GÖRÜNTÜ SINIFLANDIRMADA DERİN KONVOLÜSYONEL SİNİR AĞLARI İLE İSTATİSTİKSEL YÖNTEMLERİN KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Ece AYDOĞDU

Eskişehir 2023

GÖRÜNTÜ SINIFLANDIRMADA DERİN KONVOLÜSYONEL SİNİR AĞLARI İLE İSTATİSTİKSEL YÖNTEMLERİN KARŞILAŞTIRILMASI

Ece AYDOĞDU

Yüksek Lisans Tezi

İstatistik Anabilim Dalı

İstatistik Teorisi Bilim Dalı

Danışman: Doç. Dr. Özer ÖZDEMİR

Eskişehir

Eskişehir Teknik Üniversitesi

Lisansüstü Eğitim Enstitüsü

Nisan 2023

JÜRİ VE ENSTİTÜ ONAYI

Ece AYDOĞDU'nun "Görüntü Sınıflandırmada Derin Konvolüsyonel Sinir Ağları ile İstatistiksel Yöntemlerin Karşılaştırılması " başlıklı tezi 24/04/2023 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Eskişehir Teknik Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği"nin ilgili maddeleri uyarınca, İstatistik Anabilim dalında Yüksek Lisans Tezi olarak kabul edilmiştir.

Jüri Üyeleri	**Unvan Adı Soyadı**	**İmza**
Üye (Tez Danışmanı)	: Doç. Dr. Özer ÖZDEMİR	
Üye	: Dr. Öğr. Üyesi Gültekin ATALIK	
Üye	: Dr. Öğr. Üyesi Mustafa ÇAVUŞ	

Prof. Dr. Murat TANIŞLI
Lisansüstü Eğitim Enstitüsü Müdürü

24/04/2023

DANIŞMAN ONAYI

Danışmanlığını yürüttüğüm İstatistik Anabilim Dalı Yüksek Lisans öğrencisi Ece AYDOĞDU, "Görüntü Sınıflandırmada Derin Konvolüsyonel Sinir Ağları ile İstatistiksel Yöntemlerin Karşılaştırılması" başlıklı tez çalışmasını tamamlamıştır. Hazırlamış olduğu tez tarafımca incelenmiş ve öğrencinin tez savunma sınavına alınması bilimsel ve etik açıdan uygun görülmüştür.

Tez Danışmanı
Doç. Dr. Özer ÖZDEMİR

ÖZET

GÖRÜNTÜ SINIFLANDIRMADA DERİN KONVOLÜSYONEL SİNİR AĞLARI İLE İSTATİSTİKSEL YÖNTEMLERİN KARŞILAŞTIRILMASI

Ece AYDOĞDU

İstatistik Anabilim Dalı

İstatistik Teorisi Bilim Dalı

Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Nisan 2023

Danışman: Doç. Dr. Özer ÖZDEMİR

Yapay sinir ağları makine öğrenmesi alanında birçok problem için kullanılmıştır. 2000'lerin başına kadar çok üzerinde durulmamış fakat 2000'lerde tekrar göz önüne gelmiştir. Bilgisayar sistemlerinin gelişimiyle beraber derin ağlara geçilmiştir. Görüntü işlemeden, medikal uygulamalara kadar birçok alanda yeri mevcuttur. Özellikle görüntü işlemede birçok yardımcı algoritma ile kullanılmaktadır. Makine öğrenmesi alanında yapay sinir ağları birçok problemin çözümünde sıklıkla kullanılmıştır. Aynı şekilde ortaya çıkan ve gelişime açık istatistiksel yöntemler de görüntü sınıflandırma da yer alır. Özellikle regresyon analizi yöntemleri yaygın olarak kullanılmaktadır. Bu çalışmada, konvolüsyonel sinir ağlarının tarihçesi, görüntü sınıflandırmadaki yeri, istatistiksel görüntü sınıflandırma yöntemleri ve konvolüsyonel sinir ağları ile istatistiksel sınıflandırma yöntemleri yapılan uygulama sayesinde karşılaştırılmıştır. Bu uygulamanın amacı hangi yöntemin daha etkili olabileceğiyle ilgilidir.

Anahtar Sözcükler: Derin öğrenme, Konvolüsyonel sinir ağları, Görüntü sınıflandırma, İstatistiksel sınıflandırma yöntemleri.

ABSTRACT

COMPARISON OF DEEP CONVOLUTIONAL NEURAL NETWORKS AND STATISTICAL METHODS IN IMAGE CLASSIFICATION

Ece AYDOĞDU

Department of Statistics

Programme in Theory of Statistics

Eskisehir Technical University, Institute of Graduate Program, April 2023

Supervisor: Assoc. Prof. Özer ÖZDEMİR

Artificial neural networks have been used for many problems in the field of machine learning. It was not emphasized much until the early 2000s, but it came to the fore again in the 2000s. With the development of the computer systems, it has moved to deep networks. It has a place in many fields from image processing to medical applications. It is used with many auxiliary functions especially in image processing. Artificial neural networks have been used to solve many problems in the field of machine learning. It is also included in the image concept through the extensions that emerge and are open to development. Especially regression analysis methods are widely used. This situation has been compared with the history of convolutional neural networks, their place in the image world, the expected image expectations methods and the application of convolutional neural networks and forcing methods. This operational intent is linked to which method may be more effective.

Keywords: Deep learning, Convolutional neural networks, Image classification, Statistical classification methods.

24/04/2023

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Eskişehir Teknik Üniversitesi tarafından kullanılan "bilimsel intihal tespit programı"yla tarandığını ve hiçbir şekilde "intihal içermediğini" beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

Ece Aydoğdu
(imza)

TEŞEKKÜR

Bu tez çalışmasında görsel sınıflandırma tekniklerinin doğruluk bakımından karşılaştırılması istenmiştir**.** Bu doğrultuda bir uygulamaya yer verilmiştir.

Tezimle ilgili her türlü konuda bana yardımcı olan tez danışmanım Doç. Dr. Özer Özdemir'e teşekkürlerimi sunarım. Bu zorlu tez sürecinde benden bir an için bile desteğini esirgemeyen arkadaşlarıma, tüm eğitim hayatım boyunda daima yanımda olan ve maddi manevi desteklerini esirgemeyen değerli aile üyelerime teşekkürlerimi borç bilirim.

Ece Aydoğdu

Nisan 2023

İÇİNDEKİLER

Sayfa

TABLOLAR DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

1. GİRİŞ

Son birkaç yıldır değişik kaynaklar aracılığıyla büyük sayılarda veriler toplanmaktadır. Bu verileri işlemek ve makine öğrenmesi yoluyla kullanılabilir hale getirmek için oldukça fazla yöntem geliştirilmiştir. Bunlardan biri de 20. yüzyılın ortalarında insan beynine olan benzerliği ile öne çıkan ve hayatımızı büyük ölçüde değiştirecek olan yapay sinir ağları (YSA) yöntemleridir. YSA, görüntü sınıflandırmada çok farklı alanlarda kullanılabilen bir makine öğrenmesi yöntemidir. Bu alanlara sağlık, bilim, otomotiv gibi alanlar dahildir. Günlük çekilen fotoğraf ve videolar, sosyal medya üzerinden yapılan yorumlar, video izleme platformlarından izlenen videolar, hastanelerde yapılan testler, çevrim içi alışveriş sitelerinden yapılan işlemler gibi çalışmalar daha sonra makine öğrenmesi modellerinde kullanılmak üzere istatistiksel kaynaklardan da yararlanılmaktadır [26]. Makine öğrenmesi, istatistiksel yöntemleri de kullanabilir ve geliştirir bu sayede uygulamalardaki verilerin bir sonuca bağlanılmasını sağlar böylece geleceği öngörülebilir hale getirebilir. Bu istatistiksel yöntemler içerisinde görüntü sınıflandırmada kullanılmak üzere regresyon analizi modelleri, bulanık mantık veya karar ağaçları gibi yöntemleri söyleyebiliriz (http-1).

Makine öğrenmesi ne kadar iyi eğitilirse öngörülebilirlik o kadar iyi arttırılır. Herhangi bir yöntemin kullanılması sırasında, yöntem kadar içeriğine uyum sağlayan fonksiyonlarda önemlidir. İyi bir fonksiyon ve yöntemle uygulamanın doğruluk oranı artar. Mesela parmak izini tanıyabilen bir modelde, kişiyi tanıması beklenir. Veya çiçek dokularına ait verilerin kullanıldığı bir modelde, uygulamanın verileri yaprak boyutları, rengi vb. özelliklerine göre iyi sınıflandırması beklenir. Bu sınıflandırmaların doğru yapılabilmesi için daima en uygun yöntemin saptanması gerekir.

En uygun yöntemin bulunabilmesi için, çalışmaya uygun makine öğrenmesi yöntemlerinin incelenmesi, nihayetinde homojen verilerle karşılaştırılması gerekebilir.

Bu bağlamda yapılan literatür taraması sırasında özellikle yapay zeka ve görüntü sınıflandırmayla ilgili oldukça fazla uygulama olduğu tespit edilmiştir. Bu uygulamalar ile beraber bilinen yöntemler ayrı ayrı veya karşılaştırılmalı olarak ele alınmıştır.

Ayhan, Karslı ve Tunç, çalışmalarında, uzaktan algılanmış görüntülerin sınıflandırmasında kullanılan iki genel yaklaşımı, denetimsiz ve denetimli sınıflandırma

yöntemlerini tanıtmıştır. Eğitimli ve eğitimsiz sınıflandırma yöntemlerini karşılaştırmış ve bu sınıflandırma modellerinin her birinin, barındırdıkları özelliklere göre daha kullanılabilir olduğunu sonuçlandırmışlardır. Sınıflandırılmak istenen piksel, olasılığı en fazla olan sınıfa atanır. Örneğin, En az uzaklık ve paralel kenar yöntemi ile karşılaştırıldığında, en yüksek olasılık sınıflandırma yöntemi en az hata oranını verdiğini saptamışlardır [2].

Yiğit ve Uysal, uygulama alanı olarak; Kütahya 1:250.000'lik sınırları içerisinde kalan ormanlık alan açısından yoğun olan 1:25.000 ölçekli I22-b1 paftasından alınan uydu görüntüleriyle yaptıkları çalışmada, piksel tabanlı sınıflandırma yönteminin yanlış sınıflandırmaya sebep olabileceğini saptamışlardır [11].

Saberioon, Císaˇr ve Labbé, makalelerinde görüntü tabanlı özellikleri kullanarak balıkları kültür sırasında diyetlerine göre değerlendirmek için destek vektör makineleri, rassal orman, lojistik regresyon ve k en yakın komşu yaklaşımlarını analiz edip karşılaştırmışlardır. Karmaşık modellerin basit modellere göre daha iyi sınıflandırıcılar olduğunu saptamışlardır. Bu yüzden karmaşık modelleri önermişlerdir [23].

Wang ve diğerleri [25], çalışmalarında oftalmik görüntüleri otomatik olarak tanımlamak için, görüntü özelliği çıkarma ve görüntü sınıflandırması adına sekiz temsili yöntem uygulamışlardır. Bu çalışmada, elde edilen sekiz şemanın performansı birçok açıdan karşılaştırılmıştır. Hangi özellik çıkarma yöntemi benimsenirse benimsensin, Destek Vektör Makinelerinin (DVM) sınıflandırmada aldığı sonuç, diğer kNN (k-en yakın komşu) yaklaşımı gibi bazı geleneksel yöntemler de nispeten tatmin edici sonuçlar verebilmiştir. ELM (aşırı öğrenme makinesi) uygulandığında, diğer sinir ağı mimarilerinden daha hızlı olmasına rağmen, sınıflandırma yapıldığında beklenen sonuçları verememiştir. ELM'ye benzer olarak, seyrek tabanlı yöntemlerde bu sorunu çözmede iyi sonuç vermemiştir.

Dovbnych ve Wójcik, konvolüsyonel sinir ağları (KSA) ve geleneksel sinir ağı mimarileri arasında performans karşılaştırması yapmışlardır [12].

Bu tez için yapılan araştırma sırasında, makine öğrenimi için kullanılan yöntemler karşılaştırıldığında, KSA'ların aynı verilere uygulanan diğer istatistiksel yöntemlere

göre, doğruluk ve veri kaybı açısından en kullanılabilir sınıflandırma yöntemi olduğuna karar verilmiştir.

En iyi yöntemin saptanması için aynı verilerin kullanıldığı ve çoğu istatistiksel görüntü sınıflandırma yöntemleriyle konvolüsyonel sınıflandırma yöntemine yer verilen birkaç uygulamaya yer verilmiştir. Doğruluk analizlerinden sonra hangi yöntemin daha kullanılabilir olduğu saptanmıştır. Bu yöntemlerin sağlanan görsel ve sayısal veriler ile yapılacak analizleri için, Python programlama dili kullanılmıştır.

Çalışma yedi bölümden oluşmaktadır ve ikinci bölümde YSA genel olarak bahsedilmektedir. Bu bağlamda YSA'nın gelişimi, geleneksel sistemlerden farkları incelenmiş, üstünlükleri ve uygulama alanları hakkında bilgiler verilmiş ve KSA'dan bahsedilmiştir.

Çalışmanın 3. ve 5. Bölümleri arasında ise görüntü sınıflandırmada genel olarak kullanılan istatistiksel yöntemlere değinilmiş ve 6. Bölüm olan uygulama kısmında sırasıyla bu yöntemlerle ilgili analizler yapılmış ve daha sonrasında da 7. Bölümde sonuçlara yer verilmiştir

2. YAPAY SİNİR AĞLARI

YSA, bilgileri öğrenme ve yardım almadan çoğaltabilmek için geliştirilen bir analiz sistemidir. Hayatımıza ilk olarak 1943'te nörofizyolog Warren McCulloch ve matematik dehası Walter Pitts'in beyin nöronlarının nasıl çalıştığını elektrik devreleriyle modellemeleriyle girmiştir. Daha sonra 1949'da Donald Hebb bu sinir ağlarının kullanıldıkça güçlenmesi ile ilgili bir makale yazmıştır [29]. 1950'lerden sonra ise ilerleyen teknolojiyle beraber bu ağların modellenme sürecine girilmiş oldu. 1959'da ADALINE (Delta kuralı) ve MADALINE geliştirildi. MADALINE günümüzde YSA'nın ilk gerçek sinir ağıdır. 1969'dan sonra geliştirilmeye devam etmiş fakat bu gelişme devam ederken çok katmanlı ağlar hakkında yazılan eleştirel yazılar sonucunda ilgiyi büyük ölçüde kaybetmişti fakat 1970'li yıllar YSA için bir dönüm noktasıdır diyebiliriz. Buna yönelik, 1972'de Kohonen ve Anderson bir ağ geliştirmişlerdir [29]. Ve ilk çok katmanlı denetimsiz ağ 1975'te geliştirildi. 1984 yılında Kohonen danışmansız öğrenmenin mantığını ortaya koymuştur. 1986 yılında ise Rumelhart ve McClelland çok katmanlı ağ yapılarında geri yayılımlı ağ algoritmasını ortaya koymuşlardır [29]. 1987'de ise IEEE (elektrik elektronik mühendisliği enstitüsü) tarafından YSA'nı konu alan ilk konferans yüksek bir katılımla gerçekleştirilmiştir. O zamandan beri gelişerek her alanda kullanılmaya devam etmektedir.

YSA sınıflandırma, analiz, regresyon, optimizasyon gibi birçok alanda kullanılabilir.

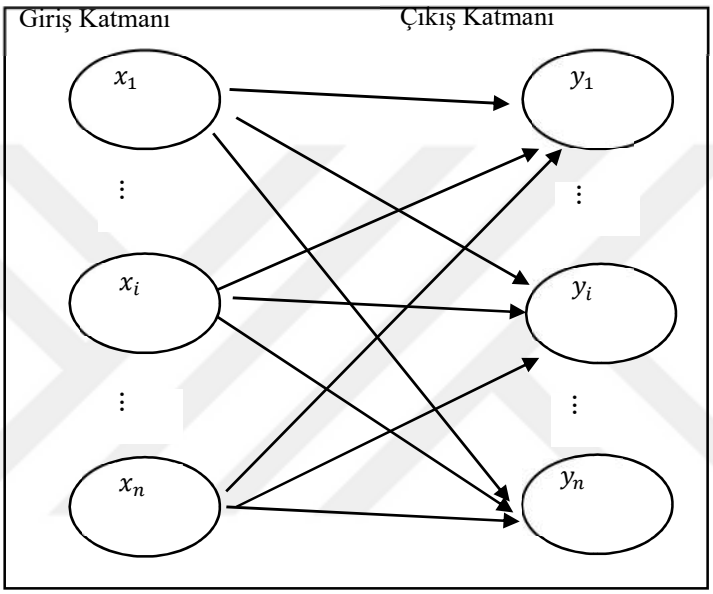
YSA yapı olarak biyolojik sinir hücrelerine benzemektedir. Model içerisindeki nöronlar bir sonraki katmanda bulunan nöronlar ile bağlantı oluşturarak ağ meydana getirmektedir. YSA içerisinde 3 adet katman bulundurmaktadır. Bu katmanlar; girdi katmanı, gizli katman ve çıktı katmanıdır. Bu katmanlarda öğrenme gerçekleşir. Bu öğrenmenin gerçekleşmesi için ise aktivasyon fonksiyonlarına ihtiyaç vardır. Bilgi girdi katmanında alınır, aktivasyon fonksiyonlarıyla gizli katmanda öğrenilir ve çıktı katmanında öğrenilen bilginin analiz edildiğini görebiliriz [27].

2.1. Yapay Sinir Ağlarının Yapısı

YSA'da bağlantının cinsine ve nöronların durumlarına göre YSA farklı şekillerde adlandırılmaktadır. Mimarileri ise bağlantının akış yönüne göre farklılaşır. Akış yönüne

göre incelendiğinde ileri beslemeli ve geri beslemeli ağlardan söz edilmektedir. Yapısı tek katmanlı (perceptron) ağlar ve çok katmanlı ağlar olmak üzere ikiye ayrılır.

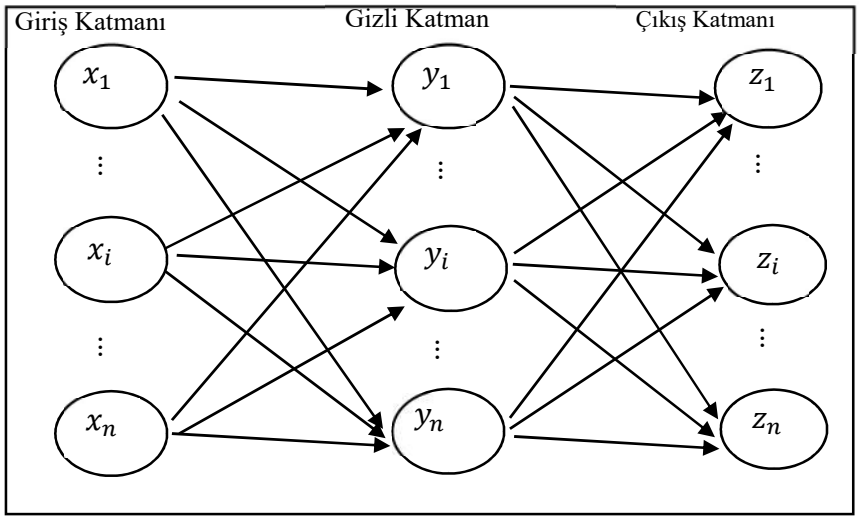
Tek katmanlı sinir ağlarda gizli katman yoktur sadece girdi ve çıktı katmanları bulunur. Karmaşık problemlere cevap verebilme yeteneğine sahip değildirler. Bu yüzden problem çözme konusunda oldukça sınırlıdırlar. Daha çok doğrusal olmayan problemlerde kullanılırlar. Tek katmanlı ağlarda eşik değeri oldukça önemlidir. Bu değer ağın iyi bir sınıflandırma yapmasına yardımcı olmaktadır.



Şekil 2.1. *Tek katmanlı sinir ağı (http-6)*

Çok katmanlı ağlar kontrollü öğrenme yöntemini kullanan ve tek katmanlı ağların çözemediği daha karmaşık ve doğrusal olmayan XOR gibi problemleri çözebilen ağlardır. Bu modele hatayı geriye yayma modeli de denilmektedir (http-6). Bu ağların asıl amacı öğrenme sırasında oluşan hatayı en aza indirgemektir. Çok katmanlı ağlar girdi, çıktı ve bu iki katman arasındaki bağlantıyı sağlayan diğer ara katmanlardan (gizli katman) oluşmaktadır.

Delta öğrenme kuralı denilen bir öğrenme yöntemiyle kullanılmaktadır. Bu öğrenme kuralında ağın çıktısı ileri doğru hesaplama ile hesaplanır ve geriye doğru hesaplama ile de ağırlıklar güncellenir.



Şekil 2.2. *Çok katmanlı ağı (http-6)*

İleri doğru hesaplamada, eğitim setinde bulunan verinin girdi katmanından ağa sunulması ile başlar. Girdi katmanından k. Proses elemanının çıktısı:

$$G_k = C_k \tag{2.1}$$

Ara katmanda bulunan tüm elemanlar kendinden önceki iletilen bilgileri bağlantı ağırlıklarıyla beraber alır. Ara katmana iletilen net girdi:

$$Net\ _j^{\ a} = \sum_{k=1}^{n} W_{kj} * Ç_k^i \tag{2.2}$$

j. ara katmanın çıktısı net girdinin aktivasyon fonksiyonu sonucu ile hesaplanmaktadır. Belirlenen aktivasyon fonksiyonu sigmoid ise:

$$Ç_j^a = \frac{1}{1+e^{-(Net\ _j^{\ a} + b\ _j^{\ a})}} \tag{2.3}$$

Geriye doğru hesaplamada, ağa sunulan girdi ve çıktısı ile gerçek sonuç karşılaştırılır. Bunun sonucunda fark hata değeri ortaya çıkmaktadır. Burada amaç hatayı en aza düşürmektir. Hata ağırlık değerlerine her seferde dağıtılır. m. işlem için hesaplanan hata:

$$E_m = B_m - Ç_m \tag{2.4}$$

Toplam hata ise:

$$TH = \frac{1}{2}\sum_m E_m^2 \tag{2.5}$$

Toplam hatanın 0 olması için tüm çıktı hatalarının kareleri toplanarak elde edilen sonucun karekökü alınır.

2.2. Yapay Sinir Ağlarında Eğitim

Bağlantıların ağırlık değerlerinin optimum değerinin bulunmasına ağ öğrenmesi denir. Ağın belirli bilgileri çıkararak bilinmeyen örneklere yorum getirmesine, adaptif öğrenme denir.

Burada öğrenme iki aşamalıdır. Birinci aşamada ağa gösterilen örnek için ağın üreteceği çıkış belirlenir. İkinci aşamada ise ağın bağlantılarının sahip olduğu ağırlıklar güncellenmektedir. Ağın eğitiminin tamamlanmasından sonra performans ölçümü için test edilmelidir. Test için ağa daha önce görmediği örnekler verilir ve bu örneklere çözüm üretilmesi beklenir. Test aşamasında ise ağırlıklar güncellenmez.

2.3. Konvolüsyonel Sinir Ağları

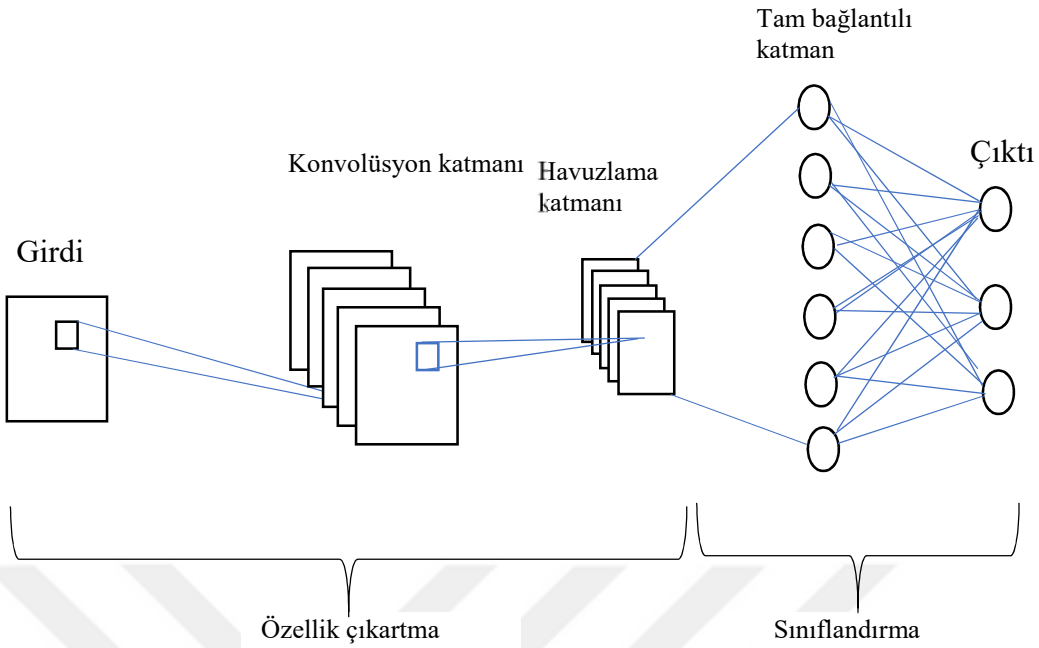
Konvolüsyonel sinir ağı algoritmaları, yapay zeka, makine öğrenimi ve derin öğrenmeyi içeren hiyerarşik terminolojinin bir alt sınıfıdır [17]. Yapay zeka, genellikle insan zekası gerektiren sorunları çözen algoritmaları tanımlar. Makine öğrenimi, açıkça programlanmadan, öğrenme yeteneğine sahip algoritmalar oluşturmaya adanmış bir yapay zeka alt sınıfıdır. Derin öğrenme, hiyerarşik terminolojide bir sonraki alt sınıftır. Derin öğrenme ile klasik makine öğrenimi arasındaki temel fark, ikincisinde insan uzmanların görsel verileri en iyi şekilde temsil eden görüntüleme özelliklerini seçmesi, derin öğrenmede ise hiçbir özellik seçiminin kullanılmamasıdır. Bunun yerine, derin öğrenme algoritmaları hesaplama görevi için hangi özelliklerin en iyi olduğunu kendi başlarına öğrenir. Derin öğrenme algoritmalarının çoğu YSA'ya dayalıdır. Bir KSA, girdilerin görüntü olduğu açık varsayımını yapan yapay sinir ağının bir alt kategorisidir. Son birkaç yılda, KSA teknolojisi, bilgisayarla görme alanındaki en etkili yeniliklerden bazılarının temeli olmuştur [17,2].

KSA'ların girdileri görüntülerdir ve bu yönden YSA'ya çok benzer. Bu benzerlik, YSA'daki belirli özellikleri KSA mimarisine kodlamamıza izin verir. Tipik KSA mimarisi, bir görüntünün hiyerarşik özelliklerinin öğrenilmesini sağlayan birkaç katmandan oluşur. Bu katmanlarda görüntü girdisi piksellere ayrılarak incelenir ve makine tarafından öğrenilmesi sağlanmış olur.

2.3.1. KSA yapısı

KSA yapısı veya mimarisi, görüntü hacmini çıktı sınıfı puanlarına dönüştüren bir dizi katman içerir. Her katman, türevlenebilir bir işlev aracılığıyla bir aktivasyon hacmini diğerine dönüştürür. Bir KSA oluşturmak için birleştirilen ana katman türleri, aşağıda daha ayrıntılı olarak ele alınan konvolüsyonel katman, havuzlama katmanı, normalleştirme katmanı, tam bağlantılı katman, kayıp katmanıdır (Şekil 3). YSA'nın gücü, çoklu nöronların çoklu derin gizli katmanlardaki entegrasyonunda yatar. Bir katmanın çıktıları, bir sonraki katmanın girdileri olarak işlev görür. Son nöron katmanı, ileri yayılım adı verilen bir süreç olan, belirtilen verilerin etiketlerini tahmin etmede ağın mevcut doğruluğunu tahmin eden bir kayıp fonksiyonundan oluşur. Kayba bağlı olarak, geri yayılım adı verilen bir süreçte ağın ağırlıklarında küçük değişiklikler yapılır. Veri kümesinin tamamında tekrarlanan ileri ve geri yayılım yinelemeleri sonunda optimize edilmiş bir ağ oluşturur [18].

Bir KSA genellikle 3 boyutlu ve 3 kanallı (R,G,B renk kanalları) görüntüleri işler. Daha yüksek boyutlu görüntüleri de işleyebilir. Girdi pikseli bir dizi işlemden geçer. Pikselin işlendiği adımlara katman adı verilir. Bu katmanlar; konvolüsyonel katman, havuzlama katmanı, normalleştirme katmanı, tam bağlantılı katman gibi katmanlar olabilir.



Şekil 2.3. *Konvolüsyonel Sinir Ağları mimarisi [8], (http-7)*

Konvolüsyonel katman giriş görüntüsünü olduğu gibi ele alabilir (yakın piksellerin değerleri ve uzak pikseller farklı şekillerde işlenir). Evrişim işlevi filtrelerin yerini adımlar olarak tekrar tekrar değiştirerek tanır [16]. Konvolüsyon işleminden sonra aktivasyon fonksiyonu devreye girer. Bir sinir ağındaki nöronun başlıca görevi bilgi taşımak ve iletmektir. YSA'nın en küçük parçası olan nöronlar, n sayıda girdi alır ve tek bir çıktı üretirler. Bir yapay nöron x girdileri ile w ağırlıklarını çarpıp bias değeri eklenerek elde edilir. Derin Öğrenme uygulamalarında amacımız oluşturduğumuz model için en iyi doğruluk ve hata skorlarını verecek "w" ve "b" parametre değerlerini hesaplamaktır. Aktivasyon fonksiyonu (6)'da görüldüğü üzere y değerini kontrol etmek için bir diğer ifadeyle söz konusu nöronun aktiflik durumunun kararını belirlemek için kullanılmaktadır [13].

$$y = \text{Aktivasyon} \left(\sum (w * x + b) \right) \tag{2.6}$$

Havuzlama katmanının amacı, girdi görüntüsü çok büyük olduğundan, görüntüyü küçültmek için parametre sayısını azaltmaktır (http-6). İki tür havuzlama yöntemi vardır. Bunlar maksimum havuzlama ve ortalama havuzlama yöntemleridir. Bu katmanda genellikle maksimum havuzlama tercih edilir (http-3). Konvolüsyonel katmanlardan sonra giriş verilerinin uzamsal boyutunu azaltmak için maksimum havuzlama yöntemi

kullanılır. İki kıvrımlı katman arasında yer alır. Maksimum havuzlama yöntemi, dönüş ve konumlarını değiştirmeden yüksekliklerini ve genişliklerini azaltarak temel özellikleri kaldırarak bir modeli etkili bir şekilde eğitme sürecini sürdürür (http-3).

Normalleştirme katmanında veriler normalleştirilir (Verilerin dağılımı değiştirilir, böylece ortalama ve varyans sırasıyla 0 ve 1 olur) minibatch birimlerinde (bir grup daha sonra açıklanacak olan veriler). Bu teknik bilinen aşırı yükleme sorunu riskini azaltmak için kullanışlıdır [13]. Bu aynı zamanda öğrenme sürecini hızlandırır.

Tam bağlantılı katman, adından da belli olduğu gibi nöronların çıkış birimine bağlanmasını sağlayan katmandır. Çok fazla nöron ağı olabileceği için diğer ağlara göre daha karmaşık bir hesaplama sistemi vardır bu yüzden tıkanmaya da elverişlidir.

2.3.2. Konvolüsyonel sinir ağları eğitimi

İnsanlar gibi sinir ağları da sürekli örneklerle öğrenir. Bir sinir ağı bir dizi girdi verisi ile eğitilmeye başlanır. Ağları eğitmenin amacı ise sözde hatayı en aza indirmektir. Hata dediğimiz durum ise sinir ağından istenen çıktı ile sinir ağının çıktısı arasındaki farktır. Yaygın bir hata işlevi, sinir ağları çıktısı ile istenen çıktı arasındaki kare farklarının toplamıdır[19].

Sinir ağının amaçlandığı gibi çalıştığını doğrulamak için bir doğrulama seti kullanılabilir. Buna validation- test (doğrulama) set verisi denir. Doğrulama seti genellikle eğitim yani training setinin boyutunun %10-40'ı kadardır (http-10) . Eğitim seti gibi, doğrulama seti de girdi verilerini içerir. Aradaki fark, doğrulama setini kullanırken sadece doğruluğu gözlemlemek için olduğu için nöronlar arasındaki bağlantılarla ilişkili ağırlıkların değişmemesidir. Kısaca veriler belli oranlarda bölünerek test ve train setleri olarak ayrılırlar (http-10). Ve bu şekilde doğrulanarak eğitilirler.

KSA eğitim verileri ile geri yayılım algoritması kullanılır. Bir konvolüsyonel katmanının özellik haritalarında, tüm sinirler aynı ağırlıkları ve eşik değerlerini paylaşırlar. Bu yüzden bu ağlarda aynı boyutlu YSA'lara göre daha az parametre bulundururlar. KSA eğitildikten sonra ileri beslemeli ağ olarak çalışır. Bu sayede sınıflandırma tamamlanmış olur.

3. YAPAY SİNİR AĞLARINDA NESNEYE YÖNELİK SINIFLANDIRMA

YSA ile nesneye yönelik sınıflandırma veya diğer adıyla nesne tanıma yapabilmek için, nesnelere ait özel kameralar veya fotoğraf makinalarıyla elde edilmiş görüntüler bulunmalıdır. Lazerler görüntüleri segmentlere ayırır ve bu şekilde tanınmalarını sağlar [20].

Nesne algılama teknolojisinin amacı işlenecek görüntüyü tespit etmektir. Nesne tabanlı sınıflandırma yöntemi direkt olarak tekil pikseller üzerinde çalışmaz. Bu yöntem, görüntüyü anlamlı bir şekilde segmentasyon işlemi ile gruplandırılmış, birçok pikselden oluşan objeler üzerinden çalışır. Daha sonra sınıflandırma ögesi olarak pikseller yerine bu objeleri kullanır. Homojen olarak oluşturulan bu objeler; "objeleri" spektral, şekil, boyut, doku gibi özellikleri ile birbirleri arasındaki ilişkileri kullanarak sınıflandırma işlemini gerçekleştirir [11].

3.1. Öğrenme Metotlarına Göre Sınıflandırma

3.1.1. Piksel tabanlı sınıflandırma

Piksel tabanlı sınıflandırma tekniği adından da anlaşılacağı gibi görüntüdeki her pikselin değerini kullanarak her birini bir sınıfa bağlar. Pikselleri özelliklerine (spektral özelliklerine) göre sınıflandırma işlemine spektral örüntü denir [9].

Bu teknik kullanım kolaylığı açısından diğerlerine göre daha fazla tercih edilir. Piksel tabanlı sınıflandırma yönteminde ek olarak birkaç yöntem kullanılır. En fazla tercih edilenler ise; en çok benzerlik, en kısa mesafe ve paralelyüz yöntemleridir. Bu yöntemler tamamen istatistiksel yöntemlerdir ve sınıf belirleyicileri olarak görev alırlar. Sınıfları karakterize ettikleri için birbirine yakın özellikteki sınıflar hataya sebep olabilir. Bunu önlemek için daha kuvvetli yöntemler tercih edilebilir.

3.1.1.1. Kontrollü sınıflandırma

Kontrollü öğrenme yöntemi, belirli girdi-çıktı eğitim örneklerine dayalı olarak, bir sistemin girdi-çıktı ilişkilerini öğrenmek için kullanılan bir yöntemdir. Bu sınıflandırma yöntemi optimizasyon probleminin çözümünde kullanılan bir yöntemdir. Kullanıldığı

alanlar; belge indeksleme, web sayfası sınıflandırması ve spam filtreleme; biyoloji, kimya ve tıp gibi alanlardır.

Kontrollü sınıflandırmada önce test alanı belirlenir. Daha sonra özelliklerine göre ayırt edilir ve son olarak görüntüler bu özelliklere göre sınıflandırılır. Sınıflandırmanın çalışabilmesi için beraberinde kullanılabilecek çeşitli algoritmalarında olması gerekir. Bu algoritmaların en yaygın ve kullanışlı olanları; en büyük benzerlik (max likelihood), paralelyüz (paralel lepiped) ya da en kısa mesafe (min distance)'dir. Bu algoritmaların ortak özelliği hepsinin mesafeye dayalı olmasıdır.

En çok benzerlik yöntemi, en çok tercih edilen yöntemdir. Amacı piksellerin en iyi sınıfa atanmasını sağlamaktır. Sınıflardaki verilerin normal dağıldığı varsayılır veya normalleştirilir [10]. Piksellerin hangi sınıfa daha uygun olabileceğini hesaplar ve sınıflandırır. Eğer belirlenen eşiğin altında kalan pikseller olursa sınıflandırmadan kalırlar.

Her bir pikseli sınıflandırmak için aşağıdaki diskriminant denklemi uygulanır.

$$g_i(x) = 1np(w_i) - {}^{1}/_{2}\, 1n|\sum i| - {}^{1}/_{2}\, (x - m_i)^T \sum \dot{1}^{-1} (x - m_i) \qquad (3.1)$$

i= sınıf

x = n boyutlu veri (burada n, bant sayısıdır)

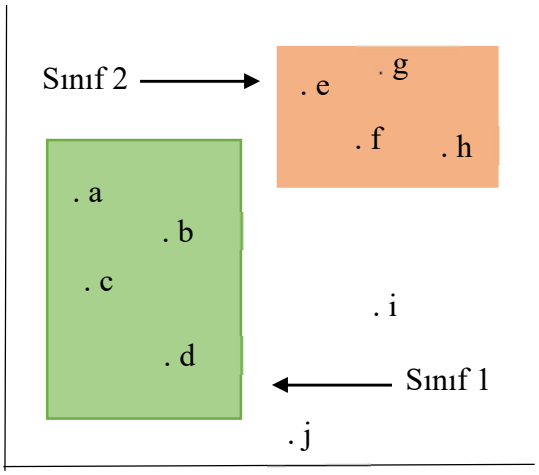
p(ωi) = görüntüde ωi sınıfının ortaya çıkma olasılığı ve tüm sınıflar için aynı olduğu varsayılır.

|Σi| = ωi sınıfındaki verilerin kovaryans matrisinin belirleyicisi

Σi-1 = ters matrisi

mi = ortalama vektör

Paralelyüz yaklaşımı, çok bantlı görüntüler için yaygın olarak kullanılan denetimli sınıflandırma algoritmalarından biridir [10]. Her spektral (sınıf) özellik eşiği, verilen bir pikselin sınıf içinde olup olmadığını belirlemek için eğitim verilerinde tanımlanır. Histogramların incelenmesine bağlı bir yöntemdir [8].



Şekil 3.1. *Paralel yüz yaklaşım grafiği (http-4)*

Yukarıdaki grafikte gördüğümüz gibi kutuların içindeki harfler sınıflandırılmış, fakat dışarıda kalanlar sınıflandırılmamış olarak kalmıştır. Paralelyüz yöntemi çok fazla bilgi gerektirmez. Belirtilen sınıfların her biri için kullanıcı, bantların her birinde minimum ve maksimum piksel değerinin bir tahminini sağlar. Ya da, her özelliğin ortalamasının her iki tarafında belirli sayıda standart sapma birimi cinsinden ifade edilen bir aralık kullanılabilir. Bu değerler paralel yüzün sınırlarını belirler (http-4). Ortalama ($\bar{X}$) ve Standart sapmanın formülü (S) aşağıdaki gibidir:

$$\bar{X} = \frac{Sr}{n} \tag{3.2}$$

$$S = \sqrt{\frac{\Sigma(r-x)^2}{n}} \tag{3.3}$$

x = ortalama

s = standart sapma

r = piksel değeri

n = banttaki toplam piksel sayısı

Bu yöntem oldukça pratiktir. Ama bir sınıfın özelliğinden daha az ya da daha çok özelliğe göre atama yapabileceğinden çokta güvenilir değildir.

En kısa mesafe sınıflandırıcısı, bilinmeyen görüntü verilerini, çok özellikli uzayda görüntü verileri ile sınıf arasındaki mesafeyi en aza indiren sınıflara ayırmak için kullanılır [10]. Mesafe, bir benzerlik indeksi olarak tanımlanır, böylece minimum mesafe yaklaşımı mantık olarak maksimum benzerlikle aynıdır. Bu prosedürde genellikle aşağıdaki mesafeler kullanılır.

Piksel ile küme merkezi arasındaki Öklid mesafesi eşitliği ile hesaplanır [10,19].

$$D_E^2 = (x_i - \mu_j)^2 \tag{3.4}$$

Bu eşitlikte, D_E , Öklid mesafesini; x_i , *i*'inci pikselin gözlem vektörünü; μ_j , *j*'inci kümenin ortalama vektörünü göstermektedir. x_i vektörünün boyutu, girdi olarak kullanılan görüntünün bant sayısına eşittir.

Mahalanobis mesafesi [23],

$$D^2 = (x_k - v_i)^T M_i^{-1} (x_k - v_i) \tag{3.5}$$

eşitliği ile hesaplanır. (3.5) eşitliğinde, D , *i* sınıfının merkezi ile *k* pikseli arasındaki Mahalanobis mesafesini; M_i , *i* sınıfı için varyans-kovaryans matrisini; v_i , *i* sınıfı için ortalama vektörü göstermektedir. Mesafe küçüldüğünde *i* ve *j* objeleri arasındaki benzerlik artar. Her bir sınıf için sınıf merkezi ya da spektral vektör, eğitim veri setlerinden belirlenir. Tanımsız bir piksel, kendi piksel değeri ile her bir sınıf merkezi arasındaki mesafe hesaplanarak etiketlenir.

Her bir sınıfın şekli kullanılan mesafe fonksiyonuna bağlıdır. Bu sınıflandırıcı, matematiksel olarak basittir. Mahalanobis mesafesinin Öklid mesafesinden daha iyi sonuç verdiği bilinmektedir. En Küçük Mesafe algoritması, En Çok Benzerlik sınıflandırma tekniğinden daha hızlı olduğu için daha cazip hale gelmiştir. Ancak, En Küçük Mesafe algoritmasının doğruluğu, En Çok Benzerlik algoritmasının önüne geçememiştir [15].

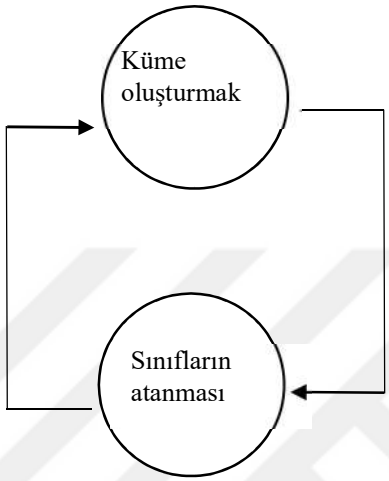
3.1.1.2. Kontrolsüz sınıflandırma

Kontrolsüz sınıflandırma; görüntüdeki veri tanımlanamadığında başvurulan bir yöntemdir. Bilgisayar tarafından otomatikleştirilmiş bir sınıflandırmadır diyebiliriz (http-2). Kullanıcı, sınıfların sayısını belirler ve spektral sınıflar, yalnızca verilerdeki sayısal bilgilere (yani, her bir bant veya indeks için piksel değerlerine) dayalı olarak oluşturulur. Yani, veri bandı değerleri yardımı ile benzer piksellerin otomatik olarak bulunması temel alınmaktadır [2].

Kontrolsüz sınıflandırma yapabilmek için örneklere ihtiyacımız yoktur ve bu yüzden bir görüntüyü segmentlere ayırmanın ve anlamanın kolay bir yoludur diyebiliriz.

Kontrolsüz sınıflandırma için iki temel adım:

1. Küme oluşturmak,
2. Sınıfların atanması.



Şekil 3.2. *Kontrolsüz sınıflandırma adımları*

Kümeleme algoritmaları, verilerin istatistiksel olarak gruplandırılmasını sağlar. Bu algoritma seçildikten sonra kaç adet grup oluşturulacaksa belirlenir. Bu sayede pikseller benzer özelliklerine göre sınıflandırılır. Çok fazla sayıda da küme oluşturulabilir. Daha az küme sayısı daha çok benzerlik demektir. Daha fazla küme olması gruplar içinde değişkenliklerin fazla olmasına sebep olabilir. Bu tarzdaki kümelere sınıflandırılmamış kümeler denir.

Sınıflandırma işlemi tamamlandıktan sonra, sınıflar, sınıf sayısı veya değişiklik eşiği gibi değerlere göre yorumlanmalıdır. Bu yorumlara göre etiketlenmeli ya da renk kodlanmalıdır.

4. NESNE TABANLI SINIFLANDIRMA

4.1. Uzaktan Algılama (Hipersektral) Sınıflandırma

Hiperspektral algılayıcıların gelişimiyle beraber hipersektral görüntüleme çok fazla dalga boyutunda ölçüm yapılmasını sağlayabilir. Ve bu dalga boyutlarını birbirleriyle ilişkilendirebilir. Oluşturulan verilerin birbirleriyle ilişkilendirilmesi oldukça zordur. Bu yüzden boyut indirgenmesi yapılmalıdır. Fakat geleneksel sınıflandırma yöntemleri bu konuda yetersiz kalabilir. Ayrıca bununla ilgili kesin bir yöntemde bulunmamaktadır. Son yıllarda kullanılan derin öğrenme yöntemleri verileri sınıflandırmada daha kullanılır bir yaklaşım olmuştur. Özellikle konvolüsyonel sinir ağları yöntemleri hipersektral yöntemlerle kullanılabilecek en iyi yöntemler olduğu söylenebilir.

4.2. K-ortalamalar yöntemi

K-ortalamalar yöntemi, sınıfı tahmin edebilen başka bir parametrik olmayan yöntemdir. k-ortalamalar verilen bir veri seti üzerinden belirli sayıda kümeyi (k adet) gruplamak için geliştirilmiş en sade ve basit algoritmadır. Bir nesnenin k en yakın komşusunun sınıfına göre k-ortalamalar yöntemi üç aşamada gerçekleştirilir;

1. Önce bir y_i gözleminden diğer tüm y_j gözlemlerine olan mesafe (N_0) kullanılarak hesaplanır (Buna ek olarak uzaklık fonksiyonu kullanılır. Örneğin Öklid fonksiyonu gibi).
2. Daha sonra yanıt değerleri eşit olan N_0 'da ki noktaların kesri j için aşağıdaki koşullu olasılık formülü kullanılarak tahmin edilir [23] :

$$\Pr(Y = j|X = x_0) = \frac{1}{k}\sum_{i\in N_0} I(y_i = j) \tag{4.1}$$

Öklid uzaklığı (http-5),

$$\mathrm{d(p,q)} = \mathrm{d(q,p)} = \sqrt{(\mathrm{q_1} - \mathrm{p_1})^2 + (\mathrm{q_2} - \mathrm{p_2})^2 + \cdots + (\mathrm{q_n} - \mathrm{p_n})^2}$$
$$= \sqrt{\sum_{\mathrm{i=1}}^{\mathrm{n}} (\mathrm{q_i} - \mathrm{p_i})^2} \tag{4.2}$$

3. Daha sonra bu komşular kullanılarak Bayes kuralına göre sınıflar belirlenir. Sınıflar belirlenmediği takdirde bu algoritma devamlı olarak uygulanır.

5. GÖRÜNTÜ SINIFLANDIRMADA KULLANILAN İSTATİSTİKSEL YÖNTEMLER

5.1. Regresyon Analizi Modelleri

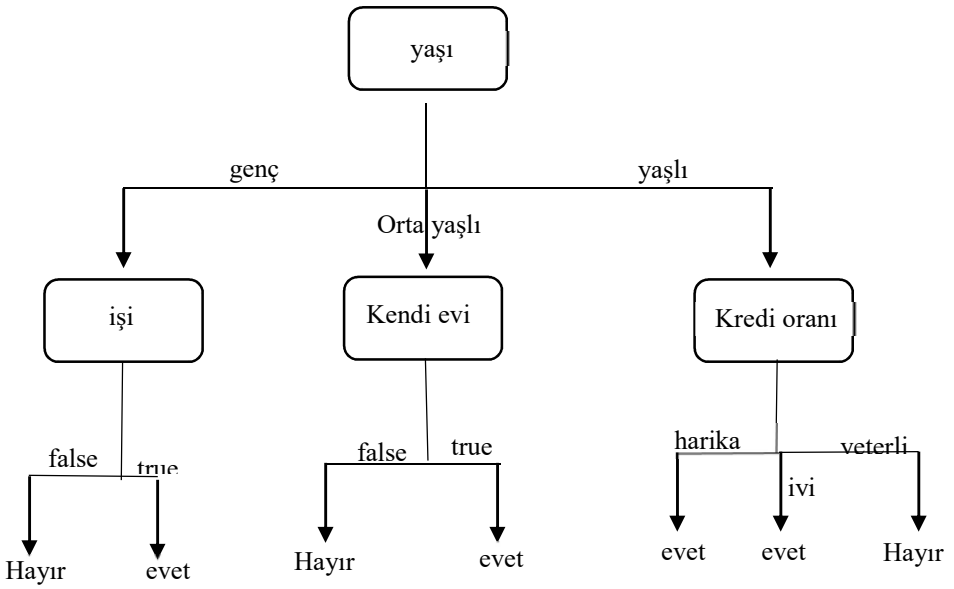
Regresyon analizi değişkenler arası ilişkilerin araştırılmasında kullanılır. Yanıt değişkeni ve açıklayıcı değişkenler arasındaki ilişkiyi tahmin etmek, tanımlamak veya sınıflandırmak için kullanılır. Bağlantıyı aşağıdaki formülle ifade edebiliriz [24,35]:

$$Y = f(X_1, X_2, \dots, X_n) + \varepsilon \quad (5.1.)$$

Burada Y yanıt değişkenini, f fonksiyonu açıklayıcı değişkenleri ve ε ise rastgele bir hata vektörünü temsil eder.

5.1.1. Sınıflandırma ve regresyon ağaçları

Sınıflandırma ve Regresyon Ağaçları (CART) yöntemi; çoklu bağlantı ve kayıp verinin etkilerine karşı dirençli bir yöntemdir. Ayrıca modelde hangi değişkenin içerilmesi gerektiğinden önce, tahminciler arasındaki ilişkileri tanımlama yeteneğinden dolayı parametrik yaklaşımlar yerine tercih edilebilir. Sınıflandırma ve regresyon ağaçlarında, maksimum Regresyon 3 kısımdan oluşan ağaç ile oluşturulur. Uygun ağaç genişliği seçilir, oluşturulan ağaçtan hareketle yeni veriler sınıflandırılır.



Şekil 5.1. *Sınıflandırma ve regresyon ağacı örnek çıktısı (http-8)*

CART yönteminde en iyi ağaçlandırmayı oluşturabilmek için birkaç yöntem kullanabilir, bunlardan en yaygın olanı gini safsızlık algoritmasıdır .

$$Gini\ safsızlık = 1 - Gini = 1 - \sum_{i=1}^{n} p_i^2 \quad (5.2)$$

Burada P_i düğümün seçilme olasılığını belirtir.

İki alt düğüm için en iyi ağırlıklı Gini safsızlık sonucu, ana düğüm için Gini safsızlıktan daha düşük değilse**,** ana düğümü daha fazla ağaçlandıramayız.

Veya CART yönteminde kullanılabilecek bir kavramda entropidir. Entropi şu şekilde kullanılabilir :

$$Entropi = -\sum_{i=1}^{n} p_i log_2(p_i) \quad (5.3)$$

Entropi eğer düşük ise iyi bir sınıflandırma yapılmış diyebiliriz. Fakat iki alt düğümün entropisi bir ana düğümün entropisinden daha düşük değilse, daha fazla ağaçlandırma yapılamaz.

5.1.2. En küçük açı regresyonu

LARS olarak belirtilen en küçük açı regresyonunun (LAR) sonuna eklenen ‘S’ ise "Lasso" ve "Stagewise" anlamına gelmektedir [22]. Lasso en küçük kareler yönteminin sınırlandırılmış bir fonksiyonudur. Bir anlamda stagewise in biçimlendirilmiş bir sürümü de diyebiliriz. Hesaplamaları hızlandırmak için basit bir matematiksel formül kullanır. Bu şekilde hesap yükünü azaltır.

Bu yöntemi uygulayabilmek için önce bütün veriler normalleştirilir. En yüksek düzeyde ilişkili olan veri saptanır, aynı veya daha yüksek korelasyona sahip başka bir değişkene ulaşana kadar o yönde regresyon çizgisi hareket ettirilir.

LARS tahminlerini şu formülle oluşturur: $\hat{\mu} = X\hat{\beta}$. Birbirini izleyen adımlar halinde, her adımda modele değişken ekleyerek işleme devam edilir. Aşağıda algoritma adımları verilmiştir [5,26]:

1. Tahminciler ortalamaları 0 veya birim normda olacak şekilde standartlaştırılır. Ve artık fonksiyonla başlanır: $r = y - \bar{y}, \beta_1, \beta_2, \ldots, \beta_p = 0$
2. r ile en çok ilişkisi bulunan x_j tahmincisi bulunur.

3. Başka bir rakip xj, mevcut artık ile aynı miktarda korelasyon katsayısına sahip olana kadar; β, 0'dan en küçük kareler katsayısına doğru (x_j, r) hareket ettirilir.
4. Başka bir rakip xj, mevcut artık ile aynı miktarda korelasyon katsayısına sahip olana kadar; β_j ve β_k, (xj, xk) üzerindeki mevcut kalıntının ortak en küçük kareler katsayıları tarafından tanımlanan yönde hareket ettirilir.
5. Tüm p tahmincileri girilene kadar bu şekilde devam edilir. min(N-1,p) adımdan sonra LARS çözümüne ulaşırız.

5.1.3. Lojistik regresyon

Lojistik Regresyon, sınıflandırmada kullanılan ve olasılık fonksiyonlarıyla beraber makine öğrenmesinde de kullanılan bir tekniktir. Bu teknikte, tek bir denemenin olası sonuçlarını tanımlayan olasılıklar bir lojistik fonksiyon kullanılarak modellenmiştir ve bu amaç için tasarlanmıştır. Lojistik regresyon algoritması bağımsız değişkenlerin tek bir sonuca etkisini anlamak için en kullanışlı regresyon algoritmasıdır. Lojistik regresyon ikili yanıt değişkenini modelleyebilir. Bu regresyon modeli denklem (5.4)'de tanımlandığı gibi çoklu regresyon formülünü kullanır:

$$\text{Logit(p)}=\log\left(\frac{p(Y=1|X)}{1-p(Y=1|X)}\right)=\beta 0+\beta 1X1+\beta 2X2+\beta 3X3+\cdots+\beta kXk \qquad (5.4)$$

Buradaki Y değeri (0,1) yani ikili değişkendir. Referans seviyesinden yüksekse 1, değilse 0'dır. X açıklayıcı değişkendir. $\beta_0, \beta_1, \beta_2 \ldots$ değerleri tahmin edilen regresyon katsayılarıdır. p değeri, karakteristik özelliğin var olması olasılığıdır. Bir diğer önemli değer ise odds değeri olarak adlandırılan değerdir. Odds değeri p' nin yani karakteristik özelliğin var olma olasılığının, var olmama olasılığına yani 1-p değerine bölünmesiyle şu şekilde elde edilir, Odds=p/p-1=karakteristiğin var olma olasılığı/karakteristiğin var olmama olasılığı.

Odds değerinin anlamı iki farklı durumun gerçekleşmesi olayını tanımlamasıdır. Bu değerlerin oranının anlamı ise gerçekleşme olasılıklarını göstermesidir. Örneğin bir grupta sigara içenlerin yaşlarına göre ölüm risklerinin araştırılmasında, hangi yaşın daha çok ölüm olasılığının olduğunun hesaplanması sağlanabilmektedir.

Görüntü içeriklerinin açıklanması için çoğunlukla lojistik regresyon algoritması önerilebilir. Doğruluk oranları diğer algoritmalara göre daha anlamlıdır.

5.1.4. Çoklu lojistik regresyon

Çoklu lojistik regresyon modelini kullanmamızın amacı çok sayıda parametreyi kullanabilmemizdir. Çoklu lojistik regresyon modeli sonuçlandırıldığında karmaşıklıkları çok daha kolay basitleştirebilmektedir.

P adet bağımsız değişken var ise çoklu lojistik regresyon modeli aşağıdaki gibi oluşturulur:

$$g(x) = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k \tag{5.5}$$

Bu eşitliğin logit modeli,

$$\pi(x) = \frac{e^{g(x)}}{1+e^{g(x)}} \tag{5.6}$$

şeklinde ifade edilir.

5.2. Karar Ağaçları

Karar ağaçları yukarıdan aşağıya tarama yaparak ilerleyen bir sınıflandırma çeşididir. Tek özelliği ya da 2 özelliği olan verileri homojen hale getirmek için entropi kullanılır. Ağaç oluşturulurken iki çeşit entropi hesaplanmalıdır. Bunlar tek özellik içeren veriler için kullanılan ve iki özellik içeren veriler için kullanılan entropidir.

- Tek özellik içeren veriler için entropi :

$$E(S) = \sum_{i=1}^{c} -p_i log2 p_i \tag{5.7}$$

E(S)= S özelliğinin entropisi

p_i = oluşturulacak yaprakların olasılıkları

- İki özellik içeren veriler için entropi:

$$E(T,x) = \sum_{c\epsilon x}^{c} P(c)E(c) \tag{5.8}$$

E(T,x)= T ve x özelliklerinin entropisi

P(c)= oluşturulacak yaprakların olasılıkları

E(c)= oluşturulacak yaprakların entropileri

Entropiyle beraber ID3 algoritmasından yararlanılır. Bu algoritma karar ağacı oluşturmak için yukarıdan aşağıya doğru ilerler. Düğüm oluşturmak için, hesaplanan en iyi özellik kullanılır. Daha önce belirtildiği gibi, ID3 algoritması bir Karar ağacı oluştururken her adımda en iyi özelliği seçer. ID3 en iyi özelliği bulmak için, Information

Gain'i yani bilgi kazanımını kullanır. Information Gain için önce entropi hesaplanmalıdır [14]. Eğer entropi 0 ise veriler homojen kümelenmiş, 1 ise veriler homojen kümelenmemiştir. Homojen hale gelen veriler ile karar ağacı düğümleri oluşturulur ve sınıflandırma aşamasına geçilir [2].

Adımları:

1) Hedefin entropisi hesaplanır .
2) Özellik sütununa A dersek entropininde dahil olduğu ID3 algoritması şu şekilde hesaplanır [14]:

$$IG(S,A) = Entropy(S) - \sum(\frac{(|S_v|)}{(|S|)} * Entropy(S_v)) \quad (5.9)$$

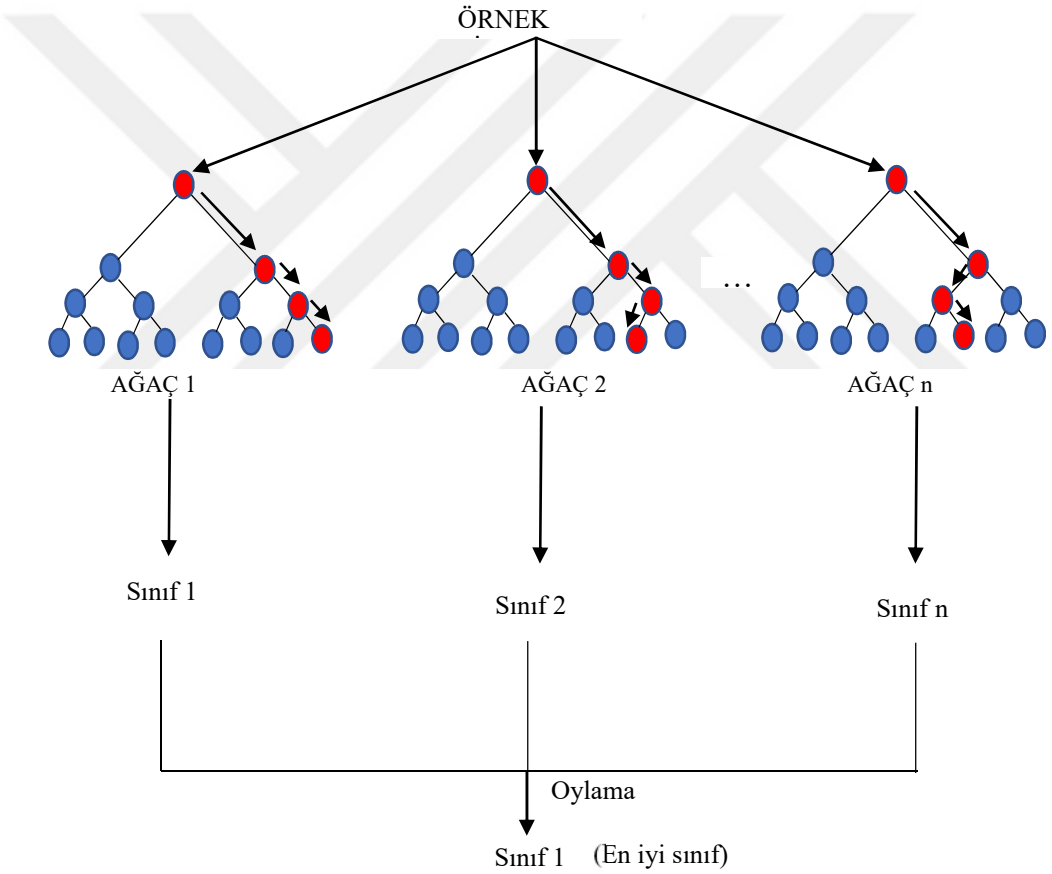
   Burada S_v S'de ki sütunların kümesini, $|S_v|$ bu sütunların sayısını belirtir. Aynı sekilde |S| ise satırların sayısını belirtir.
3) Bu algoritma yardımıyla her satırın bir düğümü olana kadar yinelemeli olarak bölme işlemi devam eder.
4) Veriler farklı özelliklerde olacak şekilde bölünür ve oluşturulacak her yeni dal için entropi hesaplanır.
5) Bir karar düğümü seçilmesi gerekir ve bu da en çok bilgiyi saklayan düğüm olacaktır. Veri seti dallara bölündükten sonra aynı işlem tekrarlanır.
6) Entropinin hesaplanması verilerin homojen olarak dağılmasını belirleyen bir formüldür. ve eğer entropi 0 çıkarsa, veriler gerektiği gibi homojen olarak dağılmıştır fakat 1 ise daha fazla dallanma gerektirebilir.
7) Tüm veriler sınıflandırılana kadar yapraksız dallarda bu işlem ID3 algoritmasıyla tekrarlanır [2].

5.3. Rassal Orman

Rassal orman yöntemi 1'den fazla karar algoritmasını kullanan ve bu sayede diğer doğrusal olmayan algoritmalara göre daha başarılı bir sonuç elde edebilen bir kolektif öğrenme yöntemidir. Diğer algoritmalarla kıyaslandığında oldukça fazla avantajı vardır. 2 çeşit veri tipinin (kategorik,sürekli) de kullanılabilmesi veya verilerde uydurma sonucu oluşan hataların en aza indirilebilmesi en önemli avantajlarından biridir. Ayrıca hata oranlarını hesaplayabilir bu da bir diğer önemli avantajıdır.

Aynı zamanda karar ağaçlarının topluluğunun bir üyesidir ve başarısı açısından algoritmalar arasında sıkça kullanılan bir yöntemdir. Karar ağaçları yöntemiyle aynı şekilde, birbirine en çok benzeyen verilerin Gini indeksi gibi bir özellik ayırt etme katsayısıyla beraber sınıflandırılmasına dayanır.

Rassal orman ağaçların oluşturulabilmesi için CART algoritmasından yararlanır. İlk olarak {h(x,θk),k=1,…} bir karar ağaçları kümesi olarak tanımlanır, h(x,θk) burada bir meta sınıflandırıcıdır yani CART kullanılarak oluşturulan henüz işleme girmemiş bir ağaçtır. Girdi verilerine göre her karar ağacı bir sonuç verir [3]. Bu sonuçların birleşimi ile en iyi sınıfın bulunduğu sınıf olası sonuç olarak seçilir.



Şekil 5.2.*Rassal orman çalışma yapısı [4]*

5.4. Naive Bayes

Naive bayes sınıflandırıcıları bayes teoremini uygulamaya dayanan olasılıklı sınıflandırıcılardır. Bu model en basit bayes modelleri arasındadır fakat yoğunluk

tahminleriyle birleştiğinde çok yüksek doğruluk sonuçları alabilirler. Sınıflandırılan her özellik çifti birbirinden bağımsızdır [21].

Kullanılan veri seti özellik matrisi ve yanıt vektörü olarak ikiye ayrılır. Özellik matrisi, her vektörün bağımlı özelliklerin değerinden oluştuğu veri kümesinin vektörlerini (satırlarını) içerir [21]. Yanıt vektörü özellik matrisinin her satırı için sınıf değişkeninin (tahmin veya çıktı) değerini içerir.

Tablo 5.1. *Naive Bayes veri seti*

Hava durumu	Nem	Rüzgâr	Sıcaklık	Pikniğe gitmek
Güneşli	Hafif	Yok	Sıcak	Evet
Güneşli	Fazla	Yok	Sıcak	Evet
Yağmurlu	Fazla	Var	Soğuk	Hayır
Rüzgarlı	Hafif	Var	Soğuk	Hayır
Güneşli	Fazla	Var	Sıcak	Hayır
Rüzgarlı	Hafif	Var	Sıcak	Evet
Yağmurlu	Hafif	Var	Soğuk	Hayır

Yukarıdaki tabloda özellik matrisi hava durumu iken yanıt vektörü de pikniğe gitmek veya gitmemek şeklindeki tepkilerdir. Yani hava durumu göze alındığında her satır koşulları piknik yapmak için uygun veya değil şeklinde sınıflandırır.

Naive Bayes sınıflandırıcıda özniteliklerin birbirinden bağımsız olduğu kabul edilmektedir. Yani her değişken için ayrı olasılık hesaplanır. Verilerin hangi sınıfa ait olduğu maksimum koşullu olasılıkla belirlenir. Koşullu olasılık formülü ise aşağıdaki gibidir:

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)} \tag{5.10}$$

Burada P(A/B) = B olayı gerçekleştiğinde A nın gerçekleşme olasılığı,

P(A) = A olayının gerçekleşme olasılığı,

P(B/A) = A olayı gerçekleştiğinde B olayının gerçekleşme olasılığı,

P(B) = B olayının gerçekleşme olasılığıdır.

5.5. Destek Vektör Makineleri

Destek Vektör Makineleri (DVM) parametrik olmayan, denetimli ve çekirdek tabanlı bir istatistiksel öğrenme yöntemidir. Çekirdek tabanlı öğrenmede, girdi verileri tanımlanan yüksek boyutlu bir örnek uzaya bir haritalama ile çekirdek tarafından aktarılır. Başka bir deyişle çekirdek tabanlı öğrenme, doğrusal olmayan problemler için bir doğrusal hiperdüzlem kullanır. Veri boyutuna fazladan bir boyut daha eklenir ve ardından DVM veri uzayında bir düzlem oluşturur. Böylece veriler doğrusal olarak ayrılmış olur, doğrusal olmayan uzayda dönüşüm sağlanır.

Test örneklerini iki sınıfa ayırmak amacıyla en uygun hiperdüzlemin bulunması amaçlanmaktadır. Eğitim aşamasında sınıfları birbirinden ayıran hiperdüzlemin eğitim örneklerine uzaklığı mümkün olan maksimum düzeyde tutulmaya çalışılır.

DVM optimum çözüm ve her özelliğin kısmi farklılaşmasını hesaplamak için Lagrange çarpanını kullanır. Sonuç olarak model, eğitim verilerinin karmaşıklığını bir alt kümeye kadar azaltır. N veri noktasından oluşan ve $\{x_k, y_k\}_{k=1}^{N}$ giriş verilerine sahip bir eğitim veri seti düşünelim, n boyutlu bir veri vektörü olan $(x_k \in R^n)$ ve çıkış tek boyutlu vektör uzayı $(y_k \in r)$ DVM'yi oluşturur [23]. Bu bağlantı denklem (5.11) de gösterilmiştir.

$$y_x = \text{sign}[\sum_{k=1}^{N} \alpha_k y_k \psi(x, x_k) + b] \tag{5.11}$$

Burada α_k pozitif gerçek sabitlerdir, b ise gerçek sabittir. $\psi(x, x_k)$ ise radyal tabanlı fonksiyondur ve aşağıda gösterildiği gibi ifade edilir

$$\psi(x, x_k) = \exp\left\{-\frac{\|(x,x_k)^2\|}{2\sigma^2}\right\}, k = 1,2,\dots,N \tag{5.12}$$

Burada σ, bir grid arama yöntemiyle belirlenen ve tekrarlanan çapraz doğrulama yaklaşımı kullanılarak elde edilen radyal temel fonksiyonunun genişliğidir.

6. UYGULAMA

Görsel programlama uygulamaları için günümüze kadar çeşitli yollara başvurulmuştur (http-9), [6]. Veri çeşidine bağlı olarak, uygulanan deney için kullanılan yöntemlerin doğruluk oranlarına bakılırsa, optimum yöntemle daha uygun bir uygulama gerçekleştirilmesini sağlanabilir. Bütün çalışmalarda MNİST [28] veri seti kullanılarak Python programlama diliyle, gerekli kodlar oluşturulmuştur. Model oluşturulurken önce veriler normalleştirilmiş daha sonra model kurulmuştur. MNİST veri seti toplamda 70000 28x28 pikselli el yazısı görselinden oluşmaktadır. Ayrıca bu el yazısı görsellerine gelen label yani etiket değerleri bulunmaktadır. Bu şekilde bir veri seti Python içinde vektörize edilmiş bir biçimde, bir başka deyişle resimlerin piksel değerleriyle sayısallaştırılmış haliyle, hazır halde bulunur.

6.1. Konvolüsyonel Sinir Ağları Yöntemi ile Yapılan Uygulama

İlk aşamada KSA modellemesi için MNİST veri setinin 10000'i test 60000'i ise training olarak eğitilmiştir. Totalde 10 adımdan sonra %98 doğruluk değeriyle başarım gerçekleştirerek sınıflandırma işlemi tamamlanmıştır. Sınıflandırma yaparken tanımlanmak istenen el yazısı rakam oldukça güçlü bir doğruluk değeriyle tanımlanabilmiştir. Uygulamada 4444. sırada belirtilen rakamın 9 çıkmasına ilişkin bilgilendirme yapılmış ve çıktı olarak elde edilen sonuç bize doğrudan 9 rakamına ulaştırmıştır. KSA adım (Epoch) değerleri ve tanımlanan el yazısı rakam aşağıdaki şekillerde gösterildiği gibidir. Verilerin çıktıları Şekil 6.1'de olup, aşamaların kodları EK-1 de yer almaktadır.

```
Epoch 1/10
1875/1875 [==============================] - 55s 29ms/step - loss: 0.1502 - accuracy: 0.9554
Epoch 2/10
1875/1875 [==============================] - 43s 23ms/step - loss: 0.0512 - accuracy: 0.9846
Epoch 3/10
1875/1875 [==============================] - 43s 23ms/step - loss: 0.0341 - accuracy: 0.9894
Epoch 4/10
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0215 - accuracy: 0.9933
Epoch 5/10
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0148 - accuracy: 0.9952
Epoch 6/10
1875/1875 [==============================] - 42s 23ms/step - loss: 0.0105 - accuracy: 0.9965
Epoch 7/10
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0086 - accuracy: 0.9972
Epoch 8/10
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0061 - accuracy: 0.9978
Epoch 9/10
1875/1875 [==============================] - 42s 23ms/step - loss: 0.0048 - accuracy: 0.9984
Epoch 10/10
1875/1875 [==============================] - 42s 22ms/step - loss: 0.0046 - accuracy: 0.9985
313/313 [==============================] - 3s 8ms/step - loss: 0.0719 - accuracy: 0.9846
```

Şekil 6.1. *Konvolüsyonel Sinir Ağları adım değerleri*

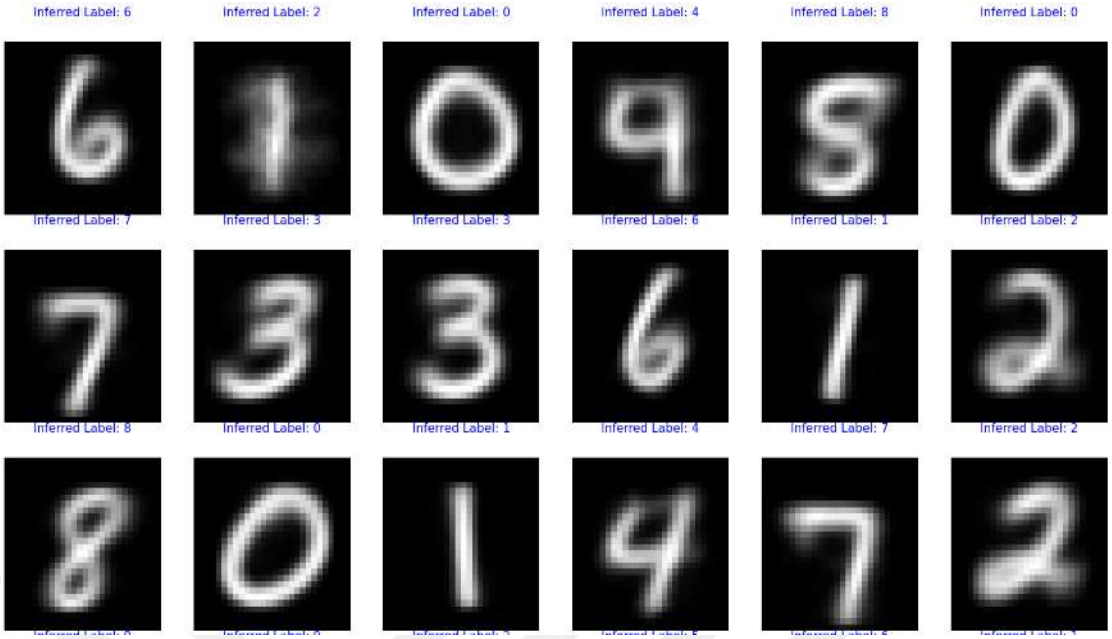
6.2. K-Ortalamalar Yöntemi ile Yapılan Uygulama

İkinci aşamada k-ortalamalar yöntemi ile aynı veri seti yardımıyla sınıflandırma ele alınmıştır. Bu uygulama ile k-ortalamalar yönteminin sınıflandırmada kullanılabilir olduğunu fakat KSA yöntemi kadar doğruluk oranı vermeyeceğini çıktılardan anlayabiliriz. Bu çalışmada doğruluk oranı %80-90 arasında olur. Daha düşük çıktığı uygulamalarda algoritma yeniden optimize edilerek daha yüksek doğruluk oranlarına ulaşmak mümkündür. Veri çıktıları, modelleme ve sınıflandırma kodları EK-2'te olup çıktıları aşağıdaki gibidir.

```
[8 0 4 1 9 2 1 8 1 7 3 1 3 6 1 7 2 1 6 7]
[5 0 4 1 9 2 1 3 1 4 3 5 3 6 1 7 2 8 6 9]
```

Şekil 6.2. *K-ortalamalar ilk veri çıktısı*

Yukarıdaki görsel bize model kurulduğunda istediğimiz ve elde ettiğimiz çıktıyı göstermektedir. Burada veriler birbiriyle uyuşmamaktadır. Daha sonra model geliştirildiğinde ve tekrar optimize edildiğinde yüksek oranda doğrulukla sınıflandırma yapabilir.

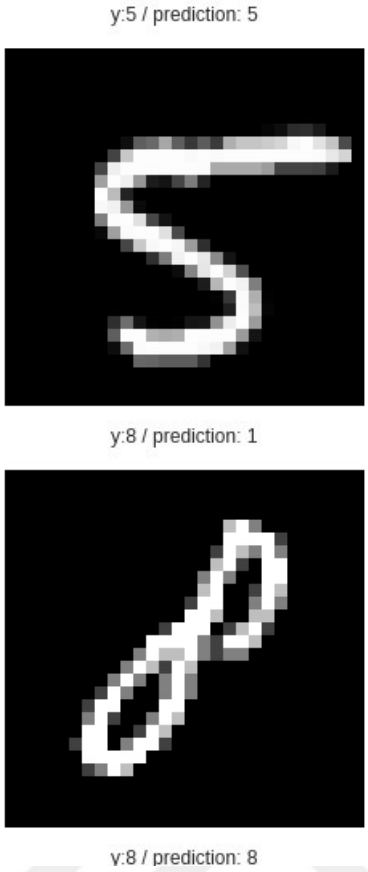


Şekil 6.3. *K-ortalamalar modeli geliştirildikten sonra elde edilen verilerin bir kısmı*

6.3. Lojistik Regresyon Yöntemi ile Yapılan Uygulama

Üçüncü uygulamada lojistik regresyon kullanılarak sınıflandırma yapılmıştır. Burada tekrar MNİST [28] verileri kullanılmıştır. Fakat bu sefer sayı olarak daha az veriye yer verilmiştir. Çıktı olarak aldığımız sonuca bakılırsa, istenilen aralıklarda aldığımız sonuç doğrudur diyebiliriz. Kullanılan görsellerin pikselleri farklı olduğundan çıktımız daha bulanık halde olacaktır. Doğruluk oranı bu uygulamada %91,7 oranındadır.

Lojistik regresyon çoklu verilerle yapılan çalışmaların çoğunda oldukça kullanışlıdır. Bunu alınan doğruluk oranlarının yüksekliğinden de anlayabiliriz. Ayrıca k-ortalamalar yönteminde olduğu gibi optimize etmemize gerek kalmayabilir. Veri çıktıları Şekil 6.4, Şekil 6.5 ve kodları da EK-3' te yer almaktadır.



Şekil 6.4. *Lojistik regresyon görüntü çıktıları*

```
train_acc: 0.859 test_acc: 0.871
```

Şekil 6.5. ***Lojistik regresyon doğruluk oranı çıktısı***

Çoklu regresyon da sınıflandırmalarda lojistik regresyonla benzer şekilde kullanılabilir. Doğruluk oranları neredeyse aynı sonuç verir. Sınıflandırma yaparken en önemli kural kullanılan yöntem için doğru verileri seçmektir. Veri çeşitlerine göre doğruluk oranları da yanlış sonuç verebilir. Hatta sınıflandırma modellemeleri yapılırken bu tarz hatalara oldukça fazla rastlayabiliriz. Yanlış veri yanlış sınıflandırmaya yol açar.

6.4. Karar Ağaçları Yöntemi ile Yapılan Uygulama

Dördüncü uygulamada karar ağaçları sınıflandırma yöntemi ele alınmıştır. Bu yöntemde 70000 adet MNİST verisiyle Python programlama dili kullanılmıştır. Uygulama modellendikten sonra sınıflandırma yapılmış ve istenilen 7 rakamına ulaşılmıştır. Elde edilen doğruluk oranı %85 olmuştur. Buradan anlayabiliriz ki, karar ağaçları yöntemi sınıflandırma için kullanılabilecek yöntemler içine girebilir. Şekil 6.6'da veri görselleri eklenmiştir ve EK-4'te ki gibi kodlanmıştır.

```
0.8511111111111112
```

Şekil 6.6. *Karar ağaçları doğruluk oranı çıktısı*

6.5. Rassal Orman Yöntemi ile Yapılan Uygulama

Beşinci uygulamada rassal orman sınıflandırıcısı MNİST veri setine uygulanmıştır. Uygulamada kullanılan veri ile karar ağaçlarında kullanılan veri aynıdır. Aralarında görülebilir bir fark olarak yaklaşık %97 oranında bir doğruluk oranına ulaşmıştır. Bu durumda ikisi arasında kullanılması daha etkili olan sınıflandırma yöntemi rassal orman yöntemidir. Uygulamanın çıktıları şekil 6.7'de, kodları EK-5'de yer almaktadır.

```
siniflandirma raporu
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       980
           1       0.99      0.99      0.99      1135
           2       0.96      0.97      0.96      1032
           3       0.96      0.96      0.96      1010
           4       0.98      0.97      0.98       982
           5       0.97      0.96      0.97       892
           6       0.98      0.98      0.98       958
           7       0.97      0.96      0.97      1028
           8       0.96      0.96      0.96       974
           9       0.95      0.95      0.95      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

Şekil 6.7. *Rassal orman veri çıktısı*

6.6. Naive Bayes Yöntemi ile Yapılan Uygulama

Altıncı uygulamada naive bayes ele alınmıştır. Burada MNİST verilerinden yararlanılmıştır. Doğruluk oranı %80 olarak bulunmuştur. EK-6'de kodları belirtilmiştir ve görsel çıktısı aşağıdaki gibidir.

```
Overall Accuracy of Naive Bayes Model: 0.8013
0.8013
```

Şekil 6.8. Naive Bayes veri çıktısı

6.7. Destek Vektör Makineleri Yöntemi ile Yapılan Uygulama

Son olarak DVM sınıflandırıcısı ele alınmıştır. MNİST verileri kullanılmış ve %94 oranında başarım sağlanmıştır. Çıktı ve görsel çıktı şekil 6.9'daki gibidir. Sınıflandırma raporundan da anlayacağımız gibi 10 adımda %98 doğruluk oranına ulaşılmıştır. Kodları EK 7 de yer almaktadır.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	45
1	0.96	0.98	0.97	47
2	1.00	0.98	0.99	43
3	1.00	0.98	0.99	47
4	0.95	1.00	0.98	40
5	0.98	1.00	0.99	56
6	1.00	0.98	0.99	49
7	0.97	0.97	0.97	35
8	0.89	0.97	0.93	34
9	1.00	0.93	0.96	54
accuracy			0.98	450
macro avg	0.98	0.98	0.98	450
weighted avg	0.98	0.98	0.98	450

Şekil 6.9. *Destek Vektör Makineleri veri çıktısı*

Görüntü sınıflandırma yöntemleri benzer veriler için sırasıyla ele alınmış ve en iyi sonucu alabilmek için karşılaştırmalar yapılmıştır. Uygulamada kullanılan veriler MNİST veri seti kullanılarak sağlanmıştır. Uygulama için Python programlama dillerinden yararlanılmıştır.

7. SONUÇ

YSA'nın kullanıldığı alanlardan biri olan görüntü sınıflandırma temel alınarak ortaya çıkarılan bu çalışmada istatistiksel yöntemlere de yer verilerek hangi modellemenin daha iyi sonuç verdiğiyle ilgili karşılaştırma yapılmıştır. İstatistiksel yöntemlerden faydalanmak YSA modellemelerine göre daha riskli bir hal almaktadır. Sınıflandırma yapılırken, sınıfların oluşması aşamasında veya veri işlemede YSA'ya göre daha başarısız olabilirler.

Yapılan bir çalışmada, aynı şekilde mnist verileri kullanılarak DVM, Rastgele Orman Sınıflandırıcısı (RFC) ve KNN (K-En Yakın Komşu) içeren en yaygın ve popüler ML Algoritmalarının doğruluklarını KSA ile karşılaştırmaya çalışılmıştır. Ayrıca RFC kullanarak yaklaşık %96,85, SVM kullanarak %97,73, KNN kullanarak %96,80 doğruluk elde ederken, CNN kullanarak %98,72 doğruluk elde edilmiştir [31].

Kushalatha, Prashantha ve Shetty 2021 yılında yayımladıkları makalelerinde, DVM ve k en yakın komşu yöntemleri arasında bir karşılaştırma yapmış ve sonuç olarak; DVM'nin, 0,9 oranında ve K en yakın komşu yönteminin ise 0,96 oranında bir sınıflandırma doğruluğuna sahip olduğunu belirtmişlerdir. Dolayısıyla, K en yakın komşu yönteminin, kullanılan veriler ile yapılan sınıflandırmada (KSA) ile daha iyi performans gösterdiği tespit edilmiştir [30].

Ayrıca Babiker Hamdan Y. ve Prof. Satish istatistiksel yöntemler ve DVM ile birlikte kullanılan istatistiksel yöntemleri karşılaştırdıkları çalışmalarında, en yüksek oranı DVM nin belirlediğini tespit etmişlerdir [32].

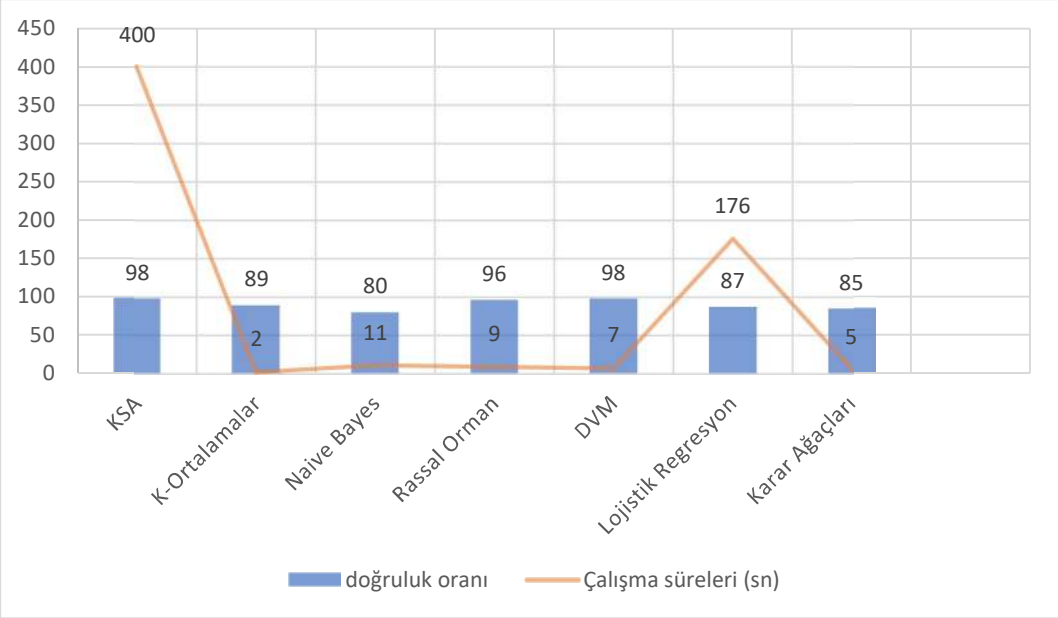
Henrique A.S. ve diğerleri, fashion mnist adlı görsel sınıflandırma verileriyle beraber; KSA, DVM, random forest , multilayer perceptron, k neighbors, logistic regression, karar ağaçları, bayes ve diğer birkaç istatistiksel yöntem arasında karşılaştırma yapmışlardır. En yüksek sonucu ortalama %98 doğruluk oranıyla KSA ve en düşük sonucuda %51 ile bayes yönteminin verdiğine kanaat getirmişlerdir [1].

Bu durumda, benzeri çalışmalarla doğruluk oranları karşılaştırıldığında, bariz olarak KSA yönteminin kullanışlı olduğunu söyleyebiliriz.

Uygulama kısmında 7 uygulamaya yer verilmiştir. Bu karşılaştırmanın sonunda, görüntü sınıflandırma yapılırken, çoklu verilerle çalışabilen uygulamalarda konvolüsyonel sinir ağları kullanımının daha etkili olduğu görülmüştür. Ayrıca regresyon analizi yöntemleri arasında, karar ağaçları ve lojistik regresyon yöntemlerinin de diğerlerine göre daha iyi sonuçlar verdiği tespit edilmiştir. Karşılaştırma sonuçları aşağıda yer almaktadır.

Tablo 7.1. *Sonuçların karşılaştırılması*

Sınıflandırma yöntemi	**Veri Çeşidi**	**Çalışma Süresi (sn)**	**Doğruluk Oranı**
KSA	MNİST(70000 adet el yazısı verisi)	400 (adım başı yaklaşık 40 sn)	%98
DVM	MNİST(70000 adet el yazısı verisi)	7	%98
Rassal orman	MNİST(70000 adet el yazısı verisi)	9	%96
K – Ortalamalar	MNİST(70000 adet el yazısı verisi)	2	%89
Lojistik Regresyon	MNİST(70000 adet el yazısı verisi)	176	%87
Karar Ağaçları	MNİST(70000 adet el yazısı verisi)	5	%85
Naive Bayes	MNİST(70000 adet el yazısı verisi)	11	%80



Şekil 7.1. *Renklendirilmiş sonuç grafiği*

Çalışılan MNİST veri setlerinden 70000 görsele sahip olan veri seti temel alınarak yapılan karşılaştırmada; kullanılan yöntemin hacmi arttıkça test süreside orantılı olarak artıyor diyebiliriz. 7 yöntem arasında sınıflandırma için en uygun olan yöntem KSA olarak belirlenmiştir. Bunun bir nedeni olarak, katmanları sayesinde veri kaybının en aza indirgenmesini gösterebiliriz. Diğer yandan, geriye kalan yöntemlerin de doğruluk oranlarının tatmin edici boyutlarda olduğu görülmektedir. Bu açıdan ikinci bir seçenek olarak diğer yöntemlerden de faydalanabiliriz.

Burada ele alınan yöntemler veri yapısına göre geliştirilebilir ve sürdürülebilir yöntemlerdir. Daha önce yapılan çalışmalar temel alındığında benzer olarak KSA'ların başarı oranlarının daha yüksek olduğunu söyleyebiliriz ve en düşük olarak bayes yöntemleridir diyebiliriz.

KAYNAKÇA

[1] Henrique A.S., Fernandes A.M.R. , Lyra R. , Leithardt V.R.Q., Correia S.D., Crocker P. , Dazzi R.L.S. (2021). "Classifying Garments from Fashion-MNIST Dataset Through CNNs". *Advances in Science, Technology and Engineering Systems Journal Vol. 6,* No. 1, 989-994.

[2] Ayhan, E., Karslı, F. ve Tunç, E. (2003). Uzaktan Algılanmış Görüntülerde Sınıflandırma ve Analiz (Classıfıcatıon Of Remote Sensıng Images And Analysıs), *Harita Dergisi,* 130, 32-46.

[3] Yingchun L. C. (2014). Random Forest Algorithm İn Big Data Environment. *Computer Modelling and New Technologies Journal*, 18(12A) 147-151.

[4] Ned H. (2010). Random Forests: An algorithm for image classification and generation of continuous fields data sets. *Report of International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences* .

[5] Aytekin H.T. (2021). Makine Öğreniminin Araştırmacıların Veri Analizi Bağlamında Potansiyel Önemi*. Ufuk Üniversitesi. 1. Uluslararası Sosyal Bilimler Kongresi'nde sunulmuş bildiri*.

[6] Çınar U.K.(2002). *Tensorflow Lite Modeli ile Colab Üzerinden Görüntü Sınıflandırma: Derin Öğrenme Uygulaması.* Yüksek Lisans Tezi. İstanbul: Yıldız Teknik Üniversitesi,.

[7] Cömert, O., Hekim, M. Ve Adem, K. (2019). Faster R-CNN Kullanarak Elmalarda Çürük Tespiti (Bruise Detection in Apples using Faster R-CNN), *Uluslararası Mühendislik Araştırma ve Geliştirme Dergisi*, Cilt/Volume:11 Sayı/Issue:1.

[8] Özen, M. (2009). Uzaktan algılama da Piksel ve Nesne Tabanlı Sonuçlarının Değerlendirilmesi, Doğruluk Analizlerinin Yapılması ve CBS Ortamına Aktarımı. *Anadolu Üniversitesi Yayınları,* S. 101 115.

[9] Öztürk D. (2020). *Piksel Tabanlı Sınıflandırma, Obje (Nesne) Tabanlı Sınıflandırma*, Samsun.

[10] Yılmaz E. Ö. (2020). *Uzaktan Algılanmış Verilerin Derin Öğrenme Yöntemiyle Sınıflandırılması.* Yüksek Lisans Tezi. Gebze: Gebze Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

[11] Yiğit, A.Y. ve Uysal, M. (2019). Nesne Tabanlı Sınıflandırma Yaklaşımı Kullanılarak Yolların Tespiti. *Türkiye Fotogrometri Dergisi*, 1(1); 17-24.

[12] Dovbnych M., Wójcik M.P. (2021). "A comparison of conventional and deep learning methods of image classification". *Journal Computer Science Institute,* JCSI 21 (2021) 303–308.

[13] Kayalı, N.Z. ve İlhan Omurca, S. (2021). Konvolüsyonel Sinir Ağları (CNN) ile Çin Sayı Örüntülerinin Sınıflandırması. *Journal of Computer Science.* ISSN: 2548-1304 Volume: IDAP-2021, Issue: Special , pp: 184-191.

[14] Jin C., De-lin L., Fen-xiang M. (2009). An Improved ID3 Decision Tree. *4th International Conference on Computer Science & Education* 978-1-4244-3521-0/09.

[15] Akar, Ö., Güngör, O., ve Akar, A. (2012). Rastgele Orman Sınıflandırıcısı ile Arazi Kullanım Alanlarının Belirlenmesi. *Jeodezi ve Jeoinformasyon Dergisi,* Cilt 1, Sayı 2, ss.139-146, Kasım 2012, Dergi No.106, Doi: 10.9733/jgg.241212.1t.

[16] Yasaka, K., Akai, H., Kunimatsu, A., Kiryu, S., and Abe, O. (2018). Deep Learning With Convolutional Neural Network İn Radiology. *Japanese Journal of Radiology*, 36:257–272.

[17] Klang, E. (2018). Deep learning and medical imaging. *J Thorac Dis,*10(3):1325–1328.

[18] Soffer, S., Cohen, B., Shimon, O., Amitai, M.M., Greenspan, H., and Klang, E. (2019). Convolutional Neural Networks for Radiologic Images. *Radiology Journal*, 290:590–606.

[19] Bergendal R., Rohlén A. (2019). A Comparison of Training Algorithms When Training a Convolutional Neural Network For Classifying Road Signs. 142X EECS/KTH 2019-05.

[20] Habermann, D., Hata, A., Wolf, D. And Osório, F.S. (2013). Artificial Neural Nets object recognition for 3D point clouds. *Report of Brazilian Conference on Intelligent Systems*, 19-24 October 2013 Date Added to IEEE *Xplore*: 30 January 2014,Electronic ISBN:978-0-7695-5092-3.

[21] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning,* 29, 131–163.

[22] Efron, B., Hastie, T., Johnstone and Itıbshıranı, R. (2004). Least Angle Regression. *The Annals of Statistics,* Vol. 32, No. 2, 407–499.

[23] Saberioon, M., Císaˇr, P., Labbé, L., Souˇcek, P., Pelissier, P. and Kerneis, T. (2018). Comparative Performance Analysis of Support Vector Machine, Random Forest, Logistic Regression and k-Nearest Neighbours in Rainbow Trout (Oncorhynchus Mykiss) Classification Using Image-Based Features. *Sensors,* 18(4) 1027.

[24] Wald A. (1947). *The Annals of Mathematical Statistics.* Vol. 18, No. 4 (Dec., 1947), pp. 586-589 (4 pages).

[25] Wang L., Zhang K., Liu X., Long E., Jiang J., YingyingAn, Zhang J., Liu Z., Lin, Li Z.X., JingjingChe, QianzhongCao, Li J., XiaohangWu, DongniWang, Li W., Lin H. (2017). Comparative analysis of image classification methods for automatic diagnosis of ophthalmic images. *Scientific Reports,* | 7:41545 | DOI: 10.1038/srep41545 1.

[26] Alan A., Karabatak M. (2020). Veri Seti-Sınıflandırma İlişkisinde Performansa Etki Eden Faktörlerin Değerlendirilmesi. *Fırat Üniversitesi Müh. Bil. Dergisi,* 32(2), 531-540.

[27] Özdemir Ö., Aydoğdu E. (2020). Deep Learnıng Classıfıcatıon Wıth Convolutıonal Neural Networks. *Report of V. Internatıonal Scıentıfıc And Vocatıonal Studıes Congress – Scıence And Health (Bılmes Sh 2020)*.

[28] MNİST veri kütüphanesi.

[29] Keskenler M.F., Keskenler E.F. (2017). Geçmişten Günümüze Yapay Sinir Ağları ve Tarihçesi. *Takvim-i Vekayi*, ISSN: 2148-0087 Basım Tarihi: 29 Aralık 2017 / 11 Rebîulâhir 1439 Cilt: 5 No: 2 Sayfa: 8-18 (2017).

[30] Kushalatha. M. R., Prashantha H. S., Shetty R. (2021). Comparison of Image Classification Techniques and Its Applications. *Asian Journal of Convergence in Technology,* ISSN NO: 2350-1146 I.F-5.11.

[31] A. R. Rastogi, C. Verma, D. Sharma and P. K. Goyal (2022). "A Comparative Statistical Analysis Between ML Algorithms & DNN Techniques Using MNIST Dataset,". *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India, 2022, pp. 317-321, doi: 10.1109/ICAC3N56670.2022.10074302.

[32] Hamdan Y.B., Prof. Satish (2021). "Construction of Statistical SVM based Recognition Model for Handwritten Character Recognition". *Journal of Information Technology and Digital World,* Vol. 03/ No. 02 Pages: 92-107.

http-1: https://acikders.ankara.edu.tr/pluginfile.php/187507/mod_resource/content/1/Ders 7_Image_processing%20%5BUyumluluk%20Modu%5D.pdf (Erişim tarihi :7.04.2022).

http-2: https://blog.esri.com.tr/tag/reclassify/ (Erişim tarihi: 12.04.2022).

http-3: Derin Öğrenme ve Evrişimsel Sinir Ağları. Erişim Adresi: https://medium.com/deep-learning-from-deepest/deri%CC%87n-%C3%B6%C4%9Frenme-ve-evri%CC%87%C5%9Fi%CC%87msel-si%CC%87ni%CC%87r-a%C4%9Flari-446dc8a8d2f (Erişim tarihi: 16.06.2022).

http-4: http://staff.fit.ac.cy/eng.ls/Skevi/ACET450/ACET%20450%20Supervised%20classification%20II.pdf (Erişim tarihi: 26.04.2022).

http-5: https://erdincuzun.com/makine_ogrenmesi/kumeleme-algoritmalari-k-means-algoritmasi/ (Erişim tarihi:15.07.2022).

http-6: https://kadirguzel.medium.com/geriyay%C4%B1l%C4%B1ml%C4%B1-%C3%A7ok-katmanl%C4%B1-yapay-sinir-a%C4%9Flar%C4%B1-1-47daa3856247 (Erişim tarihi: 09.10.2022).

http-7: https://www.ahmetcevahircinar.com.tr/2017/08/09/matlab-ile-derin-ogrenmeye-giris-introducing-deep-learning-with-matlab/ (Erişim tarihi: 11.10.2022).

http-8: https://iq.opengenus.org/cart-algorithm/ (Erişim tarihi: 09.10.2022).

http-9: https://kaggle.com (Erisim tarihi: 25.11.2022).

http-10: https://yigitsener.medium.com/makine-%C3%B6%C4%9Frenmesinde-train-validation-ve-test-kavramlar%C4%B1-ile-python-uygulamas%C4%B1-a60214fb5448 (Erisim tarihi: 11.10.2022).

EKLER

EK-1

```
from sklearn.datasets import load_digits

from keras.datasets import mnist
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedShuffleSplit


from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D

from tensorflow.keras.layers import MaxPool2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dropout

from tensorflow.keras.layers import Dense


(trainX,trainY) , (testX,testY)=mnist.load_data()

trainX = trainX.reshape((trainX.shape[0], trainX.shape[1], trainX.shape[2], 1))

testX = testX.reshape((testX.shape[0], testX.shape[1], testX.shape[2],1))
```

```
print(testX.shape)

print(testX.shape)

trainX = trainX /255

testX = testX /255

model=Sequential()

model.add(Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))

model.add(MaxPool2D(2,2))

model.add(Flatten())

model.add(Dense(100,activation='relu'))

model.add(Dense(10,activation='softmax'))


model.fit(trainX,trainY,epochs=10)

model.evaluate(testX,testY)

from teensorflow.keras.optimizers import RMSprop,SGD,Adam

adam=Adam(lr=0.001)

model.compile(optisadmizer='adam', loss='categorical_crossentropy', metrics = ['acc'])

bs=30

import cv2

import glob

from tensorflow.keras s.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator( rescale = 1.0/255. )

test_datagen  = ImageDataGenerator( rescale = 1.0/255. )

train_generator=train_datagen.flow_from_directory(train_dir,batch_size=bs,class_mode
='categorical',tar  get_size=(180,180))

validation_generator =  test_dat agen.flow_from_directory(validation_dir,
```

```
batch_size=bs,
class_mode  = 'categorical',
target_size=(180,180))
```

EK-2

```
import sys

import sklearn

import numpy as np

import matplotlib.pyplot as plt

print('Python: {}'.formmat(sys.version))

print('Scikit-learn: {}'.format(sklearn.__version__))

print('NumPy: {}'.format(np.__version__))

import tensorflow as tf

from tensorflow.keras.datasets import mnist

(X_train, y_train), (X_teest, y_test) = mnist.load_data()


print("Test: {}".format(X_test.shape))

fig, axs = plt.subpljots(3, 3, figsize = (12, 12))

plt.gray()


for i, ax in enumerate(axs.flat):

   ax.imshow(x_train[i])

   ax.axis('off')

   ax.set_title('Nuember {}'.format(y_train[i]))

   plt.show()

X_train = X_train.reshape(len(X_train), -1)

print(X_train.shape)

from sklearn.cluster import MiniBatchKMeans
```

```
kmeans = MiniBattchKMeans(n_clusters=n_digits)
kmeans.fit(X_train)
from sklearn.metrics import homogeneity_scoreİ
def calc_metricIs(estimator, data, labels):

    print('Number of Clusters: {}'.format(estimator.n_clusters))
    inertia = estimator.inertia_
    print("Inertia: {}".format(inertia))
    homogbjeneity = homiogeneity_score(labels, estimator.labels_)
    print("Homoogeneity score: {}".format(homogeneity))
    return inertia, homogeneity
from sklearn.metrics import accuracy_score

for n_clusters in clusters:
    estimator = MiniBatcMhKMeans(n_clusters=n_clusters)
    estimator.fit(X_train)
    inertia, homo = calc_metrics(estimator, X_train, y_train)
    iner_list.appeoınd(inertia)
    homo_list.append(homo)

    cluster_labels = infer_ıcluster_labels(estimator, y_train)
    prediction = infer_data_labels(estimator.labels_, cluster_labels)
    acc = accurkkacy_score(y_train, prediction)
    acc_list.append(acc)
```

```
print('Accuracy: {}\n'.format(acc
```

EK-3

```
from sklearn.datasets import fetch_openml
from sklearn.lilnear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils import check_random_state
print(__doc__)
t0 = time,time()
train_size = 60000
test_size = 10000
random_state = check_random_state(0)
permutation = random_state.permutation(X.shape[0])
X = X[permutation]
y = y[permutation]
X = X.reshape((X.shape[0], -1))
testimg    = mnist.test.images
testlabel  = mnist.test.labels
print ("MNIST loaded")
x = tf.placeholder("float", [None, 784])
y = tf.placeholder("float", [None, 10])  # None is for infinite
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
actv = tf.nn.softtmax(tf.matmul(x, W) + b)
accr = tf.reduce_mean(tf.cast(pred, "float"))
init = tf.global_variables_initializer()
training_epochs = 50, batch_size      = 100, display_step    = 1
for epoch in range(training_epochs):
avg_cost = 0.
num_batch = int(mnist.train.num_examples/batch_size)
for i in ranege(num_batch):
batch_xs, batch_ys = mnist.train.next_batch(batch_size)
sess.run(optm. feed_dict={x: batch_xs, y: batch_ys})
```

```
feeds = {x: batchh_.xs, y: batch_ys}
avgg_cost += sees.run(cost, feed_dict=feeds)/num_batch
if epoch % display_step == 0:feeds_train = {x: mnist.train.images, y: mnist.train.labels}
train_acc = sess.cfrun(accr, feed_dict=feeds_train)
test_acc = sess.run(accr, feed_dict=feeds_test)
print ("Epoch: %03d/%03d cost: %.9f train_acc: %.3f test_acc: %.3f")
print ("DONE")
actv = tf.nn.softmax(tf.matmul(x, W) + b)
cost = tf.redmuce_mean(tf.squared_difference(y, actv))
optm = tf.train.GradientDescentOptimizer(0.01).minimize(cost
pred = tf.equal(tf.argmax(actv, 1), tf.argmax(y, 1))
accr = tf.reducje_mean(tf.cast(pred, "float"))
init = tf.glolbal_variables_initializer()
training_epochs = 50,batch_size      = 100,display_step    = 10
for epoch in range(training_epochs):
avg_cost = 0.
for i in range(num_batch):
batch_xs, batch_ys = mnist.train.next_batch(batch_size)
sess.run(optmi, feedd_dict={x: batch_xs, y: batch_ys})
feeds =, {x: batch_xs, y: batch_ys}avg_cost += sess.run(cost, feed_dict=feeds)/num_batc
if epoch % display_step == 0:
train_acc = sesss.run(accr, feed_dict=feeds_train)
prediction = sess.ruun(actv, feed_dict={ x : mnist.test.images })nsample = 5
randidx = np.random.randint(mnist.test.images.shape[0], size=nsample)
plt.yticks([])
plt.title('y:' + str(curr_label) + ' / prediction: ' + str(np.argmax(prediction[i])))
```

EK-4

```
mnist = load_digits()

mnist.data.shape
mnist.data[0]
import matplotlib.pyplot as plt
%matplotlib inline
plt.imshow(mnist.imagesh[2],cmap='gray')
from sklearn.model_selection import train_test_split
trainX,testX,trainY,testY = train_tedst_split(mnist.data, mnist.target)
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(trainX,trainY)
dt.score(testX,testY)
```

EK-5

```
mnist = load_digits()

mnist.images.shape

import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from tqdm import tqdm
from time import time
%matplotlib inline
plt.imshow(mnist.images[9],cmap='gray')
from sklearn.model_selection import train_test_split
trainX,testX,trainY,testY = train_test_split(mnist.data, mnist.target)
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=90, n_jobs=-1)
rf.fit(trainX,trainY)
rf.score(testX,testY)
```

EK-6

```
def naive_bayes(data,label):
    n_s,n_f=data.shape
    classes=np.unique(label)
    n_c=len(classes)
    totall_data=np.zeros([n_s,n_f+1])
    total_ddata[:,:-1]=data
    total_data[:,-1]=label
    np.randomd.shuffle(total_data)
    trainX=total_data[:60000,:]
    np.rdandom.shuffle(trainX)
    testX=total_data[60000:,:]
    np.random.shuffle(testX)
    testX_c=testX[:,:-1]
    testX_l=testX[:,-1]
    mean_v=np.zeros([n_c,n_f])
    var_v=np.zeros([n_c,n_f])
    c_prob=[]
    confusiion_matrix=np.zeros([n_c,n_c])
    d_acc=[]

    for c in classes:
        trainX_c=trainX.[trainX[:,-1]==c]
        trainX_c=trainX_c[:,:-1]
        c_prob.append
(len(trainX_.kc)/len(trainX))
        mean_v[int(c),:]=trainfX_c.mean(axis=0)
        var_v[int(c),:]=trainX_c.var(axis=0)

    var_v=var_vop+1000
    count=0
    for i in range(testX.shape[0]):
```

```
        lists=[]
        for j in range(n_c):
            numerator=np.exp(-((testX_c[i]-mean_v[j])**2)/(2*var_v[j]))
            denominator=np.sq
rt(2*np.pi*(var_v[j]))
            prob_xc=numhherator/denominator
            ratio=np.sum(np.log(prob_xc))
                    count=count+1 f
            confusion_mattrix[int(testX_l[i])][int(testX_l[i])]=confusion_matrix[int(testX_l[
i])][int(testX_l[i])]+1
        else:
            for k in ranyge(n_c):
                if pred == k:
                    confusion_matrix[int(testX_l[k])][int(testX_l[i])]=confusion_matrix[int(tes
tX_l[k])][int(testX_l[i])]+1
    for l in classes:
        check=testXjj[thjüestX[:,-1]==l]
        a=(confusion_matrixuj[int(l)][int(l)])/chehck.shape[0]
        d_acc.append(a)



    o_acc=count/testX.shape[0]
    return(d_acc,o_acc,confusion_matrix,mean_v,var_v)

(digit_accuracyıo,overall_accuracy,matrix,mean_v,var_v)=naive_bayes(data,label)
print('Overall Accuracy of Naive Bayes Model: '+str(overall_accuracy))
overall_accuracy
```

EK-7

```
print('training datanin buyuklugu',trainX.shape)
print('training labelsin buyuklugu', trainY.shape)
print('test datanın buyuklugu',testX.shape)
print('test labelsin buyuklugu', testY.shape)
trainX=trainX/255
testX=testX/255
print ('training datanin normallestirmeden sonraki buyuklugu',trainX.shape)
print ('test datanin normallestirmeden sonraki buyuklugu',testX.shape)
pipe_1= Pipeline([('scaler', MinMaxScaler()),("classifier", SVC(kernel= 'linear', C=1))])

pipe_1.fit(trainX,trainY.ravel())
acc=cross_val_score(pipe_1, trainX,trainY.ravel(),cv=2)
print("training accuracy:{:.2f} %". format(acc.mean()*100))
y_pred= pipe_1.predict(testX)
CR= classification_report(testY,y_pred)
print ('siniflandirma raporu \n')
print(C
```

ÖZGEÇMİŞ

Adı Soyadı: Ece Aydoğdu

Yabancı Dili: İngilizce

Eğitim Durumu (Kurum ve Yıl)

Lise: Yatağan Lisesi (2006-2010)

Lisans: Karadeniz Teknik Üniversitesi (2010-2017)

Yüksek Lisans: Eskişehir Teknik Üniversitesi (2020-)

Çalıştığı Kurum/Kurumlar ve Yıl:

Freelance Grafiker – 2013-2015

Limon Ağacı Kişisel Gelişim Kursu- 2019-2020

Freelance Analist – 2020-

Yayınları (SCI ve diğer): Aydoğdu Ece, ÖZDEMİR ÖZER, "Deep Learning Classification With Convolutional Neural Networks", V. International Scientific And Vocational Studies Congress-Science And Health (BILMES 2020)