DOKUZ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEKSTİL SEKTÖRÜNDE YAPAY SİNİR AĞLARI VE TOPLULUK ÖĞRENME ALGORİTMALARI KULLANILARAK KUMAŞ ENİNİN TAHMİNİ

İrem SOYLU

Eylül, 2023
İZMİR

TEKSTİL SEKTÖRÜNDE YAPAY SİNİR AĞLARI VE TOPLULUK ÖĞRENME ALGORİTMALARI KULLANILARAK KUMAŞ ENİNİN TAHMİNİ

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi
İstatistik Anabilim Dalı, Veri Bilimi Programı

İrem SOYLU

Eylül, 2023
İZMİR

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

İREM SOYLU tarafından **PROF.DR. NESLİHAN DEMİREL** yönetiminde hazırlanan **"TEKSTİL SEKTÖRÜNDE YAPAY SİNİR AĞLARI VE TOPLULUK ÖĞRENME ALGORİTMALARI KULLANILARAK KUMAŞ ENİNİN TAHMİNİ"** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Prof.Dr. Neslihan DEMİREL

Danışman

Prof.Dr. Burcu HÜDAVERDİ AKTAŞ

Jüri Üyesi

Prof.Dr. Gözde ULUTAGAY

Jüri Üyesi

Prof. Dr. Okan FISTIKOĞLU

Müdür

Fen Bilimleri Enstitüsü

TEŞEKKÜR

Tez çalışmamın her aşamasında bana rehberlik eden bilgi ve tecrübeleriyle beni yönlendiren desteklerini hiç esirgemeyen değerli hocamız Prof.Dr. Neslihan Demirel'e sonsuz teşekkürlerimi sunarım. Sabrı, özverisi ve destekleri sayesinde çalışmamın kalitesini artırdım. İçtenlikle kendisine minnettarım.

Bu süreçte maddi ve manevi olarak her zaman yanımda olan sevgili aileme sabırları, anlayışları ve teşvikleri için içten teşekkürlerimi sunuyorum.

İrem SOYLU

TEKSTİL SEKTÖRÜNDE YAPAY SİNİR AĞLARI VE TOPLULUK ÖĞRENME ALGORİTMALARI KULLANILARAK KUMAŞ ENİNİN TAHMİNİ

ÖZ

Tekstil sektöründe yeni geliştirilen kumaşlarda, kumaş ilk defa üretileceği için kumaş eni parametresinin tahminlenmesi son derece önemlidir. Bu çalışmada kumaş eninin doğru tahminlenmesi için gerekli etkenlerin belirlenerek yapay sinir ağları ve topluluk öğrenme yöntemleri ile modellenmesi amaçlanmıştır. Bu çalışma, kumaş eni değerinin insandan bağımsız sistematik bir şekilde belirlenmesi ihtiyacından doğmuştur. Mevcut durumda, yeni geliştirilen kumaşlar üretilirken kumaş eni değeri tamamen çalışanın bilgi ve tecrübesine göre belirlenmektedir. Bu da tutarsız ve güvenilir olmayan sonuçların elde edilmesine sebep olmaktadır. Bu şekilde tanımlanan en parametresi nedeniyle kumaş iadesi, kumaş tamiri vb. sorunlar fabrikada ciddi zaman kaybına ve maliyet artışına sebep olmaktadır. Bu çalışma kumaş eni insan deneyimden bağımsız bir şekilde makine öğrenimi algoritmaları kullanılarak tahminlenmiştir. Çalışmada kumaş enini etkileyen faktörler gramaj, ilmek iplik uzunluğu, elastan oranı, viskon oranı, pamuk oranı, polyester oranı, elastan numarası, iplik numarası ve fein değeri olarak belirlenmiştir. Yapay sinir ağları modelleri, regresyon ağaçları, rassal ormanlar regresyonu, gradient boosting regresyonu ve extreme gradient boosting yöntemleri kullanılarak sonuçlar karşılaştırılmıştır. Modelleri değerlendirirken karşılaştırma için hata metriklerinden R^2, Hata Kareler Ortalaması (MSE), Kök Hata Kareler Ortalaması (RMSE) ve Ortalama Mutlak Hata (MAE) kullanılmıştır. Sonuçlar incelendiğinde yapay sinir ağları modellerinden tek katmanlı yapay sinir ağı %88 açıklayıcılık değeri ile en iyi sonucu vermiştir.

Anahtar kelimeler: Tekstil, kumaş eni, yapay sinir ağları, karar ağaçları, torbalama yöntemi, rassal ormanlar, XGBoost

PREDICTION OF FABRIC WIDTH USING ARTIFICIAL NEURAL NETWORKS AND ENSEMBLE LEARNING ALGORITHMS IN THE TEXTILE INDUSTRY

ABSTRACT

In this study, the aim was to model the accurate prediction of fabric width in newly developed fabrics in the textile industry using artificial neural networks and ensemble learning methods. This study emerged from the need for systematically determining the fabric width value independent of human judgment. Currently, in the production of newly developed fabrics, the fabric width value is entirely determined based on the knowledge and experience of the worker. This leads to inconsistent and unreliable results. Due to the variability in fabric width, problems such as fabric returns, fabric repairs, etc., cause significant time loss and cost increase in the factory. In this study, the fabric width was predicted using machine learning algorithms, independent of human experience. The factors influencing fabric width were determined as weight per square meter, loop yarn length, elastane ratio, viscose ratio, cotton ratio, polyester ratio, elastane number, yarn number, and fine value. Artificial neural network models, regression trees, random forest regression, gradient boosting regression, and extreme gradient boosting methods were used, and the results were compared. When evaluating models for comparison, the following error metrics were used: R^2 (R-squared), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). Upon examination of the results, the single-layer artificial neural network yielded the best result with a coefficient of determination 0.88.

Keywords: Textile, fabric width, artificial neural networks, decision trees, bagging method, random forests, XGBoost

İÇİNDEKİLER

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

TABLOLAR LİSTESİ

Sayfa

KISALTMALAR

GBM	: Gradyan Artırma Makinesi (Gradient Boosting Machine)
GRU	: Kapılı Tekrar Birimi (Gated Recurrent Unit)
LSTM	: Uzun Kısa Dönemli Bellek (Long Short-Term Memory)
MAE	: Ortalama Mutlak Hata (Mean Absolute Error)
MLP	: Çok Katmanlı Algılayıcı (Multi-Layer Perceptron)
MSE	: Hata Kareler Ortalaması (Mean Square Error)
RMSE	: Kök Hata Kareler Ortalaması (Root Mean Square Error)
RNN	: Tekrarlayan Sinir Ağları (Recurrent Neural Network)
RF	: Rassal Ormanlar (Random Forests)
XGBoost	: Aşırı Gradyan Artırma (Extreme Gradient Boosting)

BÖLÜM BİR
GİRİŞ

Günümüzde tekstil endüstrisinde kumaş üretiminde teknoloji çok önemli bir rol oynamaktadır. Kaliteli ürün talebi ve hızla değişen ekonomik koşullar, hazır giyim üreticilerinin doğru kalitede ürün üretme konusundaki önemini artırmaktadır. Ayrıca üretim hatalarını en aza indirmek, süreç performansını iyileştirmek ve sürdürmek için üretim sırasındaki değişkenlik kaynaklarının belirlenmesi son derece önemlidir. Tekstil sektöründe artan talep ve yüksek müşteri beklentileri nedeniyle üretim süreçlerinde iyileştirme, hız, minimum hatalı üretim, maksimum ürün kalitesi faktörlerinin karşılanması zorunlu bir ihtiyaç haline gelmiştir. Örme süprem kumaşların ham kumaş üretiminden itibaren daha rahat ve boyutsal stabilite ile üretilmesi, işletme de manuel olarak belirlenen kumaş kalite karakteristiğinin bir ölçüsü olan en değerlerinin daha sistematik belirlenerek insana bağlı hata payının azaltılması, kumaşları tek seferde üretebilme başarısının arttırılması böylece üretim hattında hızlı üretimin gerçekleşebilmesi için bu çalışmanın temel amaçlarıdır. Bu çalışma da kumaş eninin tahminlenmesinde denetimli makine öğrenmesi yöntemlerinden yapay sinir ağları ve topluluk öğrenme yöntemlerine başvurulmuştur.

Tekstil endüstrisinde tahmine dayalı çalışmalarda Yapay Sinir Ağlarının (YSA) kullanımı son zamanlarda artmıştır. Yapay sinir ağları modelleri, tekstil imalatındaki hataların sınıflandırılması, iplik kalitesi parametrelerinin tahmini, dokuma ve örme kumaşların sınıflandırılması, kumaş davranış özelliklerinin tahmini, giysi konforunun tahmini, dokuma ve örme kumaşların hava geçirgenliği özellikleri, kumaş dökümünün tahmini, boya maddesi konsantrasyonunun tahmini gibi konularda pek çok çalışmalarda yer almıştır. Aynı zamanda ağaca dayalı çok yapılı optimize edilmiş güçlü yöntemlerden topluluk öğrenmesi modellerinden Rassal Ormanlar (RF), Gradyan Artırma (GBM) ve Aşırı Gradyan Artırma (XGBoost) algoritmalarının avantajlarını değerlendirmek bu yöntemler ile yapay sinir ağlarını modellerini karşılaştırarak incelemek ve en iyi modeli belirlemek bu çalışmanın konusu olmuştur. Modelleri karşılaştırırken belirtme katsayısı R^2, Hata Kareler Ortalaması (MSE), Kök Hata Kareler Ortalaması (RMSE) ve Ortalama Mutlak Hata (MAE) kullanılmıştır.

Beltran vd. (2006), süprem ve ribana saf yün örme kumaşların elyaf, iplik ve kumaş özellikleri ile boncuklanma eğilimi arasındaki ilişkileri belirlemek için çok katmanlı yapay sinir ağı modellemesi gerçekleştirmiştir. Aktivasyon fonksiyonu olarak sigmoid kullanmıştır. Yapay sinir ağı modeli ile çoklu doğrusal regresyon modelini karşılaştırmış ve yapay sinir ağı modeli ile daha yüksek açıklayıcılığa sahip sonuç elde etmiştir.

Ersöz vd. (2021), veri madenciliği yöntemleri kullanarak hazır giyim üretimi yapan bir firmanın hatalı üretim verileri ile hatanın ana nedenlerinin belirlenmesini amaçlamıştır. Çalışmada Karar Ağaçları, Naive Bayes, Rassal Ormanlar ve Gradyan Artırma algoritmaları kullanılmıştır. Araştırma sonunda modellerin doğruluk oranları karşılaştırılmıştır. Karar ağaçları algoritmalarının kullanılan diğer sınıflandırıcı algoritmalara göre daha yüksek doğruluk oranlarına sahip olduğu sonucuna varılmıştır. Hataların ana sebeplerinin üretim kaynaklı hatalar ve kumaş hataları olduğu görülmüştür.

Oğulata vd. (2006), streç dokuma kumaşların uzama ve elastikiyet değerlerini çok katmanlı yapay sinir ağları ve doğrusal regresyon modelleri kullanarak tahmin etmişlerdir. Her iki modelin sonucunun da tahmin doğruluğunun yüksek olması ile birlikte çok katmanlı yapay sinir ağı modeli daha iyi sonuç vermiştir.

Atik vd. (2022), makine öğrenmesi modelleri kullanılarak bir tekstil firmasında kumaş israfının tahminlenmesini gerçekleştirmişlerdir. Sabit fire oranı ile üretim yapmak yerine üretim sürecini etkileyen birçok farklı parametreyi değerlendirerek siparişe göre değişen fire oranları elde etmeyi amaçlamışlardır. Çalışmada dört makine öğrenme yöntemi, Destek Vektör Makineleri, Rassal Ormanlar, Karar Ağacı ve Torbalama algoritmaları kullanılmıştır. Sonuçlar incelendiğinde torbalamanın diğer yaklaşımlara göre %86 ile en yüksek açıklayıcılık değerini ve minimum tahmin hatasını verdiğini göstermektedir.

Gharehaghaji vd. (2007), yapay sinir ağları ve çoklu doğrusal regresyon yöntemlerini kullanarak, pamuk karışımlı naylon özlü ipliklerin kopma mukavemeti

ve kopma uzaması tahminlemiştir. Geliştirilen modellerde test verileri hata kareler ortaması ve belirtme katsayısı ile karşılaştırılmıştır. Sonuçlar, yapay sinir ağı algoritmasının çoklu doğrusal regresyona kıyasla daha iyi performans gösterdiğini göstermiştir.

Balla vd. (2021), yaptıkları çalışmada tekstil sektöründe hazır giyim çalışanlarının fiili üretkenliklerinin, zaman zaman üretim hedeflerine ulaşmak için yetkililer tarafından belirlenen verimlilik hedeflerine ulaşamaması ve bunun da büyük kayıplara neden olması durumunu incelemişlerdir. Hazır giyim çalışanlarının üretkenliğini tahmin etmek için makine öğrenimi yöntemlerinden rassal orman modeli, doğrusal regresyon ve yapay sinir ağı modellerini kulanmışlardır. Model sonuçlarını belirtme katsayısı, MAE ve RMSE değerleri ile karşılaştırmışlardır. En küçük hata değerini rassal ormanlar modeli, en yüksek belirtme katsayısını yapay sinir ağları ile elde etmişlerdir.

Guruprasad vd. (2015), pamuklu dokuma kumaşların bükülme sertliğini yapay sinir ağları ve genetik algoritma yöntemleri ile tahminlemişlerdir. İlk model geri yayılımlı yapay sinir modeli, ikinci model ise hibrit genetik algoritma modelidir. Geri yayılımlı sinir ağı modeli ile genetik algoritma modellerinin tahmin değerleri karşılaştırılıp en iyi sonucun genetik algoritma ile kurulan modelin olduğuna varılmıştır.

Bu çalışma teorik ve uygulama olarak iki kısımdan oluşmaktadır. Çalışmanın ilk bölümünde tekstil sektöründe makine öğrenmesi yöntemlerini kullanmanın önemini içeren bir giriş ve geniş çaplı bir literatür taraması yapılmıştır. İkinci bölümünde tez çalışması kapsamında yapay sinir ağlarının tanımı, özellikleri, yapay sinir ağlarında öğrenme ve test edilmesi açıklanmıştır. Üçüncü bölümde topluluk öğrenme yöntemlerinden Regresyon Ağaçları, Rassal Ormanlar (RF), Gradyan Artırma (GBM) ve Aşırı Gradyan Artırma (XGBoost) algoritmaları açıklanmıştır. Dördüncü bölümde bir tekstil firmasında üretilen örme süprem kumaşlarda kumaş eninin doğru tahminlenmesi için kumaş enini etkileyebilecek değişkenler belirlenerek veri toplanmıştır. İnsan faktörünün ortadan kaldırılarak sistematik bir şekilde kumaş enlerinin belirlenmesi ve bu sebeple ortaya çıkan tamir oranının azaltılması için yapay

sinir ağları ve topluluk öğrenme yöntemleri ile modeller geliştirilmiştir. Ele alınan modeller arasında performans kıyaslaması yapılarak en uygun model belirlenmiştir. Son bölümde tez çalışması sonucu yorumlanarak konuyla ilgili ileride yapılabilecek çalışmalar için bazı önerilere yer verilmiştir.

BÖLÜM İKİ
YAPAY SİNİR AĞLARI

2.1 Yapay Sinir Ağlarının Tanımı

Yapay Sinir Ağları (YSA), biyolojik sinir sistemlerinin bilgiyi işleme biçiminden ilham alan bir bilgi işleme yaklaşımıdır. Belirli sorunları çözmek için uyum içinde çalışan çok sayıda birbirine bağlı işleme elemanlarından (nöronlar) oluşur. Başka bir deyişle, yapay sinir ağları biyolojik nöronlar gibi davranan bileşenleri kullanarak insan beynine benzer şekilde çalışan makineler yaratma girişimidir.

Bir insan beyni, insan duyuları (özellikle görme) tarafından gönderilen verileri kullanarak büyük miktarda bilgiyi işleyebilir. İnsan beyni, birbirine bağlı milyarlarca nörondan oluşan bir sistem aracılığıyla bilgiyi işleyen karmaşık bir nöron ağıdır. Beyin fonksiyonlarını anlamak her zaman zor olmuştur ancak bilgi işlem teknolojilerindeki gelişmeler nedeniyle günümüzde sinir ağları yapay olarak modellenebilmektedir. Yapay sinir ağları disiplini, sorunu çözmeye çalışan aynı insan beyninin işleyişini taklit etme düşüncesinden doğmuştur. Makine öğrenmesi uygulamaları, bilgisayarları insan zekası ile simüle etmek amacıyla bilgisayarların öğrenmeleri ve sorunları çözmeleri için programlamak üzere bilişsel yetenekler vermeyi amaçlayan bir çalışma alanıdır. Yapay zeka, insan zekasını tamamen taklit edemez fakat insan beyninin sadece bazı yönlerini taklit edecek şekilde programlanabilir. Makine öğrenmesi, bilgisayarların girdi verilerine göre kendilerini programlamalarına yardımcı olan bir yapay zeka dalıdır. Makine öğrenmesi yapay zekaya veri tabanlı problem çözme yeteneği verir.

Yapay sinir ağları son yıllarda özellikle makine öğrenmesi alanında çok önemli bir role sahip olmuştur. Örneğin, finansal verilerin analizi, görüntü işleme, doğal dil işleme, ses tanıma, robotik gibi birçok uygulama alanında kullanılmaktadır. Tekstil sektöründe de artan bir kullanım yerine sahiptir. Yapay sinir ağları aynı zamanda büyük veri kümeleri üzerinde çalışabilen ve karmaşık yapıları tanıyabilen bir modeldir. Bu özellikleri sayesinde, özellikle yapay zeka ve derin öğrenme alanında

büyük ilgi görmektedir. Yapay sinir ağları birçok farklı alanda kullanılmakta ve önemi giderek artmaktadır.

Bir sinir ağı, deneyimsel bilgiyi depolamak ve onu kullanıma hazır hale getirmek için doğal bir eğilime sahip basit işlem birimlerinden oluşan büyük ölçüde paralel dağıtılmış bir işlemcidir. İnsan beynine iki yönden benzemektedir: Bilgi, ağ tarafından bir öğrenme süreci ile çevresinden elde edilir ve nöronlar arasındaki ağırlıklar edinilen bilgileri depolamak için kullanılır.

Bir yapay sinir ağı modelinin üç temel öğesi aşağıdaki gibi tanımlanabilir:

- Ağırlıkları ile karakterize edilmiş bağlantı noktaları
- Bağlantı noktalarındaki ağırlıkların toplanması ile öğrenme algoritmasının uygulanması,
- Aktivasyon fonksiyonu

2.2 Yapay Sinir Ağlarının Özellikleri

Bir sinir ağı bilgi işleme gücünü ilk olarak büyük ölçüde paralel dağıtılmış yapısından ikinci olarak ise öğrenme ve dolayısıyla genelleme yeteneğinden alır. Genelleme, sinir ağının eğitim (öğrenme) sırasında daha önce karşılaşılmayan girdiler için makul çıktılar üretmesini ifade eder. Bu iki bilgi işleme yeteneği, sinir ağlarının zor olan karmaşık ve büyük ölçekli sorunlara iyi yaklaşık çözümler bulmasını mümkün kılar (Haykin, 2009). Yapay sinir ağlarının literatürde yer alan özellikleri aşağıdaki gibi özetlenmiştir:

Paralel işleme özelliği, yapay sinir ağlarının büyük miktarda veriyi işlemesine ve hesaplamaları hızlı bir şekilde gerçekleştirmesine izin verir. Girdi verilerinin çok büyük ve karmaşık olabildiği görüntü ve konuşma tanıma gibi görevler için özellikle önemlidir. Genel olarak, sinir ağlarının paralel işleme yetenekleri, onları büyük miktarda veriyi işlemek ve karmaşık hesaplamaları hızlı bir şekilde gerçekleştirmek için oldukça verimli hale getirir. Bu durum, sinir ağlarının makine öğrenimi ve yapay

zeka uygulamaları için bu kadar popüler bir araç haline gelmesinin temel nedenlerinden biridir.

Doğrusal olmama özelliği, sinir ağlarının verilerdeki karmaşık kalıpları ve ilişkileri modellemelerine izin veren temel özelliklerinden biridir. Yapay sinir ağları, girdiler ve çıktılar arasındaki doğrusal olmayan ilişkileri modelleyerek karmaşık kalıpları öğrenmelerine olanak tanır.

Doğrusal bir modelde çıktı, girdi özelliklerinin doğrusal bir kombinasyonudur ve bu durum her girdinin etkisinin ağırlığıyla orantılı olduğu anlamına gelir. Bununla birlikte birçok gerçek yaşam problemlerinde ilişki doğrusal değildir, yani her girdinin etkisi diğer girdilerin değerine bağlı olarak değişebilir.

Sinir ağları, her bir nöronun aktivasyon fonksiyonu aracılığıyla doğrusal olmama özelliğini ortaya çıkarır. Aktivasyon fonksiyonu, her nörondan gelen giriş sinyalini bir sonraki katmana ileten bir çıkış sinyaline dönüştürür. Literatürde en sık kullanılan aktivasyon fonksiyonu, girdilerin ağırlıklı toplamını alan ve bunu 0 ile 1 arasında bir değere eşleyen sigmoid fonksiyonudur. Aktivasyon fonksiyonu yapay sinir ağlarında girdiler ve çıktılar arasındaki karmaşık ve doğrusal olmayan ilişkileri modelleyebilmektedir. Böylece, doğrusal bir model tarafından yakalanamayacak örnekleri öğrenmeye olanak tanır. Örneğin, görüntü tanıma uygulamasında bir sinir ağı, bir görüntüdeki doğrusal bir model tarafından kolayca tanımlanamayan kalıpları tanımayı öğrenebilir. Ağ, farklı ölçeklerde ve yönlerde kenarları, köşeleri ve diğer özellikleri algılamayı öğrenebilir ve görüntüdeki nesneleri tanımak için bu özellikleri birleştirebilir.

Genel olarak, doğrusal olmama, sinir ağlarının verilerdeki karmaşık ilişkileri modellemelerine izin veren çok önemli bir özelliğidir ve çok çeşitli makine öğrenimi görevleri için bu kadar güçlü olmalarının nedenlerinden biridir.

Verilerden öğrenme özelliği sinir ağlarının en önemli diğer özelliklerinden biridir. Bir yapay sinir ağı bir veri kümesi üzerinde eğitildiği ve verilerde keşfettiği kalıplara

dayalı olarak tahminlerde bulunmayı veya verileri sınıflandırmayı öğrenmektedir. Bu öğrenme süreci genellikle derin öğrenme olarak adlandırılır ve verilerdeki karmaşık kalıpları öğrenmek için kullanılır. Genel olarak, sinir ağlarının verilerden öğrenme yeteneği, onları görüntü ve konuşma tanımadan doğal dil işlemeye ve tahmine dayalı modellemeye kadar çok çeşitli makine öğrenimi görevleri için uygun hale getiren temel bir özelliktir. Bir sinir ağındaki öğrenme süreci tipik olarak aşağıdaki adımları içerir:

- Başlama: Ağ, rastgele ağırlıklar ve sapma (bias) ile başlatılır.
- İleri yayılım: Girdi verileri ağdan geçirilir ve her bir nöronun çıktısı, ağırlıklara ve sapmaya göre hesaplanır.
- Kayıp fonksiyonunun hesaplanması: Tahminlenen çıktı ile gerçek çıktı arasındaki fark, hata kareler ortalaması veya çapraz entropi gibi bir kayıp fonksiyonu kullanılarak hesaplanır.
- Geriye doğru yayılma: Hata sinyali ağ üzerinden geri yayılır ve kayıp fonksiyonunu en aza indirmek için her bir nöronun ağırlıkları ve sapmaları ayarlanır.

Ağ, eğitim verilerinde kabul edilir iyi bir performans düzeyine ulaşana kadar ileri yayılım, kayıp fonksiyonunun hesaplanması ve geriye doğru yayılım süreci tekrarlanır.

Uyarlanabilirlik özelliği, sinir ağlarının girdi verilerindeki veya ortamlarındaki değişikliklere uyum sağlama ve bu değişikliklere rağmen doğru tahminler veya sınıflandırmalar yapmaya devam etme yeteneğini ifade eder. Belirli bir ortamda çalışmak üzere eğitilmiş bir sinir ağı, çalışma ortamı koşullarındaki küçük değişikliklerle başa çıkmak için kolayca yeniden eğitilebilmektedir. Bir sistemi ne kadar uyumlu hale getirirsek ve kararlı kalmasını sağlarsak sistemin durağan olmayan bir ortamda çalışması gerektiğinde performansının muhtemelen o kadar güçlü olacağı söylenebilir. Sinir ağlarının uyarlanabilirlik özelliğini sergileyebilmesinin birkaç yolu vardır:

- Sağlamlık (Robutness): Sinir ağları, gürültüye ve diğer girdi varyasyonlarına karşı dayanıklı olacak şekilde tasarlanabilir. Örneğin, yüz görüntülerini tanımak için eğitilmiş bir sinir ağı, aydınlatma koşullarındaki veya yüz ifadelerindeki değişikliklere karşı dayanıklı olacak şekilde tasarlanabilir.
- Transfer öğrenimi: Sinir ağları bir görev üzerinde eğitilebilir ve daha sonra daha az eğitim örneği ile başka bir ilgili göreve uyarlanabilir. Örneğin, kedilerin görüntülerini tanımak için eğitilmiş bir sinir ağı, yalnızca az miktarda ek eğitim verisi ile kaplanların görüntülerini tanıyacak şekilde uyarlanabilir.
- Çevrimiçi öğrenme: Sinir ağları, yeni veriler kullanıma sunuldukça gerçek zamanlı olarak güncellenebilir. Bu, borsa tahmini veya hava tahmini gibi girdi verilerinin sürekli değiştiği uygulamalarda özellikle kullanışlıdır.
- Takviyeli öğrenme: Sinir ağları, ağın performansına dayalı olarak ödüller veya cezalar şeklinde geri bildirim sağlamayı içeren takviyeli öğrenme kullanılarak eğitilebilir. Bu yaklaşım, ağın çevresinden gelen geri bildirime dayalı kararlar almayı öğrenmesi gereken oyun oynama gibi görevler için özellikle uygundur.

Genel olarak, sinir ağlarının uyarlanabilirliği, robotik ve kontrol sistemlerinden doğal dil işlemeye ve tahmine dayalı modellemeye kadar çok çeşitli uygulamalarda kullanılmalarına izin veren önemli bir özelliktir. Sinir ağlarının girdi verilerindeki veya ortamdaki değişikliklere uyum sağlama yeteneği, onları karmaşık gerçek dünya problemlerini çözmek için güçlü bir araç haline getirir.

Genelleştirme özelliği, bir sinir ağının daha önce görmediği veriler üzerinde doğru tahminler veya sınıflandırmalar yapmasını ifade eder. Genelleme özelliği aynı zamanda bir sinir ağının, eğitim sürecinde kullanılmayan veriler üzerinde doğru tahminler veya sınıflandırmalar yapabilmesi gereken gerçek dünya uygulamalarında kullanılmasına izin verir. Ağı düzenleme, bırakma, veri artırma ve diğer teknikleri kullanarak sinir ağları, gerçek dünya verileri üzerinde etkili bir şekilde genelleme yapmak ve doğru tahminler veya sınıflandırmalar yapmak için modellenebilmektedir (Haykin, 2009).

Hata toleransı özelliği, bir sinir ağının bazı bileşenleri (nöronlar veya bağlantılar gibi) hasar gördüğünde bile doğru tahminler veya sınıflandırmalar yapmaya devam etme yeteneğini ifade etmektedir. Sinir ağlarının hata toleransı gösterebilmesinin birkaç yolu vardır. Bunlardan ilki sinir ağları, bu bileşenlerden bazıları arızalansa bile çalışmaya devam etmelerini sağlayan yedek bağlantılar veya nöronlar ile tasarlanabilir. İkincisi, sinir ağları kendi iç yapılarını girdi verisindeki veya ortamdaki değişikliklere yanıt olarak yeniden düzenlemelerine izin veren kendi kendini organize etme davranışı sergileyebilir. Bu, ağın hasara veya arızalara uyum sağlamasına ve doğru tahminler veya sınıflandırmalar yapmaya devam etmesine yardımcı olabilir. Genel olarak, sinir ağlarının hata toleransı, onları kontrol sistemleri, robotik ve güvenlik açısından kritik uygulamalar gibi güvenilirlik ve sağlamlığın kritik olduğu uygulamalar için çok uygun kılan önemli bir özelliktir.

2.3 Yapay Sinir Ağlarının Tarihçesi

1940 ve 1950'lerin başlarında, sinir ağları üzerindeki ilk çalışmalar da beyin yapısı ve işlevinden ilham alınarak hesaplama modelleri geliştirme üzerine odaklanıldı. Bir nöronun ilk hesaplamalı modeli, 1943'te basit bir yapay nöron modeli öneren Warren McCulloch ve Walter Pitts'in çalışmasıdır.

1940'lardaki ümit verici başlangıca rağmen, sinir ağı araştırmaları sonraki yıllarda pek çok engelle karşılaşmıştır. En büyük zorluklardan biri, bir ağın ağırlıklarını örneklerden öğrenebilecek verimli eğitim algoritmalarının olmamasıydı. 1980'lerde geri yayılma algoritmasının geliştirilmesi, çok katmanlı ağların verimli bir şekilde eğitilmesine izin veren büyük bir atılımdır (Rumelhart vd., 1986)

Son yıllarda derin öğrenme, görüntü ve konuşma tanıma, doğal dil işleme ve diğer alanlarda önemli ilerlemelere yol açmıştır. Derin öğrenmenin en önemli yeniliği, çok katmanlı sinir ağlarının kullanılmasıdır. Bu sayede onların verilerden daha karmaşık özellikleri ve kalıpları öğrenmelerine olanak tanımaktadır.

2.4 Yapay Sinir Hücresi

Yapay sinir hücresi perceptron, biyolojik bir sinir hücresinin davranışını simüle etmek için kullanılan matematiksel bir modeldir. Yapay sinir ağlarının en basit şeklidir ve ikili sınıflandırma görevleri için kullanılabilir.

Yapay sinir hücresinin formülasyonu, Rosenblatt (1958) tarafından matematiksel olarak şu şekilde ifade edilmiştir: Çıktı, ağırlık vektörünün girdi vektörü ile iç çarpımının eşik değerinden büyük veya küçük olmasına bağlı olarak 1 veya 0'dır. $\sum_j w_j x_j + b$ ifadesi ağırlık vektörü w_j ve girdi vektörü x_j arasındaki iç çarpımın sapma terimiyle toplamını temsil eder. Yapay sinir hücresi, her biri bir ağırlıkla ilişkili birkaç giriş sinyali alır ve bir eşik fonksiyonuna dayanarak ikili bir çıkış sinyali üretir. Eşik değeri, sinir hücresinin çıktısını belirleyen bir eşik değeridir. Eğer iç çarpım eşik değerinden büyükse çıktı 1'dir; aksi takdirde çıktı 0'dır. Formülasyon eşitlik 2.1'de verilmiştir.

$$y = \begin{cases} 1 \ , \sum_j w_j x_j > Eşik \ \ değeri \\ 0 \ , \sum_j w_j x_j \leq Eşik \ \ değeri \end{cases} \tag{2.1}$$

Ağırlıklar ve sapma terimi, yapay sinir hücresinin eğitim aşamasında denetimli bir öğrenme algoritması kullanılarak öğrenilir. Amaç, perceptron'un belirli bir giriş sinyali için doğru çıkış sinyalini üretmesi için ağırlıkları ve sapma terimini ayarlamaktır. Perceptron algoritması, ikili sınıflandırma görevleri için kullanılabilen basit ama güçlü bir algoritmadır. Ancak, daha karmaşık problemleri çözme konusunda sınırlamaları vardır.

2.5 Yapay Sinir Ağının Yapısı

Yapay sinir ağlarının temel yapı taşı yapay sinir hücresidir. Bu hücreler, katmanlar halinde düzenlenir ve birbirleriyle bağlantılıdır. Bu bağlantılar, hücreler arasındaki sinyallerin iletimini sağlar ve ağın çıktısını belirlemektedir.

Biyolojik nöron yapısına benzer şekilde, yapay sinir ağları bir dizi girdiden bir çıktı üretmek için matematiksel bir işlem gerçekleştiren merkezi bir işlem birimi olarak tanımlanır. Bir nöronun çıktısı, girdilerin ağırlıklı toplamı artı yanlılığın bir fonksiyonudur. Her nöron alınan toplam sinyal miktarı bir etkinleştirme eşiğini aşarsa etkinleştirmeyi içeren bir işlem gerçekleştirir. Esasen yapay sinir ağları, bir dizi matematiksel fonksiyon yaklaşımıdır. Yapay sinir ağının bileşenleri temel olarak aşağıdaki gibidir:

- Katmanlar
- Ağırlıklar
- Sapma
- Toplama Fonksiyonu
- Aktivasyon Fonksiyonları

2.5.1 Yapay Sinir Ağlarında Katmanlar

Yapay sinir hücreleri katmanlarda organize edilir ve bu katmanlar bir ağ oluşturur. Bir yapay sinir ağının temel yapısı, giriş katmanı, gizli katman ve çıkış katmanı olmak üzere üç katmandan oluşur. Giriş katmanı, ağın girdilerini aldığı ilk katmandır. Çıkış katmanı, ağın çıktılarını üreten son katmandır. Giriş ve çıkış katmanları arasında bir veya daha fazla gizli katman vardır ve bunlar girdiler üzerinde hesaplamalar yapar (Goodfellow, Bengio ve Courville, 2016).

Girdi katmanı verilerin ağa giriş noktasıdır. Bu katman, dışarıdan verilerin ağa aktarılmasına ve daha sonra ağın geri kalan kısımlarında işlenmesine izin verir. Girdi katmanı, girdi verilerinin her özelliği için bir nöron içerir. Her bir nöron, girdi verilerinin bir özelliğini temsil eder. Girdi katmanındaki nöronlar, ağırlıklar ile ilişkilendirilir. Bu ağırlıklar, girdi verilerinin özelliklerinin ağırlıklı toplamını hesaplamak için kullanılır ve sonuçlar bir sonraki katmana aktarılır. Ancak bu katmanda girdi verileri üzerinde belirli bir işlem yapılmaz. Girdi katmanındaki nöronlar sadece verileri bir sonraki katmana iletirler ve girdi verileri üzerinde herhangi bir işlem yapmazlar.

Girdi katmanı verileri güzel bir şekilde paketlenmiş olarak ağa almanın yolu olarak düşünülebilir (Nielsen, 2015). Bu ifade, girdi katmanının, girdi verilerini ağın diğer katmanlarına iletmek için bir araç olarak kullanıldığını ifade etmektedir. Güzel bir şekilde paketlenmiş ifadesi, girdi verilerinin, ağın diğer katmanlarında daha etkili bir şekilde işlenebilmesi için uygun bir formatta sunulduğunu ifade eder. Bu, verilerin ön işlemeye tabi tutulması gerektiği anlamına gelmektedir ve normalizasyon veya özellik çıkarma gibi işlemler yapılabilir.

Gizli katman, girdi katmanı ve çıktı katmanı arasındaki katman olup bir veya daha fazla gizli nöron içerir. Bu nöronlar önceki katmandan girdiler alır ve bir ağırlık toplamı yaparlar, ardından bir aktivasyon fonksiyonu kullanarak bir çıktı üretirler ve çıktı sonraki katmana aktarılır. Gizli birimler girdilere erişebilir ve birbirleriyle etkileşime girebilirler ve bu etkileşimler doğrudan girdiler veya çıktılar tarafından etkilenmez (Goodfellow, Bengio ve Courville, 2016). Bu, gizli birimlerin girdinin çıktı için yararlı ara temsillerini oluşturmasına olanak tanır. Başka bir deyişle, gizli katmanlar, sinir ağının daha karmaşık özellikleri öğrenmesine ve temsil etmesine olanak sağlar. Her gizli nöron, girdi verilerindeki belirli desenleri algılamayı ve temsil etmeyi öğrenir, sonraki katmanlar tarafından birleştirilerek ve dönüştürülerek daha yüksek düzeyde özellikler oluşturulur.

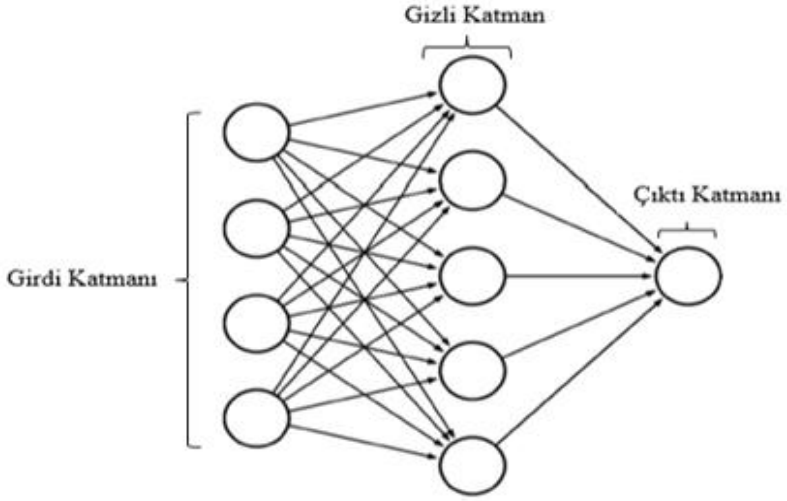
Gizli katman sayısı ve her gizli katmandaki nöron sayısı, sinir ağının öğrenme ve genelleme yeteneğini büyük ölçüde etkileyebilen hiperparametrelerdir. Az sayıda gizli nöron veya katman, yetersiz öğrenmeye yol açabilirken, çok sayıda gizli nöron veya katman, aşırı öğrenmeye neden olabilir. Az sayıda gizli nöron veya katman kullanıldığında, sinir ağı modeli çok basit olur ve karmaşık veri yapılarını veya özelliklerini yakalamakta zorlanır. Bu durumda, yetersiz öğrenme söz konusu olabilir ve modelin performansı düşük olabilir. Diğer yandan, çok sayıda gizli nöron veya katman kullanıldığında, sinir ağı modeli çok karmaşık hale gelir ve eğitim verilerine aşırı uyum sağlamaya başlar. Bu durumda, model eğitim verilerinde yüksek performans gösterebilir, ancak yeni verilere veya test verilerine uygulanırken kötü bir genelleme yapabilir ve aşırı öğrenme söz konusu olabilir (Haykin,2009). Yapay sinir ağlarında gizli katman ve nöron sayısını belirlemek, ağın performansını etkileyen

kritik bir karardır. Bu sayıların doğru seçilmesi, ağın doğruluğu, hızı ve genel performansı açısından önemlidir. Gizli katman ve nöron sayısı seçimi, genellikle deneme yanılma yoluyla yapılır. Goodfellow, Bengio ve Courville (2016), gizli katman sayısını belirlerken başlangıçta bir veya iki gizli katman kullanılmasını, ağın performansına bağlı olarak katman sayısının sonradan arttırılmasını önermişlerdir.

Nöron sayısı seçimi ise, gizli katmanların toplam sayısına ve problem büyüklüğüne bağlıdır. Bishop (1995), nöron sayısının çok yüksek seçilmesi, ağın aşırı öğrenmesine neden olabilir, bu da ağın genelleme performansını olumsuz yönde etkileyebilir diye belirtmektedir. Bu nedenle nöron sayısı, ağın kapasitesinin problem büyüklüğüne uygun şekilde ayarlanması gerekmektedir. Bu seçim sürecinde, ayrıca kavramsal anlayış, problem özellikleri ve veri seti boyutu gibi faktörler de göz önünde bulundurulmalıdır.

Genel olarak, gizli katmanlar, yapay sinir ağının karmaşık özelliklerini öğrenmesine ve temsil etmesine olanak tanıyan önemli bir bileşendir.

Çıkış katmanı, bir sinir ağındaki nöronların son katmanıdır ve giriş verilerine ve eğitim sırasında öğrenilen ağırlıklara dayalı olarak ağın çıkış değerlerini üretmekten sorumludur. Çıkış katmanı tipik olarak bir veya daha fazla nöron içerir ve çıkış katmanındaki nöron sayısı çözülmekte olan problemin doğasına göre belirlenir. Sinir ağlarındaki katman yapısı Şekil 2.1'de gösterilmiştir.



Şekil 2.1 Sinir ağlarının katman yapısı

2.5.2 Yapay Sinir Ağlarında Ağırlıklar ve Yanlılık

Bir yapay sinir ağında ağırlıklar, eğitim sürecinde öğrenilen ve nöronlar arasındaki bağlantıların gücünü belirleyen parametrelerdir. Bu ağırlıklara başlangıçta rasgele değerler atanır ve tahmin edilen çıktı ile gerçek çıktı arasındaki hatayı en aza indirmek için eğitim süreci sırasında bir optimizasyon algoritması aracılığıyla güncellenir. Ağırlıklar, nöronların her birinin diğerini ne kadar güçlü etkilediğini belirleyen sayısal parametrelerdir.

Nielsen (2015)'e göre düğümler arasındaki her bağlantının, bir düğümün diğeri üzerindeki etkisinin gücünü belirleyen bir ağırlığı vardır. Bu ağırlıklar, tahmin edilen çıktı ile gerçek çıktı arasındaki hatayı en aza indirmeyi amaçlayan bir optimizasyon algoritması yoluyla eğitim sırasında öğrenilir. Russell ve Norvig (2010) ağırlıkları şu şekilde tanımlamıştır: Ağırlıklar, nöronlar arasındaki bağlantıların gücünü temsil eder ve eğitim sırasında ağın belirli bir girdi için istenen çıktıyı üretmesi için ayarlanırlar.

Özetle, yapay sinir ağlarındaki ağırlıklar, nöronlar arasındaki bağlantıların gücünü belirleyen ve tahmin edilen çıktı ile gerçek çıktı arasındaki hatayı en aza indirmek için bir optimizasyon algoritması aracılığıyla eğitim sürecinde öğrenilen parametrelerdir.

Yapay sinir ağlarında yanlılık (bias), her bir sinir hücresinin çıktısına eklenen sabit bir değerdir ve modelin doğruluğunu artırmaya yardımcı olur. Bu değer, hücrenin aktivasyon fonksiyonunu şekillendirmekte ve çıktıya etki etmektedir. Goodfellow, Bengio ve Courville (2016), yanlılık birçok farklı sinir ağı mimarisinde bulunan bir parametredir. Bir sinir hücresinin yanlılığı, o hücrenin çıktısına eklenir ve ağın çıktılarının doğruluğunu artırmaya yardımcı olur, şeklinde açıklamaktadır. Yanlılıklar, sinir ağının öğrenmesi sırasında diğer parametreler gibi ayarlanabilir ve genellikle rastgele başlatılırlar. Daha sonra model eğitim verilerine uygun hale gelene kadar bu değerler optimizasyon algoritmaları kullanılarak güncellenir.

2.5.3 Yapay Sinir Ağlarında Toplama Fonksiyonu

Yapay sinir ağındaki toplama fonksiyonu, ağırlıkların ve girdi değerlerinin çarpımının toplanmasıyla hesaplanır. Bu toplama işlemi, her bir girdi ve ağırlık çifti için yapılır ve sonuçlar toplam fonksiyonu olarak bir araya getirilir. Yanlılık, doğrusal bir denklemde eklenen kesişme noktası gibidir. Nörona girişlerin ağırlıklı toplamı ile birlikte çıkışı ayarlamak için kullanılan ek bir parametredir. Bu işlem, ağın her bir katmanında gerçekleştirilir ve son katmanın çıktısı üretilir.

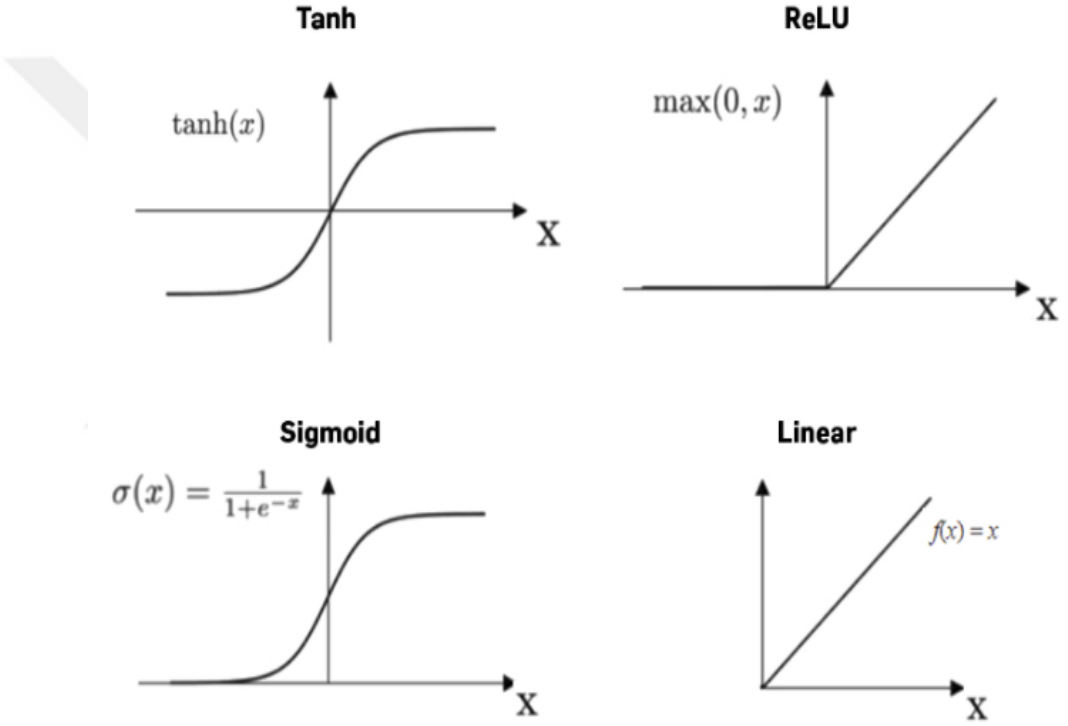
Bu işlem matematiksel olarak eşitlik (2.2)' deki gibi ifade edilir: Tipik bir nöron için girdiler x_i ve bunlara uygulanacak ağırlıklar girdi vektörü w_i ise olarak gösterilir.

$$y = \sum_i w_i x_i + b \tag{2.2}$$

2.5.4 Yapay Sinir Ağlarında Aktivasyon Fonksiyonu

Yapay sinir ağında, aktivasyon fonksiyonu, bir nöronun ağırlıklı girdilerine uygulanan doğrusal olmayan bir fonksiyondur. Aktivasyon fonksiyonunun amacı, nöronun çıktısına doğrusallık dışı bir özellik katmak ve yapay sinir ağının girdi ve çıktı arasındaki karmaşık ilişkileri öğrenmesini sağlamaktır. Aktivasyon fonksiyonu, nöronun ağırlıklı girdi toplamına uygulanan matematiksel bir formüldür ve nöronun

çıktısını belirler. Sigmoid, ReLU, hiperbolik tanjant (tanh) ve doğrusal gibi birçok farklı aktivasyon fonksiyonu yapay sinir ağlarında kullanılabilir. Nöron ve katman seçiminde olduğu gibi aktivasyon fonksiyonunun seçimi de tasarımcının denemelerine bağlı olarak belirlenebilir. Her bir fonksiyonun avantajları ve dezavantajları vardır ve farklı yapay sinir ağı tipleri ve görevleri için uygun olabilirler. Aktivasyon fonksiyonu, yapay sinir ağının önemli bir bileşenidir ve her nöronun çıktısını ve sonuçta ağın performansını belirlemede önemli bir rol oynar. Literatürde en sık kullanılan aktivasyon fonksiyonu Sigmoid fonksiyonudur.



Şekil 2.2 En sık kullanılan aktivasyon fonksiyonları

Literatürde en sık kullanılan aktivasyon fonksiyonlarının denklemsel gösterimi belirtilmiştir.

- Doğrusal fonksiyon:

$$y = f(x) = x \tag{2.3}$$

- Sigmoid fonksiyonu:

$$y = f(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

- Tanh fonksiyonu:

$$y = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.5}$$

- ReLU fonksiyonu:

$$y = \mathrm{f(x)} = \begin{cases} 0 & eğer \ \ \mathrm{x} < 0, \\ x & eğer \ \ x \geq 0 \end{cases} \tag{2.6}$$

2.6 Yapay Sinir Ağının Mimarisi

Ağ mimarisi, bir yapay sinir ağının nasıl tasarlandığını ve organize edildiğini belirler. Başka bir deyişle, katman sayısı kullanılan nöron türleri ve aralarındaki bağlantılar dahil olmak üzere ağın genel tasarımı ve organizasyonunu ifade eder. Sinir hücrelerinin birbirine nasıl bağlandığını ve bilginin ağ içinde nasıl akacağı gibi faktörleri belirleyen mimari, yapay sinir ağının yeteneklerini ve performansını etkileyen kritik bir faktördür.

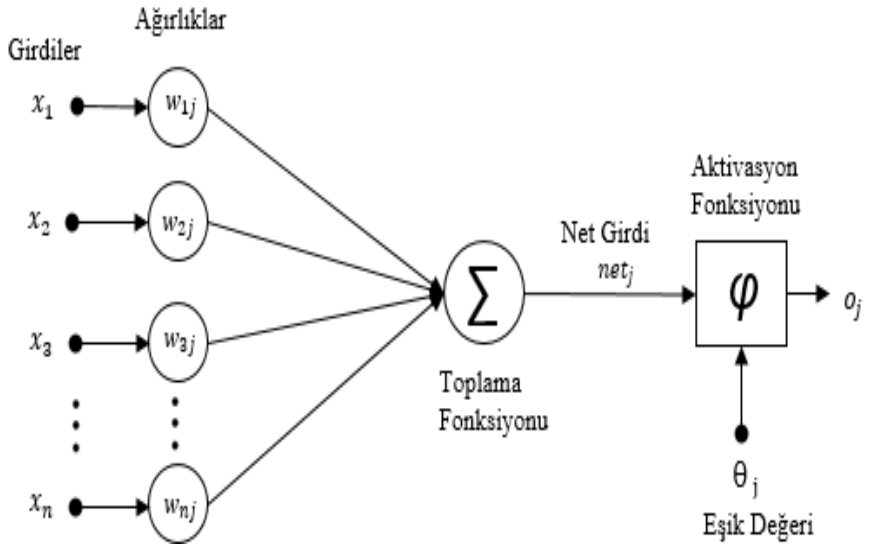
Sinir ağı mimarileri genel olarak üç farklı ağ mimarisi ile açıklanabilir: Bunlar, tek katmanlı ileri beslemeli ağlar, çok katmanlı ileri beslemeli ağlar ve tekrarlayan (recurrent) ağlar.

2.6.1 Tek Katmanlı İleri Beslemeli Ağlar

Katmanlı yapay sinir ağlarında, nöronlar katmanlar halinde düzenlenir. En basit haliyle, bir katmanlı ağın girdi katmanına doğrudan kaynak düğümlerinden gelen

veriler girilir ve çıktı katmanına da doğrudan çıktı verileri alınır. Bu ağ sadece ileriye doğru işlem yapar. Bu şekilde nöronların sadece çıktı katmanına atandığı yapılara tek katmanlı ağ denir. Girdi katmanındaki kaynak düğümlerde herhangi bir hesaplama yapılmaz.

Bu tür bir yapay sinir ağı, her biri bir girdi değişkenine bağlanan ve bu girdileri birleştiren ağırlıklarla çarpılan nöronlardan oluşur. Bu nöronlar, bir aktivasyon fonksiyonu tarafından işlenir ve tek bir çıktı elde edilir. Şekil 2.3'te bu ağın yapısı gösterilmiştir.



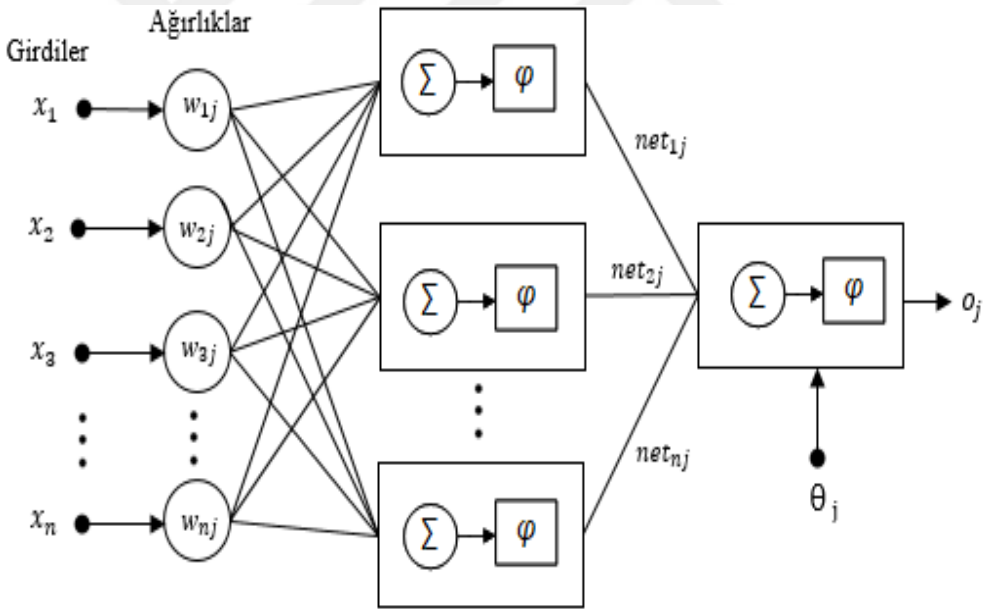
Şekil 2.3 Tek katmanlı ileri beslemeli ağların yapısı

2.6.2 Çok Katmanlı İleri Beslemeli Ağlar

Çok katmanlı ileri beslemeli ağlar, aynı zamanda çok katmanlı algılayıcılar olarak da bilinir ve yapay sinir ağlarının en yaygın türüdür. Çok katmanlı algılayıcılar da, her bir katmandaki her nöron bir sonraki katmandaki her nörona bağlıdır. Nöronlar arasındaki bağlantılar, ağ performansını iyileştirmek için eğitim sırasında ayarlanan ağırlıklara sahiptir. Gizli ve çıkış katmanındaki her nöron, girdilerinin ağırlıklı toplamına bir aktivasyon fonksiyonu uygulayarak bir çıkış sinyali üretir.

Şekil 2.4'te çok katmanlı ileri beslemeli ağların yapısı gösterilmiştir. Girdi katmanı, ağa veri girişini sağlar ve daha sonra bu girdiler, gizli katmanlardaki nöronlar tarafından işlenir. Gizli katmanlarda her nöron, önceki katmandaki tüm nöronlara bağlıdır ve bu bağlantılar ağın performansını artırmak için eğitim sırasında ayarlanan ağırlıklara sahiptir. Bu bağlantıların ağırlıklı toplamı bir aktivasyon fonksiyonuna uygulanarak her nöronun çıkışı belirlenir. En sonunda, çıkış katmanı, ağın nihai çıktısını üretir. Nöronlar arasındaki her bağlantının, ağın performansını artırmak için eğitim sırasında ayarlanan bir ağırlığı vardır.

Genel olarak, çok katmanlı ileri beslemeli ağlar, makine öğrenimi ve yapay zekadaki karmaşık sorunları çözmek için güçlü araçlardır. Bu ağlar, birden çok nöron katmanını birbirine bağlayarak, verilerdeki karmaşık kalıpları ve ilişkileri öğrenebilir ve doğru tahminler ve kararlar verebilir.



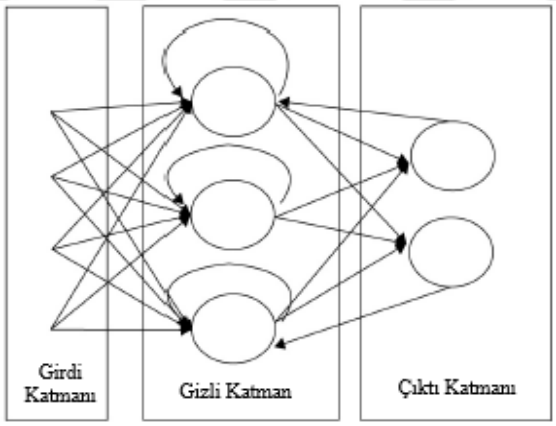
Şekil 2.4 Çok katmanlı ileri beslemeli ağların yapısı

2.6.3 Tekrarlayan Sinir Ağları

Yapay sinir ağlarında tekrarlayan sinir ağları, zaman serileri ve sıralı veriler gibi yapısal olarak bağlı verileri işlemek için tasarlanmış bir sinir ağı türüdür. İşleyişleri hakkında Goodfellow, Bengio ve Courville (2016) şöyle bir açıklama yapmışlardır:

Tekrarlayan sinir ağları, aynı ağırlıkların birden fazla adımda kullanılabildiği ve bir adımın çıktısının bir sonraki adımın girdisi olarak kullanılabildiği yapay sinir ağı türüdür.

Tekrarlayan sinir ağları, bir dizi bağlantılı nöron tabakasından oluşur. Her nöron, geçmişteki bir adıma ait girdi ve önceki adımda nöronların çıkışlarıyla birleştirilmiş bir girdi alır. Bu nedenle tekrarlayan sinir ağları, girdilerin geçmişinin bilgisini koruyabilir ve çıkışı, geçmiş girdilere ve geçmiş çıktılara dayalı olarak hesaplayabilir. Tekrarlayan sinir ağlarında, nöronlar arasındaki bağlantılar bir döngü oluşturarak dâhili bellek kullanır. Bu sayede, zaman içinde yayılan gizli katmanlar arasındaki bağlantıları içeren bir yapay sinir ağı sınıfı oluşur. Tekrarlayan sinir ağları, ileri beslemeli sinir ağlarından farklı olarak, belirli bir anda verilen kararın hemen ardından alınacak kararı da etkileyen geri besleme döngüsüne sahiptir. Bu özellikleri sayesinde Tekrarlayan sinir ağları, ileri beslemeli ağların yapamadığı görevleri gerçekleştirebilir. Şekil 2.5'te tekrarlayan sinir ağların yapısı gösterilmiştir.

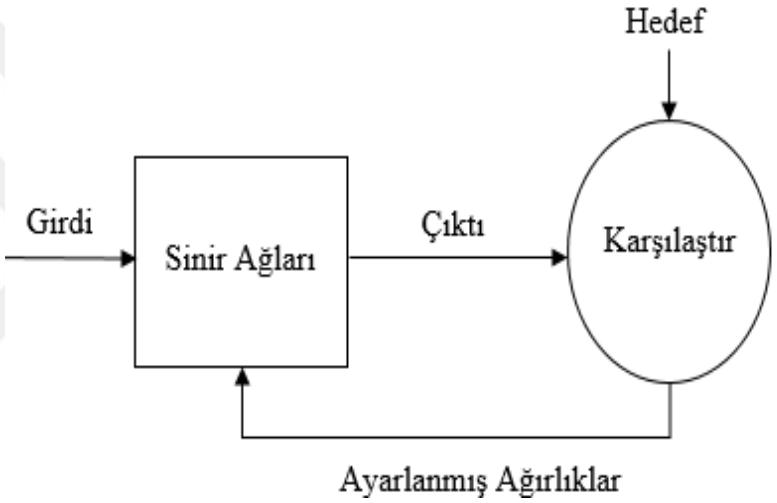


Şekil 2.5 Tekrarlayan sinir ağların yapısı

Ağın performansını artırmak için, tekrarlayan sinir ağları da diğer sinir ağları gibi nöronlar arasındaki bağlantıların ağırlıkları eğitim sırasında ayarlanır. Bu ağırlıkların güncellenmesi, geri yayılım algoritması gibi yöntemler kullanılarak yapılır. Ayrıca, tekrarlayan sinir ağları olarak kullanılan uzun kısa süreli bellek ve kapılı tekrarlayan birimler gibi özel hücre yapıları, ağın bellek mekanizmasını daha etkin hale getirmek için tasarlanmıştır.

2.7 Yapay Sinir Ağlarında Öğrenme

Yapay sinir ağları, verilerden öğrenen ve oluşturulan modeli kullanarak tahminler verme yeteneğine sahip olan bir makine öğrenimi algoritmasıdır. Evrensel bir fonksiyon yaklaşımıdır. Yapay sinir ağlarının öğrenmesi için iki temel süreç vardır: İleri besleme ve geri yayılım. İleri besleme, ağın verileri alıp işlemesi ve bir çıktı üretmesidir. Geri yayılım, ağın sonuçlarını kontrol ederek, hatalarını belirler ve bu hataları kullanarak ağın katmanlarını geriye doğru günceller. Yapay sinir ağlarında ileri ve geri yayılımın gösterimi Şekil 2.6'daki gibidir.



Şekil 2.6 Yapay sinir ağlarında ileri ve geri yayılım

2.7.1 Yapay Sinir Ağlarında İleri Yayılım

İleriye yayılma bir girdi alma, onu bir sinir ağından geçirme ve bir çıktı alma işlemidir. Sinir ağının her katmanı, girdiye bir dizi ağırlık uygular ve sonucu bir çıktı üretmek için bir aktivasyon fonksiyonundan geçirir ve bir sonraki katmana aktarılır (Goodfellow vd., 2016).

İleri yayılım süreci, temelde bir girdi verilen bir sinir ağının çıktısını hesaplama sürecidir. Ağın çıktısı, modelin hatasını veya kaybını hesaplamak için

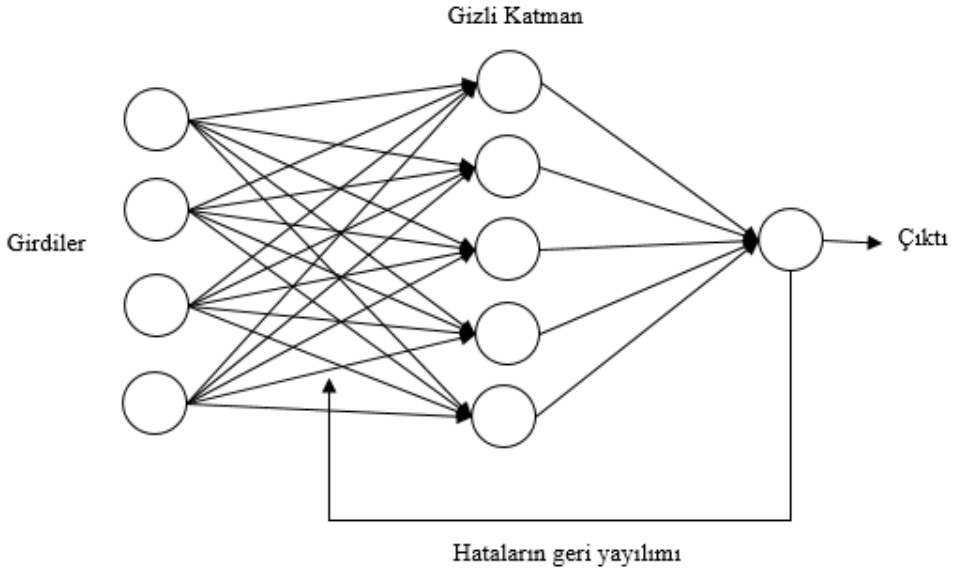
kullanıldığından, sinir ağı eğitim sürecinin çok önemli bir parçasıdır, daha sonra ağın ağırlıklarını güncellemek için geri yayılım sürecinde kullanılır (Géron, 2019).

Yapay sinir ağındaki ileri yayılım sürecinin adım adım açıklaması aşağıdaki gibidir:

1. Giriş verileri, ağın giriş katmanına beslenir.
2. Giriş katmanındaki her bir nöron, birinci gizli katmandaki her bir nörona bağlıdır ve her bağlantının kendisiyle ilişkili bir ağırlığı vardır.
3. Giriş verileri, her bir bağlantıyla ilişkili ağırlıklarla çarpılır ve sonuçlar, birinci gizli katmandaki her bir nöron için bir çıkış değeri üretmek üzere toplanır.
4. Birinci gizli katmandaki her bir nöron, giriş değerine bir aktivasyon fonksiyonu uygulayarak, bir sonraki katman için girdi haline gelen bir çıkış değeri üretir.
5. Adım 2-4 ağdaki sonraki her katman için tekrarlanır ve bir önceki katmanın çıktısı bir sonraki katmanın girdisi olur.
6. Nihai çıktı değeri, ağın çıktı katmanı tarafından üretilir.

2.7.2 Yapay Sinir Ağlarında Geri Yayılım

Geri yayılım, bir yapay sinir ağındaki ağırlıkların, ağın tahmin edilen çıktısı ile gerçek çıktısı arasındaki farka dayalı olarak güncellenmesi işlemidir. Bu işlem, eğitim sırasında ağın performansını optimize etmek için kullanılır. İleri beslemeli bir sinir ağındaki öğrenme süreci aslında geri yayılım aşamasında gerçekleşir. Geri yayılım Şekil 2.7'de gösterilmiştir.



Şekil 2.7 Yapay sinir ağlarında hataların geri yayılım süreci

Yapay sinir ağındaki geri yayılım sürecinin adım adım açıklaması aşağıdaki gibidir:

- Geri yayılımın ilk adımı ileri yayılımdır. İleri yayılım sırasında, girdi sinir ağına beslenir ve çıktı, girdiyi ağ katmanı boyunca katman katman yayarak hesaplanır. Her katman, girdiye bir dizi ağırlık uygular, ardından doğrusal olmayan bir aktivasyon işlevi uygular ve sonucu bir sonraki katmana verir.
- İleri yayılımdan sonra, tahmin edilen çıktı ile gerçek çıktı arasındaki farkı hesaplanır. Bu fark, çözmeye çalışılan belirli soruna bağlı olan bir kayıp fonksiyonu kullanılarak ölçülür.
- Geriye geçiş sırasında, her katmandaki ağırlıklara göre kayıp fonksiyonunun gradyanları hesaplanır. Bu gradyanlar daha sonra stokastik gradyan iniş gibi bir optimizasyon algoritması kullanılarak ağırlıkları güncellemek için kullanılır.
- Son adım, geriye doğru geçişte hesaplanan gradyanlara dayalı olarak ağdaki ağırlıkları güncellemektir. Ağırlıkların güncellenme miktarı, gradyan yönünde atılan adım boyutunu belirleyen bir hiperparametre olan öğrenme oranı tarafından belirlenir (Géron, 2019).

Sinir ağı öğrenimi, ileri beslemeli ağlarda büyük ölçüde geri yayılıma dayanır. İleri yayılımın ve hata düzeltmenin adımları aşağıda açıklanmıştır (Haykin,2009).

1. Rastgele ağırlıklar w_j atanarak sinir ağı ileri yayılımını başlar ve gizli katmandaki nöronların her birine sapma değerleri b_j atanır.

2. Her bir nörondaki toplam net girdi hesaplanır. Eşitlik 2.7'de gösterildiği gibi net_j, j. nöronun ağırlıklı giriş toplamıdır, w_{ij} i. girdinin j. nörona olan ağırlığıdır, x_i ise i. girdidir ve b_j j. nöronun yanlılık değeridir.

$$net_j = \sum_j w_{ij} x_i + b_j \tag{2.7}$$

3. Her nöronda aktivasyon fonksiyonu uygulanır. Literatürde genellikle sigmoid fonksiyonu denklemi (2.8)'deki gibidir, burada y_j j. nöronun çıkışıdır.

$$y_j = \frac{1}{1 + e^{-net_j}} \tag{2.8}$$

4. Bu çıktılar bir sonraki katmanın nöronuna iletilir. Bu adımda, önceki katmandaki nöronların çıktıları, bir sonraki katmandaki nöronların girdisi olarak kullanılır.

5. Eğer katman çıktı katmanı ise, ağırlıklı toplam alınır. Eşitlik (2.9)'da net_k çıkış nöronunun ağırlıklı giriş toplamıdır, w_{jk} j. nöronun k. çıkışa olan ağırlığıdır, y_j ise j. nöronun çıkışıdır ve b_k k. çıkış nöronunun yanlılığıdır.

$$net_k = \sum_j w_{jk} y_j + b_k \tag{2.9}$$

6. Yine çıkış katmanı nöronunda aktivasyon fonksiyonu uygulanır. Eşitlik (2.10)' da gösterilmiştir ve y_k k. çıkış nöronunun çıkışıdır.

$$y_k = \frac{1}{1 + e^{-net_k}} \tag{2.10}$$

7. Gerçek çıktıyı ve aktivasyon fonksiyonu çıktısını çıkararak hata terimi hesaplanır. Eşitlik (2.11)'de e_k, k. çıkış nöronunun hata terimidir, d_k ise k. çıkış nöronunun gerçek çıktısıdır.

$$e_k = d_k - y_k \tag{2.11}$$

8. Hataların toplamına ulaşmak için tüm çıkış nöronlarının hata terimleri toplanır. *E* ağın genel hatasıdır. Hata fonksiyonu *E* sonradan türevlendiğinde üssü iptal etmek için 1/2 faktörü kullanılır.

$$E = \frac{1}{2}(d_k - y_k)^2 \tag{2.12}$$

9. Gradyan iniş algoritması, bir yapay sinir ağının hatalarını en aza indirmek için kullanılan bir optimizasyon algoritmasıdır. Bu algoritma, ağırlıkların doğru şekilde ayarlanmasını sağlar ve ağın daha doğru sonuçlar üretmesini sağlar. Bu algoritma, ağın ağırlıklarını güncellemek için türev zincir kuralını kullanır. Ağırlıkların güncellenmesi, hatayı en aza indirmek için eğimin ters yönünde yapılır. Bu işlem, hata fonksiyonunun en küçük değerine ulaşıncaya kadar tekrarlanır.

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}} \tag{2.13}$$

10. Bu üç türev çarpıldıktan sonra öğrenme oranı ile birlikte ağırlıklar güncellenir. Eşitlik (2.14)' teki ifade gradyan iniş algoritmasında ağırlıkların güncellenmesini gösterir. Burada yeni ağırlık, önceki ağırlığa öğrenme oranı ile çarpılmış hata teriminin (E) w_{jk}'ye göre kısmi türevi çıkarılarak elde edilir. Yani, ağırlıkların güncellenmesi için hata teriminin ağırlıklara göre kısmi

türevi hesaplanır ve öğrenme oranı ile çarpılarak eski ağırlıktan çıkarılır. Bu şekilde ağırlıklar, hatanın azaltılmasına yönelik olarak güncellenir.

$$w_{(yeni\ ağırlık)} = w_{(yeni\ ağırlık)} - \text{Öğ}renmeOranı\ \frac{\partial E}{\partial w_{jk}} \qquad (2.14)$$

BÖLÜM ÜÇ
TOPLULUK ÖĞRENME YÖNTEMLERİ

Topluluk öğrenme, birden fazla öğrenme modelinin bir araya getirilerek daha iyi bir tahmin performansı elde edilmesi amacıyla kullanılan bir makine öğrenmesi tekniğidir. Topluluk öğrenme yöntemleri, bileşen modellerden elde edilebilecek herhangi bir tahminden daha iyi tahmin edici performans elde etmek için birden fazla model kullanırlar (Dietterich, 2002). Topluluk öğrenme yöntemleri çoğunluk oyu alarak veya tahminleri birleştirmek için bir meta-model kullanarak model tahminlerini çeşitli şekillerde birleştirerek çalışır.

Topluluk yöntemleri, sonuç modellerin kararlılığını ve genelleme yeteneğini artırabilir ve eğitim verilerine aşırı uyum riskini azaltabilir (Kuncheva, 2014). Birden fazla modelin birleştirilmesiyle, topluluk öğrenmesi, bireysel modellerdeki rasgele hataların veya yanlılığın etkisini azaltarak daha doğru ve güvenilir tahminlere yol açabilir.

Topluluk öğrenmesi, makine öğrenmesinde popüler bir teknik haline gelmiş olup, torbalama (bagging), güçlendirme (boosting) ve birleştirme (stacking) gibi çeşitli topluluk öğrenme yöntemleri bulunmaktadır. Breiman (2001), Random Forests adlı makalesinde tanımladığı gibi, torbalama yöntemi eğitim verilerinin farklı alt kümelerini kullanarak birden fazla model oluşturmayı ve tahminlerin ortalamasını veya çoğunluk oyunu alarak birleştirmeyi içerir. Güçlendirme (boosting) yöntemi, zayıf öğrenenleri sırayla eğiterek, her sonraki model ile önceki modelin hatalarını düzeltmeye çalışır (Freund ve Schapire, 1999). Birleştirme (stacking) yöntemi ise, birden fazla model eğiterek tahminlerini birleştirmek için bir meta-model kullanmayı içerir (Wolpert, 1992).

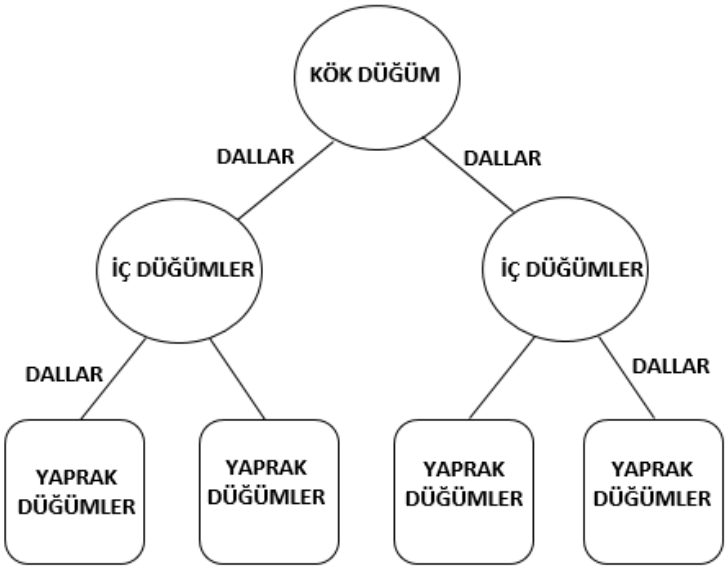
Bu çalışma da topluluk öğrenme yöntemleri modelleri olarak regresyon ağacı, torbalama yöntemi ile regresyon ağaçları, rassal ormanlar, gradyan artırma regresyon ağaçları ve xgboost algoritmaları incelenmiştir.

3.1 Regresyon Ağaçları

Veri bilimi kapsamında son zamanlarda en performanslı tahmin başarısı olan algoritmalar ağaca dayalı yöntemler, yapay sinir ağları ve çok yapılı optimize edilmiş karar ağaçlarıdır. Karar ağaçları, hem sınıflandırma hem de regresyon problemleri için kullanılabilen denetimli makine öğrenimi algoritmasıdır.

Bir karar ağacının bileşenleri Şekil 3.1'de gösterilmiştir, Şekil 3.1 incelendiğinde bileşenler aşağıdaki gibi açıklanabilir.

- Kök Düğüm: Ağacın en üst düğümüdür. Tüm veriler burada toplanır ve bir girdi özelliğine göre bölünür.
- İç Düğümler: Kök düğüm dışındaki her düğüm, bir girdi özelliği ve eşik değeri ile tanımlanır. Veriler bu düğümlerde alt kümeler halinde bölünür. İç düğümler alt kümelere ayrılabilir.
- Yaprak Düğümler: Ağacın en alt düğümleri, yani alt dallara ayrılmayan son düğümlerdir. Bu düğümler, bir tahmin değerini veya sınıflandırma sonucunu temsil eder.



Şekil 3.1 Bir karar ağacının yapısı

Karar ağaçlarındaki amaç veri seti içerisindeki karmaşık yapıları basit karar yapılarına dönüştürmektir. Hastie vd. (2009) tarafından açıklandığı gibi, regresyon ağaçları girdi alanını bir dizi dikdörtgen bölgeye böler ve her bölgedeki yanıt değişkeni varyansını en aza indirir. Bir diğer ifade ile heterojen veri setini belirlenmiş bir hedef değişkene göre homojen alt gruplara ayırır. Bir veri setini bazı karar kriterleri işleterek alt parçalara bölerek dallandırmaktadır. Algoritma şöyle çalışır:

1. Tüm veri kümesi, ağacın kök düğümü olarak başlar.
2. Her girdi özelliği için, hedef değişken değerlerindeki varyans azaltımını maksimize eden en iyi ayrım noktası hesaplanır.
3. En iyi ayrım noktasına sahip girdi özelliği, mevcut düğümün karar kuralı olarak seçilir.
4. Karar kuralına göre veriyi iki alt kümeye böler ve mevcut düğüme iki alt düğüm oluşturur.
5. Her alt düğümler için, adım 2-4 tekrarlanarak, belirli bir durma kriteri (örneğin, yaprak düğümlerindeki minimum örnek sayısı) karşılanana kadar, alt düğümlerin alt kümeleri oluşturulur.

Regresyon ağaçlarının temel ilkeleri ve yapısı ve ilkeleri, eğitim ve test verisi olarak ayırma, yinelemeli bölümleme, bölünme kriterleri, durdurma kriterleri, budama ve tahmin şeklindedir.

Veri seti eğitim ve test verisi olarak ayrılır ve eğitim seti; regresyon ağacını oluşturmak test seti; performansını değerlendirmek için kullanılır. Regresyon ağacı, yinelemeli bir bölümleme algoritması kullanılarak oluşturulur. Regresyon ağacı yöntemini Breiman vd. (1984) , bir veri kümesini giderek daha küçük alt kümelere bölen ve aynı zamanda ilişkili bir karar ağacının artımlı olarak geliştirildiği yinelemeli bir bölümleme prosedürüdür şeklinde açıklamıştır. Başka bir deyişle algoritma, tüm gözlemleri içeren kök düğümde başlar ve tahmin değişkenlerine dayalı olarak verileri yinelemeli olarak daha küçük alt kümelere böler. Bu yinelemeli bölümlendirme işlemi, en büyük derinlik, yaprak düğümündeki minimum gözlem sayısı veya yanıt değişkeninin varyansında belirgin bir iyileşmenin olmaması gibi bir durdurma kriteri

elde edilene kadar devam eder. Amaç, her bir alt küme içindeki yanıt değişkeninin homojenliğini maksimize eden, en iyi ayrımı bulmaktır.

Regresyon ağaçlarında, veri kümesinin en iyi ayrımını sağlayan özellik seçilerek kök düğümü belirlenir. Bunun için, her bir özellik değerinde hedef değişkenin belirsizliği ölçülür ve en düşük belirsizliğe sahip özellik seçilir. Bir regresyon ağacı oluşturmanın ana fikri, eğitim kümesini tekrarlı olarak daha küçük alt kümelerine ayırmak ve aynı zamanda her alt küme için basit bir model uydurmaktır. Her bir düğümde, hedef değişkenin hata kareler ortalamasını en aza indirecek özelliği ve ayrım noktası seçilir (Müller ve Guido, 2016). Regresyon ağaçlarında kök düğümü, hedef değişkenin hata kareler ortalamasını en aza indirecek özelliğe göre seçilir.

Bölünme kriterleri olarak varyansta azalma, hata kareler toplamı ve ortalama mutlak sapma dahil olmak üzere her adımda en iyi ayrımı belirlemek için kullanılabilecek birkaç kriter vardır. En yaygın olarak kullanılan kriter, bölünmenin bir sonucu olarak yanıt değişkeninin varyanstaki azalımıdır (Hastie vd., 2009). Öte yandan, belirsizlik ölçüleri Ki-Kare, Gini ve Entropi, sınıflandırma problemleri için kullanılır. Bu ölçüler, kök düğümden itibaren her bir dallanmanın ayrılmalarını belirlemek için kullanılır. Ki-Kare ölçüsü, herhangi bir değişkenin hedef değişkendeki bağımsızlığını ölçer. Gini ve Entropi ölçüleri, verilerin homojenliğini ölçer ve bir dalın daha homojen olup olmadığını belirlemek için kullanılır. Bu belirsizlik ölçüleri sınıflandırma problemleri için kullanılırken, varyans azalması, hata kareler toplamı ve ortalama mutlak sapma regresyon problemleri için kullanılır.

Kuhn ve Johnson (2013) ‘a göre yinelemeli bölümleme işlemi, minimum düğüm boyutu veya maksimum ağaç derinliği gibi bazı durdurma kriterleri karşılanana kadar devam eder. Yinelemeli bölümleme işlemi, bazı durdurma ölçütleri sağlanana kadar devam eder. En yaygın durdurma kriterleri, maksimum ağaç derinliği, yaprak düğümü başına minimum gözlem sayısı ve bölme kriterinde minimum iyileştirme ve karmaşıklık parametresidir. Bu kriterin arkasındaki mantık, aşırı derecede karmaşık ağaçların verilere gereğinden fazla uyması ve bunun sonucunda zayıf genelleme performansına yol açmasıdır.

Regresyon ağacı oluşturulduktan sonra, fazla uyumu önlemek için budama yöntemi kullanılır. Ağaçların çok dallanıp budaklanması veriyi temsil etme kabiliyeti yükseldikçe, eğitim seti üzerinde çok güzel öğrenebilmekte ancak test verisinde tahmin başarısını düşürmektedir, bu durum aşırı öğrenme (overfitting) sorununa yol açmaktadır. Budama, bir doğrulama veri kümesinde ağacın performansını önemli ölçüde artırmayan dalların çıkarılmasını içeren bir ağaç basitleştirme işlemidir. (Loh, 2011). En yaygın budama algoritması, daha basit ağaçları desteklemek için bölme kriterine bir ceza terimi eklemeyi içeren maliyet karmaşıklığı budamasıdır. Breiman 1984 yılında modeli oluşturup karmaşıklık parametresini belirlemiştir. Bu parametre özellikle dallanma işleminin nerede duracağını belirleyen bir parametredir. Belirlenen bir hata metriğinin altında düşüş olup olmamasına göre kendisini ayarlayıp bu dallanma işleminin de kesilmesini sağlamıştır. Budama işlemi aşırı öğrenmeyi azaltarak daha genelleştirilebilir modeller elde etmemizi sağlar. Bazı durumlarda veri setindeki örneklerin sayısı az olduğunda budama işlemi daha kötü sonuçlar verebilir. Bu genellikle, modele uygulanan budama işleminin ağacın yapısını çok fazla değiştirmesi veya ağacın yeterince büyük olmaması durumunda görülür. Eğer veri setinde çok fazla gürültü veya değişkenlik varsa, ağacın budanması doğru bir seçim olabilir. Ancak, veri setinde yeterli sayıda örnek varsa ve ağacın yapısı özellikle karmaşık değilse, budama işlemine gerek olmayabilir.

Son olarak, regresyon ağacı yeni veriler üzerinde tahmin yapmak için kullanılır. Bir dizi öngörü değişkeni verildiğinde, algoritma, yanıt değişkeninin değerini tahmin etmek için ağaç tarafından tanımlanan karar kurallarını izler. Regresyon ağaçlarının çalışma prensibi özetle aşağıdaki gibidir;

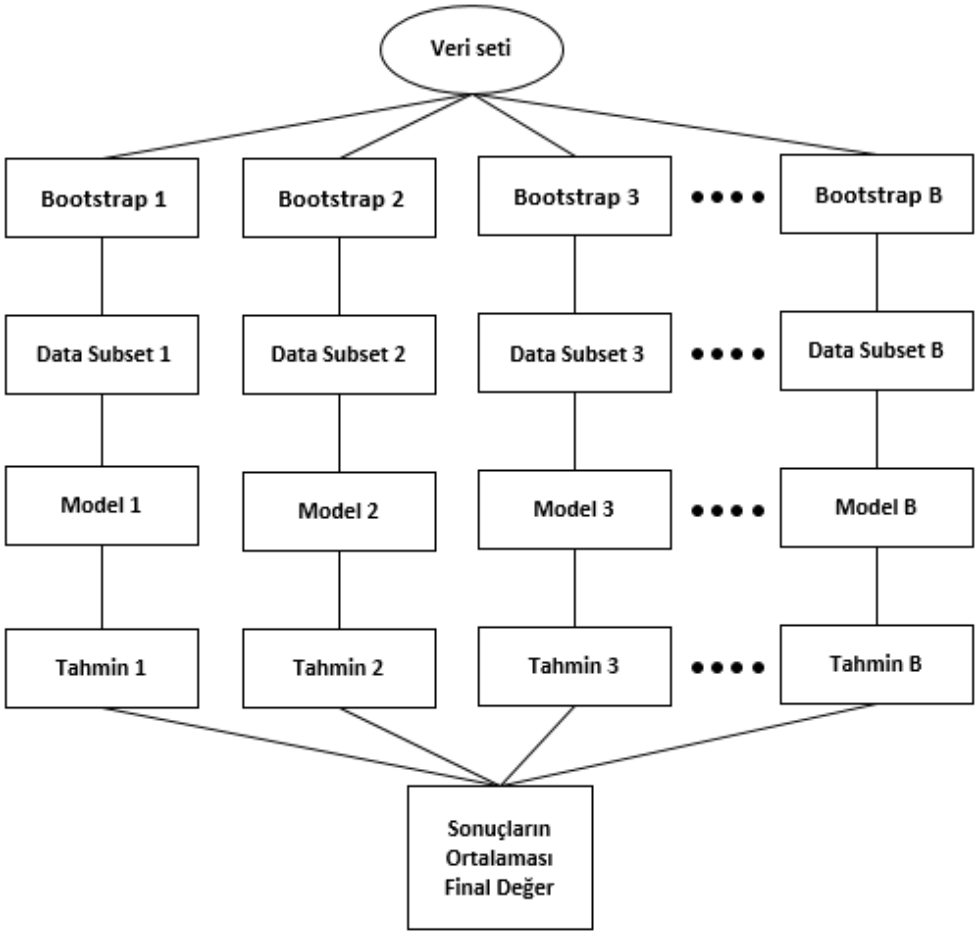
- Heterojen verisetleri belirlenmiş bir hedef değişkene göre homojen alt gruplara ayrılır.
- Alt gruplara ayırma işlemi birtakım karar verme kuralları ile gerçekleştirilir.
- Dallanma kökten yapraklara doğru bölünme kriterleri uygulanarak gerçekleşir. İlk değişken en çok etkisi olan ve sonrası şeklinde işlemler sürdürülür.

3.2 Torbalama Yöntemi ile Regresyon Ağaçları

Topluluk öğrenme yöntemleri, birden fazla öğrenme modelini birleştirerek daha iyi bir performans elde etmeyi amaçlar. Bu yöntemler arasında torbalama (bagging) yöntemi, regresyon ağacı kullanarak yapılan tahminleri birleştiren ve varyansı azaltan bir yöntemdir. Breiman 1996 yılında bagging yöntemi ile regresyon ağaçlarını ortaya çıkarmıştır. Breiman'ın ifadesiyle, torbalama, varyansı düşürmenin en etkili yolu gibi görünmektedir (Breiman, 1996). Bagging yöntemi, bootstrap aggregation kısaltmasıdır. Yani yerine koymalı örneklemleri birleştirme anlamına gelmektedir. Bu yöntem, her alt örneklemde veri setinden tekrar tekrar örneklem alma işlemi yaparak oluşturulur. Bu sayede, aynı örneklemin birden fazla alt örnekleme dahil edilmesi mümkün olur. Torbalama yöntemi ile regresyon ağacı, veri setinden birden fazla alt örneklem (bootstrap örneklem) oluşturarak, her bir örneklem için ayrı bir regresyon ağacı eğitir. Wu vd. (2019) tarafından belirtildiği gibi, torbalama regresyon ağacı yöntemi şu şekilde özetlenebilir: N gözlem ve M özellikten oluşan bir eğitim seti verildiğinde, torbalama yöntemi ile regresyon ağacı algoritması eğitim verilerinin B bootstrap örneğini oluşturur, her örneğe bir regresyon ağacı uydurur ve tüm ağaçların tahminlerinin ortalamasını alır.

Bu yöntemin bir avantajı, çok sayıda özellikle yüksek boyutlu verilerle başa çıkabilmesidir. Zhang vd. (2018) ‘e göre, özellik sayısı çok büyük olduğunda, bagging regresyon ağacı algoritması, ağaç oluşturma sürecine özellik alt örneklemesi getirerek daha iyi bir tahmin performansı elde edebilir. Başka bir avantajı, bağımsız değişkenler ile yanıt değişkeni arasındaki doğrusal olmayan ilişkilerle başa çıkabilmesidir. Breiman'a (1996) göre, bagging regresyon ağaçları, etkileşimler de dahil olmak üzere bağımsız değişkenlerle yanıt değişkeni arasındaki karmaşık doğrusal olmayan ilişkileri yakalayabilir, doğrusal olmayan verilerde etkilidir ve birçok ağacın ortalaması gürültü etkilerini azaltır. Bu yöntemin dezavantajı ise dengesiz veri kümeleri veya aykırı değerlerle başa çıkmakta iyi performans göstermeyebilir. Li vd. (2017) şöyle belirtmiştir: Veri dengesiz olduğunda, bagging yöntemi çoğunluk sınıfına doğru yanlı olabilir ve kötü performansa neden olabilir. Ayrıca, eğer veri aykırı değerler içeriyorsa, ağaç bunlara aşırı uyum sağlayarak tahminlerin yanlış olmasına neden olabilir.

Bagging regresyon ağacı algoritmasında şu adımlar izlenir ve algoritma yapısı Şekil 3.2'de gösterilmiştir:



Şekil 3.2 Bagging regresyon ağacı yapısı

- Veri Örnekleme: Bagging algoritmasında, verilerin bir kısmı rastgele seçilen örneklemlere bölünür. Bu örneklemler "bootstrap örneklemleri" olarak adlandırılır. Bu adım, rastgele örnekleme yapmak için birçok kez tekrarlanır ve her bir örnekleme için bir adet regresyon ağacı oluşturulur.
- Regresyon Ağacı Oluşturma: Her bootstrap örneklemindeki veri seti, ayrı ayrı bir regresyon ağacı oluşturmak için kullanılır. Regresyon ağacı, bağımsız değişkenlerin kullanılmasıyla veri setindeki hedef değişkeni tahmin eden bir yapıdır. Regresyon ağacı, veri setini rastgele bir şekilde bölerek her bir alt

kümeye bir karar ağacı uygular ve böylece veri setindeki yapısal özellikleri anlar. Model sayısı, bootstrap örneklemleriyle oluşturulacak olan farklı veri kümelerinin sayısına eşittir.

- Tahmin: Oluşturulan her bir regresyon ağacı, eğitim verilerindeki hedef değişkenini tahmin etmek için kullanılır. Her ağaç, bağımsız değişkenlerin kullanımıyla hedef değişkeni tahmin eder.
- Sonuçların Ortalaması: Tüm regresyon ağaçlarının tahminleri toplanır ve sonuçlarının ortalaması alınarak bir tahmin değeri elde edilir. Bu, bagging algoritmasının temel prensibidir ve hedef değişkeni tahmin etmek için daha doğru bir sonuç sağlar. Bu adım, her bir bootstrapped örnekleminde oluşturulan tüm ağaçların tahminlerinin ortalaması alınarak gerçekleştirilir.
- Test Etme: Bagging algoritması, tahminlerin doğruluğunu ölçmek için bir test veri kümesi kullanır.
- Sonuçların Değerlendirilmesi: Bagging algoritması sonuçlarının doğruluğu, test veri kümesindeki tahminlerin gerçek değerleriyle karşılaştırılarak değerlendirilir. Bu, algoritmanın doğruluğunu ölçmek için kullanılan bir yöntemdir.

3.3 Rassal Ormanlar Regresyonu

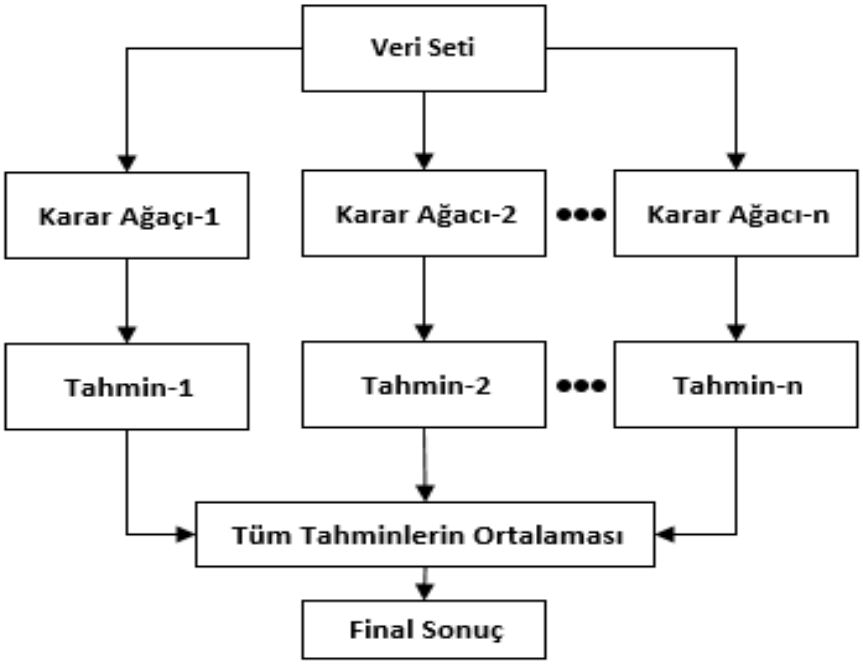
Temeli birden çok karar ağacının ürettiği tahminlerin bir araya getirilerek değerlendirilmesine dayanır. Rassal ormanlar regresyonu farkını gözlem ve değişken seçiminde rastgele seçim yapmasından alır. Breiman 2001 yılında rassal ormanları da ortaya koyarak yine bu ağaca dayalı yöntemler konusunda ciddi bir ilerlenme sağlanmasına sebep olmuştur. Breiman 1996 yılında gözlem seçiminin yanlılığı ve hatayı azalttığı rassalığı koruduğu bilgisini ortaya çıkarmıştı, sonrasında gözlem seçimini de kullanarak başarılı sonuçlar elde edince değişkenlere de uyarladı ve değişkenlere de uyarlayınca başarılı sonuçlar elde edebilmiştir.

Rassal ormanlar regresyonunun özelliklerinden biri bagging ve random subspace (rastgele alt küme) yöntemlerinin birleşimi ile oluşmasıdır. Random subspace değişken seti arasından rastgele bir alt küme seçmek anlamına gelir. Bu yöntem, aşırı öğrenme (overfitting) sorununu azaltmaya yardımcı olur ve özelliklerin seçilmesini ve

özellikler arasındaki etkileşimi araştırmayı kolaylaştırır. Ağaçlar için gözlemler bootstrap rastgele örneklem seçimi yöntemi ile değişkenler random subspace yöntemi ile seçilir.

Karar ağacının her düğümünde en iyi dallara ayırıcı değişken tüm değişkenler arasından rastgele seçilen daha az sayıdaki değişken arasından seçilir. Rassal ormanlar regresyonu yönteminde, toplam özellik sayısı üçe bölünerek elde edilen sayıda değişken seçilir. Bu yöntemin rastgelelik özelliği yüksektir. Büyük hacimli ve fazla değişkenli veri setlerinde yüksek performans gösterir. Rassal ormanlar yöntemi, bagging regresyon ağacına göre genellikle daha düşük bir varyans sağlar. Rassal ormanlar yöntemi, küçük veri setleri için daha karmaşık olabilir ve aşırı uyum (overfitting) riski daha yüksek olabilir (Breiman, 2001).

Rassal ormanlar algoritmasında aşağıdaki adımlar izlenir ve algoritma yapısı Şekil 3.3'de gösterilmiştir:



Şekil 3.3 Rassal ormanlar regresyonu yapısı

- Veri Hazırlama: Regresyon için rastgele orman algoritmasının uygulanmasındaki ilk adım, uygun bir veri kümesi seçmektir. Veri seti, hedef değişkeni ve hedef değişkeni tahmin etmek için kullanılacak bağımsız değişkenleri (özellikleri) içermelidir. Modeli oluşturmadan önce eksik değerlerin, aykırı değerlerin ve alakasız değişkenlerin ele alınması gibi ön işleme adımları da gerçekleştirilmelidir (Khandelwal vd., 2018).
- Eğitim-Test Bölünmesi: Veri seti, eğitim ve test setlerine ayrılır.
- Özellik Seçimi: Rastgele orman regresyonu, modeli oluşturmak için kullanılacak en önemli özellikleri seçmek için özellik seçimi adı verilen bir teknik kullanır. Bu, eğitim veri kümesindeki her bir özelliğin hedef değişken üzerindeki etkisinin değerlendirilmesini içerir. Regresyon yönteminde toplam özellik sayısı 3'e bölünerek elde edilen sayıda özellik seçilir.
- Çoklu Karar Ağaçları Oluşturma: Rastgele orman regresyonu, eğitim veri kümesinin farklı alt kümelerinden birden çok karar ağacı oluşturmayı içeren bir teknik olan torbalama kullanarak birden çok karar ağacı oluşturur. Karar ağaçlarının tahmin edilen değerleri daha sonra nihai tahmini yapmak için birleştirilir. Her karar ağacı, verilerin rastgele bir örneği ve özelliklerin rastgele bir örneği kullanılarak oluşturulur. Bu, her ağacın farklı olmasını ve modelin verilere fazla uymamasını sağlar (Khandelwal vd, 2018).
- Ayırma Kriterleri: Rastgele orman modelini oluşturan karar ağaçlarında her bir düğüm için bölme kriterlerinin de belirlenmesi gerekmektedir. Bu, verileri her düğümde bölmek için bir yöntem seçmeyi içerir. Örneğin, verileri homojen gruplara ayırmak için ayırma kriterleri olarak varyansta azalma, karesel hataların toplamı ve ortalama mutlak sapma gibi kriterler kullanılabilir.
- Tahmin: Rastgele ormandaki her karar ağacı, seçilen özelliklere ve kullanılan bölme kriterlerine dayalı olarak hedef değişkenin bir tahminini sağlar. Son tahmin daha sonra tüm tahminlerin ortalaması olarak hesaplanır (Biau ve Scornet, 2016).

Özet olarak, rastgele orman regresyon algoritması, verilerin hazırlanmasını, eğitim ve test kümelerine bölünmesini, en önemli özelliklerin seçilmesini, çoklu karar ağaçlarının oluşturulmasını, bölme kriterlerinin belirlenmesini ve ardından tüm

tahminlerin ortalamasını alarak hedef değişkenin tahmin edilmesini içerir. Değişkenler arasındaki karmaşık ilişkileri modellemek için kullanılan etkili ve yaygın bir yöntemdir. Özellikle büyük veri kümeleri, yüksek boyutlu özellik uzayları ve doğrusal olmayan ilişkilerle uğraşırken oldukça faydalıdır.

3.4 Gradyan Artırma ile Regresyon Ağaçları

Gradyan artırma ile regresyon ağaçları (gradyan boosting regresyonu), regresyon ve sınıflandırma görevleri için kullanılan bir makine öğrenimi algoritması türüdür. Gradyan arttırma makinesi algoritması (GBM), zayıf modelleri bir araya getirerek doğru tahminler yapan güçlü bir model oluşturmak için tekrarlayan şekilde çalışırlar.

Gradyan artırma regresyon ağaçlarının en önemli avantajlarından biri, yüksek doğrulukta tahminler üretebilmeleridir. GBM' lerin mantığı, toplu öğrenme kavramına dayanmaktadır. Hastie, Tibshirani ve Friedman (2009) açıkladığı gibi, topluluk öğrenme yöntemleri, herhangi bir tek model üzerinden daha iyi performans elde etmek için birden fazla modeli birleştirir. Gradyan artırma regresyonunun temel fikri, her bir modelin önceki modelin tahminlerinin hatalarına odaklanmasıdır. Bu nedenle, her bir model önceki modelin hatalarını azaltmaya çalışır. Bu yaklaşım, hatayı minimize eden bir toplam model oluşturur ve bu da daha iyi bir tahmin performansı sağlar.

Gradyan arttırma regresyonunun bir diğer avantajı, geniş bir veri türü yelpazesini işleyebilmeleri ve diğer makine öğrenimi algoritmalarına kıyasla aşırı uyum sorununa daha az duyarlı olmalarıdır. Ek olarak, GBM'ler değişken seçimi yapabilir, bu da doğru tahminler için gereken özellik sayısını azaltmaya yardımcı olabilir.

Gradyan arttırma regresyonu, gradyan iniş algoritmasını birkaç adımda kullanır. Bu adımlardan biri, hata hesaplama ve gradyan inişi kullanarak yeni bir modelin uyumunu iyileştirmektir. Gradyan arttırma regresyonu şu adımlar izler:

- Veri kümesinin bölünmesi: Önceki makine öğrenimi tekniklerinde olduğu gibi, veri kümesi öncelikle eğitim ve test veri setleri olarak bölünür. Eğitim veri seti,

modelin öğrenmesi gereken verileri içerirken, test veri seti, modelin performansını değerlendirmek için kullanılır.

- Başlangıç tahmininin yapılması: GBM, başlangıç tahminini yapmak için bir basit model (genellikle ortalama veya medyan değer) kullanır. Belirli bir hedef fonksiyonunu minimize etmeye çalışır.

$$F_0(x) \quad = a\text{rgmin} \ (\sum_{i=1}^{n} (\mathrm{L}(\mathrm{y_i}, \mathrm{c})) \tag{2.15}$$

- Hataların hesaplanması: Her bir modelin öğrenme algoritması, başlangıç tahmininden sonra hataların hesaplanmasına dayanır. Bu hatalar, gerçek değerler ve tahmin edilen değerler arasındaki farktır. Hata fonksiyonu, tahmin edilen değerlerin gerçek değerlerden ne kadar farklı olduğunu ölçer.
- Hataların ağırlıklandırılması: Hataların hesaplanması sonrasında, her bir örnek için bir hata ağırlığı hesaplanır. Bu ağırlıklar, daha sonraki modelin öğrenme algoritmasının, daha fazla dikkat edilmesi gereken hatalara odaklanmasına yardımcı olur.
- Bir sonraki modelin eğitimi: Hata ağırlıklarının belirlenmesinin ardından, bir sonraki model öğrenme algoritmasıyla eğitilir. Bu öğrenme algoritması, daha önceki modellerin hatalarına odaklanarak, her bir örnek için yeni bir tahmin yapar.
- Modelin performansının değerlendirilmesi: Yeni modelin performansı, test veri seti üzerinde değerlendirilir. Bu, modelin doğruluğunu ve genelleştirilebilirliğini kontrol etmek için önemlidir.
- Yeni tahminlerin hesaplanması: Test veri seti üzerindeki performansın değerlendirilmesinin ardından, yeni tahminler yapmak için önceki modelin tahminleri ve yeni modelin tahminleri birleştirilir.
- Sonlandırma kriterine bağlı olarak adımların tekrarlanması: Bu adımlar, sonlandırma kriterine ulaşılıncaya kadar tekrarlanır. Sonlandırma kriteri, önceden belirlenmiş bir iterasyon sayısı, hedef performans seviyesi veya aşırı öğrenme (overfitting) gibi faktörler olabilir.

- Topluluk modelinin oluşturulması: GBM, tüm modellerin tahminlerini birleştirerek, topluluk bir model oluşturur. Bu, daha doğru tahminler yapmak için modellerin gücünü bir araya getirir.

GBM, başlangıçta verilen bir öğrenme oranı (learning rate) ve ağaç sayısı gibi parametrelerle çalışır. Öğrenme oranı, her bir modelin veriye ne kadar dikkat etmesi gerektiğini kontrol ederken, ağaç sayısı, modele ne kadar karmaşıklık eklenmesi gerektiğini kontrol eder. GBM, sınıflandırma ve regresyon problemlerinde kullanılabilir ve diğer topluluk tekniklerine kıyasla daha yüksek performans sağlar. Ancak, diğer ensemble tekniklerinde olduğu gibi, GBM de aşırı öğrenmeye yol açabilir ve bu nedenle, doğru parametre ayarlaması ve sonlandırma kriterleri kullanılması önemlidir.

3.5 XGBoost

XGBoost, gradyan artırma algoritmasının geliştirilmiş bir versiyonudur ve makine öğrenimi alanında büyük bir etkiye sahip olan bir algoritmadır. Bu algoritma, 2014 yılında Tianqi Chen tarafından oluşturulmuştur. XGBoost'un ortaya çıkışı, temel fikirleri, gradyan artırma yöntemine olan katkıları ve başlıca özellikleri ve farklılıkları şu şekilde açıklanabilir: XGBoost, gradyan artırma yönteminin performansını artırmak amacıyla çeşitli yenilikler getirilmiştir. Gradyan artırma, zayıf öğrenicilerin birleştirilerek daha güçlü bir model oluşturduğu bir topluluk öğrenme tekniğidir. XGBoost ise bu temel fikri benimseyerek gradyan artırma yönteminin etkinliğini artırmak için çeşitli yenilikler sunar.

XGBoost regresyonu, XGBoost algoritmasının regresyon problemlerine uyarlanmış bir varyasyonudur. Bu algoritma, girdi özelliklerine dayanarak sürekli sayısal değerler tahmin etmek için kullanılır. XGBoost regresyonunun temel avantajlarından biri yüksek tahmin doğruluğudur. Chen ve Guestrin'in (2016) XGBoost makalesine göre, XGBoost, çeşitli makine öğrenimi yarışmalarında yüksek tahmin doğruluğu ve verimliliği nedeniyle geniş bir şekilde tanınmıştır. XGBoost, bu doğruluğu benzersiz optimizasyon algoritmasıyla elde eder. XGBoost'un

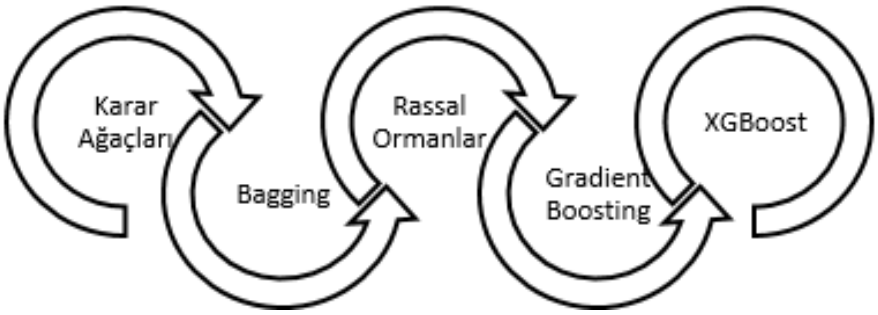
optimizasyon amacı, modelin parametrelerinin en iyi değerlerini bulmak suretiyle kayıp fonksiyonunu minimize etmektir. XGBoost, gradyan artırma ve düzenleme tekniklerinin bir kombinasyonunu kullanarak modeli iteratif olarak iyileştirir ve tahmin hatalarını azaltır.

XGBoost regresyonunun mantığı, topluluk öğrenmeye dayanır. Bu yöntemde, zayıf öğreniciler (baz modeller) birleştirilerek daha güçlü bir tahmin modeli oluşturulur. Her bir zayıf öğrenici, verinin farklı yönlerini yakalamaya odaklanır ve tahminleri ağırlıklı bir şekilde birleştirilerek son tahmin yapılır. Chen ve Guestrin (2016) açıklamasına göre, XGBoost, güçlü bir model oluşturmak için birçok zayıf modeli sistemli bir şekilde birleştiren bir yöntemdir.

XGBoost, gradyan artırma algoritmasının üzerine geliştirilen çeşitli özelliklerle daha etkili bir hale getirilmiştir. XGBoost'un başlıca özellikleri aşağıda belirtilmiştir:

- Ölçüt Optimizasyonunda İkinci Dereceden Türevler: XGBoost, ölçüt optimizasyonunda ikinci dereceden türevleri kullanarak daha hassas bir optimizasyon sağlar. Bu, GBM'e kıyasla daha etkili ve doğru bir ölçüt optimizasyonu sağlar.
- Hız ve Paralel İşleme: XGBoost, özel olarak tasarlanmış algoritma optimizasyonları sayesinde hızlı ve ölçeklenebilir bir şekilde çalışır. Paralel hesaplama yetenekleri ve düşük bellek kullanımı ile büyük veri kümeleri üzerinde etkili bir şekilde işlem yapabilir. XGBoost, özellikle büyük veri setleri ve karmaşık modellerle çalışırken GBM'den daha hızlı hesaplama yapar. Bu, daha hızlı eğitim süreleri ve daha hızlı tahminler elde etmek için avantaj sağlar.
- Düzenleme (Regularization): XGBoost, aşırı öğrenmeyi önlemek için düzenleme tekniklerini kullanır. İki tür düzenleme olan L1 düzenlileştirme (Lasso regresyonu) ve L2 düzenleme (Ridge regresyonu) kullanılabilir. Bu, modelin genelleştirme yeteneğini artırır ve aşırı uyum sorununu azaltır. Aynı zamanda, XGBoost'un daha iyi bir genelleştirme yeteneğine sahip olmasını sağlar.

- Eksik Veri İşleme: XGBoost, eksik değerleri doğrudan işleyebilir ve modele dahil edebilir. Eksik değerler için özel bir işaretleyici kullanarak, modelin eksik değerlere karşı dayanıklı olmasını sağlar.
- Çapraz Doğrulama: XGBoost, çapraz doğrulama (cross-validation) yöntemlerini kullanarak modelin performansını değerlendirebilir. Veri setini parçalara ayırarak eğitim ve değerlendirme işlemlerini gerçekleştirir ve modelin genel performansını daha güvenilir bir şekilde değerlendirebilir.
- Erken Durdurma: XGBoost, erken durdurma yöntemini kullanarak eğitim sürecini optimize eder. Belirli bir aşamada modelin performansında gelişme durduğunda eğitimi durdurarak gereksiz hesaplamaları önler. Bu, eğitim süresini kısaltır ve modelin aşırı uyum riskini azaltır.
- Ağaç Budama: XGBoost, ağaçları gereksiz dallarından budayarak daha basit ve daha etkili bir model oluşturur. Bu, modelin karmaşıklığını azaltır ve daha iyi bir genelleştirme yeteneği sağlar.
- Hızlı Büyüme: XGBoost, hızlı büyüme yeteneğiyle bilinir. Bu, hızlı ve etkili bir şekilde büyük bir ağaç ansamblesi oluşturulmasını sağlar. Hızlı büyüme, eğitim sürelerinin kısaltılmasına ve daha karmaşık modellerin kullanılmasına olanak tanır.



Şekil 3.4 Karar ağaçlarının XGBoost'a evrimi

Karar ağaçlarının XGBoost'a evrimi Şekil 3.4'deki gibi olmuştur. Karar ağaçları, topluluk öğrenme yöntemlerinde önemli bir rol oynamış ve farklı algoritmaların evriminde etkili olmuştur. İlk olarak, bagging yöntemi, birçok karar ağacının rastgele örnekleme ve ardışık eğitim ile oluşturulduğu bir topluluk öğrenme algoritmasıdır. Bu,

karar ağaçlarının varyansını azaltırken genel performansı artırır. Daha sonra, rassal ormanlar algoritması, bagging yöntemine dayanarak, her bir ağacın oluşturulmasında rastgele özellik seçimini ekleyerek modelin çeşitliliğini artırır. Gradyan artırma algoritması, ardışık olarak oluşturulan zayıf öğrenicilerin hatalarını en aza indirmek için gradyan iniş algoritmasını kullanan bir topluluk öğrenme yöntemidir. Bu yöntemde her bir ağaç, önceki ağaçların hatalarını düzeltmeye odaklanarak modelin performansını geliştirir. Son olarak XGBoost, gradyan artırma algoritmasının gelişmiş halidir. XGBoost, gradyan iniş algoritmasının yanı sıra paralel işleme, ağaç budama, eksik değerleri ele alma ve düzenleme gibi optimizasyon tekniklerini kullanarak performansı artırır ve daha hızlı, ölçeklenebilir ve genelleştirilebilir modeller elde edilmesini sağlar.

İlk olarak, bir başlangıç tahmini yapılır. Bu, veri setindeki ortalama değer veya basit bir modelin tahmini olabilir. Genellikle hedef değişkenin ortalaması olarak belirlenen başlangıç tahmin değeridir. İlk hatayı hesaplamak için gerçek hedef değerlerinden tahminleri çıkarılır.

XGBoost algoritmasında, zayıf öğreniciler olarak genellikle karar ağaçları kullanılır. Hesaplanan hatalar kullanılarak ağaçlar oluşturulur. İlk ağaç, hedef değerler ile başlangıç tahminleri arasındaki hataların gradyanına (türev) göre oluşturulur. Ardından, her bir sonraki ağaç önceki ağaçların tahminlerinden kaynaklanan hataların gradyanına göre oluşturulur. Karar ağacı, belirli bir maksimum derinlik sınırına veya diğer ağaç büyütme kriterlerine ulaşana kadar özyinelemeli olarak büyütülür.

Hataların hesaplanması ve güncellenmesi adımları şu şekilde gerçekleştirilir:

- İlk hatayı hesaplamak: İlk olarak, veri setindeki her bir örneğin gerçek değerlerini (etiketler) ve başlangıç tahminlerini kullanarak hataları hesaplarız. Hatalar, gerçek değerlerden başlangıç tahminlerini çıkararak elde edilir.
- Hataları hedef değişken olarak kullanarak yeni bir ağaç oluşturmak: Hesaplanan hataları, bir sonraki ağaç oluşturmak için hedef değişken olarak

kullanırız. Bu ağaç, hataların gradyanına (türev) göre oluşturulur. Yani, hataların gradyanı üzerinden bir öğrenme işlemi gerçekleştirilir.

- Ağaçları birleştirerek modeli güncellemek: Oluşturulan yeni ağacı, önceki ağaçlarla birleştirerek modeli güncelleriz. Bu, her bir ağacın tahminlerinin ağırlıklı olarak birleştirilmesini içerir. Ağaçların birleştirilmesi, toplam tahminin daha doğru ve güvenilir olmasını sağlar.
- Hataları güncellemek: Yeni ağacın tahminlerini kullanarak hataları güncelleriz. Bu, yeni ağacın tahminlerini önceki artıklardan çıkararak gerçekleştirilir. Elde edilen yeni hatalar, bir sonraki iterasyonda kullanılmak üzere güncellenir.

Bu adımlar, XGBoost algoritmasının iteratif olarak çalışmasını sağlar. Her bir iterasyonda, yeni ağaçlar oluşturulur, model güncellenir ve hatalar yeniden hesaplanır. Bu şekilde, model zamanla daha doğru ve verimli hale gelir. XGBoost'un GBM'den farklılıklarından biridir. GBM'de genellikle sadece hatalar kullanılarak yeni ağaçlar oluşturulurken, XGBoost'ta daha karmaşık bir optimizasyon işlemi gerçekleştirilir ve ağaçlar daha etkili bir şekilde birleştirilir.

XGBoost algoritmasında düzenleme (regularization) ve küçültme (shrinkage) teknikleri, modelin karmaşıklığını kontrol etmek ve aşırı uyumu önlemek için kullanılır. Düzenleme, XGBoost algoritmasında kullanılan bir tekniktir ve modelin karmaşıklığını kontrol etmeye yardımcı olur. Bu, ağaçların aşırı uyuma yatkınlığını azaltarak daha genelleştirilebilir modeller oluşturmayı hedefler. Düzenleme için yaygın olarak kullanılan yöntemler L1 düzenlemesi (Lasso) ve L2 düzenlemesi (Ridge) dir.

- L1 Düzenlemesi (Lasso): L1 düzenlemesi, ağaçlardaki ağırlıkları sıfıra yaklaştırarak gereksiz özellikleri elemek için kullanılır. Bu şekilde, modelin özellik seçimi yapması ve daha basit bir yapı oluşturması sağlanır. L1 düzenlemesi, özellikler arasındaki bağıntıyı azaltabilir ve düşük etkili özellikleri ortadan kaldırabilir.

- L2 Düzenlemesi (Ridge): L2 düzenlemesi, ağaçlardaki ağırlıkları küçültmek ve aşırı uyumu önlemek için kullanılır. Ağaçların ağırlıkları sıfıra yaklaşır, ancak sıfır olmaz. L2 düzenlemesi, modelin genel karmaşıklığını azaltarak daha istikrarlı ve genelleştirilebilir bir model elde etmeyi sağlar.

Küçültme, XGBoost algoritmasının bir diğer önemli bileşenidir. XGBoost algoritmasında küçültme kavramı genellikle öğrenme oranı (learning rate) olarak da adlandırılır. Küçültme, her bir zayıf öğrenici (karar ağacı) için bir öğrenme oranı belirleyerek gerçekleştirilir. Daha küçük bir öğrenme oranı, her bir ağacın etkisini azaltır ve modelin daha yavaş bir şekilde öğrenmesini sağlar. Bu, modelin genelleştirme yeteneğini artırır ve aşırı uyumu önler. Küçültme faktörü, her bir ağacın katkısını kontrol eden bir ağırlık olarak düşünülebilir. Daha küçük bir küçültme faktörü, her bir ağacın daha az etkili olmasını sağlar ve modelin daha fazla iterasyon yapmasını gerektirir. Bu, modelin daha sağlam ve istikrarlı olmasını sağlar. Düzenleme ve küçültme, XGBoost algoritmasının performansını ve genelleştirme yeteneğini artırmak için kullanılan önemli tekniklerdir. Bu teknikler, modelin aşırı uyumu kontrol etmesini, özellik seçimini yapmasını ve daha istikrarlı bir tahmin yapabilmesini sağlar.

Ağaçlar birleştirme adımında oluşturulan ağaçlar, başlangıç tahminine ve diğer ağaçların tahminlerine göre ağırlıklı olarak birleştirilir. Bu şekilde, her bir ağaç, önceki ağaçların hatalarını düzeltmek için katkıda bulunur. XGBoost algoritması, zayıf öğrenicilerin tahminlerini birleştirerek nihai tahmini oluştururken, öğrenme hızıyla ağırlıklandırma yapar. Her zayıf öğrenicinin tahmini, bir önceki tahminle ağırlıklandırılır. İlk tahminde bu ağırlık 1'dir.Ağırlıklandırılmış tahminler toplanır ve nihai tahmin elde edilir. XGBoost algoritması, birden fazla zayıf öğreniciyi (genellikle karar ağaçlarını) birleştirerek tahminlerin doğruluğunu artırmaya çalışır. Her bir zayıf öğrenici, önceki öğrenicilerin hatalarını azaltmaya çalışır ve kalan hatayı düzeltir. Bu iteratif süreç, modelin daha güçlü ve genelleştirilebilir olmasını sağlar.

Yeni bir zayıf öğrenici oluşturmak için adımlar tekrarlanır. Bu adımlar, belirli bir iterasyon sayısı veya önceden belirlenmiş bir durdurma kriterine kadar devam eder.

İterasyonlar tamamlandığında, tüm zayıf öğrenicilerin tahminleri birleştirilerek final tahmin elde edilir.

3.6 Model Performans Metrikleri

Modellerin regresyon performansını değerlendirmek için tahminlerimizin gerçek verilerden ne kadar uzak olduğunun bir ölçüsü olan Hata Kareler Ortalaması (MSE), Ortalama Mutlak Hata (MAE), Kök Hata Kareler Ortalaması (RMSE) ve belirtme katsayısı R^2 kullanılmıştır. Diğer modellerin sonuçlarıyla da değerlendirilerek daha anlamlı yorumlar yapılabilir.

3.6.1 Hata Kareler Ortalaması (MSE)

Hata kareler ortalaması, gerçek ve tahmin edilen değerler arasındaki farkların karelerinin ortalamasıdır. Yüksek MSE değeri, gerçek değer ile tahmin değerleri arasındaki farkın büyük olduğunu model performansının iyi olmadığını gösterir. MSE, hataların büyüklüğünü dikkate alırken, pozitif ve negatif hataları dengeleyen bir metriktir.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2 \tag{2.16}$$

3.6.2 Ortalama Mutlak Hata (MAE)

Ortalama mutlak hata (MAE), gerçek ve tahmin edilen değerler arasındaki mutlak farkların ortalamasını hesaplar. MAE, hataların büyüklüğünü dikkate alırken, pozitif ve negatif hataları dengeleyen bir metriktir. Daha düşük MAE değeri, daha iyi bir model performansını gösterir.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \tag{2.17}$$

3.6.3 Kök Hata Kareler Ortalaması (RMSE)

Kök hata kareler ortalaması (RMSE), gerçek ve tahmin edilen değerler arasındaki farkların karelerinin ortalamasının karekökünü hesaplar. RMSE, MSE'ye benzer şekilde hataların büyüklüğünü dikkate alır, ancak daha yüksek hata değerlerine daha fazla ağırlık verir. Daha düşük RMSE değeri, daha iyi bir model performansını gösterir.

$$RMSE = \frac{1}{n}\sqrt{\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \tag{2.18}$$

3.6.4 Belirtme Katsayısı (R^2)

R^2 bir regresyon modelinin uyumunu ve açıklama gücünü ölçen bir istatistiksel ölçüttür. R^2 değeri, bağımlı değişkenin varyasyonunun, bağımsız değişkenler tarafından açıklanan yüzdesini gösterir. R^2 değerinin bire yakın çıkması bağımlı değişkendeki değişimin model tarafından yüksek performansta açıklanabildiğini gösterir. R^2, modelin tahmin performansının bir göstergesidir.

$$R^2 = 1 - \sum_{j=1}^{n}\frac{(y_j - \hat{y}_j)^2}{(y_j - \frac{\hat{y}_j}{n})^2} \tag{2.19}$$

BÖLÜM DÖRT
ANALİZ VE UYGULAMA

Uygulamanın yapıldığı fabrika 2000 yılından bu yana örme kumaş üretimi gerçekleştirmektedir. Örme kumaşlar, çeşitli giyim ve tekstil ürünlerinde yaygın olarak kullanılır ve bu tür kumaşların eni, kullanım amacına ve konfor düzeyine doğrudan etki eder. Kumaş eni, örme kumaşların konfor özelliklerini etkileyen önemli bir faktördür. Bu çalışma, kumaş ilk kez oluşturulurken kumaş eni değerinin insandan bağımsız otomatik bir şekilde sistem tarafından verilmesi ihtiyacından doğmuştur. Geliştirilen kumaşlarda kumaş eninden sapma daha fazla hammadde tüketimine yol açmaktadır. Bu tüketim sonucu kumaş tamir oranları artmaktadır. Kumaş kalitelerindeki tamir ve hurda miktarını azaltmak son derece önemli olduğundan bu çalışmanın konusu kumaş enini doğru tahminlemek olmuştur. Yeni geliştirilen kumaşlarda da kumaş eninin doğru tahminlenmesi için gerekli etkenlerin ve bu etkenlerin düzeylerinin belirlenmesi gerekmektedir. Yeni bir kumaş üretildiğinde doğru bir en modellemesi ile hatasız üretim gerçekleştirilmesi hedeflenmektedir.

Gerçek zamanlı tekstil veri setinde pamuk elyaf karışımlı 162 adet kumaş tipinin enlerinin sistematik olarak belirlenmesi için uygulama yapılmıştır.

4.1 Veri Setinin İncelenmesi ve Ön İşleme

Veri seti olarak, süprem örgü tipinde %51 veya daha fazlası pamuk içerikli elyaf içeren kumaş verileri kullanılmıştır. Burada metinsel kumaş tanımlarından kumaş eninin tahminlenmesi için önemli özellikler keşfedilip ilgili bağımsız değişkenler oluşturulmuştur. Kumaşın özelliklerini etkileyen ve modelde yer alan bağımsız değişkenler; fein, gramaj, 1. ilmek iplik uzunluğu, 2. ilmek iplik uzunluğu, 3. ilmeğinin olması, elastan oranı, viskon oranı, pamuk oranı, polyester oranı, saf pamuk içerikli olması, elastan numarası, iplik numarası ve likralı olma özelliği olarak belirlenmiştir.

Kumaş tanımları metinsel bir ifade ile kaydedilmektedir. Örneğin: 40/1 Ring Combed 40D Full Lyc Sıngle Jersey – (90%Co 10%Ea) - 1. İlmek uzunluğu 29, 2.

İlmek uzunluğu 11 (cm/100). Kullanılan değişkenlerin oluşturulması ve örnek verilen bir örme kumaşa ait özellik çıkarımı Tablo 4.1'de gösterilmiştir:

Tablo 4.1 Örme kumaşa ait özellik çıkarımı

ÖZELLİKLER	DEĞERLER
GRAMAJ	170
FEIN	28
ILMEKUZ_1	29
ILMEKUZ_2	11
IIU_3	0
EA_ORANI	10
VIS_ORAN	0
PAMUK_ORAN	90
PES_ORAN	0
EA_NO	40
IP_NO	40
LIKRALI_MI	1
PAMUK100_MU	0
KUMAS_ENI	188

Gramaj, kumaşın ağırlığını ve yoğunluğunu ifade eder. Daha yüksek gramaj değeri, daha yoğun ve kalın bir kumaşa işaret eder. Fein, kumaşın ince liflerden oluşup oluşmadığını belirler. Daha küçük bir fein değeri, ince ve yumuşak bir kumaşa işaret eder. 1. ve 2. ilmek iplik uzunlukları, kumaşın dayanıklılığını ve elastikiyetini etkiler. Uzun ilmek iplikleri daha dayanıklı ve esnek bir kumaşa işaret eder. Elastan oranı, kumaşın esnekliğini ve geri dönüş kabiliyetini sağlar. Daha yüksek elastan oranı, daha elastik bir kumaşa işaret eder. Viskon, pamuk ve polyester oranları, kumaşın dokusunu, yumuşaklığını ve nefes alabilirliğini etkiler. Daha yüksek viskon oranı, daha parlak ve yumuşak bir kumaşa işaret ederken, pamuk kumaşın doğal ve nefes alabilir yapısını sağlar ve polyester kumaşın dayanıklılığını artırabilir. Elastan ve iplik numaraları, kumaşın elastikiyetini ve dokusunu etkiler. Daha yüksek elastan ve iplik numarası, daha elastik ve ince bir kumaşa işaret eder. Likralı olma özelliği, kumaşın elastik özelliklerini artırır ve giysilerde daha iyi oturmasına yardımcı olur. Bu özelliklerin kombinasyonu, kumaşın genel performansını ve kullanım alanlarını belirler ve tasarım sürecinde önemli bir rol oynar.

Bu çalışma da bir bağımlı ve 13 bağımsız değişkene karşılık 162 adet gözlem değeri bulunmaktadır. Bağımlı ve bağımsız değişkenlerin özet istatistikleri Tablo 4.2'de gösterilmiştir.

Tablo 4.2 Bağımlı ve bağımsız değişkenlerin özet gösterimi

GRAMAJ		**FEIN**	
Minimum:	70.00	Minimum:	14.00
1.Çeyrek:	140.00	1.Çeyrek:	28.00
Medyan:	165.00	Medyan:	28.00
Ortalama:	171.80	Ortalama:	27.15
3.Çeyrek:	198.80	3.Çeyrek:	28.00
Maksimum:	310.00	Maksimum:	44.00
ILMEKUZ_1		**ILMEKUZ_2**	
Minimum:	18.80	Minimum:	0.00
1.Çeyrek:	26.75	1.Çeyrek:	0.00
Medyan:	29.80	Medyan:	8.05
Ortalama:	29.52	Ortalama:	8.24
3.Çeyrek:	31.00	3.Çeyrek:	11.00
Maksimum:	49.20	Maksimum:	43.60
IIU_3		**EA_ORANI**	
0:	159	Minimum:	0.00
1:	3	1.Çeyrek:	0.00
		Medyan:	0.00
		Ortalama:	2.72
		3.Çeyrek:	5.00
		Maksimum:	19.00
VIS_ORAN		**PAMUK_ORAN**	
Minimum:	0.00	Minimum:	57.00
1.Çeyrek:	0.00	1.Çeyrek:	93.00
Medyan:	0.00	Medyan:	100.00
Ortalama:	0.23	Ortalama:	94.93
3.Çeyrek:	0.00	3.Çeyrek:	100.00
Maksimum:	38.00	Maksimum:	100.00

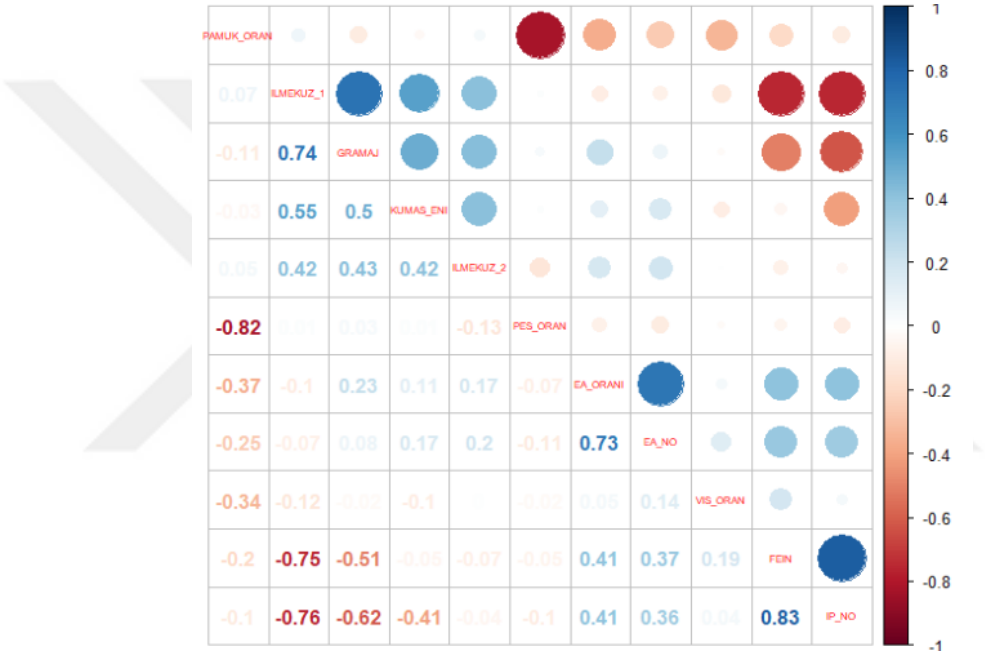
PES_ORAN		EA_NO	
Minimum:	0.00	Minimum:	0.00
1.Çeyrek:	0.00	1.Çeyrek:	0.00
Medyan:	0.00	Medyan:	0.00
Ortalama:	1.98	Ortalama:	79.16
3.Çeyrek:	0.00	3.Çeyrek:	133.00
Maksimum:	40.00	Maksimum:	265.00
LIKRALI_MI		**PAMUK100_MU**	
0:	99	00:00	73
1:	63	01:00	89
IP_NO		**KUMAS_ENI**	
Minimum:	10.00	Minimum:	152.00
1.Çeyrek:	24.00	1.Çeyrek:	180.00
Medyan:	30.00	Medyan:	188.50
Ortalama:	32.56	Ortalama:	188.90
3.Çeyrek:	40.00	3.Çeyrek:	199.80
Maksimum:	100.00	Maksimum:	223.00

Yapay sinir ağı matematiksel modellere dayandığından, sinir ağları için tüm bağımlı ve bağımsız değişkenlerin sayısal olması gerekir. Yapay sinir ağları, girdi verilerini sayısal değerlerle temsil eden matrislere dönüştürür ve bu matrisleri kullanarak işlemler yapar. Bu nedenle, sayısal olmayan veri tipleri (örneğin metin veya kategorik veriler) doğrudan yapay sinir ağlarına verilemez. Ancak, bazı yöntemlerle sayısal olmayan verileri yapay sinir ağlarında kullanmak mümkündür. Örneğin, metin verileri için önceden işleme teknikleri kullanılarak metinler sayısal vektörlere dönüştürülebilir. Bu nedenle veri tipleri düzenlenmiştir, faktör tipindeki değişkenler numerik veri tipine çevrilmiştir.

- Binary değişkenlerin etiketleri 0-1 şeklinde yeniden düzenlenmiştir.
- Tüm iplik numaraları tek bir birimde toplanmıştır.

Veri setinin istatistiksel özetine bakıldığında değişkenlerin birimi birbirinden farklı olduğundan minimum-maksimum değerleri, ortalamaları ve standart sapmaları birbirinden uzak olduğu görülmektedir.

Pamuk süprem kumaşlarının en değerlerini belirlemek için kullanılan girdi değişkenlerin Şekil 4.1'deki korelasyon değerlerine bakıldığında; kumaş eni ile pozitif yönlü orta düzeyde ilişkili olan değişkenler sırasıyla 1. ilmek iplik uzunluğu (r= 0.55), gramaj (r=0.50) ve 2. ilmek iplik uzunluğu (r=0.42)'dir.



Şekil 4.1 Korelasyon Matrisi

Korelasyon matrisine bakıldığında kumaş eninin girdi değişkenleri ile pozitif ve negatif orta düzey ve zayıf korelasyonları mevcuttur. Orta ve zayıf düzey korelasyon değerlerine sahip verilerde yapay sinir ağları daha başarılı olabilir. Zayıf korelasyon, değişkenlerin birbirinden bağımsız olduğu veya ilişkilerinin doğrusal olmadığı anlamına gelir. Yapay sinir ağları, karmaşık ve doğrusal olmayan ilişkileri yakalama yetenekleriyle bilinirler ve bu sayede düşük korelasyonlu verilerde de etkili bir şekilde çalışabilirler.

Hem topluluk öğrenme yöntemleri hem de yapay sinir ağları için veri ön işleme önemlidir. Girdi verileri normalize edilmeli veya standartlaştırılmalıdır. Ayrıca, gürültülü gözlemlerin veya gereksiz değişkenlerin temizlenmesi veya çıkarılması da yapılabilir.

Sayısal verileri ölçeklendirmek her zaman gerekli değildir. Fakat verisetindeki değişkenlerin birimleri ve uzaklıkları fazla olduğundan ölçeklendirilebilir. Ölçeklendirme ile veri birimleri ortadan kaldırılarak farklı konumlardan verileri kolayca karşılaştırabiliriz. Bununla birlikte, sayısal değerler ölçeklendirildiğinde, sinir ağı oluşumunun genellikle daha verimli olduğu ve daha iyi tahmine yol açtığı bilinmektedir.

Yapay sinir ağları, ağırlıkları güncellemek ve optimize etmek için gradyan tabanlı bir optimizasyon yöntemi kullanır. Bu nedenle, değişkenlerin ölçekleri arasında büyük farklılıklar olduğunda, ağırlıkların güncellenmesi dengesiz olabilir ve algoritma performansı etkilenebilir. Veri setindeki değişkenleri ölçeklendirme, bu farklılıkları ortadan kaldırarak ağın daha dengeli bir şekilde öğrenmesini sağlar. Ölçeklendirme genellikle değişkenleri 0 ile 1 arasında veya standart normal dağılıma uygun hale getirme yöntemleriyle yapılır. Karar ağaçları da veri setinin ölçeklendirilmesinden etkilenebilir, ancak ölçeklendirme gerekliliği yapay sinir ağlarından daha azdır. Karar ağaçları, veri setini belirli eşik değerlerine göre böler ve dallanır. Bu nedenle, değişkenlerin ölçekleri arasındaki farklılıklar, karar ağacının yapısını doğrudan etkilemez. Ancak, bazı ölçeklendirme yöntemleri, karar ağacının daha iyi performans göstermesini sağlayabilmektedir.

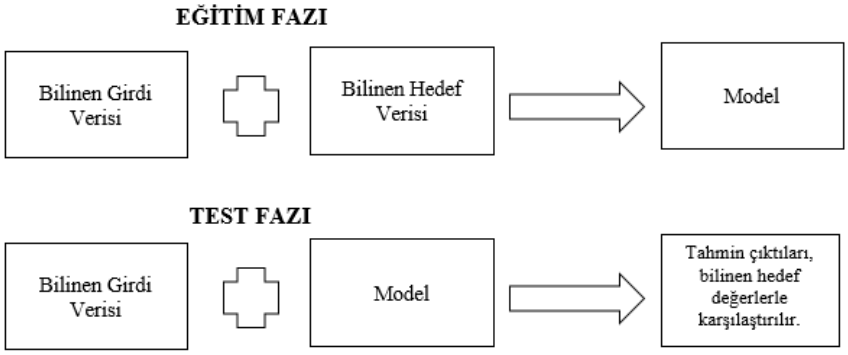
Verileri ölçeklendirmek için farklı yöntemler vardır. Bu çalışma da, [0,1] aralığındaki tüm ölçeklenmiş verileri almak için minmax yöntemi (genellikle özellik ölçekleme olarak adlandırılır) kullanılmıştır. Min-max ölçekleme yöntemi, özellikle yapay sinir ağları ve diğer algoritmalar için yaygın olarak kullanılan bir ölçekleme yöntemidir.

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

4.2 Modellerin Eğitilmesi ve Test Edilmesi

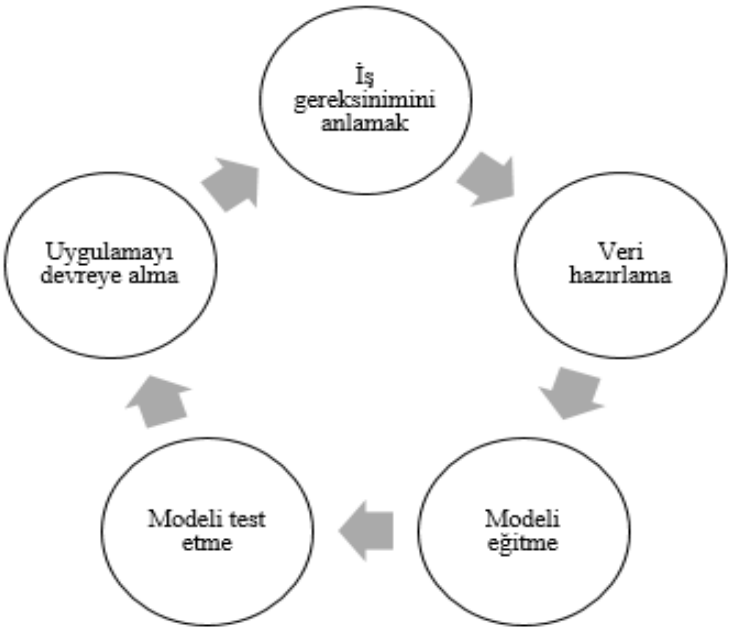
Bu çalışma da veri seti %70'i eğitim, %30'u test olarak ayrılmıştır. Model eğitim veri seti ile kurulup, test verisi ile model performansları kontrol edilmiştir.

Eğitim aşamasında, eğitim verileri sinir ağına beslenerek girdiler ile çıktılar arasındaki karmaşık ilişkileri öğrenmesi sağlanır. Yapay sinir ağı, eğitim verilerine dayanarak ağırlıklarını, yanlılıklarını ve aktivasyon fonksiyonlarını ayarlar ve performansını geliştirir. Yeterli bir yakınsama sağlandıktan sonra, model bellekte saklanır ve test aşamasına geçilir. Bu adımda, test verileri modele geçirilir ve tahmin edilen çıktılar gerçek çıktılarla karşılaştırılır. Modelin performansını değerlendirmek için çeşitli ölçütler kullanılabilir. Şekil 4.2' de eğitim ve test fazları gösterilmiştir.



Şekil 4.2 Eğitim ve test fazları

Veri, model oluşturma ve öğrenme süreci için temel bir bileşeni oluşturur. Veri toplanmalı, temizlenmeli, dönüştürülmeli ve ardından modele öğrenme için beslenmelidir. Genel veri modelleme yaşam döngüsü Şekil 4.3'deki gibi gösterilir:



Şekil 4.3 Veri modelleme döngüsü

4.2.1 Yapay Sinir Ağları Modellerin Oluşturulması ve Test Edilmesi

Yapay sinir ağı modelleri R programı ile kurulmuştur. Farklı modeller oluşturulup modeller birbirleri ile karşılaştırılmıştır. Model başarısı farklı kriterler de değerlendirilip sonuçlar incelenmiştir.

- Performans doğruluk ölçütlerine göre (MAE, MSE, RMSE ve R^2) ve
- Tahmin edilen kumaş eninin mevcut kumaş eninden ± %5 sapma miktarına göre değerlendirilmiştir.

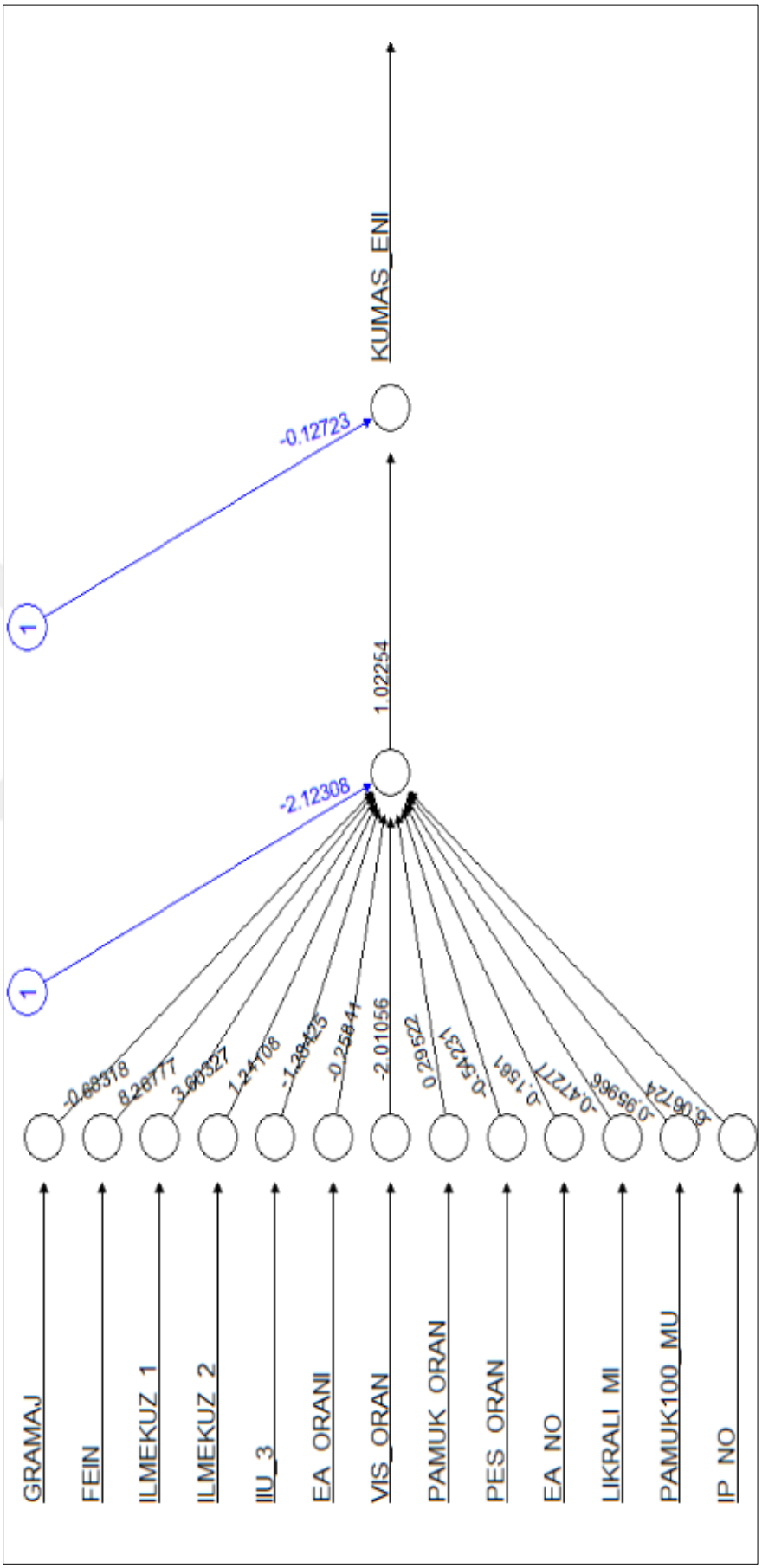
İlk model 1 katmanlı 1 nöronlu basit bir sinir ağı modeli olarak oluşturulmuştur. R programında "neuralnet" fonksiyonu ile sinir ağları modeli oluşturulmuştur.

Aşağıdaki argümanlar ile ayarlar yapılabilir:

- lineer.output argümanı, regresyon (linear.output=TRUE) veya sınıflandırma (linear.output=FALSE) yapmak isteyip istemediğimizi belirtmek için kullanılır.

- hidden argümanı ile gizli katman ve nodül sayısı ayarlanabilmektedir.
- learningrate argümanı ile modelin geri yayılım tarafından kullanılan öğrenme oranını belirten sayısal değeri belirlenir. Neuralnet paketinde öğrenme hızı (learning rate) için varsayılan değer 0.01'dir. Genellikle, öğrenme hızı için 0.01 ile 0.1 arasında bir değer seçilir ve bu değer modelin performansına göre ayarlanır. Ancak, her veri seti ve model için en uygun öğrenme hızı farklılık gösterebilir. Daha küçük bir öğrenme hızı, modelin daha istikrarlı bir şekilde öğrenmesini sağlar, ancak eğitim sürecini yavaşlatabilir. Daha büyük bir öğrenme hızı ise modelin daha hızlı öğrenmesini sağlar, ancak aşırı uyuma riskini artırabilir.
- stepmax argümanı, sinir ağının eğitimi için kullanılan maksimum adım sayısını belirtilir. Ayarlanabilir hiperparemetrelerden biridir.
- act.fact argümanı, aktivasyon fonksiyonlarını belirtir. Act.fct (aktivasyon fonksiyonu) parametresi "logistic" olarak seçildiğinde, sigmoid (lojistik) fonksiyonu kullanılır.

Model eğitim veri seti ile kurulup, test verisi ile kontrol edimiştir. Yukarıda bahsedilen parametrelerin varsayılan değerleri kullanılarak model kurulmuştur. Bir katmanlı bir nöronlu basit yapay sinir ağı modeli Şekil 4.4’de gösterilmiştir.



Şekil 4.4 1 katmanlı 1 nöronlu basit yapay sinir ağı modeli

Grafikte, siyah çizgiler (bu çizgiler giriş düğümlerinden başlar) her katman arasındaki bağlantıları ve her bağlantıdaki ağırlıkları gösterirken, mavi çizgiler her adımda eklenen sapmayı gösterir. Bu sapma, doğrusal bir modelin kesişimi olarak düşünülebilir.

Gizli katmandaki nöronların her birine rastgele ağırlıklar ve yanlılık değeri atanarak sinir ağı ileri yayılımı başlar. Girdiler ve ağırlıklar çarpılır ve bias terimi ile birlikte toplanır. Ardından aktivasyon fonksiyonu (sigmoid) gizli katmandaki nörona uygulanır ve çıkış nöronuna iletilir. Çıktı katmanındaki nöronundaki ağırlık ile çarpılmış giriş değerlerinin yanlılık ile toplamının genel toplamı alınır.

Gerçek ve tahmin edilen değerler arasındaki farka hata terimi denir. İleri beslemeli bir sinir ağındaki öğrenme süreci aslında geri yayılım aşamasında gerçekleşir. Model, her yinelemede hata terimi azaltılarak ağırlıklarla ince ayarlar yapılır. Geri yayılım işleminde gradient descent optimizasyon algoritması kullanılır. Ağın ağırlıklarına göre hata teriminin (E) kısmi türevleri hesaplanır. Her nörondaki (herhangi bir katmandaki) ağırlık w_{ij}, öğrenme oranı ile birlikte bu kısmi türev ile güncellenir. Bu adımlar, çok düşük hata terimi veya belirli sayıda iterasyona kadar tekrarlanır.

Tablo 4.3 Bir katmanlı bir nöronlu basit yapay sinir ağı değerlendirme metrikleri

RMSE	**MAE**	**MSE**	**R^2**
4,98	4,07	24,77	0,88

Bir diğer başarı değerlendirme ölçütü; tahmin edilen en değerlerinin ±%5 kabul sınırları içerisinde kalmasıdır. Tahmin değerleri şirket tarafından belirlenmiş olan bu sınırlar içerisinde kalıyorsa o en değeri kabul edilebilir. Sonuçlara bakıldığında, test verilerinin %94'ü kabul sınırları içerisindedir. 47 adet test verisinden 44'ü kabul edilebilir sınırlar içerisindedir. Aşağıda Tablo 4.4'te örnek olarak 8 kumaş örneğinin tahmin değerleri ve gerçek değerleri arasındaki sapmaya bağlı olarak kabul kararları gösterilmiştir.

Tablo 4.4 Model sonuçlarının gerçek değerlere göre sapmasının kabul değerleri

Tahmin Değeri	Gerçek Değeri	Kontrol
157,38	153,00	Kabul
161,76	159,00	Kabul
170,06	165,00	Kabul
157,65	168,00	Red
172,61	172,00	Kabul
170,47	174,00	Kabul
170,35	175,00	Kabul
184,60	176,00	Kabul

Yapay sinir ağlarında yapının kaç gizli katmandan oluşacağı ya da kaç tane nöronla birleşeceği gibi sayıların belli bir kuralı yoktur. Bu sayılar belirlenirken deneme yanılma yolu kullanılır. Bu çalışmada validasyon işlemleri ile bu sayı belirlenmeye çalışılacaktır. Büyük yapılar için genelde bir ya da iki gizli katman sayılarının yeterli olacağı literatür de belirtilmiştir.

Tahmin faktörleri az sayıda nöronun yakalayamayacağı kadar karmaşık olabileceğinden, az sayıda nöron sistemde yüksek hataya yol açacaktır. Çok sayıda nöron, eğitim verilerine fazla uyacaktır ve iyi genelleme yapmayacaktır. Buna göre farklı iterasyonlardaki model sonuçları incelenmiştir, sonuçları Tablo 4.5'teki gibidir;

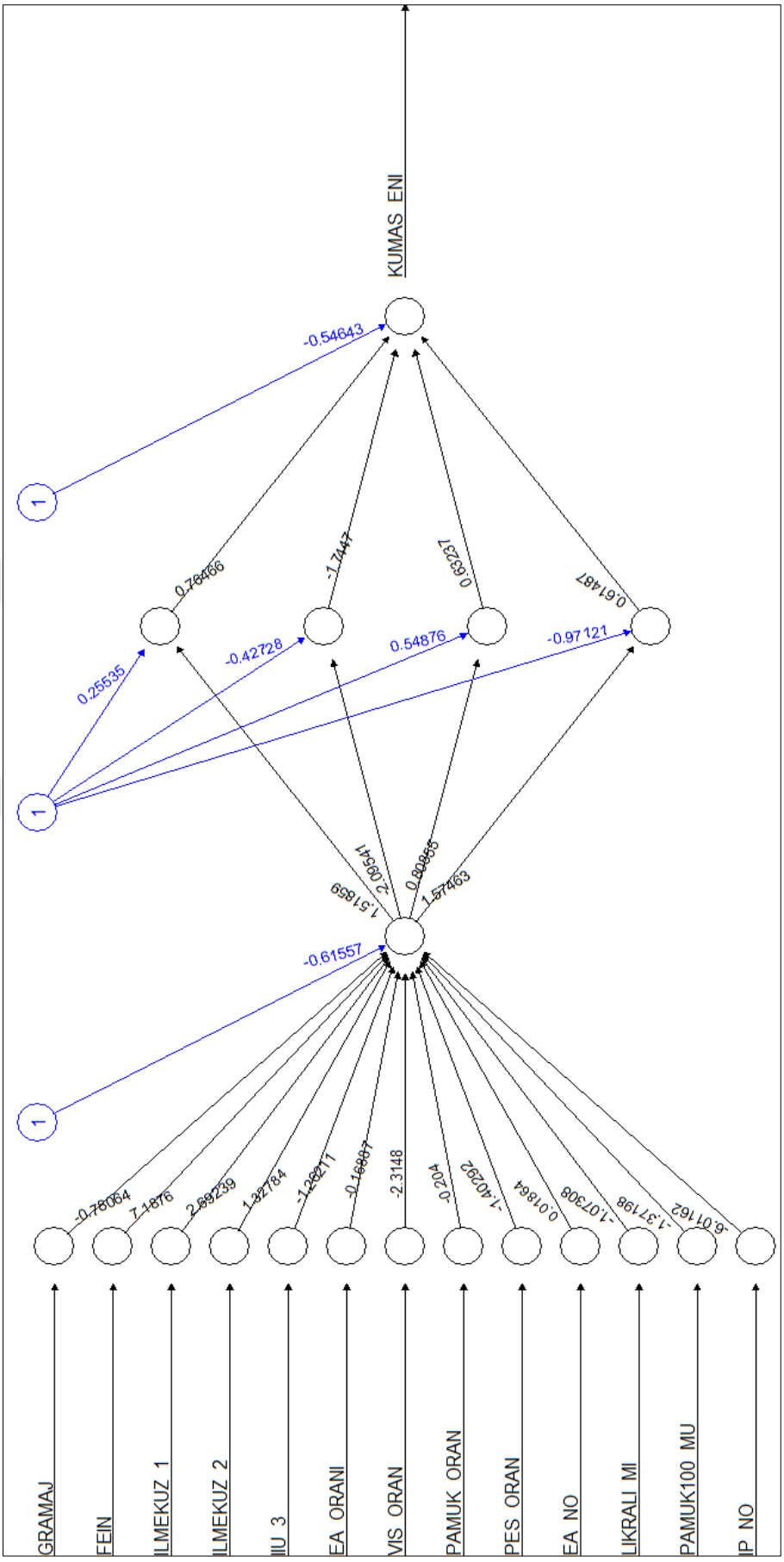
Tablo 4.5 Farklı iterasyonlardaki model sonuçları

Katman1	**Katman2**	**RMSE**	**R^2**	**MAE**
1	0	0,1051042	0,7139540	0,07691649
1	1	0,1018654	0,7188142	0,07676025
1	2	0,1098544	0,6899358	0,07957616
1	3	0,1031015	0,7077403	0,07692775
1	4	0,1008585	0,7274597	0,07375670
2	0	0,1018280	0,7290340	0,07358435
2	1	0,1033819	0,7170853	0,07720425
2	2	0,1058577	0,7006600	0,07617896
2	3	0,1072083	0,7093822	0,07715048
2	4	0,1066608	0,7004794	0,07663777
3	0	0,1104428	0,6929157	0,07693439
3	1	0,1114703	0,6854803	0,08050297
3	2	0,1147369	0,6660468	0,08008315
3	3	0,1090225	0,6930023	0,07769442
3	4	0,1076878	0,6940627	0,07696438
4	0	0,1097850	0,6826200	0,07644268
4	1	0,1108312	0,6746947	0,08145098
4	2	0,1049116	0,7101058	0,07668714
4	3	0,1059582	0,7051065	0,07599374
4	4	0,1094442	0,6958398	0,07846108

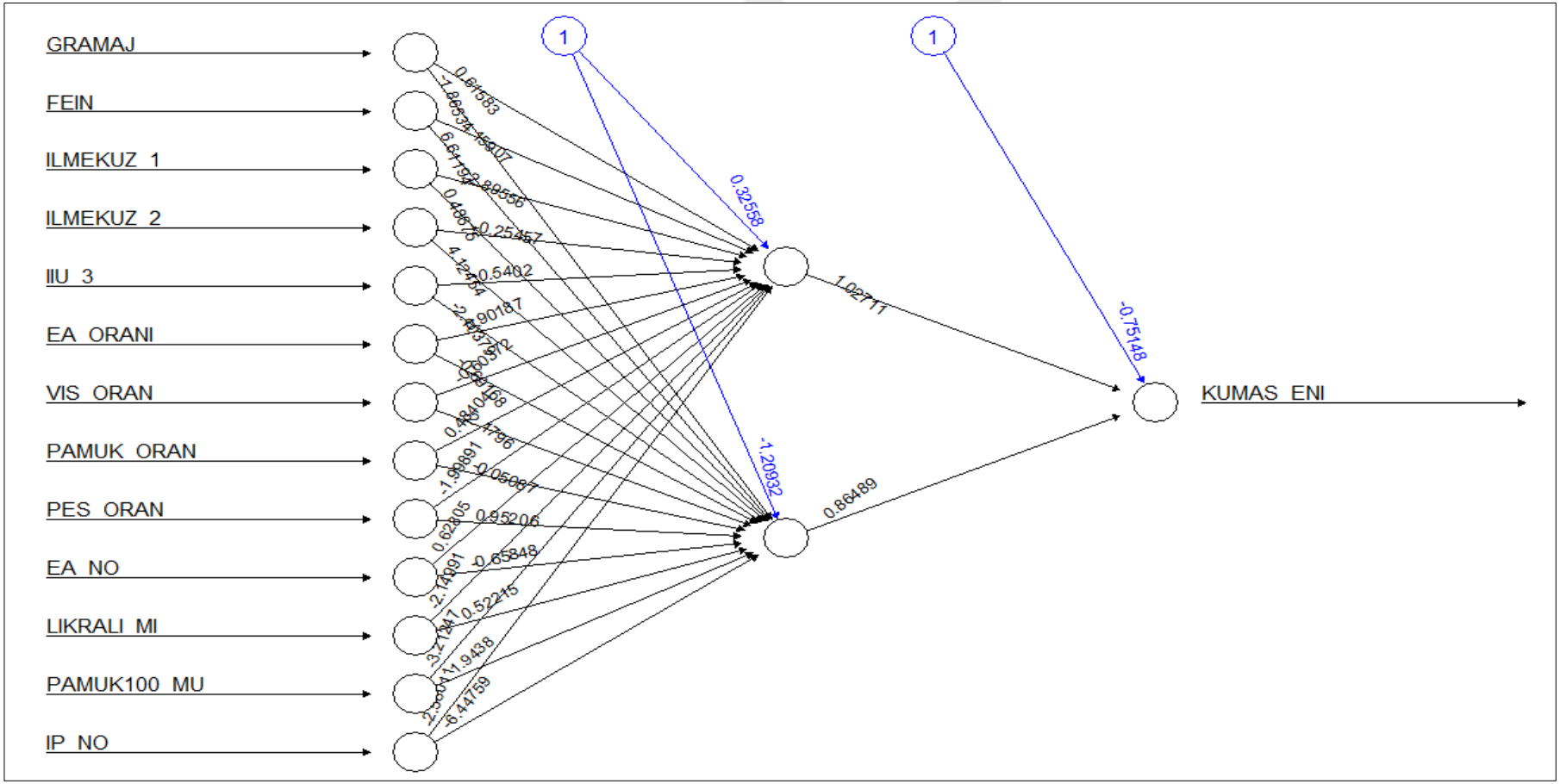
Sonuçlar incelendiğinde 1. katmanda 1 nöron, 2. katmanda 4 nöronun olduğu durumda model en düşük RMSE değerini vermiştir. RMSE değeri 0.10, R^2 değeri 0.727 olarak hesaplanmıştır. Validasyon işlemleri sonucu en yüksek R^2 değerinin (0.729) oluştuğu model 1. katmanında 2 nöronun olduğu yapay sinir ağı modelidir. Bu iki model de incelenip sonuçları değerlendirilmiştir.

- 1. katmanda 1 nöron, 2. katmanda 4 nöron bulunan yapay sinir ağı (en düşük RMSE)
- 1. katmanında 2 nöron bulunan yapay sinir ağı (en yüksek R^2)

Birinci katmanında bir nöron ikinci katmanında dört nöron bulunan yapay sinir ağı modeli Şekil 4.5'de, birinci katmanında iki nöron bulunan yapay sinir ağı modeli Şekil 4.6'da gösterilmiştir.



Şekil 4.5 1. katmanda 1 nöron, 2. katmanda 4 nöron bulunan



Şeki 4.6 1. katmanında 2 nöron bulunan 3. yapay sinir ağı

Kurulan üç yapay sinir ağı modeli içinde sonuçlar aşağıdaki gibi karşılaştırılmıştır. Sonuçlar incelendiğinde tüm hata değerlendirme kriterleri için de 1 katmanlı 1 nöronlu ilk modelin daha iyi sonuç verdiği görülmektedir. 2. model için eğitim verisiyle modelin daha iyi öğrenmiştir fakat test verisinde sonuçlarının ilk modele göre daha düşük olduğunu görülmektedir. 3. modelin kabul edilebilirlik yüzdesine baktığımızda %89 evet oranı ile 2. modelden daha iyidir. Tablo 4.6 ve Tablo 4.7'de değerlendirme sonuçları yer almaktadır.

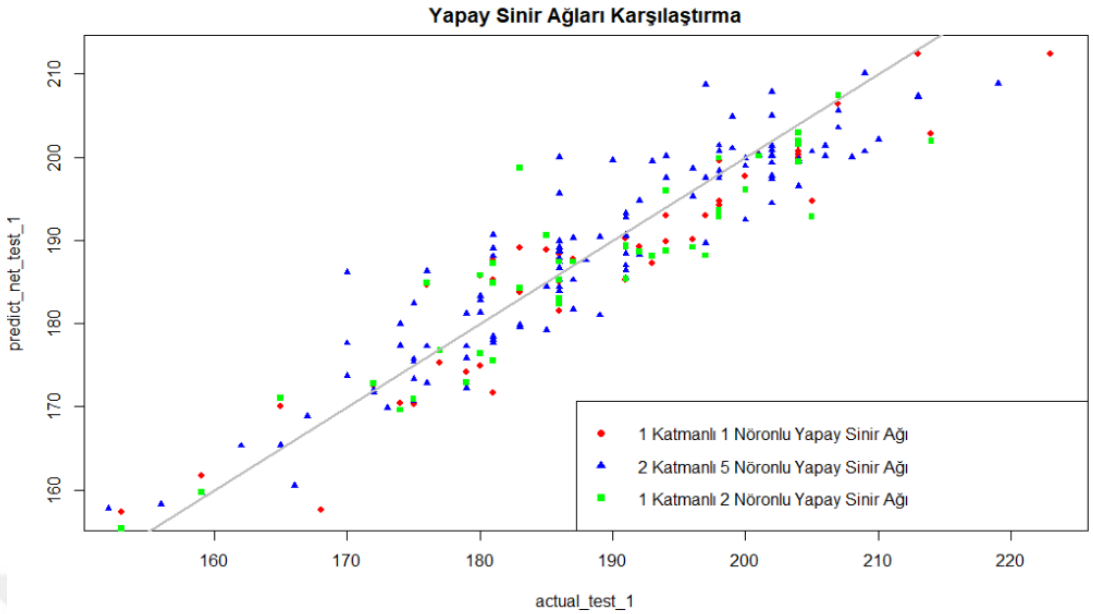
Tablo 4.6 Tüm yapay sinir ağı değerlendirme metrikleri ile karşılaştırılması

Modeller	**RMSE**	**MAE**	**MSE**	**R^2**
1. Model	4,98	4,07	24,77	0,88
2. Model	5,95	4,72	35,45	0,83
3. Model	5,74	4,45	32,94	0,84

Tablo 4.7 Model sonuçlarının kabul edilebilirlik yüzdelikleri

Modeller	**Kabul Yüzdelikleri**
1. Model	%94
2. Model	%83
3. Model	%89

Modeller karşılaştırıldığında, Şekil 4.7' de modellerden elde edilen tahminlerin birbirine yakın olduğu görülmektedir. Tüm performans metrikleri karşılaştırıldığında en iyi sonuçların tek katmanlı tek nöronlu yapay sinir ağı modellemesi ile elde edildiği görülmektedir.

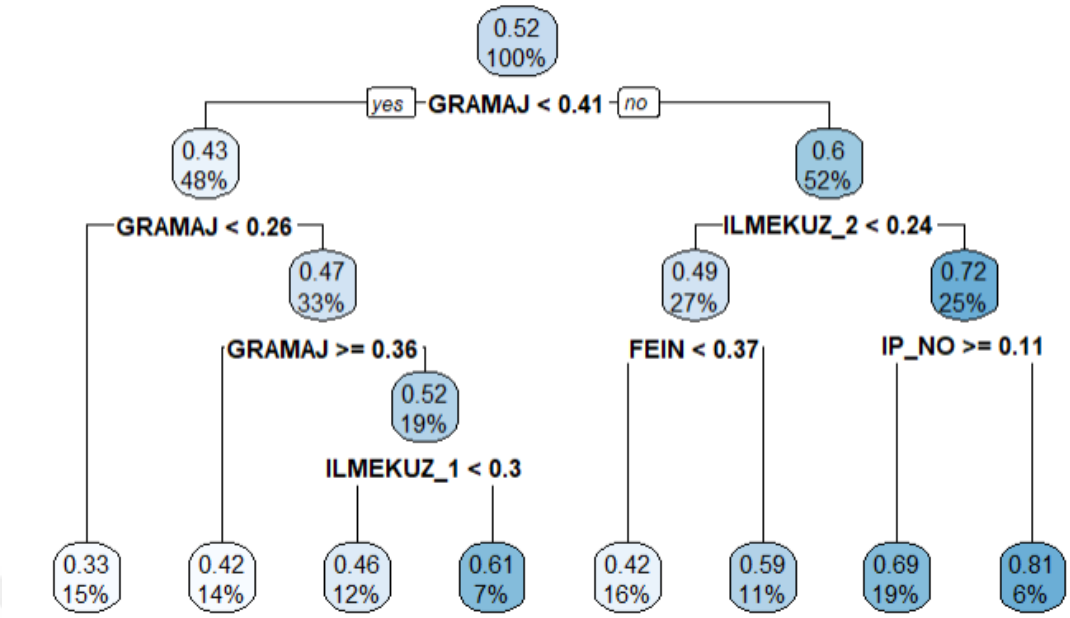


Şekil 4.7 Yapay sinir ağı model sonuçlarının karşılaştırılması

4.2.2 Topluluk Öğrenme Modellerin Oluşturulması ve Test Edilmesi

Temeli birden çok karar ağacının ürettiği tahminlerin bir araya getirilerek değerlendirilmesine dayanan topluluk öğrenme yöntemlerinden regresyon ağacı, bagging regresyon ağaçları, rassal ormanlar, gbm ve xgboost modelleri incelenmiştir.

İlk olarak topluluk öğrenme modellerinin temeli olan regresyon ağaçları ile model kurulmuştur. Kurulan model Şekil 4.8'de yer almaktadır. En önemli özellik yani kök düğüm kumaşın gramajı olarak belirlenmiştir. Gramaj dışında 2. ilmek iplik uzunluğu, fein 1. ilmek iplik uzunluğu ve iplik numarası değişkenleri de önemli özellikler arasında yer almıştır.



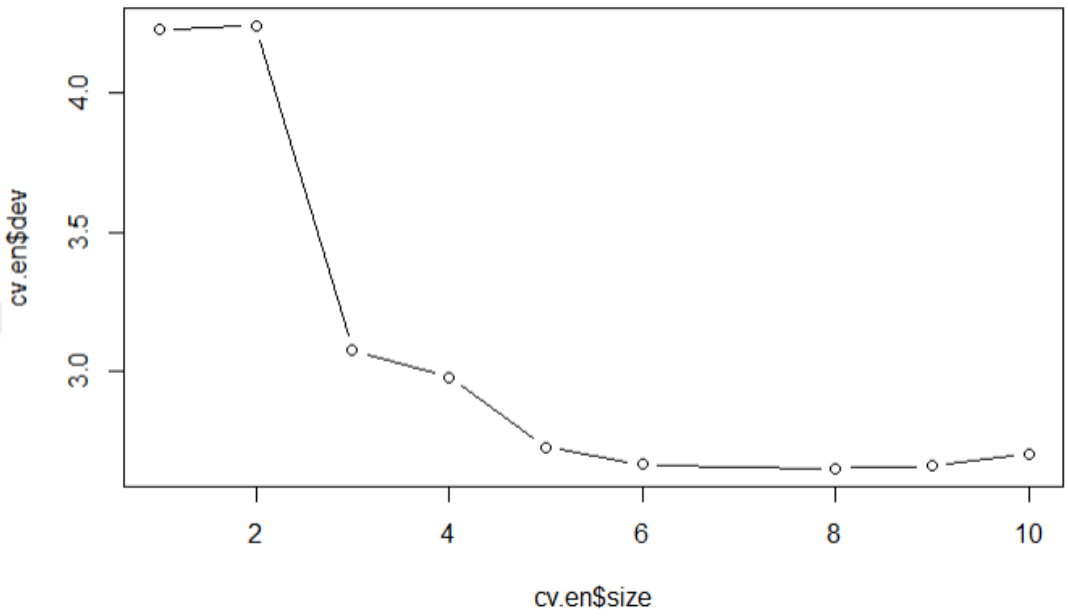
Şekil 4.8 Regresyon ağacı modeli

Regresyon ağacına göre kumaşların %15'inin ölçeklendirilmiş en değerleri 0.33, %14' ünün 0.42, %12' sinin 0.46, %7' sinin 0.61, %16' sının 0.42, %11' inin 0.59, %19' unun 0.69, %6' sının 0.81 olarak tahmin edildiği görülmektedir.

Modelin performasını değerlendirmek üzere ağaç budama yöntemlerine başvurulmuştur. Çapraz geçerlilik (cross validation), mevcut veri setini eğitim ve test kümelerine ayırma işlemini tekrarlayarak modelin genel performansını ölçer. Veri seti belirli bir sayıda parçaya bölünür ve her bir parça sırayla test seti olarak kullanılırken, diğer parçalar eğitim seti olarak kullanılır. Bu işlem, veri setinin her bir parçası üzerinde modelin nasıl performans gösterdiğini değerlendirmek için yapılır.

Çapraz geçerlilik yönteminde, sapma değerini hesaplamak için her bir iterasyonda eğitilen modellerin performansı değerlendirilir. Bu değerlendirme sürecinde sapma, modelin tahminlerinin gerçek değerlere ne kadar iyi uyduğunu ölçmek için kullanılabilir. Sapma değeri, genellikle ağaç tabanlı modellerde kullanılan bir istatistiksel ölçüdür. Daha düşük bir sapma değeri, daha iyi bir uyum ve daha az hata anlamına gelir. Ağaç budama işlemi yapılırken R programında genellikle sapma değerine bakılır. Budama işlemi sırasında, ağacın dalları kırpılarak veya düğümler

birleştirilerek ağaç yapısı sadeleştirilir. Sapma değeri en düşük olan budama noktası seçilir ve ağaç o noktada kırpılır veya birleştirilir. Şekil 4.9'da çapraz geçerlilik yöntemi ile sapma değerleri gösterilmiştir. En düşük sapma değerinin olduğu düğüm sayısı dikkate alınarak ağaç budanmalıdır.



Şekil 4.9 Çapraz geçerlilik ile sapma değerleri

Terminal node sayısı 6 olduğu durumda deviance değeri en düşük çıkmaktadır ve ağaç terminal node sayısı 6 olacak şekilde budanmıştır. Model sonuçları tablo 4.8'de gösterilmiştir.

Tablo 4.8 Regresyon ağacının değerlendirme sonuçları

RMSE	**MAE**	**MSE**	**R^2**
10,74	7,58	115,46	0,43

Regresyon ağacından bagging yöntemi ile regresyon ağaçları geçiş, daha iyi performans elde etmek amacıyla yapılan bir modelleme yaklaşımıdır. Bagging, birden çok karar ağacının bir araya getirilerek daha güçlü ve kararlı bir model oluşturulmasını sağlar. Bu yöntem, aşırı öğrenmeyi azaltır, modelin genelleme yeteneğini artırır ve

daha güvenilir tahminler yapılmasına yardımcı olur. Bu nedenle bagging yöntemi ile model kurulup test edilmiştir sonuçları Tablo 4.9'daki gibidir:

Tablo 4.9 Bagging yöntemi ile regresyon ağaçlarının değerlendirme sonuçları

RMSE	**MAE**	**MSE**	**R^2**
7,72	5,77	59,61	0,71

Sonuçlar kıyaslandığında bağımsız değişkenlerin bağımlı değişkeni açıklama yüzdeliği %43'den , %71'e çıkmıştır. Aynı zamanda bagging yöntemi ile regresyon ağaçlarında performans metrikleri açısından daha başarılı sonuçlar vermiştir.

Topluluk öğrenme yöntemlerinden rassal ormanlar ile de model kurulup sonuçları kıyaslanarak incelenmiştir. Rassal ormanlar ve bagging regresyon ağaçları için R programında "randomForest" paketi kullanılmıştır. Rassal ormanlar da tek farkı kullanılan argümanlardan "mtry" parametresinin farklı verilmesidir. Rassal Ormanlar, bagging yönteminden farklı olarak, her bir ağaç oluşturulurken rastgele örnekleme (bootstrap) ve rastgele özellik seçimi yapar. Bu nedenle model kurulurken mtry değerine özellik sayısının 3'e bölünmesiyle elde edilen sayı yazılır. Hata metriklerine göre sonuçları Tablo 4.10' daki gibidir:

Tablo 4.10 Rassal ormanlar yönteminin değerlendirme sonuçları

RMSE	**MAE**	**MSE**	**R^2**
8,57	6,45	73,39	0,64

Sonuçlar kıyaslandığında rassal ormanlar yönteminde bagging yöntemi ile regresyon ağaçları yöntemine göre hata metrikleri açısından daha düşük değerler yani daha zayıf performans sonuçları elde edilmiştir.

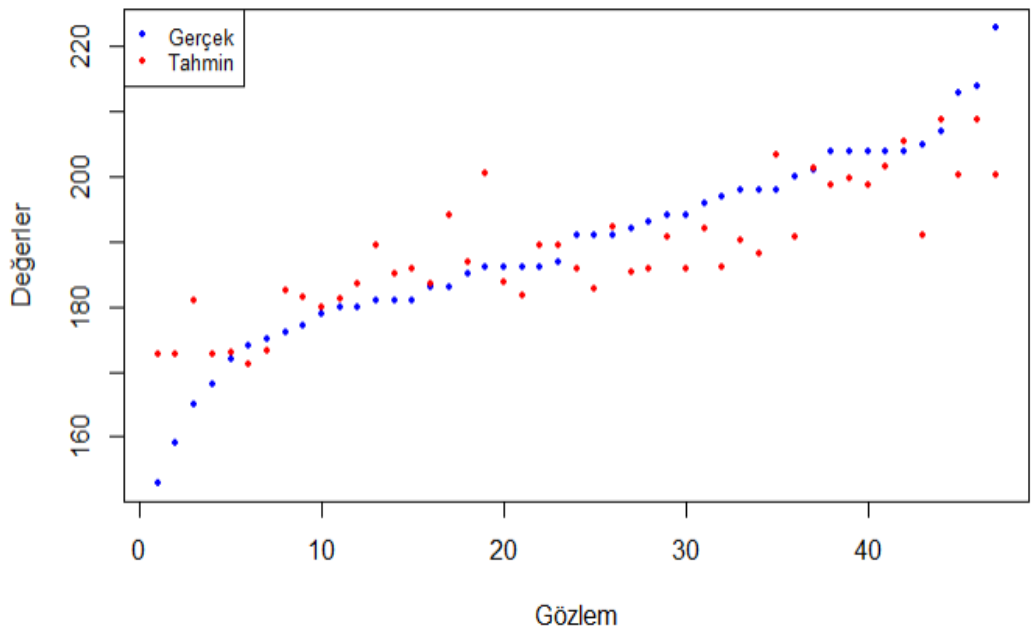
Bir diğer topluluk öğrenme yöntemlerinden gradyan artırma regresyonu (GBM), rassal ormanlara kıyasla daha fazla esneklik ve model performansını artırma yeteneği

sunar. Ardışık eğitim süreci, hataları düzeltmek için optimize edilmiş bir kayıp fonksiyonu kullanır. Bu şekilde, GBM daha iyi tahmin yeteneklerine sahip olabilir. Bu nedenle GBM yöntemi ile model kurulup test edilmiştir sonuçları Tablo 4.11'deki gibidir:

Tablo 4.11 Gradyan artırma regresyonunun değerlendirme sonuçları

RMSE	**MAE**	**MSE**	**R^2**
8,14	6,33	66,27	0,68

Sonuçlar kıyaslandığında gradient boosting yöntemi rassal ormanlara göre hata metrikleri açısından daha düşük, R^2 değeri bağımsız değişkenlerin bağımlı değişkeni açıklama yüzdesi açısından daha başarılı sonuç vermiştir. Test verisinde gerçek değerlerin tahmin değerlerinden farkları Şekil 4.10'da gösterilmiştir. Kırmızı noktalar tahmin değerlerini, mavi noktalar ise gerçek değerleri temsil eder.



Şekil 4.10 GBM yönteminde gerçek sonuçların tahmin değerleri ile karşılaştırılması

Son olarak topluluk öğrenme yöntemlerinden XGBoost yöntemi, gradyan artırma regresyonunun bir geliştirilmiş versiyonudur ve genellikle daha iyi performans sağlaması nedeniyle tercih edilen bir geçiş modelidir. XGBoost, GBM'nin temel prensiplerini benimserken, bir dizi yenilik ve iyileştirme getirir. Bu nedenle XGBoost yöntemi ile model kurulup test edilmiştir sonuçları Tablo 4.12'deki gibidir:

Tablo 4.12 XGBoost değerlendirme sonuçları

RMSE	**MAE**	**MSE**	**R^2**
7,54	5,56	56,92	0,72

Sonuçlar kıyaslandığında XGBoost yöntemi gradyan artırma regresyonu yöntemine göre hata metrikleri açısından daha düşük, R^2 değeri bağımsız değişkenlerin bağımlı değişkeni açıklama yüzdesi açısından daha başarılı sonuç vermiştir. Topluluk öğrenme yöntemlerinde kurulan modellerin genel sonuçlarına bakıldığında en iyi model XGBoost modelidir. Tablo 4.13'de topluluk öğrenme yöntemlerinin kıyaslanması yer almaktadır.

Tablo 4.13 Topluluk öğrenme yöntemleri ile kurulan modellerin değerlendilmesi

Modeller	**RMSE**	**MAE**	**MSE**	**R^2**
Model_tree	10,74	7,58	115,46	0,43
Model_brt	7,72	5,77	59,61	0,71
Model_rfr	8,57	6,45	73,39	0,64
Model_gbm	8,14	6,33	66,28	0,68
Model_xg	7,54	5,56	56,92	0,72

BÖLÜM BEŞ
DEĞERLENDİRME VE SONUÇ

Kumaş endüstrisinde, kumaşın eninin doğru bir şekilde tahmin edilmesi büyük bir avantaj sağlar. Doğru en tahmini, üretim sürecinde verimlilik, maliyet tasarrufu ve kalite kontrolü açısından önemlidir. İlk olarak, malzeme israfını önler. Yanlış en tahminleriyle kesilen fazla kumaş parçaları atık olarak kalır ve maliyetleri artırır. Doğru en tahminiyle, kumaş israfı azalır ve malzeme verimliliği artar. İkinci olarak, doğru en tahmini, üretim sürecinde zaman ve iş gücü tasarrufu sağlar. Kumaşın doğru bir şekilde kesilmesi ve dikilmesi için ölçülere uygunluğun sağlanması gerekir. Yanlış en tahminleriyle, kumaşın yeniden kesilmesi veya uygun şekilde düzeltilmesi gerekebilir, bu da zaman ve iş gücü kaybına neden olur. Doğru en tahminiyle, süreçteki hatalar ve düzeltmeler azalır, zaman ve iş gücü verimliliği artar. Son olarak, doğru en tahmini, ürün kalitesini ve müşteri memnuniyetini artırır. Yanlış en tahminleriyle üretilen kumaşlar, uygun olmayan şekilde kullanılabilir veya hatalı ürünlerin üretilmesine neden olabilir. Doğru en tahminiyle, ürünlerin doğru ölçülerde üretilmesi ve müşteri taleplerine uygun şekilde sunulması sağlanır. Tüm bu avantajlar göz önüne alındığında, kumaş endüstrisinde doğru en tahminlenmesi büyük bir öneme sahiptir.

Bu çalışmada süprem türündeki örme kumaşlarda 13 bağımsız değişken ile gerçek zamanlı 162 adet gözlemin olduğu veri setinde yapay sinir ağları ve topluluk öğrenme modelleri kurulmuş sonuçlar MSE, RMSE, MAE ve R^2 performans metrikleri ile karşılaştırılmıştır. İşletmenin belirlediği kabul kriterine göre de modellerin kıyaslaması Tablo 4.14'de yer almaktadır. Tüm modeller hem hata metrikleri hemde kabul yüzdelikleri açısından kıyaslandığında yapay sinir ağları ile kurulan ilk modelin en iyi sonucu verdiği gözlemlenmiştir.

Tablo 4.14 Tüm model sonuçlarının kabul edilebilirlik yüzdeliklerine kıyaslanması

Modeller	**RMSE**	**MAE**	**MSE**	**R^2**	**Kabul Yüzdelikleri**
Model_tree	10,74	7,58	115,46	0,43	%72
Model_brt	7,72	5,77	59,61	0,71	%79
Model_rfr	8,57	6,45	73,39	0,64	%77
Model_gb	8,14	6,33	66,28	0,68	%81
Model_xg	7,54	5,56	56,92	0,72	%83
Model_ysa1	4,98	4,07	24,77	0,88	%94
Model_ysa2	5,95	4,72	35,45	0,83	%83
Model_ysa3	5,74	4,45	32,94	0,84	%89

Sonuç olarak bu çalışma yapay sinir ağları, tekstil sektöründe kumaş eni tahminlemesi gibi karmaşık problemlerin çözümünde daha iyi sonuçlar vermiştir. Bu uygulama kumaş eni, birçok faktörün etkileşimiyle belirlenen karmaşık bir özelliktir. Yapay sinir ağlarının geniş veri setleri üzerinde etkili çalışabilme, gizli desenleri yakalama yeteneği ve esnek parametre ayarları gibi avantajları, tekstil sektöründe tercih edilebilir.

KAYNAKLAR

Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test, 25*(2), 197-227.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123-140. https://doi.org/10.1007/BF00058655

Breiman, L. (2001). Random forests. *Machine learning, 45*(1), 5-32.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* ,785-794. https://doi.org/10.1145/2939672.2939785

Chen, T., & Guestrin, C. (2016). *XGBoost Documentation*. XGBoost GitHub Repository. Retrieved from https://github.com/dmlc/xgboost

Ciaburro, G. ve Venkateswaran, B. (2017). *Neural network with R*. Packt Publishing Ltd. Livery Place

Dietterich, T. G. (2002). Ensemble learning. *The handbook of brain theory and neural networks, 2*, 110-125.

Freund, Y., & Schapire, R. E. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence, 14*(5), 771-780.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics, 29*(5), 1189-1232. DOI: 10.1214/aos/1013203451

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

Haykin, S. (2009). *Neural Networks and Learning Machines (3rd ed.)*. Prentice Hall.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R.* Springer Science & Business Media.

Khandelwal, A., Jha, D., & Singh, S. K. (2018). A comprehensive review on random forest algorithm. *Journal of King Saud University-Computer and Information Sciences*, *30*(4), 395-407.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

Kuncheva, L. I. (2014). *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R News*, *2*(3), 18-22.

Li, Z., Huang, J., Li, S., Wang, C., & Li, W. (2017). A novel ensemble model based on bagging regression trees for imbalanced data classification. *Journal of Ambient Intelligence and Humanized Computing*, *8*(6), 917-927. https://doi.org/10.1007/s12652-017-0511-9

Loh, WY (2011). Classification and Regression Trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*, 14-23. https://doi.org/10.1002/widm.8

Müller, A. C., & Guido, S. (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Inc.

Rocca, J. (23 Nisan 2019). *Ensemble methods: bagging, boosting and stacking.* https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323(6088),* 533-536.

Wolpert, D. H. (1992). Stacked generalization. *Neural networks, 5*(2), 241-259.

Wu, Z., Li, J., Wang, C., Liu, X., & Yin, H. (2019). An improved bagging regression tree algorithm based on mutual information and chaotic mutation. *IEEE Access, 7*, 22538-22548. https://doi.org/10.1109/ACCESS.2019.2890374

Zhang, Y., Lin, X., Li, Y., Xiong, H., & Yu, P. (2018). High-dimensional regression using Bagging regression trees. *Journal of Big Data, 5*(1), 1-18. https://doi.org/10.1186/s40537-018-0127-3

Zhang, Y., Zhu, X., Song, X., & Dai, Q. (2018). A new approach of bagging regression tree algorithm based on feature subsampling. *Journal of Intelligent & Fuzzy Systems, 35*(2), 1707-1717.

Zhou, Z. H., Wu, J., & Tang, W. (2012). Ensembling neural networks: Many could be better than all. *Artificial intelligence, 137(*1-2), 239-263.