MAKİNE ÖĞRENMESİ TEKNİKLERİ İLE EKOLOJİK VERİLERİN DEĞERLENDİRİLMESİ

Alişan BALTA

Yüksek Lisans Tezi
Ekobilişim Anabilim Dalı
Danışman: Dr. Öğr. Üyesi Şengül DOĞAN
EYLÜL-2018

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİ TEKNİKLERİ İLE EKOLOJİK VERİLERİN DEĞERLENDİRİLMESİ

YÜKSEK LİSANS TEZİ

Alişan BALTA

Fen Bilimleri Enstitüsü
Ekobilişim Ana Bilim Dalı

Danışman: Dr. Öğr. Üyesi Şengül DOĞAN

EYLÜL -2018

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİ TEKNİKLERİ İLE EKOLOJİK VERİLERİN DEĞERLENDİRİLMESİ

YÜKSEK LİSANS TEZİ

Alişan BALTA

(151141101)

Tezin Enstitüye Verildiği Tarih: 13.08.2018
Tezin Savunulduğu Tarih: 14.09.2018

Danışman: **Dr. Öğr. Üyesi Şengül DOĞAN (F.Ü.)**
Diğer Jüri Üyeleri: **Doç. Dr. Muhammed Fatih TALU (İ.Ü.)**
Dr. Öğr. Üyesi Erhan AKBAL (F.Ü.)

EYLÜL -2018

ÖNSÖZ

Bu tez çalışmasında, konu seçiminde, yürütülmesinde ve gerçekleştirilmesinde isteklerimi göz önüne alarak bana her konuda desteğini, anlayışını ve en önemlisi sabrını esirgemeyen hocam Dr. Öğr. Üyesi Şengül DOĞAN' a çok teşekkür ederim.

Alişan BALTA

ELAZIĞ-2018

İÇİNDEKİLER

Sayfa No

ÖZET

Ekoloji, organik ve inorganik sistemler üzerinde çalışan doğa bilimcilerin bu sistemler ile ilgili karmaşık ilişkileri incelemesini, anlamasını ve bu ilişkilerden çıkarım yapmalarını amaçlayan bir bilim dalıdır. Bu karmaşık ilişkilerin anlaşılabilmesi ve ileriye dönük tahminlerin yapılabilmesi noktasında Makine Öğrenmesi teknikleri istatistiksel metotlara göre daha avantajlı olabilmektedir. Makine Öğrenmesi ile ekolojik sistemlere ait doğrusal olmayan, yüksek hacimli, fazla niteliğe sahip veya çok sayıda eksik veri içerebilen veri kümeleri, eğitim algoritmaları ile yapay öğrenme sürecinden geçirilip bu veri kümeleri ile ilgili kestirimler yapılabilmektedir.

Bu tez çalışmasında, Makine Öğrenmesi' nde yaygın kullanılan Yapay Sinir Ağları, K-NN, K-Means, Naive Bayes, ve Karar Ağacı Sınıflandırma (ID3) yöntemlerinin, ekolojik verilerde ne şekilde kullanılabileceğinin anlaşılması amacıyla, ayrıca bu yöntemlerin daha iyi kavranmasını sağlamak için iki adet örnek veri kümesi, bu yöntemler kullanılarak değerlendirilmiştir. Bu beş yöntem de modellenerek geliştirilen bir deney ortamında test edilmiştir. Test süreçlerinde farklı parametreler kullanılmış ve seçilen bir doğrulama yöntemi üzerinden en iyi performans değerleri elde edilmeye çalışılmıştır. Sonuç olarak elde edilen performans değerleri, literatürde yaygın olarak kullanılan diğer algoritmaların performanslarına yakın olduğu gözlemlenmiştir.

Anahtar Kelimeler: Ekoloji, Ekobilişim, Makine Öğrenmesi, Yapay Sinir Ağları, K-means, K-NN, Karar Ağaçları, Naive Bayes

SUMMARY

An Evaluation of Ecological Data with Machine Learning Techniques

Ecology is a scientific discipline that aims to allow natural scientists working on biotic and abiotic systems to study, understand, and inference complex relationships related to these systems. Machine learning techniques could be more profitable than statistical methods at the point where these complex relationships can be understood and prospective predictions can be made. With Machine Learning, the datasets which are nonlinear, high volume, have high amount attributes or can contain a large number of missing data that belong to ecological systems can be passed through the artificial learning process by using training algorithms and predictions can be made.

In this thesis, two datasets are evaluated in order to understand how Artificial Neural Networks, K-NN, K-Means, Naïve Bayes, and Decision Tree Classification (ID3) methods which are widely used in Machine Learning can be used in ecological data. It is also aimed to have a better understanding of these methods. Five methods were tested in a designed and developed experiment set. In the testing processes, it has been tried to obtain the best performance values through a selected validation method by means of different parameters. In result, it was observed that the performance values were similar to those of other algorithms commonly used in the literature.

Keywords: Ecology, Ecological Informatics, Machine Learning, Artificial Neural Networks, K-means, K-NN, Decision Trees. Naïve Bayes.

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

SEMBOLLER VE KISALMALAR LİSTESİ

MÖ: Makine Öğrenmesi

YSA: Yapay Sinir Ağları

KO: K-Ortalamalar

KA: Karar Ağaçları

NB: Naïve Bayes

o: Çıkış fonksiyonu

η: Öğrenme sabiti

t: Hedef niteliği

w: Ağılık vektörü

x: Giriş vektörü

D: Eğitim örnekleri kümesi

E: Hata fonksiyonu

sgn: İşaret fonksiyonu

σ: Sigmoid fonksiyonu

1. EKOBİLİŞİM VE MAKİNE ÖĞRENMESİ

1.1. Giriş

Eski çağlarda, miktar açısından sahip olunan bilgi ne kadar fazla ise o kadar değerli olarak görülüyordu. Bu yüzden eski medeni toplumlar, sahip oldukları kütüphaneler ve kitap sayıları ile övünürlerdi. Matbaa bulunmadan önce kitaplar el ile yazılır, bunları çoğaltmak ve yedeklemek için büyük bir çaba sarf edilirdi. Günümüzde ise bilgi teknolojilerinin gelişmesi ile elde edilen ve işlenmesi gereken verilerin boyutları çok büyük miktarlara ulaştı. Önceden bilgiyi çoğaltmak için harcanılan emek şu anda bu bilgileri sadeleştirmek ve anlamlı yapılar elde edebilmek için harcanmaktadır. Büyük miktarda olan bu veriler bir yerlerde bir gün kullanılabilir düşüncesiyle saklı tutulmaktadır. Günümüzde bu verilerin miktarı insanın hesaplayabilme kapasitesinin çok üstünde olduğu için bu işi akıllı sistemler aracılığı ile yaptırma ihtiyacı ortaya çıkmıştır. Bu noktada, makinenin hesaplama kabiliyeti ile insanın sahip olduğu akıl yürütme ve öğrenme kabiliyeti birleştirebilinir mi? sorusu bilim dünyasında araştırılan popüler bir konu haline gelmiştir. Bu problem esasında yapay zekâ başlığı ile ilişkili olan büyük ve karmaşık bir problemdir. Bu problemi çözmek için Yapay zekânın alt konulardan biri olan Makine Öğrenmesi (MÖ), daha çok bir yapay öğrenme sistemi modelleyip, deneyimler üzerinden çıkarımlar yaparak bilgi elde eden bir yaklaşım kullanır. MÖ günümüzde üzerinde çokça çalışılan araştırma alanlarındandır.

Birçok alanda olduğu gibi dünyamızdaki yaşamın korunması ve sürdürülebilmesi için MÖ tekniklerinden faydalanabiliriz. Yaşamımız düşünüldüğünde özellikle son yüz yılda sahip olduğumuz birçok konfor beraberinde bir sürü doğaya ait unsuru yok etmektedir. Bu tarz problemlere doğanın dengesini bozmayacak çözümleri bizim yerimize fark edebilecek yapılara ihtiyacımız bulunmaktadır. MÖ tekniklerinin ekolojik verilerde kullanılması ekolojik sistemleri anlama, ileriye dönük tahminlerde bulunma ve istenmeyen durumlara karşı tedbir alma açılarından ekolojistlere ve diğer doğa bilimcilerine büyük fırsatlar sunmaktadır.

1.2. Bölümlere Genel Bakış

Birinci bölümde bilgi teknolojilerinin ekoloji alanında kullanımı (Ekobilişim) konusuna giriş yapılmış, ekoloji alanında kullanılabilecek temel hesaplama teknolojilerine ve sürdürülebilirlik kavramına değinilmiştir. Ayrıca MÖ' ye giriş niteliğinde temel kavramlardan ve yapay zekâ ile ilişkisinden bahsedilmiştir. İkinci Bölüm, MÖ uygulamalarının tasarım süreçleri ve buna bağlı olarak eksik verilerin işlenmesi, normalizasyon ve kategorik verilerin kodlanması gibi ön işleme aşamaları anlatılmıştır. Üçüncü Bölümde geliştirilen bir MÖ modelinin performansının ölçülmesi ile ilgili yöntemler anlatılarak giriş yapılmıştır. Ardından en çok kullanılan yöntemlerden beş tanesi tanıtılmış, tasarlanmış ve uygulamaları gerçekleştirilmiştir. Örnek veri kümeleri ile test edilen uygulamaların sonuçları paylaşılmıştır. Bu bölümde geliştirilen modeller: Yapay sinir ağları ile sınıflandırma, Karar ağaçları ile sınıflandırma, K-NN ile sınıflandırma, Naive Bayes ile sınıflandırma, K-ortalamalar (K-means) ile kümeleme şeklindedir. Dördüncü bölümde geliştirilen modeller üzerinde yapılan testlerin sonuçlarına ve tez çalışmasına dair genel bir değerlendirme yapılmıştır.

1.3. Ekoloji ve Hesaplama

1.3.1. Giriş

Ekoloji, mekâna ve zamana bağlı olarak birçok şekilde açıklanabilecek, temelde organizmalar ile organizmaların çevreleri arasındaki etkileşimi ile ilgilenen bir disiplindir. Bu disiplin organizmaların dağılımını ve varlığını etkileyen süreçleri, bunların sistem içerisindeki etkileşimlerini ve enerjinin dönüşümünü içeren konuları kapsar [1].

Ekolojik sistemlerin modellenmesi ve analizi birçok duruma bağlı olarak güç olabilmekte ve büyük emek harcanması gereken süreçlerdir. Ekolojistlerin topladıkları veriler işleme karmaşıklığı açısından birçok probleme sahip olabilmektedir. Ekolojik verilerin doğrusal olmayan veya beklenmeyen bir yoğunlukta olması, veri sayısının çok büyük olması, veri kümesinin çok fazla sayıda özellik içermesi ve veri kümesi içerisinde aşırı miktarda boşlukların olması gibi durumlar bu problemlere örnek verilebilir [2].

Ekolojik bir model, bir sistemdeki değişiklikleri tanımlayabilmeli, sistemi oluşturan bileşenlerin işleyiş şeklini zamansal ve mekânsal açılardan genelleştirebilmelidir. İdeal

anlamda ekolojik bir model, bir sistemin gelecekte nasıl bir fonksiyon sergileyeceği ile ilgili tahminde bulunmaya elverişli olmalıdır [3].

Ekolojik modellerin uygulama alanları ile ilgili popüler başlıklar şu şekildedir [4, 8]:

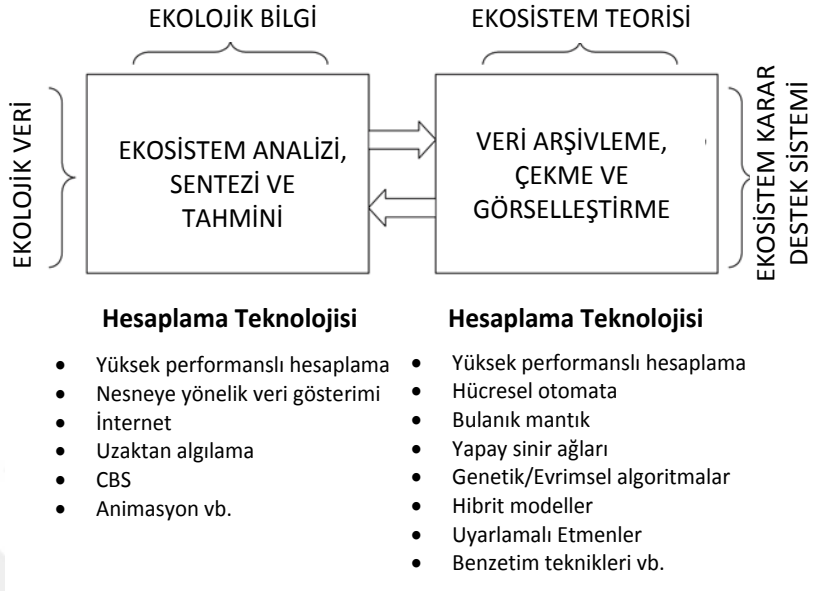
- Nüfus dinamikleri.
- Doğal habitat modellemesi.
- Gen akışı modellemesi.
- Risk modellemesi.

Ekolojik modellemede giriş değerleri çevresel verilerden oluşur. Bu veriler üç şekilde karşımıza çıkmaktadır: İlk tip veri inorganik özelliklerden oluşur. Çevrenin fiziksel ve kimyasal yönünü temsil eder. İkinci tip veri biyolojik özelliklerden oluşur. Diğer özellikler ise insanların etkileri ve bunların sonuçlarının etken olduğu özelliklerdir [9].

1.3.2. Ekobilişim

Ekobilişim kavramı ilk olarak 2000 yılında "Ekolojik Modellemede Makine Öğrenmesi Uygulamaları" adlı konferansta önerilmiştir. Sonrasında "Uluslar Arası Ekobilişim Topluluğu" kurulmuştur [10]. Ekobilişim, gelişmiş hesaplama teknolojileri ile ekolojik verilerin işlenmesi, arşivlenmesi, analizi ve sentezlenmesi işlerinin gerçekleştirildiği disiplinler arası bir altyapıya sahip; sürdürülebilir ekoloji, biyolojik çeşitlilik, küresel ısınma gibi konularda net kararlar verilebilmeyi hedefleyen araştırma alanıdır. Veri işleme ve arşivlenmesi ile meta veri ve nesneye dayalı programlama aracılığıyla ekolojik verilerin standardizasyonunun sağlanması amaçlanır. Analiz ve sentezleme aracılığıyla, bilginin işlenme ilkelerinin belirlenmesi, ekosisteme ait modellerin yapılandırılıp işler hale getirilmesi ve ekosistemin davranışlarının biyolojik kökenli hesaplama teknikleri veya diğer teknikler ile tahmin edilmesine çalışılır. Ekosistemlerin simülasyonu, ileriye dönük tahminler ve karmaşık veri modellerinden bilgi çıkarılması gibi işlemlerde; bulanık mantık, yapay sinir ağları, evrimsel algoritmalar ve uyarlanabilir akıllı etmenler gibi biyolojik sistemlerden ilham alınan hesaplama teknikleri (Örn: genetik algoritma) kullanabilir. Bu teknikler Ekobilişim' de kullanılan yaygın hesaplama teknikleri olarak kabul edilir [11].

Şekil 1.1’ de Ekobilişim’ in kapsamı genel hatları ile verilmiştir.



Şekil 1.1 Ekobilişim’ in kapsamı [11].

Şekil 1.1’ de gösterildiği gibi Ekosistem analizi, sentezi ve tahmini aşağıda verilen başlıkları içerir.

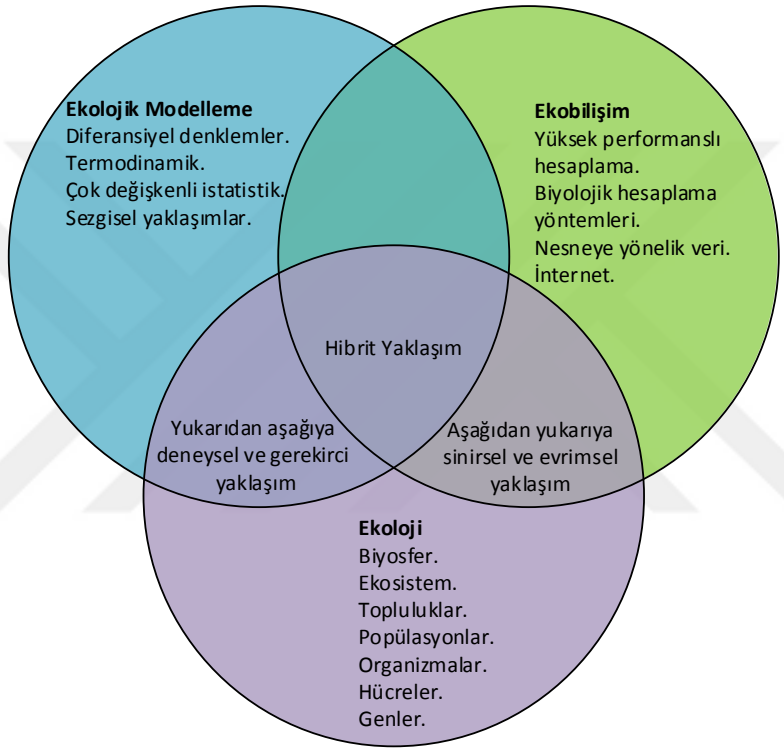
- Yüksek Hızlı Veri Erişimi ve İşlenmesi: Yüksek performanslı donanım ve büyük boyutlu dâhili bellek gerektirir.
- Nesne yönelimli veri gösterimi: Meta veri ve veri işlemlerinin bir veri yapısına dönüştürülmesi neticesinde verinin standartlaştırılması ve kolay anlaşılabilir olması açılarından kullanılır.
- İnternet: Veri kümeleri, dinamik ve dağıtık bir yapıda gönderilir.
- Uzaktan Algılama ve Coğrafi Bilgi Sistemleri (CBS): Mekânsal verinin elde edilmesini ve görselleştirilmesini sağlamak için kullanılır.
- Animasyonlar: verinin görselleştirerek simülasyonların daha rahat anlaşılması açısından kullanılır.

Verileri arşivleme, çekme ve görselleştirme ise aşağıda maddeler halinde verilen konuları kapsamaktadır.

- Hücresel Otomasyon (Cellular Automata): Simülasyonu kolaylaştırmak için kullanılır.
- Bulanık Mantık: Belirsiz veriyi temsil etmek ve işlemek için kullanılır.

- Yapay Sinir Ağları: Çok değişkenli doğrusal olmayan regresyon; sınıflandırma ve kümeleme işlemlerini, küçük veya büyük boyutlu resim işlemeyi ve çok değişkenli zaman serilerinin analizini kolaylaştırmak için kullanılır.
- Uyarlamalı Etmenler: Ekosistemin yapısına ve gelişimine yönelik simülasyonların ve tahminlerin yapılmasını kolaylaştırmak için kullanılır [11].

Ekobilişim, ekolojik modelleme ve ekolojinin ortak ve ayrık noktaları Şekil 1.2' de görülebilmektedir.

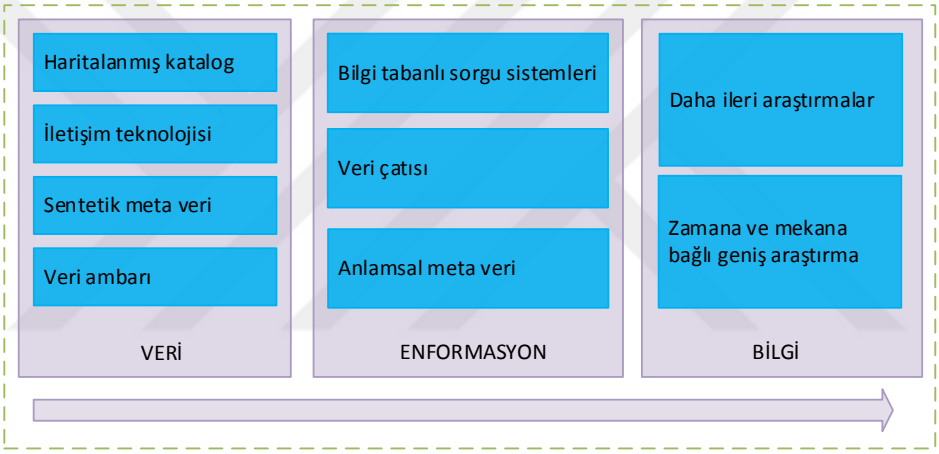


Şekil 1.2 Disiplinler arası yaklaşım [11].

Gelişen bilgi teknolojileri ekolojistlerin ihtiyaçlarını karşılayacak yeni araçlar üretilmesine imkân sağlamıştır. Birçok disiplinden araştırmacıların işbirliği ile gerçekleştirilen projeler katma değeri yüksek ve verimli sonuçlar ortaya çıkartılmıştır. Örneğin LTER (Long Term Ecological Research), ekolojide uzun vadeli bilimsel araştırmalara odaklanmış bir topluluktur. Ekolojik verilerin niceliğinin, niteliğinin ve karmaşıklığının artması LTER' in Bilgisayar Bilimleri alanı ile özellikle ilgilenmesini sağlamıştır. Bu açıdan bilgi teknolojisinin basit verilerin ötesinde uygulanması, depolama ve elektronik yayıncılığın geliştirilmesi, küresel entegre bilgi ağının kapasitesinin geliştirilmesi ve bilgiyi keşfetmek, erişmek, yorumlamak, işlemek gibi süreçlerin

sağlanması için Bilgisayar Bilimleri özel bir rol oynamaktadır. Bunun oluşturulması ise altyapı ve kaynak yatırımı gerektirir. Şekil 1.3' de gösterildiği gibi verinin enformasyona, enformasyonun ise bilgiye dönüşme süreci, bu dönüşümü kapsayan üç geniş alan ile özetlenebilir [12].

- Ekolojik verilere uzun vadeli erişim ve yönetim için bir ağa bağlı veri depolama sistemi tasarlanması.
- İlkel verilerin entegrasyonunu ve sentezini kolaylaştırmak için araçların ve prosedürlerin geliştirilmesi.
- Arşivlenmiş veri kaynakları kullanılarak yapılacak araştırma faaliyetlerinin ve uygulamaların desteklenmesi ve teşvik edilmesi.



Şekil 1.3 Ekobilişim altyapı modeli [12].

1.3.3. Sürdürebilirlik

İnsanların bireysel veya toplumsal çıkarları için meydana getirdiği değişiklikler, yaşayan her türlü canlı sistemi küresel iklim değişikliğine maruz bırakarak ekosistem ve biyolojik çeşitlilik üzerine olumsuz etkileri olmuş ve bu etkiler günümüzde hissedilen bir oranda artmıştır [13]. Bu noktada yaşamın her alanında uygulanabilecek sürdürülebilir politikalara ihtiyaç vardır. Sürdürülebilirlik, basit bir tanımla günümüzde mevcut olan kaynakların gelecekte her türlü canlı nesil için kullanılabilir kalmasını sağlayabilmek şeklinde tanımlanabilir. Bununla birlikte S. Kellogg, "*Toolbox for Sustainable City Living*" adlı kitabında klasik sürdürülebilirlik kavramının büyük şirketler tarafından içinin boşaltıldığını ve bağlamından koparılarak pazarlanabilir bir kavrama dönüştürüldüğünü

iddia ederek radikal sürdürülebilirlik kavramını ortaya atmıştır. Radikal sürdürülebilirlik bakış açısı, ekolojik ve toplumsal olayların bir birleriyle ilintili olduğunu, bir probleme çözüm bulunurken başka bir soruna yol açmamasını veya problemi daha da kötü hale getirmemesi gerektiğini ifade etmektedir [14].

Sürdürülebilirlik son yıllarda akademik alanda, devletlerin çevre politikalarında ve temiz endüstriyel üretim teknolojileri alanlarında büyük ilgi görmüştür. Bununla ilgili yapılmış bazı yapay zekâ çalışmaları, bilgisayar bilimlerinin ve farklı disiplinlerin bu konudaki çalışmalara katılmasını sağlamıştır. Yapay zeka tekniklerinin ekoloji bilimi için kullanılıp uygulamalara dönüşmesi geçmişte öngörülen, günümüzde ise gerçekleştirilen uygulamaları olan bir süreçtir. Özellikle coğrafi bilgi sistemleri gibi bir çok ekolojik hesaplama ve yönetim araçları, sürdürülebilirlik hedeflerine ulaşma noktasında büyük katkılar sağlamaktadır [15,17].

Sürdürülebilirlik hedeflerine ulaşmak, bilgisayar bilimcileri ile diğer alanlarda çalışan bilim insanlarının çok disiplinli araştırma alanlarında işbirliği yapmalarını gerekli kılmıştır. Bu ilişkiden, endüstriyel üretimde optimizasyon (Örn: ekolojik ayak izini azaltmak için tedarik zincirinin kısaltılması), düşük enerji kullanımının sağlanması, yeşil bilişim vb. kavramlar ortaya çıkmıştır [18].

Sürdürülebilir doğal yaşam ve nüfus değişim dinamiklerinin araştırılmasında kullanılan modellerde MÖ teknikleri ile elde edilen bilgiler ileriye dönük tahminler yapılmasına olanak sağlayablir. Tahmine dayalı modelleme (Predictive Modeling) olarak anılan bu yaklaşım MÖ alanında en fazla kullanılan yaklaşımlardan biridir. Tüme varım yöntemi şeklinde de isimlendirilmektedir. Bu yaklaşım ile geçmiş deneyimlerden (eğitim verileri) faydalanarak öngörülerde bulunulur. Temelde amaç hedef özelliklerinden hareketle tahminin başarılı bir şekilde yapılması için modeli öğrenmektir. Tahmine dayalı modelleme ekolojik verilerde, özellikle de otonom çalışan sistemlerde artan bir şekilde kullanılmaktadır.

1.4. Makine Öğrenmesi

1.4.1. Giriş

Ekoloji ve yer bilimleri gibi doğa bilimleri alanında çalışan araştırmacılar, organik ve inorganik sistemler arasındaki karmaşık ilişkiler üzerinde çalışarak bu konuları anlamaya ve öngörüde bulunmaya çalışırlar. Bu karmaşık ilişkileri çözmek için kullanılan MÖ

yöntemleri geleneksel istatistiksel metotlara göre daha avantajlı olabilmektedir. Doğrusal olmayan, yüksek boyutlu veriler üzerinde işlem yapılabilmekte ve eksik veriler değerlendirilebilmektedir [19].

MÖ, kökleri yapay zekâ ve istatistiğe dayanan veri kümelerinden bilgi üretmemize olanak sağlayan tekniklere sahip bir araştırma alanıdır [20]. Bu bilgiler, verilerin MÖ algoritmaları aracılığıyla öğrenilmesi neticesinde tahminlerde bulunmamızı sağlayan modellerden elde edilir. Öğrenme ise deneyimleri bilgiye ve uzmanlığa dönüştürme işidir [21]. Öğrenme işini gerçekleştirmek için farklı yaklaşımlara sahip birçok algoritma geliştirilmiştir. Temelde bir öğrenme algoritmasının girişleri, eğitim verilerinden oluşur. Eğitim verileri, deneyimlerden (etiketlenmiş, niteliği bilinen veriler) elde edilen verilerdir. Çıkışlar ise, deneyimler kullanılarak elde edilen, başka sistemlerin de kullanabileceği üzerinde fikir yürütülebilecek yeni bilgilerdir.

Gündelik hayatımızda yaptığımız birçok rutin işlemlerin veya hayvanların sahip olduğu birçok davranışın (Örn: Ses veya görüntünün tanınması) programlamış bir sistem tarafından yapılması, oldukça karmaşık yapılar üzerinde çalışmayı gerektirebilmektedir. Bununla birlikte insanların akıl yürütme ile yeterli zamanda veya hiç gerçekleştiremeyeceği (Örn: Go oyununu hatasız oynamak) görevler de MÖ konuları içerisindedir. Geniş bir alanda kullanılan MÖ yöntemleri çokça uygulandığı alanlar ise şu şekildedir [20]:

- Metin veya belge sınıflandırma (İstenmeyen veya reklam içerikli elektronik postaların tespiti.)
- Doğal dil işleme (Morfolojik analiz, istatistiksel çözümleme.)
- Konuşma tanıma, konuşma sentezi, konuşmacı doğrulama
- Optik karakter tanıma
- Hesaplamalı biyoloji uygulamaları (protein fonksiyonları, yapısal tahminler)
- Bilgisayar görmesi (resim tanıma, yüz tanıma)
- Dolandırıcılık tespiti ve ağa izinsiz girme (kredi kartı ve telefon dolandırıcılığı)
- Oyunlar (satranç, tavla)
- Sürücüsüz araç kontrolü (robot, yön bulma)
- Tıbbi teşhis
- Öneri sistemleri, arama motorları, bilgi çıkarma sistemleri

MÖ' de öğrenme problemlerine dair temel çözüm yaklaşımları şu şekildedir:

- **Sınıflandırma:** Her bir veri örneğinin bir kategorik özelliği vardır. Sınıflandırma ile her bir örneğin bu kategorik özelliği bulunmaya çalışılır. Bu özellikler genellikle az sayıdadır (hava durumu, belge sınıflandırması vb.). Ancak optik karakter tanıma, metin sınıflandırma, konuşma tanıma gibi alanlar daha fazla hatta sınırsız kategorik özellikler barındırabilir.
- **Kümeleme:** Örnekleri homojen bölgelere bölmek için kullanılır. Genellikle çok büyük veri kümelerini analiz ederken kullanılır. Sosyal ağ analizi gibi büyük toplulukların tanımlanmasında sıklıkla kullanılır.
- **Regresyon:** Ekonometri gibi alanlarda çokça kullanılan istatistiksel bir yöntemdir. Değişkenler arasındaki ilişkiyi ölçmek için kullanılır. Bağımlı bir değişken ile bağımsız değişkenler arasındaki ilişkiyi analiz ederek ilişkinin gücü hakkında bilgi edinilir.
- **Sıralama:** Belirli bir kritere göre sıralama işlemidir. Arama motorlarında yapılan aramalarda en uygun sonucu getirmek üzere "PageRank" adlı algoritmada [22] başarılı bir şekilde kullanılmıştır.

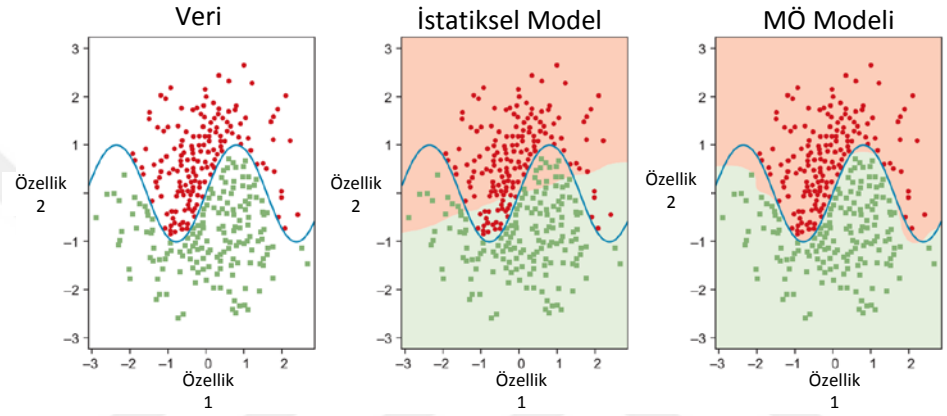
1.4.2. Makine Öğrenmesi Tipleri

Basit anlamda öğrenme, belli bir görevin işleyişini iyileştirmek için yeni bilgilerin kullanılmasıdır [2]. Öğrenme işi genel olarak verilerden çıkarılan örüntüler temel alınarak yapılır. Bu açıdan MÖ ile veri madenciliği verilerden örüntüleri yakalama konusunda benzeşir. Bundan farklı olarak MÖ bir programın işleyişinde gerektiğinde değişikler yapabilir. Veri madenciliği veriler üzerinde bilinmeyen bilgiler elde etmek üzerine odaklanırken makine öğrenmesi tanımlama ve kestirim üzerine odaklanır. Tanımlama ve kestirim, deneysel eğitim verilerinin bilinen özelliklerinin bilgisayar algoritmaları ile öğrenilmesi ile gerçekleştirilir. Eğitim verilerinden öğrenmeye tümevarım (inductive learning) denir. Tümevarımın amacı eğitim verilerinden hedef değerlerinin tahminine yönelik bir model gerçekleştirmek olduğunda bu öğrenme yaklaşımına, kestirimsel (predictive) veya gözetimli (supervised) öğrenme denir [4].

- **Gözetimli Öğrenme:** Öğrenci, eğitim verisi olarak bir dizi etiketli (hangi sınıfa ait olduğu belli) örnek alır ve bu bilinen örneklerden hareketle bilinmeyen örnekler için tahminlerde bulunur. Sınıflandırma, regresyon ve sıralamayla ilgili problemlerle ilişkili olarak en yaygın kullanılan öğrenme biçimidir.

- **Gözetimsiz Öğrenme:** Öğrenci, yalnızca etiketsiz eğitim verilerini alır ve bilinmeyen tüm örnekler için tahminler yapar. Kümelenme ve boyut azaltma denetimsiz öğrenme problemlerine örnek verilebilir.

Öğrenme yeteneğine sahip bir modelin başarısı Şekil 1.4' te gösterilmiştir. İki değişkenli doğrusal olmayan bu sınıflandırma probleminde uygulanan istatistiksel bir modelin yetersiz ve hatalı bir sonuç verdiği görülebilir. Diğer yandan MÖ modeli, gerçek sınıflandırma sınırlarını keşfedebilmiştir.

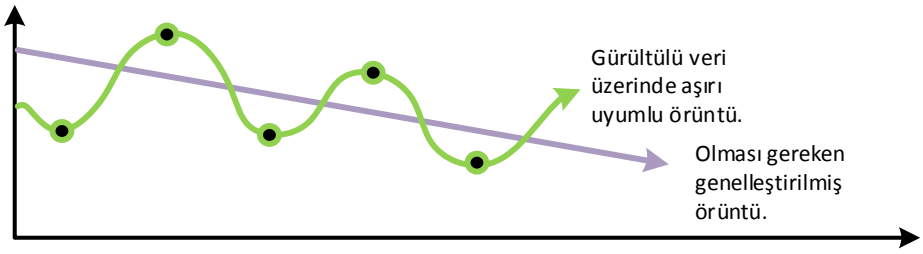


Şekil 1.4 Makine öğrenmesi modelinin başarısı [23].

1.4.3. Aşırı Uyum (Overfitting) Problemi

Kullanılan bir modelin başarısı giriş verilerinin kalitesine bağlıdır. Giriş verilerindeki beklenmeyen veya dağılıma aykırı değerler gürültü olarak adlandırılır. Gürültünün kaynağı yanlış ölçümleme, veri toplama araçlarının düşük kalitesi, verilerin yeterli zaman aralığında toplanmaması, hatalı kodlama veya veri kayıplarının oluşması vb. nedenler olabilir.

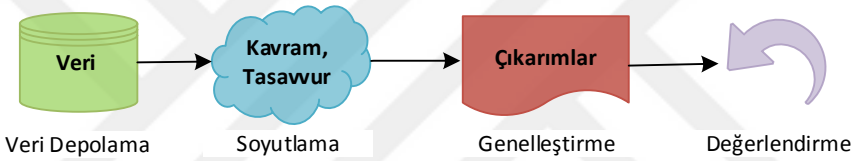
Gürültü, modelin öğrenilmesi aşamasında aşırı uyum (overfitting) probleminin oluşmasına yol açar. Aşırı uyum, eğitim aşamasında görültülü verilerinde öğrenilmesi veya aynı verilerin çok fazla öğrenilmesi ile oluşan algoritmanın genelleşme yeteneğini Şekil 1.5' te görüldüğü üzere kaybetmesine yol açan bir durumdur [24].



Şekil 1.5 Aşırı uyum problemi [24].

1.4.4. Makine Öğrenmesi Evreleri

Biyolojik bir sistem veya bir makine için öğrenmenin kavramsal olarak Şekil 1.6' da gösterildiği gibi dört adımlı bir işleyişi vardır.



Şekil 1.6 Öğrenme süreçleri [24].

- **Veri Depolama:** Bu aşama, gözlem yapılması, bilgilerin hafızaya alınması ve tekrar kullanılabilmesi işlemlerini gerçekleştirmek için kaynaklık sağlar.
- **Soyutlama:** Kayıtlı bilgilerin bir bağlam içerisinde farklı çapta veya nitelikte anlamlara gelebilmesini sağlayan aşamadır.
- **Genelleştirme:** Soyutlanmış verilerden yeni bir bağlamda bilgi üretmeyi ve çıkarım yapma aşamasını oluşturur.
- **Değerlendirme:** Öğrenilen bilgilerin kalitesini ölçmeyi ve muhtemel geliştirmeler açısından bildirimde bulunmayı sağlayan aşamadır.

Kavramsal olarak işleyişin yanında MÖ algoritmaları, uygulama süreçlerini içeren bir yapıda çalışırlar. Bu yapıyı beş adımdan oluşan bir gelişim evresi ile özetlenebilir. Bu evreler şu şekildedir:

- **Veri Toplama:** MÖ algoritmalarında öğrenme işlevinin yapılabilmesi için gerekli eğitim verilerinin elde edilmesidir. Verilerin elde edilme, elektronik sistemler (örn: Kablosuz sensör ağları), sosyal ağlar, anket çalışmaları vb.

yöntemler ile gerçekleştirilir. Bu veriler metin, tablo, XML dosyası veya veri tabanından oluşabilir.

- **Veri Keşfi ve Hazırlanması***:* Giriş verilerinin kalitesi örnek modelin başarısı ile doğru orantılıdır. Bu açıdan giriş verileri için uygun verilerin aranarak seçilmesi gerekir. Veri hazırlama ise gereksiz bilgilerin çıkarılması ve hatalı olduğu bilinen verilerin ayıklanmasını kapsar.
- **Modelin Eğitimi:** Eğitim için hazırlanmış veri setine uygun olarak seçilmiş modelin öğrenmesi işi gerçekleştirilir.
- **Modelin Değerlendirilmesi:** Kullanılan modelden elde edilen her sonucun bir sapma değeri vardır. Bu değerlerin belirlenmesi için test verilerinden faydalanılır.
- **Modelin Geliştirilmesi:** Modelin değerlendirilmesi ile elde edilen performans yeterli görülmeyebilir. Bu durumda yeni veriler ile eğitimin desteklenebilir veya başka bir modele geçiş yapılarak yeni bir eğitim süreci yürütülebilir [24].

1.4.5. Veri Analizi Yaklaşımları

MÖ‘ de öğrenme sürecinde doğru veriler ile çalışmak oluşturulacak modelin başarımını doğrudan etkileyen bir unsurdur. Bu açıdan uygun bir yaklaşım ile veri kümeleri ele alınmalıdır. Veri analizinde yaklaşım tarzı, ulaşılmak istenen hedef, verinin bağlamı, düzeni, hacmi vb. etkenler ile şekillenmektedir. Veri analizi, probleme uygun bir biçimde sorular sorarak ve bu sorulara cevap verilerek sağlıklı bir şekilde yapılmış olur. Bu açıdan veri üzerinde yapılacak analiz altı yaklaşımla ele alınabilir [25]:

- **Tanımlayıcı Yaklaşım:** Veri kümesinin genel karakteristik yapısı özetlenmeye çalışılır. Veri kümesinin ortalaması, özellik sayıları, veri sayısı vb. nitel ve nicel bilgilerden hareketle bu veri kümesini tanımlayan özet bilgilerin ne olduğu ile ilgilenilir. Veri kümesinin dışında kalan diğer bilgiler veri kümesinin tanımlarken önemli değildir.
- **Keşifçi, Araştırıcı Yaklaşım:** Bu yaklaşımda, veri kümesinin genel yönelimi, değişkenlerin birbirleri ile olan ilişkileri vb. sahip olunan verilerden çıkarılacak bilgilerin hedefte olduğu bir veri analizi amaçlanmaktadır. Tanımlayıcı yaklaşımda olduğu gibi veri kümesinin dışında kalan veriler ile ilgilenilmez.

- **Çıkarımsal Yaklaşım:** Bu yaklaşımda, tanımlayıcı ve keşifçi yaklaşımlar ile yapılan analizlerin sonuçları kullanılmakla beraber veri kümesi dışındaki veriler hakkında da belirli önermelerde bulunulur. Bu açıdan diğer yaklaşımlara nazaran daha zorlayıcı bir analizdir. Gözlemlenemeyen veriler ile ilgili çıkarım yapıldığından dolayı kullanılacak yöntemler daha dikkatli bir şekilde belirlenmelidir.
- **Kestirimsel Yaklaşım:** Veri kümesinde trende göre gelecekte olabilecek durumların tahminine yönelik yapılan analizi kapsamaktadır. Veri kümesinde birçok farklı özellik arasındaki korelasyon göz önüne alınarak sonuca ulaşılmaya çalışılır. Var olan verilerin açıklaması ile ilgilenmek yerine sistemin nereye doğru gittiği ile alakalı sorularla ilgilenen yaklaşımdır.
- **Nedensel Yaklaşım:** Belirli bir özelliğin rastgele veya önceden tanımlı değişimi sonucu başka bir özellik veya özellikleri etkilenmesi esas alınarak nedensel bir ilişki aramaya yönelik bir yaklaşımdır.
- **Mekanik Yaklaşım:** Birbirleri arasında kontrollü bir bağlantı varmış gibi duran, başka bir ifade ile aralarında gerekirci (deterministik) bağlantılar içeren özellikleri ortaya çıkarmaya yönelik soruların sorulduğu yaklaşımdır.

1.4.6. Yapay Zekâ ve Makine Öğrenmesi

MÖ, matematiğin altında yer alan istatistik, bilgi teorisi, oyun teorisi ve optimizasyon gibi konularla ortak başlıklara sahip olmakla beraber bilgisayar bilimleri içerisinde yapay zekânın altında yer alan bir disiplin olarak düşünülebilir. Yapay zekâ insan ve hayvan zekâsının sahip olduğu mekanizmaları keşfedip benzer akıllı davranışlar gösterebilme üzerine odaklanmıştır. MÖ ise akıllı davranışın taklit edilmesini sağlamaya çalışmak yerine bilgisayarların gücü ve özel yeteneklerini kullanmaya çalışmaktadır. Bunlar genellikle insan yeteneklerini aşan konuları kapsamaktadır. Örneğin büyük bir veri tabanının analizini ancak bir bilgisayar gerçekleştirebilir [21].

Yapay zekâ ise dört yaklaşım ile özetlenebilir. İlk yaklaşım, insan gibi düşünme: İnsanların sahip olduğu düşünme, karar verme, problem çözme, öğrenme işlerinin nasıl taklit edilebileceği sorusuna cevap arar. İkinci yaklaşım, insan gibi hareket etme: Zekâ gerektiren işlemleri gerçekleştirebilecek sistemleri tasarlamak ve insanların yapmakta iyi olduğu işleri bilgisayarlara yaptırmak ile ilgili çalışmaları kapsar. Üçüncü yaklaşım, akılcı

düşünme: Hesaplama modelleri kullanarak zihinsel yeteneklerin (algılama, hareket etme vb.) araştırılması ile ilgili çalışmaları ifade eder. Dördüncü yaklaşım ise akılcı hareket etme: Akıllı etmenlerin tasarlanması ile ilgili çalışmalardır [26].

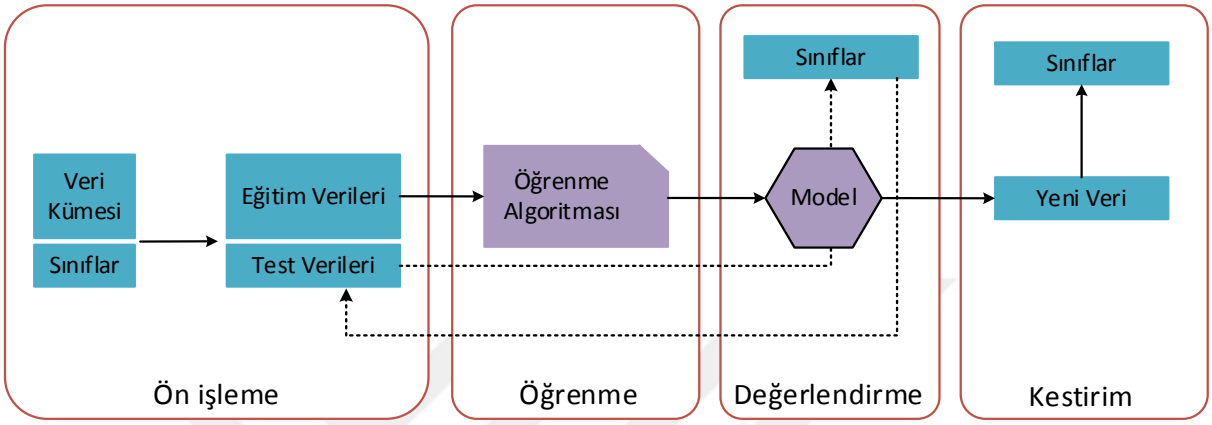
Yapay zekâ ile ilgili somut çalışmalar ikinci dünya savaşı sonrasında başlamıştır. Bu dönemde savaşta da önemli bir rol oynayan Alan Turing bir makalesinde [27] "Makineler düşünebilir mi?" sorusunu tartışmayı ister. Bunu da "Taklit Oyunu" adını verdiği bir oyun ile yapmaya çalışır. Bu oyun ayrı odalarda yer alan ve birbirleri ile sadece mesajlaşarak iletişim kuran üç kişiden oluşur. İlki erkek, ikincisi kadın üçüncü kişi ise sorgulayıcıdır ve cinsiyeti önemsizdir. Sorgulayıcıdan hangi kişinin kadın olduğunu bulması istenir. Hem erkek hem de kadın, sorgulayıcıya kendilerinin kadın oldukları konusunda ikna etmeye çalışır. Kadın doğru cevaplar verir erkek ise sorgulayıcı yanıltmaya çalışır. Turing, eğer oyunda bir makine erkeğin rolünü üstlenseydi ve sorgulayıcı yeterli zamanda bir makine ile haberleşmekte olduğunu fark etmeseydi bu durumu nasıl karşılardık? Sorusunu ortaya atar. Turing' in bu örneğine atfen bir yapay zekâ sisteminin başarısının ölçüsü olarak görülen Turing testi çeşitli şekillerde yapılmakta ve hatta yapay zekâ yarışmalarında değerlendirme kriteri olarak kullanılmaktadır [28].

Yapay zekâ ile ilgili bir başka önemli örnek ise Searle' ın "Çin Odası" [29] adındaki düşünce deneyidir. Searle, insanların makinelerden farklı olarak düşünme becerisine sahip olduğunu ve makinelerin yalnızca bir dizi adımları uygulayarak akıllı davranışlar göstermesinin, insanlarda olduğu gibi makinelerin bilince sahip olduklarını düşünmemiz için Turing testinin aksine yeterli olmayacağını savunur. Çin odası deneyi özetle şu şekildedir: İçinde yalnızca içeriye ve dışarıya kâğıt parçaları alıp vermekte kullanılan delikler bulunan kapalı bir oda olduğunu düşünelim. Odanın içerisinde Çince bilmeyen bir insan ve onun ana dilinde yazılmış bir talimatlar kitabı durmaktadır. Birisi üzerinde Çince yazılmış bir soruyu içeriye gönderir. Yeterli bir zaman geçtikten sonra içerideki kişi, üzerinde Çince yazılmış bir cevabı dışarıya yollar ve bu böyle devam eder. Odanın dışındaki kişiler, içerideki kişinin sorulara uygun cevaplar verdiği için Çince bildiğini düşünmektedir. Searle, bu noktada içerideki kişinin aldığı kâğıtlarda sadece anlaşılmaz simgeler gördüğünü ve yalnızca talimatları izleyerek cevap verdiğini söyler. Bilgisayarlarında bu şekilde düşünmeden algoritmaları uyguladığını, insanlarda olduğu gibi bir anlama işlevinin gerçekleşmediğini dolayısıyla bunun bir yapay zekâ olmayacağını iddia eder.

2. UYGULAMA TASARIM SÜREÇLERİ

2.1. Giriş

MÖ uygulama tasarımı dört aşamalı bir yapı şeklinde gerçekleştirilebilir (Şekil 2.1).



Şekil 2.1 Uygulama tasarım süreçleri [30].

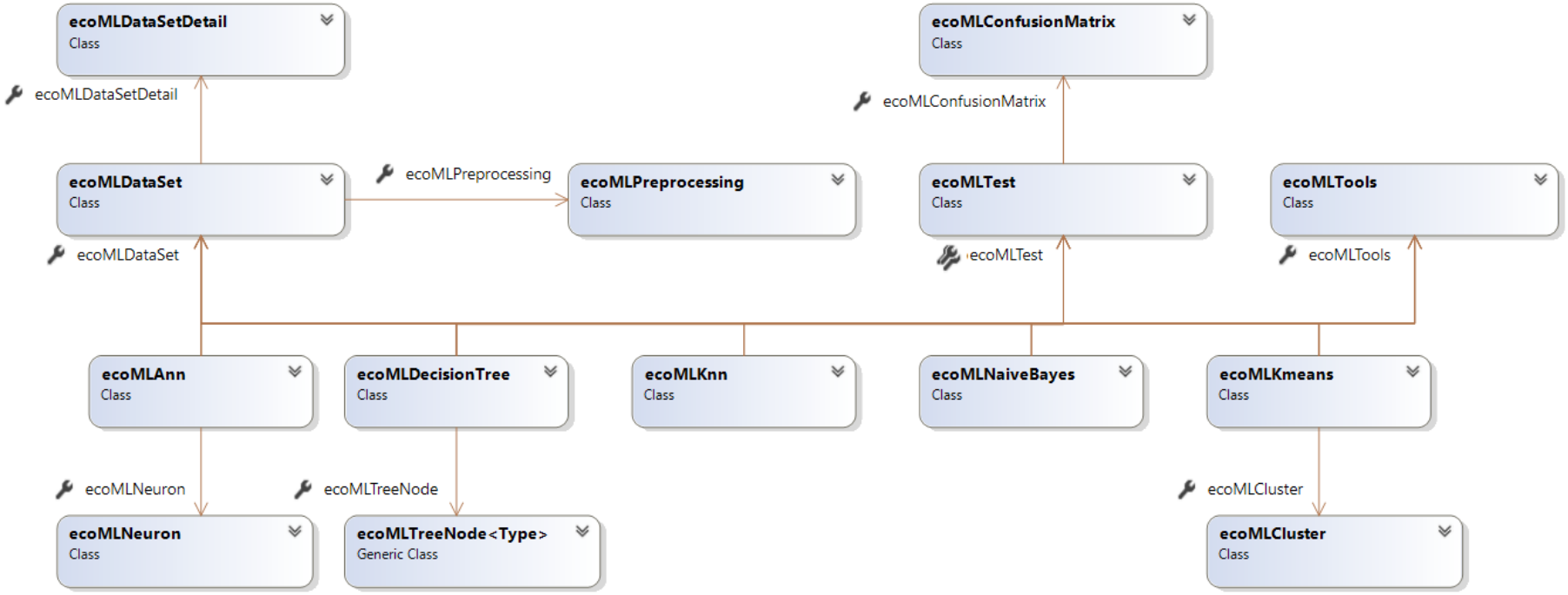
- **Ön işleme:** Veri kümeleri ilk etapta yapısına göre bir ön işlem sürecinden geçirilmektedir. Bu adım diğer adımların performansını doğrudan etkiler. Veri kümleri, eksik ve gereksiz özellikler barındırabilmekte veya MÖ algoritmasının fazla kaynak tüketmesine sebep olabilecek yapıda olabilmektedir. Eksik verilerin silinmesi veya uygun değerler ile güncellenmesi, veri analizi ile gereksiz görülen bir özelliğin veya örneklerin silinmesi, sürekli değerlerin belirli bir ölçekte normalize edilmesi veya boyut azaltma teknikleri kullanılarak veri kümesinin boyutunun düşürülmesi vb. işlemler ön işleme sürecinde yapılır.
- **Öğrenme:** Bu kısımda veri kümesinin yapısı göz önüne alınarak kullanılacak algoritmaya yönelik tercihler yapılır. Veri kümesinin özelliklerinin bütünüyle sürekli, kategorik veya bunların karması şeklinde değerlerden oluşabilmesi, veri kümesinin örnek sayısının veya özellik sayısının büyüklüğü, hedef sınıfların niteliği ve sayısı, öğrenme sürecinde performansı etkileyen faktörlerdir. Bununla birlikte öğrenme algoritmasının ne kadar sistem kaynağı (bellek kullanımı, işlemci gücü vb.) tüketeceği de öğrenme sürecinde göz önüne alınması gereken bir unsurdur. Bu faktörler değerlendirilerek gerçekleştirilen bir MÖ modeli ile öğrenme süreci gerçekleştirilir.

- **Değerlendirme:** Öğrenme için seçilen algoritma belirli parametrik değerler ile sınanır (hyperparameter optimization). Bu süreç ile optimize edilen parametreler ile son model oluşturulur. Bu model mevcut veri kümesi için optimize edilmiş en ideal model olarak kabul edilir.
- **Kestirim:** Bu kısımda daha önceki aşamalardan elde edilmiş son modelin yeni, bilinmeyen verileri tahmin etmesi şeklinde gerçekleştirilir.

2.2. Uygulama Tasarımının Gerçekleştirilmesi

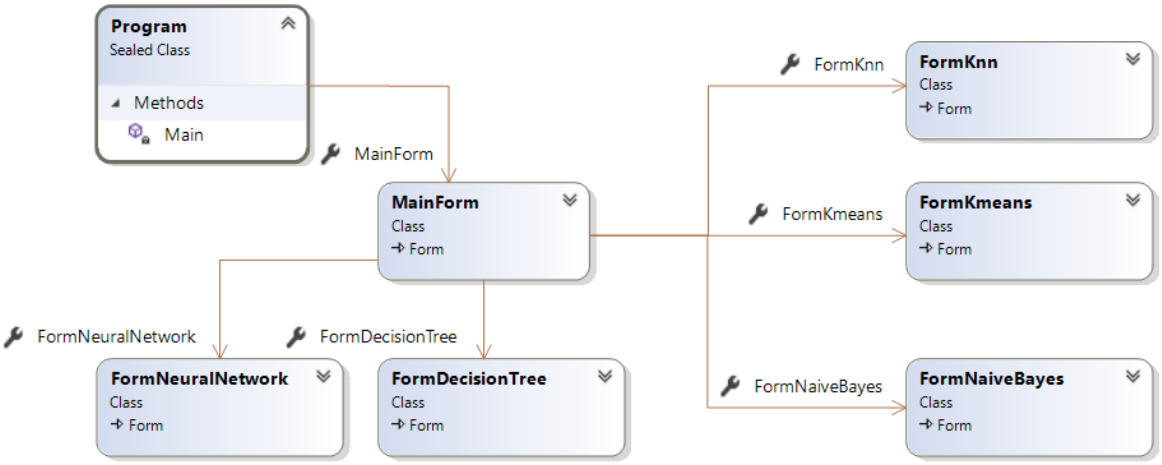
Bu kısımda 3. Bölümde anlatılan MÖ modellerinin uygulanması için bir deney ortamı oluşturulacaktır. Bu açıdan MÖ modellerini sistematik bir şekilde uygulayabilmek için "ecoMLFramework" adında bir proje gerçekleştirildi (Şekil 2.2). Test süreçleri ve sonuçlarının görselleştirilebilmesi için "ecoMLApp" adında ikinci bir proje daha gerçekleştirildi (Şekil 2.3). Uygulamalar, Microsoft Visual Studio 2017, .NET Framework ve C# dili kullanılarak gerçekleştirildi. Test edilen veri kümeleri, virgül ile ayrılmış değerlerin (CSV) olduğu bir yapı halinde "txt" uzantılı dosyalardan okunur. Veri kümelerinin temel tanımlayıcı bilgilerini (özellik isimleri, özellik tipleri, hedef özellik) ve modellere özgü parametreleri kullanıcı girer.

Gerçekleştirilen altyapı uygulaması, ecoMLFramework' un genel hatları şu şekildedir:

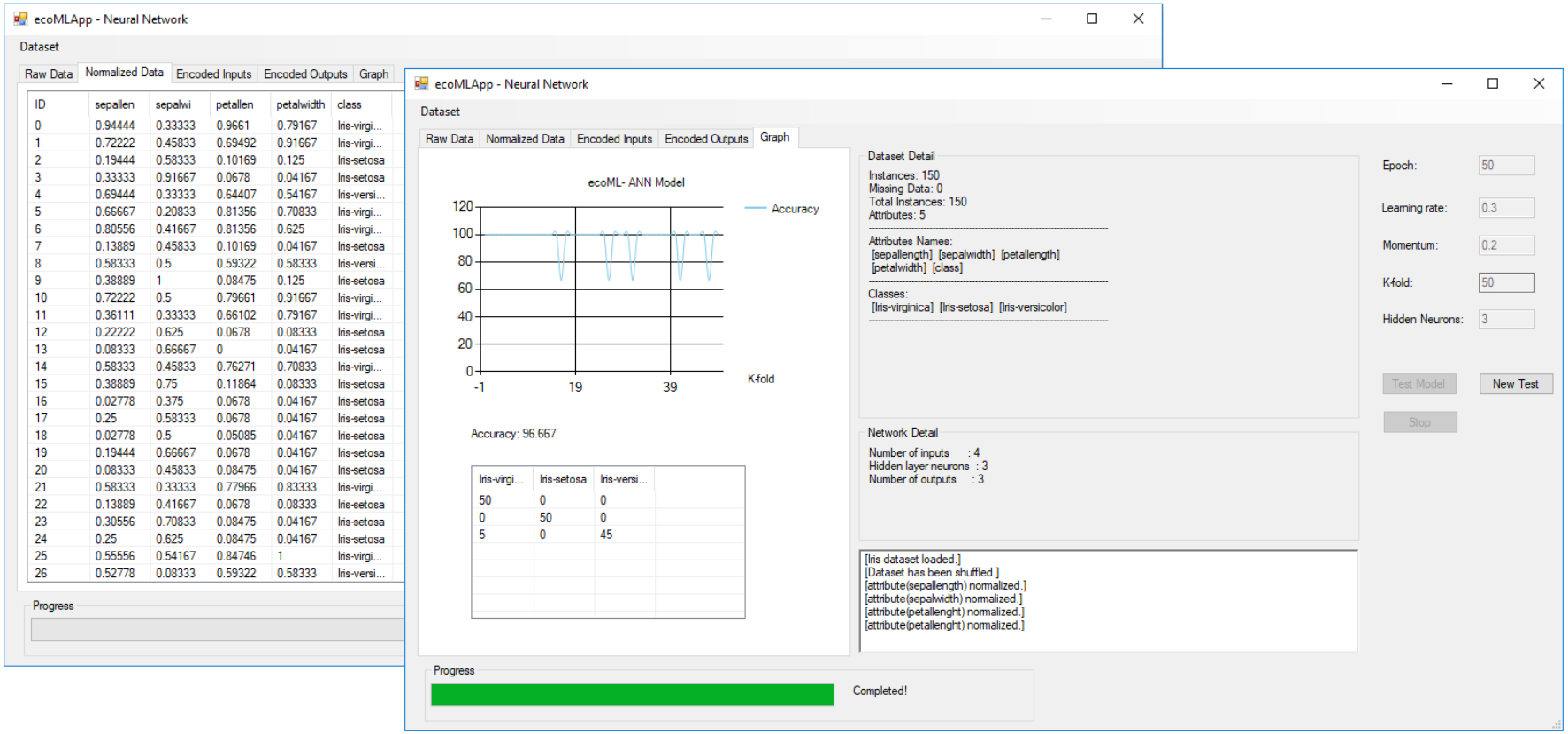


Şekil 2.2 ecoMLFramework uygulaması arabirimleri.

Deney ortamının oluşturulup testlerin yapıldığı görsel uygulama arabirimleri ve örnek arayüz görseli Şekil 2.3 ve Şekil 2.4 görülebilir.



Şekil 2.3 ecoMLApp uygulaması arabirimleri.



Şekil 2.4 ecoMLApp uygulaması örnek arayüz görseli.

2.3. Veri Kümesi

Gündelik yaşamımızda veya ekolojik sistemlerde bir MÖ problemini çözebilmek için veri olarak kullanılabilecek sayısız farklı tipte veri kümesi gözlemlenerek veya ölçülerek elde edilebilir. Çözülmek istenen probleme göre bu örneklerden en uygun olanlarının seçilmesi hedeflenir. Her bir örneğin sahip olduğu özelliklerin hepsi, örnek verinin gereksiz bilgilerden arındırılmış, başka bir ifade ile örnek verinin sıkıştırılmış haline benzemektedir. Bu örnekler, bütünsel olarak bir popülasyon şeklinde düşünülebilir. Bu noktadan hareketle popülasyona eklenen her bir örneğin diğer örneklere benzer özelliklere sahip olması beklenir. Popülasyonun veri bütünlüğünün sağlanması açısından bir standardizasyona ihtiyaç vardır. Bu açıdan tüm örnekler, bir MÖ modelinde kullanılmak üzere Tablo 2.1' deki gibi bir matris formatı halinde ifade edilebilir.

Tablo 2.1 Örnek veri kümesi.

Özellikler				Hedef
Hava Durumu	**Sıcaklık**	**Nem**	**Rüzgar**	**Golf oyna**
Yağmurlu	Sıcak	Yüksek	Yok	Hayır
Yağmurlu	Sıcak	Yüksek	Var	Hayır
Açık	Sıcak	Yüksek	Yok	Evet
Güneşli	Ilık	Yüksek	Yok	Evet
Güneşli	Serin	Normal	Yok	Evet
Güneşli	Serin	Normal	Var	Hayır

Örnekler

Veri kümesi içerisinde her bir satır bir adet örneği temsil etmektedir. Her bir sütün ise örneklere ait karakteristik bir özelliği göstermektedir. Her bir sayısal veya kategorik değer ise örneklerin sahip olduğu nitelik veya nicelik değerlerini ifade etmektedir.

2.4. Eksik Veriler

Veri kümeleri, çeşitli nedenlere bağlı olarak bazı nitelikleri özellikle girilmemiş veya gözlemlenememiş olabilmektedir. Bu nedenler, sistem hataları veya çalışma koşullarından kaynaklı algılanamayan sensör verileri olabileceği gibi bir anket çalışmasında verilmeyen cevaplar da (bilinmiyor, cevap vermeyi reddetme vb.) olabilir.

Veri kümesi üzerinde eksik verilerin ne şekilde dağılım gösterdiği, eksik veri problemini çözmek için bir gösterge olarak kullanılır. Bu açıdan eksik verilerin silinmesi veya eksik değerlerin yerine yeni değerlerin koyulması işlemleri, bu verilerin rastgeleliğine (randomness of missing data) bakılarak gerçekleştirilir. Eksik verilerin rastgeleliği üç başlık halinde incelenebilir [31]:

- **MCAR:** "missing completely at random", veri kümesinde eksik verilerin rastgelelik durumunun en yüksek düzeyde olmasını ifade eder. Eksik değerlerin olasılığı, bilinen veya eksik değerlerden bağımsız şekilde dağılım gösterir. Böylece herhangi bir silme veya değer koyma yöntemi kullanılarak eksik veri problemi çözülebilir.
- **MAR:** "missing at random", eksik verinin olasılığının bilinen verilere bağlı olması, bununla beraber eksik verinin kendisinden bağımsız olması durumudur.
- **NCAR:** "not missing completely at random", eksik verinin olasılığının bulunduğu özelliğe bağlı bir şekilde olması durumudur.

Eksik veri problemini çözmek için kullanılan yöntemler ise şu şekildedir:

- **Eksik Verinin Çıkarılması:** Bu işlem iki şekilde yapılır. İlk metot, eksik verilerin olduğu bütün örneklerin çıkarılmasıdır. İkinci yöntem ise eksik verinin olduğu özelliğin veri kümesinden çıkarılması şeklindedir. Bu yöntemler çok fazla veri kaybına sebep olabilir ve sistemin performansını olumsuz etkileyebilir. Bu nedenle bu yöntemler, eksik verilerin rastgeleliği MCAR olduğu durumda uygulanmalıdır.
- **Parametre Kestirimi:** Veri kümesinin bütünü göze alınarak olasılık fonksiyonları (expectation-maximization algoritması [32] vb.) ile eksik veriler uygun parametreler vasıtasıyla tahmin edilmeye çalışılır.
- **"Imputation":** Eksik verilerin değerleri, bilinen veriler üzerinde bir takım fonksiyonlar uygulanarak tahmin edilmeye çalışılır. En çok kullanılan yöntemlerden bazıları, ortalama alma (sürekli verilerde), en çok tekrar eden değeri alma (kategorik veya sürekli verilerde) şeklindedir [33].

2.5. Normalizasyon

Veri setleri birçok farklı ölçekte nümerik özellikler içerebilmektedir. Özelliklerden biri yaş bilgisi gibi küçük ölçekte bir dağılımdan oluşurken bir başkası bir yıllık kalp atış sayısı gibi yaş bilgisine oranla daha büyük ölçekte bir dağılım gösterebilir. Bu şekilde farklı ölçekteki nümerik özellikler, öğrenme süreci içerisinde uygulanan modeli dengesiz bir şekilde etkileyerek modelin performansını düşürebilmektedir. Çözülmek istenen problemin karakteristiğine göre nümerik özelliklerin dağılımı, belirli sınır değerler gözetilerek standart hale getirilebilir.

MÖ algoritmalarında çokça kullanılan bazı normalizasyon yöntemleri şu şekildedir:

- **Min-Max normalizasyonu:** Bu yöntemde özelliğe ait veriler, genellikle 0 ile 1 aralığında denklem (2.1)' de gösterilen şekilde ölçeklendirilerek normalize edilmektedir [34].

$$x' = \frac{x - x_{min}}{x_{maks} - x_{min}} \tag{2.1}$$

  Bununla birlikte veriler, a, b gibi bir aralık belirlenerek denklem (2.2)' deki gibi normalize edilebilir.

$$x' = a + \frac{(x - x_{min})(b - a)}{x_{maks} - x_{min}} \tag{2.2}$$

- **Z-Score Normalizasyonu:** Veri kümesinde normalizasyon uygulanmak istenen özelliğe ait her bir değer x, ortalama $\tilde{x}$ ve standart sapma S olmak üzere denklem (2.3)' deki dönüşüm fonksiyonu kullanılarak her bir değere karşılık yeni bir değer üretilir [35].

$$z = \frac{x - \tilde{x}}{S} \tag{2.3}$$

  S, N örnek sayısı olmak üzere denklem (2.4)' de gösterilmiştir.

$$S = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \tilde{x})^2}{N-1}} \tag{2.4}$$

2.6. Kategorik Veriler

Kategorik veriler, genellikle sınıflandırma algoritmalarında hedef sınıf olarak karşımıza çıktığı gibi bir veri kümesinde örneklerin nitelik özelliklerini de ifade edebilmektedir. Nitelik özellikleri, herhangi bir niceliksel ağırlığa sahip olmadıklarından bu tipte özelliklerin MÖ algoritmasında bir ağırlığa sebep olmayacak şekilde veya duruma göre bir ağırlık verilerek sayısal olarak kodlanmasına ihtiyaç duyulabilir. Bununla birlikte Karar Ağaçları gibi bazı algoritmalarda kodlamaya gerek olmadan da kullanılabilmektedir

Kategorik verilerin kodlanarak yazılması birçok şekilde yapılabilir. Bu yöntemlerden "one hot encoding", "binary encoding", "sum coding", "helmert coding" vb.bazıları K. Potdar ve ark.' nın yapay sinir ağı kullanarak yaptığı bir çalışmada karşılaştırılmıştır. Tablo 2.2' de test sonuçları görülebilir [36].

Tablo 2.2 Kodlama teknikleri karşılaştırılması.

Kodlama Tekniği	Performans
one hot encoding	% 90
ordinal coding	% 81
sum coding	% 95
helmert coding	% 89
polynominal coding	% 91
backward difference coding	% 95
binary coding	% 90

Üç farklı kodlama tekniği, Tablo 2.3' deki örnek bir veri kümesi kullanılarak Tablo 2.4' de gösterilmiştir.

Tablo 2.3 Iris veri kümesi.

ID	Sepal Length	Sepal Width	Petal Length	Petal Width	Tür
1	5.1	3.5	1.4	0.2	Iris Setosa
⋮					
50	6.4	3.5	4.5	1.2	Iris Versicolour
⋮					
150	5.9	3	5	1.8	Iris Virginica

Tablo 2.4 Iris veri kümesinde sınıfların kodlanması.

Tür	Onlu (Decimal)	İkili (Binary)	"One hot encoding"
Iris Setosa	0	000	001
Iris Versicolour	1	001	010
Iris Virginica	2	010	100

3. MAKİNE ÖĞRENMESİ MODELLERİNİN UYGULANMASI

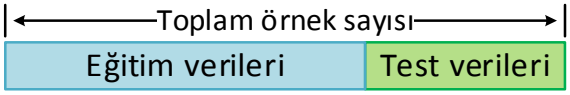
Bu bölümde 5 adet MÖ tekniğinin uygulaması gerçekleştirilmiştir. Bu tekniklerden Yapay Sinir Ağları (YSA), K-NN, Naïve Bayes (NB) ve ID3 karar ağacı kullanılarak sınıflandırma, K-Means tekniği ile de kümeleme işlemleri gerçekleştirilmiştir.

3.1. Model Başarımının Ölçülmesi

Model başarımını ölçmek MÖ uygulamalarının önemli bir aşamasıdır. Bu aşama ile modelin geliştirilmesi, başka bir ifade ile modelin, örnek uzay içerisinde göz ardı edilebilecek hata oranına yaklaşabilmesi hedeflenmektedir. Veri kümesinin boyutu, kullanılan algoritmaya uygunluğu, sınıf dağılımı vb. unsurlar öğrenme algoritmasının başarımını doğrudan etkilemektedir. Bu noktada veri kümesinin nasıl ele alınacağı sorusu önem kazanmaktadır. Veri kümesi test ve öğrenme verileri şeklinde çeşitli tekniklerle ayrılarak algoritmaya giriş olarak verilir. Bu tekniklerden üçü, "Holdout", "Three-way Split" ve "K-fold Cross Validation" (K-katmanlı Çapraz Doğrulama) aşağıda anlatılmıştır. Bunlardan K-katmanlı Çapraz Doğrulama, MÖ uygulamalarında kullanılmıştır [37].

3.1.1. Holdout

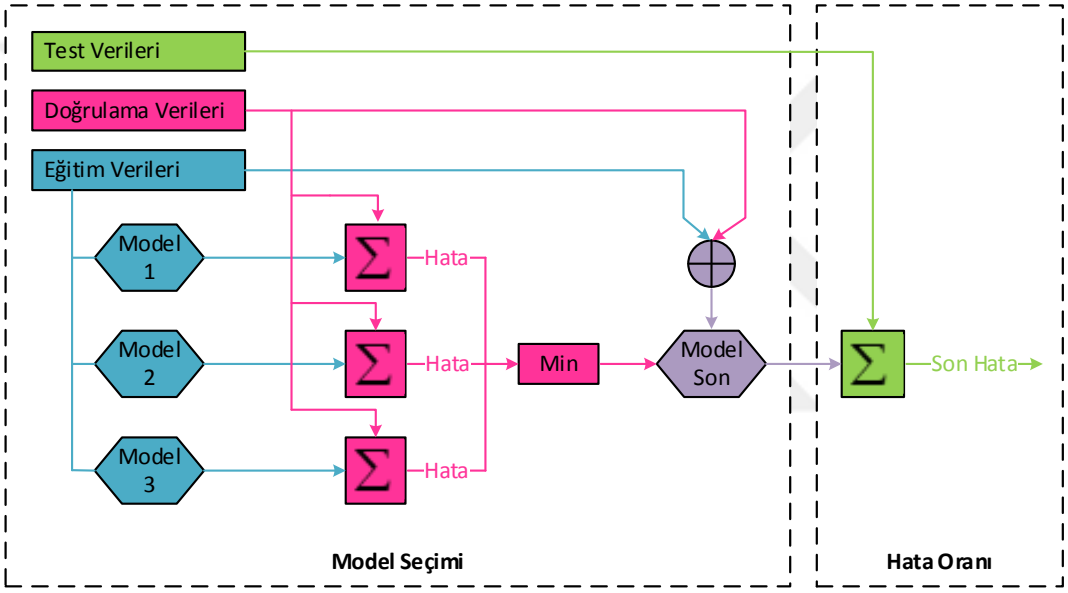
Bu teknikte ile eğitim verileri ve test verileri belirli bir oranda ayrılır ve sınıflandırma algoritmasında uygulanarak test edilir. Test verileri eğitim verilerine oranla daha küçük bir orandadır (Şekil 3.1). Veri kümesinin hacmi yeterince büyük değilse, sınıflandırma algoritması test verilerini öğrenmede kullanmadığı için sınıflandırıcı performansı düşebilmektedir. Kötü bir bölümlendirme yapılması halinde, tek bir test süreci işletildiği için sınıflandırma algoritmasının yanıltıcı bir hata oranı hesaplamasına sebep olabilmektedir.



Şekil 3.1 Holdout tekniği.

3.1.2. Three-way Split

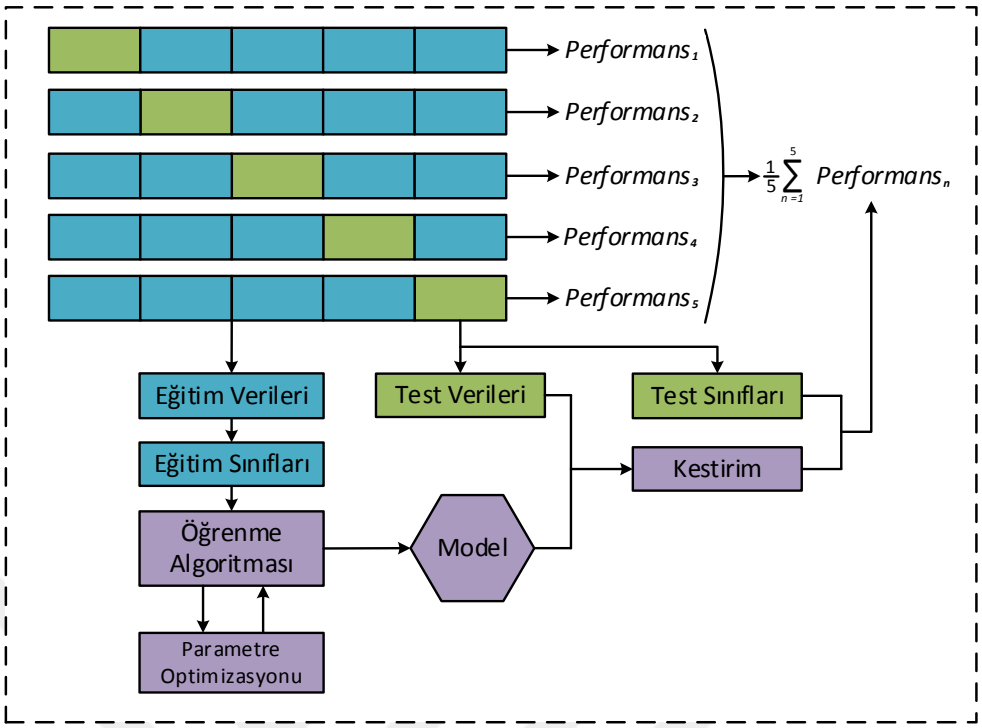
Bu teknikte veri kümesinin bölümlendirme işlemi uygun sınıflayıcı modelin seçimi ile paralel hareket eder. Veri kümesi, eğitim kümesi, doğrulama kümesi ve test kümesi olarak üçe ayrılır. Şekil 3.2‘ de görülebileceği gibi doğrulama kümesi uygun modelin seçimi için parametre görevi görür. Doğrulama kümesi eğitim kümesi ile birlikte birden çok modelin performanslarının hesaplanması işin kullanılır. Sonuç olarak en düşük hata oranına sahip model ideal model olarak seçilir. Bu noktada parametreleri belirlenmiş son modelin performansı, test kümesi kullanılarak hesaplanır [38].



Şekil 3.2 Three-way-split tekniği [37].

3.1.3. K-katmanlı Çapraz Doğrulama

Bu yöntemde veri kümesi *k* kadar parçaya bölünür. Her bir bölüm sıra ile test kümesi, geri kalan parçalar ise birleştirilerek eğitim kümesi olarak kullanılır (Şekil 3.3). Bu şekilde sınıflayıcının bütün veri kümelerini hem test hem de eğitim verisi olarak kullanması sağlanarak örnek uzayın daha verimli bir şekilde test edilmesi amaçlanır. Hata oranı bütün hataların ortalaması (denklem (3.1)) alınarak elde edilir [37].



Şekil 3.3 K-katmanlı çapraz doğrulama [39].

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i \tag{3.1}$$

3.1.4. Hata Matrisi

Sınıflandırma algoritmalarının performansını gözlemleyebilmek için kullanılan, "Confusion Matrix", "Error Matrix" adlarıyla bilinen bir hata gösterim yöntemidir. Sınıflandırma başarısının her bir sınıf için ne ölçüde olduğu bilgisi, kullanılan modeli geliştirmek veya değiştirmek için etkili bir göstergedir. Bu açıdan hata matrisi, kullanıcıya modelin nasıl bir trend izlediği, hangi tip hataların yapıldığı vb. konularda daha iyi bir bakış açısı sunmaktadır [40].

Hata matrisi test verileri üzerinden iki boyutlu bir matris ile ifade edilir. İlk boyut test verilerinin gerçek sınıflarını temsil eder. İkinci boyut ise modelin tahmin ettiği sınıfları göstermektedir. Şekil 3.4' de üç sınıflı bir hata matrisi gösterilmiştir. İlk satırda *A* sınıfına ait sınıflandırılma sonuçları gözlemlenebilir. 13 adet *A* sınıfına ait test verisinin 10' u doğru sınıflandırılmışken ikisi *B* sınıfı, bir örnek ise *C* olarak sınıflandırılmıştır. Bu şekilde

diğer satırlarda dikkate alınırsa bu algoritma için başarı oranı, matris üzerinde sol üst köşeden sağ alt köşeye doğru çizilen köşegen üzerinde bulunan sayıların toplamının tüm test verileri sayısına oranı olarak ifade edilir.

		Tahmin Edilen Sınıflar		
		A	B	C
Gerçek Sınıflar	A	10	2	1
	B	0	6	1
	C	0	3	8

Şekil 3.4 Hata matrisi.

Hata matrisi, sınıfların her biri için doğru ve hatalı sınıflandırılma sayılarının olduğu iki boyutlu bir matris şeklinde de yazılabilir. Bu şekilde algoritmanın performans analizi her bir sınıf için de yapılabilir. *A* sınıfı için hata matrisi Şekil 3.5' de gösterilmiştir.

		Tahmin Edilen Sınıflar	
		Pozitif	Negatif
Gerçek Sınıflar	Pozitif	TP	FN
	Negatif	FP	TN

A sınıfı →

		Tahmin Edilen Sınıflar	
		Pozitif	Negatif
Gerçek Sınıflar	Pozitif	10	3
	Negatif	0	18

Şekil 3.5 *A* sınıfı için hata matrisi.

- *T (True),* doğru sınıflandırmayı, *F (False)* ise hatalı yapılan sınıflandırmayı temsil etmektedir.
- *P (Positive), A* sınıfına ait örnek sayısı, *N (Negative)* ise diğer sınıfların toplam örnek sayısıdır.
- *TP (True-Positive),* algoritmanın *A* **sınıfı** olarak hesapladığı çıktının gerçek değerinin de *A* sınıfı **olduğu** sonuç sayısını ifade etmektedir.
- *TN (True-Negative),* algoritmanın *A* **sınıfından farklı** hesapladığı çıktının gerçek değerinin de *A* sınıfı **olmadığı** sonuç sayısını ifade etmektedir.

- *FN (False-Negative),* algoritmanın *A* **sınıfından farklı** hesapladığı çıktının gerçek değerinin *A* sınıfı **olduğu** sonuç sayısını ifade etmektedir.
- *FP (False-Positive),* algoritmanın *A* **sınıfı** olarak hesapladığı ve gerçek değerinin *A* sınıfı **olmadığı** sonuç sayısını ifade etmektedir.

A sınıfına ait hata matrisinden elde edilen sonuçlar ise şu şekildedir:

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{3.2}$$

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} \tag{3.3}$$

$$FPR = \frac{FP}{N} = 1 - TNR \tag{3.4}$$

$$FNR = \frac{FN}{P} = 1 - TPR \tag{3.5}$$

$$ACC = \frac{TP + TN}{P + N} \tag{3.6}$$

- *TPR (True positive rate - sensivity)*, algoritmanın gerçek çıkışın *A* sınıfı **olduğu** durumlarda bu sınıfı **doğru** tahmin etme oranı.
- *TNR (True negative rate - specificity)*, algoritmanın gerçek çıkışın *A* sınıfı **olmadığı** durumlarda çıkışın *A* sınıfı olmadığını **doğru** tahmin etme oranı.
- *FPR (False positive rate)*, algoritmanın gerçek çıkışın *A* sınıfı **olduğu** durumlarda bu sınıfı **yanlış** tahmin etme oranı.
- *FNR (False negative rate),* algoritmanın gerçek çıkışın *A* sınıfı **olmadığı** durumlarda çıkışın *A* sınıfı olmadığını **yanlış** tahmin etme oranı.
- *ACC (Accuracy), A* sınıfı için algoritmanın başarı oranını gösterir.

3.2. Yapay Sinir Ağları

Bilgisayarların insan gibi düşünebilmesini gerçekleştirmek için insan beyninin çalışma prensiplerinden faydalanılarak modellenmeye çalışılmıştır. Bununla birlikte insan beyni modellenemeyecek kadar karmaşık bir yapıya ve çok fazla hücre sayısına sahiptir. Bu sebeple bilim insanları, beyni oluşturan nöron adında sinir hücrelerinin davranışlarını modelleyerek Yapay Sinir Ağları (YSA)' nı geliştirmişlerdir [41]**.**

YSA, biyolojik beyinlerin çalışma şeklinden ilham alınarak geliştirilmiş, makine öğrenmesinde sık kullanılan bir öğrenme modelidir. Biyolojik çalışan sitemlerde olduğu gibi algılayıcılardan alınan bir dizi giriş sinyali işlenmek üzere alınır. Bu sinyaller nöron adı verilen düğümlerden geçer. Belirli bir eşik fonksiyonu uygulanması sonucunda çıkış sinyalleri elde edilir.

YSA, birbirlerine çoklu bağlantı yapmış nöronların oluşturduğu karmaşık bir yapı halinde paralel işlemciler gibi çalışan çok yönlü bir MÖ tekniğidir. Sınıflandırma, nümerik kestirim (numerical prediction), denetimsiz öğrenme, örüntü tanıma vb. alanlarda kullanılabilir [42].

Bilgisayar programları ile belirli bir akış içerisinde kodlanarak çözülebilen problemlerin YSA ile çözülmesi uygun değildir. Bir problemin çözüm adımları rahat bir şekilde belirlenebiliyorsa normal teknikler çözüm için yeterlidir. Bununla birlikte bir programın algoritmasında değişiklik yapılma sayısı yüksek ise bu durumda YSA, öğrenme özelliği sayesinde bu tarz problemleri çözmede daha başarılı olabilmektedir. Değişmeyen kurallardan oluşuyorsa YSA' yı kullanmak için bir neden yoktur.

Geleneksel yöntemlerle çok fazla kod yazılarak oluşturulan uygulamalar YSA ile birkaç satır kodla gerçekleştirilebilir. Dolayısıyla hangi problemlerin YSA' ya uygun olduğuna karar vermek önemlidir. YSA en çok örüntü tanıma, sınıflandırma ve veri madenciliği gibi sıralı akış içeren programlama ile çözülemeyen alanlarda kullanılır [41]**.**

YSA, öğrenme prosedürü sonucu veriler üzerinde genelleştirme ve ilişkilendirme yeteneğine sahip olur. Başarılı bir eğitimden sonra benzer sorunlar (test verileri) için makul çözümler elde edilebilmektedir. [43]**.**

YSA' yı oluşturan temel unsurlar ise şu şekildedir:

- Bilgi işlemenin gerçekleştiği nöron adı verilen birçok temel birimden oluşur (Mimari yapı).

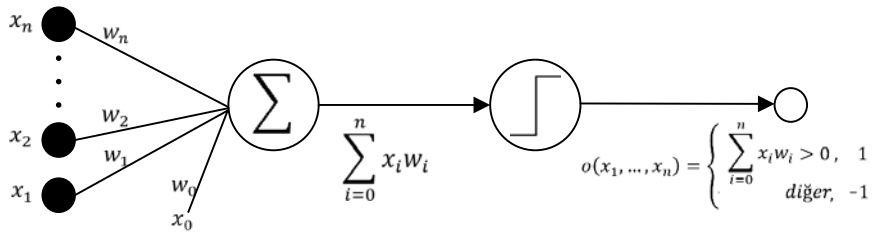
- Nöronlar arası bağlantı hatları boyunca sinyaller iletilir. (Algoritma, eğitme veya öğrenme).
- Her bir bağlantı hattı bağlılık durumunu gösteren nümerik bir ağırlığa sahiptir. Bu ağırlık sinyal değeri ile çarpılır.
- Her bir nöron, giriş sinyallerinin toplamını genellikle doğrusal olmayan bir aktivasyon fonksiyonuna giriş olarak kullanıp bir çıkış sinyali üretir (Aktivasyon fonksiyonu).

3.2.1. Nöron

YSA, nöron adında birimlerden oluşur. Bunlar, biyolojik sistemlerdeki nöronlara karşılık düşünebilinir. Nöronların giriş değerleri ilişkili ağırlık değerleri ile çarpılarak elde edilen sonuçlar toplanır ve bir eşik değere göre bir çıktı üretilir. Giriş değerleri x_1 ile x_n arasında, ağırlıklar $(w_0 \ldots w_n)$ olmak üzere çıkış fonksiyonu $o(x_1, \ldots, x_n)$, denklem (3.7)' de gösterilmiştir [20].

$$o(x_1, \ldots, x_n) = \begin{cases} 1, & w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1, & diğer \end{cases} \quad (3.7)$$

w_0 değeri nöronun eşik değerini temsil eder. w_0 değeri negatif olarak değerlendirilir ve bu değerin diğer ağırlık-giriş çarpımları ile toplamı sıfırdan büyük ise o fonksiyonu 1 çıkışını üretir (nöron aktif olur). Aksi durumda ise -1 değerini üretir. Bu şekilde iki çıktı üreten aktivasyon fonksiyonlarına basamak fonksiyonu (step function) denir.



Şekil 3.6 Yapay nöron modeli [20].

3.2.2. Eğitim

Tek nöron ile eğitim mekanizması, eğitim kümesi örneklerine uygulanan ağırlık vektörlerinin doğru çıktı $(1, -1)$ üretmesi için ağırlıkların çevrimsel olarak değiştirilmesi üzerine kuruludur. Tek nöron ile eğitimin yöntemi daha sonra ele alınacak çoklu nöron veya çok katmanlı YSA' da eğitimin nasıl olacağına dair temel teşkil edecektir.

Doğru ağırlıkların bulunması işlemi ağırlık vektörlerinin rastgele bir şekilde belirlenmesi ile başlar. Eğitim kümesindeki her bir örnek nörona giriş değeri olarak verilerek çıkış hesaplanır. Ağırlıklar sınıflandırma hataları göz önüne alınarak değiştirilir. Bu işlem bütün eğitim kümesi belli bir oranda doğru sınıflandırılana kadar devam eder. Ağırlıklar, nöron eğitim kuralına (denklem (3.8)) göre her adımda yeniden belirlenir.

$$w_i \leftarrow w_i + \Delta w_i$$
$$\Delta w_i = \eta(t - o)\, x_i \tag{3.8}$$

Denklem (3.8)' de t, eğitim örneğine ait hedef özelliği gösterir. o, nöronun ürettiği çıkış değeridir. x_i, eğitim örneğinin giriş değerini temsil eder. η değeri ise öğrenme sabitidir. η sabiti ağırlıkların her adımda değişiminin yumuşatılmasında kullanılır. Genellikle 0,1 gibi küçük bir değer alır. Eğitim sırasında eğitim örneklerinin doğru sınıflandırıldığı düşünülürse $t - o$ sıfır olacaktır ve ağırlık değişimi olmayacaktır.

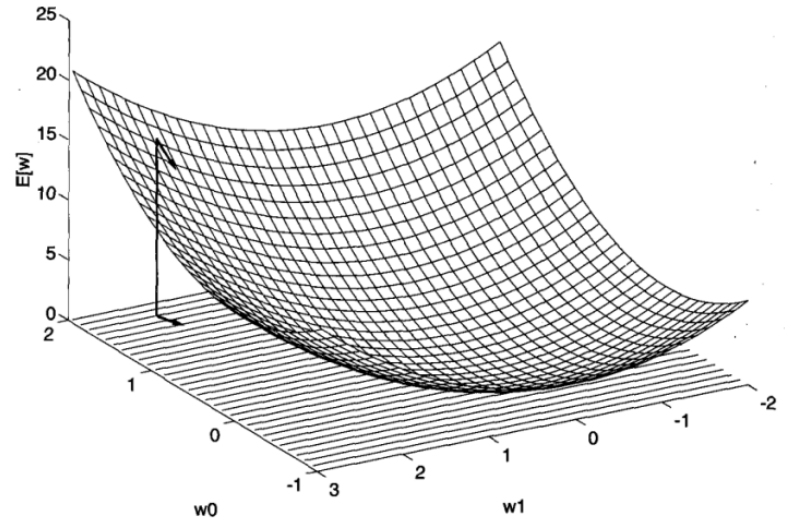
Nöron kuralı, eğitim örnekleri doğrusal olarak sınıflandırılabiliyorsa eğitimin başarılı bir şekilde yapılabilmesi için yeterli olabilmektedir. Aksi durumda ise başarısız olacaktır. Bu durumda ikinci bir yöntem olarak Delta Kuralı(DK), hedef fonksiyona yakınsama açısından daha iyi bir yaklaşım sağlar [20].

DK' nın çalışma prensibi Meyilli Azalım (Gradient Descent) kavramına dayanır. Bu açıdan DK, Meyilli azalım (MA) kuralı şeklinde de adlandırılır. MA, arama uzayında eğitim örnekleri ile uyuşan olası ağırlık vektörlerini belirlemek için kullanılır. MA daha sonra ele alınacak geriye yayılım algoritmasının da temelini oluşturur.

Eğitim örnekleri üzerinden elde edilecek hatanın hesaplanması için kullanılacak yöntem şu şekildedir:

$$E(\vec{w}) = \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2 \tag{3.9}$$

D eğitim örnekleri kümesini ifade eder. t_d, d. örnek için hedef çıkışı temsil eder. o_d ise d. örneğin gerçek çıkışını göstermektedir. Ağırlık vektörüne bağlı olarak çıkış değerleri belirlendiği için E hata miktarı $\vec{w}$ ' ye bağlı bir fonksiyon olmalıdır. Ayrıca eğitim örneklerinin çıkışları (o) ile hata oranı tespit edildiği için bu çıkışlar da fonksiyonda kullanılmıştır. Sonuç olarak DK ile E hata miktarı minimize edilmeye çalışılır.



Şekil 3.7 Hatanın minimize edilmesi [20].

E hata değerini minimize etme süreci MA ile bir ağırlık vektörü tespit edilmesi ile başlar. Ardından bu süreç ağırlık vektörü üzerinde her adımda küçük değişiklikler yapılarak hata yüzeyi üzerinden en uç noktaya (global minimum) ulaşıncaya kadar sürmektedir [20].

MA, hata yüzeyi boyunca E nin her bir ağırlığa göre kısmi türevine göre hesaplanması ile gerçekleştirilir. E vektörün türevi, E nin $\vec{w}$' ye göre gradyanı şeklinde ifade edilir ve $\nabla E(\vec{w})$ şeklinde yazılır.

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right] \tag{3.10}$$

Ağırlık vektörleri gibi $\nabla E(\vec{w})$ de aynı zamanda bir vektördür ve bu vektörün negatifi düşüşün doğrultusunu göstermektedir. Örneğin Şekil 3.7' te görülen ok işareti, w_0, w_1

düzleminde, gösterdiği noktaya karşılık negatif gradyanı (*-E(w)*) göstermektedir. MA için eğitim kuralı ise şu şekildedir:

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$
$$\Delta\vec{w} = -\eta \nabla E(\vec{w}) \tag{3.11}$$

η pozitif öğrenme hızı sabitidir ve MA aramasında adım boyutunu belirler. Denklemde negatif olarak alınmasının sebebi ağrılık vektörlerinin E' yi minimize etmek üzere belirlenebilmesini sağlamaktır. DK kuralı denklem (3.12)' de gösterilmiştir.

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id} \tag{3.12}$$

DK (veya MA) kuralı ile eğitim şu şekilde gerçekleşmektedir:

- Başlangıç için rastgele ağırlık vektörü seçilir.
- Her bir eğitim örneğine ağırlıklar uygulanır.
- Her ağırlık için denklem (3.12) kullanılarak Δw_i hesaplanır ve ağırlıklara eklenerek ağırlıklar güncellenir.
- Aynı adımlar tekrar eder.

Yukarıda belirtilen kurallara göre eğitim algoritması aşağıda yazılmıştır. Algoritma, global minimum hata değerine ulaştığında bir ağırlık vektörüne yakınsayacaktır [20].

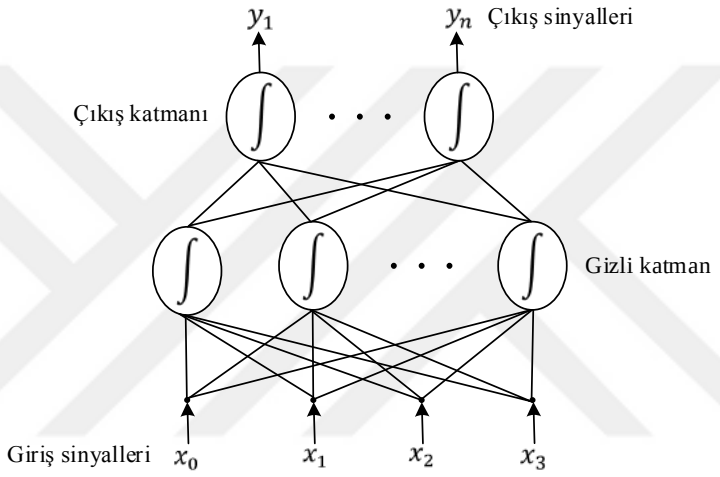
MaKuralı (D, n)

"Her bir eğitim örneği d, (x, t) *formundan oluşur.* x *giriş vektörünü temsil eder.* t *hedef çıkışı* η *ise öğrenme hızı sabitidir."*

- *Her bir* w_i *ağırlığı rastgele bir değer verilir.*
- *Bitiş koşulu sağlanıncaya kadar çevrim yap.*
  - *Her bir* Δw_i *değerini sıfırla.*
  - *Her* (x, t) *örneği boyunca çevrim yap.*
    - x *giriş vektöründen* o *çıkış değerini hesapla.*
    - *Her* w_i *değeri için* Δw_i *değerini hesapla (*$\Delta w_i \leftarrow \Delta w_i + n(t - o)x_i$*).*
  - *Her* w_i *değeri için güncelleme yap (*$w_i \leftarrow w_i + \Delta w_i$*).*

3.2.3. Çok Katmanlı Ağlar

Çok katmanlı ağlar, genellikle üç katmanlı bir yapıda veya daha fazla katman kullanılarak oluşturulan Şekil 3.8' de bir örneği gösterilen YSA modellemesidir. Giriş katmanı gerçek değerleri giriş olarak alan nöronlardan oluşur. Gizli katman ise giriş katmanından gelen sinyaller ile beslenen nöronlardan oluşur. Gizli katman sayısı, problemin çözümündeki başarısına göre artırılabilir. Son olarak çıkış katmanında bulunan nöronlardan çıkış sinyali elde edilir.



Şekil 3.8 Çok katmalı ağ modeli [42].

Şekil 3.8' de dikkat edilirse aynı katmanda bulunan nöronlar arası hiçbir bağlantı mevcut değildir. Bununla birlikte komşu katmanlar arasında bütün nöronlar birbirleri ile tamamen bağlıdırlar [42]**.**

3.2.4. Aktivasyon Fonksiyonu

Çok katmanlı YSA' da kullanılan çeşitli aktivasyon fonksiyonları vardır. Bunlardan en yaygın olanı Sigmoid olarak adlandırılan fonksiyondur.

$$o = \sigma(\vec{w}.\vec{x}) \tag{3.13}$$

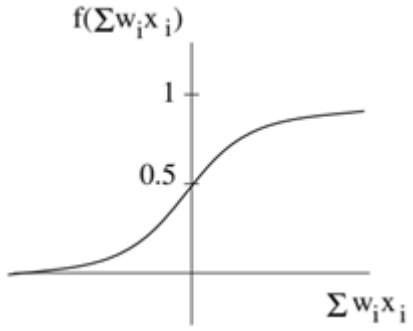
o, çıkış olmak üzere σ sigmoid fonksiyonu şu şekildedir:

$$\sigma(y) = \frac{1}{1 + e^{-y}} \tag{3.14}$$

Bu denklemin türevi ise şu şekildedir:

$$\frac{d\sigma(y)}{dy} = \sigma(y).(1 - \sigma(y)) \tag{3.15}$$

Sigmoid fonksiyonun grafiği ise Şekil 3.9' da gösterilmiştir.



Şekil 3.9 Sigmoid fonksiyonu [42].

Sigmoid fonksiyonuna göre ağırlıklandırılmış nöron giriş değerleri toplamı 0 ise bu fonksiyon 0.5 şeklinde çıktı üretir. Fonksiyon artı sonsuza giderken limiti 1 dir. Eksi sonsuza giderken ise limiti 0 dır. Sigmoid fonksiyonu 0 ile 1 arasında bir çıktı üretmektedir (denklem (3.16)) [42].

$$\begin{aligned} \sigma(-\infty) &= 0 \\ \sigma(\infty) &= 1 \\ f(0) &= 0.5 \end{aligned} \tag{3.16}$$

3.2.1. Geriye Yayılım Algoritması

Çok katmanlı ağlarda kullanılan bir yöntemdir. Ağdaki hatayı azaltmak için MA kullanır. Birden fazla çıkış değeri olabileceğinden hata fonksiyonu denklem (3.17)‘ de gösterilmiştir.

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in Cikislar} (t_{kd} - o_{kd})^2 \tag{3.17}$$

k çıkış birimlerini temsil etmektedir. t_{kd} ve o_{kd} ise d. eğitim örneğinde k. çıkış biriminin hedef ve çıkış değerlerini göstermektedir. Geriye yayılım yöntemi ile ağdaki bütün birimler üzerinden toplam hata miktarı hesaplanır. Denklem (3.18)’ de her bir örnek için uygulanacak hata fonksiyonu gösterilmiştir.

$$E_d(\vec{w}) = \frac{1}{2} \sum_{k \in Cikislar} (t_k - o_k)^2 \tag{3.18}$$

Her bir çıkış birimi için uygulanacak hata terimi şu şekildedir.

$$\delta_k = o_k(1 - o_k)(t_k - o_k) \tag{3.19}$$

Her bir gizli katman birimi için uygulanacak hata terimi şu şekildedir.

$$\delta_h = o_h(1 - o_h) \sum_{k \in Cikislar} w_{kh}\delta_k \tag{3.20}$$

Denklem (3.19), denklem (3.20)’ de yerine yazılırsa denklem (3.21) elde edilir.

$$\delta_h = o_h(1 - o_h) \sum_{k \in Cikislar} w_{kh} o_k(1 - o_k)(t_k - o_k) \tag{3.21}$$

Yeni ağırlıkların belirlenmesi için öğrenme sabiti ve giriş sinyali de eklenerek sonuçta denklem (3.22) elde edilir.

$$
\begin{aligned}
w_{ji} &\leftarrow w_{ji} + \Delta w_{ji} \\
\Delta w_{ji} &= -\eta \frac{\partial E_d}{\partial w_{ji}} \\
&= \eta \delta_J x_{Ji} \\
&= \eta(1 - o_j) o_j (t_j - o_j) x_{Ji}
\end{aligned}
\tag{3.22}
$$

Elde edilen bağıntılar ile oluşturulan geriye yayılım algoritması şu şekildedir [20]:

GeriyeYayılım $(D, n, l_{giris}, l_{cikis}, l_{gizli})$

"Her bir eğitim örneği d$(\vec{x}, \vec{t})$ *formundan oluşur.* $\vec{x}$ *giriş vektörünü temsil eder.* $\vec{t}$ *hedef çıkışı* n *ise öğrenme hızı sabitidir.* l_{giris} *giriş katmanı birim sayısını,* l_{cikis} *çıkış katmanı birim sayısını ve* l_{gizli} *ise gizli katmanın birim sayısını temsil etmektedir.* i*. giriş biriminin j. ara katmana bağlantı değeri* x_{ji}*, ağırlığı ise* w_{ji} *olarak ifade edilmektedir."*

- l_{giris}, l_{cikis} *ve* l_{gizli} *sayılarını kullanarak ileri beslemeli bir ağ oluştur.*
- *Bütün ağın ağırlık değerlerini* $(-0.5, 0.5)$ *gibi küçük bir aralıkta rastgele bir şekilde belirle.*
- *Bitiş koşulu sağlanıncaya kadar çevrim.*
  - *D eğitim kümesi içerisinde her bir* $(\vec{x}, \vec{t})$ *örneği için çevrim.*
    - x *giriş vektöründen ağa ait bütün çıkış değerlerini hesapla (ağ boyunca ileri doğru yayılım).*
    - *Her bir k çıkış birimi için* δ_k *(denklem (3.19)) hata terimini hesapla (ağ boyunca geriye doğru yayılım).*
    - *Her bir h gizli birim için* δ_h *(denklem (3.20)) hata terimini hesapla (ağ boyunca geriye doğru yayılım).*
    - *Her bir* w_{ji} *değerini* Δw_{ji} *(denklem (3.22)) ile güncelle (ağ boyunca geriye doğru yayılım).*

3.2.2. Momentum

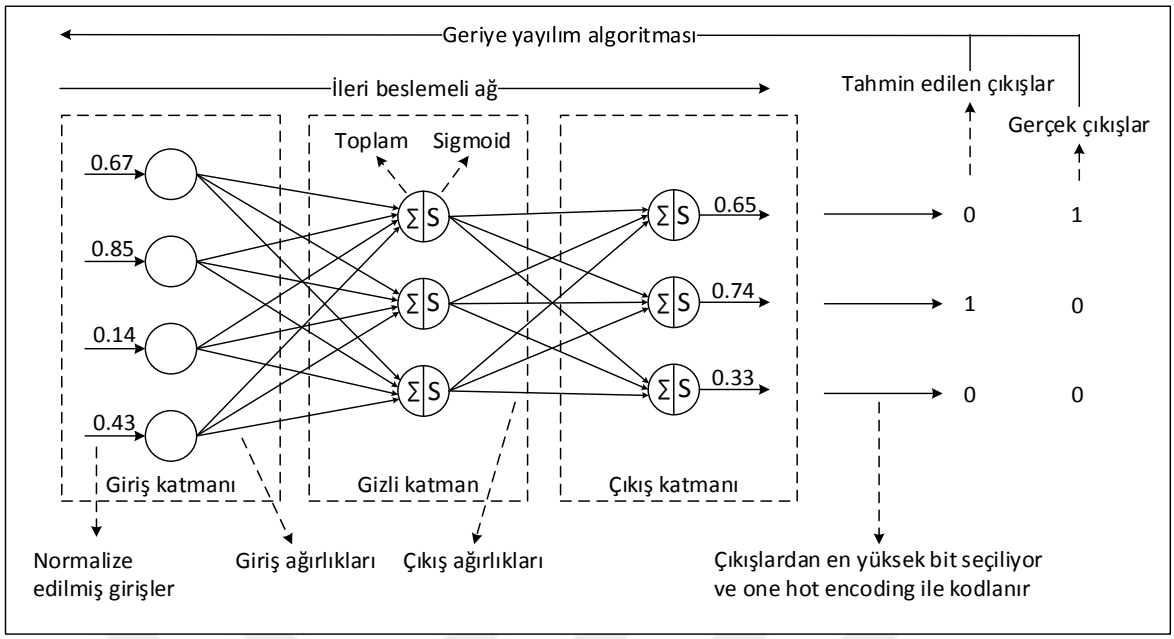
Momentum, Δw_{ji} ağrılık değişiminin n. iterasyonu $\Delta w_{ji}(n)$ olmak üzere her bir iterasyonda bir önceki değişimin ($\Delta w_{ji}(n-1)$), bir α sabiti oranında $\Delta w_{ji}(n)$ değişimine etki etmesi olarak özetlenebilir. Momentum sabiti $0 \leq \alpha < 1$ aralığında seçilir. Denklem (3.23)' de gösterilmiştir.

$$\Delta w_{ji}(n) = \eta \delta_J x_{Ji} + \alpha \Delta w_{ji}(n-1) \tag{3.23}$$

Momentumun, hata yüzeyinde yerel minimum noktalarına doğru hareketin devam ettirilmesi ve hata yüzeyinde eğimin değişmediği durumlarda arama boyutunu arttırarak uygun yörüngeye yaklaşımı hızlandırmak için kullanılır [20].

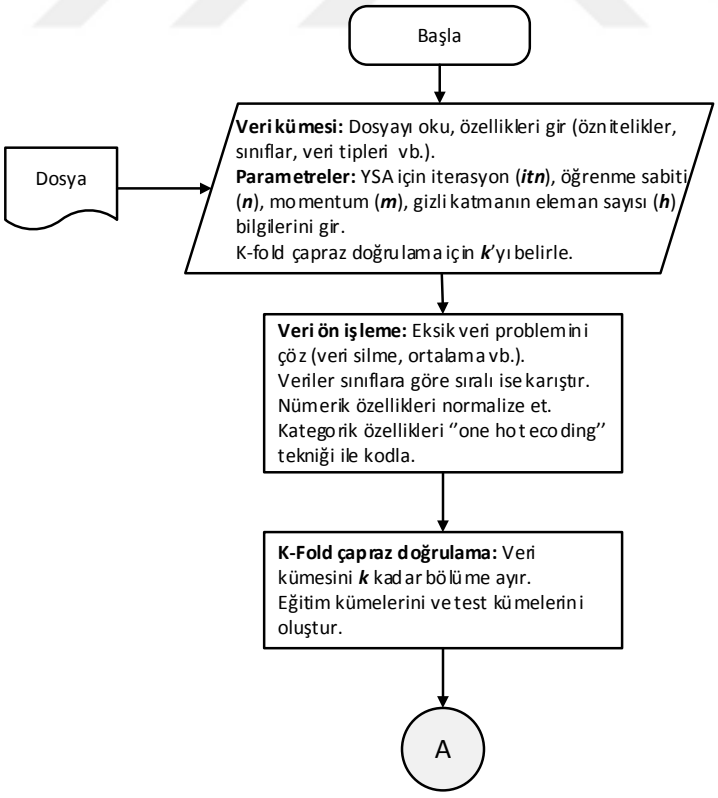
3.2.3. Uygulama

Üç katmanlı (giriş, gizli ve çıkış) bir şekilde tasarlanan YSA sınıflandırıcısı genel hatları ile Şekil 3.10' da gösterilmiştir. Eğitim kümesi verileri, 2. bölümde anlatılan eksik verilerin değerlendirilmesi, normalizasyon ve kodlama işlemleri sonucu ağa giriş verileri olarak girilmektedir. Giriş katmanı ağırlıkları 0 ile 1 arasında rastgele bir şekilde belirlenir. Seçilen aktivasyon fonksiyonu ile diğer katmanlardaki nöronlar tetiklenerek çıkış sinyallerini oluşturur. Bu işlem, geriye yayılım algoritması ile belirlenen çevrim sayısı süresince tekrarlanır ve her bir nöronun ağırlığı güncellenerek son çıkışlar üretilir. Bu çıkışların her biri bir sınıfa karşılık düşmekte ve niceliği de o sınıfa aidiyet oranını temsil etmektedir. Çıkışların değerleri göz önüne alınarak "one-hot encoding" tekniğine göre kodlanarak sınıflayıcının tahmin ettiği sınıf belirlenmiş olur. Geriye yayılım algoritmasına göre her bir adımda tahmin edilen sınıf ile eğitim verisine ait gerçek sınıf karşılaştırılarak hata oranı hesaplanır. Sonuç olarak YSA sınıflandırıcı, kendi ağında bulunan nöronlarının ağırlıklarını güncelleyerek öğrenme verilerini öğrenmiş olur.

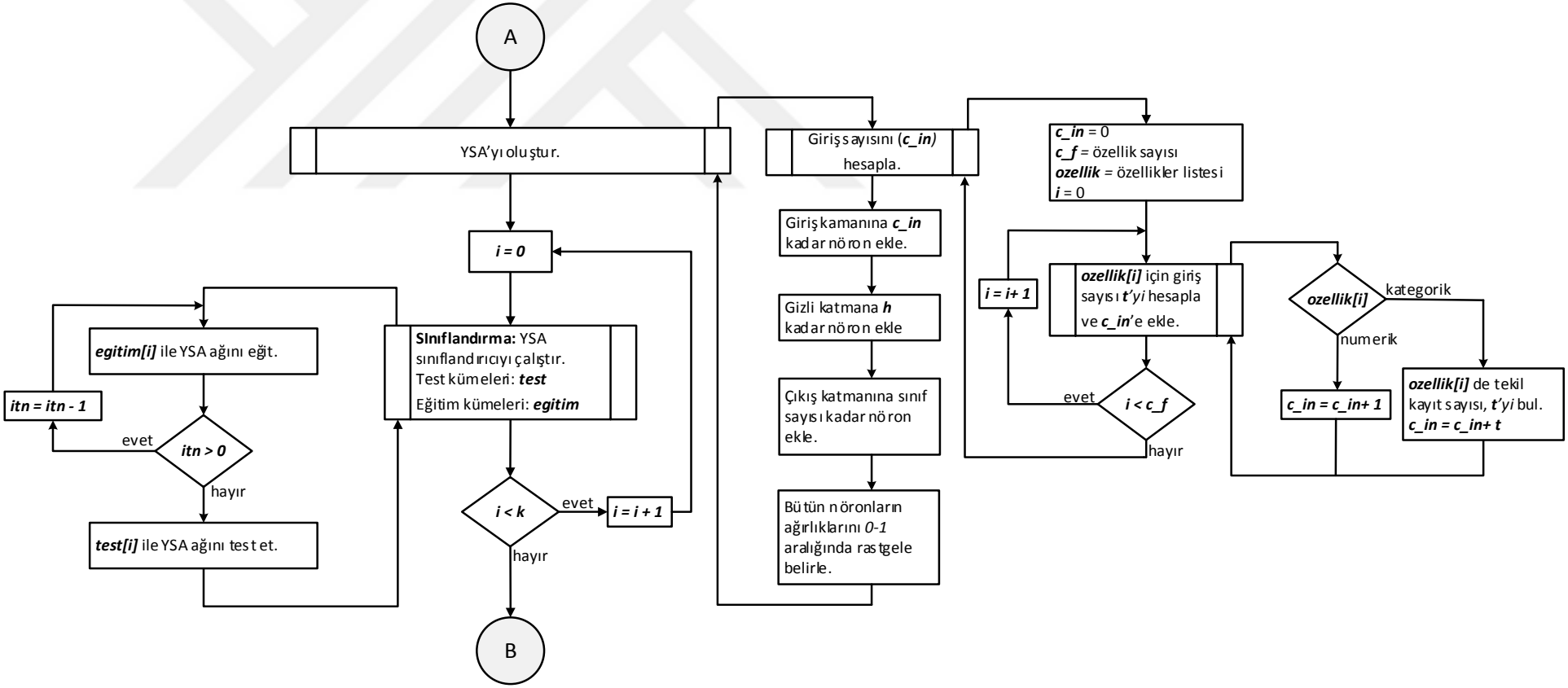


Şekil 3.10 Geriye yayılım algoritmasının çalışma şekli.

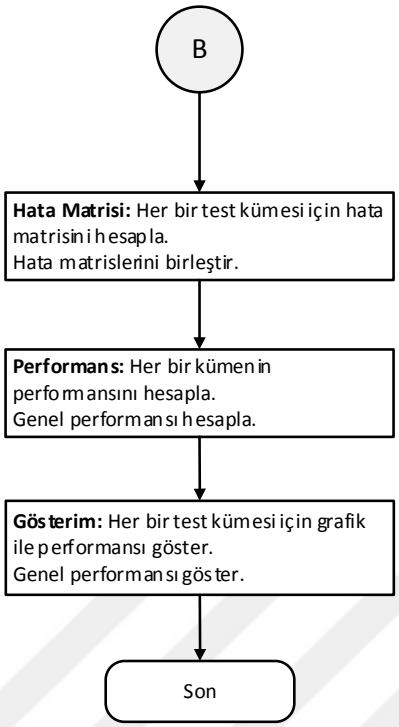
YSA algoritması kullanılarak gerçekleştirilen modelin akış diyagramı Şekil 3.11, Şekil 3.12 ve Şekil 3.13' de gösterilmiştir.



Şekil 3.11 YSA akış diyagramı 1.



Şekil 3.12 YSA akış diyagramı 2.



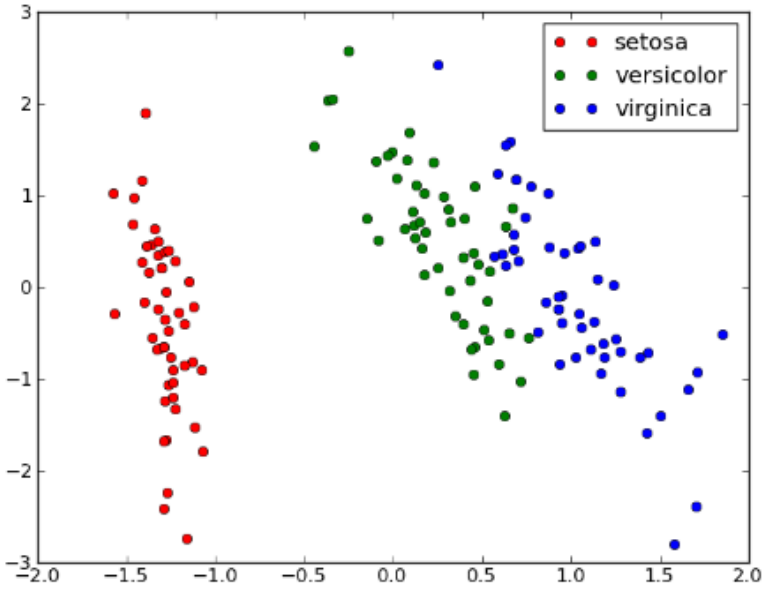
Şekil 3.13 YSA akış diyagramı 3.

3.2.4. Deneyler

YSA sınıflandırıcı, "Machine Learning and Intelligent Systems - University of California, Irvine (UCI)" adlı kaynaktan alınan "mushroom" ve "iris" veri kümeleri farklı parametreler ile test edilmiştir [44][45].

3.2.4.1. "Iris" Veri Kümesi

Örüntü tanıma literatüründe çokça kullanılan popüler bir veri kümesidir. Bu veri kümesi, her bir sınıfın bir *süsen* çiçeği türünü temsil ettiği ve her biri 50 adet örneğe sahip 3 sınıfı içerir. Bir sınıf diğer ikisinden doğrusal olarak ayrılabilir; diğer ikisi ise birbirlerinden doğrusal olarak ayrılmaz (Şekil 3.14).



Şekil 3.14 Iris veri kümesinin türlere göre dağılımı [46].

Örnek sayısı: 150 adet.

Özellik sayısı: 4 adet.

Özelliklerin niteliği: Nümerik (Sürekli) özellikler.

Özelliklerin içerikleri:

1. ***Sepallength:*** cm cinsinden uzunluk.
2. ***Sepalwidth:*** cm cinsinden uzunluk.
3. ***Petallength:*** cm cinsinden uzunluk.
4. ***Petalwidth:*** cm cinsinden uzunluk.

Eksik verilere sahip örnekler: 0 adet.

Sınıflar: Iris Setosa, Iris Versicolour, Iris Virginica.

Sınıf dağılımı: Iris Setosa (%33.3), Iris Versicolor (%33.3), Iris Virginica (%33.3)

3.2.4.2. ”Mushroom” Veri Kümesi

Bu veri kümesi, mantar türleri içinde *Agaricus* ve *Lepiota* ailesindeki 23 çeşit solungaç mantarına karşılık gelen varsayımsal örneklerin özelliklerini içermektedir. Her tür kesinlikle yenilebilir, kesinlikle zehirli veya bilinmeyen şeklinde tanımlanır. Veri kümesinin açık erişim adresinden temin edilen versiyonunda, bilinmeyen son sınıf zehirli olan sınıf ile birleştirilerek iki sınıfa (yenilebilir, zehirli) sahip bir veri kümesi elde

edilmiştir. Açık erişimde bulunan tanımlayıcı bilgilerde bu tür mantarların yenilebilirliğini belirlemek için basit bir kural olmadığı ayrıca belirtilmiştir [44].

Örnek sayısı: 8124 adet.

Özellik sayısı: 22 adet.

Özelliklerin niteliği: Kategorik özellikler.

Özelliklerin olası değerleri:

1. ***cap-shape:*** bell (b), conical (c), convex (x), flat (f), knobbed (k), sunken (s)
2. ***cap-surface:*** fibrous (f), grooves (g), scaly (y), smooth (s)
3. ***cap-color:*** brown (n), buff (b), cinnamon (c), gray (g), green (r), pink (p), purple (u), red (e), white (w), yellow (y)
4. ***bruises?:*** bruises (t), no (f)
5. ***odor:*** almond (a), anise (l), creosote (c), fishy (y), foul (f), musty (m), none (n), pungent (p), spicy (s)
6. ***gill-attachment:*** attached (a), descending (d), free (f), notched (n)
7. ***gill-spacing:*** close (c), crowded (w), distant (d)
8. ***gill-size:*** broad (b), narrow (n)
9. ***gill-color:*** black (k), brown (n), buff (b), chocolate (h), gray (g), green (r), orange (o), pink (p), purple (u), red (e), white (w), yellow (y)
10. ***stalk-shape:*** enlarging (e), tapering (t)
11. ***stalk-root:*** bulbous (b), club (c), cup (u), equal (e), rhizomorphs (z), rooted (r), missing (?)
12. ***stalk-surface-above-ring:*** fibrous (f), scaly (y), silky (k), smooth (s)
13. ***stalk-surface-below-ring:*** fibrous (f), scaly (y), silky (k), smooth (s)
14. ***stalk-color-above-ring:*** brown (n), buff (b), cinnamon (c), gray (g), orange (o), pink (p), red (e), white (w), yellow (y)
15. ***stalk-color-below-ring:*** brown (n), buff (b), cinnamon (c), gray (g), orange (o), pink (p), red (e), white (w), yellow (y)
16. ***veil-type:*** partial (p), universal (u)
17. ***veil-color:*** brown (n), orange (o), white (w), yellow (y)
18. ***ring-number:*** none (n), one (o), two (t)
19. ***ring-type:*** cobwebby (c), evanescent (e), flaring (f), large (l), none (n), pendant (p), sheathing (s), zone (z)

20. ***spore-print-color:*** black (k), brown (n), buff (b), chocolate (h), green (r), orange (o), purple (u), white (w), yellow (y)
21. ***population:*** abundant (a), clustered (c), numerous (n), scattered (s), several (v), solitary (y)
22. ***habitat:*** grasses (g), leaves (l), meadows (m), paths (p), urban (u), waste (w), woods (d)

Eksik verilere sahip örnekler: 2480 adet. Eksik veriye sahip bütün örneklerin sadece 11 numaralı, "stalk-root" özelliği bilinmemektedir. Bu değer veri kümesine "?" şeklinde girilmiştir.

Sınıflar: Yenilebilir (Edible = e), Zehirli (Poisonous = p).

Sınıf dağılımı: Edible: 4208 (51.8%), Poisonous: 3916 (48.2%).

3.2.4.3. Sonuçlar

Tablo 3.1' de gerçekleştirilen deneyler gösterilmiştir.

Tablo 3.1 YSA ile yapılan deneyler.

	Giriş parametreleri							**Sonuçlar**
No	Veri kümesi*	Örnek sayısı	İterasyon sayısı	Öğrenme sabiti	Momentum sabiti	K-fold değeri	Gizli katman nöron s.	Performans
1	Mushroom-1	8124	5000	0.3	0.2	2	3	% 100
2	Mushroom-1	8124	500	0.3	0.2	5	3	% 50.64
3	Mushroom-2	5644	5000	0.3	0.2	2	3	% 100
4	Mushroom-2	5644	5000	0.3	0.2	5	3	% 99.92
5	Mushroom-2	5644	5000	0.3	0.2	10	3	% 100
6	Iris	150	1000	0.3	0.2	2	3	% 95.33
7	Iris	150	1000	0.3	0.2	5	3	% 96.00
8	Iris	150	1000	0.3	0.2	10	3	% 95.33

*****Mushroom veri kümesi iki şekilde test edilmiştir. Eksik veriler, veri kümesinde en fazla tekrar eden değer ile güncellenerek test işlemi elde edilen 8124 örnekli Mushroom-1 üzerinden gerçekleştirilmiştir. 5644 örnekli Mushroom-2 veri kümesinde ise eksik veriler veri kümesinden çıkartılmıştır.

3.3. K-NN

K en yakın komşu algoritması (k-nearest neighbor), uygulaması basit ve esnek bir sınıflandırma yöntemidir. K-NN sınıflandırması, genellikle öğrenme verilerinin nümerik olduğu durumlarda kullanılır. K-NN, veri kümesini bütünüyle hafızaya alıp işlem yaptığından dolayı parametrik olmayan bellek tabanlı bir MÖ yaklaşımıdır. Veri setinden ayırıcı bir fonksiyon ile öğrenme yapmadığı için literatürde bu tarz öğrenme sekline "lazy

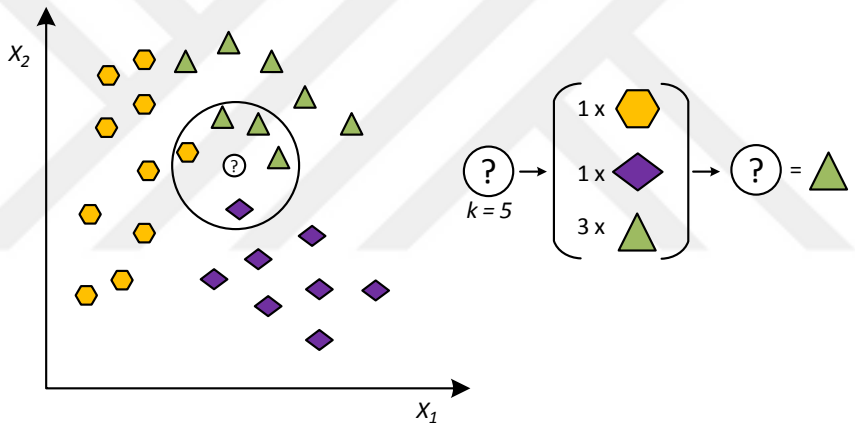
learning" de denir. Kategorik özelliklerin olduğu durumlarda bu veriler, uygun şekilde kodlanıp nümerik hale getirilerek sınıflandırma işlemi yapılabilir [30].

3.3.1. Algoritma

eğitim veri setinde her bir örnek için uygula.

- *eğitim örneği ile test verisi arasındaki uzaklığı hesapla.*
- *uzaklıkları küçükten büyüğe sırala.*
- *en küçük k kadar örnek içerisinde baskın olan sınıfı belirle.*
- *baskın sınıfı test örneğinin sınıfı olarak döndür.*

K-NN sınıflayıcının kabaca çalışma şekli Şekil 3.15' de gösterilmiştir.



Şekil 3.15 K-NN ile sınıflandırma örneği [47].

K-NN algoritması ile yeni eğitim verileri hızlı bir şekilde eklenip test için kullanılabilir. Veri kümesi, az özellik sayısına sahip olduğu durumlarda daha hızlı çalışır. Bununla birlikte eğitim verilerinin artması hesaplama maliyetini artırarak kaynakların yetersiz kalmasına sebep olabilmektedir. Bu açıdan algoritma "kd-tree" [48] gibi bir veri yapısı üzerinde gerçekleştirilirse işlem karmaşıklığı *O(logn)* gibi bir logaritmik fonksiyona dönüşerek azalabilir.

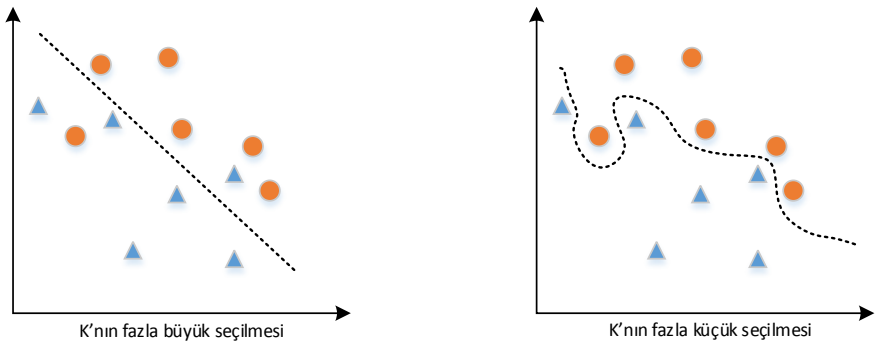
K-NN sınıflayıcı, seçilen uzaklık yöntemine göre öğrenme verilerinden elde ettiği uzaklık bilgilerinden en yakın *k* kadar örneği "majority voting" [49] yöntemine göre değerlendirerek test verisinin sınıfını belirler. Kullanılabilecek uzaklık ölçme yöntemleri ise denklem (3.24), denklem (3.25) ve denklem (3.26)' da gösterilmiştir [50].

$$Euclidean\ Uzaklık(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.24}$$

$$Manhattan\ Uzaklık(x,y) = \sum_{i=1}^{n}|x_i - y_i| \tag{3.25}$$

$$Minkowski\ Uzaklık(x,y) = \left(\sum_{i=1}^{n}(x_i - y_i)^p\right)^{\frac{1}{p}} \tag{3.26}$$

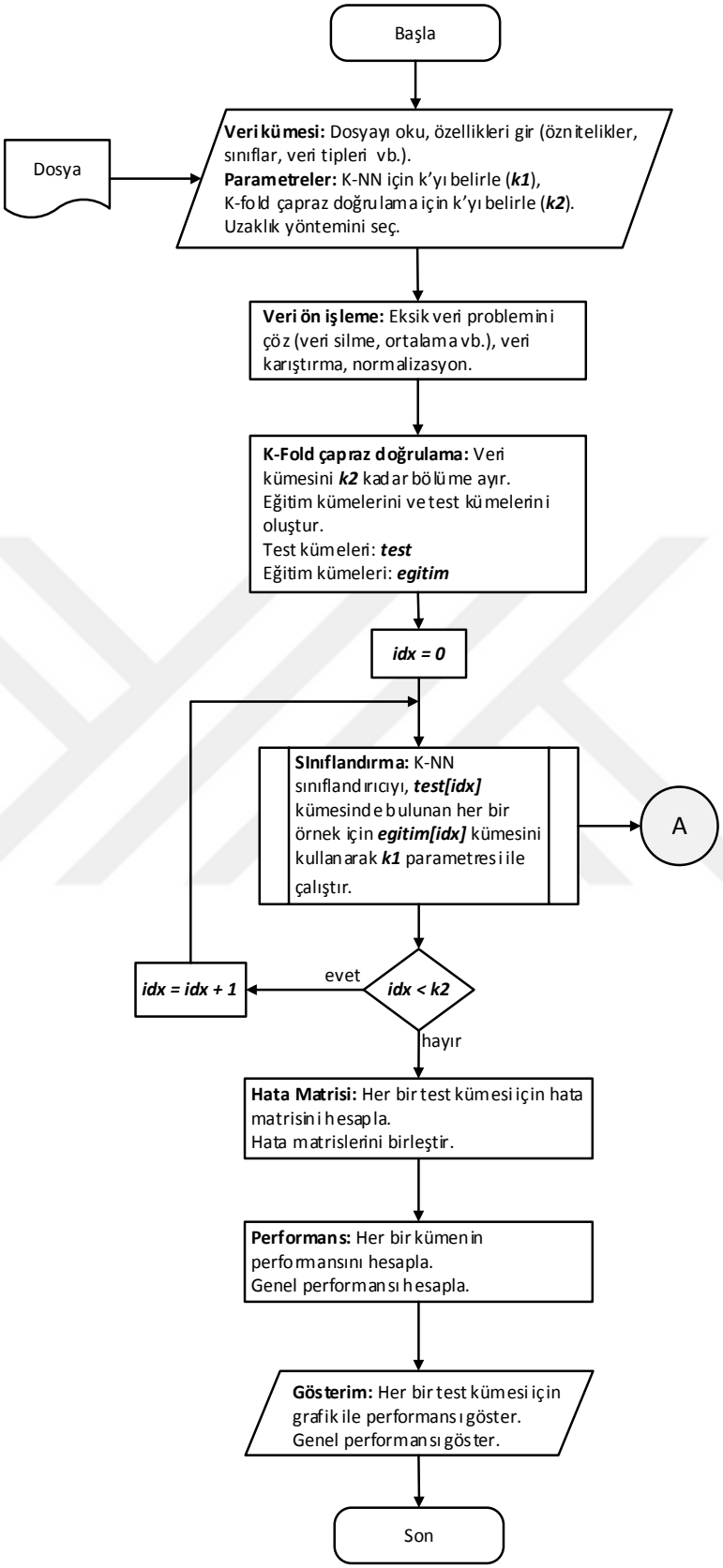
K-NN algoritmasında k (komşu noktalar) parametresinin seçimi, genelleştirme problemi açısından önemli bir yer tutmaktadır. Eğitim aşamasında aşırı uyum (overfitting) veya yetersiz uyum (underfitting) arasındaki denge problemi olarak bilinen "bias variance tradeoff" [51], k' nın değerine göre değişmektedir. k' nın büyük seçilmesi gürültülü verilerin etkisini azaltır. Öte yandan bu durum bazı küçük boyutlu ve önemli olabilecek örüntülerin de gözden kaçmasına sebep olabilmektedir (Şekil 3.16).



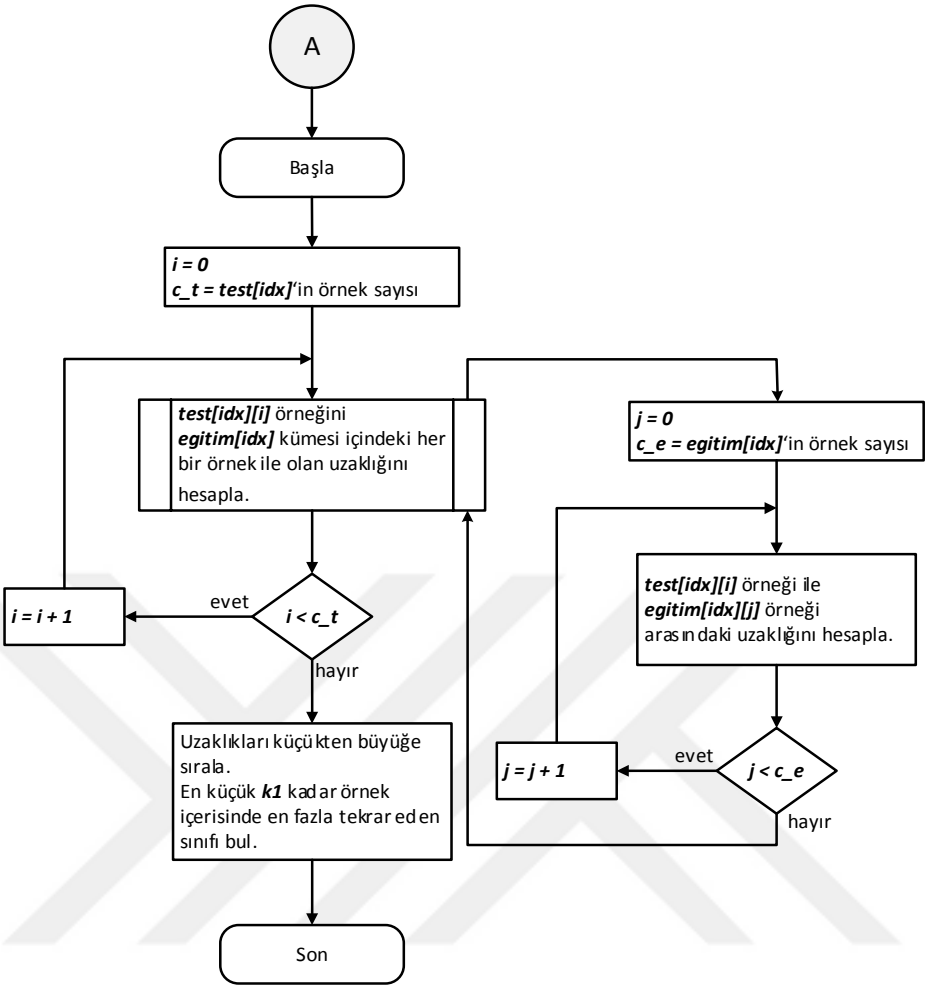
Şekil 3.16 K' nın büyüklüğünün öğrenmeye etkisi [24].

3.3.2. Uygulama

K-NN modelinin uygulanması Şekil 3.17 ve Şekil 3.18' deki akış diyagramlarında gösterilmiştir. K-NN sınıflandırıcı, "iris" veri kümesi ile test edilmiştir. Bölüm 3.2.4.1 de veri kümesi ile ilgili bilgiler verilmiştir.



Şekil 3.17 K-NN akış diyagramı 1.



Şekil 3.18 K-NN akış diyagramı 2.

3.3.3. Deneyler

Tablo 3.2' de Iris veri kümesi ile yapılan deneylerin sonuçları gösterilmiştir.

Tablo 3.2 K-NN deney sonuçları.

No	K	Uzaklık ölçme yöntemi	K-fold değeri	Performans
1	3	Euclidean	5	% 94.66
2	5	Euclidean	5	% 96.00
3	7	Euclidean	5	% 96.00
4	3	Manhattan	5	% 94.66
5	5	Manhattan	5	% 94.66
6	7	Manhattan	5	% 95.33
7	3	Minkowski	5	% 94.66
8	5	Minkowski	5	% 95.33
9	7	Minkowski	5	% 95.33

3.4. Naive Bayes Sınıflandırma Yöntemi

İstatistik biliminde birçok alanda kullanılan Naive Bayes (NB), adını 18. yy. matematikçilerinden Thomas Bayes' den alır. Thomas Bayes, bir durumun olasılığını ve bu duruma eklenecek yeni koşulların olasılığı nasıl değiştireceğine dair temel prensipleri tanımlamıştır. Bu prensipler genel olarak "Bayesian" yöntemleri olarak bilinir. Bu yöntemler esas alınarak sınıflandırma problemlerinin çözümü için geliştirilen algoritmalar, "Bayesian" veya Bayes sınıflandırıcısı olarak geçmektedir.

3.4.1. Bayes Teorisi

Bir olayın gerçekleşme olasılığın başka bağımsız olayların gerçekleşme olasılığına bağlı olması durumu üzerinden tanımlanan bir teoridir. Başka bir ifade ile bir olayın başka bir olayın koşulu (koşullu olasılık) ile gerçekleşmesi durumudur. Bir A olayının gerçekleşme olasılığı, $P(A)$ şeklinde tanımlanır. Bu ifade, Bayes Teorisinde başka olasılıklarda devreye girdiğinden "önsel olasılık" olarak ta ifade edilir.

$$P(A\backslash B) = \frac{P(B\backslash A)P(A)}{P(B)} \tag{3.27}$$

$P(A)$, A' nın önsel olasılığını,
$P(B)$, B' nin önsel olasılığını,
$P(A\backslash B)$, A' nın B koşullu sonrasal olasılığını,
$P(B\backslash A)$, B' nin A koşullu sonrasal olasılığını göstermektedir.

3.4.2. Bayes Sınıflandırıcısı

Bayesian Sınıflandırıcısı, eğitim verileri üzerinde özellik vektörlerini baz alarak hedef sınıflarının olasılık dağılımını hesaplar. Yeni gelen test verisinin sınıfını gözlemlenmiş olasılık değerlerine en yakın değeri gözeterek tahmin etmeye çalışır. Veri kümesi içerisinde x örneği ve n kadar özelliği olduğu düşünülürse bu örnek, $x = (x_1, \dots, x_n)$ şeklinde bir vektör olarak ifade edilebilir. C, sınıf kümesi olmak üzere x örneğinin C_k

sınıfında olma olasılığı $p(C_k|x_1, \dots, x_n)$ şekilde ifade edilir. Bu denklem Bayes Teoremine uygun şekilde yazılırsa denklem (3.28) elde edilir.

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{P(x)} \tag{3.28}$$

Örnek verinin, C_k sınıfı için çözümü:

$$p(C_k, x_1, \dots, x_n) = p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{3.29}$$

Denklem (3.29)' daki işlem her bir sınıfa uygulandıktan sonra en yüksek olasılıklı sınıf, örneğin sınıfı olarak tahmin edilmektedir (denklem (3.30)).

$$\hat{y} = \underset{k\epsilon(1,\dots,K)}{argmax}\ p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{3.30}$$

Veri kümesinin sürekli değere sahip özelliği olması durumunda kategorik özelliklerde olduğu gibi olasılık hesabı yapılmaz. Bunun yerine örneklerin bu özellikte normal dağılım gösterdiği varsayılarak her bir örneğe normal (Gauss) dağılım fonksiyonu (denklem (3.31)) uygulanabilir. v, örneğin sürekli özellikteki değerini, μ, veri kümesinin sürekli özellikteki ortalamasını, σ, bu özellikteki standart sapmayı göstermektedir.

$$p(x = v|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}} \tag{3.31}$$

3.4.3. Sıfır Olasılık Problemi

Denklem (3.29)' da herhangi bir örneğin bir özelliği ile ilgili olasılık hesabında sonuç sıfır çıkabilir. Bu durumda o örneğin diğer özelliklerine bağlı olasılıkları etkisiz kalacaktır. Her bir olasılığa denklem (3.32)' de gösterildiği gibi küçük bir değer eklenerek bu problem çözülebilmektedir [52]. 0 ile 1 arasında seçilen bir *k* değeri pay ve paydaya eklenir. Paya

eklenen *k* değeri bir *p* katsayısı ile çarpılır. *p* katsayısı, özelliğin olasılık hesabı yapılmak istenen değerinin özellik içerisinde olası bütün tekil kayıtlarına oranıdır.

$$\frac{n + (k)(p)}{d + k} \tag{3.32}$$

3.4.4. Algoritma

Sınıfı bilinmeyen bir örneğin NB ile sınıflandırılması kabaca şu şekildedir:

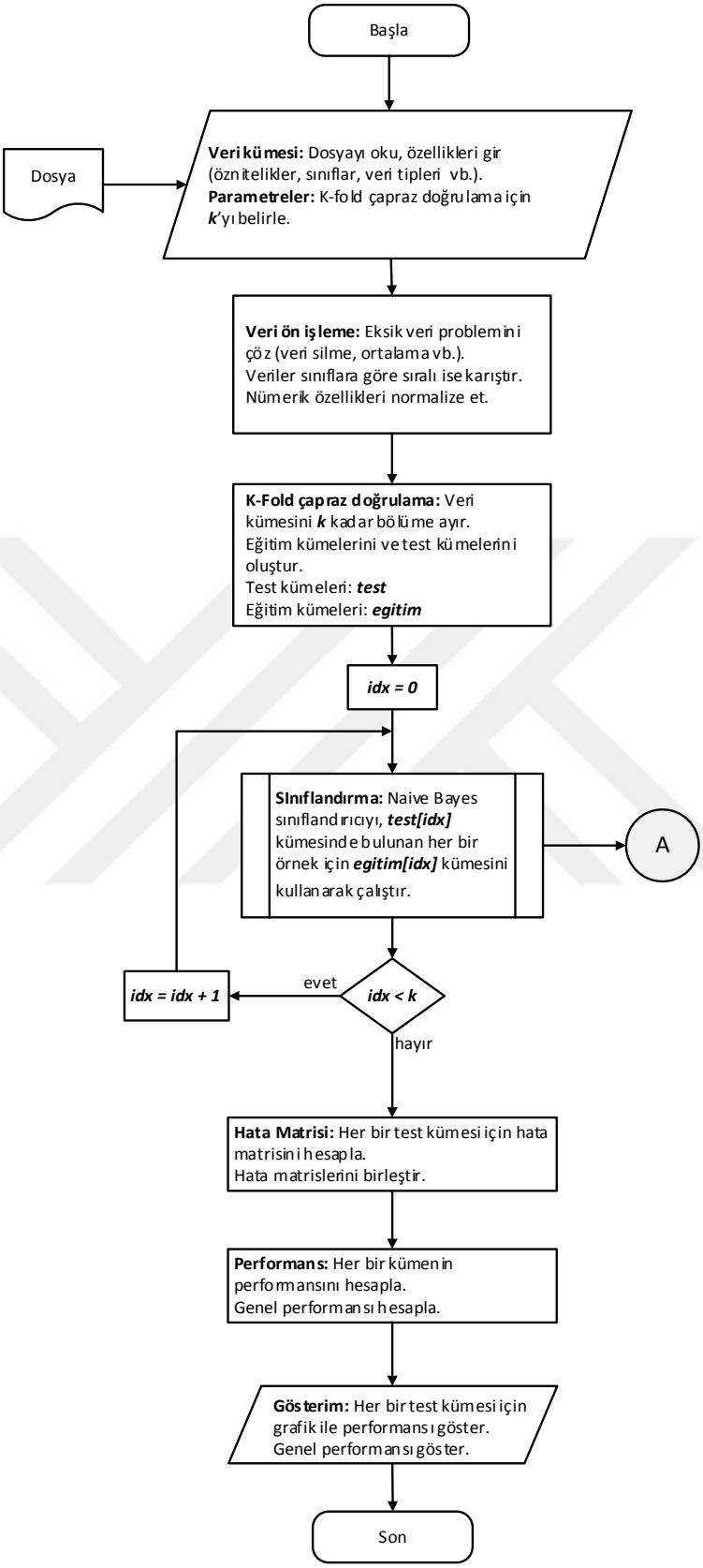
veri kümesinden her bir sınıf için önsel olasılıkları $(p(C_k))$ *hesapla.*
her bir sınıf (C_k) *için hesapla.*

- *sınıflandırılmak istenen örneğin her bir özelliği* (x_i) *için hesapla.*
  - C_k *koşullu olarak özelliğin sonrasal olasılığını* $(p(x_i \backslash C_k)$ *hesapla.*
- *bulunan bütün sonrasal olasılıkların çarpımını* $(\prod_{i=1}^{n} p(x_i|C_k))$ *bul.*
- *bu çarpımı sınıfın önsel olasılığı ile çarp* $(p(C_k) \prod_{i=1}^{n} p(x_i|C_k))$.

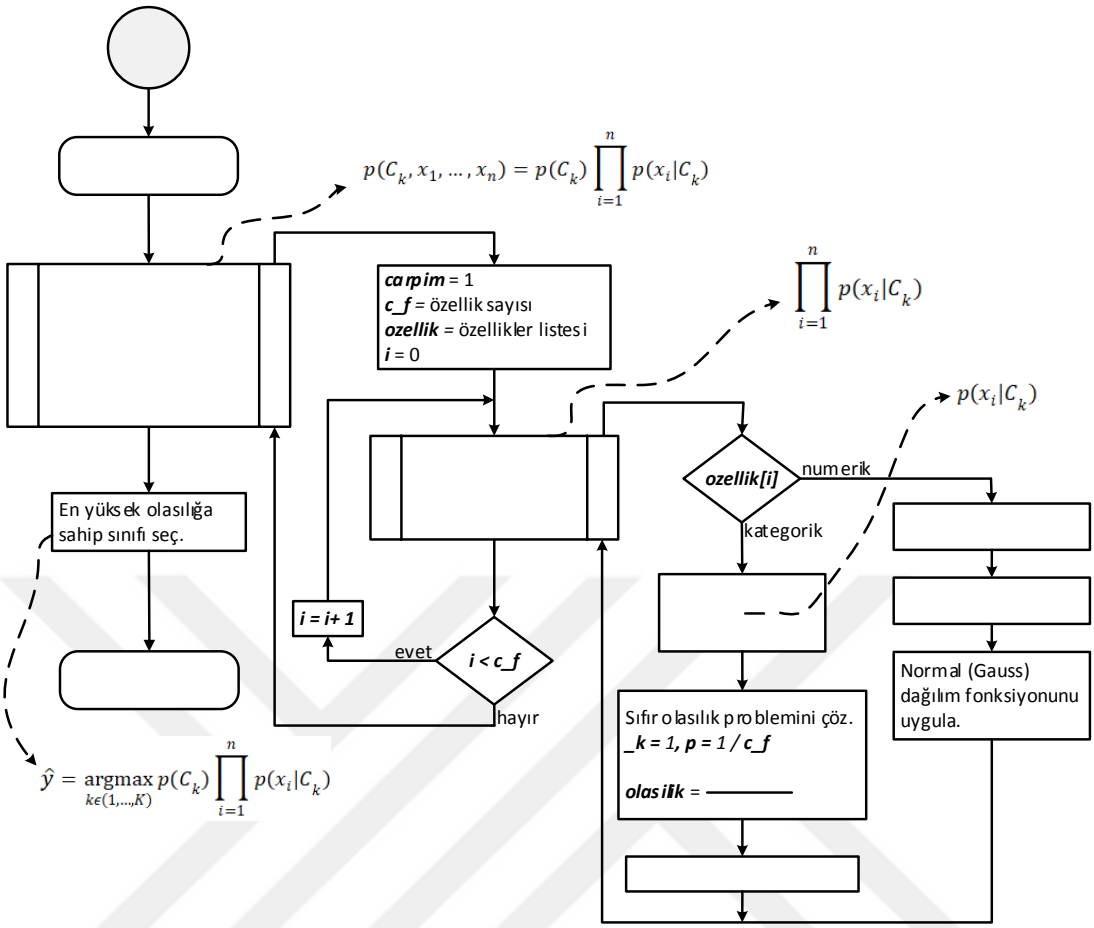
her bir sınıf için bulunan çarpımlardan en yüksek olasılık değerine sahip sınıfı seç $(\underset{k\epsilon(1,\ldots,K)}{\text{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$.

3.4.5. Uygulama

NB modelinin uygulanması Şekil 3.19 ve Şekil 3.20' deki akış diyagramlarında gösterilmiştir. NB sınıflandırıcı, "iris" ve "mushroom" veri kümeleri ile test edilmiştir. Bölüm 3.2.4' de veri kümeleri ile ilgili bilgiler verilmiştir.



Şekil 3.19 Naive Bayes akış diyagramı 1.



Şekil 3.20 Naive Bayes akış diyagramı 2.

3.4.6. Deneyler

Tablo 3.3' de gerçekleştirilen deneylerin sonuçları gösterilmiştir.

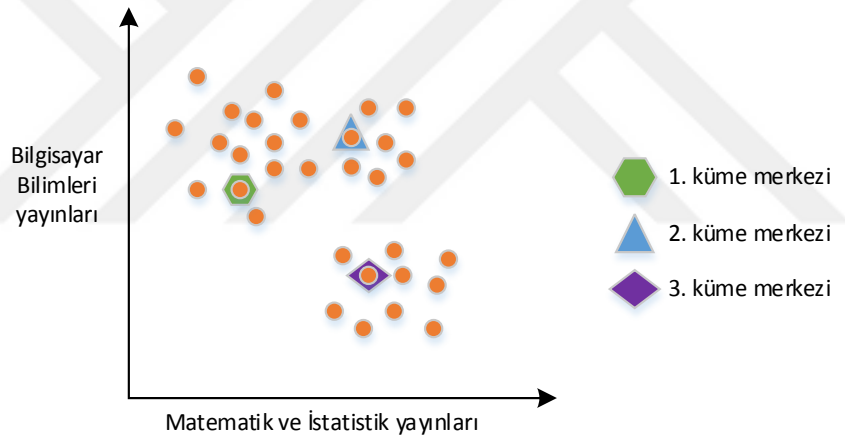
Tablo 3.3 Naive Bayes ile yapılan deneyler.

No	Veri Kümesi*	Örnek sayısı	K-fold değeri	Performans
1	Iris	150	2	% 88.00
2	Iris	150	5	% 91.33
3	Iris	150	10	% 92.66
4	Mushroom-1	5644	2	% 98.99
5	Mushroom-1	5644	20	% 99.45
6	Mushroom-1	5644	4000	% 99.46
7	Mushroom-2	8124	2	% 97.32
8	Mushroom-2	8124	20	% 98.11

*Mushroom-1' de eksik veriler çıkartıldı. Mushroom-2' de eksik veriler veri kümesinde en fazla tekrar eden değer ile güncellendi.

3.5. K-Ortalamalar (K-Means)

Veri seti üzerinde bakılmak istenen bilgi tam olarak bilinmiyorsa, başka bir ifadeyle hedef sınıflar mevcut değilse kümeleme işlemi yapılarak örnekler benzerliklerine göre gruplandırılmış olur. Kümeleme işlemi için popüler yöntemlerden biri K-Ortalamalar (KO) algoritmasıdır. KO yöntemi, kabaca veri seti içerisindeki farklı küme merkezlerini ve bu kümelerin sınırlarını bulmaya çalışır. Küme analizi ile benzer özelliklere sahip örnekler aynı sınıfa dâhil edilir. Bu işlem bir benzerlik fonksiyonuna göre gerçekleştirilir. Kullanılacak benzerlik fonksiyonu verilerin dağılımına göre en uygunu seçilir. Kolay uygulanabilir bununla birlikte yerel minimum noktalarına yakınsama oranı yüksektir. Büyük veri setlerinde yavaş çalışır. Çoğunlukla sayısal veri setlerinde kullanılır [53]. Şekil 3.21' de *k* değeri 3 alınarak kümeleme yapılmış bir örnek gösterilmiştir.



Şekil 3.21 Kümeleme örneği [24].

KO, *k* kadar kümeyi veri seti üzerinde arayıp merkezlerini belirler. *k* değeri kullanıcı tarafından tanımlanır. Her bir küme, başlangıçta kabaca bir noktadan meydana gelir. Bu noktalar kümelerin merkezleri (centroids) kabul edilir ve başlangıçta rastgele bir şekilde seçilirler. Daha sonra veri setindeki her bir örnek için merkezlere uzaklıklarına bakılarak küme aidiyetleri belirlenir. Her bir küme için yeni merkezler, küme elemanlarının ortalaması bulunarak belirlenir. Her bir örnek bir kümeye dâhil edilene kadar bu işlemlere devam edilir. Son olarak küme merkezlerinin pozisyonu değişmeyene veya belirli bir eşik değerden küçük olana kadar devam edilerek sonuçta K kadar kümeleme işlemi gerçekleştirilmiş olur.

KO algoritması, kümeleme işlemini denklem (3.33)' de gösterildiği gibi yapar [40].

$$\sum_{i=1}^{K}\sum_{j=1}^{|C_i|} Uzaklık\left(x_j, merkez(i)\right) \tag{3.33}$$

C_i, i. kümede bulunan eleman sayısını göstermektedir. *Uzaklık* fonksiyonu, x_j örneği ile i. küme merkezi arasındaki uzaklığı bulmaktadır. Bu fonksiyon için denklem (3.24), denklem (3.25) ve denklem (3.26)' da yazılan uzaklık yöntemlerinden biri seçilerek kümeleme işlemi gerçekleştirilebilir.

3.5.1. Algoritma

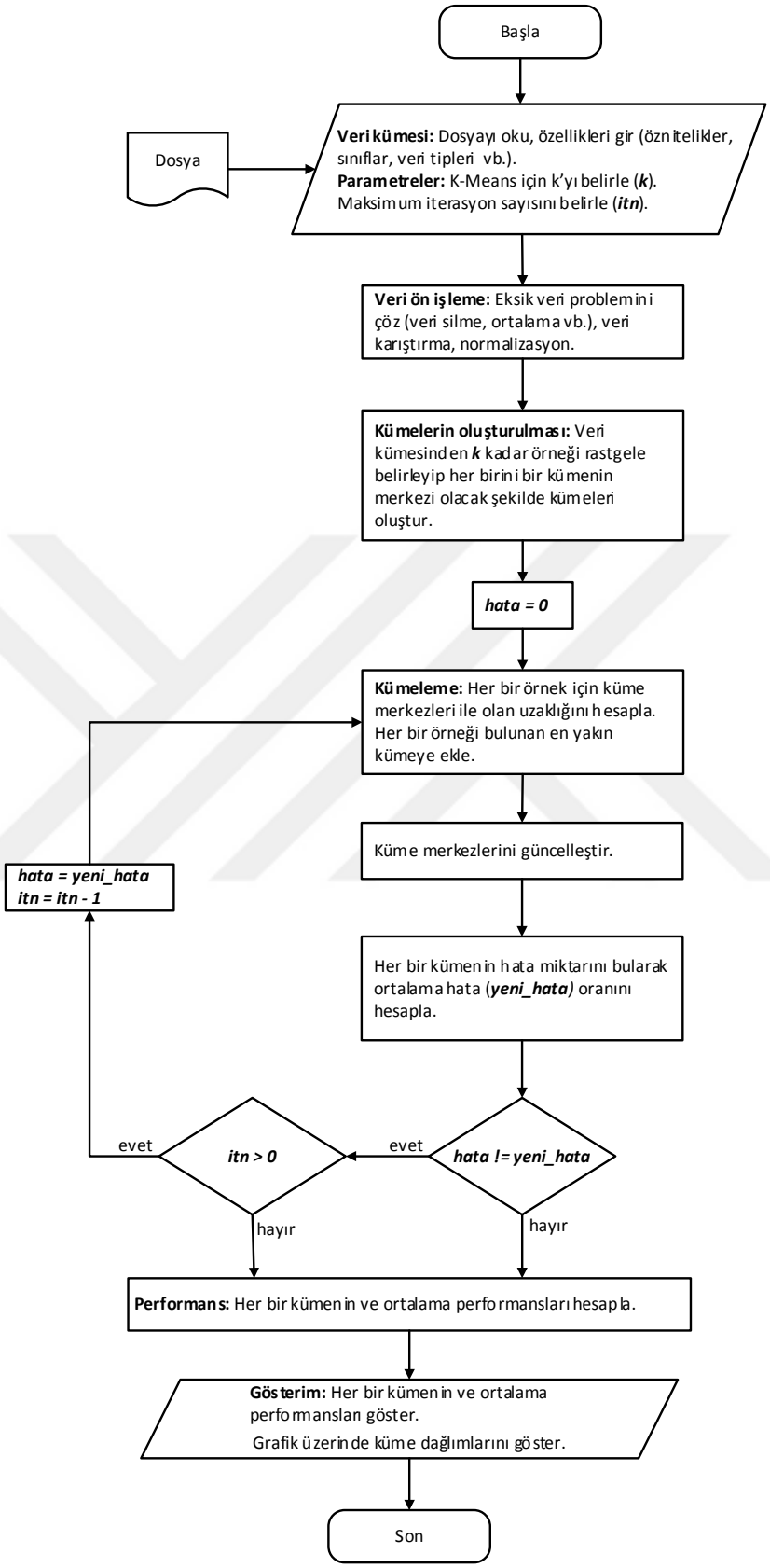
k kadar küme merkezini rastgele oluştur.

ortalama toplam hata oranı değişmeyene kadar veya maksimum iterasyon sayısı aşılana kadar uygula.

- *her bir örnek için küme merkezleri ile olan uzaklığını hesapla ve bulunan en yakın kümeye ekle.*
- *her bir küme için uygula.*
  - *örneklerin ortalamasını hesapla.*
  - *küme merkezini ortalamaya göre güncelle.*
  - *kümenin toplam hata oranını hesapla.*

3.5.2. Uygulama

KO algoritmasının uygulanması Şekil 3.22' deki akış diyagramında gösterilmiştir. Bu uygulama, "iris" veri kümesi ile test edilmiştir. Bölüm 3.2.4.1 de veri kümesi ile ilgili bilgiler verilmiştir.



Şekil 3.22 K-Means akış diyagramı.

3.5.1. Deneyler

Iris veri kümesi kullanılarak yapılan bazı testlerin sonuçları Tablo 3.4' de gösterilmiştir. Bütün sonuçlar maksimum iterasyon sayısına ulaşmadan elde edilmiştir.

Tablo 3.4 K-means deney sonuçları.

Giriş parametreleri				**Sonuçlar**		
No	K	İterasyon sayısı	Uzaklık ölçüm yöntemi	Sonlanan iterasyon numarası	Küme dağılımları*****	Prf.******
1	3	120	Euclidean	4	Küme 1: 22 örnek (*a*: 18, *b*: 4, *c*: 0) ~15% Küme 2: 96 örnek (*a*: 0, *b*: 46, *c*: 50) ~64% Küme 3: 22 örnek (*a*: 32, *b*: 0, *c*: 0) ~21%	% 77.97
2	3	120	Euclidean	7	Küme 1: 50 örnek (*a*: 50, *b*: 0, *c*: 0) ~33% Küme 2: 61 örnek (*a*: 0, *b*: 47, *c*: 14) ~41% Küme 3: 39 örnek (*a*: 0, *b*: 3, *c*: 36) ~26%	% 89.79
3	3	120	Manhattan	8	Küme 1: 61 örnek (*a*: 0, *b*: 46, *c*: 15) ~41% Küme 2: 50 örnek (*a*: 50, *b*: 0, *c*: 0) ~33% Küme 3: 39 örnek (*a*: 0, *b*: 4, *c*: 35) ~26%	% 88.38
4	3	120	Manhattan	7	Küme 1: 62 örnek (*a*: 0, *b*: 47, *c*: 15) ~41% Küme 2: 38 örnek (*a*: 0, *b*: 3, *c*: 35) ~25% Küme 3: 50 örnek (*a*: 50, *b*: 0, *c*: 0) ~33%	% 89.31
5	3	120	Minkowski	4	Küme 1: 50 örnek (*a*: 50, *b*: 0, *c*: 0) ~33% Küme 2: 51 örnek (*a*: 0, *b*: 40, *c*: 11) ~34% Küme 3: 49 örnek (*a*: 0, *b*: 10, *c*: 39) ~33%	% 86.01
6	3	120	Minkowski	11	Küme 1: 56 örnek (*a*: 0, *b*: 10, *c*: 46) ~37% Küme 2: 50 örnek (*a*: 50, *b*: 0, *c*: 0) ~33% Küme 3: 44 örnek (*a*: 0, *b*: 40, *c*: 4) ~29%	% 91.02

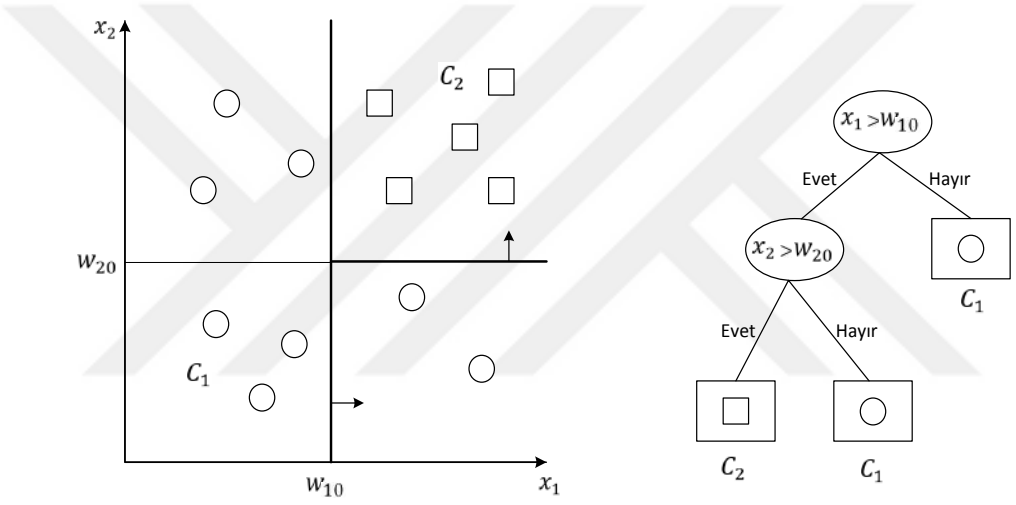
***** *a*: iris-setosa, *b*:iris-versicolor, *c*:iris-virginica.
****** Her bir kümenin içeriğindeki en fazla sınıf o kümenin sınıfı kabul edilerek performans hesabı yapılmıştır.

3.6. Karar Ağacı Sınıflandırması

Karar ağacı (KA), böl ve fethet yaklaşımı uygulanarak oluşturulmuş hiyerarşik bir veri yapısıdır. Parametrik olmayan bir metot olarak hem sınıflandırma hem de regresyon için kullanılan güçlü bir yöntemdir [54]. MÖ açısından baktığımızda KA öğrenmesi, bir karar ağacı ile temsil edilebilen öğrenilmiş bir fonksiyonu ayrık değerli bir fonksiyona yaklaştırmaya çalışan tümevarımsal bir yöntemdir. Kredi risk değerlendirmesi, tıbbi teşhis algoritmalarında KA çokça kullanılmaktadır [20].

KA yöntemiyle veriler, küçük parçalara ayrılırken bir yandan da karar ağacı ilişkileri kurulur. Verilerin bölünmesi açısından son noktaya gelindiğinde düğümlerden ve yapraklardan oluşan bir ağaç yapısı elde edilir.

Basit bir veri kümesi öğrenilerek oluşturulmuş KA, Şekil 3.23' de gösterilmiştir. Şekildeki elipsler düğümleri, dikdörtgenler ise yaprakları temsil etmektedir. İlk düğüm kök olmak üzere diğer düğümlerle beraber veri setine ait özellikleri temsil etmektedir. Düğümlerden yapılan dallanmalar ise veri setindeki değerlerdir. KA' nın oluşturulduğu veri seti eğitim kümesini oluşturur. Test edilecek bir veri örneği kök düğümden başlanılarak koşullara bağlı olarak bir yaprağa ulaşıncaya kadar değerlendirilir ve bir sınıf etiketi ile etiketlenir.



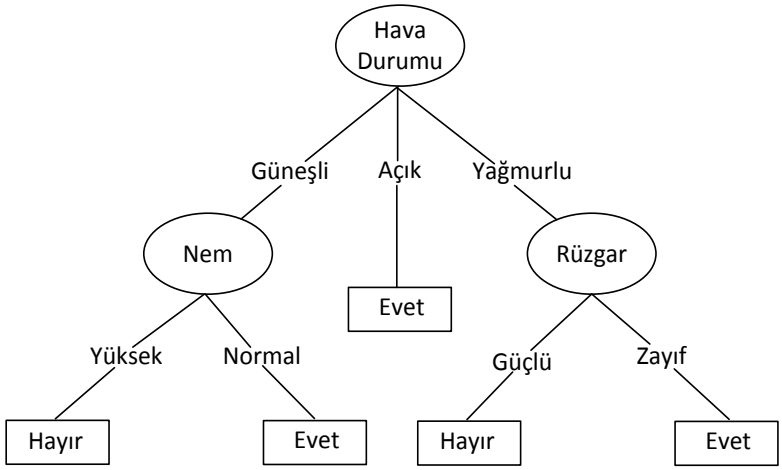
Şekil 3.23 Örnek karar ağacı [54].

Şekil 3.23' de gösterilen örnek, sürekli (nümerik) değerleri olan özelliklere sahip bir örnektir. İstatiksel alanda çokça örneği olan bu tür problemler Quinlan' nın C4.5 [55] algoritması kullanılarak çözülebilir. C4.5 algoritmasından önce geliştirilen ve bu algoritmanın temellerini oluşturan, kategorik veriler üzerinde başarılı bir şekilde uygulanabilen diğer bir algoritma ise ID3' tür [56]. Ayrıca Quinlan tarafından C4.5 algoritması geliştirilerek oluşturulan C5.0 [57] algoritması da kullanılmaktadır. Bunlar gibi yaygın olarak kullanılan diğer bir algoritma ise CART [58] adlı algoritmadır. Bu algoritmaların karşılaştırılması ile ilgili S. Singh ve meslektaşı bir inceleme [59] yapmışlardır. Tablo 3.1' de bu inceleme görülebilir.

Tablo 3.5 Karar ağacı algoritmalarının özellikleri [59].

	Karakteristik					
Algoritma		**Bölme Kriteri**	**Özellik Tipi**	**Eksik Veriler**	**Budama Tekniği**	**Aykırı Değer Tespiti**
	ID3	Bilgi kazancı	Kategorik veriler	Eksik verileri işleyemez.	Budama yapılamaz.	Aykırı değerlere karşı hassastır.
	CART	Çekme (Towing) kriteri	Kategorik ve numerik veriler	Eksik verileri değerlendirebilir.	Maliyet karmaşıklığı yaklaşımı.	Aykırı değerlerle başa çıkabilir.
	C4.5	Kazanç oranı	Kategorik ve numerik veriler	Eksik verileri değerlendirebilir.	Hata tabanlı bir budama yaklaşımı.	Aykırı değerlere karşı hassastır.

KA algoritmaları öğrenme aşamasında karar ağacını çıkartırken veri kümesinin hangi özelliğinin kök düğüm olacağına karar vermek zorundadır. Örneğin ID3 algoritması bu karara istatiksel bir analiz yaparak ulaşır. Yapılan analiz ile elde edilen en iyi özellik kök düğümü oluşturur. Kök düğümün özelliği dışında kalan diğer özellikler kök düğümden koşullu bir şekilde dallanır. Dallanma sayısı, dallanma yapılan kök özelliğinin sahip olduğu kategorik değer sayısı kadardır. Kök düğüme bağlantı yapılan her bir düğüm için aynı işlem tekrar eder. Quinlan' ın ID3' ü tanıtırken verdiği örnek, Şekil 3.24' de verilmiştir.



Şekil 3.24 Kategorik değerlere sahip karar ağacı [56].

Quinlan' nın bu örneğinde tenis oynamak için dışarı çıkacak birinin hava durumu koşullarına bağlı olarak dışarı çıkıp çıkmamasına karar verecek bir sistemden bahseder [56].

ID3 algoritması ile karar ağacı oluşturulmasına dair yapılacak seçimler, maliyet açısından en büyük bilgi kazancının elde edileceği özellikleri seçerek olmalıdır. Bu açıdan istatiksel bir kazanç hesabı gerçekleştirilir. Bu hesap veri kümesinde bulunan, sınıflandırılmak istenen hedef özelliğe göre yapılmalıdır. Kazancı hesaplayabilmek için öncelikle entropi hesabı yapılmalıdır.

3.6.1. Entropi

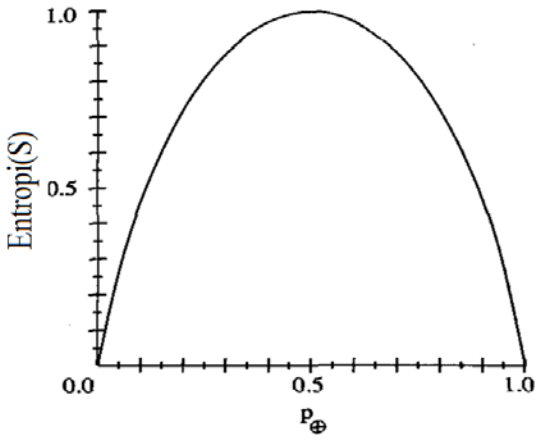
Düzensizlik anlamına gelir. Eğitim kümesindeki bilgilerin hedef özelliğe göre dağılımlarındaki düzensizlik oranını ifade eder. Şekil 3.25' de entropi değerinin bir özelliğine göre dağılımı verilmiştir.

$$Entropi(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i \tag{3.34}$$

Denklem (3.34)' de S, eğitim örneklerini;

c, hedef özelliğine ait kategorik değer sayısını;

p_i, eğitim örneklerindeki hedef özelliğinin i. kategorik değerinin oranını temsil eder.



Şekil 3.25 Entropi fonksiyonu [20].

Şekil 3.25' e göre bütün S değerleri aynı sınıfa ($p_{\oplus}$) ait ise entropi değeri 0 çıkacaktır. Diğer sınıf etiketi ($p_{\ominus}$) açısından düşündüğümüz zaman da aynı şekilde entropi değeri 0 olacaktır. Başka şekilde ifade edilirse, düzensizlik olmadığı için düzensizlik fonksiyonu da 0 olacaktır. Eğitim örnekleri içerisinde her iki sınıftan eşit miktarda bulunduğu durumda ise entropi değeri 1 olacaktır. Diğer durumlarda ise sıfır ve bir arasında değer almaktadır.

3.6.2. Bilgi Kazancı

Hedef özellik dikkate alınarak seçilecek en uygun özellik için kazanç fonksiyonu şu şekilde olacaktır:

$$Kazanç(S, A) = Entropi(S) - \sum_{v \in Degerler(A)} \frac{|S_v|}{|S|} Entropi(S_v) \tag{3.35}$$

Denklem (3.35)' de,

A, kazancın hesaplanacağı özelliği;

v, A özelliğine ait kategorik değerlerini;

$|S_v|$, eğitim örnekleri içerisindeki A özelliğinde v değerleri olan toplam örnek sayısını;

$|S|$, eğitim örneklerinin toplam sayısını;

$\frac{|S_v|}{|S|}$, eğitim örnekleri içerisinde A özelliğinde v değerleri olan örneklerin sayısının toplam örnek sayısına oranını temsil etmektedir.

3.6.3. ID3 Algoritması

ornekler, eğitim örneklerini temsil etmektedir. *hedef_ozellik*, algoritma tarafından test verilerinin sınıflandırılmasında kullanılacak kategorik değerleri barındıran özelliktir. *ozellikler*, eğitim örneklerinin sahip olduğu *hedef_özellik* dışındaki özellikler koleksiyonudur. Dönüş değeri ise test verilerini doğru bir şekilde sınıflayabilecek, eğitim örneklerini öğrenmiş bir KA' dır.

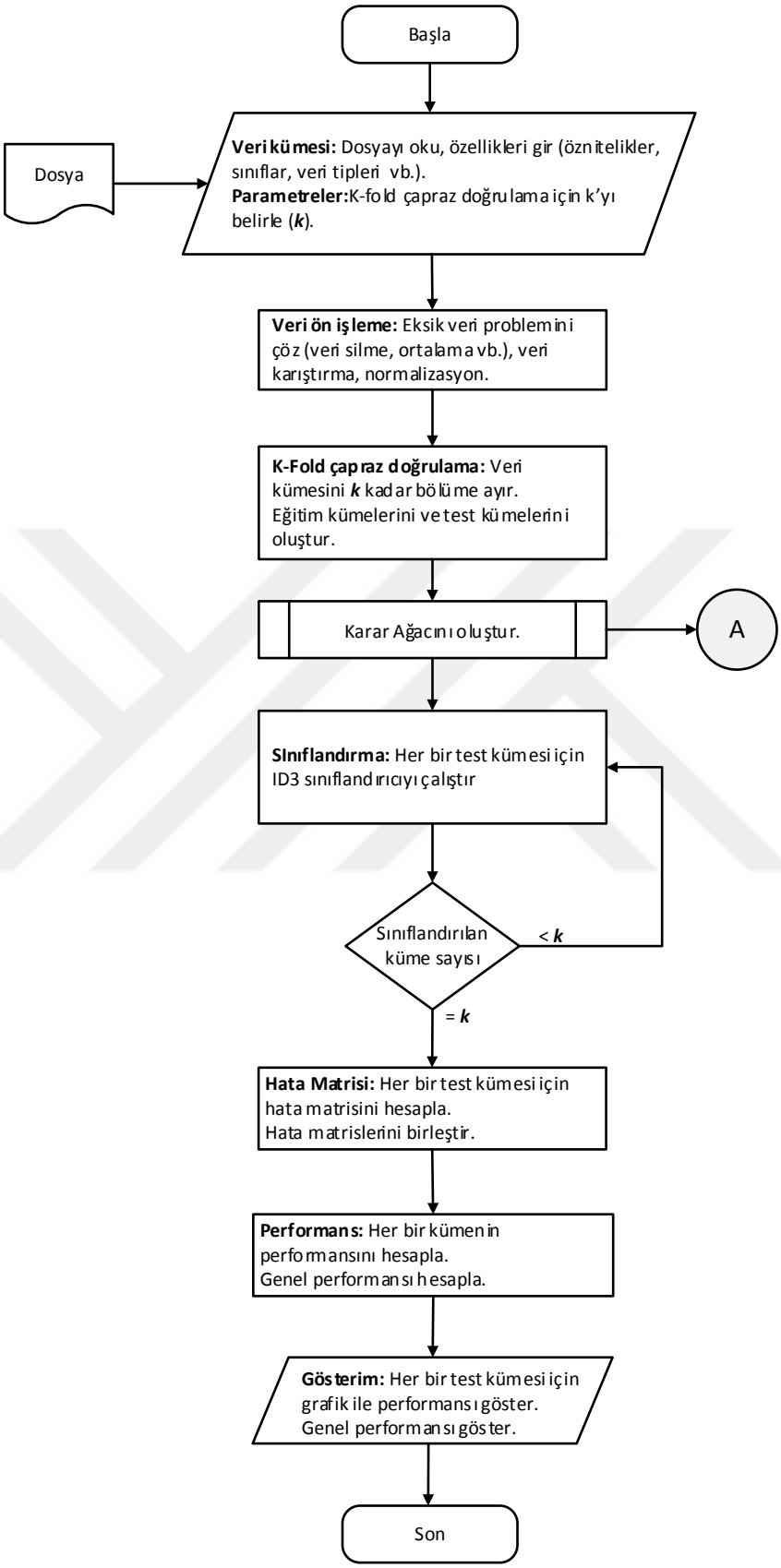
ID3 (ornekler, hedef_ozellik, ozellikler)

- *Bir kök düğümü oluştur.*
- *IF tüm ornekler, hedef özellikleri bakımından aynı c kategorik değerine sahip ise tek düğümlü c etiketine sahip bir ağaç döndür.*
- *IF ozellikler koleksiyonu boş ise ornekler içinde hedef özellikleri bakımından en fazla sayıda bulunan c kategorik değeri ile etiketlenmiş bir ağaç döndür.*
- *ELSE*
  - *A ← ozellikler içinden en iyi kazanç değerine sahip özelliği seç.*
  - *Kök düğümü için karar ← A*
  - *FOREACH A özelliği içindeki her v_i değeri için çevrim.*
    - *Kök düğümün altına A=v_i ye uyan yeni bir düğüm ekle.*
    - *A özelliği v_i olan $ornekler_{vi}$ kümesini ornekler den türet.*
    - *IF $ornekler_{vi}$ boş ise ornekler içinde hedef özellikleri bakımından en fazla sayıda bulunan c kategorik değeri ile etiketlenmiş bir yaprak düğümü ekle.*
    - *ELSE ID3 ($ornekler_{vi}$, $hedef\_ozellik$, ozellikler – {A}) den gelen yeni bir kök düğüm ekle.*
- *END*

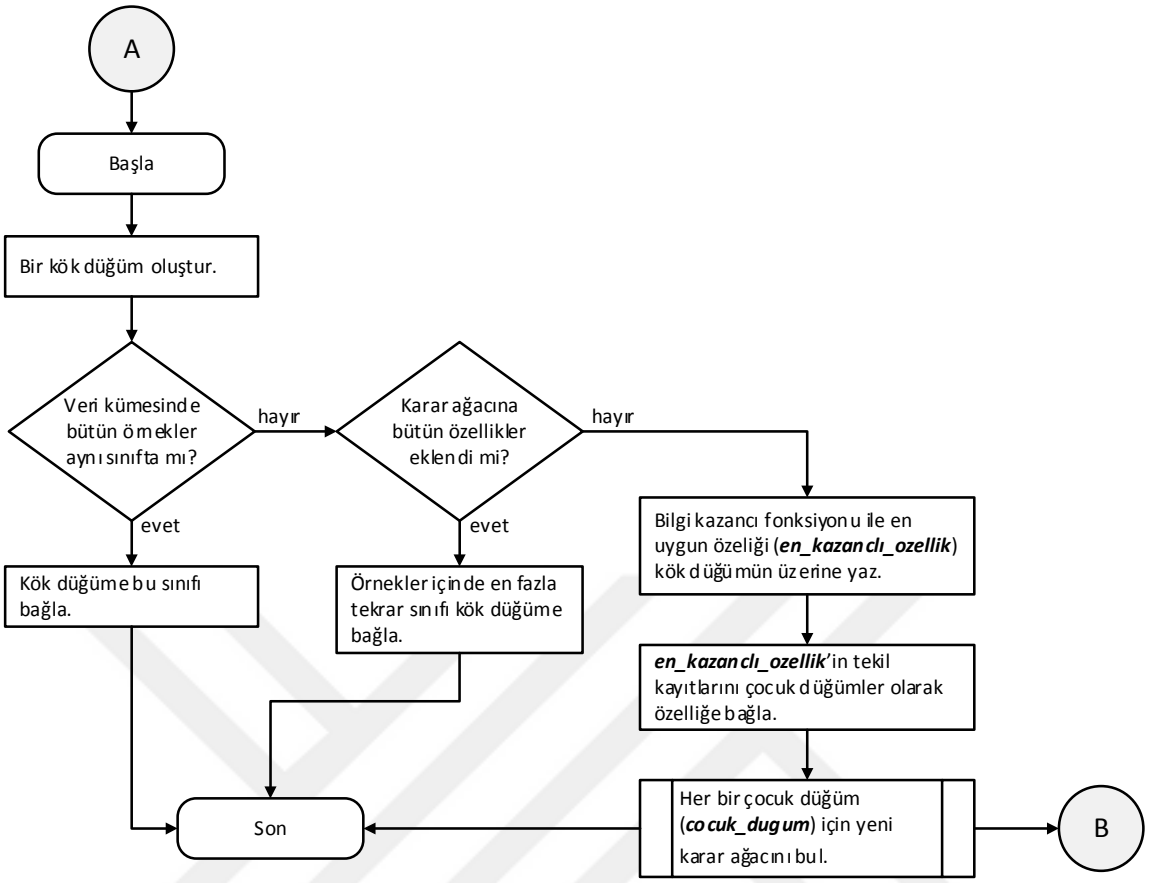
RETURN kök

3.6.4. Uygulama

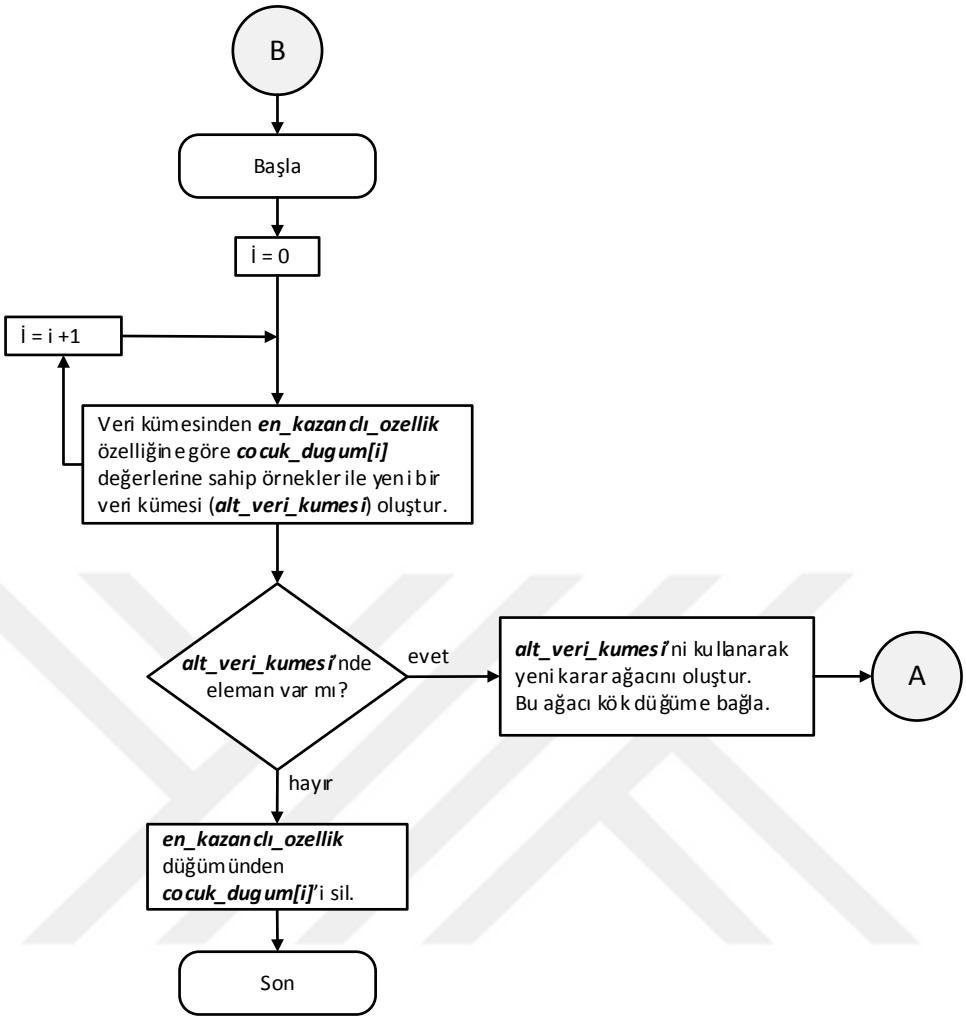
ID3 algoritmasının uygulanması Şekil 3.26, Şekil 3.26 ve Şekil 3.27' deki akış diyagramlarında gösterilmiştir. Bu sınıflandırıcı, "mushroom" veri kümesi ile test edilmiştir. Bölüm 3.2.4.2 de veri kümesi ile ilgili bilgiler verilmiştir.



Şekil 3.26 ID3 akış diyagramı 1.



Şekil 3.27 ID3 akış diyagramı 2.



Şekil 3.28 ID3 akış diyagramı 3.

3.6.5. Deneyler

Tablo 3.6' da yapılan deneylerin sonuçları gösterilmiştir.

Tablo 3.6 Karar ağacı sınıflandırma deney sonuçları.

No	Veri Kümesi*	Örnek sayısı	K-fold değeri	Performans
1	Mushroom-1	5644	2	% 98.44
2	Mushroom-1	5644	20	% 98.44
3	Mushroom-1	5644	200	% 98.44
4	Mushroom-2	8124	2	% 99.40
5	Mushroom-2	8124	20	% 99.40
6	Mushroom-2	8124	200	% 99.40

*Mushroom-1' de eksik veriler çıkartıldı. Mushroom-2' de eksik veriler veri kümesinde en fazla tekrar eden değer ile güncellendi.

4. SONUÇLAR VE DEĞERLENDİRME

Bu tez çalışmasında Makine Öğrenmesi literatüründe yaygın kullanılan beş yöntem incelendi. Ayrıca bu yöntemleri sınamak için oluşturulan modeller, geliştirilen test ortamında farklı parametrik değerler ve "one hot encoding" doğrulama yöntemi ile test edildi. Testlerde iki adet ekolojik veri kümesi (mushroom, iris) kullanıldı. Veri hacmi, veri kümesinin kategorik veya sürekli özelliklerden oluşması, özellik sayısı gibi farklılıklar, modellerin performanslarını etkilediğinden dolayı birbirine benzemeyen iki farklı veri kümesi seçilerek modeller test edildi.

Mushroom veri kümesi, YSA, Naive Bayes ve Karar Ağacı sınıflandırıcılarında kullanıldı. Veri kümesinin sahip olduğu eksik veriler iki şekilde değerlendirildi. İlk yöntemde, eksik değerlere sahip örnekler veri kümesinden çıkartılarak testler gerçekleştirildi. İkinci yöntemde ise eksik değerler en fazla tekrar eden değerler ile güncellenerek test işlemleri gerçekleştirildi. Iris veri kümesi, YSA, Naive Bayes, K-NN sınıflayıcıları ve K-means kümeleme algoritmasında kullanılmıştır. Modellenen tüm MÖ teknikleri, uygun parametrik değerler ile %90 'nın üzerinde başarım oranına ulaşmıştır.

YSA' da eğitim için kullanılan ağ, üç katmalı bir yapı şeklinde modellendi. Bu modele gizli katman sayısının artırılması fonksiyonu eklenerek dört veya daha fazla katmana sahip bir YSA modeli geliştirilebilir. Hem kategorik hem de sürekli verilere sahip veri kümeleri üzerinde çalışabilen YSA sınıflayıcı, kategorik giriş değerlerini "one hot encoding" yöntemi kullanarak kodlar. Bu tekniğin dışında başka kodlama yöntemleri ile sınıflayıcı performansı artırılabilir.

Karar ağacı sınıflandırma ID3 algoritması kullanılarak gerçekleştirildi. Eğitim verileri kullanılarak oluşturulan karar ağaçlarında budama işlemi gerçekleştirilmemiştir. Bu durum aşırı uyum problemine sebep olabilmektedir. Bu açıdan oluşturulan modele budama işlemi eklenerek modelin genelleştirilmesi artırılabilir.

Naive Bayes sınıflandırma yöntemi ile hem kategorik hem de sürekli özelliklere sahip verilerin sınıflandırma işlemi gerçekleştirilmiştir. Sürekli verilerde normal (Gauss) dağılım kullanılarak sınıflandırma işlemi yürütüldü.

K-means kümeleme ve K-NN sınıflandırma işlemleri sürekli özelliklere sahip veri kümesi (iris) ile gerçekleştirildi. Üç farklı uzaklık ölçüm yöntemi her iki teknikte de uygulanarak test edildi.

5. KAYNAKLAR

[1] M. Gillman and R. Hails, "An Introduction to Ecological Modeling: Putting Theory into Practice," *Methods Ecol. Ser. Blackwell Sci. Oxford*, 1997.

[2] A. Fielding, *Machine learning methods for ecological applications*. Springer Science & Business Media, 1999.

[3] P. A. Whigham and G. B. Fogel, "Ecological Applications of Evolutionary Computation," in *Ecological Informatics*, Springer, 2006, pp. 85–107.

[4] M. Debeljak, *Inductive Machine Learning in Ecological Modeling: Invited Talk*. University of Tennessee, Oak Ridge National Laboratory, 2012.

[5] J. Laganis, A. Pečkov, and M. Debeljak, "Modeling radial growth increment of black alder (Alnus glutionsa (L.) Gaertn.) tree," *Ecol. Modell.*, vol. 215, no. 1, pp. 180–189, 2008.

[6] D. Stojanova, P. Panov, V. Gjorgjioski, A. Kobler, and S. Džeroski, "Estimating vegetation height and canopy cover from remotely sensed data with machine learning," *Ecol. Inform.*, vol. 5, no. 4, pp. 256–266, 2010.

[7] A. Ivanovska, L. Todorovski, M. Debeljak, and S. Džeroski, "Modelling the outcrossing between genetically modified and conventional maize with equation discovery," *Ecol. Modell.*, vol. 220, no. 8, pp. 1063–1072, 2009.

[8] M. Debeljak, D. Kocev, W. Towers, M. Jones, B. S. Griffiths, and P. D. Hallett, "Potential of multi-objective models for risk-based mapping of the resilience characteristics of soils: demonstration at a national level," *Soil use Manag.*, vol. 25, no. 1, pp. 66–77, 2009.

[9] S. DŲeroski, *Machine learning applications in habitat suitability modeling*. Springer, 2009.

[10] ISEI, "International Society for Ecological Informatics," 2000. [Online]. Available: http://conference.ecoinformatics.org/index.php/isei/. [Accessed: 09-Jan-2017].

[11] F. Recknagel, "Ecological Informatics–Scope, Technique and Applications." Springer, Berlin, Germany, 2006.

[12] J. W. Brunt, P. McCartney, K. Baker, and S. G. Stafford, "The future of ecoinformatics in long term ecological research," in *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics: SCI*, 2002, pp. 14–18.

[13] A. Demir, “Küresel iklim değişikliğinin biyolojik çeşitlilik ve ekosistem kaynakları üzerine etkisi,” *Ankara Üniversitesi Çevre Bilim. Derg.*, vol. 1, no. 2, pp. 37–54, 2009.

[14] S. Kellogg and S. Pettigrew, *Toolbox for Sustainable City Living: A do-it-Ourselves Guide*. South End Press, 2008.

[15] E. J. Rykiel, “Artificial intelligence and expert systems in ecology and natural resource management,” *Ecol. Modell.*, vol. 46, no. 1–2, pp. 3–8, 1989.

[16] ARIES, “ARIES - ARtificial Intelligence for Ecosystem Services,” 2007. [Online]. Available: http://aries.integratedmodelling.org/?page_id=632. [Accessed: 10-Jan-2017].

[17] M. Pascual, “Computational ecology: from the complex to the simple and back,” *PLoS Comput Biol*, vol. 1, no. 2, p. e18, 2005.

[18] D. H. Fisher, “Computing and AI for a Sustainable Future,” *IEEE Intell. Syst.*, vol. 26, no. 6, pp. 14–18, 2011.

[19] A. E. Thessen, “Adoption of machine learning techniques in Ecology and Earth Science,” PeerJ PrePrints, 2016.

[20] T. M. Mitchell, “Machine learning,” *New York*, 1997.

[21] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[22] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: bringing order to the web.,” 1999.

[23] H. Brink, J. Richards, and M. Fetherolf, *Real-world machine learning*. Manning, 2014.

[24] B. Lantz, *Machine learning with R*. Packt Publishing Ltd, 2013.

[25] Roger D. Peng, “Six Types of Questions - Managing Data Analysis | Coursera.” [Online]. Available: https://www.coursera.org/learn/managing-data-analysis/lecture/8hYa6/six-types-of-questions. [Accessed: 10-Jun-2018].

[26] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*, vol. 2. Prentice hall Upper Saddle River, 2003.

[27] A. M. Turing, “Computing machinery and intelligence,” *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[28] B. Whitby, *Artificial intelligence*. The Rosen Publishing Group, 2009.

[29] J. R. Searle, “Minds, brains, and programs,” *Behav. Brain Sci.*, vol. 3, no. 03, pp. 417–424, 1980.

[30] S. Raschka and V. Mirjalili, *Python Machine Learning - Second Edition.* Packt Publishing, 2017.

[31] R. J. A. Little and D. B. Rubin, *Statistical analysis with missing data*. Wiley, 2002.

[32] A. P. Dempster, ; N M Laird, and ; D B Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *J. R. Stat. Soc. Ser. B*, vol. 39, no. 1, pp. 1–38, 1977.

[33] G. E. A. P. A. Batista and M. C. Monard, “An analysis of four missing data treatment methods for supervised learning,” *Appl. Artif. Intell.*, vol. 17, no. 5–6, pp. 519–533, May 2003.

[34] Y. Dodge, F. H. C. (Francis H. C. Marriott, and International Statistical Institute., *The Oxford dictionary of statistical terms*. Oxford University Press, 2003.

[35] E. Kreyszig, *Advanced engineering mathematics*, Fourth. Wiley, 1979.

[36] K. Potdar, C. Pai, T. S. Pardawala, and C. D. Pai, “A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers,” *Artic. Int. J. Comput. Appl. Int. J. Comput. Appl.*, vol. 175, no. 4, pp. 975–8887, 2017.

[37] R. Gutierrez-Osuna, “Introduction to Pattern Analysis.”

[38] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge University Press, 1996.

[39] Sebastian Raschka, “Model evaluation, model selection, and algorithm selection in machine learning.” [Online]. Available: https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html. [Accessed: 02-Jul-2018].

[40] C. Sammut and G. I. Webb, Eds., *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010.

[41] J. Heaton, “Introduction to Neural Networks for C#, Heaton Research,” *Inc.*, 2008.

[42] M. Kubat, “Artificial neural networks,” in *An Introduction to Machine Learning*, Springer, 2015, pp. 91–111.

[43] D. Kriesel, “A brief Introduction on Neural Networks,” 2007.

[44] Jeff Schlimmer, “UCI Machine Learning Repository: Mushroom Data Set.” [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Mushroom. [Accessed:

16-Jul-2018].

[45] R. O. Duda and P. E. (Peter E. Hart, *Pattern classification and scene analysis*. Wiley, 1973.

[46] scikit-learn.org, “PCA 2d projection of of Iris dataset — scikits.learn v0.6-git documentation.” [Online]. Available: http://scikit-learn.sourceforge.net/0.5/auto_examples/plot_pca.html. [Accessed: 01-Aug-2018].

[47] S. Raschka and V. Mirjalili, *Python Machine Learning - Second Edition*. Packt Publishing, 2017.

[48] R. Panigrahy, “An Improved Algorithm Finding Nearest Neighbor Using Kd-trees,” in *LATIN 2008: Theoretical Informatics*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 387–398.

[49] S. S. Safi and B. Bouikhalene, “Printed Noisy Greek Characters Recognition Using Hidden Markov Model, Kohonen Network, K Nearest Neighbours and Fuzzy Logic,” *Int. J. Signal Process. Image Process. Pattern Recognit.*, vol. 8, no. 10, pp. 241–256, Oct. 2015.

[50] A. Singh, A. Yadav, and A. Rana, “K-means with Three different Distance Metrics,” *Int. J. Comput. Appl.*, vol. 67, no. 10, pp. 975–8887, 2013.

[51] Anonim, “Bias–variance tradeoff.” [Online]. Available: http://www.wiki-zero.org/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQmlhc-KAk3ZhcmlhbmNlX3RyYWRlb2Zm.

[52] R. J. Roiger, *Data mining : a tutorial-based primer*. .

[53] P. Harrington, *Machine learning in action*. Manning Publications, 2012.

[54] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.

[55] J. R. Quinlan, “Bagging, boosting, and C4. 5,” in *AAAI/IAAI, Vol. 1*, 1996, pp. 725–730.

[56] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[57] R. Quinlan, “Data mining tools See5 and C5. 0,” 2004.

[58] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[59] S. Singh and P. Gupta, “Comparative study id3, cart and c4. 5 decision tree algorithm: A survey,” *Int. J. Adv. Inf. Sci. Technol. Vol*, vol. 27, pp. 97–103, 2014.

ÖZGEÇMİŞ

27.04.1986 Tunceli`de dünyaya geldim. İlkokulu Tunceli' de orta ve liseyi ise İstanbul' da tamamladım. Lisans eğitimimi 2009 yılında Sakarya Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümünde tamamladım. Daha sonra çeşitli devlet kurumlarında 2011-2014 yılları arasında üç yıl kadar çalıştım. Fırat Üniversitesi Fen Bilimleri Enstitüsü Ekobilişim Yüksek lisans programına devam etmekteyim.