. Ancak ne var ki her yaprağın sahip olduğu olay sayısı vardır. Bir yaprağı yaratan sadece 1 olayın varlığı değildir. Öyle ki şekil 2.14’teki alt ağaçta Renk=beyaz dalından oluşan **hayır** yaprağı 35 olayı kapsayabilir. Yani bu noktadaki bölünmüş veri kümesinin tümü hayır sınıfında olduğu için bu noktada durulmuş ve hayır yaprağı oluşturulmuş olabilir. Renk=mavi dalından oluşan **evet** ve renk=yeşil dalından oluşan **evet** yaprağının kapsadığı olay sayısı sırasıyla 5,15 olursa böyle bir alt ağaç budama işleminin yaratacağı hata oranı birden 35/(30+5+10)=35/50=70% ‘a fırlar ki bu budama çok zararlı olur. Böyle bir aşamada evet yapraklarının olasılığı 30% olduğundan bu alt ağacı silip yerine **hayır** yaprağı konur. Bu noktada söz edilmesi gereken şey her yaprağın kapsadığı bir olay sayısı ya da her yaprağın sahip olduğu bir ağırlık bilgisi vardır. Bu durumda ağaç budama algoritmasını güncellemek gerekecek. Şekil 2.17’teki yaprakların ağırlık bilgileri ile daha doğru ifade edilmiş olur.



Şekil 2.17 - Budamanın ağırlık bilgisi göz önüne alınarak yapılması

Şekil 2.17’teki işlemler olay sayısı ile yapılmıştır. Ancak bu aşamada başka bir noktaya dikkat edilmesi gerekecek. Yapraklar tarafından temsil edilen bölünmüş kümelerini göz önüne alınırken o kümelerin olay sayısı dikkate alındı. Oysa ki kayıp verilere sahip veri kümesinde bölünmüş kümelerde olasılıksal olarak bulunan satırlar da vardır. Yani öyle bir satırın o kümede bulunması bir ihtimalle ifade edilmiştir; daha doğrusu bir ağırlık bilgisiyle. Bu sebeple algoritmada olay sayısı yerine olay ağırlıkları göz önünde bulundurulmalıdır.

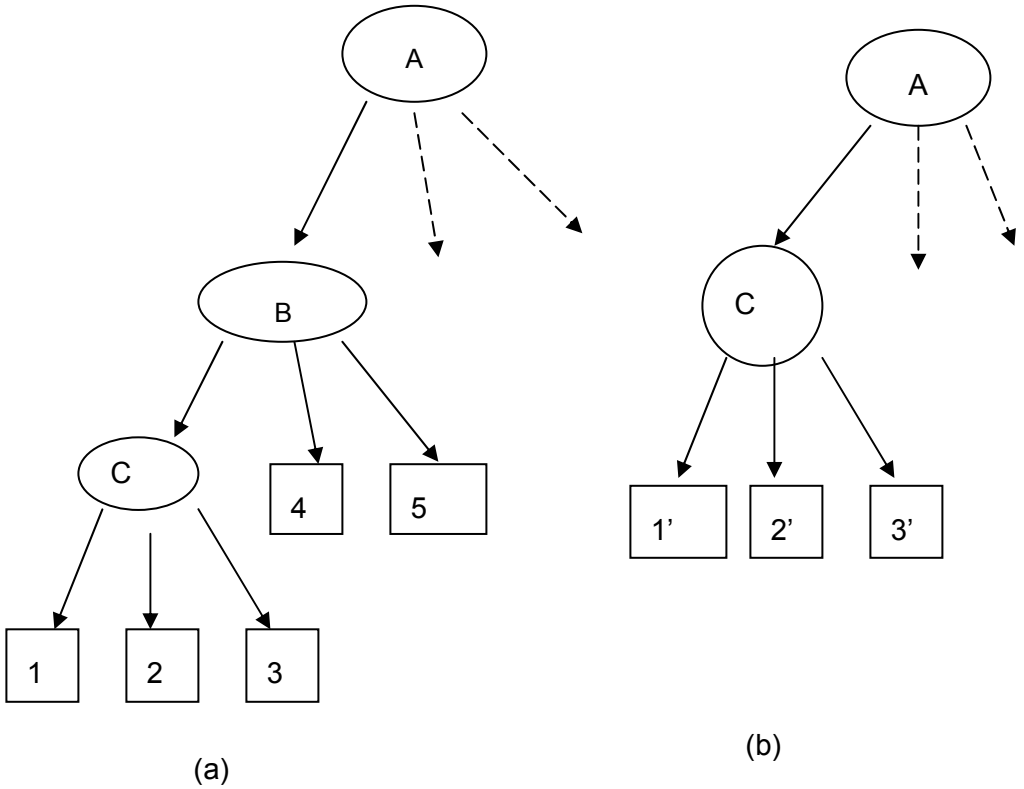


Şekil 2.18 - Ağırlık bilgisi ile yapılan budama algoritması

Şekil 2.18'de de özetlendiği gibi yaprak ağırlıkları hesap edilirken bir önceki konularda geçen kayıp verilerden kaynaklanan ağırlık bilgileri baz alınır. Aslına bakılırsa hayır yaprağının 35 tane olayı kapsaması o yaprağın ağırlığının 35 olduğu anlamına gelmez. Söz konusu yaprağın ağırlığı hesap edilirken o olayların toplamı yerine o olayların ağırlık bilgilerinin toplamı göz önünde bulunur. Kayıp veriler yüzünden oluşturulan bu olay tabanlı ağırlık değerleri söz konusu yaprağın ağırlığı etkiler. Yukarıda verilen örnekte veri tabanında herhangi kayıp bir verinin olmadığı varsayılmıştır. Hayır etiketini yaratan bu 35 olay 20 tanesi 1, 10 tanesi 0.2 ve 5 tanesi 0.1 ağırlığına sahipse bu yaprağın ağırlığı 20x1 + 10x0.2 + 5 x 0.1 =22.5 olur.

Alt ağaçları yükseltme

Alt ağaç yükseltme çok karışık ve belirsiz bir yöntemdir. Sonuçları bazı durumlarda çok yararlı olmayabilir. Fakat C4.5 algoritması bu yöntemi çok etkili bir şekilde kullanmayı başarabilmiştir. Bu yöntemin daha iyi anlatılabilmesi için şekil 2.19'daki örnek verilmiştir.



Şekil 2.19 - Ağaç yükseltme

Şekil 2.19(a)’da gösterilen ağaçtaki C düğümü, Şekil2.19(b)’de de gösterildiği gibi B düğümünün yerine yükseltilmiştir. B düğümü yok edilmiştir. Aslında 4 ve 5 numaralı yapraklar tamamıyla silinmemiştir. Oluşan bu yapı içersinde 1,2,3,4 ve 5 numaralı yaprakların sahip olduğu olaylar tekrar değerlendirilmiş ve bu yeni ağaç yapısı içersinde tekrar değerlendirilmiştir. (b) ile gösterilen ağaçta 1,2 ve 3 numaralı yapraklar 1’,2’ ve 3’ ile numaralandırılmıştır. Yani o yapraklarla ilgili olarak sınıf ve ağırlık bilgileri güncellenmiştir.

Ağaç yükseltme metodu zamanı çok tüketen bir yöntemdir. Bu yöntem mevcut uygulamalarda belirli şartlar içersinde çalışmaktadır. Şekil 2.19’daki örnekte böyle bir yükseltmeyi yapmak için şu şartın olması gerekir: B’ den C’ ye giden dalın taşıdığı olay sayısı ya da daha doğru bir ifade ile olay ağırlıkları toplamı, B’ den 4’e giden ve B’ den 5’e giden olayların toplam ağırlığından çok fazladır. Bu noktada yine bir orantı yapılmış ve bir eşik değeri belirlenmiştir. Bu eşik değerinin altında kaldıkça bu yöntem devreye

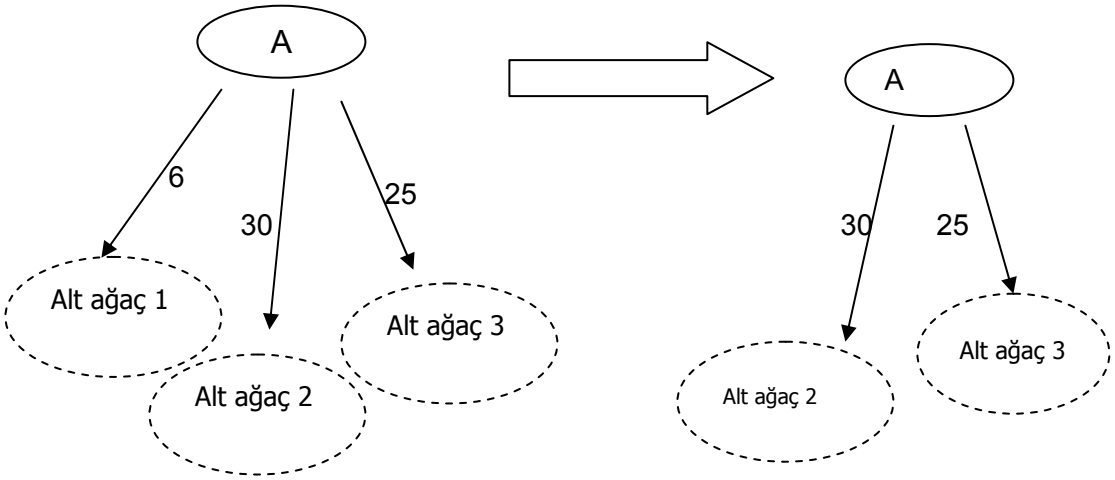
girmektedir. Bu aşamada eğer 4 numaralı yaprağın ağırlığı fazla olsaydı bu sefer 4 numaralı yaprak B düğümünün yerini alacaktı.

Dal kesme

Dal kesme özellik değerlerini yok ettiğinden çok riskli bir budama metodudur. Ancak ağacı sadeleştirme açısından yararlı görülmektedir. Dal kesme işlemi diğer iki budama metoduna çok benzemektedir. Bu işlem yapılırken yine yaprakların ağırlık bilgisine bakılır. Yinelemeli olarak oluşan ağaç taranır. Her düğüme gelindiğinde o düğümün altında ki dalların taşıdığı yüke bakılır. Bu dallardan bazıları çok düşük bir ağırlık değerine sahipse o dal kesilerek ağaçtan atılır. Bu budama işlemi alt ağaçları silip yaprak oluşturma yönteminden farklıdır. Ağacı yaprağa dönüştürme yönteminde söz konusu düğüm altındaki iki sınıf oransal olarak karşılaştırılır. Bu yöntemde söz konusu düğümden çıkan dalların birbirine oranına bakılır. Bu noktada eşik değeri kullanırken çok dikkatli olmak gereklidir. Örneğin 4 tane 25 ağırlığına sahip dalı olan bir düğümde her bir dal %25'lik bir orana sahiptir. Bu durumda %35 bir eşik değeri göz önüne alındığında bütün dallar kesilmiş olur. Bu çok tehlikelidir. Bu yüzden bu hesaplamalar içine o düğümdeki dal sayısı çarpan olarak girer;

(Dal ağırlık bilgisi x dal sayısı) / tüm dalların ağırlık sayısı (2.12)

formülü ile hesap edilir. Şekil 2.20' teki örnek bu yöntemi daha da açıklayacaktır.



Şekil 2.20 - Dal budama

Bu ağaçtaki dalların oranı hesaplanırsa

Dal1= (6 x 3)/ (6+30+25)= (18/61)= 0.29.5

Dal2= (30 x 3)/ (6+30+25)= (18/61)=1.47.5

Dal3= (25 x 3)/ (6+30+25)= (18/61)=1.22

sonuçları çıkar. Eşik değeri 0.5 seçilirse birinci dal düğümden silinir. Bu örnekte birinci dal çok az örnek yüzünden yaratıldığı için ağaçta bir şişmeye yol açmıştır.

Ancak dal kesme KA için çok önemli bir işlem değildir. Bazen zararlı da olabilir. Sadece ağacın daha sade görünmesini ve kolay anlaşılmasını sağlar. Kesilen dala ilişkin yeni bir olay sınıflandırıldığında bu işlem olasılıksal olarak yapılacaktır. Dal kesmenin bir diğer anlamı ise şudur: ilgili kategorik özelliğin öyle bir değeri var ki, bu değer o veri kümesi içinde yok denecek kadar az. Bu durumda böyle bir değer yokmuş gibi davranılmaktadır.

Budama İşlemlerine C4.5 sisteminin getirdiği yaklaşımlar yerinde olmuştur. Ancak bu sistem budanacak ağacın hangi oranla budanacağına karar veren bir mekanizmaya sahip değildir. O bakımdan bu mekanizmayı oluşturacak bir modül oluşturmak gerekir. Bu modül için Bölüm 4'te bahsedilen ağacın doğruluğunu hesaba katmak gerekecektir. Ağacın doğruluğunu maksimum yapacak bir budama oranını belirlemek gerekir.

2.5 Sayısal özellikler için bir iyileştirme

Sayısal verilerle uğraşmak ve onların bilgi kazancını hesaplamak için bir eşik değeri seçilmiş bu eşikten küçük değerler bir kümeye diğerleri karşı kümeye konmuştur. Bu ikili bölme sayısal özelliklerle için rahatlatıcı bir yöntem olmuştur. Ancak denemeler göstermiştir ki sadece sayısal özelikleri iki kümeye bölmek her zaman iyi sonuç vermeyebilir. Ya da daha iyi bir sonuç bulunması ihtimaldir. Bunu için bazı denemeler gerçekleştirilmiştir. Bu denemelerde amaç sayısal özeliklerin 3'e ve 4'e bölünmeleriyle bilgi kazancının nasıl etkileneceğini hesaplamaktır. Bu denemelerden çıkan sonucun olumlu çıkması çok doğaldır. Çünkü sayısal özelliklerin içine gizlenmiş kuralın sayısal değer aralığının üç kısmında bulunması ihtimaldir. Ya da benzer bir şekilde dört bölgesinde bulunma ihtimali de çok yüksek olabilir. Örnek olarak sıcaklık özelliğini

içeren bir sınıflandırma ele alınsın. Bu sınıflandırmada sıcaklığın orta değeri için olumlu diğer değerleri için olumsuz sonuç oluşsun. Bu aşamada T1 dereceden düşük ve T2 dereceden yüksek ortamlarda sınıflandırmanın sonucu olumsuz olsun. Bu örneği spor yapmak için uygun ortamın bulunması ile birleştirirsek 5 dereceden düşük ortamlarda ve 35 dereceden yüksek ortamlarda spor yapmak mümkün değildir. Bu kuralı bulmak için sayısal özelliği ikiye değil üçe bölmemiz gerekecek. Dörde bölme içinde uygun bir örnek verilebilir.

Buna yönelik olarak Tablo 2.3'teki sayısal özellik olan sıcaklık ve nemin bilgi kazançları üçe bölünerek yeniden hesaplanmış ve Tablo 2.11'de gösterildiği gibi bir sonuç alınmıştır.

Tablo 2.11 İkili bölme ve Üçlü bölmede bilgi kazançları

Bilgi Kazancı	İkili Bölme	Üçlü Bölme
Nem	0.15	0.21
Sıcaklık	0.11	0.16

Bu tabloya bakıldığında üçlü bölmede bilgi kazancı biraz yükselmiştir. Ancak ele aldığımız örnek için çok fazla bir şey ifade etmez çünkü elde edilen sonuçlar çok düşük seviyededir.

Üçe bölmenin sağlayacağı yararı görmek için Tablo 2.3'te ki kümenin sıcaklık bilgisini bilgi oranı üçlü bölümleme için yüksek olacak şekilde değiştirilsin. 70 ile 80 arasındaki sıcaklık değerleri için sınıflama hayır, diğer iki bölge için evet olsun. Bu düzenleme sonucunda Tablo 2.12'deki gibi bir sonuç oluşur.

Tablo 2.12 - Sıcaklık bilgisine göre düzenlenmiş tablo

	Hava durumu	Sıcaklık (°F)	Nem(%)	Rüzgar	Sonuç
1	Güneşli	75	70	Var	Hayır
2	Güneşli	80	90	Var	Evet
3	Güneşli	85	85	Yok	Evet
4	Güneşli	72	95	Yok	Hayır
5	Güneşli	69	70	Yok	Evet
6	Kapalı	72	90	Var	Hayır
7	Kapalı	83	78	Yok	Evet
8	Kapalı	64	65	Var	Evet
9	Kapalı	81	75	Yok	Evet
10	Yağmurlu	71	80	Var	Hayır
11	Yağmurlu	65	70	Var	Evet
12	Yağmurlu	75	80	Yok	Hayır
13	Yağmurlu	68	80	Yok	Evet
14	Yağmurlu	70	96	Yok	Evet

Bu tablonun sıcaklık özelliğinin ikili ve üçlü bölme için verdiği bilgi kazancı değerleri hesaplandığında ikili bölme için 0.3 ve üçlü bölme için 0.94 değeri elde edilir. Görüldüğü gibi sıcaklık özelliğine gizli bir kural koyduğumuz için ikili bölme için bilgi oranı 0.11'den 0.3'e yükselmiştir. Ancak çok net bir kural yerleştirilmesine karşın bilgi oranı yinede 0.3 civarlarında kaldı ki bu değer çok iyi değildir. Diğer yandan üçlü bölme için bilgi oranı 0.16'dan 0.94'e yükselmiştir. Bu değer zaten sıcaklık özelliğinin tek başına tüm ağacı oluşturacağına işarettir. Yani üçlü bölme gizlenmiş olan kuralı çıkarmıştır.

Eğer kural ikili bölme içinde gizlenmiş olsaydı bu sefer üçlü bölmenin bilgi kazancı düşük olacaktı. Çünkü gereksiz bölgeler oluşturulmuş ve ağaca gereksiz dallar eklenmiş olacaktı. Üçlü bölme için yapılan bu yaklaşımın aynısı dörtlü ve de beşli bölme içinde söylenebilir. Yani kurallar sayısal özelliğin yarattığı uzayın dört bölgesine ya da beş bölgesine yerleşmiş olabilir. Beşten fazla bir bölme işleminin yapılması fazla bir yarar getirmeyecektir. Bu bakımdan beşli bölme işlemine kadar sayısal özelliklerin bilgi kazançlarına bakılmasında fayda vardır. Bundan dolayı sayısal özelikler için şu şekilde bir algoritma uygulanacaktır:

- İkili bölme için maksimum bilgi oranını veren noktayı ve bilgi kazancını hesapla.
- Üçlü bölme için maksimum bilgi oranını veren noktaları ve bilgi kazancını hesapla.
- Dörtlü bölme için maksimum bilgi oranını veren noktaları ve bilgi kazancını hesapla
- Beşli bölme için maksimum bilgi oranını veren noktaları ve bilgi kazancını hesapla.

Tüm sonuçlar arasındaki en iyi bilgi kazancını veren noktalara ya da noktaya göre bölme işlemini gerçekleştir.

3. KARAR AĞACI OLUŞTURMA UYGULAMASI

3.1 Amaç

Bu tezde amaç Quinlan'ın KA oluşturma yaklaşımını ve diğer alternatif yaklaşımları göz önüne alarak, Java programlama dili desteği ile bir KA oluşturma yazılımı geliştirmektir. Yazılan program belli formatta toplanmış veri yığınından KA oluşturma ve bu ağacı budama işlemlerini yerine getirmektedir. Bu veriler önceki bölümlerde sözü edilen algoritmalar aracılığıyla karar ağacını oluşturur. Nesneye yönelimli yapı sayesinde ağaç yapısı programla dili seviyesinde de rahatlıkla temsil edilebilmiştir. Oluşturulan ağaç Java grafik kütüphaneleri aracılığıyla görselleştirilmiştir. Yazılım 13 adet veri kümesini denemiştir. Bu deneyin sonuçları kaydedilmiş ve değerlendirilmiştir. Bu sebeple Java programlama dili ayrıntılı olarak incelenecektir.

3.2 Java Programlama Dili

Java programlama dili C,C++ gibi geleneksel dillerin sentakslarını kullanır. Aynı zamanda "Nesneye Yönelimli Programlama" (Object-Oriented Programming) kavramının başını çeken bir dildir. Öncelikle Java'daki her şey birer nesnedir. Bütün kodlar class diye tabir edilen bir deyim ile başlar ki bu deyim o kod parçasının bir nesne olduğunu tanımlar. Yaratılmış olan bu class isimli kod parçacıkları derlendikten ve çalıştırıldıktan sonra tekrar başka bir kod parçası tarafından kullanılabilirler. Bir anlamda her Java programcısı kendi veri modellerini class olarak yaratırlar.

Java platformu, bilgisayar ağının önemi hesaba katılarak ve aynı yazılımın birçok değişik bilgisayar ortamında veya değişik makinelerde çalışması fikri ile geliştirilmiş yeni bir teknolojidir. Java teknolojisi kullanılarak aynı uygulamayı değişik ortamlarda çalıştırılabilir (Örnek: Microsoft,Unix).

Java platformu hem programlama dili hem de bir ortam olarak düşünülebilir. Programlama dili olarak, açık, nesneye yönelik (object-oriented), güvenli, sağlam, internet için elverişli bir teknolojidir. Ortam olarak da işletim sistemi, veri tabanı ve orta katman (middleware) teknolojileri bulunur

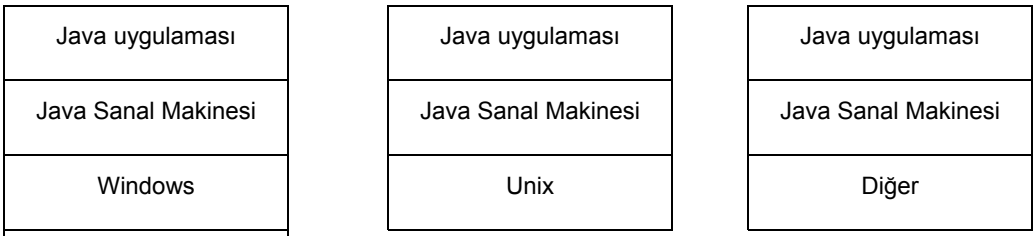
Gerek Java programlama dili, gerekse bu dile bağlı alt teknolojiler tarafından geliştirilmiş olan başlı başına ürünler değillerdir. Java ve bu dile bağlı alt teknolojiler, Sun MicroSystems tarafından verilmiş açıklamalardan oluşmaktadır. Bu açıklamalara sadık kalan her yazılım firması JVM (Java Virtual Machine - Java Sanal Makinası) veya Java programlama diline bağlı alt teknolojiler yazabilir (örnek: Application Server - Uygulama sunucusu). Eğer bu açıklamalara sadık kalmayıp, standart dışı bir JVM veya Java programlama diline bağlı alt teknolojiler yazılmaya kalkılırsa hukuki bir suç işlemiş olur (Microsoft firmasının yaptığı gibi). Şu anda en yaygın kullanılan JVM ler, IBM'in ve Sun Microsystems' in üretmiş olduklarıdır, ayrıca HP, Apple ve daha bir çok firmanın üretmiş oldukları JVM'ler bulunmaktadır.

Java Programlama dili ile projeler diğer programlama dillerine göre daha kolay ve sağlıklı bir şekilde yapılabilmektedir. Java programlama dili şu başlıklara destek verebilmektedir.

- GUI (graphical user interface, grafiksel kullanıcı ara yüzü) uygulamaları ve appletler.
- Dağınık bileşenler (Distributed components) (örnek: EJB, RMI, CORBA).
- Web tabanlı uygulamalar (Applet, servlet, Jsp).
- Veri tabanlarına erişim ile ilgili uygulamalar(JDBC).
- Cep telefonları, Smart kart' lar için uygulamalar.
- Diğer…

3.2.1 Çalışma Disiplini

Java uygulamaları JVM (Java Virtual Machine) tarafından yorumlanır. Şekil 3.1'de de gösterildiği gibi JVM, işletim sisteminin en tepesinde bulunur. Bu sayede yazılmış olan Java uygulamaları değişik işletim sistemlerinde, herhangi bir değişiklik yapılmadan çalışabilir. Böylece Java programlama dilinin felsefesi olan "Bir kere yaz her, yerde çalıştır" sözü gerçekleştirilmiştir.



Şekil 3.1 – Java Sanal Makinası



Şekil 3.2 - Java kodunun geçirdiği evreler

Byte koduna çevrilen kaynak kod, Şekil 3.2'de de gösterildiği gibi JVM tarafından yorumlanır ve uygulama çalıştırılmış olur. Şekil 3.3'teki program üzerinde incelenirse;

```
public class merhaba {
    public static void main(String args[]) {
        System.out.println("Merhaba bu benim ilk Java Programım !");
```

Şekil 3.3 - ilk java programı

Bu program parçası "merhaba.java" şeklinde kaydedilmelidir. Çünkü örnek kod içersinde tanımı yapılan class'ın ismi verilmelidir. Java platformundaki ilk dikkat edilmesi gereken nokta budur. Ardından kodun hemen çalıştırılması için statik bir metodun yazılması gerekir.

public static void main(String args[])

ile başlayan metot bu class'ın ana metodudur.

3.2.2 Gelişim evreleri

Java'nın gelişim evreleri Tablo3.1'de özetlenmiştir.

Tablo 3.1 - Java'nın gelişim evreleri

1995	• Java teknolojisinin ilk çıkış yılı ; ilk olarak applet teknolojisinin dikkat çektiği senelerdir.
1996	• Java Development Kit (JDK) v1.0 çıkartıldı. Temel seviyeli işlevleri içeren bir versiyon (örnek: soket programlama, dosya I/O, GUI)
1997	• JDK 1.1 çıkartıldı. Bu sürümde Java GUI, veritabanı erişimi için JDBC, dağınık nesneler için RMI ve daha birçok yeni gelişmeler eklendi .
1998	• JDK 1.2 çıkartıldı . • JFC/Swing yayınlandı- aynı sene içersinde 500,000 üstünde download gerçekleştirildi.
1999	• Java teknolojisi J2SE, J2EE ve J2ME olarak 3'e bölündü . • Java HotSpot (performans arttırıcı) yayınlandı . • JavaServer Pages (JSP) teknolojisi yayınlandı. • J2EE platform'u yayınlandı . • Linux üzerinde J2SE platformu yayınlandı .
2000	• JDK v1.3 çıkartıldı . • Java APIs for XML teknolojisi yayınlandı .
2002	• JDK v1.4 versiyonu çıkarıldı (Merlin projesi). • Java API for XML binding yayınlandı.
2003	• JDK v1.4.1

3.2.3 Dilin Yapısı

Java'nın başarılı olmasında aşağıdaki sebepler şunlardır;

- C/C++ da olduğu gibi hafıza problemlerinin olmaması
- Nesneye yönelik (Object - Oriented) olması

- C/C++/VB dillerinin aksine dinamik olması
- Güvenli olması
- İnternet uygulamarı için elverişli olması. (Applet, JSP, Servlet, EJB, Corba, RMI)
- Çok değişik uygulamalar için hazır kütüphaneleri olması (Java3D, JDBC, vb.)
- Platform bağımsız olması

Java'da Nesne

Her programlama dilinin kendine has veri yönetim şekli bulunur. Java platformunda çalışan bir uygulama çalışma esnasına nesneler oluşturur. Java programlama dilinde her şeye nesne olarak davranılır. Her şeyin nesne olmasına karşın, bu nesneleri kullanmak için referanslara ihtiyaç duyulur.

Örneğin bir maket uçak nesne olarak düşünülsün ve bu maket uçağı kontrol etmek amacı ile bir de kumanda cihazının olduğunu düşünülsün. Bu maket uçağını havada sağa veya sola döndürmek için eldeki kumanda cihazını kullanmak zorunludur. Aynı şekilde bu maket uçağını havalandırmak veya yere indirmek içinde elde bulunan kumanda cihazından faydalanılır. Burada dikkat edilmesi gereken husus kumanda cihazından çıkan emirlerin maket uçak tarafından yerine getirilmesidir.

İlkel Tipler

Java'da her şey sınıflardan oluşmaktadır, ancak bazı veri tipleri vardır ki onlar hem bir sınıftır hem de en düşük seviyede işlemler yaparlar. Bu gruba ilkel tipler denir. Bu tiplere çoğu yerde ihtiyaç duyulur. Bu sebepten bu ilkel tipleri **new** anahtar kelimesi ile oluşturmak pek de avantajlı olmamaktadır. Yalnız burada her ilkel değişkenin bir referans olmamasıdır, yani bu ilkel tipler değerlerini kendi üzerlerinde taşırlar. İlkel tiplerin özellikleri Tablo 3.2'de gösterilmiştir.

Tablo 3.2 - İlkel tiplerin özellikeri

İlkel tip	Boyut	Minimum	Maksimum	Sarmalıyıcı tip
boolean	—	—	—	Boolean
Char	16- bit	Unicode 0	Unicode 2^{16}- 1	Character
Byte	8- bit	-128	+127	Byte
short	16- bit	-2^{15}	$+2^{15}$—1	Short
İnt	32- bit	-2^{31}	$+2^{31}$—1	Integer
Long	64- bit	-2^{63}	$+2^{63}$—1	Long
Float	32- bit	IEEE754	IEEE754	Float
double	64- bit	IEEE754	IEEE754	Double
void	—	—	—	Void

Bu tablo tüm ilkel tipleri açıklamaktadır.

Geçerlilik Alanı

Her programlama dilinde değişkenlerin geçerlilik alanı kavramı bulunur.

```
{
int a = 12;  /* sadece a mevcut*/
  {
    int b = 96;    /*  a ve b mevcut */
  }
    /* sadece a mevcut */
    /* b geçerlilik alanının dışına çıktı */
}
```

Şekil 3.4 - Java'da geçerlilik alanı

Java programlama dili içerisindeki geçerlilik kavramının nasıl olduğu önemlidir. Şekil3.4'teki örnekte iki değişkeninin geçerlilik alanları incelenmektedir. İlkel integer tipinde olan **a** değişkeninin geçerlilik alanı kendisinden daha iç tarafta olan alanlarda

bile geçerlidir. Ama ilkel int tipindeki b değişkenini incelenirse, bu değişkenin kendisinden daha dış tarafta olan alanda geçerli olmadığı görülür.

Yeni Sınıf Oluşturma

Şekil 3.5'te Java programlama dilinde özgün bir sınıf oluşturmaya örnek verilmiştir.

```
public class YeniBirSinif  {
// gerekli tanimlar ..
}
```

Şekil 3.5 - Java'da yeni bir sınıf oluşturma

Aşağıda oluşturulan sınıfın hiç bir işlevi yoktur ama içine bir şeyler yazılarak etkin bir sınıf haline çevrilebilir.

Alanlar ve Metotlar

Alanlar yani global değişkenler, ilkel bir tip veya bir başka sınıf tipinde olabilirler. Şekil 3.6'daki kod bir sınıfın global değişkenleriyle birlikte yaratılmasına örnekler.

```
public class YeniBirSinif  {
  public int i = 5 ;
  public float f = 3.23;
  public boolean b = true ;
}
```

Şekil 3.6 - Yeni bir sınıfın global değişkenlerini

Bu alanlar kullanılmadan evvel onların ilk değerlerinin belli olması gerekmektedir. Ancak ilk değerleri verilmez ise her tip için şekildeki tablo geçerli olacaktır.

Tablo 3.3 - İlkel tiplerin varsayılan değerleri

İlkel Tip	Varsayılan değer
boolean	False
char	'\u0000' (null)
byte	(byte)0
short	(short)0
int	0
long	0L
double	0.0d
float	0.0f

Eğer bir global değişkene ilk değerler verilmez ise, Java bu global değişkenlere kendi varsayılan ilk değerlerini verir, bu mevcut değerler tablo 3.3'daki gibidir. YeniBirSinif isimli sınıf içerisinde hala işe yarar bir şeyler yok gibi sadece 3 adet global değişken tanımlandı. Bu YeniBirSinif sınıfına ait bir nesne

```
YeniBirSinif   ybs = new YeniBirSinif();
```

şeklinde oluşturulur.

ybs isimli referansı YeniBirSinif nesnesini göstermektedir. Eğer YeniBirSinif nesnesi ile temas kurulmak istenirse **ybs** referansı kullanılır. Nesnenin alanlarına ulaşmak için "**.**" nokta kullanılır.

Metotlar

Metotlar, nesnelerin işe yarar hareketler yapmasına olanak sağlayan önemli bölümleridir. Şekil 3.7'deki örnekte bir metodun iskeleti incelenmektedir.

```
dönüşTipi  metodunIsmi( /* argüman listesi */ ) {
/* Metod gövdesi */
}
```

Şekil 3.7 - Geri dönüş tipinin belirlenmesi

DönüşTipi: Bir metot ya değer döndürür veya döndürmez. Bu değer bir ilkel tip olabilir veya bir nesneye bağlanmış referans olabilir. Eğer bir metot hiçbir şey döndürmüyorsa **void** eki metodun başına yerleştirilir

metodunIsmi: Java'nın kendisine ait olan kelimeler olan if, else, import, class, return gibi ve Türkçe karakterler haricinde istenilen isim kullanılır. Ama metotlar bir eylem içerdikleri için, metot isimlerinin de bir eylemi belirtmesi tercih edilir örneğin:

sayiSirala(), enBuyukSayiyiBul(), sqlCalistir() vb isimler tercih edilir.

Argüman listesi= Metot içerisinde işlemler yapabilmek için gerekli olan parametreler. Bu parametreler ilkel tipte veya sınıf tipinde olabilirler.

Metot gövdesi = Bu kısım alt metodun amacına yönelik her türlü kod ile yazılır.

```
int boyutDondur(String kelime) {
      return kelime.length() ;
  } // metodun sonu
```

Şekil 3.8 - Geri dönüş tipine bir örnek

Şekil 3.8'te boyutDondur() metodu String tipinde parametre alıyor ve bu String tipindeki değişkenin bağlı olduğu String nesnesinin boyutunu geri döndürüyor. Metodun geri döndürdüğü değer ilkel int tipindedir. Herhangi bir değer geri döndürebilmek için return anahtar kelimesi kullanılır. Eğer metot hiç bir değer döndürmeyecekse metot tanımındaki satırın başına void anahtar sözcüğü eklenir.

Diziler

Diziler bu proje kapsamında sıkça kullanılan yapılar olmuştur. Bu bakımdan açıklanmasında fayda vardır. Diziler objedir, dizi objesi içinde belli sayıda değişken bulundurur. Eğer bu sayı sıfır ise dizi boş demektir. Dizinin içersindeki değişkenlere

isimleri ile değil, negatif olmayan tam sayı ile ifade edilen dizi erişim indeks(gösterge) ile erişilir. Dizi içersindeki değişkenler dizinin elemanlarıdır.Eğer bir dizide **n** tane eleman varsa dizinin uzunluğunda n kadardır. Dizi içersindeki ilk eleman 0, son eleman n-1 indeksi ile belirtilir.

Dizi içersindeki her elemanın tipi aynı olmak zorundadır. Eğer dizi içersindeki bir elemanın tipi double ise, bu dizinin tipi için double' dır. Diziler ilkel tipteki veya herhangi bir nesne tipindeki elemanlardan oluşabilirler.

Dizi tipindeki değişkenler

Dizi tipindeki değişkenler objeye bağlanmaktadır. Dizi tipinde değişken tanımlamak, dizinin hemen kullanacağı anlamına gelmez.

```
double[] dd  ; // double tipindeki dizi
double dd[] ;  // double tipindeki dizi
float [] fd ;  // float tipindeki dizi
Object[] ao ;  // Object tipindeki dizi
```

Şekil 3.9 - Dizi yaratılması

Şekil 3.9'da sadece dizi objelerine bağlanacak olan değişkenler tanımlandı. Bu dizi objeleri hafıza alanında henüz yer kaplamamışlardır. Dizi objelerini oluşturmak için **new** anahtar kelimesinin kullanılması gereklidir**.**

Dizi oluşturmak

Dizilere herhangi bir nesne gibi ulaşılabilir. Bellekte yaratılma biçimleri Şekil 3.10'da gösterilmiştir.

```
double[] d = new double[20] ; // 20 elemanlı double tipindeki dizi
double dd[] = new double[20]; // 20 elemanlı double tipindeki dizi
float [] fd = new float [14]; // 14 elemanlı float tipindeki dizi
Object[] ao  = new Object[17]; // 17 elemanlı Object tipindeki dizi
String[] s = new String[25] ;  // 25 elemanlı String tipindeki dizi
```

Şekil 3.10 - Dizilerin bellekte yaratılma biçimleri

Örneğin **new** double[20] ifadesi ile 20 elemandan oluşan double dizisi elde edilmiş oldu.

Kalıtım

Kalıtım konusu nesneye yönelik programlamanın (object oriented programming) en önemli kavramlarından bir tanesidir. Kalıtım kavramı, kısaca bir sınıftan diğer bir sınıfın türemesidir. Yeni türeyen sınıf, türetilen sınıfın global değişkenlerine ve metotlarına

```
Class Kedi{

}
```

Şekil 3.11 – Bir java sınıfı

```
Class Kaplan extend Kedi{

}
```

Şekil 3.12 - Türetilen bir sınıf

otomatik olarak sahip olur.

Şekil 3.11'de yalın olarak ifade edilen Kedi sınıfından türeyen ve Şekil3.12'de gösterilen Kaplan sınıfı bu yapıya bir örnektir. Her kaplan kedisel özellikler taşıyacaktır ama bu

```
class Kedi {
   protected int ayakSayisi = 4 ;
      public void yakalaAv() {
      System.out.println("Kedi sinifi Av yakaladi");
   }

   public static void main(String args[]) {
      Kedi kd= new Kedi() ;
      kd.yakalaAv() ;
   }

}

class Kaplan extends Kedi {
   public static void main(String args[] ) {
      Kaplan  kp = new Kaplan();
      kp.yakalaAv();
      System.out.println("Ayak Sayisi = " + kp.ayakSayisi) ;
   }
}
```

Şekil 3.13 - Sınıf Türetme

özelliklerin üzerine kendisine bir şeyler ekler. Yazılış ifadesi olarak, türeyen sınıf isminin yanına **extends** ifadesi koyulur ve hemen sonrasında türetilen sınıfın kendisi yerleştirilir.

Şekil 3.13'te ayrıntılı olarak gösterilen Kaplan sınıfı Kedi sınıfından türemiştir. Görüldüğü üzere Kaplan sınıfının içersinde ne **yakalaAv()** metodu ne de **ayaksayisi** değişkeni tanımlanmıştır. Kaplan sınıfı bu özelliklerini kendisinin ana sınıfı olan Kedi sınıfından almıştır. Kedi sınıfının içersinde tanımlanmış ayaksayisi değişkeni, protected erişim belirleyicisine sahiptir.Bunun anlamı, bu değişkene aynı paket içersinde olan sınıflar ve ayrı paket içersinde olup bu sınıftan türetilmiş olan sınıfların erişebileceğidir. Böylece Kaplan sınıfı ister Kedi sınıfı ile aynı pakette olsun ister olmasın, Kedi sınıfına ait global int ilkel tipinindeki değişkene (ayaksayisi) erişebilir.

Gizli Kalıtım

Oluşturulan her yeni sınıf otomatik ve gizli olarak **Object** sınıfından türer. Object sınıfı Java'daki tüm diğer sınıfların ana sınıfıdır. Şekil 3.14'te bu duruma bir örnek verilmiştir.

```
public class YeniBirSinif {
public static void main(String[] args) {
YeniBirSinif ybs1 = new YeniBirSinif();
YeniBirSinif ybs2 = new YeniBirSinif();
System.out.println("YeniBirSinif.toString() " + ybs1 ) ;
System.out.println("YeniBirSinif.toString() " + ybs2 ) ;
System.out.println("ybs1.equals(ybs2) " + ybs1.equals(ybs2) ) ;
// ....
}
}
```

Şekil 3.14 - Yeni bir sınıfta gizli kalıtım

```
YeniBirSinif.toString() YeniBirSinif@82f0db
YeniBirSinif.toString() YeniBirSinif@92d342
ybs1.equals(ybs2) false
```

Şekil 3.15 - Uygulamanın çıktısı

YeniBirSinif sınıfında, toString() ve equals() metotları tanımlanmamasına rağmen bu metotlar kullanıldı. Yeni bir sınıf tanımlandığında, Java gizli ve otomatik olarak **extends Object** ibaresini yerleştirir.

Bu sayede Object sınıfına ait metotları kullanılabilir. Object sınıfına ait obje metotları aşağıdaki gibidir.

clone() : Bu objenin aynısını klonlar ve yeni bir obje geri döndürür

equals(Object obj) : obj objesi, bu objeye eşit mi kontrolü yapar

finalize() : Bu objeye herhangi bir değişken bağlı bulunmadığında, çöp toplayıcısı bu objeyi hafızadan silmeden önce çağırdığı metod

getClass() : Bu objenin çalışma anında Class bilgilerini geri döner

hashCode() : Bu objenin hash kodunu geri döner

notify() : Bu objenin monitöründe olan tek thread'i uyandırır.

notifyAll() : Bu objenin monitöründe olan tüm thread leri uyandırır.

toString() : Bu objenin String tipinden ifadesini geri döner.

wait() : O andaki thread'in beklemesini sağlar; bu bekleme notify() veya notifyAll() metodları sayesinde sona erer.

wait (long timeout) : O andaki thread'in belirtilen süre kadar beklemesini sağlar; bu bekleme notify() veya notifyAll() metodları sayesinde de sona erebilir.

wait (long timeout, int nanos) : O andaki thread'in belirtilen gerçek süre kadar beklemesini sağlar ; bu bekleme notify() veya notifyAll() metotları sayesinde de sona erebilir.

Kısacası, oluşturulan her yeni sınıf, yukarıdaki, metotlara otomatik olarak sahip olur. Bu metotlar yeni oluşan sınıf tarafından istenildiği gibi yazılabilir.

3.2.4 Java'da Grafik ve Arayüz

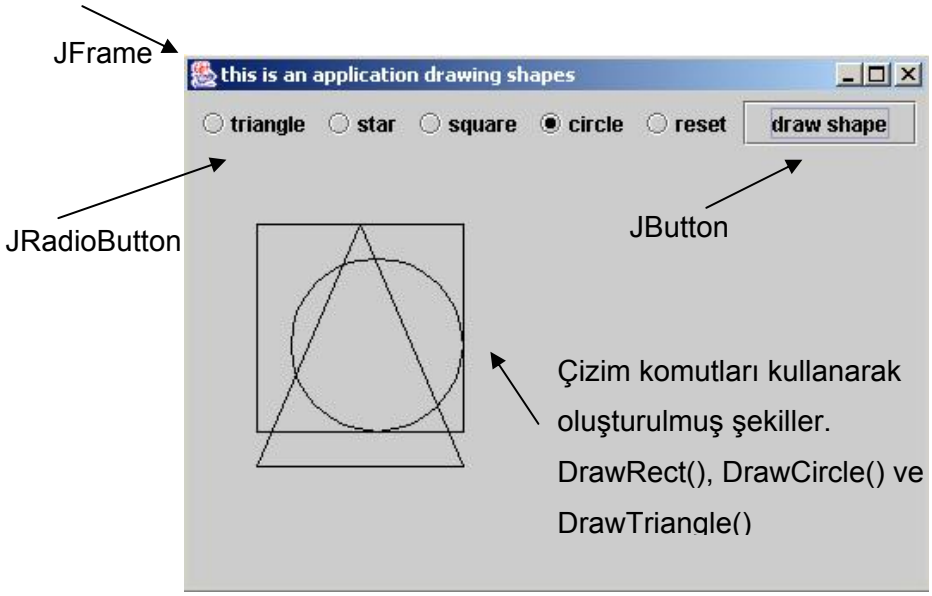
Java oldukça gelişmiş bir grafik desteğine sahiptir. Her türlü çizim işlemine olanak sağlayan Graphics kütüphanesiyle temel çizim işlemleri kolaylıkla yapılabilmektedir. Ancak bu kütüphaneyi daha gelişmiş olan Graphics2D kütüphanesi takip eder. Bu iki kütüphane ile akla gelebilen tüm iki boyutlu işlemler yapılabilir. 3D çizimi desteklemeyen bu kütüphanelerin yerine ise Graphics3D kütüphanesi kullanılır. Tüm grafik desteği bu kütüphanelerinden sağlanır. Graphics3D, C dilinin desteklediği ve grafik ortamının en iyi kütüphanesi olan openGL'in sahip olduğu tüm yeteneklere sahiptir. Aşağıdaki komutlar Graphics kütüphanesinin temel bir kaç yeteneğini temsil etmektedir.

- DrawLine() - basit çizgiler çizer
- DrawString() – Grafik zemin üzerine yazı yazar

- DrawPolygon(p) – p isimli poligonu çizer.
- DrawCircle() - Çember çizer.

Grafiksel Arayüz (GUI)

Java grafiksel arayüz bakımından oldukça gelişmiştir. AWT ve Swing isimli çok yetenekli paketlere sahiptir. Swing paketi, AWT paketine göre daha gelişmiştir. Bu kütüphaneler sayesinde grafik tüm objeler ve arayüzler yaratılabilir. JButton, JLabel, JCheckBox, JRadioButton, JTextField, JScrollBar, JMenuBar vb. değişkenler Java içinde hazır bulunan bazı nesnelerdir.

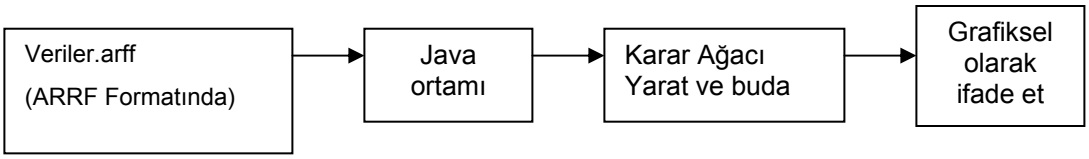


Şekil 3.16 - Grafiksel ara yüz örneği

Şekil 3.16'deki uygulama bu kütüphanelerin kullanımına güzel bir örnek teşkil etmektedir.

3.3 Program

Bu tez kapsamında yazılan program KA yaratma ve bu karar ağacını budama işlemlerini yerine getirmektedir. Programın yaptığı işlemler şekil 3.17'de gösterildiği gibi özetlenebilir. ARFF formatındaki veriler Java ortamında dizi tipindeki değişkenlere aktarılırlar ve artık bu şekilde temsil edilirler.



Şekil 3.17 - Programın aşamaları

Ardından kod seviyesinde ağaç oluşturma ve diğer işlemler yapılır. En sonunda ortaya çıkan ağaç grafik ortamda ifade edilir ve yorumlanır. ARFF formatından başlanarak bu adımlar açıklansın.

3.3.1 ARFF Format

Bu format makine öğrenmesinde kullanılan bir formattır. Waikato Üniversitesi tarafından "Weka Makine Öğrenmesi Projesi" için hazırlanmıştır. Verilerin daha düzenli ve daha sıralı olmasını sağlayan bu format aslında veritabanlarını metin tabanında taklit etmektedir. Bu format aracılığıyla okunan veriler programlama seviyesinde karakter dizileri (String Array) biçimde temsil edilirler. @relation, @attribute ve @data deyimleri aracılığıyla dosyanın yapısı belirlenir. @relation veri yığının genel amacını ya da ismini belirtir. @attribute verideki veri tabanında sütunlara karşılık gelen özellik isimlerini belirtmek için kullanılır. @data ham verilerin başladığı satıra işaret eder.

```
@relation oyun oynama

@attribute hava durumu {güneşli, kapalı, yağmurlu}
@attribute sıcaklık real
@attribute nem real
@attribute rüzgarlı {var, yok}
@attribute oyun {evet, hayır}

@data
güneşli,85,85, yok, hayır
güneşli,80,90, var, hayır
kapalı ,83,86, yok, evet
yağmurlu ,70,96, yok, evet
yağmurlu ,68,80, yok, evet
yağmurlu ,65,70, var, hayır
kapalı ,64,65, var, evet
güneşli,72,95, yok, hayır
güneşli,69,70, yok, evet
yağmurlu ,75,80, yok, evet
güneşli,75,70, var, evet
kapalı ,72,90, var, evet
kapalı ,81,75, yok, evet
yağmurlu ,71,91, var, hayır
```

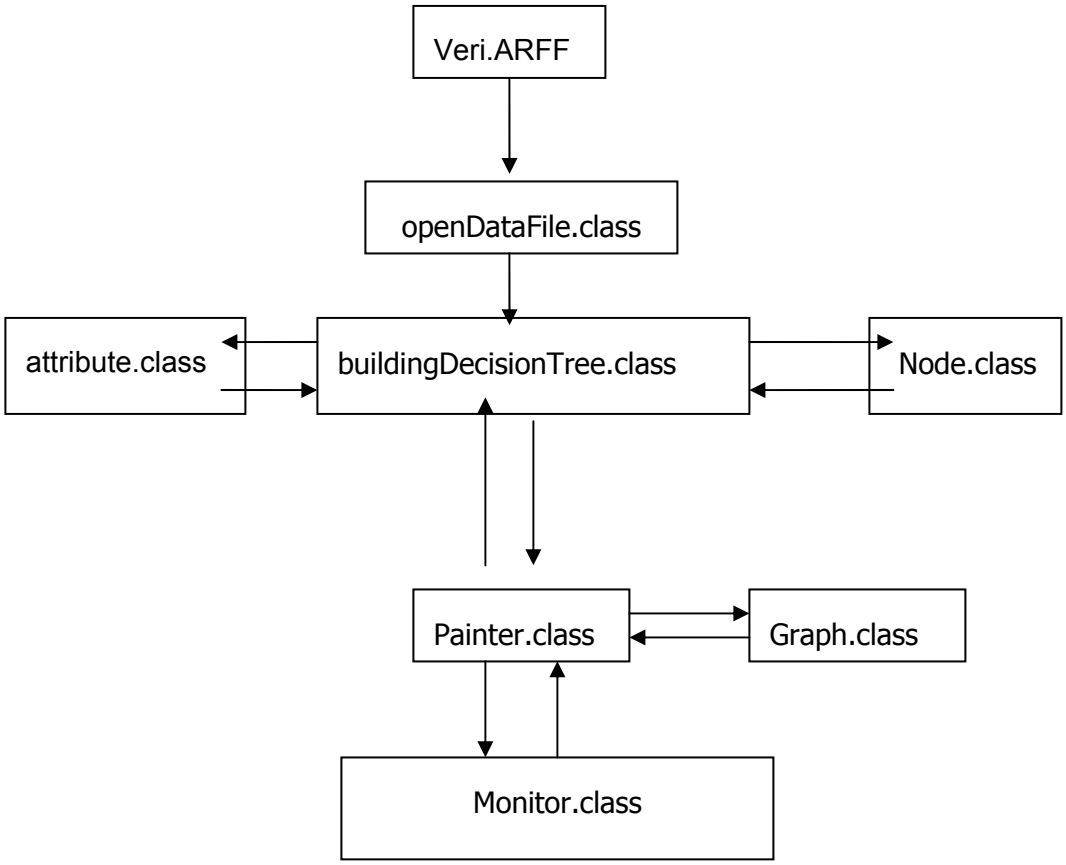
Şekil 3.18 - ARFF formatında hava durumu verisi

Şekil 3.18 ARFF formatına uygun bir veri yığınını göstermektedir. Görüldüğü gibi bu veri yığını @relation ile kendini tanımlamıştır. Verideki özelliklerin isimleri @attribute ile başlayan satırlara yazılmıştır. Bu satırdaki real deyimi o özelliğin sayısal olduğunu ifade etmektedir. @data deyiminden hemen sonraki satırdan veriler @attribute deyimi ile sıralanan özelliklerin sırasını koruyarak ve virgüllerle ayrılarak yazılırlar. Her bir olay bir satıra karşılı gelir. Yukarıdaki veride ilk olay `güneşli,85,85, yok, hayır` şekline verilmiştir. Bunun anlamı hava durumu güneşli, sıcaklık 85 F, nem 85%, rüzgar yok ve bunun sonucunda durum oyun oynamaya elverişli değildir.

3.3.2 Programın Birimleri

Kod yedi java class'ından oluşmaktadır.

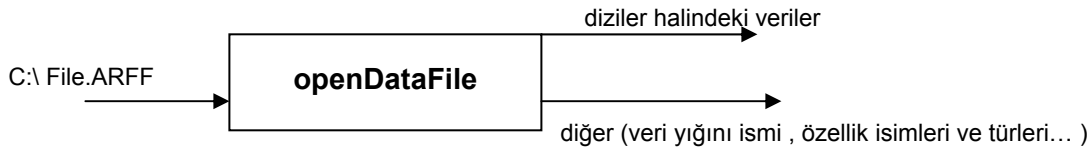
- Node.java
- openDataFile.java
- Painter.java
- Monitor.java
- BuildingDecisionTree.java
- Attribute.java
- Graph.java



Şekil 3.19 - Programın Birimleri ve Çalışma Düzeni

Şekil programın çalışma disiplini iyi bir şekilde ifade etmektedir. Şekildeki her bir kutu söz konusu Java class'larına karşılık gelmektedir. Şekilden de anlaşılacağı gibi openDataFile.class isimli birim Veri.ARFF dosyasındaki bilgileri alıp array tipinden ifade ederek buildingDecisionTree.class isimli koda gönderiyor. buildingDecisionTree.class isimli birim mevcut dizi tipindeki bilgileri attribute.class ve Node.class isimli sınıfları kullanarak ağaç yapısını oluşturuyor. Ardından oluşan ağaç olarak ifade edilmiş veriyi Painter.class birimine gönderiyor. Painter.class ağaç halindeki veriyi grafik olarak ifade eder. Bu aşamadaki Graph.class sınıfı ise bağlantılı listeler olarak ifade edilmiş ağaç yapısının ekrana çizilebilmesi için bu yapıyı sayısal ve metinsel formata dönüştürmektedir. Monitor.class isimli kod ise Painter.class'ı kullanarak kullanıcı ara yüzünü oluşturur. Bu yapıların ayrıntılı incelenmesi daha faydalı olacaktır.

OpenDataFile.class



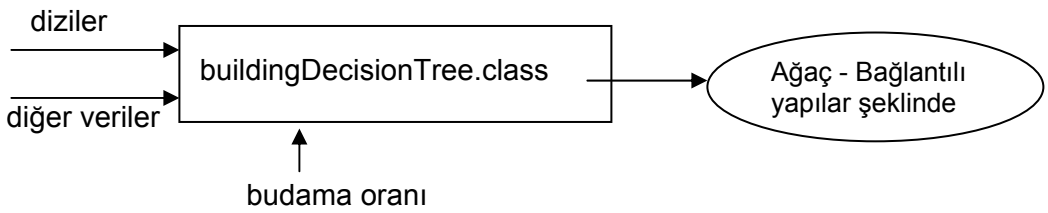
Şekil 3.20 Veri okuma birimi

Bu class adından da anlaşılacağı gibi işletim sisteminde ARFF formatlı dosyaları program seviyesine aktarmaktadır ve Şekil 3.20 bu durum canlandırılmıştır.

new DataInputStream(new FileInputStream(file.ARFF));

satırı herhangi bir dizindeki file.ARFF isimli dosyayla programın ilişki kurmasını sağlıyor. Dönüştürme sırasında ARFF formatının kuralları dikkate alınarak işlemler gerçekleştirilir. Bu sınıf, yarattığı bu veriyi dış dünyaya sunmak için ise getAllDataTable() isimli bir alt metoda sahiptir. Diğer bilgiler için de uygun metotlar vardır. Veriler buildingDecisionTree.class isimli yapıya bu metot aracılığıyla iletilmektedir.

BuildingDecisionTree.class

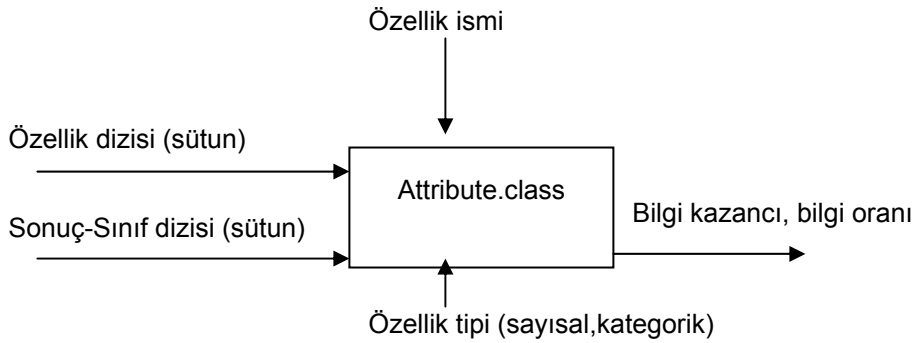


Şekil 3.21 Ağaç oluşturma birimi

Şekil 3.21'de ifade edilen bu sınıf, openDataFile sınıfının biçimlendirdiği dizi halindeki verileri alarak karar ağacını oluşturur. Bu noktada sütunları teker teker ayırarak attribute isimli sınıfa gönderir. Aldığı sonuçlar çerçevesinde böl ve yönet kurallarını uygulayarak ağacı adım adım inşa eder. Bu aşamada kullanılan algoritmalar C4.5 sisteminin daha önceki bölümlerinde sözü edilen yaklaşımlarından yola çıkılarak oluşturulmuştur. KA oluşturmadaki en önemli kriterler bu yapıda gerçekleştirilir. Ağacı oluşturma aşamasında Node.class isimli bağlantılı yapıdan yararlanır. Bu aşama aslında veriler, dizilerden bağlantılı yapılara yani ağaçlara aktarılır. Ağacın budanması ise yine bu sınıf tarafından yapılır. Ağacın hangi oranla budanacağı Monitor.class'tan yani kullanıcıdan

alınan bilgiyle belirlenir. Budama için gereken düğüm ağırlık bilgileri yine bu sınıf tarafından hesaplanır. Tüm işlerini bitirdikten sonra bağlantılı veri yapısı ile ifade edilmiş ağacı painter isimli sınıfa iletir.

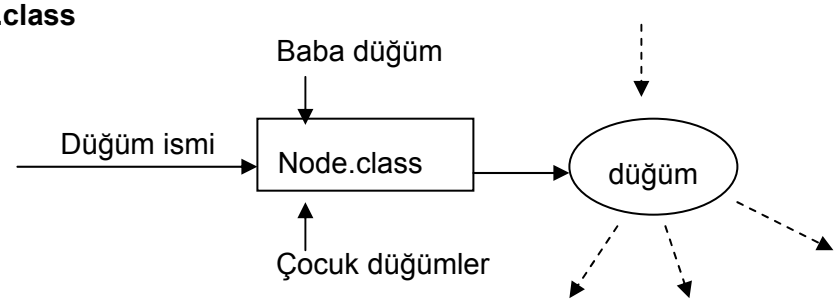
Attribute.class



Şekil 3.22 Özellik değerlerini temsil eden birim

En önemli birimlerden biri olan attibute.class, özelliklerin bilgi kazançlarını hesap eden bir birimdir. Bu birim Şekil 3.22 'de ifade edilmiştir. BuildingDecisionTree isimli yapı, verileri sütunlara ayırdıktan sonra her bir özellik sütunu ile sınıflandırmaya karşılık gelen sütun çiftini diziler şeklinde ifade ederek attribute.class'a iletir. Atribute sınıfı verileri işleyerek o özelliğin bilgi kazancını ve diğer bütün durumlarını hesaplayarak buildingDecisionTree isimli yapıya iletir. Bilgi kazancı, bilgi oranı ve entropi gibi kavramlar bu class içinde kodlanır.

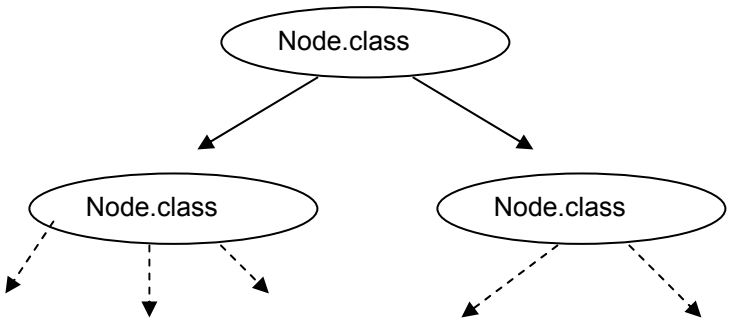
Node.class



Şekil 3.23 Düğümleri temsil eden birim

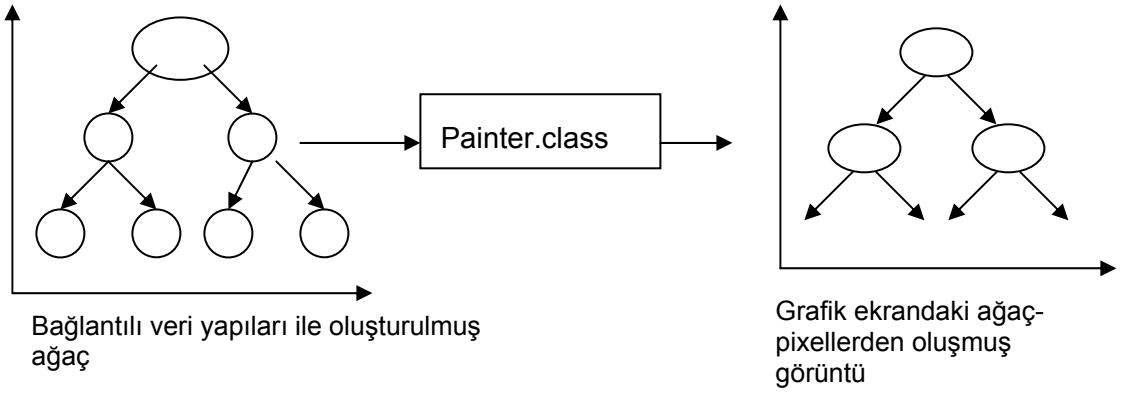
Şekil 3.23'te gösterilen bu yapı buildingDecisionTree sınıfından gelen bilgiler doğrultusunda gerekli düğümleri oluşturur. Bu yapıda bir düğümde gereken tüm bilgiler tutuluyor. Düğümün içerik bilgisi, alt çocuk sayısı, alt çocukların bağlantısı, derinlik bilgileri, ağırlık bilgileri, ilgili düğüme işaret eden dalın içerik bilgisi vb. bir çok bilgi bu

veri tipi tarafından saklanır. Bu yapının diğer bir özelliği ise ağaç üzerinde yapılan bir çok işleme olanak sağlamasıdır. Ağaca düğüm ekleme, derinlik hesaplama, ağırlık hesaplama, yaprak sayısı hesaplama, grafik için gerekli sayısal ve metinsel bilgiler hesaplama, belirli bir orana göre budama vb. bir çok metodu bulunmaktadır. Kısacası bu sınıf bağlantılı nesne yapısı sayesinde aslında ağacın kendisidir. Şekil 3.24'de de canlandırıldığı gibi her düğüm yinelemeli olarak yine kendisi tipinden bir düğüme bağlantı kurar.



Şekil 3.24 Düğümlerin rekürsif olarak birbirlerini göstermesi

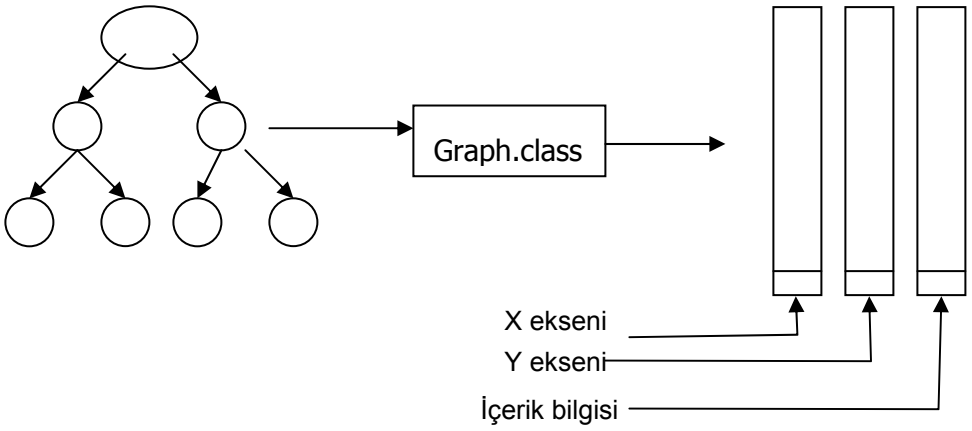
Painter.class



Şekil 3.25 Bilgisayarda grafik oluşturma

Bu yapı buildingDecisionTree.class'dan aldığı Node.class tipindeki ağaç yapısını kullanarak söz konusu ağacı grafiksel olarak görselleştirir. Node.class halindeki veriler önce Graph.class yardımıyla iki boyutlu uzayda ifade edilir. Painter.class sadece ağacı grafiksel olarak ifade eden bir tuvaldir. Bu sınıf Java'daki grafik kütüphanelerini kullanarak görsel bir yapı sunar.

Graph.class



Şekil 3.26 Ağacın veri yapılı halinden x-y koordinat verilerini oluşturma

Bu yapı Şekil 3.26'da canlandırıldığı gibi, Painter.class tarafından gelen ve Node.class tipinde olan ağaç bilgisini dizi haline çevirir. Bu işlemler sırasında her bir düğüm için çeşitli sayısal ve metinsel bilgiler hesap edilir. Düğüm oval olarak iki boyutlu bir uzayda çizilebilmesi için (x,y) koordinatları hesap edilir. Ardından bu oval içine yazılacak metin bilgisi oluşturulur. Dallar için ise (x1,y1) ve (x2,y2) olmak üzere iki nokta hesaplanır. Dal bilgisi içinse yine bir metin bilgisi oluşturulur. Ağaç bir çok düğümden oluştuğu için bu bilgiler diziler şeklinde ifade edilerek çizim yapılır. Graph.class bu diziyi tekrar Painter.class'a gönderir. Painter.class ise bu bilgiler altında ağacı çizer.

Monitor.class

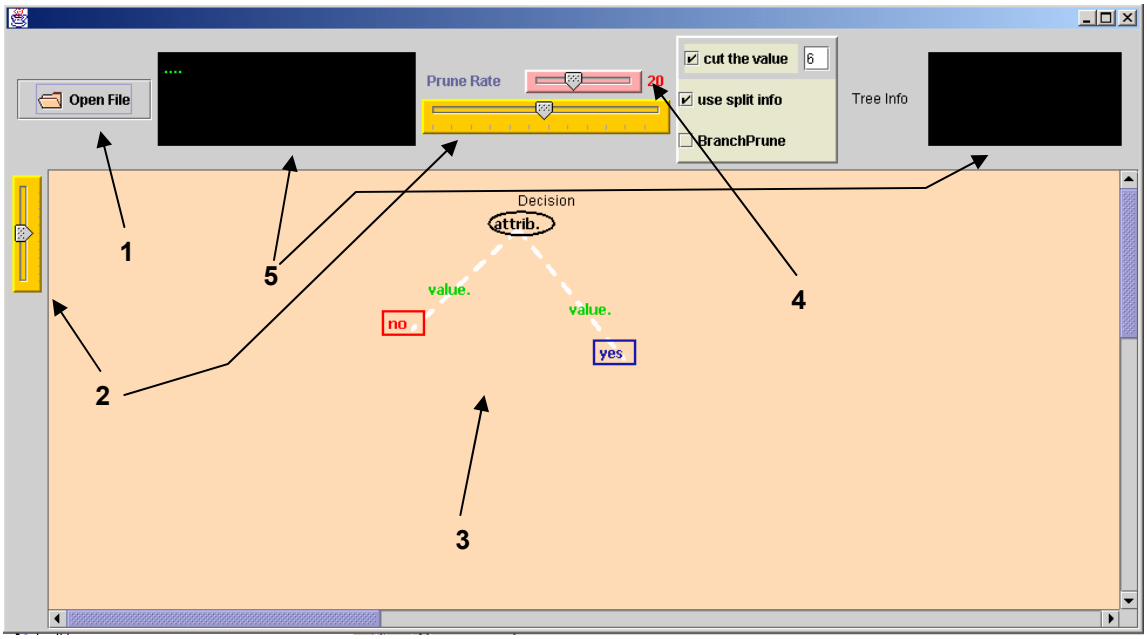
Bu sınıf programı kullanan kullanıcıların karşılaştıkları ara yüzü oluşturan koddur. Bu kod çeşitli ara yüz objelerinden oluşur. Bu ara yüz nesneleri grafiksel ara yüz başlıklı bölümde ayrıntılı olarak anlatılmıştır. Bu ara yüz Painter.class 'tan aldığı çizilmiş ağaç nesnesini içinde barındırır. Kullanıcıya çeşitli işlemler yapabilmesi için gerekli ara yüzleri sağlar. Şekil 3.19'da canlandırıldığı gibi Monitor.class'tan gönderilen komutlarla ağaç çizim işlemi gerçekleştirilir. Bu birim hangi verinin kullanacağını, hangi oranla budanacağını, ve görüntünün boyutlarının ne olacağını belirler. Tüm birimler bu belirlemenin ardından çalışmaya başlar. Bu yapının daha ayrıntılı kısmı bir sonraki bölümde ayrıntılı anlatılacaktır.

3.3.3 Programın Çalıştırılması

Bu sekiz class bir katalogun altına kopyalanır. Örnek olarak C:\c4.5\ dizini ya da /home/someone/C4.5/ altına kopyalanır. Söz konusu sistemde JDK 1.3 ya da üst versiyonu yüklü olması gereklidir. Söz konusu yazılım java.sun.com web adresinden ücretsiz indirilebilmektedir. Ardından komut satırından

java Monitor

komutu çalıştırılır. Görüldüğü gibi Monitor birimi çalıştırılıyor. Bir önceki bölümde ayrıntılı olarak açıklandığı gibi bu birim diğer birimlerle ilişki kurar. Bu komut çalıştığında



Şekil 3.27 Ana program ekranı

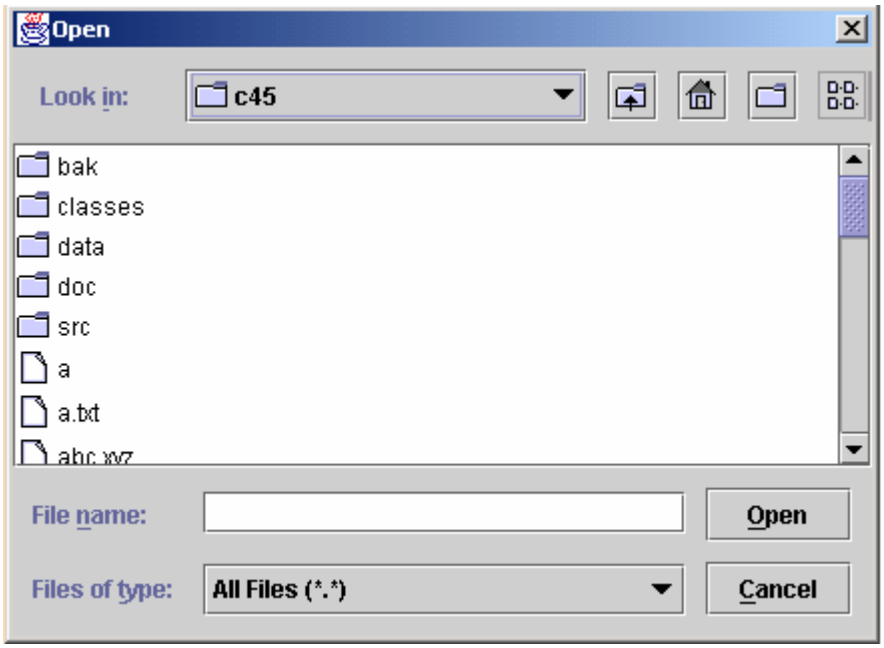
Şekil 3.27'deki ekran açılacaktır. Bu ekranda numaralandırılmış birimlerin görevleri şu şekilde açıklanır:

1. Bu birimde "open file" isimli düğme vardır. Bu düğme ile veri yığınının yeri belirtilir.Bu işlemin hemen ardından ağaç yaratılır.
2. Bu birim kaydırma çubuğundan oluşur. Bu çubuktaki işaretçi kaydırıldığında ağacın görüntüsü y ekseninde genişler ya da küçülür. Bu birim x ekseni için olanı ise Şekil 3.27'de üstte bulunmaktadır.
3. Şekilde gösterilen bu birim ağacın çizimini görüntülemektedir. Bu birim aynı zamanda Painter.class'ın oluşturduğu görüntüdür.

4. Budama oranını belirleyen kaydırma çubuğuna işaret eder. Bu çubuk kaydırılması ile ağacın hangi oranda budanacağı belirlenir. Alacağı en yüksek değer %49'dur.

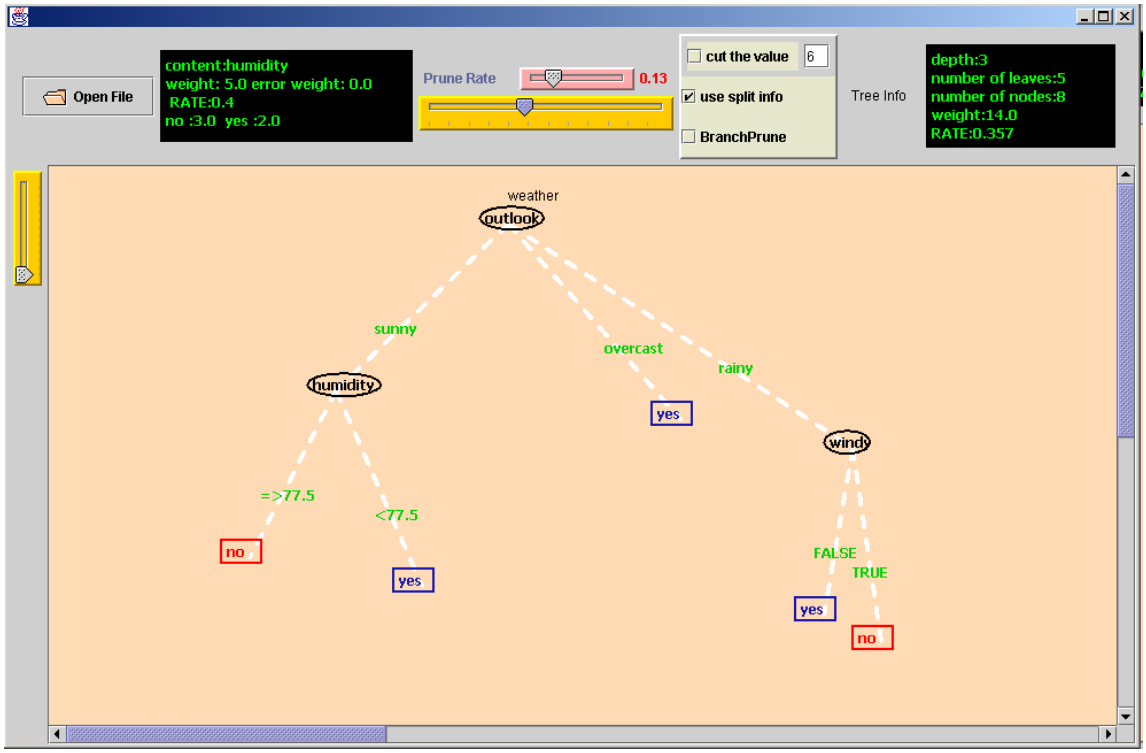
5. Bu birim ağaç ile ilgili diğer tüm bilgileri (ağaç büyüklüğü, düğümdeki hata oranı, düğümün kapsadığı sınıfların sayısal değerleri vb.) görüntüleyen ufak bir monitördür. Bu monitörden bu ekranda iki tane bulunmaktadır.

Bu ekran çıktığında ilk yapılması gereken 1 numaralı (open file) düğme ile veri dosyanın yerini belirtmektir. Bu düğmeye basıldığında şekil 3.28'deki ara yüz ekrana çıkar.



Şekil 3.28 Veri kümesini bulan ara yüz

Bu ara yüz ile verinin yeri belirtilir. Bu işlem sonunda tüm birimler devreye girerek C4.5 yaklaşımları ve yöntemlerine uygun olarak ağacı oluşturur. Bu işlemin ardından ağacın görüntüsünü ayarlamak için x ekseni ve y eksenini genişletip daraltan kaydırma çubukları kullanılır. Budama oranı yine kaydırma çubuğu ile belirtilir. Ağaçtaki düğüm, dal ve yaprak bilgileri ağacın üzerinde yazılıdır. Ancak düğümler hakkında daha ayrıntılı bilgi alabilmek için fare ile düğümün üzerine gitmek yeterli olur. Bu aşamada 5 numara ile gösterilen siyah ekranda düğüm hakkındaki tüm içerik bilgileri görülebilir. Monitörün sağındaki siyah ekranda ise ağacın boyu, derinliği, yaprak sayısı, düğüm sayısı vb. bilgiler görüntülenir.



Şekil 3.29 Programın kullanıcı ara yüz görüntüsü

Şekil3.18'deki ARFF formatlı data, program tarafından yorumlandığında Şekil3.29'daki ekran çıkar.

4. VERİ KÜMELERİ İLE ÇALIŞMA

4.1 Amaç

Bu aşamada amaç program aracılığıyla veri kümeleri üzerinde çalışarak KA oluşturma yaklaşımları için bazı sonuçlar çıkarmaktır. Diğer bir amaç da yazılımın iyi çalışıp çalışmadığını tespit etmektir. Tüm bu çalışmalar ayrıntılı olarak bu bölümde incelenecektir.

4.2 Veri Kümeleri

13 adet veri kümesi kullanılmıştır. Bu veriler "Makine Öğrenmesi" konusunda çalışan araştırmacıların kullandığı veri kümesi havuzundan alınmıştır [4]. Bu havuzda 100'den fazla veri kümesi bulunmaktadır. Ancak tezin incelediği C4.5 algoritması ikili sınıflandırma yaptığı için bu sınıflandırmaya uygun veriler ele alınmıştır. Bu veriler şöyle tanımlanabilir.

Araba Veri tabanı

Marko Bohanec tarafından 1997'de oluşturulmuş bir veri tabanıdır. Arabaların satış fiyatlarına, bakım masraflarına, kapı sayısına, kapasitesine, güvenilirliğine bakılarak bir değerlendirilme yapılır.

Gebelik Önleyici Metotların Tercihi

Bu veritabanı Tjen-Sien Lim tarafından 1997'de Endonezya'da toplanmıştır. Bu örnekler hamile olmayan evli kadınlardan oluşmaktadır. Ana soru şudur: "gebelik önleyici metotların seçiminin kadının sosyoekonomik yapısıyla nasıl bir ilgisi vardır". 1473 adet örnek toplanmıştır. Ele alınan özellikler kadının yaşı, eğitimi, dini, çalışıp çalışmadığı vb. özellikleridir.

Kredi Kartı Kabulü

Bu veri tabanı kredi kartı başvurularının kabulünü içerir. Başvuranların tüm özellikleri ile onay alıp almadığı bir veri tabanında tutulmuştur. Oluşan KA kimlere kredi kartı verildiğini ya da verilmediğini ifade eder.

Gögüs Kanseri

1999'da Tjen-sien Lim tarafından toplanmış bu veri Chicago Üniversitesinde Gögüs Kanseri olan kişilerle yapılan bir araştırmadır.

Kalp Hastalıkları Veri Tabanı

Bu veri tabanında aslında 76 adet özellik vardır. Ancak Makine Öğrenmesi için sadece 14 tanesi kullanılmaktadır.Ana amacı hastalarda kalp hastalığına yakalanma durumunu incelemektir. Bu veri tabanındaki özellikler yaş, cinsiyet, göğüs ağrı tipi, kan basıncı gibidir.

Hepatit Hastalığı

Kalp Hastalığına benzer bir şekilde Hepatit virüsünün hastada olup olmadığını hedefleyen bir veri kümesidir. G.Gong (Carnegie-Mellon Üniversitesi) tarafından oluşturulmuştur. 20 özellik ve 157 olay ele alınmıştır.

İşçi Sözleşmeleri

Bu veri kümesi çalışanlarla yapılan iş sözleşmelerini ele almaktadır. KA çalışanın iş anlaşmasını kabul edip etmediğini sınıflandırmaktadır. Bu veri tabanı İşçilere teklif edilen ilk yıl zammı, tatil süresi, prim miktarı, sosyal olanaklar, sigorta içeriği gibi özelliklerden oluşmaktadır.

Mantar

Bu veri tabanı mantarların zehirli olup olmadığını incelemektedir. Mantarın boyu, rengi kokusu, şapka şekli, tadı gibi özellikler vardır. Yaklaşık 8000 adet olay gözlemlenmiştir.

Tic-Tac-Toe

David W. Aha tarafından oluşturulan bu veri tabanı tic-tac-toe oyununu ele alır. X oyuncusunun oyunu kazanıp kazanmadığı incelenmektedir. 3x3 lük bir oyun için 9 adet özellik gerektiğinden 9 adet özellik oluşturulmuştur. Bu özelliklerin değerleri ise X,O ya da b (boş)'dir. Tüm bu olasılıklar ile X'in kazanma durumu ele alınmıştır.

Birleşmiş Milletler Kongre Oylaması

Bu veri tabanı1987'te Jeff Schlimmer tarafından oluşturulmuştur. Kongre üyelerinin kullandıkları oylara göre Cumhuriyetçi mi yoksa Demokrat mı olduğunu sınayan bir veri kümesidir. 435 olay ve 16 özelliğe sahiptir.

Avusturalya Kredi Kabulü

Quinlan tarafından toplanmış bu veri tabanı kredi kartlarıyla ilgilidir.

Almanya Kredi Kabulü

Dr. Hans Hofmann tarafından oluşturulmuş bu veri kredi vermede oluşan olayları içermektedir.

Postoperatif Bakım

Hastaların durumuna bakarak postoperatif bakıma ihtiyaçları olup olmadığını belirten bir veri kümesidir.

4.3 Sonuçlar

Bu verilerle önce budama yapılamaksızın KA oluşturulmuştur. Oluşan KA için veri kümesinin olay sayısı, baskın sınıfın genel toplama oranı, düğüm sayısı, yaprak sayısı, ağacın derinliği bilgileri kaydedilmiştir.

Tablo 4.1 - Veri kümelerinden elde edilen karar ağaçlarının durumu

Veriler	Olay sayısı	Baskın sınıfın oranı (%)	Düğüm sayısı	Yaprak sayısı	derinlik
Araba	1128	77	181	126	7
Gebelik Önleyici Metotların Tercihi	1473	58	1060	589	21
Kredi Kartı Kabulü	490	56	133	77	13
Göğüs Kanseri	306	74	187	94	17
Kalp Hastalığı	270	66	91	52	13
Hepatit Hastalığı	155	80	81	41	10
İşçi Sözleşmeleri	57	65	57	30	7
Mantar	8416	54	26	20	7
Tic-Tac-Tao	958	66	322	200	8
Birleşmiş Milletler Kongre Oylaması	300	62	50	29	9
Avusturya Kredi Kartı Kabulü	687	56	190	104	15
Almanya Kredi Kabulü	1000	70	462	286	16
Postoperatif Bakım	90	63	75	44	9

Tablo 4.1‘de tüm sonuçlar gösterilmiştir. Bu tablodan ilginç bir çıkarım yapmak söz konusu değildir. Yapılacak en basit yorum kullanılan veri kümesi büyüdükçe ağacın düğüm ve yaprak sayısı artmaktadır. Bunun iki sebebi olabilir. Birincisi veri çoğaldıkça içindeki gürültü yaratan olayların olasılığı artmasıdır. Bu durum budama yapılmadığı için

gözlenmektedir. Ancak budama ile bu sorunun üstesinden gelinmektedir. İkincisi ise özellikler ile sınıf arasında çok sıkı bir ilişkinin bulunamamasıdır. Mantar örneğinde bu durum tersinedir. Çünkü mantar veri kümesinde koku özelliği ile sınıf arasında çok sıkı bir ilişki vardır. Bu ilişki sayesinde mantar kümesi tüm kümelerden daha fazla olaya sahip olmasına karşın orantısal olarak onlardan düşüktür. Bu veriler oluşturulurken budama oranı 0 olarak kabul edildi. Budama oranının artması ile ağaç büyüklüğünün nasıl değiştiği Tablo 4.2'te gösterilmiştir.

Tablo 4.2 - Budama oranının artması ile ağacın verdiği tepki

Veriler	%0 budama	%5 budama	%10 budama	%15 budama	%20 budama	%25 budama	%30 budama
Araba	181	167	121	114	114	1	1
Gebelik Önleyici Metotların Tercihi	1060	1034	929	768	623	498	176
Kredi Kartı Kabulü	133	88	60	50	3	3	3
Göğüs Kanseri	187	177	147	105	39	27	1
Kalp Hastalığı	91	83	83	48	43	8	8
Hepatit Hastalığı	81	55	33	11	11	1	1
İşçi Sözleşmeleri	57	45	29	9	3	3	3
Mantar	26	10	10	10	10	10	10
Tic-Tac-Tao	322	322	317	236	178	118	97
Birleşmiş Milletler Kongre Oylaması	50	35	10	10	10	10	10
Avusturya Kredi Kartı Kabulü	190	123	95	91	41	3	3
Almanya Kredi Kabulü	462	434	359	248	216	164	133
Postoperatif Bakım	75	72	72	66	43	39	7

Budamanın artması ile ağaç küçülmektedir. Ancak önemli olan ağacın uygun bir budama oranı ile kesilmesidir. Her ağacın kendine özgü bir budama oranı olabilir. Örneğin Kalp Hastalığı verisi %20 ile budandığında 43 düğüm içermekte, %25 ile budandığında 8 düğüm içermektedir. 8 düğüm karar ağaçları için oldukça iyi bir sonuçtur. Buna benzer olarak işçi sözleşmeleri de %15 ile %20 budama arasında 9 düğümden 3 düğüme düşmektedir. KA için uygun bir budama oranı bu tablodan bakılarak çıkarılırsa %20 ile %25 arası bir değer çıkacaktır. Çünkü tablodaki düğüm sayılarının keskin bir inişi genellikle bu aralıkta medya gelmektedir. Ancak eğer veride çok tutarlı bir kural gizliyse ve konunun kurallarını özetleyebiliyorsa bu budama oranı çok düşük seviyelerde bile iyi sonuç verecektir. Örnek olarak Mantar kümesi 8000 olayı içermesine karşın hiç budama yapılmamasına karşın 26 düğüm oluşturmaktadır. Ancak her veri kümesi böyle sonuçlar vermemektedir. Bu bakımdan budama yapılmadan verilerdeki gizlenmiş kuralları çıkarmak ya da sadeleştirmek mümkün değildir.

Bu verilerle yapılan bir araştırma ise sağlama işlemidir. Bu konu bölüm 2.3'te ayrıntılı olarak değinildiği gibi veri kümesi iki eşit parçaya bölünür ve bu parçalardan biri ile KA oluşturulur diğeri ile ağacın tutarlılığı test edilir. Bu sağlama ile oluşan KA'nın ne kadar güvenilir olduğu tespit edilir. Ancak her veri kümesi yetersiz olan veriler ile bu işlemleri yapmak mümkün olmaz.

Gebelik önleyici metotların tercihi, Almanya'da kredi kabulü, göğüs kanseri, kalp hastalığı, postoperatif bakım veri kümeleri için sağlama işlemi yapılmıştır. Tablo 4.3 'da verilerin eğitim kümesinden oluşan karar ağaçlarının yine bu veriler ile test edilmesinin sonuçları gösterilmektedir.

Tablo 4.3 - Eğitim kümesi ile test edilen karar ağaçlarının budama oranına göre doğruluk sonuçları (%).

Budama oranı / Veri kümesi	%0	%4	%8	%12	%16	%20	%24	%28
Gebelik Önleyici	100	98	97	96	93	92	89	82
Almanya Kredi Kartları	100	99	97	94	93	93	90	88
Göğüs kanseri	98	98	98	96	91	84	84	72
Kalp Hastalığı	100	100	98	95	90	90	78	78
Postoperatif Bakım	100	99	96	95	93	84	84	78

Bu test KA'nı oluşturan küme üzerinde yapıldığından doğruluk sonuçları %0'lık budama için %100'e çok yakındır. Aslında bütün karar ağaçlarının %100 sonucu verilmesi beklenir. Çünkü her karar ağacı kendisini oluşturan veri kümesini dikkate alarak kurallarını oluşturur. Ancak eğer veri kümesinde kayıp veriler mevcut olursa o ağacın yarattığı varsayımlar bir hataya yol açar. İşte bu hata %0'lık bir budama için yapılan sağlamayı etkilemektedir. Budama oranı arttıkça ağacın güvenilirliği azalmaktadır. Bölüm2.3'te de bahsedildiği gibi budama arttıkça eğitim kümesine göre hata oranı artar.

Bu aşamada yanlışlıkla budamanın zararlı olduğu söylenebilir. Ancak KA'nın hedefi yeni olayları doğru sınıflandırmasıdır. Zaten sınıflandırılmış olaylar hedef alınmadığından ortaya çıkacak bu hata oranını yeni veriler ile belirlemek gerekecektir. Bu sebeple yapılan tüm testleri test veri kümesi ya da sağlama kümesi ile yapmak gerekir. Tablo 4.4 bu kümelerden alınmış ve KA oluşturma için kullanılmamış veriler ile yapılan test sonuçlarını göstermektedir.

Tablo 4.4 - Karar Ağaçlarının test kümesi ile test edilmesinin budama oranına göre sonuçları

Budama oranı / Veri kümesi	%0	%4	%8	%12	%16	%20	%24	%28
Gebelik Önleyici	65	65	66	66	67	67	70	71
Almanya Kredi Kartları	64	65	67	68	68	68	68	68
Göğüs kanseri	58	58	58	67	70	70	70	79
Kalp Hastalığı	75	75	75	76	78	78	78	76
Postoperatif Bakım	69	70	73	73	72	73	73	73

Bu tablo 4.4'de göze ilk çarpan sonuç budama oranı arttıkça doğruluk ya da denge oranının artmasıdır. Artışın sebebi daha öncede belirtildiği gibi overfitting'in yok edilmesidir. Yani ağaç meydana gelen gürültü ve yanlış verilerin varlığı ağacın gereksiz büyümesine yol açmaktadır. Budamanın temel amacıda zaten bu gürültüyü engellemektir.

Tablo 4.5 - Tüm verilerdeki doğruluk oranlarının ortalaması

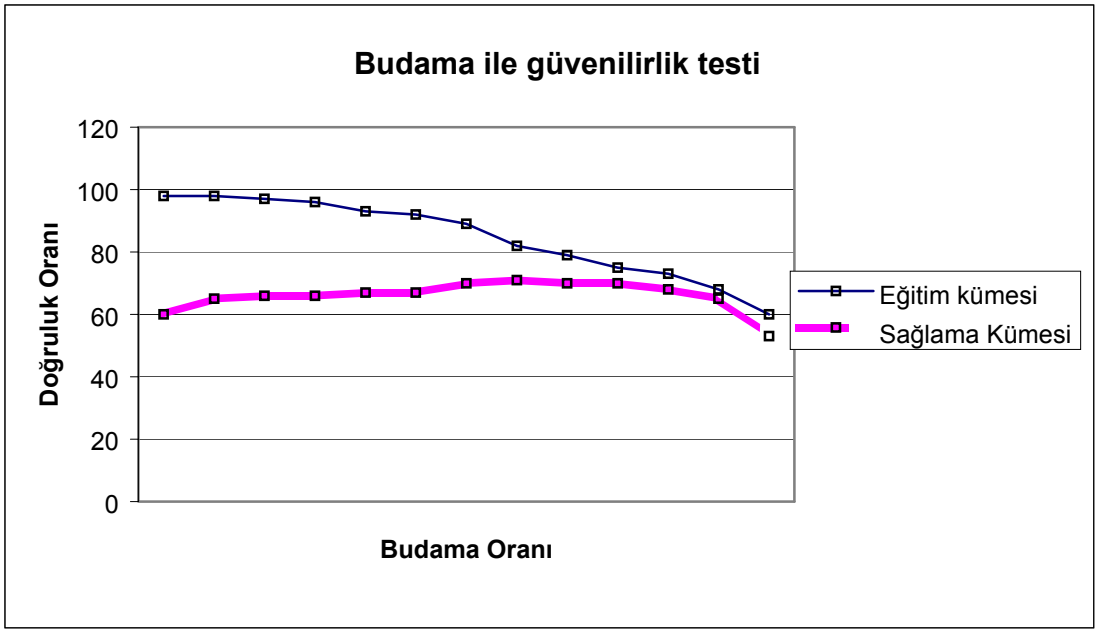
Budama oranı	%0	%4	%8	%12	%16	%20	%24	%28
Eğitim Kümesi	99.6	98.8	97.2	95.2	92	88.6	85	79.6
Sağlama Kümesi	66.2	66.6	67.8	70	71	71.2	71.8	73.4

Tablo 4.3'daki sonuçlar ile Tablo 4.4'deki sonuçların ortalaması alındığında Tablo 4.5'deki sonuçlar elde edilir. Bu sonuçlardan çıkarılacak en iyi sonuç şu olur: Budama oranı arttıkça KA'nın sağlama kümesine karşı verdiği doğruluk oranı artarken eğitim kümesine karşı gösterdiği oran düşer. Önemli olan sağlama verisi olduğu için budama işleminin çok yararlı olduğu sonucu çıkarmamız olasıdır. %35-40 gibi bir budama yapmak çok zarar verebilir. Çünkü bu çalışmada ele alınan sonuçlar %28'e varan budamalarla ilgilidir. Budama oranını %0-%48 aralığında değiştirip *"Gebelik Önleyici Metotların Tercihi"* veri kümesi test edildiğinde Tablo 4.6 elde edilir

Tablo 4.6 – Eğitim kümesi ile sağlama kümesinin budama oranlarına verdiği doğruluk oranları (%).

Gebelik Önleyici Metotların Tercihi	%0	%4	%8	%12	%16	%20	%24	%28	%32	%36	%40	%44	%48
Eğitim kümesi	98	98	97	96	93	92	89	82	79	75	65	62	55
Sağlama Kümesi	65	65	66	66	67	67	70	71	70	70	68	68	64

%28'lik bir budamadan sonra sağlama kümesi için KA güvenilirliğini azaltmaktadır. Şekil 3.30'a bu durum canlandırılmıştır.



Şekil 4.1 Budama ve doğruluk testi

Bu aşamada dikkat edilmesi gereken bir nokta Şekil 4.1'da ki sonucun Şekil 2.14 ile tutarlı olmasıdır. Şekil 2.14'te ifade edilen durumda düğüm sayısı arttıkça eğitim kümesi ile yapılan doğruluk artmakta ve sağlama kümesi için ise bir noktaya kadar artmakta ve ardından azalmaktadır. Şekil 4.1'daki grafik, budama oranı düğüm sayısı ile ters orantılı olduğundan Şekil 2.14 teki grafiği desteklemektedir.

5. SONUÇ VE TARTIŞMA

Bu tez kapsamında karar ağacı oluşturma yaklaşımları ele alınmış ve bu yaklaşımlardan ID3 ile onun takipçisi C4.5 incelenmiştir. C4.5 sisteminin getirdiği yaklaşımlar ele alınarak ağaç yaratma, budama, sağlama vb. algoritmalar detaylı olarak incelenmiş ve buna yönelik olarak Java tabanlı bir program yazılmıştır. Bu program sayesinde karar ağacı algoritmalarının tüm evreleri gözlenerek bazı sonuçlar elde edilmiştir.

Grafik ara yüze sahip bu program, oluşan karar ağacını grafiksel olarak ifade edebildiğinden ağacı gözlemleme daha rahat olabilmiştir. Çünkü ağaç yapısının görselliği çok önemlidir. Oluşan ağaçların metin tabanlı ifade edilmesi bir çok özelliğini gözden kaçırabilmektedir ve önemli bazı detayları önemsiz gibi gösterebilmektedir. Oysa ki oluşan ağacın iki boyutlu görsel bir ifadesinden daha iyi sonuçlar çıkarılabilmektedir. Bu amaçla tez kapsamında grafik çizme algoritmaları ve grafik kütüphaneleri kullanılarak karar ağaçları görselleştirilmiştir.

ID3 algoritmasının sadece kategorik verilere cevap vermesi çok büyük bir eksikliktir. Çünkü hemen hemen bir çok veri kümesi sayısal verileri içermektedir. Bu anlamda sadece kategorik verilerle işlem yapabilen bir algoritma çok fazla engel teşkil etmektedir. Bu sebepten C4.5'in sayısal verilere getirdiği bu yaklaşım çok yerinde olmuştur. Buna ek sayısal verilerle ilgili bir iyileştirme Bölüm 2.4'de ifade edilmiştir. Bu bölümde kısaca sayısal özelliklerin, iki bölgeye değil de daha çok bölgeye ayrılması ile daha iyi sonuçlar alınabileceği gözlenmiştir. Çünkü veri kümesine gizlenmiş kuralların ikiden fazla bölgeye dağılma ihtimali vardır. Bütün bu ihtimaller hesap edilmiş olur.

Bölünme bilgisi (split-information) kavramı C4.5'i ID3'den ayıran ikinci bir başlık olmuştur. Çünkü hasta kayıt numaralarına göre hastalık teşhisine karar veren bir yapının zararları engellenmektedir. Ancak bölünme bilgisi bunu engellese de, denemeler sonucunda görülmüştür ki, bu yapı bazı veri kümeleri için bunu gerçekleyememektedir. Bunun için veri kümesini yaratırken bu tip yararsız ve bilgi kazancı yüksek özelliklerin baştan elenmesi gibi bir çözüm geliştirmek gerekmektedir. Ya da yararsız özellikleri teşhis eden algoritma oluşturulmalıdır. Hasta etiket numaralarında her bir numara yegane olduğundan bu tip bir özelliği tespit eden bir algoritma rahatlıkla oluşturulabilir. Buna paralel olarak farklı karakterlere sahip yararsız özellik grupları için özgün bir teşhis algoritması oluşturulmalıdır.

C4.5 sisteminin ID3'e ek olarak getirdiği diğer bir önemli başlık ise kayıp verilerdir. Kayıp verilere ilişkin getirilen bu yaklaşım sonuçları ile yararlı görülmektedir. Ancak kayıp verilerin fazla olması ağaçtaki tutarsızlığı artırmaktadır.

Karar ağacı sağlama ile ilgili olarak bir çok sonuç Bölüm 4'te tablolarla ifade edilmiştir. Bu sonuçlardan en önemlisi karar ağacı ile onu oluşturan eğitim kümesi ve sağlama kümesi arasındaki ilişkidir. Ağaç karmaşıklaştıkça eğitim kümesi için doğruluğu artmakta, ancak sağlama kümesi için ise doğruluğu azalmaktadır. Önemli olan sağlama kümesinin doğrulanması olduğundan, ağacın karmaşıklaşması yerine daha da sadeleşmesi ve belli bir budama oranı ile budanması daha yerinde olacaktır. C4.5 sistemi budama ile ilgili yaklaşımları sunmuştur. Ancak sistemin kendi içinde en iyi budama oranını tespit edecek bir yapıdan bahsetmemiştir. Böyle bir yapının eksikliğini gidermek için şöyle bir yöntem izlenebilir: Ağacın doğruluğunu maksimum yapan budama oranını tespit eden bir modül sisteme eklenir ve bu modül bu oranı tespit ettikten sonra ağacı o oranla budar. Eğer ağacın doğruluğu yerine basitliği ve anlaşılırlığı önemsenirse bu durumda ağaç düğüm sayısı göz önüne alınarak budama işlemi gerçekleştirilir. Her ağaç için belli bir düğüm sayısı eşik değer olarak seçilebilir. Bu eşik değerinin altında kalana kadar ağaç budanır. Ancak bu yöntem ağacın doğruyu ifade etmesi bakımından yararlı olmayabilir. Ağacın insanlar için anlaşılırlığını her zaman için ön planda tutma çoğu zaman önemli değildir. Sonuç olarak ağaç insan için ne kadar karmaşık olursa olsun bilgisayarlar için her zaman anlaşılırdır.

Deneyler sonucu ağacın büyüklüğü ve doğruluğu göz önüne alındığında en iyi budama oranı aralığı %20-%25 çıkmıştır. Ancak her veri kümesinin kendine özgü bir budama oranı vardır ki, ağaç bu oranla budandığında karmaşık bir yapıdan sade bir hale bürünür. Bu tip ani dönüşüm ağaçların kırılma noktaları olmaktadır ve bu tip noktaları iyi değerlendirmek gerekmektedir. Ağaç budama oranını tespit eden modülün bakması gereken bir nokta bu olmalıdır. Çünkü ağacın ani değişimi kurallara yönelik ip uçları saklar. Ağacı sağlama işlemindeki en önemli engel ise yeterli sayıda olayın bulunmamasıdır. 60 olaylık bir veri kümesi ile hem karar ağacı eğitip hem de o ağacı test etmek çok verimsiz olacaktır. Bu açıdan veri kümesinin yeterli olaya sahip olması gereklidir.

Ağaç budama yöntemlerinden biri olan dal kesme ise dikkat edilmesi gereken bir başlıktır. Bu işlemin sisteme getireceği yarar yapının daha sade görünmesi olacaktır.

Çünkü bu budama işleminin alt ağaç yok etme işlemindeki gibi sağlama kümesinin doğruluğunu artırmamaktadır.

Oluşan ağaçların sağlama kümesini doğrulamasının dışındaki bir diğer doğruluk kriteri ise oluşan dalların kapsadığı olay sayısıdır. Her dalın sahip olduğu olay sayısını belli bir seviyenin üstünde tutmak gerekecektir. Bazı veri kümeleriyle oluşturulan karar ağaçlarında yapraklar arasında kapsadıkları olaylar bakımından bir dengesizlik olduğu görülmüştür. Bu dengesizlik ağacın güvenilirliğini ise azaltan bir unsur olmaktadır. Eğer bir yaprak birkaç tane olayı kapsıyorsa o kurala çok fazla güvenilmemelidir. Bu bakımdan her yaprak ortaya koymuş olduğu kuralın altını kalın kalın çizmek durumundadır. 100 olayı kapsayan bir yaprak ile 3 olayı kapsayan yaprağın işlevi farklı olacaktır.

Sonuç olarak ele alınan konular şöyle sıralanabilir: Görsel bir ağaç yaratma, kategorik ve sayısal veriler, kayıp veriler, bölünme bilgisi, ağacın doğruluğu ve ağaç budama. Bu başlıklar ayrıntılı olarak ele alınmış, ID3'nin ve C4.5'in yaklaşımları irdelenmiş, yorumlar yapılmış ve bazı iyileştirmeler sunulmuştur. C4.5 sisteminin genel yapısına bakıldığında öğrenme süreci için çok önemli bir mekanizma olduğu görülmüştür. Bu sistem faydalandığı bilgi teorisi ile veri kümelerinin arasındaki gizlenmiş kuralları açığa çıkarmaktadır. Ancak bu sistemde sadece ikili bir sınıflandırılma yapılmaktadır. Veri kümesinin sınıfları çoklu ya da sayısal olması durumunda bu sistemin bir yararı olmayacaktır. Buna yönelik olarak başka yöntemler mevcuttur. Çoklu sınıflandırma için CART sistemi, sayısal sonuçlar için yapay sinir ağları kullanılmaktadır. Ancak yine de C4.5 sistemi ikili bir sınıflandırma için çok güçlüdür. Bu noktada bahsedilen dezavantajlara karşılık olarak karmaşık karar sisteminde tek başına işe yaramadığı durumlarda bir modül olarak işlev kazanabilir.

KAYNAKLAR

[1] **Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J.,** 1984. Classification and Regression Trees, Wadsworth: Belmont, CA.

[2] **Di Battista, G., Eades, P., Tamassia, R., Tollis, İ. G.,** 1999. Graph Drawing, Prentice-Hall, Inc.,New Jersey.

[3] **Kocabaş, Ş.**,1991. A Review of learning. The Knowledge Engineering Review, Vol. 6. No.3, 195-222.

[4] **Merz, C. J., Murphy, P. M.,** 1998. UCI Repository Of Machine Learning Databases, ftp://ftp.ics.uci.edu/pub/machine-learning-databases.

[5] **Mitchell, T. M.,** 1997. Machine Learning, WCB/McGraw-Hill Series, NewYork.

[6] **Niemeyer, P., Knudsen, J.,** 2000. Learning Java, O'Reilly, Canada.

[7] **Pohl, I., McDowell, C.,** 1999. Java By Dissection, Addison-Wesley, California.

[8] **Quinlan, J. R.**, 1993. C4.5 : Programs For Machine Learning. San Francisco: Morgan Kaufman Publisher.

[9] **Quinlan, J. R.,** 1986. Induction Of Decision Trees. Machine Learning 1(1):81-106.

[10] **Russell, S., Norvig, P.,** 1995. Artificial Intelligence: A Modern Approach. Prentice Hall,Inc.New Jersey.

[11] **Witten, I. H., Frank, E.,** 2000. Data Mining, Morgan Kaufmann Publishers, San Francisco.

ÖZGEÇMİŞ

Savaş YILDIRIM 02/12/1975 tarihinde Erzincan'da doğdu. 1993 yılında İTÜ Kontrol ve Bilgisayar Mühendisliği Bölümünde lisans eğitimine başladı ve 1998'de bu bölümden mezun oldu. Aynı yıl İTÜ Fen Bilimleri Enstitüsünde bilgi işlem sorumlusu olarak çalışmaya başlayan YILDIRIM, 1999-2001 yılları arasında da İTÜ Proje Yönetim Merkezinde yazılım mühendisi olarak görev yaptı. YILDIRIM, 2001 yılından beri İstanbul Bilgi Üniversitesi Bilgisayar Bilimleri Bölümünde araştırma görevlisi olarak çalışmalarına devam etmektedir.