KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİNDE 1R ALGORİTMASI VE İKİNCİ KURALIN (2R) OLUŞTURULMASI

YÜKSEK LİSANS TEZİ

Tuğba DALYAN

Anabilim Dalı: Bilgisayar Mühendisliği
Tez Danışmanı: Yrd. Doç. Dr. Nevcihan DURU

KOCAELİ, 2006

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİNDE 1R ALGORİTMASI VE İKİNCİ KURALIN (2R) OLUŞTURULMASI

YÜKSEK LİSANS TEZİ

Tuğba DALYAN

Tezin Enstitüye Verildiği Tarih:

Tezin Savunulduğu Tarih:

Tez Danışmanı	Üye	Üye
Yrd.Doç.Dr. Nevcihan DURU	Prof. Dr. Oya KALIPSIZ	Prof. Dr. Kadir ERKAN
(.....)	(.....)	(.....)

KOCAELİ, 2006

MAKİNE ÖĞRENMESİNDE 1R ALGORİTMASI VE İKİNCİ KURALIN (2R) OLUŞTURULMASI

Tuğba DALYAN

Anahtar Kelimeler: Makine Öğrenmesi, 1R, 2R.

Özet: Bu çalışmada, makine öğrenmesinde kullanılan 1R algoritması örnek bir veri kümesi üzerinden detaylı olarak anlatılmıştır. Teorisi anlatılan algoritma Python programlama dili ile geliştirilmiştir. Programın işleyişi ve sayısal verilerin kategorik verilere nasıl dönüştürüldüğü, aynı veri kümesi kullanılarak açıklanmıştır. Projenin gelişim sürecinde, 1R algoritmasının arama uzayı genişletilmiş ve ikinci kuralın (2R) oluşturulması sağlanmıştır. 2R algoritması, üretilen ilk kuralı kullanarak, kuraldaki nitelik dışında kalan nitelikler arasındaki en yüksek frekansa sahip nitelik değerini seçip, ilk kurala ekler. Yedi farklı veri kümesi üzerinde test edilen algoritmaların sonuçlarına tezde ayrıca yer verilmiştir. Oluşan kurallar ve hata oranlarını kullanıcıya sunmak için basit bir arayüz hazırlanmıştır.

1R ALGORITHM IN MACHINE LEARNING AND FORMING SECOND RULE (2R)

Tuğba DALYAN

Keywords : Machine Learning, 1R, 2R.

Abstract: In this study, 1R algorithm that is used in machine learning, is explained with using a dataset in details. Algorithm, which is explained theoretically, is implemented by Python programming language. Progress of program and discretization process is clarified with same dataset. Search space of 1R is extended with some development and second rule (2R) has improved during the thesis process. 2R algorithm uses rules that is derived from 1R, choose most accurate value of attribute that remains in attribute list. Seven commonly used datasets in machine learning research is selected to test algorithms and results of tests are presented. A very simple GUI was designed to see the derived rules and error rates more explanatory.

ÖNSÖZ ve TEŞEKKÜR

Son yıllarda bilişim teknolojilerinin ve veri toplama araçlarının gelişmesi ile verileri oluşturma, toplama ve verilere ulaşma süreçleri hızlanmaktadır. Verilerden tahmin ve tanımlar çıkarma işi ise her geçen gün biraz daha önem kazanmaktadır. Makine öğrenmesi ise bu noktada devreye giren bir alan olup, birçok tekniği ile araştırmacıların özellikle son yıllarda üzerinde durduğu önemli bir araştırma konusudur.

Bu tez akademik yönü kadar bir uygulama projesi olma özelliğini de bünyesinde barındırmaktadır. 1993 yılında Robert Holte tarafından geliştirilen 1R algoritması, Python programlama dilinde uygulanmış ve üretilen kurallar kullanılarak ikinci kuralın oluşturulması sağlanmıştır.

Yapılan bu çalışmanın ülkemizde makine öğrenmesi konusunda ve gelecekte de farklı projelerle insanoğlunun hizmetine sunulmasında katkıda bulunmasını dilerim.

Hayatım boyunca edindiğim değer yargılarımı ve eğitimimi borçlu olduğum aileme, başta tez danışmanım Yrd.Doç.Dr. Nevcihan Duru olmak üzere tüm hocalarıma, yüksek lisans eğitimim ve tez çalışmam boyunca desteğini esirgemeyen nişanlıma teşekkürü bir borç bilirim.

İÇİNDEKİLER

SİMGELER

Kısaltmalar

1R : One Rule
2R : Two Rules
1RD : 1R Discretization
V.M. : Veri Madenciliği
V.T.B.K : Veritabanlarından Bilgi Keşfi

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

1. GİRİŞ

Öğrenme, psikoloji, bilişsel bilim, bilgisayar bilimleri gibi alanların üzerinde durduğu bir araştırma konusudur. Sözlük anlamıyla öğrenme; deneyim, çalışma ya da öğretim ile oluşan davranışlardaki kalıcı, ölçülebilir ve belirlenmiş değişiklik sayesinde bilgi, yetenek, tavır ve değer kazanma sürecidir [1]. Kısaca öğrenme, deneyimler vasıtasıya davranışlarda oluşan değişiklik sürecidir.

Öğrenme, birçok bilim adamı tarafından farklı şekilde tanımlanmıştır. 1983'te Simon, benzer görevleri tekrarlarken, bir sistemdeki daha etkili olabilecek herhangi bir değişimi öğrenme olarak tanımlarken, 1985'te Minsky, öğrenmeyi, aklımızda yaptığımız yararlı değişiklikler olarak ifade etmiştir. 1986'da Michalski, deneyimlerin gösterim biçimlerini oluşturma ve bunları değiştirme şeklinde öğrenmeyi açıklamıştır [2]. 1997'de Mitchell, bir bilgisayar programının, deneyimler sayesinde, bazı görevlerde performansın geliştirilmesi ile öğrenebileceğini kitabında belirtmiştir [3].

Öğrenme özellikle makine öğrenmesi, bilgisayar bilimlerinde karşımıza çıkan yapay zeka alanının ilgilendiği bir araştırma konusudur. Yapay zeka, insanın düşünme yapısını ve fonksiyonlarını anlamak ve bunun benzerini ortaya çıkarmak amacı taşımaktadır. Yapay zeka, bilgi edinme, algılama, görme, düşünme, tasarlama, bellekte tutma, problem çözme ve karar verme gibi insan zekasına özgü yeteneklerin insan eliyle yapılan yapay ortamlarca uygulanmasıdır [4]. Günümüzde çok yönlü araştırmalar ile önemini sürdürmekte olan yapay zeka ve makine öğrenmesinin gelişimi 1940'lara dayanmaktadır.

İngiliz matematikçi ve bilgisayar bilimci Alan Mathison Turing'in, 1943'te ikinci dünya savaşı sırasında Kripto Analiz gereksinimleri ile ürettiği Elektro-Mekanik cihazlar sayesinde Bilgisayar Bilimi ve Yapay Zeka kavramları doğmuştur. Şifre çözme amacı ile başlatılan çalışmalar, Turing'in prensiplerini oluşturduğu bilgisayar

prototipleri olan Heath Robinson, bombe ve Colossus bilgisayarları, Boolean cebirine dayanan veri işleme mantığı, Makine Zekası kavramının oluşmasına sebep olmuştur [5].

Alan Mathison Turing'in 1950 yılında yayınladığı Makine ve Zekanın Hesaplanması isimli eserinde “Makineler Düşünebilir mi?” sorusu ve yine aynı yıllarda John von Neumann'ın Oyun Teorisi, makine öğrenmesinin başlangıcı olarak görülmektedir.

1942'de Warren McCulloch ve Walter Pitts'in, ilk hücre modelini geliştirmeleri yapay zeka konusundaki ilk çalışmalardan biri olarak sayılmaktadır. İki bilim adamı, yapay sinir hücrelerini kullanan ve önermeler mantığı, fizyoloji ve Turing'in hesaplama kuramına dayanan hesaplama modelini inşa etmişlerdir. Herhangi bir hesaplanabilir fonksiyonun sinir hücrelerinden oluşan ağlarla hesaplanabileceğini ve mantıksal “ve” ve “veya” işlemlerinin gerçekleştirilebileceğini göstermişlerdir. Bu ağ yapılarının uygun şekilde tanımlanmaları halinde, öğrenme becerisi kazanabileceğini de ileri sürmüşlerdir [5]. Hebb, 1949 yılında hücre bağlantılarını ayarlamak için ilk öğrenme kuralını önermiş ve öğrenebilen yapay sinir ağları için önemli bir adım atılmıştır.

1950'lerde Turing'in ilk satranç programını ve Turing Test programını sunması, bilgisayarların, insan zekası ile yarışabilecek yetenekleri kazanabilmesi için programlanabileceğini göstermiştir. Yine 1950 yılında, Bilgi Teorisinin mimarı olan Claude E. Shannon, makalesinde [6] satranç oynama yeteneğine sahip olacak modern, genel amaçlı bir bilgisayarın programlanma ya da hesaplama yordamındaki problemleri detaylı şekilde açıklamıştır.

1951 yılında, ilk yapay sinir ağı temelli bilgisayar SNARC, MIT'de Minsky ve Edmonds tarafından yapılmıştır. Çalışmalarını Princeton Üniversitesi'nde sürdüren McCarthy, Minsky, Shannon ve Rochester'le birlikte 1956 yılında Dartmouth'da iki aylık bir açık çalışma düzenlemişlerdir. Bu toplantıda bir çok çalışmanın temelleri atılmakla birlikte, toplantının en önemli özelliği McCarthy tarafından önerilen yapay zeka adının konmasıdır [5].

Yapay zeka alanına katkıda bulunan diğer önemli bilim adamları ise Mantık Kuramcıları olarak anılan Allen Newell, Herbert A. Simon ve Cliff Shaw'tır. 1955 yılında, gerçekleştirmeyi planladıkları satranç makinesi çalışması, mantık çalışmasına dönüşmüş ve ilk kuram ispatlayan Mantık Teori Makinesi'ni 1956 yılında gerçekleştirmişler [7]. 1957'de Newell, Shaw ve Simon, insan gibi düşünme yaklaşımına göre üretilmiş ilk program olan Genel Sorun Çözücü'yü (General Problem Solver) geliştirmişlerdir. 1958 yılında geliştirdikleri, NSS isimli satranç programı, yapay zekada atılmış önemli bir adımdır. Byron Spice'ın Pittsburgh Post-Gazette'deki 02 Ocak 2006 tarihli yazısında, 1955-56'larda yapay zeka ismi kimse tarafından anılmıyorken, Allen Newell ve Herbert A. Simon tarafından bahsedildiğini yazmıştır [8]. Aynı yıllarda Arthur Samuel, dama oyun programı geliştirerek yapay zekanın gelişimine katkıda bulunmuştur [9].

1961'de James Slagle, 1958'de John McCarthy'ın bulduğu Lisp diliyle, hesap problemlerini çözen SAINT (symbolic automatic INTegrator) programını geliştirmiştir. Slagle, SAINT ile ilgili yazdığı makalesinde programını, birinci sınıf kolej öğrencilerinin hesaplama problemlerini çözebileceği basit sembolik bütünleme olarak tanımlamış ve programın performansını açıklamıştır [10].

60'ların başlarında Simon ve Newell, fiziksel simge sistemi varsayımını ortaya atmışlar ve bu varsayım, bilim adamlarının Yapay zekaya yaklaşımlarında iki farklı akımı ortaya çıkarmıştır: Sembolik ve Sibernetik Yaklaşım.

Sembolik yaklaşımda, sunumlar bazı özel yollar kullanılarak düzenlenmiş semboller içerir ve toplam bilginin sunumu, semboller ve bu sembollerin bileşimleri şeklindedir [11]. Sibernetik yaklaşım ise, yazılım ve donanımda organik sistemlerin çalışma yöntemlerini bire bir benzetimlendirmektir. Yapay sinir ağları çalışmaları da kısmen bu gruba girmektedir [12].

Michalski ve Wnek, 1992 yılında yayınlamış oldukları makalede, sembolik ve sibernetik öğrenmeyi deneylere dayanarak karşılaştırmıştır. Makalede, iki yaklaşım arasındaki fark açıklanırken, sembolik ve sibernetik öğrenme yaklaşımlarındaki en önemli farkın, bilgiyi sunumundaki bilişsel görüş olduğu belirtilmiş, mantık tabanlı

kuralların, karar ağaçlarının idrak etmek için kolay ve insan bilgisine yakın olduğu vurgulanmış, bu durumun yapay sinir ağları gibi sınıflandırıcı sistemler için aynı olmadığından bahsedilmiştir. [13].

Sibernetik yapay zeka yaklaşımlarından yapay sinir ağları konusunda da aynı dönemde gelişmeler olmuştur. 1958'de Rosenblatt'ın, yapay sinir ağlarının gelişiminin başlangıcı olarak kabul edilen McCulloch ve Pitts'in ilk hücre modelini geliştirmesine ve Hebb'in hücre bağlantılarını ayarlamak için ilk öğrenme kuralını önermesine ek olarak, algılayıcı (perceptron) yapısını ve öğrenme kuralını geliştirmesi, günümüzde kullanılan kuralların temelini oluşturmuştur [14].

1963 yılında, M.I.T'den Thomas Evans, IQ testlerindeki soruları çözebilen Analogy isimli programı yazmıştır. Edward A. Feigenbaum ve Julian Feldman yapay zeka konusunda yayınlanmış bazı makaleleri birleştirerek “Bilgisayarlar ve Düşünce" isimli kitabı [15] yayınlamışlardır. Kitapta, yukarıda bahsedilen çalışmaların mimarları olan Turing, Minksy, Simon, Newell, Slagle ve Samuel'e ek olarak birkaç bilim adamının da makalesi yer almaktadır. 1964 yılında, Danny Bobrow'un M.I.T'de yaptığı araştırma sonuçları, bilgisayarların basit matematik problemlerini çözebilecek derecede doğal dili anlayabildiklerini göstermiş, 1965'te Joseph Weizenbaum, İngilizce herhangi bir konu ile ilgili sohbet edebilen, etkileşimli program Eliza’yı yaratmıştır [16].

İlk uzman sistem olarak görülen DENDRAL projesi, 1967'de Bruce Buchanan, Edward Feigenbaum ve Joshua Lederberg tarafından Stanford Üniversite'sinde başlamıştır. DENDRAL, moleküler yapı analizini şekillendirmek için kimya alanında kullanılmıştır. 70'lerin başında, Meta-DENDRAL olarak geliştirilen öğrenme programı, otomatik olarak DENDRAL için kurallar üreten bir program halini almıştır [17].

1968'da Marvin Minsky ve Seymour Papert, Rosenblatt'ın geliştirdiği algılayıcı model üzerinde araştırmalar yaparak, modelin XOR gibi karmaşık mantık fonksiyonları için kullanılmayacağını ispatlamışlardır. Bunun üzerine yapay sinir

ağları konusu karanlık döneme girmiş, yapay zeka çalışmaları tekrar sembolik yaklaşım üzerine yoğunlaşmaya başlamıştır.

60'lı yılların sonlarında, Roger Schank, günümüzde yapay zeka ve dil bilimi alt kategorisi olan doğal dil işleme çalışmalarının temelini atmış ve doğal dil anlama için kavramsal bağlılık modelini tanımlamıştır. Makalesinde [18], yaptığı çalışmanın insan aklını anlamaya çalışmak olduğundan bahsetmiştir. Bunu gerçekleştirirken de, bilgisayar üzerinde insan aklını modellemek ve öğrenme, hafıza ve doğal dil işleme gibi konularla ilgilenmiştir.

60'lardaki diğer bir gelişme ise, John Holland'ın genetik algoritmalar ile ilgili çalışmasıdır. Genetik algoritmalar, Darwin'in evrim teorisinden esinlenmiştir ve problemleri çözerken evrim sürecini kullanır. Bu algoritma, makine öğrenmesine farklı bir yaklaşım sunmuştur [19]. 1966 yılında Hunt, Martin ve Stone tarafından geliştirilen Kavram Öğrenme Sistemi ile günümüzde kullanılan karar ağaçları algoritmasının temelleri atılmıştır [20]. Michalski'nin, 1969'da geliştirdiği AQ algoritması, örneklerden genel kurallar çıkaran bir algoritma olup, birçok yeni algoritmanın gelişmesine öncülük etmiştir [21]. Aynı yıl içerisinde, ilk uluslararası yapay zeka konferansı (IJCAI) önemli bir adım olmuştur [22].

70'li yıllar daha çok sembolik yaklaşımın ilerleme gösterip, yükselişe geçtiği yıllar olmuştur. O dönem, makine öğrenmesinin rönesans dönemi olarak tanımlanmış ve birçok bilim adamı, farklı strateji ve çalışmalarla yapay zeka ve makine öğrenmesi alanına katkıda bulunmuştur [23].

Patrick Winston'ın 1972'de geliştirdiği Arch programı da kavram öğrenmesinin örneklerindendir. Makalesinde, bir durum ile karşılaştığımızı ve benzer bir durum ortaya çıktığında, karşılaştırma ile idrak edebileceğimizi ve öğrenebileceğimizi belirtmiş, örnekleri baz alarak, karmaşık objelerin tanımlarını üretebileceğini göstermiş ve kavram öğrenmesinin gelişmesine katkıda bulunmuştur [24].

1974 yılında, MYCIN, Ted Shortliffe tarafından geliştirilmiştir. Kural tabanlı uzman sistem, enfeksiyonları teşhis edip, önerilerde bulunan, if-then kurallarına dayalı

olarak sunulmuştur. Mark Musen makalesinde [25], MYCIN'ı, klinik karar-destek sisteminin gelişmesindeki ilk kural tabanlı yaklaşım olarak açıklamıştır.

1978'de, Tom Mitchell'in tanımladığı Versiyon Uzayı (Version Space), bir dizi örnekten elde edilen bilginin hiyerarşik bir yapıda gösterimini, kullanılan örneklerden bağımsız bir şekilde sağlar. Tanımladığı model, versiyon uzayında çeşitli modelleri yöneterek kavram öğrenmesini gerçekleştirmiştir [3,26]. Quinlan'ın, 1979'da geliştirdiği ID3 karar ağacı algoritması, günümüzde en sık kullanılan karar ağacı algoritmasıdır. Hunt'ın geliştirdiği karar ağacı algoritması üzerine bazı yenilikler eklenerek geliştirilmiştir [20].

70'lerde durgunluk dönemi yaşayıp bunun yanında üzerinde çalışmalara devam edilen yapay sinir ağları konusu 80'lerde atağa geçmiştir. 1982 yılında Hopfield, yapay sinir ağlarının birçok problemi çözebilecek kabiliyeti olduğunu göstermiş ve aynı yıllarda Kohonen öz düzenlemeli haritayı (self-organizing map) tanımlamıştır. 1974 yılında Werbos tarafından geliştirilip, 1986 yılında Parker, Rumelhart ve McClelland tarafından tekrar meydana çıkarılan geri yayınım ağı (Back Propagation Network) ile yapay sinir ağları konusu önemini sürdürmektedir [14,27].

80'lerde Kass tarafından, CHAID ağaç sınıflandırma modeli geliştirilmiştir. Aynı dönemde C&RT ağaç algoritması, Breiman, Friedman, Olshen ve Stone tarafından sunulmuştur. C&RT, hem devamlı hem de kategorik veriler için tasarlanmıştır. [28,29]. Breiman ve Friedman kitabında [30], C&RT ile ilgili detaylı bilgi vermiştir. 80'lerde makine öğrenmesi için diğer bir gelişme ise karar listeleri yaklaşımıdır. 1987'deki Rivest'in karar listesi yaklaşımı, düzenli kurallar listesi şeklindedir ve her kural koşul ve sonuç içermektedir. 1989'da Clark ve Niblett'in geliştirdiği CN2 algoritması, karar listelerinin genişletilmiş hali olarak görülmektedir. Eğitim verilerini kullanarak if-then şeklinde kurallar üretir. Bunu yaparken, en iyi kural bulunur ve bu kuralı kapsayan örnekler eğitim verisinden atılır. İşlem iyi kurallar bulunmayana dek tekrarlanır [31,32]. Günümüzde bu, Kapsayan Algoritmalar olarak geçmektedir.

Bunu izleyen yıllarda, makine öğrenmesi alanında önemli adımlar atılmış, daha önceki yıllarda ortaya atılan teoriler kullanılarak, makine öğrenmesi alanında birçok yeni algoritma geliştirilmiş, paradigmalar sunulmuştur.

Olasılıksal bir yaklaşım olan ve uzun yıllar desen tanıma da kullanılan Bayesian tekniği, 90'lı yıllarda makine öğrenmesi ile ilgilenenler tarafından çokça kullanılmıştır [33]. 1993 yılında, Holte'nin gerçekleştirdiği 1R algoritması, basit yapısı ile makine öğrenmesi alanında yer almıştır. Basit yapısına rağmen, standart veri kümeleri üzerinde yapılan çalışmalarda verdiği sonuçlara göre şaşılacak kadar etkili bir algoritmadır [34]. Holte yazdığı makalesinde [35], makine öğrenmesinde sıkça kullanılan veri kümeleri üzerinde algoritmayı denemiş, Quinlan'ın geliştirdiği C4.5 algoritması ile karşılaştırmış ve performansı ile ilgili sonuçları yazmıştır. 1R algoritması, tezin üçüncü bölümünde detaylı olarak açıklanacaktır. Agrawal tarafından 1994 yılında geliştirilen Apriori birliktelik kuralları algoritması ile günümüzde özellikle satış-pazarlama sektöründe kullanılan yeni bir yaklaşım doğmuştur [36]. Yukarıda sayılan yaklaşımların yanında, istatiksel modeller, kümeleme algoritmaları ve örnek-tabanlı algoritmalar da, makine öğrenmesi alanında sıkça başvurulan yöntemlerdir.

Günümüzde ise, Internet'in yaygınlaşması ve teknolojinin gelişmesi ile makine öğrenmesi çalışmaları, geniş kitlelere ulaşmakta, bir çok bilim dalının ve mesleğin ilgi alanına girmektedir. Varolan algoritmaların daha iyi sonuç vermesi için geliştirilen teknikler, indüktif mantık programları, veri madenciliği, bilgi erişim kavramaları ile çok yönlü olarak önemini sürdürmektedir.

Tez kapsamında, makine öğrenmesinde kullanılan 1R algoritması incelenmiştir. 1R algoritması 1993 yılında Robert Holte tarafından bulunmuştur. Algoritma, basit bir okadar da etkilidir. 1R algoritması, bir niteliğe göre kurallar üretir ve bir seviyeli karar ağacı olarak tanımlanır.

Holte makalesinde, kural üreten 1R algoritması ile karar ağaçlarında kullanan C4.5 algoritmasını, veri kümeleri üzerinde deneysel çalışmalar yaparak karşılaştırmış ve bazı sonuçlara varmıştır. Temel sonuç ise 1R ile oluşan kuralların doğruluğunun,

C4.5 ile oluşan budanmış karar ağaçlarından çıkan kuralların doğruluk oranından çok az düşük olmasıdır [35].

Diğer bir sonuç ise basit kural öğrenme sistemlerinin, karışık kurallar üreten sistemlere alternatif olarak gösterilmesidir. Çünkü karışık kurallar üreten sistemlerin ek olarak karmaşıklığını da doğrulamak gerekmektedir. Bu yüzden Holte, makalesinde [35] yaptığı gözlemlerle, geleneksel yöntemlerden farklı olarak yeni bir araştırma yöntemine doğru yönelmiştir. Makine öğrenmesi araştırmalarının bir amacının, üretilen kuralların basitlik ve doğruluğunu arttırmak olduğu düşünülürse, bu amaç ile oluşturulan "İlk Basitlik" yöntemi, Holte'nin makalesinde [35] vurgulanmıştır. Bu yöntemde, bir-iki niteliği sınıflandırabilecek durumlarda karmaşık varsayımlara gerek yoktur. Çünkü yapılacak arama, küçük hipotezleri içeren küçük alanlarda yapılacaktır.

Holte, aynı makalesinde [35], 1R üzerinde yapılan değişikliklerin, algoritmanın gelişmesini sağlayabileceğinden bahsetmiştir. [34]'te ise, 1R üzerinde yapılan değişiklikler ve değişikliklerin yarattığı sonuçların oldukça yararlı olduğu anlatılmıştır. Bunun ile birlikte makalede [35], birçok temel değişiklik ile 1R'ın arama uzayı genişletip ve daha karmaşık kurallar üretilebileceği vurgulanmıştır. İki nitelik içeren kurallar bunlardan bir tanesidir [35].

Bu çalışma için, 1R algoritması Python programa dilinde geliştirilmiş ve algoritmanın işleyişi örnek bir veri kümesi üzerinden adım adım anlatılmıştır. Tezin sonraki bölümlerinde ise 1R algoritmasının arama uzayı genişletilmiş ve iki kural (2R) üreten bir algoritma geliştirilmiştir.

2R algoritması, üretilen ilk kuralı kullanarak, kalan diğer nitelikler arasındaki en yüksek frekansa sahip diğer bir nitelik değerini seçip, ilk kurala ekler. 2R algoritması da aynı veri kümesi kullanılarak açıklanmıştır. Oluşan kurallar ve hata oranlarını kullanıcıya sunmak için basit bir arayüz hazırlanmıştır. İki algoritma, yedi farklı veri kümesi üzerinde test edilmiş, sonuçları sunulmuştur.

Tez, Giriş bölümü birinci bölüm olmak üzere, 7 bölümden oluşmaktadır. Giriş bölümünde yapay zeka ve makine öğrenmesinin gelişiminden bahsedilmiştir. Bilim adamlarının geliştirdiği uygulamalar ve teknikler anlatılmıştır.

İkinci bölümde, makine öğrenmesi tanımı ve amaçları, bilim adamlarının görüşleri üzerinden anlatılmıştır. Makine öğrenmesinin, veri madenciliği ile olan ilişkisinden bahsedilerek, makine öğrenmesi paradigmaları ve bu paradigmalar çerçevesinde kullanılan teknikler detaylı bir şekilde açıklanmıştır. Son kısmında ise, uygulama alanları hakkında kısa bilgi verilmiştir.

Üçüncü bölümde, makine öğrenmesi alanında kullanılan 1R algoritması detaylı olarak anlatılmıştır. Literatürde sıkça karşılaşılan veri kümesi üzerinden, 1R algoritmasının işleyişine yer verilmiştir. Örnek veri kümesinin sayısal veriler içerdiğinde kategorik verilere nasıl dönüştürüldüğünden ve kayıp verilerle karşılaştığında nasıl bir çözüm sunulacağından bahsedilmiştir.

Tezin dördüncü bölümünde, 1R algoritmasının, Python programlama dili ile nasıl gerçekleştirildiği, sayısal verileri kategorik verilere çevirirken nasıl bir yol kullanıldığı açıklanmış, programın işleyişi, ikinci bölümde kullanılan veri kümesi üzerinden adım adım anlatılmıştır.

Beşinci bölümde ise, 1R algoritması kullanılarak, 2R algoritmasının nasıl oluşturulduğu anlatılmıştır. 2R sonucu ortaya çıkan kurallar ve hata oranlarından bahsedilmiştir. Bunun yanında, programın kullanıcıya sunumu için Java programlama dili ile oluşturulan basit bir arayüzden bahsedilmiştir. Program sonucunda ortaya çıkan kurallar ve hata oranları, arayüz sayesinde sunulmuştur.
Altıncı bölümde, yedi farklı veri kümesi üzerinde uygulanan iki algoritmanın hata oranları incelenmiştir.

Tezin son bölümü olan sonuç bölümünde ise, bu aşamada ele alınmamış detaylara ve çalışma sonuçlarına yer verilmiş, ilerde yapılacak çalışmalar için önerilerde bulunulmuştur.

2. MAKİNE ÖĞRENMESİ

2.1. Makine Öğrenmesi ve Amaçları

Günümüzde ilerleyen teknolojiler ve geliştirilen uygulamalar, veri toplama ve verilere ulaşma süreçlerini kolaylaştırmıştır. Verilerin toplanıp, büyük veri yığınlarının oluşturulma sürecinin yanında, bu yığınlardan işeyarar tahminlerde ya da tanımlamalarda bulunmak birçok alan için önemli bir konu haline gelmiştir.

Makine öğrenmesi konusu, bu verilerin nasıl toplandığı ya da kullanılacak formata nasıl dönüştürüldüğü ile ilgilenmez. İlgilendiği, veriler üzerinde geleceğe yönelik tahminleri ya da tanımları en iyi şekilde nasıl çıkarabileceğidir.

Makine öğrenmesi, adından da anlaşılacağı gibi öğrenme kavramı ile ilgilidir. Öğrenme, davranışlarda deneyimlere bağlı gerçekleşen değişikliklerin oluşma sürecidir ve bu süreç içerisinde bilgi artışı ve yeteneklerde gelişme görülür. Öğrenme, yapay zekanın ilgilendiği önemli bir araştırma alanıdır. Yapay zeka alanının bir kolu da makinenin öğrenmesi konusuna uzanmaktadır. 1940'larda İngiliz matematikçi ve bilgisayar bilimci Alan Mathison Turing ile başlayan yapay zeka ve makine öğrenmesi kavramı, günümüzde de bilim adamlarının geliştirdiği, teoriler, uygulamalar ve algoritmalar ile önemini sürdürmektedir.

Makinenin öğrenmesi, programda ya da verilerde bir değişiklik yapıldığında performansın gelişmesi ile olur. Yapılan değişiklik makinenin öğrenmesini sağlar. Oliver G. Selfridge, “programa nasıl hata bulacağını ve hatayı nasıl düzelteceğini öğretirsen, bir daha aynı durum ile karşılaştığında daha önceki öğrendiklerinden faydalanarak hatasını bulur ve düzeltir” diyerek açıklık getirmiştir [37].

Makine öğrenmesi tanım olarak, örnek verileri kullanarak ya da geçmişteki deneyimlerden yararlanarak, performansı en iyi hale getirecek şekilde, bilgisayarı

programlamak olarak tanımlanmıştır. Bunun için tanımlanan model, gelecek ile ilgili tahminde bulunmak için tahmin edici, eldeki verilerin tanımlanması için tanımlayıcı ya da her ikisinin birlikte kullanılması içindir [38]. Yapılan başka bir tanımda ise, makine öğrenmesinin, veri kümeleri içindeki desen ya da kuralları çıkarmaya yarayan bir teknoloji olduğu belirtilmektedir [39].

Makine öğrenmesinde, tümevarım kullanılarak çıkarımlar yapılmaktadır. Yapılan çıkarımlar, geleceğe yönelik tahminde bulunmak ya da bir tanım yapmak için kullanılır. Veri, geleceğe yönelik bilgi tahmini için kullanılacaksa, yani [38]'de tanımlandığı gibi oluşturulacak model tahmin edici model ise, iki aşama gereklidir: Eğitim Aşaması ve Test Aşaması.

Eğitim aşamasında, belirli miktarda veri kullanılarak bir model oluşturulur. Kullanılan veri, eğitim verisi olarak adlandırılır. Eğitim verisinin ne kadar ve nasıl seçileceği ayrı bir konudur. Oluşturulan model, sadece örnek veriyi değil tüm veriyi temsil eder. Test aşamasında ise, eğitim aşaması sonucunda oluşan modele, test aşaması için ayrılan ya da ileride toplanacak olan veriler sunulur. Ortaya çıkan bilgi ise tahmin etmek için kullanılır.

Makine öğrenmesinin amaçlarını Mitchell, Michalski ve Carbonell makalelerinde üç açıdan incelemişlerdir [40]:

1) Hedef-tabanlı çalışmalar: Öğrenme sisteminin gelişimi ve analizi, belirlenmiş görevleri yerine getirmek için gerçekleştirilir. Bu yaklaşım, makine öğrenmesine mühendis yaklaşımı olarak tanımlanmıştır.
2) Bilişsel simülasyon: İnsanın öğrenme sürecini araştırıp, bilgisayarda simulasyonunu gerçekleştirmek olarak tanımlanmıştır. Bu ise, makine öğrenmesine bilişsel modelleme yaklaşımıdır.
3) Teorik analiz: Uygulama alanlarından bağımsız olarak teorik olabilecek öğrenme metodları ve algoritmaları incelemek içindir.

Makine öğrenmesi konusu ile uğraşan bilim adamları bu üç yaklaşımdan birini ya da her birini birbiri ile bağlantılı şekilde kullanarak çalışmalarını sürdürmektedir.

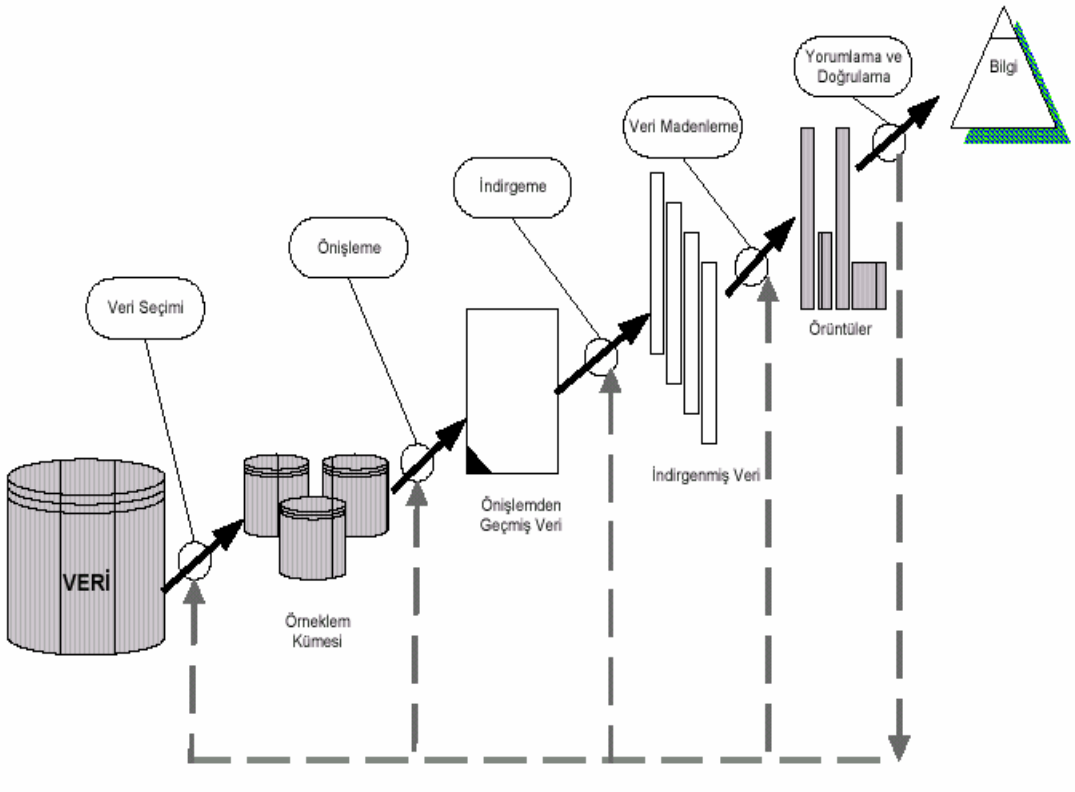
Sunulan üç yaklaşımdan ilki olan mühendis yaklaşımından yola çıkarak, Nilsson makine öğrenmesinin önemini sıralamıştır [41]. Bunlardan bazıları:

- Bazı görevler için, girdi/çıktıyı belirlesek de arasındaki ilişkiyi belirtilemeyebilir. Bu gibi durumda, makinenin kendi iç yapısını ayarlayarak, büyük veri yığınlarından giriş/çıkış fonksiyonunu bulup, arasındaki ilişkiyi tahmin etmesi beklenir.
- Büyük veri yığını içinde gizli kalmış önemli ilişkiler ve bağlantılar olabilir. Makine öğrenmesi metotları, bu ilişkileri seçip çıkarmak için kullanılır. Bu konu, veri madenciliği olarak adlandırılmaktadır. Bu konuya, Bölüm 2.2.'de değinilecektir.
- Bazı görevler için bilgi miktarı, insanın kodlaması için fazla olabilir. Bu gibi durumda, makine, insanın yapabileceğinden fazlasını yapabilir.
- Makine, değişikliklere kolayca adapte olabilir.
- Bazı görevlerde bilgiler değişebilir. Bu gibi durumlarda, yapay zeka sistemini tekrar tasarlamak pratik değildir. Makine öğrenmesi metodlarını kullanarak bu gibi değişiklikler gözlenebilir.

2.2. Makine Öğrenmesinin Veri Madenciliği ile İlişkisi

Veri yığınından yararlı ve değerli bilgileri keşfetme süreci için son birkaç yıldır Veritabanlarından Bilgi Keşfi (V.T.B.K) ifadesi kullanılsa da, bu terim yerine, bilgi çıkarma, bilgi keşfi, veri analizi araştırması, bilgi hasatlanması, desen tanıma gibi terimler kullanılmıştır. Aslında V.T.B.K. terimi, 1989'da yapılan ilk Veritabanlarında Bilgi Keşfi Atelye Çalışması'nda vurgulanmış ve veri yığınlarından bilgi edinmek için birçok adımdan oluşan süreç olarak tanımlanmıştır [42]. Veritabanlarından Bilgi Keşfi (V.T.B.K) ile Veri Madenciliği (V.M.) iki farklı kavram olsa da, günümüzde genellikle birbirlerinin yerine kullanılan terimler haline gelmiştir. V.T.B.K., birçok adımdan oluşan bir süreçtir. Bu sürecin girdileri veriler, çıktısı ise kullanıcı tarafından istenilen yararlı bilgidir. V.T.B.K. 5 adımdan oluşan bir süreç olarak tanımlanmıştır [43]. V.T.B.K. süreci Şekil 2.1'de açıklanmıştır.

1. Veri Seçme: V.T.B.K. sürecinin ilk adımı olarak veritabanından, ya da bir kaynaktan veriler elde edilir.
2. Ön İşleme: Veriler yanlış ya da eksik olabilir. Bunun yanında birçok kaynaktan gelebileceği için farklı tipte olabilir. Hatalı veriler düzeltilir ya da atılır, eksik veriler ise tedarik edilir ya da tahmin edilir.
3. İndirgeme: Farklı kaynaklardan gelen verilerin işlenmesi için genel bir formata çevrilir.
4. Veri Madenciliği: Algoritmaların kullanılması ile bilgi ya da desen elde edilir.
5. Yorumlama/Doğrulama: Veri madenciliği adımı sonucunda çıkan bilgi ya da deseni anlaşılır bir şekilde kullanıcıya sunmaktır. Bunun için görsel metotlar ya da GUI stratejileri kullanılır.



Şekil 2.1: VTBK süreci

Veri Madenciliği, Şekil 2.1'de görüldüğü üzere V.T.B.K sürecinin adımlarından bir tanesidir. Veri madenciliği de kendi başına bir süreç olarak tanımlanmaktadır. Veri Madenciliği sürecinin adımları ise şu şekilde tanımlanmıştır [45] :

1. Analiz yapmak için verileri bir araya getirmek
2. Veri madenciliği programına verileri sunmak
3. Sonuçları yorumlamak
4. Sonuçları yeni problem ya da durumlar için uygulamak

Bölüm 2.1'de makine öğrenmesinin veri madenciliği ile olan ilişkisinden bahsedilirken, makine öğrenmesi metotlarının, büyük veri yığını içinde gizli kalmış önemli bilgi ve bağlantıları ortaya çıkarırken kullanıldığı açıklanmıştır. Yukarıda belirtilen veri madenciliği sürecinden de anlaşılacağı üzere, makine öğrenmesinin uygun bir metodunu kullanmak sürecin ikinci adımı olup, bu süreç içerisinde yer almaktadır.

Makine öğrenmesi, veri madenciliği ile çok sık kullanılan bir konu olsa da, makine öğrenmesi bir veritabanı problemi değil, yapay zeka alanının bir parçasıdır. İki konu arasındaki en büyük fark; makine öğrenmesi, öğrenme metodlarını geliştirerek, tahminleri ya da tanımları en iyi şekilde, yüksek performans ile nasıl çıkarabileceği ile ilgilenirken, veri madenciliğinin ortaya çıkan bilgi ile ilgilenmesidir.

Frawley, Piatetsky-Shapiro ve Matheus, iki konu arasındaki farkları 1992'de yayınladıkları makalede belirtmişlerdir [46]. Veritabanı dinamiktir, eksik, hatalı veri içerebilir ve makine öğrenmesine göre veri sayısı daha çoktur. Aşağıdaki Tablo 2.1, [46]'da belirtildiği gibi makine öğrenmesi ve veritabanı yönetimi arasındaki farklardan bazılarını içermektedir.

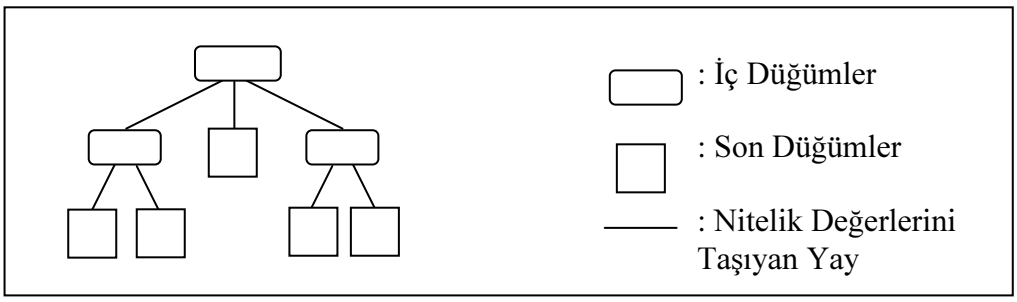
Tablo 2.1: Veritabanı yönetimi ve makine öğrenmesi arasındaki farklar

Veritabanı Yönetimi	Makine Öğrenmesi
Veritabanı aktif, gelişen varlıklar içerir	Veritabanı statik verilerin toplamıdır
Kayıtlar hatalı ya da kayıp olabilir	Veriler tam olmalıdır
Veritabanı milyonlarca kayıt içerebilir	Veritabanı yüzlerce örnek içerebilir
Yapay zeka gerçeklik elde etmeli	Veritabanları çözülmüş bir problemdir, o yüzden ilginç değildir

2.3. Makine Öğrenmesi Teknikleri

Makine öğrenmesi paradigmaları ve teknikleri ile ilgili birçok yazı yazılmıştır. 1995 yılında, Langley ve Simon, yayınladıkları makalede beş adet paradigmadan bahsetmişlerdir. Bu paradigmalar, kural çıkarımı, örnek-tabanlı ya da durum-tabanlı, yapay sinir ağları, genetik, ve analitik öğrenme algoritmalarıdır [47].

İlk paradigma, kural çıkarım algoritmalarıdır. Karar ağaçları ve koşullu kural çıkarımı, üzerinde en çok çalışılan tümevarım tekniklerindendir. Karar ağaçları, nitelik değerlerini taşıyan iç düğümler, sınıf düğümlerini taşıyan son düğümler ve nitelik değerlerini taşıyan düğümler arası yaydan oluşur. Basit bir karar ağacı yapısı aşağıdaki Şekil 2.2'de gösterilmiştir.



Şekil 2.2: Karar ağacı yapısı

Karar Ağacının oluşturulmasına yönelik olarak çeşitli ağaç oluşturma metotları vardır. Karar Ağaçları ile ilgili çalışmalar, ilk olarak Hunt ve Hoveland tarafından 1950'lerin sonlarında gerçekleştirilmiştir. Hunt'ın algoritmasından yola çıkan Quinlan, Shannon'un Bilgi Teorisine başvurmuş ve 1979 yılında ID3 Algoritması oluşturulmuştur. Quinlan'ın, 1993'te geliştirdiği C4.5 karar ağacı algoritması, ID3 algoritmasının tüm özelliklerini kendine miras alarak oluşturulmuş bir algoritmadır. ID3 algoritmasındaki bazı eksiklikler giderilerek yeni özellikler eklenmiştir. Quinlan, kitabında C4.5 algoritması ile ilgili çalışmalarını detaylı bir şekilde anlatmıştır [20].

Karar Ağaçları yaratılırken, özyinelemeli olarak, verilen örneklerini kökten yapraklara sıralayarak sınıflandırır. Her adımda, sonuçlara göre nitelik seçilir ve o niteliğe ait değerler atanır. Kalan nitelikler arasında aynı işlem tekrar edilerek, alt

ağaçlar için işlem yapılır. Bu süreç, sınıf bilgilerine ulaşılıncaya kadar devam eder. Karar ağaçları, böl ve elde et (divide-and-conquer) algoritmasının en iyi örneklerindendir.

C4.5 algoritmasının yanında, 1980'lerde Kass tarafından bulunan CHAID algoritması en eski ağaç sınıflandırma metotlarından biridir. CHAID, genelde büyük veri kümelerinin analizi için kullanılan, ikili olmayan ağaç algoritmasıdır. Ripley'e göre ise, CHAID algoritması, 1973'te Morgan ve Messenger tarafından geliştirilen THAID'i miras almıştır [48]. 1984 yılında Breiman, Friedman, Olshen ve Stone tarafından geliştirilen C&RT sınıflandırma ve regrasyon ağacı algoritması olup, hem kategorik verileri hem de devamlı veriler üzerinde çalışmaktadır [29].

Yukarıda bahsedilen CHAID ve C&RT algoritmalarından daha hızlı olup, büyük veri kümeleri için elverişli olmayan QUEST ağaç algoritması, 1997 yılında Loh ve Shih tarafından geliştirilmiştir. QUEST algoritması sadece kategorik veriler üzerinde çalışmaktadır [29,48,49].

Karar ağaçları yanında diğer bir yaklaşım ise kural üretimidir. Quinlan, kitabında sınıflandırıcı modelin kurallar şeklinde gösterilmesinin daha akıllıca olabileceğinden bahsetmiştir [20]. Kural üretmenin bir yolu karar ağaçlarını kullanmaktır. Ağaçtaki her bir son düğüm bir kuralı ifade etmektedir. Quinlan'ın geliştirdiği C4.5RULES, C4.5 algoritmasının oluşturduğu karar ağacından kurallar üreterek çalışmaktadır. Üretim kuralları (Production rules) olarak adlandırdığı kuralları, “en basit formda L -> R olmak üzere, L, solda tarafta birleştirilmiş nitelik-tabanlı testleri ve R, sağ taraftaki sınıfları ifade etmektedir” şeklinde açıklamıştır [20].

Karar Ağaçlarında kural üretmenin dışında, doğrudan kural üreten algoritmalar da vardır. Michalski'nin, 1969'da geliştirdiği AQ algoritması, örneklerden genel kurallar çıkaran bir algoritma olup, birçok yeni algoritmanın gelişmesine öncülük etmiştir [21]. Michalski ilerki yıllarda AQ algoritmasının eksiklerini gidererek, AQ algoritmasını geliştirmiştir (AQ11, AQ15, AQ17) [50]. AQ algoritması, kapsayan algoritmalara (covering algorithm) örnek olarak verilmektedir. Kapsayan algoritmalar, bir kural bulur ve kuralın kapsadığı tüm veriler atılır, kalan veriler

üzerinden işlem tekrarlanır. Kapsayan algoritması denmesinin sebebi ise, her adımda bazı örnekleri kapsayan kurallar tanımlanmasından dolayıdır [3,51]. Clark ve Niblett'in 1989 yılında geliştirdikleri CN2 algoritması da, kapsayan algoritmalara bir örnektir. Clark ve Niblett, CN2 ile ilgili yazdığı makalesinde, CN2 algoritmasını, "ID3'ün kirli verilerle uğraşırken ki etkili ve kabiliyeti yaklaşımından oluşan if-then kuralları ile AQ algoritma ailesinin arama stratejisinin birleşimidir" şeklinde açıklamışlardır [31]. Cendrowska'nın 1987 yılında geliştirdiği PRISM algoritması da, kapsayan algoritma örneklerindendir [51].

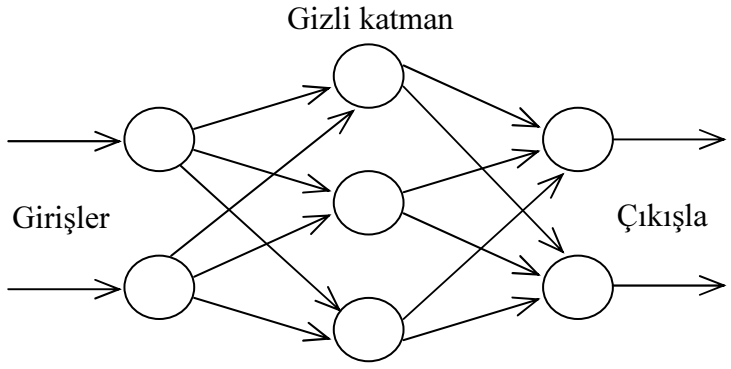
İkinci paradigma, örnek-tabanlı ya da durum-tabanlı öğrenmedir. Örnek-tabanlı öğrenmede, eğitim verileri tamamen depolanır ve uzaklık fonksiyonu kullanılarak, eğitim kümesindeki elemanların, test örneklerine ne kadar yakın olduğu belirlenir. En yakın eğitim örneği yerleştirilerek, test örnekleri için sınıf tahmini yapılır [51]. En basit ve iyi çalışan En Yakın-Komşu (Nearest-neighbor) algoritmasıdır. En Yakın-Komşu yönteminde, sınıflandırılacak örnekler ile eğitici sınıflardaki örnekler arasındaki uzaklıklar öklit uzaklık (euclidean distance) metodu kullanılarak hesaplanır ve en yakın olan örneğin sınıfı belirlenir. Sınıflandırılacak örnek, bu en yakın örneğin sınıfına atanır. K-En Yakın-Komşu yöntemi, En Yakın-Komşu yönteminin benzeridir. Bu yöntemde sınıflandırılacak örneğe k tane en yakın örnek belirlenir. Bu örnekleri en fazla içeren sınıfa yeni örnek atanır. En Yakın-Komşu ile ilgili ilk analiz, 1951'de Fix ve Hodges tarafından yapılmıştır. 1961'de ise Johns, sınıflandırma problemleri için kullanmıştır. 1992'de Aha'nın yaptığı çalışmalar ile En Yakın-Komşu metodu, makine öğrenmesinde önemli rol oynamıştır. [51,52]

Durum tabanlı öğrenmede, En Yakın-Komşu sınıflandırıcısından farklı olarak eğitim örneklerini noktalar olarak depolamak yerine; örnek ya da durumları karmaşık sembol tanımlamasıyla depolar. Durum tabanlı sonuçlandırmada, önceki durumlar ya da deneyimler kullanılır [53].

Üçüncü paradigma olan yapay sinir ağları kavramı, 1942 yılında McCulloch ve Pitts'in ilk hücre modelini geliştirmesi ile başlamıştır. Beynin bazı fonksiyonlarını ve özellikle öğrenme yöntemlerini benzetmek için tasarlanan yapay sinir ağları, yapay sinir hücrelerinin (nöronların) birbirleri ile çeşitli şekilde bağlanmasından oluşur.

Bunlar, ağırlıklandırılmış şekilde birbirlerine bağlanmıştır. Her bir nöron, komşularından veya dışsal kaynaklardan girdileri alır ve bunları çıktı sinyalini üretmek için kullanır. Çıktı sinyali diğer birimler boyunca yayılır. Nöronların diğer bir görevi ise ağırlık bilgilerini düzenlemektir. Bir yapay sinir ağı birimine giren toplam girdi, girdilerin ağırlıklarla çarpımının bir eşik değeri ile toplanması yöntemiyle bulunmaktadır. Bir işleme birimi, toplam girdi hesaplandıktan sonra, bir etkinlik fonksiyonu yardımıyla, o anki çıktı ve toplam girdi kullanılarak bir etkinlik değeri üretmektedir [54]. Şekil 2.3'de yapay sinir ağlarının yapısı gösterilmektedir.

60'ların sonlarında yapay sinir ağlarının beklentilere yanıt verememesi, araştırmaların belirli bir süre durmasına yol açmıştır. Yapay sinir ağlarına olan ilgi, 1980'lerin sonunda yapılan araştırmalarla ve işleme gücünü arttıran donanımsal gelişmelerle tekrar artmıştır. 1982 yılında Hopfield yapay sinir ağlarının birçok problemi çözebilecek kabiliyeti olduğunu göstermesi, 1984 yılında Kohonen öz düzenlemeli haritayı (self-organizing map) tanımlayıp, kendi adıyla anılan danışmasız öğrenme ağlarını geliştirmesi ile yapay sinir ağları konusu hareketlenme yaşamıştır. Bu çalışmaların yanında, 1974 yılında Werbos tarafından geliştirilip, 1986 yılında Parker, Rumelhart ve McClelland tarafından tekrar meydana çıkarılan geri yayınım ağı (Back Propagation Network) ile yapay sinir ağları konusu günümüzde önemini sürdürmektedir [14,54,55].



Şekil 2.3: Yapay sinir ağı yapısı

Diğer bir paradigma ise genetik algoritmalardır. Genetik algoritmaların temelleri 60'lı yıllarda John Holland tarafından atılmıştır. Amacı hem problemleri çözmek, hem de evrimsel sistemleri modellemektir. Genetik algoritmalar bir çözüm

uzayındaki her noktayı, kromozom adı verilen ikili bit dizisi ile kodlar. Her noktanın bir uygunluk değeri vardır. Tek bir nokta yerine, genetik algoritmalar bir populasyon (population) olarak noktalar kümesini muhafaza eder. Başlangıçta topluluğu oluşturma işlemi yapılmaktadır. Toplulukları evrim sürecine sokmadan önce yapılması gereken bir başka işlemde, başlangıç bireylerinin değerlendirilmesidir. Bu aşama evrim süreci içerisinde bir sonraki nesle döl verecek olan bireylerin belirlenmesi için gerekmektedir. Bu topluluk içerisinden uygunluk değeri yüksek olan bireyler seçilerek, en yüksek uygunluk değeri olan birey, uygunluk değeri düşük olan bireyin yerini almaktadır. Daha sonraki adımda ise genetik algoritma, çaprazlama ve mutasyon gibi genetik operatörleri kullanarak, yeni bir populasyon oluşturur. Mutasyon, örnekteki bazı değerleri rastgele olarak değiştirirken, çaprazlamada ise örnek çiftlerin farklı değerlerini birleştirerek, yeni örnekler oluşturulur. Birkaç kuşak sonunda, populasyon daha iyi uygunluk değerlerine sahip üyeleri içerir. Bu, Darwin'in rastsal mutasyona ve doğal seçime dayanan evrim modellerine benzemektedir. Genetik algoritmalar, makine öğrenmesinde ve en-iyileme (optimization) problemlerinde halen kullanılmaktadır [56,57,58].

Son paradigma ise, analitik öğrenmesine dayalıdır. Analitik öğrenme, mantıksal formdaki kurallar şeklinde gösterilir ve Horn cümlecikleri, kullanılan en genel tekniktir. Problemler teorem olarak ifade edilir ve ispatlar aranır. Öğrenme mekanizması, geçmişteki verileri kullanarak, ispatları ya da açıklamaları oluşturur. Az arama yaparak ya da tek adım ile benzer problemleri çözecek karmaşık kurallar, ispatları derleyerek oluşturulur [47,56].

Langley'in 1998'de yayınladığı diğer makalesinde, sınıflandırma için tanımladığı beş paradigma ise karar listeleri, karar ağaçları, yapay sinir ağları, örnek-tabanlı ya da durum-tabanlı, olasılık tabanlı algoritmalardır [59]. Yukarıda saydıklarımızdan farklı olarak, olasılıksal yöntemler de bu paradigmalara eklenmiştir. Olasılıksal yöntemler de, makine öğrenmesi alanında sıkça başvurulan yöntemlerden biridir.

Olasılıksal yöntemlerden Bayesian, en sık kullanılan tekniktir. 1763 yılında Bayes tarafından ortaya atılan olasılık teorisi, 1973'te Duda ve Hart tarafından desen algılamada kullanılmıştır. Sonraki yıllarda ise makine öğrenmesinde daha ciddi

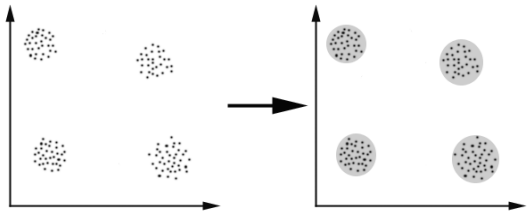
olarak ele alınmıştır [33]. Bu yaklaşımda, sınıf olasılığı ve verilen sınıfın nitelik-değer çiftinin koşullu olasılığı hesaplanır.

Bunun yanında, istatiksek modeller, birliktelik kuralları, kümeleme algoritmaları ya da daha basit kural üretme algoritmaları da makine öğrenmesinde kullanılmaktadır. İstatiksel modellerde genellikle doğrusal regresyon (linear regression) tekniği kullanılır. Doğrusal regresyon, tüm niteliklerin ve sınıf değerlerinin sayısal olma durumunda kullanılan bir tekniktir. Bu yaklaşımda, sınıf, önceden ağırlıkları belirlenmiş, niteliklerin doğrusal değerlerinin birleşimi olarak ifade edilir. Ağırlıklar, eğitim verilerinden hesaplanır. i. kayıtın gerçek sınıf değeri ile tahmin edilen sınıf değeri farkının karelerinin toplamı bulunur. Uygun katsayı, karelerin toplamı en küçük olacak şekilde seçilmelidir. Doğrusal regrasyon yöntemi sayısal tahminler için uygun ve kolay bir yöntem olarak tanımlanmıştır [51].

Birliktelik kuralları diğer bir öğrenme metotudur. Verinin potansiyel ilişkilerini tanımlar. Kurallar, destek ve güvenilirlik seviyelerine göre ayarlanır. Güçlü bir kural, çok büyük bir destek ve yüksek seviyeli bir güvenilirliğe sahiptir [61]. Birliktelik kurallarında kullanılan algoritmalardan biri Apriori'dir. Agrawal, 93 ve 94 yıllarında yazdığı makalelerde birliktelik kuralları ve geliştirdiği Apriori algoritmasından detaylı olarak bahsetmiştir [36,60].

Makine öğrenmesi, danışmanlı (supervised), danışmansız (unsupervised) ve takviyeli (reinforcement) öğrenme olarak incelenmektedir. Danışmanlı öğrenmede, kullanılacak girdiler ve istenilen çıktı belirlenmiştir. Her girdi için hedeflenmiş bir çıktı vardır. Eğitim verileri kullanılarak fonksiyon değeri tahmin edilir. Takviyeli öğrenme, bir taraftan danışmanlı öğrenmeye benzese de iki farklı öğrenme çeşididir. Takviyeli öğrenmede, çevre tarafından geri besleme olur ve çok detaylı bilgi bulunmaz. Geri besleme sinyalleri öğretici değil, değerlendircidir [62]. Oyun oynama takviyeli öğremeye için örnek olarak verilmiş, oyun oynarken yapılan bir hareketin önemli değil, doğru yaptığın hareket dizisinin önemli olduğu belirtilmiştir [38]. Diğer bir öğrenme tipi ise danışmansız öğrenmedir. Danışmansız öğrenmede ise elde sadece veriler bulunur, doğru cevaba yönelik veri bulunmaz. Amaç girdiler içinde bir düzen oluşturmaktır.

Kümeleme yöntemi, genelde danışmansız öğrenme metotlarından biri olarak görülse de danışmanlı öğrenmede de kullanılır. İkisi arasındaki fark; danışmanlı öğrenmede etiketli verilerden yararlanarak yeni veriyi bulmaya çalışırken, danışmansız öğrenmede ise etiketsiz veriler kullanılarak anlamlı kümeler çıkarmaktır [63]. Küme, benzer nesnelerin oluşturduğu bir gruptur. Kümelemenin en iyi uygulama örneği koordinat değerleri ile ifade edilen verilerin benzerliklerine göre gruplanmasıdır. Her bir kayıt bir nokta şeklinde gösterilir. Şekil 2.4'de basit bir şekilde gösterilmiştir [64].



Şekil 2.4: Kümeleme örneği

Kümeleme algoritmaları; Özel (Exclusive), Örtüşümlü (Overlapping) , Hiyerarşik (Hierarchical), Olasılıksal (Probabilistic) Kümeleme olarak dört şekilde incelenmektedir.

Özel kümelemede, her veri belli bir kümeye aittir ve kesinlikle başka kümeye dahil olamaz. K-means algoritması özel kümelemeye girmektedir. Örtüşümlü kümelemede ise, verileri kümelemek için bulanık kümeler kullanılır. Böylece her nokta iki ya da daha fazla kümeye farklı derecelerle dahil olabilir. Fuzzy C-means örtüşümlü kümeleme için örnek olarak gösterilmektedir. Hiyerarşik kümeleme algoritmaları ise en yakın iki küme arasındaki birleşim ile oluşur. Başlangıçta her verinin bir küme olduğu düşünülür ve birkaç tekrardan sonra sonuç kümesine ulaşılır. Sonuncu kümeleme algoritması ise, olasılıksal yaklaşımın kullanıldığı kümeleme algoritmasıdır. Her küme matematiksel olarak Gaussian ya da Poisson dağılımları ile ifade edilir. Tüm veri kümesi bu dağılımların karışımı ile modellenir [64]. Kümeleme algoritmalarının en önemli parçası veri noktaları arasındaki uzaklık ölçüsüdür.

Euclidean (öklit) ve yüksek boyutlu veriler için Minkowski en sık kullanılan uzaklık ölçülerindendir [63,64].

1R algoritması gibi [51]'de geniş olarak bahsedilen basit ve etkili kural üretme algoritması da makine öğrenmesinde kullanılan diğer bir tekniktir. 1993 yılında, Holte'nin gerçekleştirdiği algoritmaya tezin ikinci bölümünde detaylı bir şekilde değinilecektir.

Yukarıda değinilen tüm paradigmaların dışında, son yıllarda yapılan çalışmalarda ise Melez metot (Hybrid Method) denilen, birçok tekniği birarada bulunduran teknikler kullanılmaktadır. Örneğin, karar ağaçları yaratılırken, eşik birimi ve bazı teknikler kullanılarak sinir ağlarında kurallara dönüştürülmesi ya da analitik metodlar kullanılırken, geçmişe ait bilgilere çıkarım metotları ile ulaşılması ya da yapay sinir ağlarında kullanılacak verinin kümelenip, birliktelik kuralları ile sıralanması, melez metotlardan bazılarıdır [47, 65].

2.4. Makine Öğrenmesinin Uygulama Alanları

Makine öğrenmesi metotları kullanılarak geliştirilen uygulamalar, bilimde ve özellikle sanayi alanında çok sık kullanılır hale gelmiştir. Uygulamalar, desen anlama, konuşma algılama, robotik gibi alanların dışında, bankacılık, pazarlama, tıp, haberleşme, üretim gibi alanlarda da sıkça kullanılmıştır. Aşağıda, [47,61,66,67]'de yeralan bazı uygulamalardan kısaca bahsedilmiştir.

Kredi başvuruları ile ilgili olarak, Michie tümevarım metotlarından karar ağaçlarını kullanarak, doğruluk oranı yüksek kurallar bulmuştur. Şirket bu kuralları kullanarak kredi için başvuranlara, sebepleri ile birlikte kararlarını söylemiştir [68].

Giordana, Neri ve Saitta'nın, tümevarım algoritması ile mekanik bulgulara ait kurallar üretmiş [69], Leech'in kimyasal süreç kontrolünde, verimi arttırmak için, karar ağaçlarını kullanarak kurallar oluşturmuştur [70].

Evan ve Fisher'in baskılama süreci ile ilgili tümevarım çalışması sayesinde Amerika'nın büyük baskı şirketlerinden biri olan R.R.Donnelley için kazanç sağlamış olması sunulabilecek örneklerdendir [71].

Diğer bir uygulama ise U. M. Fayyad ve P. Smyth'in astronomide kullanılan objeleri, görüntü işleme ve karar ağacı metodu ile, astronomların kullanacağı şekilde sınıflandırma yapmasıdır [72].

Petrolden gazın ayrıştırılması ile ilgili olarak, Guilfoyle, hazırladığı raporda, İngiliz Petrol'ün (British Petroleum) karar ağacı kullanarak, kurallar ürettiğinden bahsetmiştir [73].

Arthur L. Delcher, Douglas Harmon, Simon Kasif, Owen White, Steven L. Salzberg'in, Glimmer programını yaratarak biyolojideki genlerin yüzde 97-99'unu teşhis etmeleri, uygulama örneklerinden biridir [74].

Yukarıda bahsedilen uygulamaların yanında, farklı teknikler kullanılarak haberleşme, biyoloji, üretim, tıp, medya gibi birçok alanda, makine öğrenmesi uygulamaları geliştirilmiştir.

3. 1R ALGORİTMASI

3.1. 1R Algoritmasına Detaylı Bakış

Makine öğrenmesinin amaçlarından bir tanesi, makine öğrenmesi tarafından üretilecek kuralların basitlik ve doğruluğunu arttırmaktır. Bu amaç ile yapılan araştırmalarda benimsenen geleneksel yöntemler, büyük hipotez uzayında karmaşık hipotezler içerir. Günümüzde makine öğrenmesindeki gelişmeler, karmaşık hipotezler yerine basit hipotezlere yönelmiştir [35].

1R, kural üretmek için kullanılan en basit algoritmalardan biridir. Algoritma, 1993 yılında Robert C. Holte tarafından geliştirilmiştir [34]. 1R algoritması, bir niteliğe ait kurallar üretir. Bu yüzden bir seviyeli ağaç olarak da tanımlanır.

Algoritmanın amacı, doğruluk oranı en yüksek niteliğe ait kurallar üretmektir. Bunun için de veri kümesindeki her bir nitelik değerinin ait olduğu sınıflar bulunur. En çok görülen sınıf değerleri alınır. Çıkan sınıf değerlerine göre, o niteliğe ait kurallar üretirilir. Çıkan kuralların doğruluk değerleri ya da hata oranları hesaplanır. Doğruluk oranı en yüksek ya da hata oranı en düşük nitelik seçilir. Algoritmanın işleyişi şöyledir:

Her bir a niteliği için:
    Her bir a niteliğinin v değeri için:
        a niteliğine ait değerlerin kayıtları seçilir
        her bir değerin ait olduğu c sınıflarından en sık karşılaşılan seçilir
        eğer a niteliğinin v değeri ise sınıf c'dir diye kural çıkartılır
    Çıkartılan kuralın sınıflandırma doğruluk değeri hesaplanır
En yüksek doğruluk değerine sahip olan kural kullanılır

Algoritmanın işleyişi için, en sık karşılaşılan hava veri kümesi kullanılmıştır. Hava veri kümesi, sahip olduğu niteliklerin değerlerine göre golf oynanıp oynanamayacağına dair veriler içerir. Hava veri kümesi Görünüş, Sıcaklık, Nemlilik, Rüzgar olmak üzere 4 nitelikten oluşur. Bunun yanında sonuç verilerini içeren Sınıf niteliği bulunur. Görünüş niteliğinin değerleri güneşli, yağmurlu, bulutlu; Sıcaklık niteliğinin değerleri sıcak, ılık, serin; Nemlilik niteliğinin değerleri yüksek, normal; Rüzgar niteliğinin değerleri var, yok'tur. Sonuç verilerinin bulunduğu Sınıf niteliği ise evet, hayır olmak üzere 2 sınıfa ayrılmıştır. Kullanılacak veri kümesi 14 adet örnek içerir. Veri kümesi Tablo 3.1'deki gibidir.

Tablo 3.1: Golf ile ilgili hava veri kümesi

Görünüş	Sıcaklık	Nemlilik	Rüzgar	Sınıf
güneşli	sıcak	yüksek	yok	hayır
güneşli	sıcak	yüksek	var	hayır
bulutlu	sıcak	yüksek	yok	evet
yağmurlu	ılık	yüksek	yok	evet
yağmurlu	serin	normal	yok	evet
yağmurlu	serin	normal	var	hayır
bulutlu	serin	normal	var	evet
güneşli	ılık	yüksek	yok	hayır
güneşli	serin	normal	yok	evet
yağmurlu	ılık	normal	yok	evet
güneşli	ılık	normal	var	evet
bulutlu	ılık	yüksek	var	evet
bulutlu	sıcak	normal	yok	evet
yağmurlu	ılık	yüksek	var	hayır

1R algoritmasını adım adım incelersek, ilk yapacağımız nitelik değerlerinin frekanslarını bulmaktır. Aşağıdaki Tablo 3.2'de niteliklere ait değerlerin frekansları

gösterilmiştir. İlk olarak Görünüş niteliğine ait güneşli değerinin evet sınıfında 2, hayır sınıfında 3 olmak üzere toplam 5 kez görüldüğü belirtilmiştir. Bulutlu değerine baktığımızda, sadece evet sınıfına ait 4 değeri olduğu görülür. yağmurlu değerinde ise evet sınıfına ait 3, hayır sınıfına ait 2 değer olduğu belirtilmiştir. Diğer nitelik değerlerinin frekansları da aynı şekilde bulunmuştur.

Sıcaklık niteliğinin sıcak değerine baktığımızda ise her iki sınıfa ait değerlerin sayıları aynıdır. Bunun gibi durumlarda rastgele bir seçim yapılarak, o değere ait sınıf seçilir.

Tablo 3.2: Nitelik değerlerinin frekansları

Görünüş	evet	hayır
güneşli	2	3
bulutlu	4	0
yağmurlu	3	2
Sıcaklık	evet	hayır
sıcak	2	2
ılık	4	2
serin	3	1
Nemlilik	evet	hayır
yüksek	3	4
normal	6	1
Rüzgar	evet	hayır
var	3	3
yok	6	2

Değerlerin frekansları bulunduktan sonraki adım ise, yüksek frekans değerini seçerek kural oluşturmaktır. Oluşturulan kuralların doğruluk değerleri hesaplanır ve en yüksek doğruluk değerine sahip nitelik seçilir ve o niteliğe ait kurallar kullanılır. Aşağıdaki Tablo 3.3'de, frekans değerlerine göre kurallar oluşturulmuş ve bir niteliğe ait kuralların doğruluk değerleri hesaplanmıştır.

Tablo 3.3: Oluşan kurallar ve doğruluk değerleri

Nitelik	Kurallar	Doğruluk Değeri
Görünüş		(10/14)
	Eğer güneşli ise hayır (3/5)	
	Eğer bulutlu ise evet (4/4)	
	Eğer yağmurlu ise evet (3/5)	
Sıcaklık		(9/14)
	Eğer sıcak ise hayır (2/4)	
	Eğer ılık ise evet (4/6)	
	Eğer serin ise evet (3/4)	
Nemlilik		(10/14)
	Eğer yüksek ise hayır (4/7)	
	Eğer normal ise evet (6/7)	
Rüzgar		(9/14)
	Eğer var ise hayır (3/6)	
	Eğer yok ise evet (6/8)	

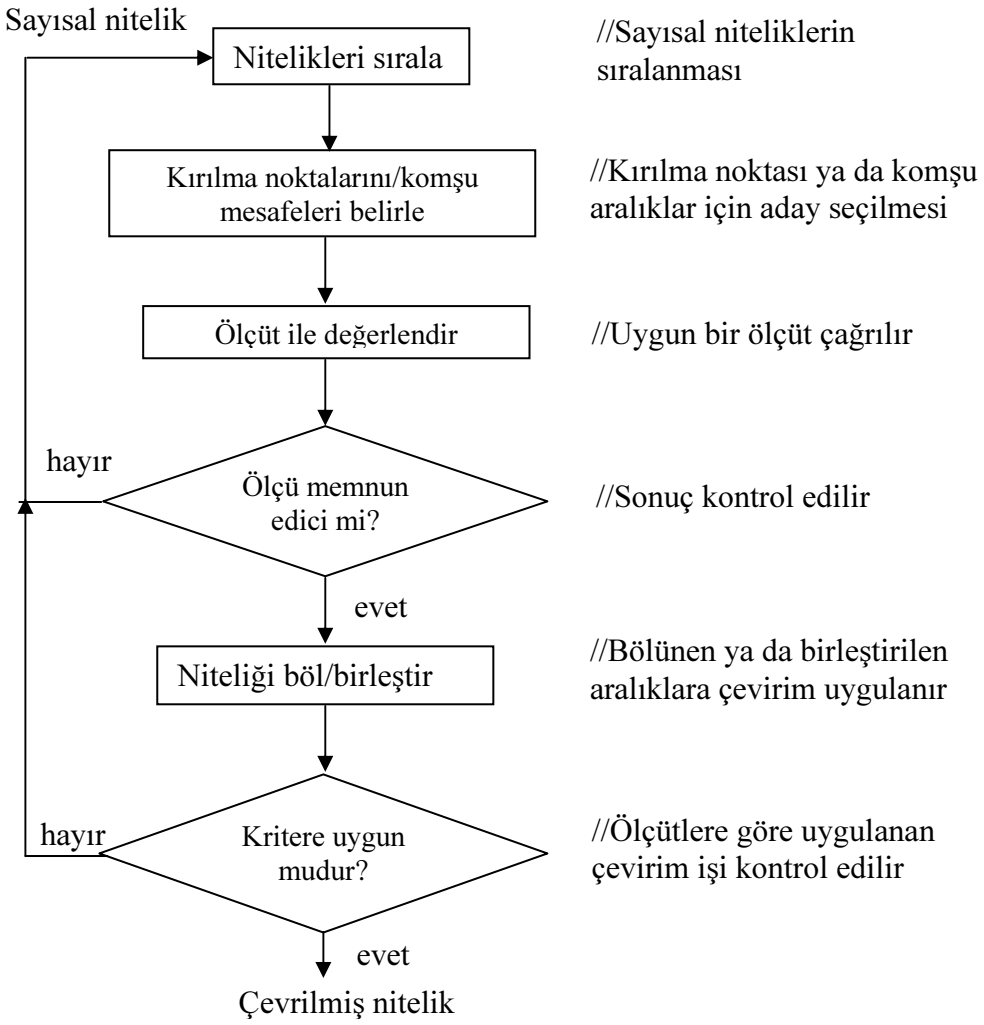
Buna göre en yüksek doğruluk değeri 10/14'tür. Bu değere sahip 2 nitelik vardır: Görünüş ve Nemlilik. Bu iki niteliğin arasında hangisine ait kurallar üretileceği, rastgele seçilebileceği gibi ilk sıradaki nitelik hangisi ise ona ait kurallar da üretilebilir. Eğer ilk sıradaki niteliğe ait kuralların seçildiğini varsayarsak, Görünüş niteliği seçilir. Ortaya çıkan kurallar:

Eğer Görünüş güneşli ise hayır
Eğer Görünüş bulutlu ise evet
Eğer Görünüş yağmurlu ise evet

Sonuç olarak, bu veri kümesinden hava güneşli ise golf oynamamalı, bulutlu ve yağmurlu ise oynanabilir kuralları çıkmıştır. Çıkan kurallar kullanılarak, tahmin ya da tanım yapılabilmektedir.

3.1.1. Sayısal özellikler

Veri kümelerinde karşılaşılan nitelikler kategorik (nominal) ve sayısal olmak üzere iki tiptir. Makine öğrenmesi alanında geliştirilen algoritmaların çoğu sadece kategorik değerlerle uğraşmaktadır. Fakat, gerçek dünya ile ilgili veri kümelerinin çoğu sayısal özellikler içerir. Bu yüzden, kategorik değerler ile çalışan algoritmalar üzerinde yapılacak ilk işlem, sayısal değerleri kategorik değerlere dönüştürmektir. Genel anlamda bu süreç [75]'de Şekil 3.1'de açıklanmaktadır:



Şekil 3.1: Sayısal verileri kategorik verilere dönüştürme süreci

Son yıllarda sayısal nitelikleri kategorik hale çevirmek makine öğrenmesi alanında önem kazanmış ve birçok teknik geliştirilmiştir. Eşit Genişlikli Aralık (equal interval width), ChiMerge, StatDisc gibi ya da entropy-tabanlı birçok teknik geliştirilmiş ve karşılaştırmalar yapılarak avantaj ve dezavantajları açıklanmıştır. Bu tekniklerden bir tanesi de, Robert Holte tarafından geliştirilen ve basit bir yöntem olarak görülen 1RD (1R Discretization) metodudur [76]. Bu bölümde 1RD metodu açıklanacaktır. Anlatırken kullanacağımız veri kümesinde iki tip veri vardır: kategorik ve sayısal. Kategorik veri daha önceki bölümde kullanılan veri tipidir. Örnek olarak Görünüş niteliği güneşli, bulutlu ve yağmurlu, Sıcaklık niteliği ise sıcak, ılık ve serin ile ifade edilen kategorik değerler içerir. Bu bölümde ise bir önceki bölümde kullandığımız veri kümesinin iki niteliğinin sayısal değerler içermiş hali kullanılacaktır.

Tablo 3.4: Sayısal nitelik içeren hava veri kümesi

Görünüş	Sıcaklık	Nemlilik	Rüzgar	Sınıf
güneşli	85	85	yok	hayır
güneşli	80	90	var	hayır
bulutlu	83	86	yok	evet
yağmurlu	70	96	yok	evet
yağmurlu	68	80	yok	evet
yağmurlu	65	70	var	hayır
bulutlu	64	65	var	evet
güneşli	72	95	yok	hayır
güneşli	69	70	yok	evet
yağmurlu	75	80	yok	evet
güneşli	75	70	var	evet
bulutlu	72	90	var	evet
bulutlu	81	75	yok	evet
yağmurlu	71	91	var	hayır

Tablo 3.4’de, iki niteliği sayısal değerler içeren hava veri kümesi gösterilmektedir. Görünüş ve Rüzgar nitelikleri kategorik değerler içerirken, Sıcaklık ve Nemlilik nitelikleri ise sayısal veriler içerir.

1RD metodunda ilk olarak sayısal değerler içeren niteliğin değerleri ve değerlerin ait oldukları sınıflar küçükten büyüğe doğru sıralanırlar. Veri kümemizdeki iki niteliğin sıralanmış hali Tablo 3.5’deki gibidir.

Tablo 3.5: Sayısal nitelik değerlerinin küçükten büyüğe sıralanmış hali

Sıcaklık niteliği:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
evet	hayır	evet	evet	evet	hayır	hayır	evet	evet	evet	hayır	evet	evet	hayır

Nemlilik niteliği:

65	70	70	70	75	80	80	85	86	90	90	91	95	96
evet	hayır	evet	evet	evet	evet	evet	hayır	evet	hayır	evet	hayır	hayır	evet

Daha sonraki adımda ise, kırılma noktaları denilen, sınıfların değiştiği yerlerden parçalara ayrılır. Tablo 3.6’da parçalara ayrılmış hali gözükmektedir.

Tablo 3.6: Kırılma noktaları

Sıcaklık niteliği:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
evet	hayır	evet	evet	evet	hayır	hayır	evet	evet	evet	hayır	evet	evet	hayır

Nemlilik niteliği:

65	70	70	70	75	80	80	85	86	90	90	91	95	96
evet	hayır	evet	evet	evet	evet	evet	hayır	evet	hayır	evet	hayır	hayır	evet

Sıcaklık niteliği için kırılma noktaları 64.5, 66.5, 70.5, 72, 77.5, 80.5, 84. Bu örnekte 72 değeri problem yaratabilir. Çünkü iki adet 72 değeri vardır ve iki ayrı sınıfa ayrılmıştır. Bu gibi durumda 72 değeri bir yukardaki 73.5 değeri ile yer değiştirir. Nemlilik niteliği için kırılma noktaları 67.5, 70, 82.5, 85.5, 88, 90.5, 95.5 değerleridir.

Diğer bir problem de, bir nitelik için kırılma noktaları ile küçük parçalara ayrılmış aralıklardan fazla sayıda kural çıkmasıdır. Robert C. Holte bu probleme şu şekilde bir çözüm sunmuştur: Daha önceden belirlenen bir sayı kadar, aynı sınıfa ait olan veriler seçilerek bölümlere ayrılmalı, kırılma noktaları belirlenmelidir. Küçük veri kümeleri (50 den az örneğe sahip veri kümeleri) için üç, büyük veri kümeleri için altı sayısı uygun bulunmuştur.

Veri kümemizde 14 adet örnek bulunduğu için, sayı 3 olarak seçilir. Daha sonra ise nitelikte aynı sınıf değeri minimum 3 olacak şekilde bölümlere ayrılır. Sıcaklık ve Nemlilik niteliği incelenecek olursa, yukarıdaki bölünmeler elenmiş, sınıf sayısı minimum 3 olacak şekilde yeni bölümler oluşturulmuştur. Tablo 3.7'de yeni kırılma noktaları gösterilmiştir.

Tablo 3.7: Sınıf sayısı en az 3 olacak şekilde belirlenmiş kırılma noktaları

Sıcaklık niteliği:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
evet	hayır	evet	evet	evet	hayır	hayır	evet	evet	evet	hayır	evet	evet	hayır

Nemlilik niteliği:

65	70	70	70	75	80	80	85	86	90	90	91	95	96
evet	hayır	evet	evet	evet	evet	evet	hayır	evet	hayır	evet	hayır	hayır	evet

Görüldüğü üzere Sıcaklık niteliğinde ilk iki bölüm evet sınıfına aittir ve bu, iki bölümde de en sık görülen sınıf değerinin evet olduğu anlamına gelir. Eğer yanyana bulunan bölümler aynı sınıf değerlerine sahipse birleştirilir. Eğer bölümdeki sınıf sayısı eşit ise önceki sınıf değerlerine bakılarak anlamlı olacak şekilde hangi sınıfa ait olması gerektiğine karar verilir. Bahsedilen durum Sıcaklık niteliği için geçerlidir. Son bölümde evet ile hayır sayısı birbirine eşit olmasına rağmen hayır değeri seçilmiştir. evet değeri seçilseydi, nitelik için kırılma noktası kalmayacak, tüm değerler bir sınıfa ait olacaktı. Tablo 3.8, kırılma noktalarının son halini içerir.

Tablo 3.8: Kırılma noktalarının son hali

Sıcaklık niteliği:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
evet	hayır	evet	evet	evet	hayır	hayır	evet	evet	evet	hayır	evet	evet	hayır

Nemlilik niteliği:

65	70	70	70	75	80	80	85	86	90	90	91	95	96
evet	hayır	evet	evet	evet	evet	evet	hayır	evet	hayır	evet	hayır	hayır	evet

Tüm birleştirmeler yapılıp, kırılma noktaları da belirlendikten sonra sayısal değerlere kategorik değerler atanır.

Sıcaklık : <= 77.5 --> evet
> 77.5 --> hayır
Nemlilik: <= 82.5 --> evet
> 82.5 ve <= 95.5 --> hayır
> 95.5 --> evet

Yeni atanan bu değerlere göre, veri kümemiz Tablo 3.9'daki gibidir. Değerler yerine evet ya da hayır yazılmasa da programın işleyişine göre, belirtilen sayısal değerler

için evet ve hayır değerleri kullanılacaktır. Belirlenen bu kurallar için değerler yerleştirildiğinde, Sıcaklık niteliğinin hata sayısı dört, Nemlilik niteliğinin hata sayısı üç olarak çıkmaktadır.

Tablo 3.9: Veri kümesinin kategorik değerlere çevrilmiş hali

Görünüş	Sıcaklık	Nemlilik	Rüzgar	Sınıf
güneşli	hayır	hayır	yok	hayır
güneşli	hayır	hayır	var	hayır
bulutlu	hayır	hayır	yok	evet
yağmurlu	evet	evet	yok	evet
yağmurlu	evet	evet	yok	evet
yağmurlu	evet	evet	var	hayır
bulutlu	evet	evet	var	evet
güneşli	evet	hayır	yok	hayır
güneşli	evet	evet	yok	evet
yağmurlu	evet	evet	yok	evet
güneşli	evet	evet	var	evet
bulutlu	evet	hayır	var	evet
bulutlu	hayır	evet	yok	evet
yağmurlu	evet	hayır	var	hayır

Veri kümesinin yeni hali ile en sık görülen sınıf değerleri hesaplanır ve en büyük olanlar seçilerek, doğruluk oranı en yüksek nitelik bulunur. Bulunan niteliğe göre kurallar üretilir.

Tablo 3.10'da, değişen veri kümesine göre hesaplanan frekans değerleri gösterilmektedir. Görüldüğü üzere, Tablo 3.2'de çıkan frekans değerleri ile Tablo 3.10'da çıkan bazı frekans değerleri farklıdır. Sayısal niteliklerin frekans değerleri tamamen değişmiştir.

Tablo 3.10: Değişen veri kümesine göre frekans değerleri

Görünüş	evet	hayır
güneşli	2	3
bulutlu	4	0
yağmurlu	3	2
Sıcaklık	evet	hayır
evet (<= 77.5)	7	3
hayır (> 77.5)	2	2
Nemlilik	evet	hayır
evet (<= 82.5 ve > 95.5)	7	1
hayır (82.5 < <= 95.5)	2	4
Rüzgar	evet	hayır
var	3	3
yok	6	2

Bir sonraki adımda ise, frekans değerlerine göre kurallar oluşturulmuş ve her niteliğe ait kuralların doğruluk değerleri hesaplanmıştır. Tablo 3.11, değişen veri kümesi ile oluşan kuralları ve doğruluk oranını göstermektedir. Sayısal niteliklere ait kurallar yeni değerlerine göre oluşmuştur. Doğruluk değerleri incelendiğinde ise, Nemlilik niteliğinin doğruluk değerinin değiştiği görülmektedir.

Tablo 3.11: Değişen veri kümesinin doğruluk değerleri

Nitelik	Kurallar	Doğruluk Değeri
Görünüş		(10/14)
	Eğer güneşli ise hayır (3/5)	
	Eğer bulutlu ise evet (4/4)	
	Eğer yağmurlu ise evet (3/5)	
Sıcaklık		(9/14)
	Eğer evet (<= 77.5) ise evet (7/10)	
	Eğer hayır (>77.5) ise hayır (2/4)	
Nemlilik		(11/14)
	Eğer evet (<= 82.5 ve > 95.5) ise evet (7/8)	
	Eğer hayır (82.5 < <= 95.5) ise hayır (4/6)	
Rüzgar		(9/14)
	Eğer var ise hayır (3/6)	
	Eğer yok ise evet (6/8)	

Buna göre Nemlilik niteliği, en yüksek doğruluk oranına sahip nitelik olarak seçilmiştir. Eğer havadaki nemlilik oranı 82.5'den düşük ve eşit ise ya da 95.5'ten büyük ise golf oynanabilir, 82.5'ten büyük ve 92.5'ten küçük ise oynanamaz kararına varılabilir. Ortaya çıkan kurallar şu şekildedir:

Eğer Nemlilik evet (<= 82.5 ya da >95.5) ise evet
Eğer Nemlilik hayır (>82.5 ve <=95.5) ise hayır

3.1.2. Kayıp veriler

Veri kümesindeki veri, çeşitli sebeplerden dolayı eksik olabilir. Veri toplanırken ya da veri tabanına aktarılırken çıkan sorunlardan ya da veri formatına uymayan verilerin toplanması gibi sebeplerden, veri kümesi eksik veriler içerebilir. Veri kümesindeki kayıp veriler ile baş etmek için çözüm yolları sunulmuştur. Robert C.

Holte'nin sunduğu yol ise; kayıp veriye o niteliğin başka bir değeri gibi davranmaktır. Örneğin, önceki bölümlerde kullanılan veri kümesindeki Görünüş niteliğinin kayıp veriye sahip olduğunu varsayarsak, Görünüş niteliğinin değerleri güneşli, bulutlu, yağmurlu ve kayıp olmak üzere 4 tane olur. Böylece kayıp verilere kayıp değeri verilerek, niteliğe yeni bir değer eklenmiş olur. Bu yol basit fakat etkili bir yol olarak görülmektedir [51].

4. 1R ALGORİTMASININ PYTHON PROGRAMLAMA DİLİNDE UYGULANMASI

4.1. Python Programlama Dili

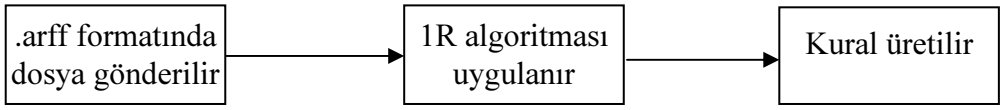
Geliştirilmeye 1990 yılında Guido van Rossum tarafından başlanan Python, nesnel, yorumlanabilen, özgür, taşınabilir, güçlü ve hızlı bir programlama dilidir. Aşağıda Python dili ile ilgili sayılan tüm özelliklerin açıklaması sıra ile verilmiştir [77].

- Python dili nesneye yönelik bir dildir. Python'da her şey bir nesnedir. Tüm değişkenler, fonksiyonlar, modüller, sınıflar birer nesnedir.
- Pythonda yazdığınız kod önce Python derleyicisi tarafından sizin isteğiniz dışında byte-code denilen bir biçime derlenir ve daha sonra yorumlanır.
- Pyhton özgür bir dildir ve bu yüzden de kaynak kodu açıktır. Kaynak kodu üzerinde değişiklik yapılabilir, ihtiyaca göre değiştirilebilir.
- Python, hemen her işlemci ve işletim sisteminde çalışabilir. Unix tabanlı işletim sistemleri, Mac, Windows bunlardan birkaçıdır.
- Python'u hızlı kılan etkenlerin başında modüllerin ana sistemden bağımsız olması gelir. İstenilenin dışında hiçbir modül hafızaya yüklenmez. Çöp toplayıcı sayesinde ise tanımlanan nesnelerle olan bağlantılar sonlanınca hemen bellekten silinir [78,79,80].

Günümüzde veritabanı, internet, GUI programcılığında sıkça kullanılan Python, dünyanın en hızlı arama motoru olarak görülen Google arama motorunda bile kullanılmaktadır.

4.2. Programın İçeriği

Tez kapsamında basit yapılar kullanılarak yazılan programın bir kısmı 1R algoritması kullanarak kural üretme işlemini yerine getirmektedir. Gerçekleştirilen programın bir kural üreten kısmı Ek-A'da yer almaktadır. Programın işleyişi Şekil 4.1.'de gösterilmiştir.



Şekil 4.1: Programın işleyişi

Programdaki ilk adım verilerin bulunduğu dosyayı seçmektir. Veriler .arff (Attribute-Relation File Format) uzantılı olmak üzere, nitelik-ilişki dosya formatı şeklinde saklanmalıdır. ARFF dosya formatı, Weka Makine Öğrenmesi yazılımında kullanılması amaçlanarak, Waikato Üniversitesi Bilgisayar Bilimleri Bölümünün geliştirdiği Makine Öğrenmesi Projesi'nde ortaya çıkmıştır. Weka Makine Öğrenmesi yazılımı, özellikle araştırmacıların deneysel araştırmaları için kullandığı, makine öğrenmesinde kullanılan birçok tekniğin barındığı bir yazılımdır [51]. ARFF formatı, verilerin daha düzenli ve daha sıralı olmasını sağlar. Tablo 4.1'de, bir bölüm önce kullandığımız veri kümesinin içeriğini .arff dosyası olarak görmekteyiz.

Tablo 4.1: Golf ile ilgili verileri içeren Hava.arff dosyası

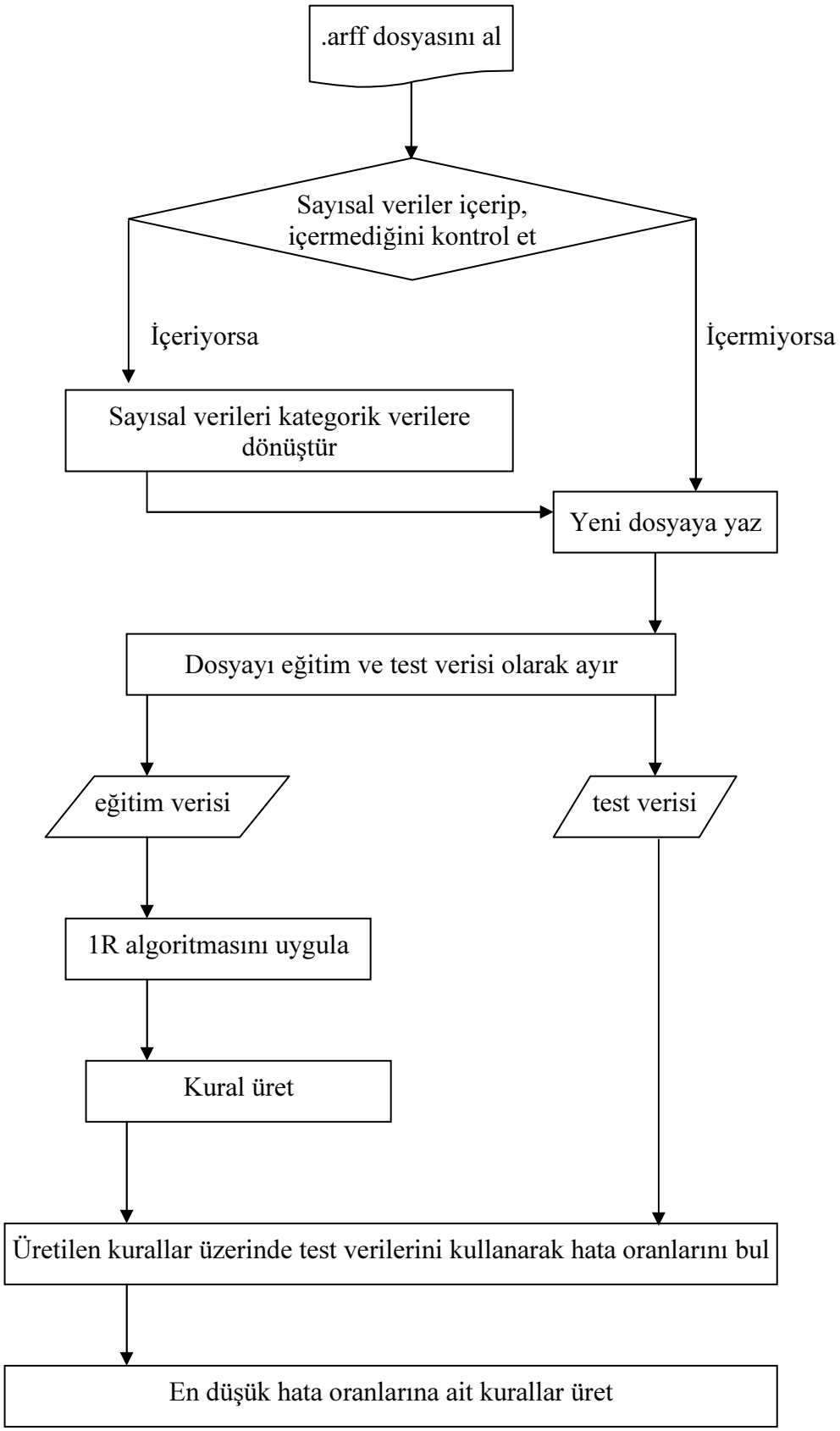
```
%Golf oynanıp oynanmayacağına karar vermek için
%kullanılan ve sayısal değerler içeren hava veri kümesi
@relation hava
@attribute Gorunus {gunesli,bulutlu,yagmurlu}
@attribute Sicaklik real
@attribute Nemlilik real
@attribute Ruzgar {var,yok}
@attribute Sinif {evet,hayır}
@data
gunesli,85,85,yok,hayir
gunesli,80,90,var,hayir
```

Tablo 4.1'de görüldüğü üzere dosya türkçe karakterler barındırmamaktadır. Örneğin, bir bölüm önce kullandığımız Görünüş niteliği, bundan sonraki bölümlerde Gorunus olarak karşımıza çıkmaktadır. Bu yüzden, program sonuçları da bu verilere göre oluşmaktadır.

ARFF formatında, % ile başlayan kısım, dosya ile ilgili bilgi vermek için kullanılan yorum kısmıdır. Genelde verinin ne işe yaradığı, kimin tarafından oluşturulduğu ve nerelerde kullanıldığına dair bilgiler içerir. @relation kısmı, veri kümesinin ne ile alakalı olduğunu ya da ismini belirtir. @attribute kısmı ise her bir niteliği ve o niteliğe ait değerleri göstermektedir. Nitelik değerleri, niteliğin yanındaki parantezler içinde yer alır. Niteliğin yanında yazan real ise, o niteliğin sayısal veriler içerdiğini belli etmektedir. Örneğin, @attribute Gorunus {gunesli,bulutlu,yagmurlu} satırı, nitelik isminin Gorunus, nitelik değerlerinin ise gunesli, bulutlu ve yagmurlu olduğunu göstermektedir. @data kısmı, isminden de anlaşılacağı üzere verilerin başladığı kısımdır. Her satırda, sınıf dahil olmak üzere tüm niteliklere ait değerler, virgül ile ayrılmış olarak gösterilmektedir. Örneğin, gunesli,85,85,yok,hayir, satırı, Gorunus niteliği için gunesli, Sicaklik niteliği için 85, Nemlilik niteliği için 85, Ruzgar niteliği için yok, Sinif niteliği için ise hayır değerlerine sahiptir. Şekil 4.1'de görüldüğü üzere, ARFF formatındaki veriler, kod seviyesinde 1R algoritmasının uygulanması için alınır ve kural üretme işlemi gerçekleştirilir. Bu bölümün alt bölümlerinde ilk adımdan başlayarak programın işleyişi detaylı şekilde açıklanmaktadır.

4.3. Programa Detaylı Bakış

Bir önceki bölümde, programın detaylarına girilmeden, şekil üzerinde kabaca işleyişinden bahsedilmişti. Bu bölümde ise, program ayrıntılı olarak incelenecektir. Programın işleyişi, Şekil 4.2'de akış diyagramı halinde detaylı olarak verilmiştir. Şekil 4.2'den de anlaşılacağı gibi program, sayısal verilerin kategorik verilere dönüştürülmesi, eğitim ve test verilerine ayrılması, 1R algoritmasının uygulanması ve hata oranlarının bulunması olarak incelenmektedir. Tezin ilerleyen kısımları da buna göre şekillenmektedir.



Şekil 4.2: Programın akış diyagramı

4.3.1 Sayısal verilerin kategorik verilere dönüştürülmesi

Programın, sayısal verileri kategorik hale nasıl dönüştürdüğü, Bölüm 3.1'de kullanılan veri kümesi üzerinden anlatılacaktır. Veri kümesi, Tablo 4.2'de görüldüğü gibi, iki adet kategorik, iki adet de sayısal veriler içeren dört niteliğe sahiptir.

Tablo 4.2: Sayısal nitelik içeren Hava.arff veri kümesi

Gorunus	Sicaklik	Nemlilik	Ruzgar	Sinif
gunesli	85	85	yok	hayir
gunesli	80	90	var	hayir
bulutlu	83	86	yok	evet
yagmurlu	70	96	yok	evet
yagmurlu	68	80	yok	evet
yagmurlu	65	70	var	hayir
bulutlu	64	65	var	evet
gunesli	72	95	yok	hayir
gunesli	69	70	yok	evet
yagmurlu	75	80	yok	evet
gunesli	75	70	var	evet
bulutlu	72	90	var	evet
bulutlu	81	75	yok	evet
yagmurlu	71	91	var	hayir

Bölüm 4.3’deki akış şeması göz önünde bulundurularak, programın işleyişindeki ilk adım, dosyanın sayısal veriler içerip içermediğini kontrol etmektir. Kullanacağımız dosya sayısal veriler içerdiğine göre, yapılacak ilk işlem sayısal verileri kategorik verilere çevirmektir. Bölüm 3.1.1’de bahsedildiği gibi sayısal verileri kategorik verilere çevirmek için birçok teknik geliştirilmiştir.

Program, bu işlem için 1RD metodunu kullanır. 1RD metodu, Bölüm 3.1.1’de detaylı olarak incelenmiştir. Bu işlemi gerçekleştirmek için, program ilk olarak, sayısal veriler içeren nitelikleri saptamaktadır. Kullanılacak dosyada bu nitelikler: Sicaklik ve Nemlilik nitelikleridir. Bu nitelikler saptandıktan sonra, niteliğe ait değerler, ait oldukları sınıf değerleri ile birlikte listeye atılır. Listeye atılmış hali Tablo 4.3'de yer almaktadır.

Tüm liste görüldüğü üzere iki listeden oluşmaktadır. İlk liste, Sicaklik niteliğine ait değerleri ve değerlerin sınıflarını, ikinci liste ise Nemlilik niteliğine ait değerleri ve değerlerin sınıflarını içerir. İlk listenin ilk elemanı olan [85.0, 'hayir'], Sicaklik niteliğinin ilk değerinin 85, sınıf değerinin de hayir olduğunu göstermektedir.

Tablo 4.3: Sayısal niteliklere ait değerler ve sınıfları

```
[[[85.0, 'hayir'], [80.0, 'hayir'], [83.0, 'evet'], [70.0, 'evet'], [68.0, 'evet'], [65.0, 'hayir'], [64.0, 'evet'], [72.0, 'hayir'], [69.0, 'evet'], [75.0, 'evet'], [75.0, 'evet'], [72.0, 'evet'], [81.0, 'evet'], [71.0, 'hayir']], [[85.0, 'hayir'], [90.0, 'hayir'], [86.0, 'evet'], [96.0, 'evet'], [80.0, 'evet'], [70.0, 'hayir'], [65.0, 'evet'], [95.0, 'hayir'], [70.0, 'evet'], [80.0, 'evet'], [70.0, 'evet'], [90.0, 'evet'], [75.0, 'evet'], [91.0, 'hayir']]]
```

Bir sonraki adım olarak, listeye atılan değerler, 1RD metotuna göre küçükten büyüğe doğru sıraya sokulur. Her iki nitelik için değerler sıralanmış şekilde Tablo 4.4'te gösterilmektedir.

Tablo 4.4: Sayısal niteliklere ait değerler ve sınıflarının küçükten büyüğe doğru sıralanmış şekli

```
[[[64.0, 'evet'], [65.0, 'hayir'], [68.0, 'evet'], [69.0, 'evet'], [70.0, 'evet'], [71.0, 'hayir'], [72.0, 'hayir'], [72.0, 'evet'], [75.0, 'evet'], [75.0, 'evet'], [80.0, 'hayir'], [81.0, 'evet'], [83.0, 'evet'], [85.0, 'hayir']], [[65.0, 'evet'], [70.0, 'hayir'], [70.0, 'evet'], [70.0, 'evet'], [75.0, 'evet'], [80.0, 'evet'], [80.0, 'evet'], [85.0, 'hayir'], [86.0, 'evet'], [90.0, 'hayir'], [90.0, 'evet'], [91.0, 'hayir'], [95.0, 'hayir'], [96.0, 'evet']]]
```

Sıralı listeden sadece sınıf değerlerine ait bilgiler alınır ve kırılma noktaları bulunmak üzere listelenir. Tablo 4.5'de sıralanmış listenin sadece sınıf değerleri alınmıştır.

Tablo 4.5: Sıralı listeye ait sınıf değerleri

```
[['evet', 'hayir', 'evet', 'evet', 'evet', 'hayir', 'hayir', 'evet', 'evet', 'evet', 'hayir', 'evet', 'evet', 'hayir'], ['evet', 'hayir', 'evet', 'evet', 'evet', 'evet', 'evet', 'hayir', 'evet', 'hayir', 'evet', 'hayir', 'hayir', 'evet']]
```

Programın bir sonraki adımı ise, kırılma noktalarını belirlemektir. Kırılma noktaları, sınıfın değiştiği yerleri ifade eder. Fakat her sınıfın değiştiği yer kırılma noktası olarak seçilirse, daha önceden de açıklandığı gibi, küçük parçalara ayrılmış aralıklardan fazla sayıda kural çıkmaktadır. Kırılma noktalarını saptamak için Robert C. Holte'nin sunduğu çözüm uygulanmıştır: Daha önceden belirlenen bir sayı kadar, aynı sınıfa ait olan veriler seçilerek bölümlere ayrılmalı, kırılma noktaları belirlenmelidir. Küçük veri kümeleri (50 den az örneğe sahip veri kümeleri) için üç, büyük veriler için altı sayısı uygun bulunmuştur.

Program da, bu kurala sadık kalarak, 14 adet örnek bulunan veri kümesi için sayıyı üç olarak seçmektedir. Sicaklik ve Nemlilik niteliği incelenecek olursa, sınıf sayısı en az üç olacak şekilde bölümler oluşturulmuştur. Tablo 4.6, kırılma noktalarına göre oluşan listeyi içermektedir.

Tablo 4.6: Sınıf sayısı en az üç olacak şekilde oluşan listeler

[[['evet', 'hayir', 'evet', 'evet', 'evet'], ['hayir', 'hayir', 'evet', 'evet', 'evet'], ['hayir', 'evet', 'evet', 'hayir']], [['evet', 'hayir', 'evet', 'evet', 'evet', 'evet', 'evet'], ['hayir', 'evet', 'hayir', 'evet', 'hayir', 'hayir'], ['evet']]]

Programın bir sonraki adımında, her bir listenin en çok kullanılan sınıf değeri ve eleman sayısı çıkarılmıştır. Bu işlem gerçekleşirken iki durum göz önünde bulundurulur. İlk olarak, listedeki sınıf sayısı eşit ise diğer sınıf değerlerine bakılarak anlamlı olacak şekilde hangi sınıfa ait olması gerektiğine karar verilir. Diğer bir durum ise, yanyana bulunan listeler, aynı sınıf değerlerine sahipse birleştirilir. Tablo 4.7, en çok görülen sınıfın değeri ve eleman sayısını göstermektedir.

Tablo 4.7: En çok görülen sınıfın değeri ve eleman sayısı

[[['evet', 5], ['evet', 5], ['hayir', 4]], [['evet', 7], ['hayir', 6], ['evet', 1]]]

Bahsedilen iki durumda Sicaklik niteliği için geçerlidir. Görüldüğü üzere Sicaklik niteliğine ait son listede, evet ile hayir sayısı birbirine eşit olmasına rağmen anlamlı olması açısından hayir değeri seçilmiştir. Eğer evet değeri seçilseydi, nitelik için kırılma noktası kalmayacak, Sicaklik niteliğinin her değeri evet sınıfına dahil olacaktı. Sicaklik niteliğinde görülen diğer bir durum ise, ilk iki listenin evet sınıfına ait olmasıdır. Bu yüzden birleştirilmesinde bir sakınca yoktur. Bu iki durumu içeren listenin yeni hali Tablo 4.8'de verilmiştir.

Tablo 4.8: Kırılma noktalarının son hali

[[['evet', 10], ['hayir', 4]], [['evet', 7], ['hayir', 6], ['evet', 1]]]

Her iki niteliğinde sayısal değerleri belli olduktan sonra, değer ataması yapılır. Örneğin, sıralı listedeki Sicaklik niteliğinin ilk 10 değeri için evet, geri kalan 4 değeri için hayir olacak şekilde kırılma noktası bulunur. Durum Tablo 4.9'da gösterilmektedir.

Tablo 4.9: Kırılma noktasının nitelik üzerindeki görüntüsü

Sıcaklık niteliği:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
evet	hayir	evet	evet	evet	hayir	hayir	evet	evet	evet	hayir	evet	evet	hayir

Nemlilik niteliği:

65	70	70	70	75	80	80	85	86	90	90	91	95	96
evet	hayir	evet	evet	evet	evet	evet	hayir	evet	hayir	evet	hayir	hayir	evet

Programın bu kırılma noktalarına göre verdiği çıktı aşağıdaki gibidir:

Kirilma Noktasi: 77.5
Eger Sicaklik <= 77.5 ise evet
Kirilma Noktasi: 77.5
Eger Sicaklik > 77.5 ise hayir

Kirilma Noktasi: 82.5
Eger Nemlilik <= 82.5 ise evet
Kirilma Noktasi:95.5
Eger 82.5 < Nemlilik <= 95.5 ise hayir
Kirilma Noktasi: 95.5
Eger Nemlilik > 95.5 ise evet

Program yukardaki değerleri göz önünde bulundurarak artık sayısal değerleri kategorik hale çevirebilir. Fakat program bunu yaparken Sicaklik niteliğinin 77.5'ten küçük ya da eşit değeri için evet yerine, niteliğin ismini küçük harflerle, 77.5'ten büyük değerleri için ise hayir yerine niteliğin ismini büyük harflerle yazmaktadır. Örneğin, Sicaklik değeri 65 ise sicaklik, 80 ise SICAKLIK şeklinde dosyaya yazar.

Böylece değerler arasındaki karışıklık önlenmiş olur. Hangi sınıfı küçük-büyük yapacağını seçerken, Sinif niteliğinin ilk değerini büyük, diğerini küçük yapar. Kullandığımız veri kümesindeki sınıfımızda ilk karşılaşılan hayir değeri olduğu için hayir değeri büyük harflerle dosyaya yazılır. Yeni dosyadan alınan liste Tablo 4.10'daki gibidir.

Tablo 4.10: Kategorik değerler içeren kayıt listesi

[['Gorunus', 'Sicaklik', 'Nemlilik', 'Rüzgar', 'Sinif'], ['gunesli', 'SICAKLIK', 'NEMLILIK', 'yok', 'hayir'], ['gunesli', 'SICAKLIK', 'NEMLILIK', 'var', 'hayir'], ['bulutlu', 'SICAKLIK', 'NEMLILIK', 'yok', 'evet'], ['yagmurlu', 'sicaklik', 'nemlilik', 'yok', 'evet'], ['yagmurlu', 'sicaklik', 'nemlilik', 'yok', 'evet'], ['yagmurlu', 'sicaklik', 'nemlilik', 'var', 'hayir'], ['bulutlu', 'sicaklik', 'nemlilik', 'var', 'evet'], ['gunesli', 'sicaklik', 'NEMLILIK', 'yok', 'hayir'], ['gunesli', 'sicaklik', 'nemlilik', 'yok', 'evet'], ['yagmurlu', 'sicaklik', 'nemlilik', 'yok', 'evet'], ['gunesli', 'sicaklik', 'nemlilik', 'var', 'evet'], ['bulutlu', 'sicaklik', 'NEMLILIK', 'var', 'evet'], ['bulutlu', 'SICAKLIK', 'nemlilik', 'yok', 'evet'], ['yagmurlu', 'sicaklik', 'NEMLILIK', 'var', 'hayir']]

Bu işlemin sonunda tüm sayısal değerler kategorik hale çevirilerek, yeni bir dosyaya atılmıştır. Aşağıdaki Tablo 4.11, veri kümesinin en son halini göstermektedir.

Tablo 4.11: Kategorik değerlere çevrilmiş veri kümesi

Gorunus	Sicaklik	Nemlilik	Ruzgar	Sinif
gunesli	SICAKLIK	NEMLILIK	yok	hayir
gunesli	SICAKLIK	NEMLILIK	var	hayir
bulutlu	SICAKLIK	NEMLILIK	yok	evet
yagmurlu	sicaklik	nemlilik	yok	evet
yagmurlu	sicaklik	nemlilik	yok	evet
yagmurlu	sicaklik	nemlilik	var	hayir

Gorunus	Sicaklik	Nemlilik	Ruzgar	Sinif
bulutlu	sicaklik	nemlilik	var	evet
gunesli	sicaklik	NEMLILIK	yok	hayir
gunesli	sicaklik	nemlilik	yok	evet
yagmurlu	sicaklik	nemlilik	yok	evet
gunesli	sicaklik	nemlilik	var	evet
bulutlu	sicaklik	NEMLILIK	var	evet
bulutlu	SICAKLIK	nemlilik	yok	evet
yagmurlu	sicaklik	NEMLILIK	var	hayir

4.3.2 Eğitim ve test verilerinin seçilmesi

Programın sonraki adımı ise, tüm niteliklerin kategorik olduğu dosyadaki örnekleri, eğitim ve test aşamaları için ayırmaktır. Eğitim ve test aşamaları için kullanılacak veri miktarını ayarlamak için birkaç yol vardır. Bunlardan biri olan ve programda kullanılan teknik, verilerin 3/4'ünü eğitim aşamasında, 1/4'ünü ise test aşamasında kullanmaktır. Eğitim aşaması için kullanılacak verilerin seçilmesine ait program parçası Tablo 4.12 ‘de gösterilmektedir.

Tablo 4.12: Programın eğitim verilerinin seçilmesini sağlayan kısmı

```
def form_training_data(random_number,num_of_ins,ins):

        training_data=[]
        x=random_number
        y=random_number
        two_three=round(num_of_ins*0.75)
        one_three=round(num_of_ins*0.25)
        i=0

        if (x<one_three):
                while x<(y+(two_three)):
                        training_data=training_data+[ins[x]]
```

```
                x=x+1
    else:
        while i< ((y+(two_three))%num_of_ins):
                training_data=training_data+[ins[i]]
                i=i+1
        while x<num_of_ins:
                training_data=training_data+[ins[x]]
                x=x+1
    return training_data
```

Fakat bu işlem yapılırken, sadece veri kümesinin ilk 3/4'lük kısmını eğitim aşaması için kullanmak iyi sonuç vermeyebilir. Bu yüzden [51]'de bahsedildiği gibi, veri kümesinden rastgele seçim yapıp, bu işlemi birkaç kez tekrarlamak daha iyi sonuç çıkması için uygulanan yöntemlerden biridir.

Programda ise eğitim ve test için kullanılacak veriler şu şekilde seçilmektedir: Öncelikle örnek sayısına göre birbirinden farklı 5 adet rasgele sayı üreten farklı bir program (random_number_gen_unique.py) yazılmıştır. Tablo 4.13, random_number_gen_unique.py programında veri kümesinin kayıt sayısına göre birbirinden farklı 5 sayı üreten program parçacığını göstermektedir.

Tablo 4.13: Verilen sayı aralığı için 5 farklı sayı üreten program parçacığı

```
def random_number_generator_unique(num_of_ins):
    return random.sample(range(num_of_ins),5)
```

Örneğin; üretilen [8, 1, 9, 11, 7] listesi için, ilk kayıdın 0. kayıt olduğunu varsayarak, veri kümesindeki 8. satırdan başlamak üzere 3/4'lük bir oranda kayıt seçilir. 14 kayıdı olan bir veri kümesinde 11'i eğitim, 3'ü test aşaması için ayrılır. Buna göre; aşağıdaki Tablo 4.14'de ok şeklinde gösterilen kayıtlar, eğitim verisi için seçilir ve seçilen eğitim verisi için 1R algoritması uygulanır.

Tablo 4.14: Rastgele seçilen kayıtlar

- gunesli, SICAKLIK, NEMLILIK, yok, hayir
- gunesli, SICAKLIK, NEMLILIK, var, hayir
- bulutlu, SICAKLIK, NEMLILIK, yok, evet

- yagmurlu, sicaklik, nemlilik, yok, evet
- yagmurlu, sicaklik, nemlilik, yok, evet

yagmurlu, sicaklik, nemlilik, var, hayir

bulutlu, sicaklik, nemlilik, var, evet

gunesli, sicaklik, NEMLILIK, yok, hayir

- gunesli, sicaklik, nemlilik, yok, evet
- yagmurlu, sicaklik, nemlilik, yok, evet
- gunesli, sicaklik, nemlilik, var, evet
- bulutlu, sicaklik, NEMLILIK, var, evet
- bulutlu, SICAKLIK, nemlilik, yok, evet
- yagmurlu, sicaklik, NEMLILIK, var, hayir

Rastgele seçilen diğer 1, 9, 11, 7 değerleri için de aynı işlem tekrar edilir. 5 kez uygulanan işlem sonucunda en yüksek doğruluk oranına sahip olan rastgele sayı seçilerek, kurallar oluşturulur. Oluşturulan bu kurallara, seçilen rastgele sayıya ait test verileri uygulanarak doğruluk ya da hata oranı tespit edilir.

4.3.3 1R algoritmasının uygulanması

Eğitim aşamasında kullanılacak verilerin belirlenmesinden sonra, 1R algoritması adım adım uygulanır. Eğitim verileri için oluşturulan rastgele listenin yukarıda verdiğimiz örnek ile aynı olduğu düşünülürse ([8, 1, 9, 11, 7]), ilk rastgele sayı olan 8 için frekans hesabı aşağıdaki gibi sonuçlanmaktadır. Aşağıdaki Tablo 4.15'teki listede, sağdan sola doğru her değerin eğitim verisi içinde kaç defa görüldüğü, en sık karşılaşılan sınıf sayısı, sınıfın değer bilgisi ve değerin isimini içermektedir.

Tablo 4.15: Seçilen eğitim verilerine göre frekans değerleri

[['gunesli', 'hayir', 2, 4], ['yagmurlu', 'evet', 3, 4], ['bulutlu', 'evet', 3, 3], [SICAKLIK', 'hayir', 2, 4], ['sicaklik ', 'evet', 6, 7], ['NEMLILIK', 'hayir', 3, 5], ['nemlilik', 'evet', 6, 6], ['yok', 'evet', 6, 7], ['var', 'evet', 2, 4], [8.0, 8.0, 9.0, 8.0]]

Örneğin; yagmurlu değeri evet sınıfında 3, hayır sınıfında 1 olmak üzere toplam 4 defa görülmüş ve en sık karşılaşılan sınıf değeri evet seçilmiştir. var değeri ise evet sınıfında 2, hayır sınıfında 2 olmak üzere toplam 4 defa görülmüştür. evet ve hayır sayısı eşit olduğu için, program ilk karşılaştığı evet değerini seçmiştir

Yukarıdaki liste, ['yok', 'evet', 6, 7], ['var', 'hayir', 2, 4] değerlerine sahip olsaydı sorun teşkil etmezdi. Çünkü yok değeri için evet, var değeri için hayır sınıfı seçilmiş olacaktı. Fakat liste ['yok', 'evet', 6, 7], ['var', 'evet', 2, 4] şeklindedir. Ruzgar niteliğinin iki değeri için de sınıf aynıdır ve bu yüzden anlamsız kurallar üretilecektir. Böyle bir durumda, program ilk iş olarak o niteliğe ait tüm değerlerin sınıflarını kontrol eder. Sınıf değerleri farklı ise sorun yoktur. Sınıf değerleri aynı ise (['yok', 'evet', 6, 7], ['var', 'evet', 2, 4] gibi), aynı durumda olan kaç tane değer olduğuna bakar. Bu sayı bir tane ise, diğer değerin sınıfının tersi atanır (['yok', 'evet', 6, 7], ['var', 'hayir', 2, 4] gibi).

Bu sayı birden fazla ise ilkine ilk sınıf değeri, ikincisine diğer sınıf değeri, üçüncüsüne ilk sınıf değeri atanarak devam eder. Yukardaki Tablo 4.15'in son hali aşağıdaki Tablo 4.16 gibidir:

Tablo 4.16: Düzelmiş frekans değerleri listesi

[['gunesli', 'hayir', 2, 4], ['yagmurlu', 'evet', 3, 4], ['bulutlu', 'evet', 3, 3], ['SICAKLIK', 'hayir', 2, 4], ['sicaklik ', 'evet', 6, 7], ['NEMLILIK', 'hayir', 3, 5], ['nemlilik', 'evet', 6, 6], ['yok', 'evet', 6, 7], ['var', 'hayir', 2, 4], [8.0, 8.0, 9.0, 8.0]]

Değerlerin frekansları bulunduktan sonraki adım ise yüksek frekans değerini seçerek, kural oluşturmaktır. Oluşturulan kuralların doğruluk değerleri hesaplanır ve en yüksek doğruluk değerine sahip nitelik seçilir ve o niteliğe ait kurallar kullanılır. Aşağıda, frekans değerlerine göre kurallar oluşturulmuş ve bir niteliğe ait kuralların doğruluk değerleri hesaplanmıştır. Tablo 4.17, kuralların doğruluk değerini göstermektedir.

Tablo 4.17: Kuralların doğruluk değeri

Nitelik	Kurallar	Doğruluk Değeri
Gorunus		(8/11)
	Eğer gunesli ise hayir (2/4)	
	Eğer bulutlu ise evet (3/3)	
	Eğer yagmurlu ise evet (3/4)	
Sicaklik		(8/11)
	Eğer sicaklik ise evet (6/7)	
	Eğer SICAKLIK ise hayir (2/4)	
Nemlilik		(9/11)
	Eğer nemlilik ise evet (6/6)	
	Eğer NEMLILIK ise hayir (3/5)	
Ruzgar		(8/11)
	Eğer var ise hayir (2/4)	
	Eğer yok ise evet (6/7)	

Gorunus dogruluk = 8.0 11 0.727272727273
Sicaklik dogruluk = 8.0 11 0.727272727273
Nemlilik dogruluk = 9.0 11 0.818181818182
Ruzgar dogruluk = 8.0 11 0.727272727273

Buna göre en yüksek doğruluk değeri 9/11'dir. Bu değere sahip nitelik Nemlilik niteliğidir. Buna göre program tarafından oluşturulan kurallar aşağıdaki gibidir:

Eger Nemlilik NEMLİLİK ise hayir (3 / 5)

Eger Nemlilik nemlilik ise evet (6 / 6)

Programın doğruluk değerlerini yazan ve kural üreten program parçası aşağıdaki Tablo 4.18'de verilmiştir.

Tablo 4.18: Doğruluğu ve kuralları gösteren program parçası

```
def one_rule(rule,value,att_name,total,c):
        listem=[]
        i=0
        while i<len(att_name)-1:
                sum=0.0
                print att_name[i],"dogruluk =",rule[len(rule)-
1][i],total,rule[len(rule)-1][i]/total
                listem=listem+[[att_name[i],rule[len(rule)-1][i]/total]]
                i=i+1
        i=0
        while i<(len(listem)-1):
                j=0
                while j<(len(listem)-1-i):
                        if (listem[j+1][1] > listem[j][1]):
                                tmp = listem[j]
                                listem[j] = listem[j+1]
                                listem[j+1] = tmp
                        j=j+1
                i=i+1
        deneme=[]
        index=att_name.index(listem[0][0])
        i=0
        while i<len(value[att_name.index(listem[0][0])]):
                j=0
                while j<len(rule):
                        if(value[index][i]==rule[j][0]):
                                print "Eger",listem[0][0],value[index][i],
"ise",rule[j][1],"  ", "(",rule[j][2],"/",rule[j][3],")"
                               deneme=deneme+[[listem[0][0],value[index][i],
rule[j][1]]]
                        j=j+1
                i=i+1
        return deneme
```

Programın bir sonraki adımı ise, test verilerini kullanarak hata oranını hesaplamaktır. 14 kayıdın dörtte biri (3 tanesi) test verisidir. Rastgele yapılan 5 denemenin her birinde, eğitim verileri ile kurallar oluşturulur ve test verileri oluşturulan kurallar

üzerinde denenerek hata oranları bulunur. Program, 5 denemedeki en yüksek doğruluk oranını seçer. [8, 1, 9, 11, 7] listesinde 8 sayısı ile oluşan kurallar ve hata oranları aşağıdaki gibidir.

1 . kural HATA = 0 / 1
2 . kural HATA = 1 / 2
0.333333333333

Buna göre, ilk kurala uyan bir adet test verisinde hata görülmemektedir. İkinci kurala uyan iki örnekten bir tanesi yanlıştır. Toplam üzerinden düşünüldüğünde hata oranı 0.33'dür. Uygulama tekrarlanarak, doğruluk oranı en yüksek olan kurallar seçilir. Oluşan kurallar sayesinde, yeni verilerle tahmin etme işlemi yapılır.

5. İKİNCİ KURALIN OLUŞTURULMASI

5.1. İki Kural Çıkarımı

Holte'nin makalesinde bahsettiği gibi, 1 ya da 2 niteliği sınıflandırabilecek durumlarda karmaşık hipotezlere gerek yoktur. Karmaşık kurallar üreten sistemlerin ek olarak karmaşıklığını da doğrulamak gerekmektedir. Öğrenecek olan sistem, basit hipotezlerin bulunduğu küçük uzayı aramalıdır. Uzay küçük olursa, uzayda yön belirlemek problem yaratmaz. Bu yüzden Holte geleneksel yöntemlerden farklı olarak yeni bir araştırma yöntemine doğru yönelmiştir. Makine öğrenmesi araştırmalarının bir amacının, üretilen kuralların basitlik ve doğruluğunu arttırmak olduğu düşünülürse, bu amaç ile oluşturulan "İlk Basitlik" yöntemi, Holte'nin makalesinde vurgulanmıştır [35].

1R algoritması üzerinde de deneyler ve değişiklikler yapılmış ve yapılan bazı değişiklikler doğruluğun artmasını sağlamıştır. Örneğin, 1R algoritmasının uygulanmasında, sayısal verileri kategorik veriler haline çevirirken ortaya atılan yeni yaklaşımın gayet iyi sonuçlar verdiği belirtilmiş, kayıp veriler için sunulan benzer bir yaklaşımın uygulanabilirliğinden bahsedilmiştir [34].

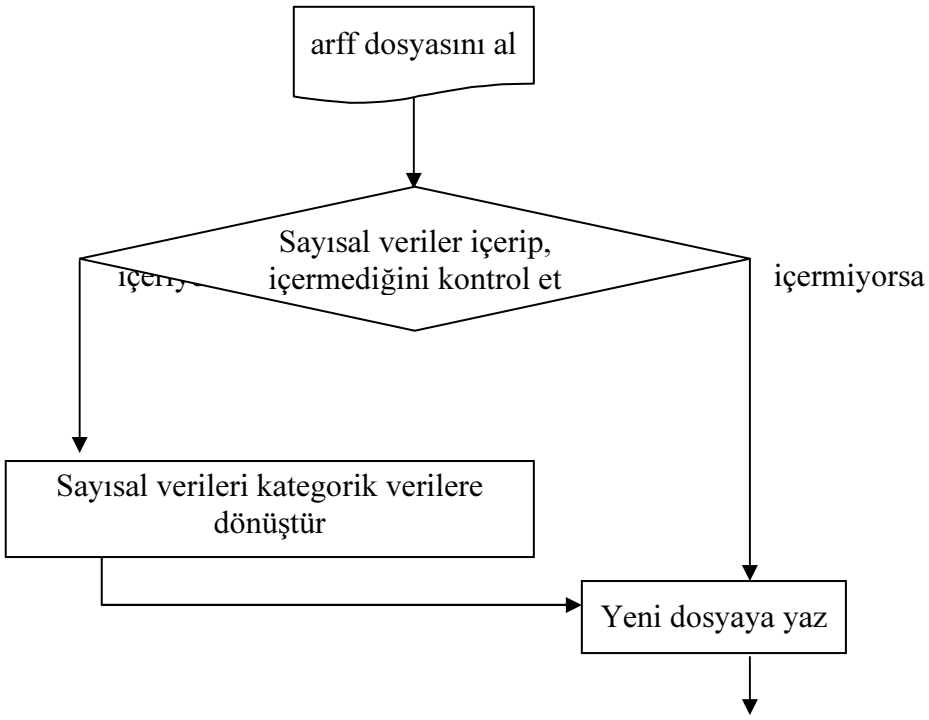
Bunun yanında Holte birçok temel değişiklik ile 1R'ın arama uzayının genişletilip, daha karmaşık kurallar üretilebileceğine makalesinde yer vermiştir. İki nitelik içeren kuralların bunlardan bir tanesi olduğunu vurgulamıştır [35].

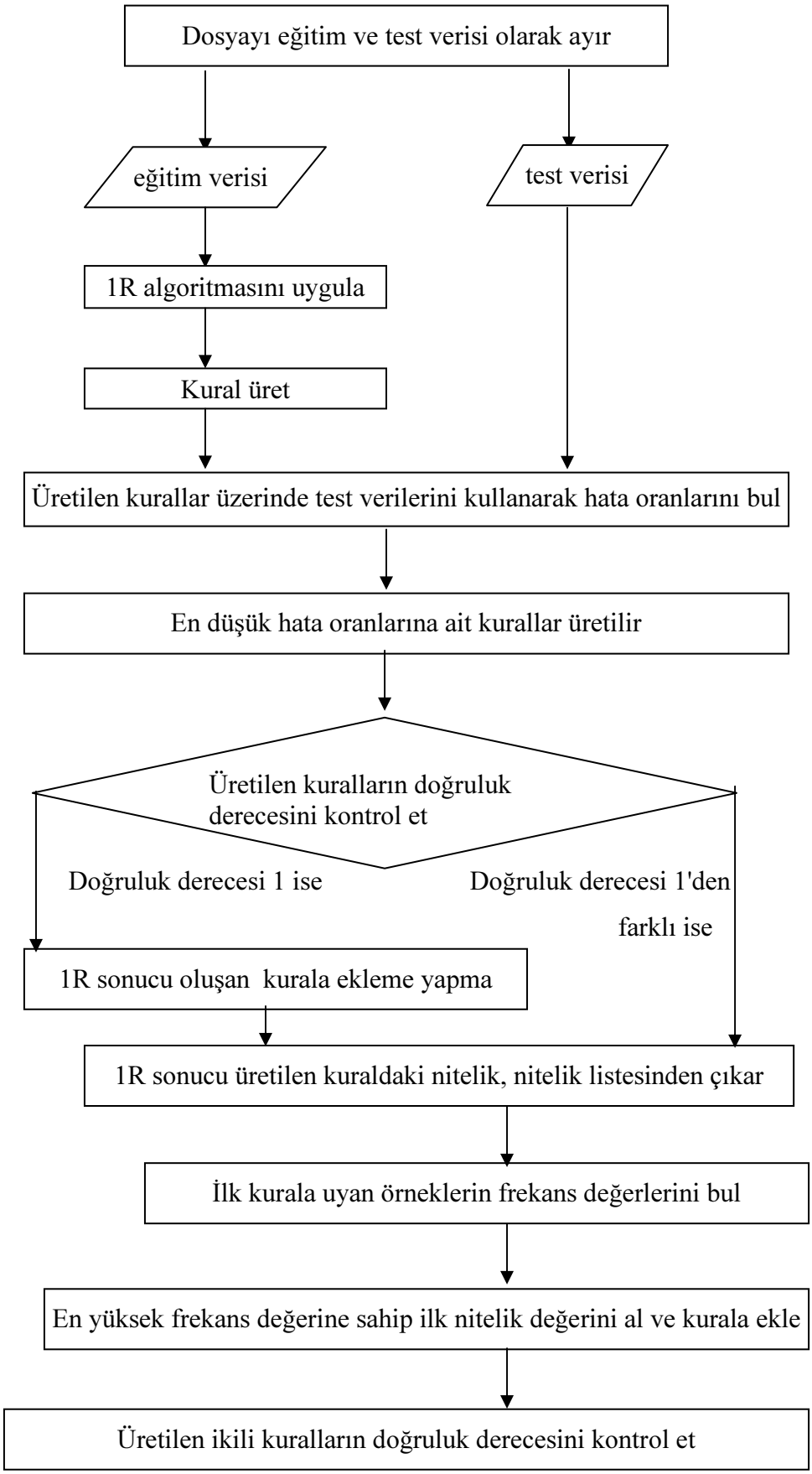
John George ve diğ., nitelikleri tek olarak düşünmenin dışında, nitelik kombinasyonlarının oluşturulabileceğine dair fikirlerini makalesinde ortaya koymuştur [81]. 1R, nitelik değerlendirilmesindeki etkisinden dolayı, nitelik kombinasyonlarının gösterilmesi için kullanılmış ve bu işleme makalede 2R adı verilmiştir. 2R sürecinde, nitelik çiftlerinin bağlanması ile yeni niteliklerin oluşturulması sağlanmış ve 1R, yeni oluşan veri kümesinde tekrar çalıştırılmıştır

[34]. Hava.arff veri kümesi için, yeni nitelik Gorunus ve Ruzgar niteliğinin birleşimi şeklinde olabilir. Böylece yeni nitelik, gunesli-yok, gunesli-var, bulutlu-yok, bulutlu-var, yagmurlu-yok, yagmurlu-var değerlerini içerir.

Holte ve diğ. tarafından oluşturulan T2 algoritması ise iki seviyeli ağaç algoritması olup, PAC (probably approximately correct) öğrenme algoritmasıdır. T2'nin oluşumda kullanılan ilk seviyeli ağacın hipotez sınıfı, Holte'nin geliştirdiği 1R algoritmasını kullanır [82].

Tez kapsamında da, 1R algoritması kullanılarak yukarıda bahsedilen T2 algoritmasının bir parçası gibi görülebilen, benzer bir uygulama geliştirilmiştir. İki kural üreten 2R algoritması Python programlama dili kullanılarak oluşturulmuştur. Önceki bölümlerde görüldüğü üzere 1R algoritması, doğruluk değeri en yüksek niteliğe ait kurallar üretmektedir. 2R algoritmasında ise, 1R sonucunda oluşan kurallardaki niteliğin dışındaki nitelikler kullanılarak, ikinci bir nitelik bulunur ve ilk kurala eklenir. Yukarıda da bahsedildiği gibi T2 algoritmasına benzer bir işleyişi olsada, T2 algoritması kadar detaya inilmemiştir. Programın işleyişi Şekil 5.1'de verilmiştir.





Şekil 5.1: 2R Programının akış diyagramı

2R algoritmasının işleyişindeki ilk adım, 1R algoritması sonucu oluşan kuralın doğruluk değerini kontrol etmektir. Eğer oluşan kuralın doğruluk değeri 1 ise, o kural için ikinci bir kural üretmeye gerek yoktur. Örneğin, program önceki bölümde oluşan Eger Nemlilik nemlilik ise evet (6 / 6) kuralına ikinci bir kural eklemez. Çünkü kuralın doğruluk değerinin 1 olması, nitelik değerinin sınıflarının aynı olduğu anlamına gelmektedir. O yüzden, program ikinci kuralı eklemez ve o kuralı geçer.

Eğer oluşan kuralın doğruluk değeri 1'den farklı ise, kural tam anlamıyla doğru değildir. Bu yüzden kurala ait nitelik, nitelik listesinden çıkarılır. Örneğin Eger Nemlilik NEMLILIK ise hayir (3 / 5) kuralı için, Nemlilik niteliği, nitelik listesinden çıkarılır ve bu kurala uyan kayıtlar veri kümesinden seçilir. Bir önceki bölümde, ilk kurala uyan üç kayıt Tablo 5.1'de gösterilmiştir.

Tablo 5.1: Kurala uyan kayıtlar

[[['gunesli', 'SICAKLIK', 'yok', 'hayir'], ['gunesli', 'SICAKLIK', 'var', hayir'], ['yagmurlu', ' sicaklik ', 'var', 'hayir']]]

Bu kurala uyan kayıtlar 1R algoritmasında da kullanılan frekans bulma fonksiyonuna gönderilmektedir. Fonksiyon, yeni listedeki her bir değerin frekansını döndürür. En yüksek frekansa ait niteliğin değeri seçilerek ilk kurala eklenir. Eğer en yüksek frekans değerine sahip birden fazla değer varsa, program ilk değeri seçer. Eger Nemlilik NEMLİLİK ise hayir (3 / 5) kuralı için oluşan frekans değerleri Tablo 5.2'deki gibidir.

Tablo 5.2: Kalan değerlerin frekansları

[['gunesli', 'hayir', 2, 2], ['yagmurlu', 'hayir', 1, 1], ['bulutlu', 'hayir', 0, 0], ['SICAKLIK', 'hayir', 2, 2], ['sicaklik ', 'hayir', 1, 1], ['yok', 'hayir', 1, 1], ['var', 'hayir', 2, 2], [3.0, 3.0, 3.0]]

En yüksek frekans değeri 2'dir ve gunesli, SICAKLIK ve var değerlerine aittir. Program ilk karşılaştığı en yüksek frekans değerini seçtiği için Gorunus niteliğine ait olan gunesli değeri, ikinci kural olarak seçilir. Oluşan kurallar aşağıdaki gibidir.

Eger Nemlilik NEMLILIK ve Gorunus gunesli ise hayir
Eger Nemlilik nemlilik ise evet

Test örnekleri kullanılarak ortaya çıkan hata oranı aşağıdaki gibidir. 3 kayıt bulunan test verisinde ilk kurala (Eger Nemlilik NEMLILIK ve Gorunus gunesli ise hayir) uyan 1 adet örnekte hata bulunmamaktadır. İkinci kurala yeni bir kural eklenmediği için hata oranı aynıdır. Toplam hata oranı ise 0.33'dür.

1 . kural HATA = 0 / 1
2 . kural HATA = 1 / 2
1.0 3.0
0.333333333333

Programda, üretilen iki kurala ait hata oranlarını bulmak için oluşturulmuş program parçacığı Tablo 5.3'deki gibidir.

Tablo 5.3: İki kurala ait hataları bulan program parçacığı

```
def error_detection(all,att_sayisi,test_data,att_name):

        temp=0.0
        error=0.0
        sum=0.0
        i=0
        while i<len(all):

                false=0
                total=0
                if(len(all[i])>3):
                        index1=att_name.index(all[i][0])
                        index2=att_name.index(all[i][2])
                        j=0
                        while j<len(test_data):
                                if(all[i][1]==test_data[j][index1]):

                                        if(all[i][3]==test_data[j][index2]):
                                                total=total+1

                                        if(all[i][4]!=test_data[j][att_sayisi-1]):
                                                false=false+1

                                j=j+1
```

```
                print i+1,".","kural HATA =",false,"/",total
                temp=temp+false
                sum=sum+total
        else:
                index1=att_name.index(all[i][0])
                j=0
                while j<len(test_data):
                        if(all[i][1]==test_data[j][index1]):
                                total=total+1
                                if (all[i][2]!=test_data[j][att_sayisi-1]):
                                        false=false+1
                        j=j+1
                print i+1,".","kural HATA =",false,"/",total
                temp=temp+false
                sum=sum+total

        i=i+1

print temp,sum
if (temp!=0.0 or sum!=0.0):
        error=temp/sum
        print error

elif(temp==0.0 and sum==0.0):
        error=0.0
        print error
return error
```

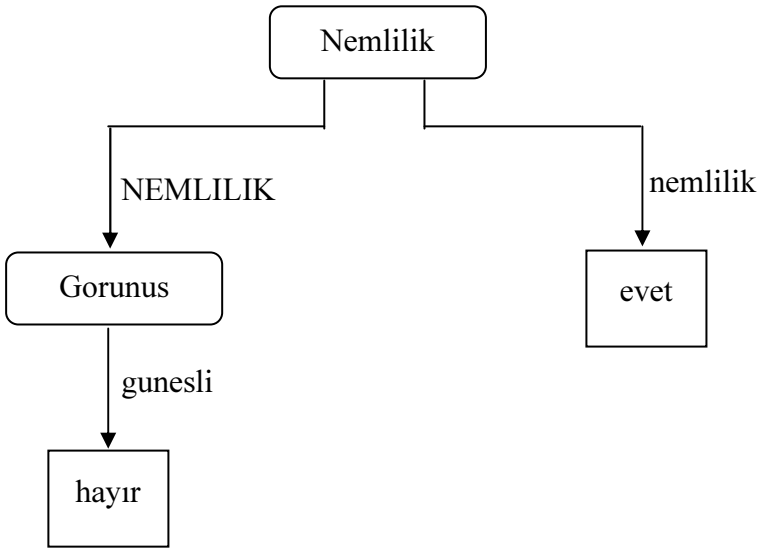
İki kural içeren bir yapının oluşturulması ile, koşul sayısının artması sağlanarak, karar verme aşamasında bir nitelik yerine iki niteliğin göz önünde bulundurulması sağlanmıştır. Çünkü bazı durumlarda bir nitelik karar vermek için yeterli olmayabilir. Örneğin, bir bankanın hayat sigortası promosyonu için, hedef olarak gördüğü müşteri profili birden fazla kriterlere sahip olabilir. Bunun için üretilen kural sonucunda;

Eğer Cinsiyeti= Bayan ve 19 <= Yaş <= 43 ise Hayat Sigortası Promosyonu =Evet gibi bir kuralın çıkması işine yararken;
Eğer Cinsiyeti= Bayan ise Hayat Sigortası Promosyonu = Evet kuralı yeterli olmayabilir.

Bu gibi durumlar için iki kural üretimi yararlı olabilmektedir. Fakat, bir yandan 1R algoritması genişlediği için karmaşıklık oranı artmış olur. Çünkü varolan 1R

algoritmasına ek olarak, ikinci kuralın oluşması için fazladan hesaplama yapılır. Bu da uzayı genişletmekle kalmayıp, karmaşıklığın artmasına sebep olur.

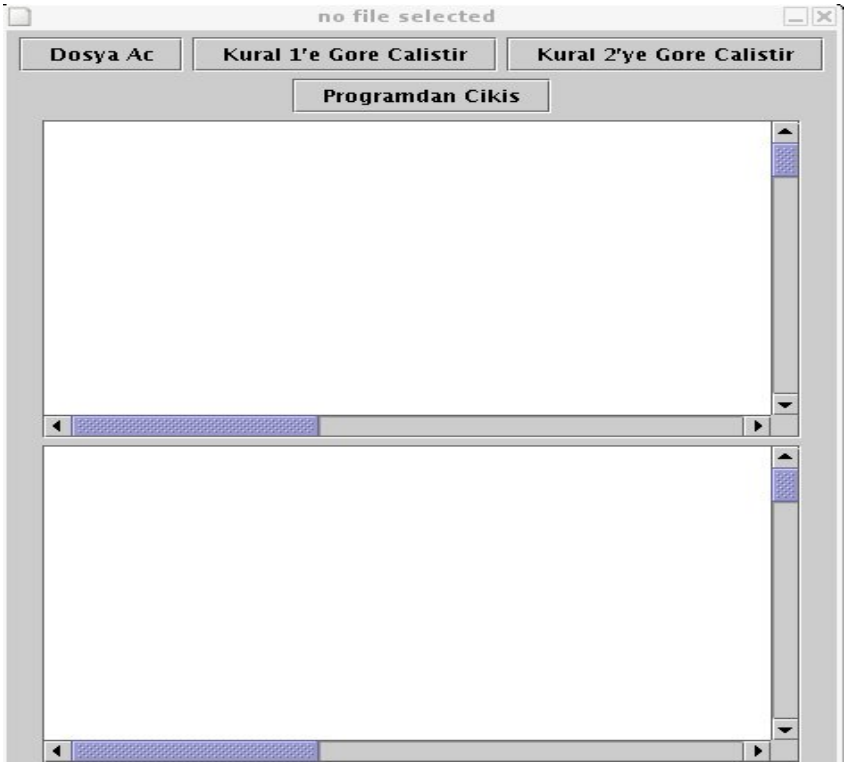
Tezin ikinci bölümünde, karar ağaçlarından kural üretileceği gibi doğrudan da üretilebildiğinden bahsedilmişti. Bu iki algoritmanın sonucunda doğrudan kural çıkarımı yapılmıştır. Karar ağaçlarından kurallar üretilebilceği gibi, oluşan kurallar kullanılarak ağaç yapısı da oluşturulabilir. Şekil 5.2'de, oluşan kurallar ağaç yapısı olarak gösterilmektedir.



Şekil 5.2: Oluşan iki kuralın ağaç yapısı olarak sunulması

5.2 Programın Arayüzünün Açıklanması

Python'da geliştirilen program sonucunda oluşan kuralların ve hata oranlarının kullanıcıya sunulması için, Java programlama dilinde basit bir yazılım geliştirilmiştir. Program kodu Ek-B'de bulunmaktadır. Yazılım arayüzünün görüntüsü Şekil 5.3.'deki gibidir.



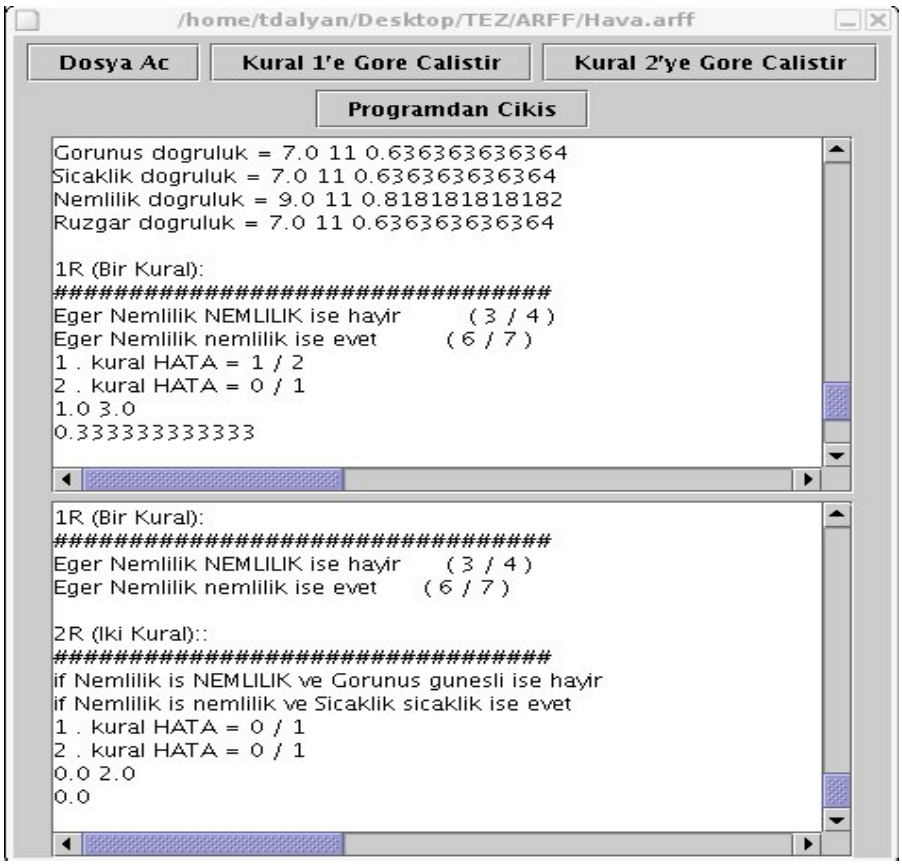
Şekil 5.3: Programın arayüzü

Tezin ilerleyen bölümlerinde, oluşturulan üç programı (1R_kural.py, 2R_kural.py, random_number_gen_unique.py) barındıran ve tarafımca geliştirilen bu yazılım, PyRuler olarak adlandırılmıştır. Bundan sonra bu isim ile anılacaktır. PyRuler'ın işleyişi şu şekildedir: arayüz sayesinde, programın bulunduğu klasörden .arff dosyası seçer. Seçilen dosya ve hepsi birbirinden farklı rastgele sayılardan oluşan bir liste, Python programında hazırlanmış 1R_kural.py ve 2R_kurallar.py programlarına gönderilir. Oluşturulan kurallar arayüz sayesinde kullanıcıya sunulur. Aşağıda programın derlenmesinden, Python programına gönderilen parametrelere kadar arayüzün çalışması gösterilmiştir.

tdalyan@tugbahome:~/Desktop/TEZ/ARFF$ javac PyRuler.java
tdalyan@tugbahome:~/Desktop/TEZ/ARFF$ java PyRuler
Gelen liste: [4, 1, 9, 13, 7]
Programa Giden Liste: 4 1 9 13 7
Giden komut: python 1R_kural.py /home/tdalyan/Desktop/TEZ/ARFF/Hava.arff 4 1 9 13 7

Giden komut: python 2R_kural.py /home/tdalyan/Desktop/TEZ/ARFF/Hava.arff 4 1 9 13 7

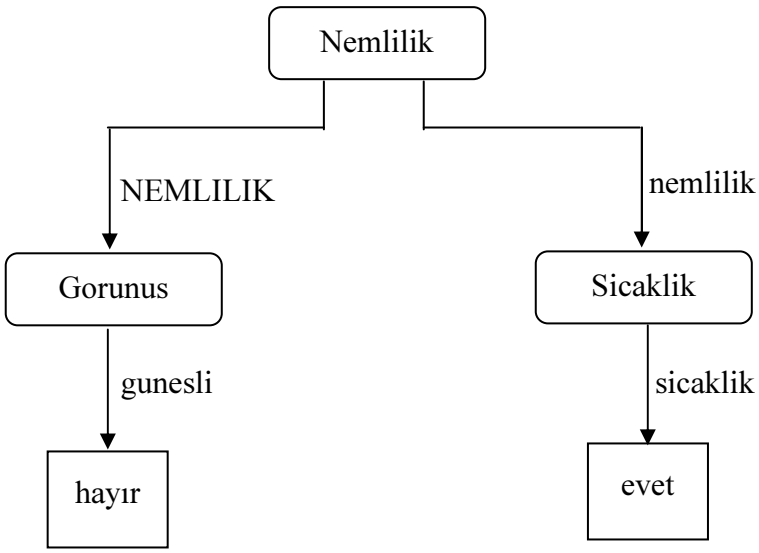
PyRuler, Kural 1'e Gore Calistir düğmesine bastığımızda, 1R_kural.py programını, Kural 2'ye Gore Calistir düğmesine bastığımızda ise 2R_kurallar.py programı çalıştırır. Şekil 5.4, Hava.arff dosyasının iki algoritmaya göre sonuçları yer almaktadır. Programdan Cıkıs düğmesi ile PyRuler kapanır.



Şekil 5.4: Kuralların ve hata oranlarını içeren programın arayüzü

Görüldüğü üzere çıkan kurallar ve hata oranları önceki bölümde çıkanlardan farklıdır. Çünkü eğitim verisinin oluşması için gönderilen rastgele sayı listesi bir öncekinden farklıdır. Daha önceki bölümlerde de bahsedildiği gibi PyRuler, en etkili kuralları (doğruluk oranı yüksek, hata oranı düşük) oluşturuluncaya kadar çalıştırılır. Şekil 5.4'de görüldüğü gibi, ikili oluşturulan kurallardaki hata oranı 0'dır.

Oluşan bu kurallar karar ağacı yapısına çevrildiğinde, Şekil 5.5'deki gibi bir ağacı oluşur.



Şekil 5.5: Arayüzde oluşan iki kurallı ağaç yapısı

6. HATA ORANLARI

6.1. Hata Oranlarının Gösterilmesi

Uygulanan iki algoritmanın işleyişini görmek için, makine öğrenmesi araştırmalarında kullanılan yedi farklı veri kümesi seçilmiştir [51,83]. Programın, kayıp verileri kabul etmemesi ve sadece iki sınıfta çalıştığı göz önünde bulundurularak bazı veri kümeleri üzerinde değişiklikler yapılmıştır. Tablo 6.1, veri kümelerinin kayıt ve nitelik sayısını içermektedir. Veri kümelerine ait detaylı bilgi Ek-C'de verilmiştir.

Tablo 6.1: Veri kümeleri ve özellikleri

Veri Kümeleri	Kayıt Sayısı	Nitelik Sayısı
Weather	14	5
Postoperative	83	9
Tae (2 sınıf)	100	6
Haberman	123	4
Bank	300	9
Pima	500	9
Contraceptive	763	10

Her seferinde veri kümesinin rastgele bir parçası seçildiğinden farklı hata oranları oluşmuş ve bunlar içinden en düşük hata oranına sahip kuralların seçilmesi için işlem tekrar edilmiştir. Buna göre ortaya çıkan en düşük hata oranları aşağıdaki Tablo 6.2.'de verilmiştir.

Tablo 6.2: Veri kümelerinin hata oranları

	Veri Kümeleri	Weather	Postoperative	Tae	Haberman	Bank	Pima	Contraceptive
1R	Hata Sayısı	1	4	4	1	16	23	54
	Test Verisi Sayısı	3	21	23	31	75	125	191
	Hata Oranı	0.33	0.19	0.17	0.03	0.21	0.18	0.28
2R	Hata Sayısı	0	4	2	1	6	10	51
	Test Verisi Sayısı	2	19	22	31	52	98	188
	Hata Oranı	0	0.21	0.09	0.03	0.11	0.1	0.27

Tablo 6.2, temel olarak veri kümelerinin iki algoritma sonucunda, test veri üzerindeki yaklaşık olarak en düşük hata oranlarını göstermektedir. Örneğin, Weather.arff dosyası 1R için, 3 adet test kayıdından, sadece 1 tanesinin hatalı çıktığını göstermektedir. Bunun anlamı, 3 kayıt içinde üretilen kurallara uymayan 1 adet örnek vardır. 2R'daki kayıt sayısı 1R'ınkinden farklıdır. Çünkü 2R, 1R algoritmasına uyan kurallar üzerinden çalışır. O yüzden kayıt sayısı çoğu durumda daha az olmaktadır.

Hata oranları incelendiğinde ise, 2R ile oluşan kuralların hata oranları eşit ya da daha düşük çıkmıştır. 2 metodun hangi durumlarda iyi sonuç verdikleri ya da birbirlerine göre doğruluk oranlarının karşılaştırılması tez kapsamında açıklanmamıştır. Bu tip bir karşılaştırma yapabilmek için, iki algoritmayı etkileyen parametreleri (nitelik sayısı, sayısal nitelik sayısı, kayıt sayısı ...) detaylı olarak incelemek gerekmektedir. Bu çalışamada ise buna yer verilmemiştir. Hata oranlarına bakılarak program sonuçlarının kötü olmadığı ve büyük hatalar çıkmadığı sonucuna varılabilir.

Dördüncü bölümde, bahsi geçen Weka Makine Öğrenmesi projesi kapsamında geliştirilen araç, kural üretmek için birçok tekniğe sahiptir. Bunlardan bir tanesi ise

1R algoritmasıdır. Birçok bilim adamı Weka sınıflandırıcısını kullanarak veri kümeleri üzerinde deneysel çalışmalar yapmaktadır.

Tezde kullanılan 7 veri kümesinden 6 tanesi Weka'nın 1R algoritması ile denenmiş, 1 tanesi hariç birebir ya da yaklaşık sonuçlar çıkmıştır. Örneğin, sonuçları birbirine çok yakın veri kümesinden Haberman.arff'nin, 2 programda da ürettiği kurallar aşağıdaki Tablo 6.4.'de gösterilmiştir.

Tablo 6.3: Programların sonuçları

Weka Aracının Sonucu	PyRuler Sonucu
=== Run information === Scheme: weka.classifiers.rules.OneR -B 6 Relation: haberman Instances: 124 Attributes: 4 Age Opr_year #_of_pos_ax_nodes class Test mode: split 75% train, remainder test === Classifier model (full training set) === #_of_pos_ax_nodes: < 17.5 -> 1 >= 17.5 -> 2 (97/124 instances correct)	Kirilma Noktasi: 17.5 Eger #_of_pos_ax_nodes <= 17.5 ise 1 Kirilma Noktasi: 17.5 Eger #_of_pos_ax_nodes > 17.5 ise 2 1 = #_OF_POS_AX_NODES 2 = #_of_pos_ax_nodes ################################## Age dogruluk = 65.0 93 0.698924731183 Opr_year dogruluk = 65.0 93 0.698924731183 #_of_pos_ax_nodes dogruluk = 67.0 93 0.720430107527 1R Sonuc: ################################## Eger #_of_pos_ax_nodes #_OF_POS_AX_NODES ise 1 (63 / 87) Eger #_of_pos_ax_nodes #_of_pos_ax_nodes ise 2 (4 / 6) 1 . kural HATA = 1 / 29 2 . kural HATA = 0 / 2 1.0 31.0 0.0322580645161

PyRuler'ın ürettiği sonucun gerekli kısmı Tablo 6.3'te gösterilmektedir. PyRuler, önce sayısal verileri kategorik hale çevirir. Tüm nitelikler sayısal değerler içerdiği için hepsine aynı işlem uygulanır. Tablo 6.3, sadece #_of_pos_ax_nodes niteliğinin

çevirimini içerir. #_of_pos_ax_nodes için kırılma noktası 17.5 bulunur. 17.5'tan küçük ve eşit değerler için niteliğin büyük harfler ile (#_OF_POS_AX_NODES=1), 17.5'ten büyük değerler için ise küçük harfler (#_of_pos_ax_nodes=2) ile yazar. Tüm niteliklerin doğruluk değeri hesaplanır. En yüksek doğruluk değerine sahip #_of_pos_ax_nodes niteliği seçilerek, kural oluşturulur.

Her iki programda da seçilen nitelikler ve kurallar aynıdır. Tek farkları, Weka 17.5 değeri için 2'yi, PyRuler ise 1 sınıf değerini seçmektedir.

7. SONUÇLAR ve ÖNERİLER

Son yıllarda, bilişim teknolojilerinin ilerlemesi ve gün geçtikçe gelişen veri toplama araçları sayesinde veri oluşturma, veri toplama ve verilere ulaşma süreci hızlanmıştır. Bu süreç, sektördeki birçok alan için önem taşımaktadır. Günümüzde verilerin, geleceğe yönelik tahminler ve tanımlamalar yapmak için kullanıldığı düşünülürse, bu verilerden çıkarımlar yapmanın önemi yadsınamaz bir gerçektir. Çünkü veri tek başına bir anlam ifade etmez. Bu noktada devreye giren makine öğrenmesi alanı, bu verilerin nasıl toplandığı ya da kullanılacak formata nasıl dönüştürüldüğü ile ilgilenmez. İlgilendiği, verileri kullanarak, geleceğe yönelik tahminleri ya da tanımları en iyi şekilde nasıl çıkarabileceğidir.

Makine öğrenmesi tekniklerinden bir tanesi olan kural çıkarımı, anlaşılması kolay olduğundan dolayı insanların tercih ettiği bir yöntemdir. Birçok bilim adamının katkıları ile geliştirilen bu yöntem, günümüzde önemini sürdürmektedir.

Makine öğrenmesi tarafından üretilecek bu kuralların basitlik ve doğruluğunu arttırmak, makine öğrenmesinin en önemli amaçlarından bir tanesidir. Bu amaç ile Robert Holte tarafından tasarlanan 1R algoritması, en yüksek doğruluk değerine sahip niteliğe ait kurallar üretir. Basit olduğu kadar etkili sonuçlar vermesinden dolayı 1R algoritması birçok araştırmada yer almaktadır.

Bu çalışmada, 1R algoritması detaylı olarak incelenmiş ve Python programlama dili ile uygulanmıştır. Uygulama sonucu ortaya çıkan kurallar ve hata oranlarına tezde yer verilmiştir. Veri kümeleri, Weka Makine Öğrenmesi Aracı ile denenmiştir. Sonuçlar PyRuler ile karşılaştırıldığında, uygulama sonucunun yeterli başarıya sahip olduğu görülmüştür.

Bu uygulama ile sadece varolan bir algoritma gerçekleştirilmiştir. Başka çalışmalarda, algoritma üzerinde yapılan değişikliklerin, algoritmanın doğruluğunu

arttırabileceğinden bahsedilse de, tez kapsamında 1R algoritması üzerinde herhangi bir değişiklik yapılmamıştır.

1R gibi basit kural öğrenme sistemleri, karışık kurallar üreten sistemlere alternatif olarak gösterilmektedir. Çünkü karışık kurallar üreten sistemlerin ek olarak karmaşıklığını da doğrulamak gerekmektedir. Bu yüzden geleneksel yöntemlere alternatif olarak gösterilen "İlk Basitlik" yöntemine son yıllarda sıkça başvurulmaktadır. Makine öğrenmesinin bir amacının da üretilen kuralların basitlik ve doğruluğunu arttırmak olduğu düşünülürse, yöntem son derece mantıklıdır. Basit sistem üzerinde ufak değişiklikler yaparak uzayı genişletmek (iki kural oluşturmak gibi) önerilen yöntemlerden bir tanesidir.

Bu fikirden yola çıkılarak, proje süresince gerçekleştirilen diğer çalışmada ise, 1R algoritmasının ürettiği kurallar kullanılarak ikinci kuralın elde edilmesi sağlanır. İlk kuralın doğruluk değerine bakılarak, ilk kurala ait niteliğin dışındaki nitelik değerlerinden frekansı yüksek olan değer ikinci kural olarak atanır. Bu yöntem T2 algoritmasının işleyişine benzese de onun kadar kapsamlı değildir. İki kural (2R) algoritması da, Python'da gerçekleştirilmiştir. İki kural içeren bir yapının olması, koşulun artmasını sağlayarak, karar verme aşamasında bir nitelik yerine iki niteliğin göz önünde bulundurulmasını sağlar. Çünkü bazı durumlarda bir nitelik karar vermek için yeterli olmayabilir. Bir yandan ise, 1R algoritması genişlediği için karmaşıklık oranı artmış olur.

Projenin gelişim süreci kısaca şöyle özetlenebilir; Varolan 1R algoritması Python programlama dili ile uygulanmıştır. Çıkan sonuçlar küçük veri kümelerinde test edilmiştir. Daha sonraki aşamada ise, yukarıda bahsedilen 2R algoritması oluşturulmuş ve aynı dil ile uygulanmıştır. Kurallara ait hata oranları basit bir arayüz sayesinde, detaylarına girilmeden kullanıcıya sunulmuştur. Her iki algoritma da, yedi veri kümesi üzerinde test edilerek yeterli sonuçlar elde edilmiştir.

Gelişim süreci içindeki eksikler incelendiğinde ise, eksiklerden bir tanesi gerçekleştirilen uygulamanın kayıp veriler içeren veri kümesinde çalışmamasıdır. Bu yüzden test edilecek veri kümeleri seçilirken kayıp verileri olmayan yada gözardı

edilebilecek kadar az olan veri kümeleri seçilmiştir. Diğer bir eksik ise, programın sadece iki sınıf değeri barındıran veri kümeleri üzerinde çalışmasıdır. İlerki zamanlarda devam edilecek olan bu çalışmada bu eksiklerin giderilmesi, bütünlüğün sağlanması ve uygulanan veri kümelerinin seçiciliğinin azalması açısından faydalı olacaktır.

Bunun yanı sıra, sayısal verileri kategorik verilere çevrirken, Robert Holte'nin yöntemine sağdık kalınarak 1RD (1R Discretization) metodu kullanılmıştır. Bu yöntem üzerinde yapılan değişikliğin, kuralların doğruluğunu arttırdığı başka çalışmalarda söylense de tezde 1RD metodu aynen uygulanmıştır. İlerki çalışmalarda, bu metot üzerinde değişiklik yapılabilir. Program, en fazla iki kural üretebilecek şekilde tasarlanmıştır. Geliştirilerek, örnekler sağlandığı sürece, nitelik sayısı (sınıf niteliği hariç) kadar kural üretilebilir. Son olarak, tezde iki algoritma sonucu ortaya çıkan kuralların hata oranları gösterilmiş, fakat detaylı bir yorum yapılmamıştır. Detaylı bir yorumun yapılması için, hata oranını etkileceyecek birçok parametre göz önünde bulundurulmalıdır. O yüzden ilerki çalışmalarda hata oranlarının incelenmesi bu alanda yapılacak diğer bir çalışma olabilir.

Bu tez, makine öğrenmesi uygulamalarının gelişiminde çok ufak bir adım olarak düşünülmüş, Türkiye'de bu konuda yapılabilecek çalışmaların bir parçası olmak için tasarlanmıştır. Bu alana katkıda bulunmak gerçekten heyecan verici bir olgu. Bu heyecanı duyan ve yaşayan insanların makine öğrenmesi alanına gelecek yıllarda yapacakları daha yoğun ve verimli katkılarıyla, önemli değişimlere neden olacaktır.

KAYNAKLAR

[1] From Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Learning, **(Ziyaret Tarihi : 12 Aralık 2005)**

[2] Kocabaş, Ş., A review of learning, ***The Knowledge Engineering Review***, Vol. 6, No.3, 195-122, (1991)

[3] Mitchell T. M., "Machine Learning", ***McGraw-Hill***, 1-3, 274-331, (1997)

[4] Sulun, H., *Zekanın Tanımı* [online], Byte Türkiye, http://www.byte.com.tr /makaleler/default.asp/Gorev/MakaleGoster/Makale/24, **(Ziyaret Tarihi : 12 Aralık 2005)**

[5] Vikipedi, özgür ansiklopedi, http://tr.wikipedia.org/wiki/Yapay_zeka, **(Ziyaret Tarihi : 12 Aralık 2005)**

[6] Shannon, C. E., "XXII.Programming a Computer for Playing Chess", ***Philosophical Magazine***, Ser.7, Vol.41, No.314, 1-17, (1950).

[7] Simon, A. H., "Biographical Memoirs", Volume 71 , 141-173,1997

[8] Spice, B., 2006, *Over the holidays 50 years ago, two scientists hatched artificial intelligence* [online], post-gazette Business News, http://www.post-gazette.com/ pg/06002/631149.stm, **(Ziyaret Tarihi : 16 Ocak 2006)**

[9] Lee, M., 2005, *Samuel's Checkers Player* [online], http://www.cs.ualberta.ca/ ~sutton/book/ebook/node109.html, **(Ziyaret Tarihi : 15 Ocak 2006)**

[10] Slagle, J.R., **"**A Heuristic Program that Solves Symbolic Integration Problems in Freshman Calculus", ***Journal of the ACM***, 507-520, (1963)

[11] Sun, R., "Artificial intelligence: connectionist and symbolic approaches", ***International Encyclopedia of Social and Behavioral Sciences***, (2001).

[12] Vikipedi, özgür ansiklopedi, http://tr.wikipedia.org/wiki/ Makine _Zekas%C4%B1, **(Ziyaret Tarihi : 15 Ocak 2006)**

[13] Wnek, J., Michalski, R.S., "Experimental Comparison of Symbolic and Subsymbolic Learning", ***HEURISTICS, The Journal of Knowledge Engineering, Special Issue on Knowledge Acquisition and Machine Learning***, Vol. 5, No.4, 1-21, (1992).

[14] Yapay Sinir Ağlarının Kısa Bir Tarihçesi, http://www.backpropagation. netfirms.com/tarihce.htm, **(Ziyaret Tarihi : 19 Ocak 2006)**

[15] Feigenbaum, E. A., Feldman, J., "Computers and Thought (Paperback)", First Edition, **AAAI Press / The MIT Press**, (1963).

[16] *Yapay Zeka ve Robotların İnsan Kaynakları Üzerindeki Etkileri*, http://www.humanresourcesfocus.com/makale09.asp, **(Ziyaret Tarihi : 19 Ocak 2006).**

[17] Kurzweil, R., *The Age of Intelligent Machines: Chronology* [online], 1990 http://www.kurzweilai.net/meme/frame.html?main=/articles/art0298.html, **(Ziyaret Tarihi : 19 Ocak 2006).**

[18] SCHANK, R., "Information Is Surprises" , ***Third Culture : Beyond the Scientific Revolution (Paperback)***, Chapter 9, (1995).

[19] Obitko, M., *Introduction to Genetic Algorithms*, [online], Czech Technical University , http://cs.felk.cvut.cz/~xobitko/ga/, **(Ziyaret Tarihi : 19 Ocak 2006).**

[20] Quinlan, J. R., "C4.5:Programs For Machine Learning", ***Morgan Kaufmann Publishers Inc***, (1993).

[21] Bala, J.W., Bloedorn, E., De Jong, K.A., Kaufman, K., Michalski, R.S., Pachowicz, P.W., Vafaie, H., Wnek, J. and Zhang, J., "A Brief Review of AQ Learning Programs and Their Application to the MONKS Problems," ***Reports of the Machine Learning and Inference Laboratory, MLI 92-9***, George Mason University, Fairfax, VA, 1992.

[22] Buchanan, B. G., *History of AI* [online], 2002, http://www.aaai.org/ AITopics/ bbhist.html **(Ziyaret Tarihi : 19 Ocak 2006).**

[23] Tecuci,G., *Machine Learning and Inference* [online], George Mason University, http://64.233.161.104/search?q=cache:w4hTx2VYha4J:lalab.gmu.edu/it811/01-Introduction.ppt+%22history+of+machine+learning%22&hl=en, **(Ziyaret Tarihi : 19 Ocak 2006).**

[24] Winston, P. H., "Learning and reasoning by analogy", ***Artificial Intelligence and Language Processing Communications of the ACM***, TheVolume 23 , Issue 12, 689 – 703, (1980) .

[25] Musen, M.A., "Stanford Medical Informatics: Uncommon research, common goals", **MD Computing,(1)**, 47-50, (1999)

[26] Winston,P., "Artificial Intelligence", 411-422, ***Addison-Wesley Publishing*** Company, (1992).

[27] The Phoney War., Neural Networks versus Artificial Intelligence, (2003).

[28] SPSS White Paper, 1998, *AnswerTree Algorithm Summary* [online], http://66.249.93.104/search?q=cache:Gqw3ZNdPt0IJ:engr.smu.edu/~mhd/8331f03/AT.pdf+AnswerTree+Algorithm+summary&hl=tr&gl=tr&ct=clnk&cd=1 **(Ziyaret tarihi: 08 Şubat 2006)**

[29] StatSoft, Inc., 2003, *Classification and Regression Trees (C&RT)* [online], http://www.statsoft.com/textbook/stcart.html, **(Ziyaret tarihi: 08 Şubat 2006)**

[30] Breiman, L., Friedman, J., Olshen, R., Stone, C., "Classification and Regression Trees", ***Wadsworth International Group***, (1984).

[31] Clark, P., & Niblett, T., "The CN2 induction algorithm", ***Machine Learning***, 3, 261—283, (1989).

[32] Webb, G. I. and N. Brkic, "Learning Decision Lists by Prepending Inferred Rules", ***Proceedings of the AI 93 Workshop on Machine Learning and Hybrid Systems***, pages 6-10, (1993).

[33] Langley, P., Sage, S., "Induction of selective Bayesian classifiers",***In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*** 99--406, (1994).

[34] Nevill-Manning, C.G., Holmes, G. and Witten, I.H., "The development of Holte's 1R classifier", ***Proc ANNES'95***, 239-242, (1995).

[35] Holte, R. C., "Very Simple Classification Rules Perform Well on Most Commonly Used Datasets", ***Machine Learning***, 11, 63—90, (1993).

[36] R. Agrawal, R. Srikant: "Fast Algorithms for Mining Association Rules", ***Proc. of the 20th Int'l Conference on Very Large Databases***, 1994.

[37] Hearst, M.A., Hirsh, H., "AI's Greatest Trends and Controversies", ***IEEE Intelligent Systems***, vol. 15, no. 1, pp. 8-17, (2000).

[38] Alpaydın, E., "Introduction to Machine Learning", ***The MIT Press***, 1-16, (2004).

[39] McQueen, R.J., Garner, S.R., Nevill-Manning, C.G. and Witten, I.H.,"Applying machine learning to agricultural data" ***J Computing and Electronics in Agriculture, 12*(4)**, 275-293, (1995)

[40] Carbonell, J. G., Michalski, R. S., Mitchell, T.M., "Machine Learning: A Historical and Methodological Analysis", ***AI magazin*** , 400 – 408,(1989).

[41] Nilsson, N., "Introduction to Machine Learning", Draft, (1996).

[42] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., "Advances in Knowledge Discovery and Data Mining". Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (editors), ***AAAI Press/ The MIT Press***,1-43, (1996).

[43] Dunhan, M. H., "Data Mining: Introductory and Advanced Topics", 1st edition , ***Prentice Hall***, 9-10, (2002)

[44] Sever, H., Oğuz, B., *Veritabanlarında Bilgi Keşfine Formal Bir Yaklaşım Kısım I: Eşleştirme Sorguları ve Algoritmalar* [online], Başkent Üniversitesi, ODTÜ-Teknokent, http://www.baskent.edu.tr/~sever/bilgi-p01-03.pdf, **(Ziyaret tarihi: 22 Şubat 2006)**

[45] Roiger, R., Geatz, M., "Data Mining: A Tutorial Based Primer", ***Addison Wesley,*** (2002)

[46] Frawley, W.J., Piatetsky-Shapiro, G. ve Matheus, C.J., "Knowledge discovery databases: An overview", G. Piatetsky-Shapiro ve W.J. Frawley (editors), ***Knowledge discovery in databases, AAAI/MIT***, 1-27, (1991)

[47] Langley, P., Simon, H. A., "Applications of machine learning and rule induction", ***Communication of the ACM***, Volume 38 , Issue 11, 54-64, (1995).

[48] StatSoft, Inc., 2003, *CHAID Analysis* [online], http://www.statsoft.com/ textbook/stchaid.html, **(Ziyaret tarihi: 08 Şubat 2006)**

[49] StatSoft, Inc., 2003, [online], *Classification Trees***,** http://www.statsoft.com/ textbook/stclatre.html, **(Ziyaret tarihi: 08 Şubat 2006)**

[50] Tolun, M.R., Abu-Soud, S.M., *An Inductive Learning Algorithm for Production Rule Discovery* [online], Middle East Technical University, Princess Sumaya University College for Technology, www.dis.uniroma1.it/~sassano/STAGE/ LearningRule.pdf, **(Ziyaret tarihi: 22 Şubat 2006)**

[51] Witten, I.H. and Frank, E. , "*Data mining: Practical machine learning tools and techniques"*, Second edition, ***Morgan Kaufmann***, 77-127, (2005)

[52] Cover, T.M., Hart, P.E., "Nearest neighbor pattern classification", ***IEEE Transactions on Information Theory, 13***, 21--27, (1967).

[53] Kumar, V., Steinbach, M., Tan, P., "Introduction to Data Mining", ***Pearson Addison Wesley***, 223-227, (2005).

[54] Stergiou, C., *What is a Neural Network?* [online], Imperial College London, http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/cs11/article1.html, **(Ziyaret tarihi: 22 Şubat 2006)**

[55] Gevaran, U., *Topics in Machine Learning* [online], Brandeis University, http://www.cs.brandeis.edu/~cs113/classprojects/~uday/cs113/bp1.htm, **(Ziyaret tarihi: 24 Şubat 2006)**

[56] Cunningham, S.J., Littin, J.N. and Witten, I.H. "Applications of machine learning in information retrieval", ***Annual Review of Information Science and Technology***, 341-419, (2001).

[57] Emel, G.G., Taşkın, Ç., "Genetik Algoritmalar ve Uygulama Alanları", ***Uludağ Üniversitesi, İktisadi ve İdari Bilimler Fakültesi Dergisi***, Cilt 21, Sayı 1, 129-152, 2002.

[58] Tuğrut, P., Arslan, A., "Sürekli Bir Kirişte Maksimum Momentlerin Genetik Algoritmalar İle Belirlenmesi", ***DEÜ Mühendislik Fakültesi Dergisi, Fen ve Mühendislik Dergisi*****,** Cilt: 3, Sayı: 3 , 1-9, (2001)

[59] Langley, P., "Human and machine learning", ***Machine Learning*****,** ***1***, 243-248, (1986).

[60] Agrawal, R., Imielinski, T., Swami, A., "Mining Associations between Sets of Items in Massive Databases", ***Proceeding of the ACM-SIGMOD 1993 Intl Conference on Management of Data***, 207-216, (1993).

[61] Tuğ, E., Şakiroğlu, M.A., Bulun, M., *Tıbbi Veritabanlarında Gizli Bilgilerin Keşfedilmesi* [online], Selçuk Üniversitesi, http://kurultay.tbd.org.tr/ kurultay20/ Bildiriler/Emine_Tug/bildiri.pdf, **(Ziyaret tarihi: 05 Şubat 2006)**

[62] Orr, G., Schraudolph, N.,Cummins, F., *Reinforcement Learning* [online], Willamette University , http://www.willamette.edu/~gorr/ classes/cs449/ Reinforcement/reinforcement0.html **(Ziyaret tarihi: 05 Şubat 2006)**

[63] Jain, A.K., Murty, M.N., Flynn, P.J., "Data Clustering: A Review", ***ACM Computing Surveys***, Vol.31, No.3, 264-323, (1999).

[64] Matteucci, M., *A Tutorial on Clustering Algorithms* [online], http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/index.html, Politecnico di Milano, **(Ziyaret tarihi: 05 Şubat 2006)**

[65] Belhadjali, M., Whaley, G.L., "A data mining approach to neural network training", ***Information Management & Computer Security***, Vol. 12, No. 1, 117-124, (2000).

[66] Holte, C.R., 2002, *Lessons Learned from Applications of Machine Learning* [online], University of Alberta, www.dcs.fmph.uniba.sk/~hegedus/su/DMLL2002-Holte.pdf, **(Ziyaret tarihi: 05 Şubat 2006)**

[67] Langley, P., "Challenges for the Application of Machine Learning", ***Proceeding of the ICML'97 Workshop on Machine Learning Application in the Real World: Methodological Aspects and Implications***, 15-18, (1997).

[68] Michie, D., "Problems of computer-aided concept formation", ***Applications of Expert Systems 2***, 310–333, (1989).

[69] Giordana, A., Neri, F., Saitta, L., "Automated learning for industrial diagnosis", ***Fielded applications of machine learning,*** 54- 64, (1995).

[70] Leech, W.J., "A rule based process control method with feedback", ***In Proceedings of the ISA/86***, 169-175, (1986).

[71] Evans, R., Fisher, D., "Overcoming Process Delays with Decision Tree Induction", ***IEEE Expert***, Vol. 9, No. 1, 60—66, (1994).

[72] Fayyad, U.M., Smyth, P., Weir, N., Djorgovski, S., "Automated analysis and exploration of large image databases: results, progress, and challenges", ***Journal of Intelligent Information Systems***, 4:7-25, (1995).

[73] Guilfoyle, C., "Ten minutes to lay the foundations", ***Expert Systems User***, 16-19, (1986).

[74] Delcher, A.L., Harmon, D., Kasif, S., White, O., Salzberg, S.L., "Improved microbial gene identification with GLIMMER", ***Nucleic Acids Research***, 27, 4636—4641, (1999).

[75] Liu, H., Hussain, F., Tan, C.L., Dash. M., "Discretization: An enabling technique", ***Journal of Data Mining and Knowledge Discovery, 6(4),*** 393--423, (2002).

[76] Dougherty, J., Kohavi, R., Sahami, M, "Supervised and Unsupervised Discretization of Continuous Features", ***Proceedings of the Twelfth International Conference on Machine Learning***, 194--202, 1995.

[77] What is Python?, http://www.python.org/doc/Summary.html, **(Ziyaret tarihi: 15_03_2006)**

[78] Başer, M., "Python", ***Pusula Yayıncılık,*** 1. Baskı, (2002).

[79] Martelli, A., "Python in a nutshell", First edition, ***O'Reilly***, (2003).

[80] Lutz, M., Ascher, D., "Learning Python", Second edition, ***O'Reilly***, (1999).

[81] John, G.H., Kohavi, R., Pfleger, K., "Irrelevant Features and the Subset Selection Problem", ***Proceeding of the 11th International Conference on Machine Learning,*** 121,129, (1994).

[82] Auer, P., Holte, R.C. and Maass, W., "Theory and applications of agnostic PAC-learning with small decision trees". ***In 12th ICML***, pp. 21-29, (1995).

[83] Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J., *UCI Repository of machine learning databases* [online], Irvine, CA: University of California, Department of Information and Computer Science. http://www. ics.uci.edu/ ~mlearn/ ML Repository.html, **(Ziyaret tarihi: 15_11_2005)**

EKLER

Ek – A : Geliştirilen kod

```
#!/usr/bin/python
import math
import copy
from math import floor
import string
import random
import sys
import time

#gives the number of line in .arff file
def satirsayisi(f):
	satir=1
	dosya=open(f)
	satirsayisi=0
	while satir:
		satir=dosya.readline()
		satirsayisi=satirsayisi+1
	return satirsayisi

#gives how many attribute we have
def attributesayisi(f):
	satir=1
	dosya=open(f)
	attribute_sayisi=0
	while satir:
		satir=dosya.readline()
		if((satir.count('@attribute'))==1 or
(satir.count('@ATTRIBUTE'))==1):
			attribute_sayisi=attribute_sayisi+1
	return attribute_sayisi

#it clears .arff file and form a new file which has only necessary part to use
def yenidosya(f,ss):
	satirsayisi=0
	temp_str=""
	satir=1
	liste=[]
	count=0
	dosya=open(f)
```

```
        myfile=open('myfile','w')
        while satir:
                satir=dosya.readline()
                if(satir.count('')!=1 and satir.count('%')!=1):
                        if(satir.count('@attribute')==1 or
satir.count('@ATTRIBUTE')==1 ):
                                myfile.write(satir)
                        elif(satir.count('@data')==1 or satir.count('@DATA')):
                                myfile.write(satir)
                                while satirsayisi<ss:
                                        satir=dosya.readline()
                                        att=satir.split()
                                        if(att==[]):
                                                count=count+1
                                        elif(att[0]=='%'):
                                                count=count+1
                                        else:
                                                if len(att)>1:
                                                        temp_str=''.join(att)
                                                        myfile.write(temp_str)
                                                else:
                                                        myfile.write(satir)
                                        satirsayisi=satirsayisi+1
                satirsayisi=satirsayisi+1
        myfile.close()
        return

#it checks that it has a real value attribute
def real(f):
        satir=1
        dosya=open(f)
        flag=0
        while satir:
                satir=dosya.readline()
                if((satir.count('real'))==1 or (satir.count('REAL'))==1):
                        flag=1
                        break;

        return flag


def yeni_dosya2(r,sa):
        if r!=1:
                satir=1
                dosya=open('myfile')
                myfile=open('myfile2','w')
                while satir:
                        satir=dosya.readline()
                        myfile.write(satir)
```

```
            myfile.close()
        else:
            att_name=attributename(sa)
            gk=gerekli_kisim(sa)
            ro=rows_order(gk)
            co=columns_order(ro,att_sayisi)
            fv=find_value(co,att_sayisi)
            c=classs(fv)
            print "class:",c
            print "########################################"
            fri=find_real_index(sa)
            print fri
            print "########################################"
            fr=find_real(sa,ro)
            print "find_real:",fr
            print "########################################"
            dso=disc_step_one(fr,fri)
            print "disc_step_one:",dso
            print "########################################"
            ce=class_element(dso)
            print "class_elements:",ce
            print "########################################"
            fi=fi=find_interval(ce,dso,c)
            print "find_interval",fi
            print "########################################"
            di=determine_interval(fi,c)
            print "determine_interval:",di
            print "########################################"
            mp=merge_partition(di,c)
            print "merge_part:",mp
            print "########################################"
            cts=convert_to_string(mp,dso,fr,c,fri,att_name)
            print "########################################"
            ana=add_new_att(cts,fri,co,att_name,c)
            lts=list_to_string(f,ana,len(ro)-1,att_sayisi,ss)
            sra=split_real_att(lts)
            print "split_real_att:",sra
        return

#sort the the list which comes from find value function [['65','yes'],['70','no']]
def disc_step_one(listem,real_index):

        list=[]
        k=0
        while k<len(listem):
            bos=[]
            bos=bos+listem[k]
            i=0
            while i<(len(listem[k])):
```

```
                        j=0
                        while j<(len(bos)-1-i):
                                if (bos[j+1][0] < bos[j][0]):
                                        tmp = bos[j]
                                        bos[j] = bos[j+1]
                                        bos[j+1]=tmp
                                j=j+1
                        i=i+1
                list=list+[bos]
                k=k+1
        return list


#it reads from file and for each line it creates a list
def split_att(yeni_dosya):
        satir=1
        count=1
        liste=[]
        att=[]
        dosya=open('myfile','r')
        while satir:
                satir=dosya.readline()
                att=satir.split()
                        if (att!=[]):
                                liste=liste+[att]
                        count=count+1

        return liste

def split_attr(yeni_dosya):
        satir=1
        count=1
        liste=[]
        att=[]
        dosya=open('myfile2','r')
        while satir:
                satir=dosya.readline()
                att=satir.split()
                        if (att!=[]):
                                liste=liste+[att]
                        count=count+1

        return liste


#from list, it pulls attribute_name
def attributename(first_split):

        att_name=[]
```

```
        i=0
        while i<len(first_split):
                if(first_split[i][0]=='@attribute' or
first_split[i][0]=='@ATTRIBUTE'):
                        att_name=att_name+[first_split[i][1]]
                i=i+1
        return att_name

#it gives only data number
def instancesayisi(first_split):

        numberofinstance=0
        i=0
        while i<len(first_split):
                if(first_split[i][0]=='@data' or first_split[i][0]=='@DATA'):
                        numberofinstance=len(first_split)-(i+1)
                        break
                i=i+1
        return numberofinstance


#it returns a list that includes attribute name and data
def gerekli_kisim(liste):

      att_list=[]
         m=0
      tam=[]
      count=0
      i=0
         while i<len(liste)-1:
                if(liste[i]!=[]):
                        if(liste[i][0]=='@attribute' or liste[i][0]=='@ATTRIBUTE'):
                           att_list=att_list+[liste[i][1]]
                        elif(liste[i][0]=='@data' or liste[i][0]=='@DATA'):
                                m=i+1
                        tam=tam+[att_list]
                                while (m<len(liste)):
                                        if(liste[m]==[]):
                                                del liste[m]
                                        else:
                                                tam=tam+[liste[m]]
                                          m=m+1


                        elif(liste[len(liste)-1]==[]):
                                del liste[len(liste)-1]
            else:
                  count=count+1
            i=i+1
```

```
        return tam


#it splits each value of rows
def rows_order(liste):
        temp_str=""
           list=[]
           list=list+[liste[0]]
            i=1
           while i<len(liste):
              temp_str=liste[i][0]
              list=list+[temp_str.split(',')]
                    i=i+1
        return list


#it gives only data not attribute name
def instances(liste):
           temp_str=""
           list=[]
            i=1
           while i<len(liste):
              temp_str=liste[i][0]
              list=list+[temp_str.split(',')]
                    i=i+1
        return list

#it gives each attribute value
def columns_order(liste,att_sayisi):
           new=[]
           i=0
           while i<att_sayisi:
                    temp=[]
                    j=1
                    while j<len(liste):
                       temp=temp+[liste[j][i]]
                             j=j+1
                    new=new+[temp]
                    i=i+1
           return new

#it gives value of attribute
def find_value(temp,att_sayisi):
           new=[]
           value=[]
           j=0
          while j<att_sayisi:
                    k=0
                    value=[]
```

```
                value=value+[temp[j][k]]
                while k<len(temp[j])-1:
                if(cmp(temp[j][k],temp[j][k+1]!=0)):
                        if(value.count(temp[j][k+1])==0):
                                value=value+[temp[j][k+1]]
                        k=k+1
                k=k+1
                new=new+[value]
                j=j+1

        return new

#it gives the name of class
def classs(liste):

        j=len(liste)-1
        return liste[j]

def random_number_generator(num_of_ins):
        return random.randint(1,num_of_ins)

#it returns a list which includes only training data
def form_training_data(random_number,num_of_ins,ins):
        training_data=[]
        x=random_number
        y=random_number
        two_three=round(num_of_ins*0.75)
        one_three=round(num_of_ins*0.25)
        i=0
        if (x<one_three):
                while x<(y+(two_three)):
                        training_data=training_data+[ins[x]]

                        x=x+1
        else:
                while i< ((y+(two_three))%num_of_ins):
                        training_data=training_data+[ins[i]]
                        i=i+1
                while x<num_of_ins:
                        training_data=training_data+[ins[x]]
                        x=x+1
        return training_data


#it returns a list which includes only test data
def form_test_data(random_number,num_of_ins,ins):

        test_data=[]
        x=random_number
```

```
        y=random_number
        two_three=round(num_of_ins*0.75)
        one_three=round(num_of_ins*0.25)
        z=int(random_number+two_three)
        t=int((random_number+two_three)%num_of_ins)
        i=0
        if (x<one_three):
                while i<y:
                        test_data=test_data+[ins[i]]
                        i=i+1
                while z<num_of_ins:
                        test_data=test_data+[ins[z]]
                        z=z+1
        else:
                while t<y:
                        test_data=test_data+[ins[t]]
                        t=t+1

        return test_data


def random_number_generator_unique(num_of_ins):
        return random.sample(range(num_of_ins),5)

#if a real attribute exists, it gives the index of real value
def find_real_index(liste):
        i=0
        tam=[]
        while i<len(liste)-1:
                if(liste[i]!=[]):
                        if(liste[i][0]=='@attribute' or liste[i][0]=='@attribute'):
                                tam=tam+[liste[i][2]]

                i=i+1
        i=0
        real_index=[]
        while i<len(tam)-1:
                if(tam[i]=='real' or tam[i]=='REAL'):
                        real_index=real_index+[int(i)]
                i=i+1
        return real_index


#it find the real value with its class like ['85.0','yes']
def find_real(liste,l):

        real2=[]
        tam=[]
        count=0
```

```
    count1=0
    i=0
        while i<len(liste)-1:
                if(liste[i]!=[]):
                        if(liste[i][0]=='@attribute' or liste[i][0]=='@attribute'):
                                tam=tam+[liste[i][2]]
                i=i+1
        i=0
        while i<len(tam)-1:
                if(tam[i]=='real' or tam[i]=='REAL'):
                        real=[]
                        j=1
                        while j<len(l):
                                real=real+[[float(l[j][i]),l[j][len(tam)-1]]]
                                j=j+1
                        real2=real2+[real]
                i=i+1
        return real2

#it finds element of class
def class_element(liste):
        step1=[]
        i=0
        while i<len(liste):
                class_elem=[]
                j=0
                while j<len(liste[i]):
                        class_elem=class_elem+[liste[i][j][1]]
                        j=j+1
                step1=step1+[class_elem]
                i=i+1
        return step1

#According to breakpoint, it determines intervals
def find_interval(class_liste,sirali_liste,class_value):
        if (len(class_liste[0])<=50):
                limit=3
                last=4
        else:
                limit=6
                last=10
        index=0
        new=[]

        for_each_att=[]
        i=0
        while i<len(class_liste):
                genel=[]
                j=0
```

```
                while j<len(class_liste[i]):
                        new=new+[class_liste[i][j]]
                                if(new.count(class_value[0])==limit):
                                if (j!=len(class_liste[i])-1):
                                        j=j+1

        index=check(class_liste[i],j,class_value[0],class_value[1],new,sirali_liste[i])
                                        while j<=index:
                                                new=new+[class_liste[i][j]]
                                                j=j+1
                                        genel=genel+[new]
                                        new=[]
                                        j=index
                                else:
                                        genel=genel+[new]

                                elif(new.count(class_value[1])==limit):
                                if (j!=len(class_liste[i])-1):
                                        j=j+1

        index=check(class_liste[i],j,class_value[1],class_value[0],new,sirali_liste[i])
                                        while j<=index:
                                                new=new+[class_liste[i][j]]
                                                j=j+1
                                        genel=genel+[new]
                                        new=[]
                                        j=index
                                else:
                                        genel=genel+[new]
                                elif ((len(class_liste[i])-1)-index <= last) &
(j==(len(class_liste[i])-1)):
                                        genel=genel+[new]
                                j=j+1
                for_each_att=for_each_att+[genel]
                new=[]
                i=i+1

        return for_each_att

def check(liste,sum,value1,value2,new,sirali_liste):
        index=0
        flag=0

        while ((flag==0) & (sum<len(liste))) :
                if (liste[sum]==value1) :
                        index=sum
                        sum=sum+1
                        flag=0
                elif (liste[sum]==value2):
```

```
                    index=sum-1
                    flag=1
                else:
                    index=sum-1
                    flag=1
        flag1=0
        while ((flag1==0) & (index!=len(liste)-1)):
            if(sirali_liste[index][0]==sirali_liste[index+1][0]):
                liste[index+1]=value1
                index=index+1
                flag1=0
            else:
                flag1=1
        return index

def determine_interval(liste,class_value):
        a=0
        b=0
        cvol=[]
        value=0
        max_value=0
        new=[]
        newest=[]
        index=0
        i=0
        while i<len(liste):
            new=[]
            j=0
            while j<len(liste[i]):
                a=liste[i][j].count(class_value[0])
                cvol=cvol+[a]
                b=liste[i][j].count(class_value[1])
                cvol=cvol+[b]
                if ((a==b) & (j==len(liste[i])-1)):
                    if new[j-1][0]==class_value[0]:
                        new=new+[[class_value[1],len(liste[i][j])]]
                    else:
                        new=new+[[class_value[0],len(liste[i][j])]]

                else:
                    max_value=max(cvol)
                    value=cvol.index(max_value)
                    index=class_value[value]
                    new=new+[[index,len(liste[i][j])]]
                cvol=[]
                j=j+1
            newest=newest+[new]
            i=i+1
        return newest
```

```
def merge_partition(liste,class_value):

        genel=[]
        for_each_att=[]
        index=0
        new=[]
        sum=0
        i=0
        while i<len(liste):
                j=0
                while j<len(liste[i]):
                        if (liste[i][j][0]==class_value[0]):
                                sum=liste[i][j][1]
                                new=new+[class_value[0],sum]
                                if j!=(len(liste[i])-1):
                                        j=j+1

        index=return_index(liste[i],j,class_value[0],class_value[1])
                                        while j<=index:
                                                sum=sum+liste[i][j][1]
                                                new=[]
                                                new=new+[class_value[0],sum]
                                                j=j+1
                                        genel=genel+[new]
                                        new=[]
                                        j=index

                                else:
                                        sum=liste[i][j][1]
                                        genel=genel+[new]
                        elif (liste[i][j][0]==class_value[1]):
                                sum=liste[i][j][1]
                                new=new+[class_value[1],sum]
                                if j!=(len(liste[i])-1):
                                        j=j+1

        index=return_index(liste[i],j,class_value[1],class_value[0])
                                        while j<=index:
                                                sum=sum+liste[i][j][1]
                                                new=[]
                                                new=new+[class_value[1],sum]
                                                j=j+1

                                        genel=genel+[new]
                                        new=[]
                                        j=index
                                else:
                                        sum=liste[i][j][1]
                                        genel=genel+[new]
```

```
                    j=j+1
                for_each_att=for_each_att+[genel]
                new=[]
                genel=[]
                i=i+1
        return for_each_att


def return_index(liste,sum,value1,value2):
        index=0
        flag=0
        while ((flag==0) & (sum<len(liste))) :
                if (liste[sum][0]==value1) :
                        index=sum
                        sum=sum+1
                        flag=0

                elif (liste[sum][0]==value2):
                        index=sum-1
                        flag=1
                else:
                        index=sum-1
                        flag=1

        return index

def convert_to_string(liste,sirali_liste,sirasiz_liste,class_value,fri,att_name):

        x=0.0
        y=0.0
        avg=0.0
        temp=0.0
        i=0
        while i<len(liste):
                sum=0.0
                count=0
                avg=0.0
                temp=0.0
                j=0
                while j<len(liste[i]):
                        sum=count+liste[i][j][1]
                        if count==0:

                                if (len(liste[i])==1):
                                        x=sirali_liste[i][0][0]
                                        y=sirali_liste[i][sum-1][0]
                                        k=0
                                        print "Eger",
att_name[fri[i]],"<=",x,"ise",liste[i][j][0]
```

```
                                        print "Eger",
x,"<=",att_name[fri[i]],"<=",y,"ise",liste[i][j][0]
                                        while k<len(sirasiz_liste[i]):
                                                if sirasiz_liste[i][k][0]<=y:

        sirasiz_liste[i][k][0]=liste[i][j][0]
                                                k=k+1

                                else:
                                        x=sirali_liste[i][sum-1][0]
                                        y=sirali_liste[i][sum][0]
                                        avg=average(x,y)
                                        print "Kirilma Noktasi:",avg
                                        k=0
                                        print "Eger",
att_name[fri[i]],"<=",avg,"ise",liste[i][j][0]
                                        while k<len(sirasiz_liste[i]):
                                                if sirasiz_liste[i][k][0]<=avg:

        sirasiz_liste[i][k][0]=liste[i][j][0]
                                                k=k+1
                                count=sum
                                temp=avg

                        elif ((count!=0) & (j==len(liste[i])-1)):
                                print "Kirilma Noktasi:",avg
                                print "Eger", att_name[fri[i]],">",avg,"ise",liste[i][j][0]
                                k=0
                                while k<len(sirasiz_liste[i]):
                                        if((sirasiz_liste[i][k][0]>avg) &
((str(sirasiz_liste[i][k][0])!=class_value[0]) &
(str(sirasiz_liste[i][k][0])!=class_value[1]))):

                                                sirasiz_liste[i][k][0]=liste[i][j][0]
                                        k=k+1
                                count=sum
                                temp=avg

                        elif (count!=0):
                                x=sirali_liste[i][sum-1][0]
                                y=sirali_liste[i][sum][0]
                                avg=average(x,y)
                                print "Kirilma Noktasi:",avg
                                k=0
                                print "Eger",
temp,"<",att_name[fri[i]],"<=",avg,"ise",liste[i][j][0]

                                while k<len(sirasiz_liste[i]):
```

```
                                        if ((temp<sirasiz_liste[i][k][0]<=avg) &
((str(sirasiz_liste[i][k][0])!=class_value[0]) &
(str(sirasiz_liste[i][k][0])!=class_value[1]))):

                                                sirasiz_liste[i][k][0]=liste[i][j][0]
                                        k=k+1
                                count=sum
                                temp=avg

                        j=j+1
                print "###################################"
                i=i+1
        return sirasiz_liste

def average(x,y):
        return (x+y)/2

def add_new_att(r_list,fri,co,att_name,class_value):
        i=0
        liste2=[]
        while i<len(r_list):
                j=0
                liste=[]
                while j<len(r_list[i]):
                        liste=liste+[r_list[i][j][0]]
                        j=j+1
                liste2=liste2+[liste]
                i=i+1
        a=0
        while a<len(fri):
                print class_value[0],"=",att_name[fri[a]].upper()
                print class_value[1],"=",att_name[fri[a]].lower()
                a=a+1
        i=0
        while i<len(liste2):
                j=0
                while j<len(liste2[i]):
                        if (liste2[i][j]==class_value[0]):
                                liste2[i][j]=att_name[fri[i]].upper()
                        elif (liste2[i][j]==class_value[1]):
                                liste2[i][j]=att_name[fri[i]].lower()
                        j=j+1
                i=i+1
        i=0
        while i<len(fri):
                co.pop(fri[i])
                co.insert(fri[i],liste2[i])
                i=i+1
        return co
```

```
def list_to_string(f,rtn,length,att_sayisi,ss):

        satirsayisi=0
        satir=1
        liste=[]
        count=0
        dosya=open('myfile')
        myfile=open('myfile2','w')
        while satir:
                satir=dosya.readline()
                if(satir.count('')!=1 and satir.count('%')!=1):
                        if(satir.count('@attribute')==1 or
satir.count('@ATTRIBUTE')==1 ):
                                myfile.write(satir)
                        elif(satir.count('@data')==1 or satir.count('@DATA')==1):

                                myfile.write(satir)

                                i=0
                                while i<length:
                                        temp_str=""
                                        j=0
                                        while j<att_sayisi:
                                                temp_str=temp_str+rtn[j][i]
                                                if(j!=att_sayisi-1):
                                                        temp_str=temp_str+","
                                                j=j+1
                                        myfile.write(temp_str)
                                        myfile.write('\n')


                                        i=i+1
                satirsayisi=satirsayisi+1

        myfile.close()
        return

def split_real_att(lts):
        satir=1
        count=1
        liste=[]
        att=[]
        dosya=open('myfile','r')
        while satir:
                satir=dosya.readline()
                att=satir.split()
                if (att!=[]):
                        liste=liste+[att]
                count=count+1
```

```
    return liste

def freq(value,liste,cls):

        d1={}
        rule=[]
        list2=[]
        s=0
        i=0

        if len(cls)==2:
                while i<len(value)-1:
                        k=0
                        while k<len(value[i]):
                                j=0
                                class1=0
                                class2=0
                                while j<len(liste):
                                        if(liste[j][i]==value[i][k]):
                                                if(liste[j][len(value)-1]==cls[0]):
                                                        class1=class1+1
                                                else:
                                                        class2=class2+1
                                        j=j+1

                                d1[value[i][k]]={cls[0]:class1,cls[1]:class2}

                                k=k+1
                        i=i+1


        elif len(cls)==3:
                while i<len(value)-1:
                        k=0
                        while k<len(value[i]):
                                j=1
                                class1=0
                                class2=0
                                class3=0
                                while j<len(liste):
                                        if(liste[j][i]==value[i][k]):
                                                if(liste[j][len(value)-1]==cls[0]):
                                                        class1=class1+1


                                                elif(liste[j][len(value)-1]==cls[1]):
                                                        class2=class2+1
                                                else:
                                                        class3=class3+1
```

```
                                        j=j+1

        d1[value[i][k]]={cls[0]:class1,cls[1]:class2,cls[2]:class3}

                                                    k=k+1
                    i=i+1
        else:
                print "error"

        index=0
        deneme=[]
        i=0
        while i<len(value)-1:
                list = []
                j=0
                while j<len(value[i]):
                        if(d1.has_key(value[i][j])==1):
                                dd = d1[value[i][j]].keys()
                                k=dd[0]
                                v = d1[value[i][j]][dd[0]]
                                toplam=0
                                for kk, w in d1[value[i][j]].iteritems():
                                        deneme=deneme+[w]
                                        toplam=toplam+w
                                        if w > v:
                                                v = w
                                                k=kk

                                if (len(deneme)==2):
                                        if (deneme[0]==deneme[1]):
                                                index=random.sample(range(len(dd)),1)
                                                v=deneme[index[0]]
                                                k=dd[index[0]]

                                elif(len(deneme)==3):
                                        if (deneme[0]==deneme[1] &
deneme[1]==deneme[2]):
                                                index=random.sample(range(len(dd)),1)
                                                v=deneme[index[0]]
                                                k=dd[index[0]]

                                deneme=[]
                                list = list + [int(v)]
                                rule=rule+[[value[i][j],k,v,toplam]]
                        j=j+1
                list2=list2+[list]
                i=i+1
        pay=[]
        i=0
```

```
        while i<len(list2):
                sum=0.0
                j=0
                while j<len(list2[i]):
                        sum=sum+(list2[i][j])
                        j=j+1
                pay=pay+[sum]
                i=i+1
        rule=rule+[pay]
        return rule

def if_same_number_of_values(rules,values,sinif_degerleri):
        i=0
        number_of_values=[]
        while i<len(values)-1:
                number_of_values=number_of_values+[len(values[i])]
                # [3 2 2 2]
                i=i+1
        i=0
        number=0
        j=0
        while i<(len(number_of_values)):
                number=number+number_of_values[i]
                sinif=[]
                iki_kat=[]
                while j<number :
                        sinif=sinif+[rules[j][1]]
                        if rules[j][3]!=0:
                                if ((rules[j][3]/rules[j][2])==2):
                                        iki_kat=iki_kat+[j]
                        j=j+1
                if(sinif.count(sinif_degerleri[0])==len(sinif)):
                        if(len(iki_kat)==1):
                                rules[iki_kat[0]][1]=sinif_degerleri[1]

                        else:
                                k=0
                                while k<len(iki_kat):
                                        if(iki_kat[k]%2==0):
                                                rules[iki_kat[k]][1]=sinif_degerleri[0]
                                        elif(iki_kat[k]%2==1):
                                                rules[iki_kat[k]][1]=sinif_degerleri[1]
                                        k=k+1
                elif(sinif.count(sinif_degerleri[1])==len(sinif)):
                        if(len(iki_kat)==1):
                                rules[iki_kat[0]][1]=sinif_degerleri[0]
                        else:
                                k=0
                                while k<len(iki_kat):
```

```
                                        if(iki_kat[k]%2==0):
                                                rules[iki_kat[k]][1]=sinif_degerleri[0]
                                        elif(iki_kat[k]%2==0):
                                                rules[iki_kat[k]][1]=sinif_degerleri[1]
                                        k=k+1

                        j=number
                        i=i+1
                return rules

def accuracy_values(rule,value,att_name,total,c):
        accurcy={}
        listem=[]
        dict={}
        i=0
        while i<len(att_name)-1:
                listem=listem+[[att_name[i],rule[len(rule)-1][i]/total]]
                i=i+1
        return listem


def one_rule(rule,value,att_name,total,c):
        accurcy={}
        listem=[]
        dict={}
        i=0
        while i<len(att_name)-1:
                sum=0.0
                print att_name[i],"dogruluk =",rule[len(rule)-1][i],total,rule[len(rule)-
1][i]/total
                listem=listem+[[att_name[i],rule[len(rule)-1][i]/total]]
                i=i+1
        i=0
        while i<(len(listem)-1):
                j=0
                while j<(len(listem)-1-i):
                        if (listem[j+1][1] > listem[j][1]):
                                tmp = listem[j]
                                listem[j] = listem[j+1]
                                listem[j+1] = tmp
                        j=j+1
                i=i+1
        list=[]
        deneme=[]
        deneme2=[]
        index=att_name.index(listem[0][0])
        i=0
        while i<len(value[att_name.index(listem[0][0])]):
                j=0
```

```
                while j<len(rule):
                        if(value[index][i]==rule[j][0]):
                                print
"Eger",listem[0][0],value[index][i],"ise",rule[j][1],"       ",
"(",rule[j][2],"/",rule[j][3],")"

        deneme=deneme+[[listem[0][0],value[index][i],rule[j][1]]]
                        j=j+1
                i=i+1
        print
        return deneme


def one_rule_result(rule,value,att_name,total,c):
        accurcy={}
        listem=[]
        dict={}
        i=0
        while i<len(att_name)-1:
                sum=0.0
                print att_name[i],"dogruluk =",rule[len(rule)-1][i],total,rule[len(rule)-
1][i]/total
                listem=listem+[[att_name[i],rule[len(rule)-1][i]/total]]
                i=i+1
        i=0
        while i<(len(listem)-1):
                j=0
                while j<(len(listem)-1-i):
                        if (listem[j+1][1] > listem[j][1]):
                                tmp = listem[j]
                                listem[j] = listem[j+1]
                                listem[j+1] = tmp
                        j=j+1
                i=i+1


        list=[]
        deneme=[]
        deneme2=[]
        index=att_name.index(listem[0][0])
        i=0
        while i<len(value[att_name.index(listem[0][0])]):
                j=0
                while j<len(rule):
                        if(value[index][i]==rule[j][0]):
                                print
"Eger",listem[0][0],value[index][i],"ise",rule[j][1],"       ",
"(",rule[j][2],"/",rule[j][3],")"
                        deneme=deneme+[[listem[0][0],value[index][i],rule[j][1]]]
```

```
                    j=j+1
            i=i+1

    return deneme

def error_detection1(all,att_sayisi,test_data,att_name):

    i=0
    temp=0.0
    error=0.0
    sum=0.0
    while i<len(all):

        false=0
        total=0
        index1=att_name.index(all[i][0])
        j=0
        while j<len(test_data):
            if(all[i][1]==test_data[j][index1]):
                total=total+1
                if (all[i][2]!=test_data[j][att_sayisi-1]):
                    false=false+1
            j=j+1
        print i+1,"","," kural HATA =",false,"/",total
        i=i+1
        temp=temp+false
        sum=sum+total
    error=temp/sum
    print error
    return error,all[0][0]


def error_detection1_result(all,att_sayisi,test_data,att_name):
    i=0
    temp=0.0
    error=0.0
    sum=0.0
    while i<len(all):

        false=0
        total=0
        index1=att_name.index(all[i][0])
        j=0
        while j<len(test_data):
            if(all[i][1]==test_data[j][index1]):
                total=total+1
                if (all[i][2]!=test_data[j][att_sayisi-1]):
                    false=false+1
            j=j+1
```

```
                print i+1,".","kural HATA =",false,"/",total
                i=i+1
                temp=temp+false
                sum=sum+total
        print temp,sum
        error=temp/sum
        print error
        return error,all[0][0]


def handout(tn,ins_sayisi,ins,fv,c,att_name,att_sayisi,gk,starttime):

        sum=[]
        error_sum=[]
        i=0
        while i<len(tn):
                ftrd=[]
                ftd=[]
                ftrd=form_training_data(tn[i],ins_sayisi,ins)
                ftd=form_test_data(tn[i],ins_sayisi,ins)
                fq=freq(fv,ftrd,c)
                if_same_number_of_values(fq,fv,c)
                ev=if_same_number_of_values(fq,fv,c)
                sum=sum+[accuracy_values(fq,fv,att_name,len(ftrd),c)]
                sum=sum+[accuracy_values(ev,fv,att_name,len(ftrd),c)]
                re=one_rule(ev,fv,att_name,len(ftrd),c)
                error_sum=error_sum+[error_detection1(re,att_sayisi,ftd,gk[0])]
                i=i+1

        index=max_accurate_att(sum,att_sayisi)
        min_error_sum=min(error_sum)
        error_index=error_sum.index(min(error_sum))
        error_index=min_error(error_sum,att_name[index])
        min_error_att(index,att_name,error_sum,error_index,tn,ins_sayisi,ins,fv,c,att
_sayisi,gk,error_sum,starttime)
        return

def min_error(error_sum,att_name_index):
        i=0
        minimum=1.0
        count=0
        while i<len(error_sum):
                if(error_sum[i][1]==att_name_index):
                                if(minimum > error_sum[i][0]):
                                minimum=error_sum[i][0]
                                count=i
                i=i+1
        return count
```

```
def max_accurate_att(sum_att_list,att_sayisi):

        i=0
        sum2=[]
        while i<att_sayisi-1:
                sum=0.0
                j=0
                while j<len(sum_att_list):
                        sum=sum+sum_att_list[j][i][1]
                        j=j+1
                sum2=sum2+[sum]
                i=i+1
        return sum2.index(max(sum2))

def
min_error_att(index,att_name,e_sum,e_index,tn,ins_sayisi,ins,fv,c,att_sayisi,gk,error
_sum,starttime):

        ftrd=[]
        ftd=[]
        liste=[]
        ftrd=form_training_data(tn[e_index],ins_sayisi,ins)
        ftd=form_test_data(tn[e_index],ins_sayisi,ins)
        fq=freq(fv,ftrd,c)
        ev=if_same_number_of_values(fq,fv,c)
        re=one_rule_result(ev,fv,att_name,len(ftrd),c)
        error_detection1_result(re,att_sayisi,ftd,gk[0])
        endtime=time.time()
        print endtime-starttime
        return

f=sys.argv[1]
def find_random():
        rngu=[]
        i=2
        while i<7:
                rngu=rngu+[int(sys.argv[i])]
                i=i+1
        return rngu

starttime=time.time()
ss=satirsayisi(f)
att_sayisi=attributesayisi(f)
yd=yenidosya(f,ss)
r=real(f)
sa=split_att(yd)
yd2=yeni_dosya2(r,sa)
sat=split_attr(yd2)
att_name=attributename(sat)
```

```
ins_sayisi=instancesayisi(sat)
gk=gerekli_kisim(sat)
ins=instances(gk)
ro=rows_order(gk)
co=columns_order(ro,att_sayisi)
fv=find_value(co,att_sayisi)
c=classs(fv)
rngu=find_random()
handout(rngu,ins_sayisi,ins,fv,c,att_name,att_sayisi,gk,starttime)
```

Ek-B: Arayüzün Oluşmasını Sağlayan Program

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.event.*;
import java.io.*;

class GUI extends JFrame implements ActionListener {

	Container c;
	JButton dosyaAcDugme, kural1Dugme, kural2Dugme, cikDugme;
	JLabel secilenDosyaEtiketi;
	JTextArea monitor, monitor2;
	JScrollPane sp, sp2;
	String random_list = "python random_number_gen_unique.py ";
	String command = "python 1R_kural.py ";
	String command2 = "python 2R_kural.py ";


	public GUI() {

		c = getContentPane();
		c.setLayout(new FlowLayout());

		this.setTitle("no file selected");
		sp = new JScrollPane();
		monitor = new JTextArea(100, 100);
		sp.getViewport().add( monitor );
		sp.setMinimumSize(new Dimension(500, 400));
		sp.setPreferredSize(new Dimension(425, 230));
		sp.setMaximumSize(new Dimension(500, 400));


		sp2 = new JScrollPane();
		monitor2 = new JTextArea(100, 100);
		sp2.getViewport().add( monitor2 );
		sp2.setMinimumSize(new Dimension(500, 400));
		sp2.setPreferredSize(new Dimension(425, 230));
		sp2.setMaximumSize(new Dimension(500, 400));

		dosyaAcDugme = new JButton("Dosya Ac");
		dosyaAcDugme.addActionListener(this);

		kural1Dugme = new JButton("Kural 1'e Gore Calistir");
		kural1Dugme.addActionListener(this);

		kural2Dugme = new JButton("Kural 2'ye Gore Calistir");
```

```
        kural2Dugme.addActionListener(this);

        cikDugme = new JButton("Programdan Cikis");
        cikDugme.addActionListener(this);
        c.add(dosyaAcDugme);
        c.add(kural1Dugme);
        c.add(kural2Dugme);
        c.add(cikDugme);
        c.add(sp);
        c.add(sp2);


        this.setResizable(false);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(475, 555);
        this.show();

}

public void actionPerformed(ActionEvent e) {

        if(e.getSource() == dosyaAcDugme) {

                JFileChooser fileChooser = new JFileChooser();

                fileChooser.setFileSelectionMode(JFileChooser.FILES_ONL
                Y);
                fileChooser.setCurrentDirectory(new File("."));
                int result = fileChooser.showOpenDialog(this);
                if(result==JFileChooser.CANCEL_OPTION)
                {
                        return;
                }
                File file = fileChooser.getSelectedFile();
                this.setTitle(file.getAbsolutePath());
                random_list += ""+file.getAbsolutePath();
                try {
                        Process p1 = Runtime.getRuntime().exec(random_list);
                        BufferedReader stdInput = new BufferedReader(new
                        InputStreamReader(p1.getInputStream()));
                        String s = stdInput.readLine();
                        System.out.println("Gelen liste: " + s);
                        String params = fromListStringtoArray(s);
                        command += ""+file.getAbsolutePath()+params;
                        System.out.println("Giden komut: " + command);
                        command2 += ""+file.getAbsolutePath()+params;
                        System.out.println("Giden komut: " + command2);

                }catch(Exception e2) {
```

```
                                System.out.println("Exception oldu");
                                e2.printStackTrace();
                                System.exit(-1);
                        }
                }
                if(e.getSource() == kural1Dugme) {
                        String s = null;
                try {
                         Writer output = new BufferedWriter( new
FileWriter("cikti.txt") );
                         Process p = Runtime.getRuntime().exec(command);
                         BufferedReader stdInput = new BufferedReader(new
                        InputStreamReader(p.getInputStream()));

            while ((s = stdInput.readLine()) != null)
                                output.write(s + "\n");
                                 stdInput.close();
              output.close();

                         BufferedReader in = new BufferedReader(new
FileReader("cikti.txt"));
                        String str, result = "";
                        while ((str = in.readLine()) != null) {
                            monitor.append(str + "\n");
                            monitor.repaint();
                 }
                         monitor.validate();
                         c.validate();
                         c.repaint();
                 in.close();
            }
            catch (IOException ev) {
                                System.out.println("exception happened ");
                                ev.printStackTrace();
                                System.exit(-1);
            }}
                if(e.getSource() == kural2Dugme) {
                        String s = null;
                try {
                         Writer output = new BufferedWriter( new
        FileWriter("cikti.txt") );
                         Process p = Runtime.getRuntime().exec(command2);
                         BufferedReader stdInput = new BufferedReader(new
                                        InputStreamReader(p.getInputStream()));

              while ((s = stdInput.readLine()) != null)
                                output.write(s + "\n");
                                 stdInput.close();
              output.close();
```

```
                        BufferedReader in = new BufferedReader(new
FileReader("cikti.txt"));
                    String str, result = "";
                    while ((str = in.readLine()) != null) {
                        monitor2.append(str + "\n");
                        monitor2.repaint();
                }
                    monitor2.validate();
                    c.validate();
                    c.repaint();
                in.close();
            }
            catch (IOException ev) {
                        System.out.println("exception happened ");
                        ev.printStackTrace();
                        System.exit(-1);
            }
                }
                if(e.getSource() == cikDugme) {
                    System.exit(0);
        }       }

        public static String fromListStringtoArray(String pythonList) {
                String []splitted = pythonList.split(",");
                String result="";
                for(int i = 0; i < splitted.length; i++) {
                    if(i == 0)
                      result += " " + splitted[i].substring(1, splitted[i].length());

                    else if(i < splitted.length - 1)
                        result +=" "+ splitted[i];

                    else
                        result += " " +splitted[i].substring(1,
splitted[i].length()-1);

                }

                System.out.println("Programa Giden Liste: " + result);
                return result;
        }
        public static void main(String arg[]) {
                new GUI();

        }}
```

Ek-C : Kullanılan Veri Kümeleri

Weather.arff	Hava durumuna göre golf oynanıp, oynanmayacağına dair veriler içerir.
Postoperative.arff	Hastanın ameliyattan sonra nereye sevk edileceğine dair veriler içermektedir.
Tae.arff	Wisconsin-Madison Üniversitesi'nde çalışan 151 öğretim görevlisinin beş dönem boyunca gösterdiği performans verilerini içermektedir.
Haberman.arff	1958 ve 1970 yıllarında, Chicago's Billings Üniversite Hastanesi'nde, göğüs kanseri ameliyatı geçiren hastaların hayatta kalıp, kalmadıklarına dair veriler içermektedir.
Bank.arff	Bankanın sigorta kararı ile ilgili veri içermektedir.
Pima.arff	Pima hintli kadınlarda şeker hastalığının çıkıp, çıkmama durumuna dair veriler içermektedir.
Contraceptive.arff	Endonezyada kadınların gebelikten korunmak için herhangi bir metot kullanıp, kullanmadığına dair veriler içerir.

ÖZGEÇMİŞ

1980 yılında İstanbul'da doğdu. İlk, orta ve lise öğrenimini İstanbul'da tamamladı. 1998 yılında girdiği İstanbul Bilgi Üniversitesi Fen-Edebiyat Fakültesi Matematik ve Bilgisayar Programcılığı Bölümü'nden 2003 yılında mezun oldu. 2003 yılında Kocaeli Üniversitesi Fen Bilimleri Enstitütüsü Bilgisayar Mühendisliği Bölümü'nde Yüksek Lisans öğretimine başladı. 2003 yılından bu yana İstanbul Bilgi Üniversitesi Fen Edebiyat Fakültesi Bilgisayar Bilimleri Bölümü'nde Araştırma Görevlisi olarak görev yapmaktadır.