K.K.T.C.
YAKINDOĞU ÜNİVERSİTESİ
SAĞLIK BİLİMLERİ ENSTİTÜSÜ

BİYOİSTATİSTİK TEMELLİ BİLİMSEL ARAŞTIRMALARDA DERİN ÖĞRENME UYGULAMALARI

Hüseyin KUTLU

BİYOİSTATİSTİK PROGRAMI

YÜKSEK LİSANS TEZİ

LEFKOŞA
2018

K.K.T.C.
YAKINDOĞU ÜNİVERSİTESİ
SAĞLIK BİLİMLERİ ENSTİTÜSÜ

BİYOİSTATİSTİK TEMELLİ BİLİMSEL ARAŞTIRMALARDA DERİN ÖĞRENME UYGULAMALARI

Hüseyin KUTLU

BİYOİSTATİSTİK PROGRAMI

YÜKSEK LİSANS TEZİ

LEFKOŞA
2018

KABUL ve ONAY SAYFASI

K.K.T.C. Yakındoğu Üniversitesi Sağlık Bilimleri Enstitüsü Müdürlüğüne,

Bu çalışma jürimiz tarafından Biyoistatistik Ana Bilim Dalı Biyoistatistik Yüksek Lisans Programı Yüksek Lisans Tezi olarak kabul edilmiştir.

Juri Başkanı: ,,

Prof. Dr. S Yavuz SANİSOĞLU

Üye (Danışman): …………………………………..

Doç. Dr. İlker ETİKAN

Üye: ………………………………….

Yrd. Doç. Dr. Özgür TOSUN

ONAY:

Bu tez Yakındoğu Üniversitesi Lisans Üstü Eğitim – Öğretim ve Sınav Yönetmenliği’nin ilgili maddeleri uyarınca yukarıdaki jüri üyeleri tarafından uygun görülmüş ve Enstitü Yönetim Kurulu kararıyla kabul edilmiştir.

………………

Enstitü Müdürü

Prof. Dr. Hüsnü Can BAŞER

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda patent ve telif haklarını ihlal edici etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tezde kullanılmış olan tüm bilgi ve yorumlara kaynak gösterdiğimi beyan ederim. 04.06.2018

Hüseyin KUTLU

“Sevgili aileme ithaf ediyorum.”

TEŞEKKÜR

Tez çalışmam sırasında benden yardımlarını ve desteğini, sabır ve hoşgörüsünü eksik etmeyen, tecrübesinden yararlandığım kıymetli hocam, tez danışmanım Doç. Dr. İlker ETİKAN hocama,

Yüksek lisans eğitimim boyunca bilgi ve tecrübeleri ile beni yönlendiren sevgili hocalarım Yrd. Doç. Dr. Özgür TOSUN ve Prof. Dr. Yavuz SANİSOĞLU'na,

Tanıdığım günden bu yana hayatımdaki tüm zorluklarda, tüm güzelliklerde yanımda olan, yardımını, anlayışını, hoşgörüsünü, eksik etmeyen sevgili eşim Emine Şahin Kutlu'ya

Doğduğum günden bu yana beni okuma ve öğrenme aşkıyla büyüten, her türlü fedakârlığa her zaman hazır olan, maddi ve manevi desteğini esirgemeyen, ilk hocam, sevgili babam İbrahim Üryan Kutlu'ya ve annem Aysel Kutlu'ya, Mahmut ve Ömer abilerime, doğumunu heyecanla beklediğim can parçama…

Sevgi, saygı ve teşekkürlerimi sunarım.

Hüseyin KUTLU

İÇİNDEKİLER

ÖZET

Biyoistatistik Temelli Bilimsel Araştırmalarda Derin Öğrenme Uygulamaları

Günümüzde, daha önce yalın bilim dalları olarak bilinen mühendislik bilimleri, biyolojik bilimler ve bilişim bilimlerinin birbirleriyle etkileşimleri sonucu, yeni alt bilim dalları ve çalışma alanları ortaya çıkmıştır. Bu alanlardan biri de Biyoistatistik ve Tıp Bilişimidir. Tıp bilişimi, bilişim teknolojilerinin tıp alanlarında kullanılması olarak tanımlanabilir. Tıp bilişimi, bir taraftan bilişim teknolojilerinin gelişmelerinden, diğer taraftan tıptaki gelişmelerden beslenmektedir. Böylelikle sağlık hizmetlerindeki çözümü zor sorunların çözümüne yardımcı olmaktadır [1]. İlerleyen yıllarda, tıbbi bilişimin tıpta anatomi kadar değerli olacağı söylenebilir.

Tıp bilişimi başta tıp ve yazılım mühendisliği olmak üzere elektronik mühendisliği, makine mühendisliği, biyomedikal mühendisliği, biyoloji, istatistik, matematik gibi bilim dallarından faydalanan çok disiplinli bir alandır.

Bilişim teknolojilerindeki yeni gelişmeler tıbbi cihazlara veya uzman sistem yazılımlarına uygulanmaktadır. Böylece daha düşük maliyetli, daha doğru ve hızlı sonuçlar veren tıbbi cihazlar üretilmektedir. Tıbbi cihaz yazılımlarında makine öğrenmesi geniş bir yer tutmaktadır. Makine öğrenmesindeki çığır açan gelişimler, Grafik İşlem Birimleri (Graphical Processing Unit, GPU) kullanılarak elde edilmektedir. GPU' lar daha çok, büyük oranda etiketlenmemiş eğitim verilerinden öznitelik saptama yapabilen sistemler oluşturmak için kullanılmaktadırlar. Derin öğrenme, ileri teknolojik GPU'ların kullanıldığı çok seviyeli "derin" nöral ağlar ile önemli derecede araştırma yapıldığı bir alandır.

Bu çalışmada, tıbbi veriler üzerinde derin yapay sinir ağları kullanılarak, sınıflandırma işlemi gerçekleştirilmiştir. Bu amaç ile öncelikle derin öğrenme algoritmaları hakkında bilgi verilmiştir. Çalışmanın amacı derin öğrenme algoritmalarının Biyoistatistik ve Tıp Bilişimi Anabilim Dalında veri sınıflandırmada yaygınca kullanılan sınıflandırma algoritmalarına karşı performansını göstermektir.

Bu tez çalışmasında veri seti olarak UCI Machine Learning Repository'de ücretsiz kullanıma açık olan veri setlerinden "Pima Indians Diabet" veri seti (https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes)kullanmıştır. [2]. Veri setinde; hamilelik sayısı, glikoz konsantrasyonu, diastolik kan basıncı, triseps kalınlığı, iki

saatlik serum insülin değeri, vücut kitle indeksi, aile yakınlık fonksiyonu, yaş ve kadının şeker hastası olup olmadığını belirten 8 öznitelik ve bu öz niteliklerin sınıfını “şeker hastalığı var” veya “şeker hastalığı yok” şeklinde belirten hedef nitelik olmak üzere dokuz nitelik bulunmaktadır.

Diyabet hastalığı pankreasın vücuda yetecek kadar insülin üretememesi veya vücudun ürettiği insülini doğru bir şekilde kullanamaması sonucu oluşan kronik bir hastalıktır. Uzun sürede kalp damar hastalıkları, göz hastalıkları, böbrek hastalıkları gibi ciddi komplikasyonlar ortaya çıkarabilen diyabet, tedavi harcamalarının yüksekliği ve iş gücü kaybı nedeni ile hastaya sosyoekonomik yük getirmesinden dolayı önemli bir sağlık sorunudur [3]. Bu çalışmada kişiye ait fiziksel özellikler ve kan testi verilerinin veri madenciliği yöntemleri ile kişilerin diyabet hastası olup olmadıklarının belirlenmesi amaçlanmıştır. Yapılan çalışmada 768 kadına ait veriler Dünya Sağlık Örgütü(WHO) kriterlerine göre düzenlenmiş, bu veriler veri seti haline getirilerek farklı sınıflandırma algoritmaları uygulanmış ve sonuçlar kıyaslanmıştır. Kıyaslama sonucunda, geliştirilen derin yapay sinir ağı modeli %89 duyarlılık, %82 hassasiyet ile naive bayes (%76 hassasiyet,%77 duyarlılık), K-neighbors (%80 hassasiyet,%80 duyarlılık), Lojistik Regresyon(%80 hassasiyet,%80 duyarlılık),Karar Ağaçları(%72 hassasiyet,%72 duyarlılık), random forest (%71 hassasiyet,%72 duyarlılık), SVM (%80 hassasiyet,%80 duyarlılık) sınıflandırma algoritmalarını geride bırakmıştır.

Anahtar Kelimeler: Derin Öğreneme, Derin Yapay Sinir Ağları

ABSTRACT

Deep Learning Applications in Biostatistics Based Scientific Research

Today, new interdisciplinary science and research fields have emerged with the intertwining of biological sciences, engineering sciences and informatics, which were previously considered as singular sciences. One of these areas is Biostatistics and Medical Informatics. Medical informatics can be generally defined as the application of information technology in medical fields. While medical informatics, on the one hand, takes power from the rapid development of information technologies, on the other hand, it receives power from the developments in medicine. Thus, it helps solving the difficult problems in health care services [1]. it can be said that medical informatics will be as valuable as the anatomy in the following years.

Medical informatics is a multidisciplinary field taking mainly the advantage of medical and software engineering as well as electronic engineering, mechanical engineering, biomedical engineering, biology, statistics and mathematics.

New developments in information technologies are being applied to medical devices or expert system software. In this way, lower costed medical devices which give more accurate and faster results are produced. Machine learning in medical device softwares has a large place. Breakthrough improvements in machine learning have been achieved using the Graphical Processing Unit (GPU). GPUs are especially used to create systems that can perform feature detection from untagged training data in large quantities. Deep learning is a field where advanced technological GPUs are used and considerable investment and research with multi-level "deep" neural Networks.

In this study, classification was performed using deep artificial neural networks on medical data. For this purpose, information about deep learning algorithms is given. The aim of the study is to show the performance against the classification algorithms commonly used in data classification in Biostatistics and Medical Informatics Division of deep learning algorithms.

In this thesis study, "Pima Indians Diabet" dataset (https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes) was used as the dataset which is free to use in UCI Machine Learning Repository. In the data set; 8 attributes indicating the number of pregnancies, glucose concentration, diastolic blood pressure,

triceps thickness, two hour serum insulin value, body mass index, family proximity function, age and whether or not the woman is diabetic, and the class of these qualities as "diabetes" or " there is no disease "as the target attribute.

Diabetes is a chronic disease in which the pancreas can not produce enough insulin to the body or use the body's insulin properly. Diabetes, which can cause serious complications such as cardiovascular diseases, eye diseases and kidney diseases for a long time, is an important health problem due to the high cost of treatment and the socioeconomic burden on the patient due to the loss of work power [3]. In this study, the physical characteristics of the person and data mining methods of blood test data were used to determine whether or not people had diabetes. In the study, 768 women's data were organized according to the World Health Organization (WHO) criteria, these data were converted into data sets and different classification algorithms were applied and the results were compared. As a result of the comparison, the developed artificial neural network model was found to have %89 precission, %82 sensitivity, naive bayes (%76 precission, %77 sensitivity), K-neighbors (%80 precission, %80 sensitivity), Logistic Regression (%80 precission,%80 sensitivity), random forest (71% precission, 72% sensitivity), SVM (80% precission, 80% sensitivity) classification algorithms.

Keywords: Deep Learning, Deep Neural Networks

SİMGELER VE KISALTMALAR DİZİNİ

GPU: Grafik İşleme Ünitesi - Graphichs Processing Unit

CPU: Merkezi işlem birimi - Central Processing Unit

DNN: Derin Sinir Ağları – Deep Neural Networks

RNN: Tekrarlayan Sinir Ağları – Recurrent Neural Networks

LSTM: Uzun Kısa Süreli Bellek - Long Short-Term Memory

CNN: Evrişimsel Sinir Ağları – Convolutional Neural Networks

UCI: Makine öğrenmesi ve akıllı sistemler merkezi - Center for machine learning and Intelligent Systems

YSA: Yapay Sinir Ağı

AUC: Eğri altındaki alan - Area Under Curve,

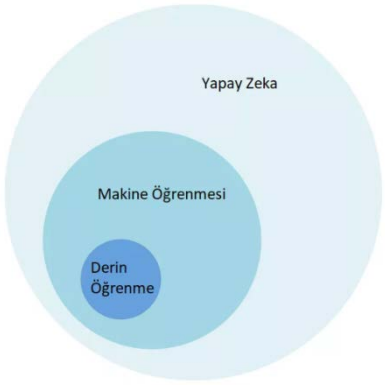
ROC: Alıcı işlem karakteristikleri - Receiver Operating Characteristics

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

1. GİRİŞ

Yapay Zekâ; bilgisayarları, insan ve hayvanlara özgü öğrenme, öngörme, karar verme gibi düşünce görevini yerine getirmeye muktedir olmakla ilgilidir [4]. Yapay zekânın bir alt dalı olan makine öğrenmesi son yıllarda elektronik, bilgisayar, yazılım, biyomedikal, ekonometri, biyoistatistik gibi birçok alanda kullanılmaktadır. Makine öğrenmesi; ham veriden bilgi elde etmek ve algoritmalar kullanarak sistemleri modellemektir [5]. Tez konusu olan derin öğrenme ise makine öğrenmesinin alt bir dalıdır. Şekil 1.1.'de derin öğrenme makine öğrenmesi, yapay zekâ arasındaki ilişki gösterilmiştir.



Şekil 1.1. Derin öğrenmenin yapay zekâ içerisindeki yeri

Makine öğrenme teknikleri ile bir model oluşturmak için öncelikle özellik vektörünün oluşturulması gerekmektedir. Özellik vektörünün oluşturulması için alanında uzman kişilere gerek duyulmaktadır. Bu işlemler oldukça fazla sürede yapıldıklarından süre kaybı yaşamakta ve bu konuda uzman kişilerin çalışma zamanını almaktadır. Makine öğrenme teknikleri, işlenmemiş bir bilgiyi, uzman yardımı olmadan ve ön işlem yapmadan işleyemezler. Buna karşılık derin öğrenme, makine öğrenimi alanındaki araştırmacıların yıllarca uğraştığı bu problemi yok ederek büyük ilerleme sağlamıştır. Çünkü derin yapay sinir ağları geleneksel makine öğrenmesi yöntemlerinin aksine, öğrenme işlemini işlenmemiş veri üzerinde yapmaktadır. Bunun için ham veriyi işlerken lazım olan bilgiyi farklı katmanlarda oluşturduğu temsillerle elde etmektedir. Derin Öğrenme 2012 yılında nesne sınıflandırma alanında yapılan, büyük ölçekli görsel tanıma (ImageNet) yarışmasında elde ettiği başarı ile dikkat çekmiştir. Derin Öğrenmenin temelleri geçmiş yıllara dayansa da özellikle son zamanlarda ilgi çeken bir çalışma alanı olmasının en

önemli sebeplerinden birincisi eğitim kümesi olabilecek kadar çok verinin olması ve ikinci olarak bu verileri işleyebilecek donanımların geliştirilmiş olmasıdır [6].

Yapay sinir ağlarında işlenecek veri miktarının artmasıyla birlikte, yapay sinir ağları mimarisinde gizli katman ve düğüm sayısı da artmıştır. Bununla birlikte donanım yetersizliği nedeniyle yapay sinir ağlarının kullanımı duraksamıştır. Ancak GPU mimarisi ve diğer donanımsal gelişmeler sayesinde yapay sinir ağları mimarisinde çok sayıda (derin) gizli katmanlar kullanılmaya başlanmıştır. GPU mimarisindeki bellek bant genişliklerinin artması ile büyük miktardaki verilerin analiz edilmesi, matris işlemlerinin hızlı gerçekleştirilmesi CPU'ya göre düşük maliyetle gerçekleştirilmiştir.

Bu tez çalışması son yıllarda mühendislik alanında oldukça popüler olan derin öğrenme algoritmalarının Biyoistatistik ve Tıp Bilişiminde kullanımını göstermeyi ve derin öğrenmenin klasik yöntemlere göre üstünlüğünü göstermeyi hedeflemektedir.

Diyabet hastalığı pankreasın vücudun ihtiyaç duyduğu kadar insülin üretememesi veya vücudun ürettiği insülinin doğru bir şekilde kullanamaması sonucu oluşan kronik bir hastalıktır. Uzun sürede kalp damar hastalıkları, göz hastalıkları, böbrek hastalıkları gibi ciddi komplikasyonlar ortaya çıkarabilen diyabet, tedavi harcamalarının yüksekliği ve iş gücü kaybı nedeni ile hastaya sosyoekonomik yük getirmesinden dolayı önemli bir sağlık sorunudur [3]. Tanı için idrar ve kanda çeşitli kimyasal testler yapılmaktadır [7]. Bu çalışmada kişiye ait fiziksel özellikler ve kan testi verilerinin veri madenciliği yöntemleri ile kişilerin diyabet hastası olup olmadıklarının belirlenmesi gerçekleştirilmiştir. Yapılan çalışmada 768 kadına ait veriler derin yapay sinir ağları ve sınıflandırma algoritmaları ile sınıflandırılmış ve sonuçlar kıyaslanmıştır.

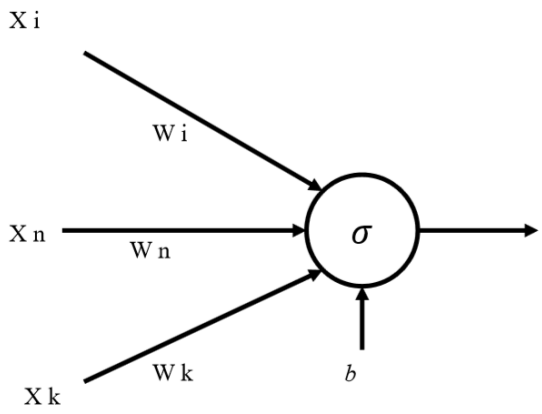
2. YAPAY SİNİR AĞLARI

Yapay sinir ağları; insan beyninin işleme yapısını modelleyerek geliştirilmiş bir yapay zekâ tekniğidir. Biyolojik sinir hücresi nöronlar sinir sistemini oluşturur. Yapay sinir ağları da nöron adı verilen birimlerden olur. Nöron sayısı, katman sayısı, katmanlar arası bağlantı türleri yapay sinir ağları mimarisini oluşturur.

2.1. Yapay Sinir Ağlarında Nöron

İnsan sinir sistemi hücresine nöron denir. İnsan beyni nöronların karmaşık bir ağ bağlantısından oluşur. Bu nöronlar beyinden vücuda bilgi iletimi görevi görürler. Bazı bilimsel çalışmalara göre insan beyni ortalama 100.000.000.000 nörona sahip olabilir [8]. Bu nöronların her biri diğerleri ile bağlantılı olup, her biri ortalama 6000 bağlantıya sahiptir. Bu bağlantılardan oluşan ağ çevremizdeki her şeyi algılamamızdan ve öğrenmemizden sorumludur.

Yapay sinir ağı nöronu girdi, girdilerin ağırlıkları, yanlılık (bias), aktivasyon fonksiyonu ve çıktıdan oluşur. Şekil 2.1. yapay sinir ağı nöronunu ve bağlantılarını göstermektedir.

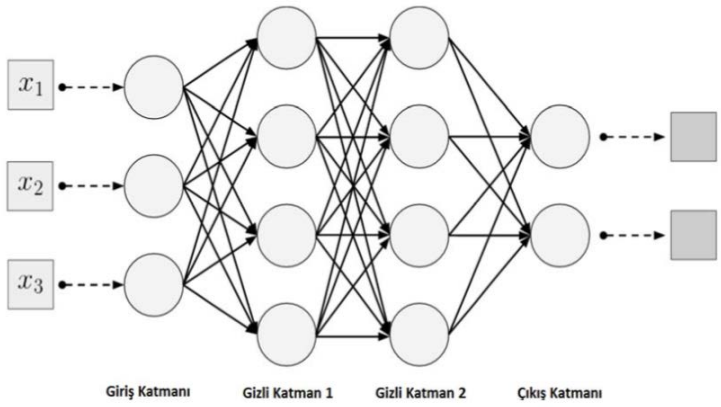


Şekil 2.1. Yapay Sinir Ağı Nöronu

Bir nöronun çıktısı girdi değerleri (X_1, X_2, …, X_n) ve ağırlıklarının (W_1,W_2,…,W_n) çarpımlarının toplamına bias (b) değerinin eklenmesi ile ($z = \sum_1^n \; X_i * W_i + b$)elde edilen değerin (z) aktivasyon fonksiyonundan α geçirilmesi ile elde edilir.

2.2. Çok Katmanlı Yapay Sinir Ağları

İnsan beyni katmanlı bir yapıya sahiptir. Duyulardan alınılan veri bir tabakadan diğerine aktarılarak bilgiye dönüşür. Örnek olarak, göz duyusundan görsel veri alınır. Bu veri görsel korteksin en alt katmanından en üst katmanına aktarılarak işlenir. Son katmanda görüntüde bir hangi nesne görüldüğüne karar verilir. İnsan sinir sistemindeki bu katmanlı yapı modellenerek yapay sinir ağı oluşturulmuş ve bu yapay sinir ağına Çok Katmanlı Yapay Sinir Ağı adı verilmiştir. Bu modeldeki en solda bulunan katmana giriş katmanı, giriş katmanındaki nöronlara ise giriş nöronları adı verilir. Ortadaki katmana gizli katman, sağdaki katmana ise çıkış katmanı adı verilir. Şekil 2.2.'de çok katmanlı yapay sinir ağı modellemesi mevcuttur.



Şekil 2.2. Çok katmanlı Yapay Sinir Ağı

2.3. **Yapay Sinir Ağlarında Öğrenme**

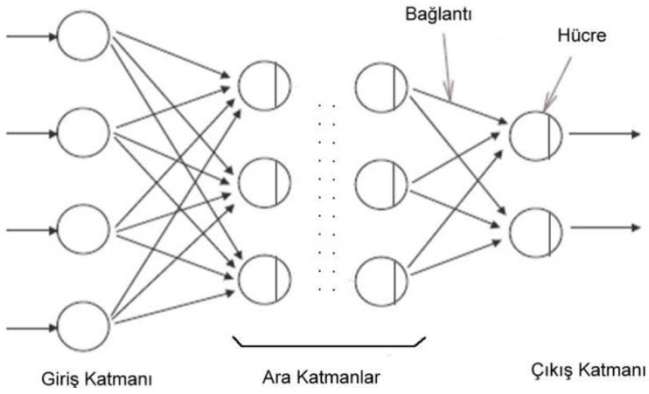
Yapay sinir ağının öğrenmesi, verilen giriş değerlerine karşılık istenilen çıkış değerlerini elde etmektir. Bu amaç ile ağın parametreleri (ağırlık değeri ve bias değeri) en uygun değere gelinceye kadar güncellenir. Bu işleme yapay sinir ağında öğrenme denir. Ağ parametrelerinin başlangıç değeri genellikle rastgele seçilir. Uygun değerlerin seçilmesi ağın daha hızlı öğrenmesini sağlar. Yapay sinir ağlarını bir örnek ile özetlemek gerekirse şu örnek verilebilir. Bir öğrenci sınava hazırlık amacı ile test sorusu çözsün. Öğrenci test kitabını istenilen düzeye gelinceye kadar, yaptığı yanlış soruların çözümünü inceleyerek, yanlış bilgilerini doğru bilgileri ile güncelleyerek ilerlesin. İstenilen seviyeye geldiğinde eğitim amaçlı çözdüğü sorular dışında aynı konudan başka soruları başarı ile çözebildiğinde öğrenci konuyu öğrenmiş sayılır. Yapay sinir ağlarında öğrenme de bu şekildedir. Eğitim veri seti ile ağ eğitilirken test veri seti ile ağ başarımı test edilir.

Eğitim setleri; girişlere göre çıkışların belli olduğu etiketli veri seti ve girişlere göre çıkışların belli olmadığı etiketsiz veri seti olmak üzere iki kısma ayrılır. Etiketli veri seti danışmalı öğrenmede kullanılırken, etiketsiz veri setleri danışmansız öğrenmede kullanılır.

Aşırı uyum (ezberleme) öğrenme sürecinde karşılaşılan problemlerdendir. Bu sorun genellikle eğitim veri kümesinin çok küçük olduğu durumlarda ortaya çıkmaktadır. Aşırı uyum sorunu ile başa çıkmak için etiketli veriler bir eğitim bir de doğrulama setine bölünürler. Veri kümesinin eğitim ve doğrulama verisi olarak ikiye bölünmesi işleminde genel eğilim mevcut verilerin %80'ni eğitim setine,%20'si doğrulama setine koymaktır[15].

2.4. İleri Beslemeli Yapay Sinir Ağları

İleri beslemeli yapay sinir ağları tek yönlü girdi sinyal akışına izin veren çok katmanlı yapay sinir ağı türüdür. Veriler her zaman ön katmana doğru iletilirler. Genellikle ileri beslemeli yapay sinir ağlarında 3 katman bulunur. Veriler giriş katmanından gizli katmana, gizli katmandan çıkış katmanına doğru ilerler. Giriş katmanı sadece verileri alır. Alınan veriler gizli katman ve çıkış katmanında işlenerek bilgi haline dönüşür. Yapay sinir ağına hem eğitim için ayrılmış eğitim kümesi giriş olarak hem de eğitim kümesinin hedef kümesi beklenen çıkış olarak verilmelidir. Yapay sinir ağı modeli kendisine verilen eğitim kümesinden (örnek girişler ve çıkışlardan) problem uzayını temsil eden bir çözüm uzayı oluşturur. Daha sonra, gösterilen bu çözüm uzayını kullanarak yeni girişlere karşılık yeni çıkışlar üretmektedir [9] . Şekil 2.3.'de ileri beslemeli yapay sinir ağı çizilerek gösterilmiştir.



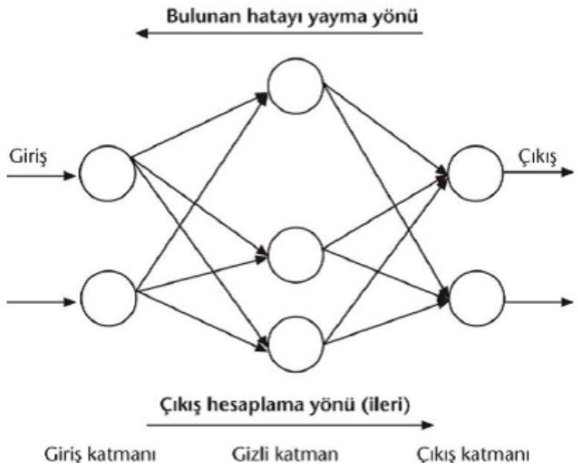
Şekil 2.3. İleri Beslemeli Yapay Sinir Ağı (Öztemel, 2006)

2.5. Geri Beslemeli Yapay Sinir Ağları

Geri beslemeli yapay sinir ağlarında en az bir hücrenin çıkışı kendisine veya başka bir hücreye giriş olarak verilir. Geri besleme aynı katmandaki hücreler arasında olabileceği gibi farklı katmandaki hücreler arasında da olabilir. Bu yapısıyla statik davranış gösteren ileri beslemeli yapay sinir ağlarının aksine dinamik bir yapı gösterir.

Geri beslemeli yapay sinir ağlarında eğitim etiketli veriler üzerinde yapılmaktadır. Eğitim etiketli veriler üzerinde yapıldığından doğruluğunun denetimi mümkündür. Bu denetim etiket verisi ile ağın çıktı sonucunun arasındaki farka yani hata değerine göre yapılır. Hata değerine göre ağırlıklar güncellenir. Ağırlık değerlerinin güncellenerek en uygun değere gelmesine *öğrenme* veya *eğitim* denir.

Yapay sinir ağlarında öğrenme dönemler halinde gerçekleşir. Dönem; eğitim verisi ce veriye bağlı çıktının bir kere işlenmesidir. Dönemin süresi ise eğitim verisinin boyutuna ve ağ mimarisine bağlıdır. Dönem sayısı ağı tasarlayan kişinin belirlediği kabul edilebilecek hata oranına göre değişir. Her dönemde ağın ürettiği çıktı, değeri beklenen çıktı değeri olan etiket verisi ile kıyaslanır. Arada oluşan fark hata değeridir ve bu değerler toplanarak toplam hata bulunur. Bu iki değer arasındaki fark negatif değer olabileceğinden tüm hata değerlerinin kareleri toplanır ve toplamın karekökü alınır. Hata geriye yayılım yöntemi ile tüm ağırlıklara yansıtılır. Hata değerinin kabul edilebilecek bir düzeye geldiği dönemde eğitim sonlandırılır. Şekil 2.4.'de geri beslemeli yapay sinir ağı çizilerek gösterilmiştir.



Şekil 2.4. Geri Beslemeli Yapay Sinir Ağı (Kabalcı,2016)

2.6. Hata Ölçüm Fonksiyonları

Yapay sinir ağlarındaki hata, hata ölçüm fonksiyonları ile bulunur. Hata, yukarıda da bahsedildiği gibi, eğitim sürecinde her dönem de oluşan hataların tümünü temsil eder. Eğitimin amacı, hatayı kabul edilebilir bir minimum seviyeye çekmektir. Bu işlemi ağırlıkları değerlerinin uygun değere getirerek gerçekleştir. Aşağıda alt başlıklar halinde yapay sinir ağlarında en çok kullanılan hata fonksiyonları açıklanmıştır.

2.6.1. Ortalama Karesel Hata

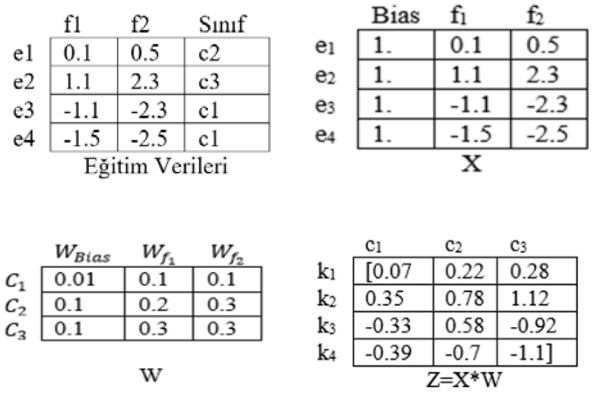
Yapay sinir ağlarında çıkış katmanında hatayı hesaplamak için kullanılır. Ortalama karesel hata eşitliği eşitlik 2.1 de gösterilmiştir;

$$C(W,b) \equiv \frac{1}{2n} \sum_x \quad \|y(x) - a\|^2 \qquad \textbf{Eşitlik 2.1.}$$

Denklem 2.1'de W; ağdaki tüm ağırlıkların toplamını, b; ağdaki tüm bias değerlerinin toplamını, n; ağın girdi sayısını, x; ağ girdisini, a; çıktı vektörünü temsil eder.

2.6.2. Log-Likelihood Hata Fonksiyonu

Log-Likelihood hata fonksiyonu; genellikle derin sinir ağlarında, softmax sınıflandırıcılarda kullanılmaktadır. Softmax sınıflandırıcıyı anlatabilmek için somut bir örnek verelim. 2 özniteliği (f1,f2), 4 örneği (e1,e2,e3,e4) olan bir veri seti düşünelim ve özniteliklerine göre 3 sınıfa (C1,C2,C3) ayıralım. Bu örnek için 4x(2+1) boyutunda X isimli bir giriş vektörü oluşturalım. Gaus dağılımına uygun rastgele değerler alan W isimli (2+1)x3bir ağırlık matrisi alalım ve bu vektörleri çarparak Z isimli vektörü elde edelim. Vektörler Şekil 2.5.'de gösterilmiştir.



Şekil 2.5. Likelihood Hata Fonksiyonu

Z vektöründeki değerler eşitlik 2.2'ye göre hesaplanırsa softmax sınıflandırıcının çıkışı olan S vektörü elde edilir. Bu matristeki değerler olasılık değerler olup bir satırın toplamı 1'dir. Softmax çıkış vektörü Şekil 2.6.'da gösterilmiştir.

$$P(y = j|(z^{(i)}) = \phi_{softmax}\ (z^i) = \frac{e^{z^{(i)}}}{\sum_{j=0}^{k}\ e^{z^{(i)}}}$$

Eşitlik 2.2.

$$C \equiv -ln\ \ \phi_{softmax}\ (z^i)$$ Eşitlik 2.3.

softmax

	c1	c2	c3
e1	0.29450637	0.34216758	0.36332605
e2	0.21290077	0.32728332	0.45981591
e3	0.42860913	0.33380113	0.23758974
e4	0.44941979	0.32962558	0.22095463

Şekil 2.6. Softmax çıkış vektörü

Şekil 2.6.'daki softmax çıkış vektörüne göre e_1 giriş durumunda çıkış sınıfı %29.45 c_1 sınıfına, %34 ihtimalle c_2 sınıfına, %36.33 ihtimalle c_3 sınıfına aittir. Sınıflandırma doğru değilse ağırlık değerleri optimizasyon teknikleri ile tekrarlamalarla (iterasyonlarla) güncellenerek sonuç değerlerine ulaşacaktır.

2.6.3. Ağırlıklara İlk Değeri Atama

Yapay sinir ağlarında ağırlıklara ilk değer atama önemli bir yer tutar. İlk değerin doğru bir şekilde atanması yakınsamayı hızlandırır ve yerel bir minimuma sıkışmayı engellemeye yardımcı olur.

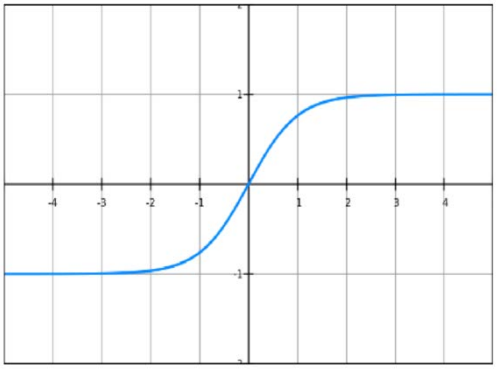
Gauss yöntemi sinir ağlarında kullanılan yaygın bir ağırlıklara ilk değer atama yöntemidir. Sıfır ortalama ve küçük varyansa sahip bir Gauss dağılımından verileri ağırlıklara ilk değer olarak atar. Glorot ve Ortogonal yöntemleri de sıkılıkla kullanılan ağırlıklara ilk değer atama yöntemlerindendir.

2.7. Aktivasyon Yöntemleri

Aktivasyon fonksiyonları; bir katmanın nöronlarının çıkışını başka bir katmana iletirken kullanılır. Yapay sinir ağları birçok ağırlık ve bias değerlerinin defalarca çarpılıp toplanması hesaplamalarını yapar. Ortaya çıkan büyük rakamları küçülterek işlem kolaylığı sağlanabilir. Bu işlem kolaylıklarının biri de aktivasyon fonksiyonu kullanarak büyük rakamları 0 ile 1 arasında bir değer ile temsil etmektir. Aşağıda yapay sinir ağlarında ve derin yapay sinir ağlarında sıklıkla kullanılan aktivasyon fonksiyonları verilmiştir.

2.7.1. Tanh Aktivasyon Fonksiyonu

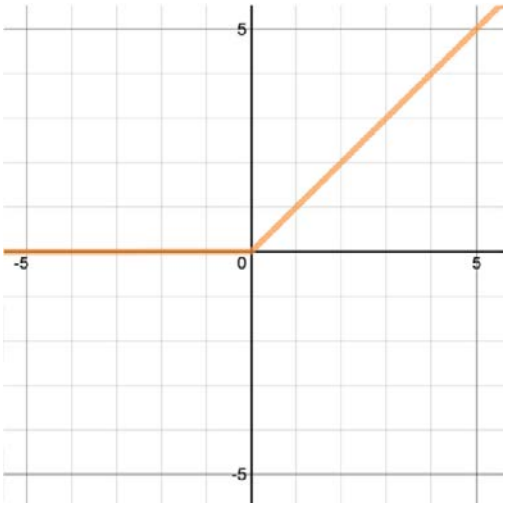
Tanh hiperbolik trigonometrik fonksiyondur. tanh'nın normaliz edilmiş aralığı -1 ile 1 arasındadır. Tanh'ın kullanılmasındaki asıl avantaj negatif sayılarla daha kolay anlaşabilmesidir. Şekil 2.7.'de Tanh fonksiyonu gösterilmiştir.



Şekil 2.7. Tanh aktivasyon fonksiyonu

2.7.2. Rectified linear aktivasyon fonksiyonu (ReLU)

Bu fonksiyon, giriş değeri sadece belirli bir miktarın üzerinde olduğunda aktif olur. Girdi sıfırın altında iken, çıkış sıfırdır, ancak girdi belirli bir miktarın üzerinde olursa, bağımlı değişkenle doğrusal bir ilişkisi olur. Şekil 2.8.'de Relu Fonsiyonu gösterilmiştir.



Şekil 2.8. Relu Hata Fonksiyonu

2.8. Optimizasyon Yöntemleri

Yapay sinir ağlarında ağırlıklar güncellenirken optimizasyon yöntemlerinden faydalanılır. Yapay sinir ağlarında bir çok optimizasyon yöntemi kullanılır. Fakat bunların bir çoğunun temelini Skolastik Gradyan İnişi Optimizasyon Yöntemi oluşturur. Aşağıda bu optimizasyon yöntemelerinin bazıları anlatılmıştır.

2.8.1. Skolastik Gradyan İnişi Optimizasyon Yöntemi

Ağırlıkları güncellerken tüm eğitim verisi yerine daha hızlı sonuç üretebilmek adına eğitim veri setinin bir kısmını kullanır [10]. Küçük-toplu işler yaygın olarak uygulanır, çünkü eğitim seti varyansını düşürür ve bu nedenle daha istikrarlı bir yakınsamaya sahiptir [10].

2.8.2. ADAM Optimizasyon Yöntemi

ADAM, KINGMA ve BA tarafından geliştirilen birinci derece gradyan tabanlı bir yöntemdir [10]. RMSProp optimizasyon yöntemi ve AdaGrad optimizasyon yöntemi iki optimizasyon yönteminden gelen fikirlerden ilham almaktadır. Pek çok problemde AdaGrad, RMSProp, SGİ ve NHG gibi diğer yöntemlerden daha üstün bir performans gösterdiği ortaya konmuştur [10]. Genel olarak ADAM Optimizasyon Yöntemi hızlıdır ve çok katı bir öğrenme programına ihtiyaç duymadığı için iyi bir seçimdir.

3. DERİN ÖĞRENME

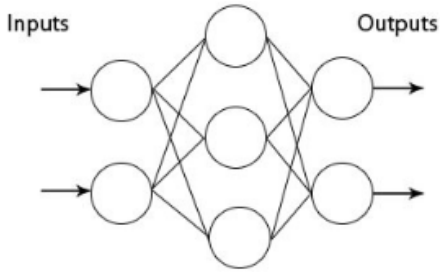
Derin öğrenme makine öğrenmesinin bir alt kümesidir. Derin öğrenmede özellik çıkarımı yapmak ve doğrusal yapıyı doğrusal olmayan yapı haline getirmek için bir çok doğrusal olmayan (aktivasyon fonksiyonları) katmanı kullanılır. Ardışık katmanlar bir önceki katmanın çıktısını girdi olarak alırlar. Derin öğrenmede, verilerin birden fazla özellik seviyesinin veya temsillerinin öğrenilmesine dayanan bir yapı söz konusudur. Hiyerarşik bir şekilde verinin üst düzey ve alt düzey temsilleri oluşturulur. Bu temsiller, soyutlamanın farklı seviyelerine karşılık gelen birden çok temsil seviyesini öğrenmektedir [11,12].

3.1. Derin Sinir Ağları (Deep Neural Network – DNN) Mimarileri

Derin öğrenme farklı problemlerin çözümü için oluşturulmuş farklı mimarilere sahiptir. Bu mimar aşağıda açıklanmıştır.

3.1.1. Tam Bağlantılı Yapay Sinir Ağları (Dense Neural Network)

Tam bağlantılı yapay sinir ağlarında, bir katmandaki her bir nöron önceki ve sonraki katmanlardaki tüm nöronlara bağlıdır. Tam bağlantılı ağlar genellikle sınıflandırma problemini çözmek için kullanılırlar. Derin yapay sinir ağlarının diğer mimarilerinin de içerisinde bir bölümü oluşturlar. Derin ağlar giriş ve çıkış katmanın dışında en az iki gizli katmandan oluşur. Genellikle matrisleri vektörlere çevirme de kullanılır. Geri ve ileri yayılım kullandığı için çok miktarda hesaplama gerektirir. Şekil 3.1.'de tam bağlı katman çizilerek gösterilmiştir.



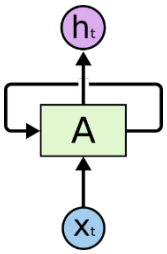
Şekil 3.1. Tam Bağlı Katman

3.1.2. Tekrarlayan Sinir Ağları (Recurrent Neural Network –RNN)

İnsanlar düşüncelerini her saniye sıfırdan başlatmazlar. Bu tezi okuduğunuzda, her sözcüğü, önceki sözcükleri anlamanıza dayanarak anlıyorsunuzdur. Her kelimede bir öncekini bir kenara atıp yeniden düşünmeye başlamıyorsunuzdur. Düşüncelerinizin bir kalıcılığı vardır.

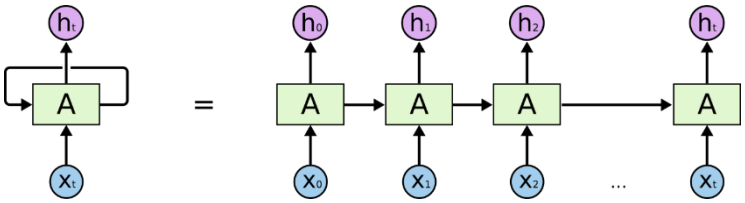
Geleneksel yapay sinir ağları bunu yapmaz ve bu durum büyük bir eksikliktir. Örneğin, bir video görüntüsünde her noktada ne tür bir olayın olduğunu sınıflandırmak istediğinizi düşünün, geleneksel bir sinir ağı, filmdeki önceki olayların sonucunu daha sonra oluşan olaylara sebep olarak nasıl bildirebildiği belli değildir.

Tekrarlayan sinir ağları bu sorunu ele almaktadır. Tekrarlayan sinir ağları, içinde döngüler bulunan ve bilginin devam etmesine izin veren ağlardır.



Şekil 3.2. Tekrarlayan Sinir ağları döngülere sahiptir

Yukarıdaki diyagramda, bir sinir ağı yığını A, bazı girdilere (x_t) bakar ve bir çıkış (h_t) değeri verir. Bir döngü bilginin yapay sinir ağının bir basamağından diğer basamağına geçmesine izin verir. Bu yapının normal bir yapay sinir ağından farklı olmadığı görülecektir. Tekrarlayan bir sinir ağı aynı ağın birbirlerine mesaj yollayan, tekrarlayan bir kopyası olarak düşünülebilir



Şekil 3.3. Döngü açıldığında elde edilen görünüm.

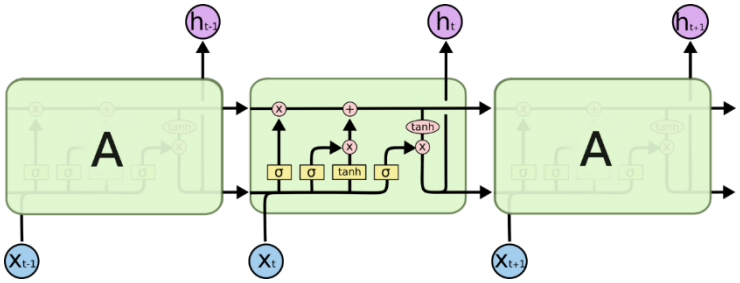
Bu zincir benzeri yapı RNN'lerin dizi ve liste veri yapıları ile yakından ilişkili olduğunu gösterir. Bunlar benzer verilerin kullanımı için sinir ağlarının doğal bir mimarisidir. RNN'ler konuşma tanımlama, dil modelleme, çeviri gibi birçok uygulamada başarılı sonuçlar vermişlerdir. Bu başarıların esası, birçok görev için standart sürümden çok daha iyi çalışan, çok özel bir RNN türü olan LSTM'lerin kullanılmasıdır.

3.1.2.1. Uzun Kısa Vadeli Hafıza Ağları (Long Short Term Memory Networks – LSTM)

RNN'lerin avantajlarından biri mevcut bilgileri önceki bilgilere bağlıyor olabilmeleridir [13] .Bu gibi durumlarda, ilgili bilginin mevcut yeri ile ihtiyaç duyulan yer arasındaki zaman farkının az olduğu durumlarda, RNN'ler geçmiş bilgileri kullanmayı öğrenebilir. Çalışmanın konusu olan çoklu nesne takibi böyle

bir duruma örnek olarak verilebilir. Nesnelerin ardışık çerçevelerdeki konum bilgileri zaman farkının az olduğu bilgilerdir.

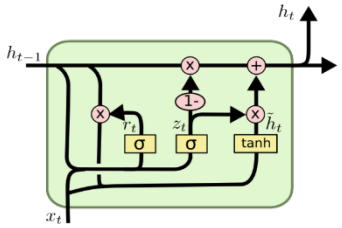
Uzun kısa vadeli hafıza ağları (Long Short Time Memory -LSTM) veriler arasındaki uzun vadeli ilişkileri (dizi, tekrar eden değerler, v.b.) öğrenebilen özel bir RNN türüdür. Yinelenen sinir ağları (RNN'ler), sinir ağı modül zinciri biçimindedir. Standart RNN'lerde, bu yinelenen modül, tek bir tanh aktivasyon katmanı gibi çok basit bir yapıya sahiptir. LSTM'lerin de bu zincir benzeri yapıları vardır ancak yinelenen modül farklı bir yapıya sahiptir. Tek bir sinir ağı katmanı yerine çok özel bir şekilde iletişim kuran dört katman vardır.



Şekil 3.4. Bir LSTM'deki yinelenen modül, etkileşimli dört katman içerir.

Yukarıdaki diyagramda, her satır, bir düğümün çıktısından başka düğümün girişlerine kadar tüm vektörü taşır. Pembe daireler, vektör eklenmesi gibi noktasal işlemleri temsil ederken, sarı kutular yapay sinir ağları ile öğrenilir. Birleştirilen satırlar birleştirme hattı, çizgi çatal çizmesi, içeriğinin kopyalanması ve kopyaların farklı yerlere gideceği anlamına gelir.

LSTM'ler probleme göre yeniden dizayn edilebilirler.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Şekil 3.5. Probleme göre özelleştirilmiş bir LSTM türü

3.1.3. Evrişimsel Yapay Sinir Ağı (Convolutional Neural Network – CNN)

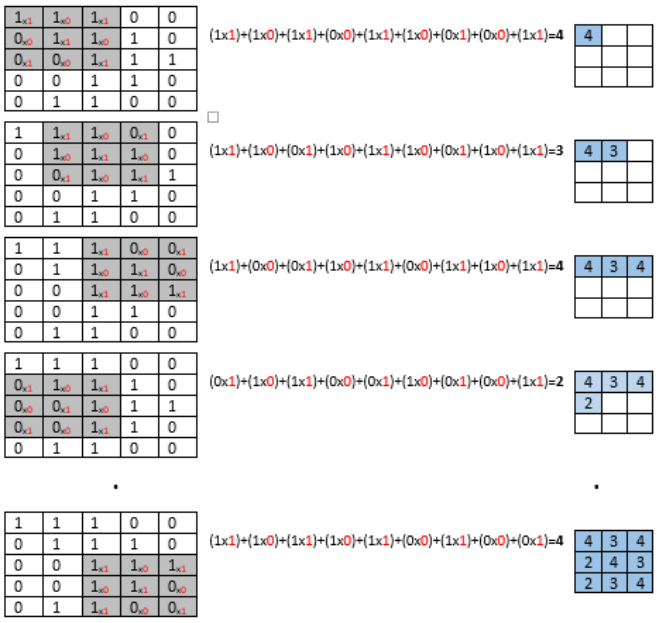
CNN, öğrenilebilir ağırlıkları ve önyargıları olan nöronlardan oluşan ileri beslemeli bir sinir ağı türüdür [14]. Evrişim ifadesi iki boyutlu giriş matrisi (çoğunlukla resim) üzerinde öz nitelik çıkarma amacı ile filtre gezdirme işlemidir. Şekil 3.6. da bu işlem gösterilmiştir.

CNN derin yapay siniri ağlarının en çok kullanılan mimarilerinden biridir. Aşağıda bu mimarinin bileşenleri ve parametreleri anlatılmıştır.

3.1.3.1. Evrişimsel Sinir Ağlarının Bileşenleri ve Parametreleri

3.1.3.1.1. Evrişim (Konvolüsyon) Katmanı

Evrişim katmanı ile öznitelikler çıkarılır ve çoğaltılır. Bu katman basit filtreleme işlemidir. Filtreler imge boyunca kaydırılır. Kaydırma sırasında filtrenin üzerinde bulunduğu imge piksel değerleri ile filtredeki değerler çarpılır ve elde edilen değerler toplanır ve net sonuç bulunur. Bu işlemi tüm görsele uyguladığımızda yeni bir imge elde etmiş oluruz. Elde edilen imgeye evrişim özniteliği (Convolved Feature) denir. Her bir filtre aslında yapay sinir ağlarına ait bir nörondur. Bu nöronun ağırlıkları ise filtre içerisindeki değerlerdir. Filtre boyutu büyüdükçe çıkış matrisi küçülecektir. Bu küçülme giriş matrisindeki bilgileri kaybetme anlamına gelmektedir. Dolayısıyla genellikle 3x3, 5x5, 7x7 boyutunda filitreler giriş matirisi üzerinde gezdirilmektedir. Şekil 3.6.'da 5x5 boyutundaki girişe 3x3 boyutundaki filtrenin gezdirimi ve 3x3 boyutunda evrişim matrisinin oluşumu gösterilmiştir.



Şekil 3.6. 5x5 boyutunda iki boyutlu girişe, 1x1 kaydırma(stride) ile 3x3'lük bir filtre uygulanarak Konvolüsyon Özniteliğinin Oluşması

Kaydırma işlemi uygulanırken filtre boyutunun dışında adım(stride) ve boşluk(padding) parametreleri de kullanılmaktadır. Adım parametresi, filtre uygulanırken filtrenin ne kadar aralıklarla kaydırılacağını belirtmektedir. Boşluk parametresi ise kenar değerlerinin işleme nasıl katılacağını ifade etmektedir.

Her bir filtre iki boyutlu giriş matrisinde ayrı bir özelliğe odaklanmaktadır. Filtreler sayesinde bir iki boyutlu giriş matrisinin çeşitli özelliklerine odaklanma sağlanmış birden farklı sürümü elde edilmektedir. Uygulanan filtreler ağırlıklardan oluşmaktadır. Ağırlıklar ise ağın eğitimi ile öğrenilmektedir. Ağda ne kadar fazla konvolüzasyon (evrişim) katmanı bulursa o kadar fazla karmaşık özellikler belirlenebilmektedir.

3.1.3.1.1.1. **Piksel Ekleme (Padding)**

Evrişim işleminden sonra giriş matrisinin boyutunda bir azalma olur. Örneğin; Şekil 3.6.'daki evrişim işleminde 5x5 giriş matrisine 3x3 bir ağırlık matrisi evrişim işlemi yapmıştır. Bu işlem sonucunda 3x3 'lün bir öz nitelik matrisi oluşmuştur. Giriş matrisinin boyutu azalmıştır. Giriş matrisinin boyutunun azalmasının istenmediği durumlarda matrisn eksik boyutlarına

değer atanır (0 atanırsa zero padding) bu işleme padding denir. Giriş matrisinin boyutu g x g, ağırlık matrisinin boyutu a x a olduğunda, çıkış matrisi (g-a+1)x(g-a+1)=(5-3+1)x(5-3+1)=3×3 şeklinde hesaplanır

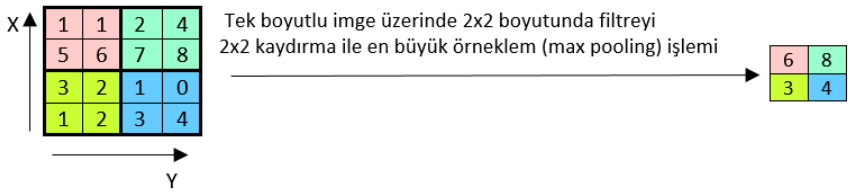
3.1.3.1.1.2. **Kaydırma Adımı (Stride)**

Stride parametresi evrişim işlemi evrişim işleminde ağırlıkarın oluşturduğu matris olan filitreyi (kernel, çekirdek) giriş matrisi üzerinde yatayda ve dikeyde kaç birim kaydıracağımızı belirler. Bu parametre padding ile birlikte çıkış öz nitelik matrisinin boyutunu belirler. Giriş matisinin boyutu gxg, ağırlıklar matrisinin boyutu a x a, padding değeri p, stride değeri s olsun ve g=5, a=3,p=1,s=2 olduğunda aşağıdaki formülden hesaplayarak çıkış değerini 3x3 olarak bulabiliriz.

$$Çıkış\ matrisi \quad boyutu \quad = \left[\frac{g+2*p-a}{s}+1\right] x \left[\frac{g+2*a-f}{s}+1\right]$$

3.1.3.1.2. Örneklem (Havuzlama – Pooling)

Evrişimsel Sinir Ağlarında örneklem katmanı evrişim katmandan sonra kullanılır. Amacı girdi olarak verilen imgenin alt örnekleme işlemini yapmaktır. Bu işlemi imge üzerinde gezen filtre boyutu içerinde imgenin en büyük değerini alarak (Max Pooling) yapar. Bu işlemle imgenin alt örneklemini elde edilerek aşırı uyum (overfittiting) probleminden kaçınılır, imgenin boyutu düşürülerek işlem hızı arttırılır. Örneklem işlemi en büyük değeri alarak (max pooling), en küçük değeri alarak (min pooling), değerlerin ortalamasını alarak (mean pooling) gerçekleştirilebilir. Şekil 3.7. de 'maxpooling' işlemi gösterilmiştir.

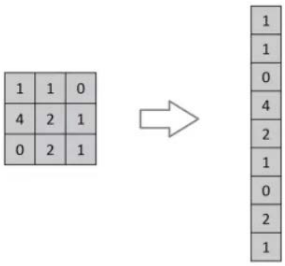


Şekil 3.7. En Büyük Örneklem (Max Pooling) İşlemi

3.1.3.1.3. Tam bağlı katman (Fully Connected Layer - Dense)

Sınıflandırma aşaması evrişimsel sinir ağlarında özellik çıkarımı aşamasından sonra yer almaktadır. Tam Bağlı Katman (Dense) özellik haritalarını girdi olarak

alır ve sınıflandırma işlemine hazırlar. Ağın çıkışında elde edilecek değerler sayı değerleri olacağı için çok boyutlu veriler (matrisler) ile işlem yapıldıktan sonra bu katmanda boyut indirgeme yapılarak öznitelik haritasından elde edilen veri tek boyutlu hale getirilir. Tek boyutlu veri sınıflandırıcıya girdi olarak verilir. Şekil 3.8. de tam bağlı katmana giren matrisin tek boyuta düşürülmesi gösterilmiştir.



Şekil 3.8. Tam bağlı katmanla tek boyuta düşürme işlemi (flatting)

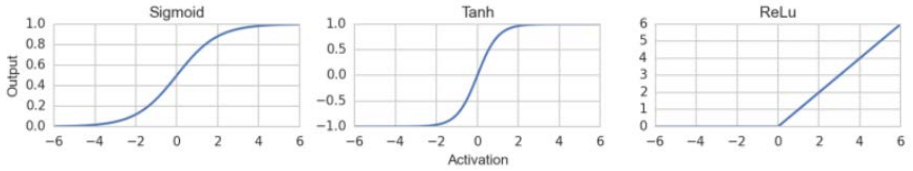
3.1.3.1.4. Sınıflandırma Katmanı

Öz nitelikler belirlendikten sonra probleme uygun bir sınıflandırıcı ile sınıflandırma işleminin yapıldığı katmandır. Softmax veya SVM sınıflandırıcılar derin öğrenme ile sıklıkla kullanılmaktadır.

3.1.3.1.5. Aktivasyon fonksiyonu

Yapay sinir ağları ve onun bir türü olan derin yapay sinir ağları giriş değerleri ve ağırlık değerlerinin çarpımlarına bias değerlerinin eklenmesi üzerine modellenmiştir. Bu modelleme lineer bir denklemdir. Lineer denklemlerin non-lineer hale dönüşümü aktivasyon fonksiyonları ile olur. Aktivasyon fonksiyonları modele non-lineerlik katarak doğrusal olmayan problemlerin çözümüne yardımcı olurlar. Geri beslemeli ağlarda, öğrenme farkı olan hata tüm düğümlere datıtılırken geri türev (gradient descent) hesaplanması yapılır. Türev işlemini kolayllaştırmak için de aktivasyon fonksiyonları kullanılır. Sürekli çarpılıp toplanarak büyüyen perception (yapay sinir ağı hücresi) çıktı değerleri değerleri hesaplama güçlüğü yaratmaması adına normalize edilmelidir. Aktivasyon fonksonları türüne göre verilen değeri 0 ile 1 arasına veya -1 ile 1 arasına bir değer ile temsil eder. Sigmoid, Tanh, ReLu en sık

kullanılan aktivasyon değerleridir. Relu ağın öğrenmesi açısından, Sigmoid'e göre daha hızlı bir aktivasyon fonsiyonudur.



Şekil 3.9. Bazı aktivasyon fonsiyonu grafikleri

3.1.3.1.6. Mini – Batch Boyutu (Mini – Batch Size)

Derin yapay sinir ağlarında öğrenme çok büyük veri setleri ile gerçekleştirilir. Bu veri setini bir anda işlenmesi çok uzun zaman alır ve bellek problemi yaşatır. Öğrenmenin her bir döneminde (epoch – döngü – iterasyon) ağırlıklar geriyeyayılım ("backpropagation") işlemi yapılır ve ağırlıklar gradient descent hesaplaması ile güncellenir. Bu hesaplama veri ne kadar fazla ise o kadar uzun olur. Bu sorunun üstesinden gelmek için veri küçük parçalara bölünür ve öğrenme işlemi bu küçük parçalardan gerçekleştirilir. Mini – Batch Boyutu veri setinden eğitim için alınan parçanın boyutunu belirtir. Bu parametre modelin aynı anda kaç veriyi işleyeceğini belirtir. Mini –Batch Size parametresi GPU belleğine sığabilecek boyutta 2 ve 2 nin katları şeklinde olması önerilmektedir[medium hiper]. 1 ile veri setinin boyutu arasında bir sayı olmalıdır. Çok küçük belirlenmesi ve Çok büyük belirlenmesi durumunda avantaj ve dezavantajlar vardır.

3.1.3.1.7. Öğrenme Hızı (Learning Rate)

Derin öğrenmede ağırlık ve bias değerlerinin güncellenmesi geriyeyayılım ("backpropagation") işlemi ile yapılmaktadır. Backpropagation işleminde bu güncelleme işi "chain rule" olarak adlandırılan geriye doğru türev (gradient descent) alarak farkın bulunması ve bulunan fark değerinin "learning rate" parametresiyle çarpılması, çıkan sonucun ağırlık değerlerinden çıkarılarak yeni ağırlık değerinin hesaplanmasıyla yapılmaktadır. Bu işlem esnasında kullanılan "learning rate" parametresi sabit değer olarak belirlenebilir, ya da adım adım artan bir değer olarak da belirlenebilir (örneğin belli bir öğrenme

adımına kadar 0.001 o adımdan sonra 0.01 gibi), momentum değerine bağlı olarak belirlenebilir ya da adaptif algoritmalar tarafından öğrenme esnasında öğrenilebilir. Bu parametre küçük seçilirse öğrene yavaş olacaktır. Başlarda yüksektutmak belirli bir epoch dan sonra düşürülmesi önerilmektedir.

3.1.3.1.8. Optimizasyon Algoritması Parametresi

Doğrusal olmayan problemlerin çözümünde en uygun (optimum) problemlerin çözümünde optimizasyon algoritmaları kullanılmaktadır. Derin yapay sinir ağları modellerinde yaygın olarak adam, stochastic gradient descent, adagrad, adadelta, adamax gibi optimizasyon algoritmaları kullanılmaktadır. Bu optimizasyon algoritmaları bir birlerine göre hız ve başarım farklılıkları vardır.

3.1.3.1.9. Eğitim Dönemi Sayısı (Epoch – İterasyon Number)

Derin yapay sinir ağlarında tüm eğitim verileri bir seferde eğitilmezler. Eğitim verileri parçalara ayrılır ilk parça eğitilir, hata değeri kullanılarak geri yayılım ile tüm ağırlıklar güncellenir. Daha sonra diğer eğitim verisi parçaları için de aynı işlemler gerekir. Bu işlem her defasında tekrarlanarak ağ ağırlıkları ve bias değerleri güncellenerek en uygun duruma gelir. Bu döngünün her adımında Eğitim Dönemi (Epoch) denir. Epoch sayısı arttıkça ağın başarımı artar. Başarım bellirli bir epoch değerinden sonra yavaş artış gösterir. Eğitim istenilen seviyeye geldiğinde sonlandırılabilir.

3.1.3.1.10. Seyreltme Değeri (Dropout Value)

Tam bağlı (Dense –Fulyy Connected) katmanlarda belli bir eşik değerinin altındaki düğümlerin seyreltilmesinin eğitim başarımını arttırdığı görülmüştür [26].

3.1.3.1.11. Katman Sayısı (Layer Number)

Derin öğrenmede katman sayısının artması öğrenme başarımını yükseltmetedir. Bunun yanında katman sayısı fazlalaştıkça geriyayılım ile ağırlıklar güncellenirken ilk katmanlara bu güncelemenin etkisi azalacaktır.

4. UYGULAMALAR VE SONUÇLAR

4.1. Çalışmada Kullanılan Veri Seti

Bu tez çalışmasında veri seti olarak UCI Machine Learning Repository'de ücretsiz ve kullanıma açık veri setlerinden "Pima Indians Diabets" veri seti kullanılmıştır. Bu veri seti National Institute of Diabetes and Digestive and Kidney Diseases tarafından 1990'da oluşturulmuştur. Veri setinde; gebelik sayısı, glikoz konsantrasyonu, diastolik kan basıncı, triseps kalınlığı, iki saatlik serum insulin değeri, vücut kitle indeksi, diyabet soyağacı fonksiyonu, yaş ve kadının şeker hastası olup olmadığını belirten hedef nitelik olmak üzere dokuz değişken bulunmaktadır. Şekil 4.1. de veri setinden ilk 6 hastanın verileri gösterilmiştir.

gebelikSayisi	glikozKonsantrasyon	diastolicKanBasinci	tricepsKalinligi	ikiSaatlikSerumInsulinDegeri	vucutKitleIndeksi	aileYakinlikDerecesi	yas	sinif
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1

Şekil 4.1. İlk 6 hastanın verileri

Veri setinin nitelikleri Tablo 4.1.'de, sınıflar Tablo 4.2. 'de, özet istatistiksel analizi Tablo 4.3. 'de, Tablo 4.4.'de gösterilmiştir.

Tablo 4.1. Pima Indians Diabetes veri seti nitelik tablosu

Veri Seti	Nitelik Sayısı	Hasta Sayısı (Örneklem)	Kayıp Veri
Pima Indians Diabets	8	768	0

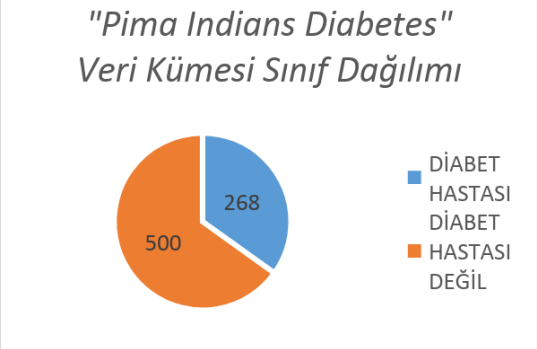
Tablo 4.2. Pima Indians Diabetes veri seti nitelik tablosu

Özellik Numarası	Veri Seti Nitelik Adı	Nitelik Tanımı
1	gebelikSayisi	Nümerik Değerler
2	glikozKonsantrasyonu	Oral Glikoz Tolerans Testinde 2. Saat Glikoz Konsantrasyonu (mg/dl)
3	diastolicKanBasinci	Diastolik Kan Basıncı (mm Hg)
4	trisepsKalinligi	Triceps Cilt Kalınlığı (mm)
5	ikiSaatlikSerumInsulinDegeri	İnsülin (mu U/Ml)
6	vucutKitleIndeksi	Ağırlık / $(\text{Uzunluk})^2$ kg/m^2
7	aileYakinlikDerecesi	Aile Yakınlık Derecesi
8	yas	Yıl
9	sinif	Diabet Test Sonucu Pozitif (1) veya Negatif (0)

Tablo 4.3. Pima Indians Diabetes veri setindeki sınıflar

"sinif" Değişkeni	Frekans (Veri Sayısı)	Yüzde (%)
0	500	65,1041
1	268	34,8959

TOPLAM	768	100,00



Şekil 4.2. Veri Kümesi Sınıf Dağılımı

Tablo 4.4. Veri tabanının özet istatistik analizi

Özellik Numarası	Ortalama	Standart Sapma	En Düşük Değer	En Yüksek Değer
1	3.8	3.4	0	17
2	120.9	32.0	0	119
3	69.1	19.4	0	122
4	20.5	16.0	0	99
5	79.8	115.2	0	846
6	32.0	7.9	0	67.1
7	0.5	0.3	0.0078	2.42
8	33.2	11.8	21	81

Çalışmada kullanılan veri seti olan “Pima Indians Diabet” data setinin istatistiksel özelliklerini Python programlama dili ile analiz edelim. Programımıza Microsoft içim Anaconda programını yükleyelim [08 -64] sonra Windows komut ekranından derin öğrenme kütüphaneleri için sırasıyla

- conda install -c conda-forge theano,

- conda install -c conda-forge tensorflow,
- conda install -c conda-forge keras

komutlarını çalıştıralım. Python spyder ortamından işlemlerimizi gerçekleştireceğiz.

4.1.1. Tanımlayıcı İstatistikler

Veri setine ait tanımlayıcı istatistikler aşağıdaki kodlar yardımı ile elde edilebilir.

Kodlar;

```
import pandas as pd
df = pd.read_csv("C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv",header=0)
print(df.describe())
```

Kodların verdiği çıktı aşağıdaki gibi olacaktır. Burada veri kümemizde her kolonun kaç satırdan oluştuğu (count), her kolonun ortalaması (mean), standart sapması (std), en küçük değeri(min),ilk çeyreklik (%25), ikinci çeyreklik (%50), üçüncü çeyreklik(%75), en büyük değeri (max) gibi tanımlayıcı istatistikleri elde edilmiştir. Çıktılar Tablo 4.5'de gösterilmiştir.

Tablo 4.5. Veri kümesine ait tanımlayıcı istatistikler

	GS	g.K	dKb	tK	iSSID	vKI	aYD	yas
count	768	768	768	768	768	768	768	768
mean	3.84	120.89	69.10	20.53	79.79	31.99	0.47	33.24
std	3.36	31.97	19.35	15.95	115.24	7.88	0.33	11.76
min	0	0	0	0	0	0	0.07	21
25%	1	99	62	0	0	27.3	0.24	24
50%	3	127	72	23	30.5	32.0	0.37	29
75%	6	140	80	32	127.25	36.6	0.62	41
max	17	199	122	99	846	67.1	2.42	81

4.1.2. Normallik Testi

Verinin normal dağılım gösterip göstermemesine göre uygulanılacak testler değiştiği için veriye normallik testi uygulanır. Aşağıdaki kodlar python programlama dilinde normallik testinin nasıl yapıldığını göstermektedir.

Kodlar:

```
 from scipy import stats
df = pd.read_csv("C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv",header=0)
a=df['sinif']
k,p=stats.mstats.normaltest(a)
if p<0.05:
 print ('veri normal dağılıma sahip değildir')
else:
 print ('veri Normal dağılmıştır')
```

Yapılan normallik testi sonucunda tüm değişkenler normal dağılıma sahip değildir. Verilerin normal dağılıma sahip ise Pearson korelasyon katsayısı, değil ise Spearman Rank korelasyon katsayısı kullanılır.

4.1.3. Korelasyon ilişkisi

Veri setine ilişkin pearson korelasyon ilişkisini hesaplatmak için aşağıdaki kodlardan faydalanılır.

Kodlar;

```
 from scipy.stats import pearsonr
import pandas as pd

def calculate_pvalues(df):
    df =  pd.read_csv("C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv",header=0)
    dfcols = pd.DataFrame(columns=df.columns)
    pvalues = dfcols.transpose().join(dfcols, how='outer')
    for r in df.columns:
        for c in df.columns:
            pvalues[r][c] = round(pearsonr(df[r], df[c])[1], 4)
    return pvalues
rho = df.corr(method='spearman') # r değerini hesapla
pval = calculate_pvalues(df) # p değerini hesapla
# maske yarat
r1 = rho.applymap(lambda x: '{:.2f}*'.format(x))
r2 = rho.applymap(lambda x: '{:.2f}**'.format(x))
r3 = rho.applymap(lambda x: '{:.2f}***'.format(x))
# p 0.10 dan küçükse *, 0.05 den küçükse **, 0.01 den küçükse ***
rho = rho.mask(pval<=0.10,r1)
rho = rho.mask(pval<=0.05,r2)
```

```
rho = rho.mask(pval<=0.01,r3)
print (rho)
```

Elde edilen sonuçlar Tablo 4.6.'da verilmiştir.

Tablo 4.6. Verisetine ait Pearson Korelasyon Analizi

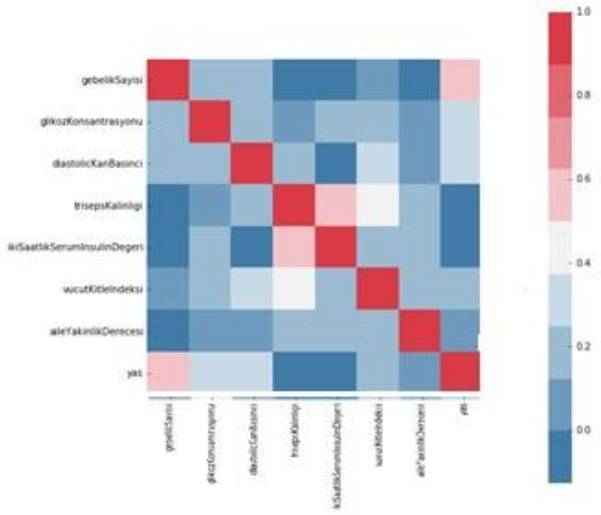
	GebSay	gli.Kon.	dKnbas	tKalın	ikiSSID	vKIndeks	aileYD	yas
GebSay	1.00***	0.13***	0.19***	-0.09**	-0.13**	0.00	-0.04	0.61***
gli.Kon.	0.13***	1.00***	0.24***	0.06	0.21***	0.23***	0.09***	0.29***
dKnbas	0.19***	0.24***	1.00***	0.13***	-0.01**	0.29***	0.03	0.35***
tKalın	-0.09**	0.06	0.13***	1.00***	0.54***	0.44***	0.18***	-0.07***
ikiSSID	-0.13**	0.21***	-0.01**	0.54***	1.00***	0.19***	0.22***	-0.11
vKIndeks	0.00	0.23***	0.29***	0.44***	0.19***	1.00***	0.14***	0.13
aileYD	-0.04	0.09***	0.03	0.18***	0.22***	0.14***	1.00***	0.04
yas	0.61***	0.29***	0.35***	-0.07***	-0.11	0.13	0.04	1.00***
sinif	0.20***	0.48***	0.14*	0.09**	0.07***	0.31***	0.18***	0.31***

Şimdi de korelasyon ilişkisini göstermek için ısı haritası çizdirelim. Bu amaç için aşağıdaki kodlardan faydalanırız.

Kodlar;

```
 import pandas as pd
df = pd.read_csv("C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv",header=0)
corr = df.corr(method = 'spearman') # Bağıntı Matresi
plt.subplots(figsize=(9, 9))
# Renk haritası oluşturmak
cmap = sns.diverging_palette(240, 10, n=9)
#Isı haritasını maskeyle çizimek
sns.heatmap(corr, cmap=cmap,square=True, xticklabels=True, yticklabels=True)
```

Kodların çıktısı ise Şekil 4.3.'deki gibi olacaktır.



Şekil 4.3. Korelasyon ilişkisini göstermek için ısı haritası

Tablodan ve şekillerden de anlaşıldığı gibi yaş ile gebelik sayısı arasında pozitif bir ilişki vardır. Glikoz konsantrasyonu ile sınıf arasında bir ilişki vardır. Triceps kalınlığı ile vücut kitle indeksi ve iki saatlik serum insulin değeri arasında bir ilişki vardır.

4.2. Sınıflandırma Algoritmalarını Problem Üzerinde Başarım Ölçütleri

Sınıflandırıcı algoritmaları kullanan modellere sınıflandırıcı denir. Sınıflandırıcı başarımı birçok farklı yöntemle ölçülebilir. Sınıflandırıcı birçok örneklemi doğru değerlendirirken, bazı örneklemleri doğruyken yanlış, bazı örneklemleri yanlışken doğru, işaretleyebilir. Bu 4 durumun oluşmasına sebep verir. Bu durumlar;

- Doğru Pozitifler (DP),(True Positive - TP); hasta tahmin etmiş iken hasta,
- Doğru Negatifler (DN), (True Negative –TN); hasta tahmin etmemişken hasta değil,
- Yanlış Pozitifler (YP),(False Positive – FP); Tip – 1 hatası olarak da bilinir. Hasta tahmin ettik ama hasta değil,
- Yanlış Negatifler (YN), (False Negative – FN); Tip – 2 hatası olarak da bilinir. Hasta olmadığını tahmin ettik ama hasta,

Bu 4 durum göz önüne alınarak, karmaşıklık matrisi ve değerlendirme skorları oluşmuştur.

4.2.1. Karmaşıklık Matrisi (Confusion Matrix)

DP, DN, YP, YN değerlerinin atandığı matristir. Tablo 4.2.1.1. ‘de bu değerlerin matrisin hangi indeksinde tutulduğu gösterilmiştir.

Tablo 4.7. Karmaşıklık Matrisi

	Tahmini Değeri Hayır	**Tahmini Değeri Evet**
Gerçek Değeri Hayır	Doğru Pozitif (DP)	Yanlış Pozitif (YP)
Gerçek Değeri Evet	Yanlış Negatif (YN)	Doğru Negatif (DN)

4.2.2. **Sınıflandırıcı Değerlendirme Raporunda Kullanılan Parametreler**

- *Doğruluk (Accuracy);* doğru tahmin edilen gözlemlerin, toplam gözleme bölünmesi ile elde edilen değerdir.

$$Doğruluk\ \ (Accuracy\ \ ) = \frac{DP+DN}{DP+DN+YP+YN}$$ Eşitlik 4.1.

- *Hassasiyet (Precission);* doğru tahmin edilmiş doğruların (DP), gerçekten sınıflandırılmış pozitiflerin tüm tahmin edilen pozitiflere oranıdır.

$$Hassasiyet\ \ \ (Precission\ \ \ ) = \frac{DP}{DP+YP}$$

Eşitlik 4.2.

- *Duyarlılık (Sensitivity - Recall – True Positive Rate –TPR);* doğru tahmin edilen pozitiflerin sayısının gerçek pozitiflerin toplamına oranıdır.

$$Duyarlılık\ (Sensitivity\ \ \ ) = \frac{DP}{YN+DP}$$

Eşitlik 4.3.

- *F1 Skoru;* Hassasiyet ve duyarlılığın ağırlıklı ortalamasından oluşur. Hem yanlış pozitif hem yanlış negatifleri hesaba katar.

$$f1\ skoru\ \ = \frac{2*(Duyarlılık*Hassasiyet\ \ \ )}{Duyarlılık+Hassasiyet}$$

Eşitlik 4.4.

4.3. Tam Bağlı Derin Yapay Sinir Ağları ile Sınıflandırma (Classification with Fully Connected Deep Neural Network)

Kodlar:

```
import pandas as pd    # pandas is a dataframe library
import matplotlib.pyplot as plt      # matplotlib.pyplot plots data
import numpy as np
import theano
import keras
import tensorflow
from IPython import get_ipython
get_ipython().run_line_magic('matplotlib', 'inline')
from sklearn import metrics
import matplotlib.pyplot as plt # side-stepping mpl backend
import matplotlib.gridspec as gridspec # subplots
import seaborn as sns # Danker visuals
import scipy
from pandas import DataFrame
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dropout, Flatten, Activation, Dense
from keras.layers.convolutional import Convolution2D, Convolution1D,MaxPooling1D
#Import models from scikit learn module:
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import KFold #For K-fold cross validation
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from keras.layers import Input, Dense
from keras.optimizers import SGD
from sklearn.preprocessing import Imputer
import pandas as pd
import matplotlib.pyplot as plt
#dosyayı oku
df = pd.read_csv("C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv",header=0)
#özellik kolon isimleri
feature_col_names =
['gebelikSayisi','glikozKonsantrasyonu','diastolicKanBasinci','trisepsKalinligi','ikiSaatlik
SerumInsulinDegeri','vucutKitleIndeksi','aileYakinlikDerecesi','yas']
#tahmin edilecek sınıf adı
predicted_class_names = ['sinif']
X = df[feature_col_names].values
y = df[predicted_class_names].values
split_test_size = 0.05
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split_test_size,
random_state=0)
# test_size = 0.3 is 30%, 42 is the answer to everything
print("{0:0.2f}% Eğitim setinde".format((float(len(X_train))/len(df.index)) * 100))
print("{0:0.2f}% Test setinde".format((float(len(X_test))/len(df.index)) * 100))
print("Eğitim setinde bulunan diabet : {0} ({1:0.2f}%)".format(len(y_train[y_train[:]
== 1]), (float(len(y_train[y_train[:] == 1]))/len(y_train) * 100.0)))
print("Eğitim setinde bulunan iyi nonDiabet: {0}
({1:0.2f}%)".format(len(y_train[y_train[:] == 0]), (float(len(y_train[y_train[:] ==
0]))/len(y_train) * 100.0)))
print("")
print("Test setinde bulunan diabet:{0}({1:0.2f}%)".format(len(y_test[y_test[:] == 1]),
(float(len(y_test[y_test[:] == 1]))/len(y_test) * 100.0)))
print("Test setinde bulunan nonDiabet:{0}({1:0.2f}%)".format(len(y_test[y_test[:]==
0]), (float(len(y_test[y_test[:] == 0]))/len(y_test) * 100.0)))
model = Sequential()
model.add(Dense(500, input_dim=8, init='uniform', activation='sigmoid')) # 1000
neurons
model.add(Dense(128, init='uniform', activation='tanh'))
model.add(Dense(8, init='uniform', activation='relu')) # 1 output neuron
model.add(Dense(1, init='uniform', activation='sigmoid')) # 1 output neuron
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, nb_epoch=1500, batch_size=3000,  verbose=2) # 150 epoch,
10 batch size, verbose = 2
predictions = model.predict(X_test)
rounded = [np.round(X_test) for X_test in predictions]
print("Accuracy:{0:4f}".format(metrics.accuracy_score(y_test, rounded)))
from sklearn.metrics import confusion_matrix,classification_report
predictions = model.predict_classes(X_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions)
```

Tablo 4.8. Tam Bağlı Yapay Sinir Ağları Modellemesinin Karmaşıklık Matrisi

	Tahmin Edilen Negatif	Tahmin Edilen Pozitif
Gerçek Negatif	DN=21	YP=7
Gerçek Pozitif	YN=0	DP=11

Tablo 4.9. Tam Bağlı Yapay Sinir Ağları Modellemesinin Sınıflandırma Raporu

Accuracy:	0.820513			
	precision	recall	f1-score	support
0	1.00	0.75	0.86	28
1	0.61	1.00	0.76	11
avg / total	0.89	0.82	0.83	39

4.4. **Naive – Bayes ile Sınıflandırma**

Kodlar:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv')

X = dataset.iloc[:,0:8].values
y = dataset.iloc[:, 8].values
```

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print (cm)
print (classification_report(y_test, y_pred ))
```

Tablo 4.10. Naive- Bayes Sınıflandırma Algoritmasının Karmaşıklık Matrisi

	Tahmin Edilen Negatif	Tahmin Edilen Pozitif
Gerçek Negatif	DN=114	YP=16
Gerçek Pozitif	YN=29	DP=33

Tablo 4.11. Naive- Bayes Sınıflandırma Algoritmasının Sınıflandırma Raporu

Accuracy:	0,77			
	precision	recall	f1-score	support
0	0.80	0.88	0.84	130
1	0.67	0.53	0.59	62
avg / total	0.76	0.77	0.76	192

4.5. K En Yakın Komşu Modeli (K-Neighbors)

Kodlar;

```
 import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv')

X = dataset.iloc[:,0:8].values
y = dataset.iloc[:, 8].values
```

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p = 2)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print (cm)
print (classification_report(y_test, y_pred ))
```

Tablo 4.11. K En Yakın Komşular Sınıflandırma Algoritmasının Karmaşıklık Matrisi

	Tahmin Edilen Negatif	Tahmin Edilen Pozitif
Gerçek Negatif	DN=114	YP=16
Gerçek Pozitif	YN=22	DP=40

Tablo 4.12. K En Yakın Komşular Sınıflandırma Algoritmasının Sınıflandırma Raporu

Accuracy:	%80,20			
	precision	recall	f1-score	support
0	0.84	0.88	0.86	130
1	0.71	0.65	0.68	62
avg / total	0.80	0.80	0.80	192

4.6. **Lojistik Regression**

Kodlar

```
 import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv')

X = dataset.iloc[:,0:8].values
y = dataset.iloc[:, 8].values
```

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print (cm)
print (classification_report(y_test, y_pred ))
```

Tablo 4.13. Lojistik Regresyon Sınıflandırma Algoritmasının Karmaşıklık Matrisi

	Tahmin Edilen Negatif	Tahmin Edilen Pozitif
Gerçek Negatif	DN=118	YP=12
Gerçek Pozitif	YN=26	DP=36

Tablo 4.14. Lojistik Regresyon Sınıflandırma Algoritmasının Sınıflandırma Raporu

Accuracy:	%80,20			
	precision	recall	f1-score	support
0	0.82	0.91	0.86	130
1	0.75	0.58	0.65	62
avg / total	0.80	0.80	0.79	192

4.7. Karar Ağacı (Decision Tree) ile Sınıflandırma

Kodlar

```
 import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv')

X = dataset.iloc[:,0:8].values
y = dataset.iloc[:, 8].values
```

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print (cm)
print (classification_report(y_test, y_pred ))
```

Tablo 4.15. Karar Ağacı Sınıflandırma Algoritmasının Karmaşıklık Matrisi

	Tahmin Edilen Negatif	Tahmin Edilen Pozitif
Gerçek Negatif	DN=102	YP=28
Gerçek Pozitif	YN=26	DP=36

Tablo 4.16. Karar Ağacı Sınıflandırma Algoritmasının Sınıflandırma Raporu

Accuracy:	%72			
	precision	recall	f1-score	support
0	0,80	0,78	0,79	130
1	0,56	0,58	0,57	62
avg / total	0,72	0,72	0,72	192

4.8. Rastgele Orman (Random Forest) Algoritması ile Sınıflandırma

Kodlar;

```
 import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv')

X = dataset.iloc[:,0:8].values
y = dataset.iloc[:, 8].values
```

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(random_state=0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print (cm)
```

Tablo 4.17. Rastgele Orman Sınıflandırma Algoritmasının Karmaşıklık Matrisi

	Tahmin Edilen Negatif	Tahmin Edilen Pozitif
Gerçek Negatif	DN=110	YP=20
Gerçek Pozitif	YN=33	DP=29

Tablo 4.18. Rastgele Orman Sınıflandırma Algoritmasının Sınıflandırma Raporu

Accuracy:	%72			
	precision	recall	f1-score	support
0	0,77	0,85	0,81	130
1	0,59	0,47	0,52	62
avg / total	0,71	0,72	0,71	192

4.9. **Destek Vektör Makinası (Support Vector Machine – SVM)**

Kodlar;

```
 import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('C:\\Users\\pc\\.spyder-py3\\pima-indians-diabetes.csv')

X = dataset.iloc[:,0:8].values
y = dataset.iloc[:, 8].values
```

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state = 0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print (cm)
```

Tablo 4.19. SVM Sınıflandırma Algoritmasının Karmaşıklık Matrisi

	Tahmin Edilen Negatif	Tahmin Edilen Pozitif
Gerçek Negatif	DN=117	YP=13
Gerçek Pozitif	YN=25	DP=37

Tablo 4.20. SVM Sınıflandırma Algoritmasının Sınıflandırma Raporu

Accuracy:	%80.20			
	precision	recall	f1-score	support
0	0.82	0.90	0.86	130
1	0.74	0.60	0.66	62
avg / total	0.80	0.80	0.80	192

BEŞİNCİ BÖLÜM

5. SONUÇLAR VE DEĞERLENDİRME

Sağlık alanındaki bilimsel çalışmalarda, sağlık hizmetlerinin tasarlanmasında, uygulamasında, sağlık politikalarının belirlenmesinde ve sonuçların değerlendirilmesinde Biyoistatistik ve Tıp Bilişimi önemli bir yer tutmaktadır. Araştırmacılar, karar vericiler, biyoistatistiğin sağladığı analiz ve çıkarım araçlarından faydalanmaktadırlar.

Bu tez çalışmasının amacı Biyoistatistik ve Tıp bilişimi alanında, derin öğrenme algoritmalarının, yeterli ve uygun veri olduğunda, sınıflandırma, regresyon, tahmin gibi çalışma alanlarında verimli bir şekilde kullanılabilirliğini göstermektir. Bu amaçla tez çalışması kapsamında sınıflandırma alanında derin yapay sinir ağları hakkında bilgi verilmiş, kodlarla uygulaması gösterilmiş ve tasarlanan derin yapay sinir ağının sınıflandırma konusunda klasik yöntemlerden üstünlüğü gösterilmiştir. Derin yapay sinir ağlarında eğitim verisinin miktarı ne kadar fazlalaşırsa, ağ o kadar doğru karar verecektir.

Bu tez çalışmasında veri seti olarak UCI Machine Learning Repository'de ücretsiz ve kullanıma açık veri setlerinden "Pima Indians Diabets" veri seti kullanılmıştır. Bu veri seti National Institute of Diabetes and Digestive and Kidney Diseases tarafından 1990'da oluşturulmuştur. Veri seti Bioistatistik ve Tıp Bilişimi Anabilim Dalı çalışmalarında kullanılabilir bir veri setidir. Bu veri setinde derin yapay sinir ağlarının performansı değerlendirilmiştir. Geliştirilen derin yapay sinir ağı modeli %89 duyarlılık, %82 hassasiyet ile naive bayes (%76 hassasiyet,%77 duyarlılık), K-neighbors (%80 hassasiyet,%80 duyarlılık), Lojistik Regresyon(%80 hassasiyet,%80 duyarlılık),Karar Ağaçları(%72 hassasiyet,%72 duyarlılık), random forest (%71 hassasiyet,%72 duyarlılık), SVM (%80 hassasiyet,%80 duyarlılık) sınıflandırma algoritmalarını geride bırakmıştır. Derin öğrenme ağları için küçük sayılabilecek bu veri setinde bile başarımın klasik yöntemlere göre yüksek olması derin öğrenme algoritmalarının üstünlüğünü göstermektedir. Derin öğrenme algoritmaları eğitim süresi klasik algoritmalara göre daha yüksektir. Derin öğrenme performansı birçok parametreye bağlıdır. Bu parametreler probleme özgü olup genellikle deneme yanılma yolu ile belirlenmektedir. Bu parametrelerden bazıları gizli katman sayısı, optimizasyon, hata fonksiyonu, aktivasyon fonksiyonudur. Bu fonksiyonların seçimi ağ başarımını değiştirmektedir.

Çalışmada veri setinin gerçek verilerden oluşmasından dolayı sınıflandırma başarılarının yeteri kadar yüksek olduğu söylenebilir. Yapılan korelasyon analizinde "Glukoz" değerinin diyabet için en önemli kriter olduğu görülmüştür.

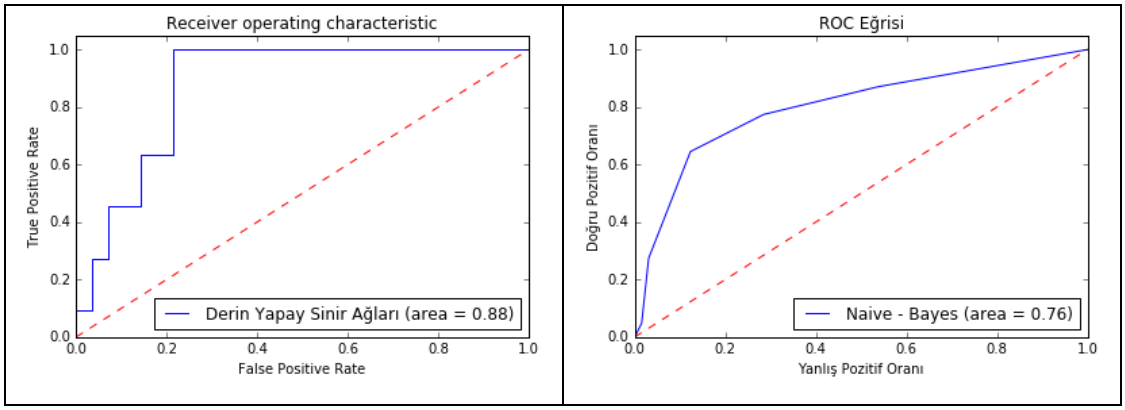
ROC (Alıcı işlem karakteristikleri, Receiver Operating Characteristic) tıbbi karar verme aşamasında, testin ayırt ediciliğini tespit etmek amacıyla kullanılır. ROC eğrisi sınıflandırıcı model seçiminde kullanılan bir eğridir. Eğrinin x eksenini, yanlış pozitif oran (False Positive Rate - FPR) oluşturur. FPR, gerçekte yok olan fakat var diye tahmin denilen hastaların gerçekte hasta olmayanlara oranıdır. Eşitlik de formülü verilmiştir.

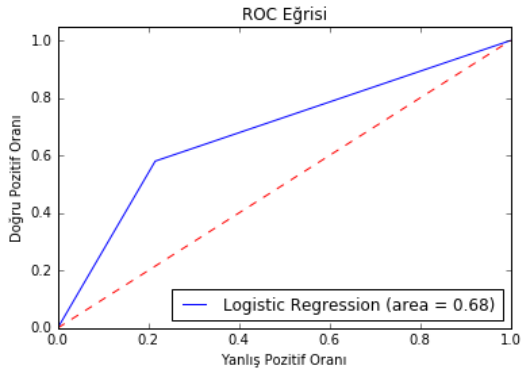
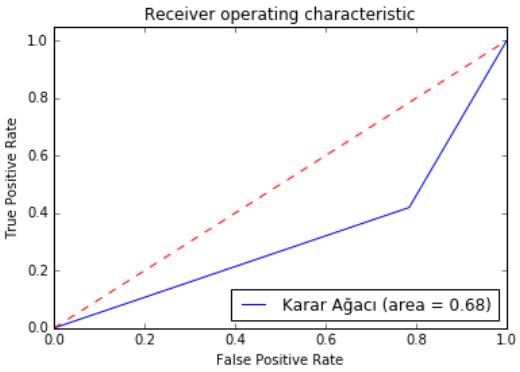
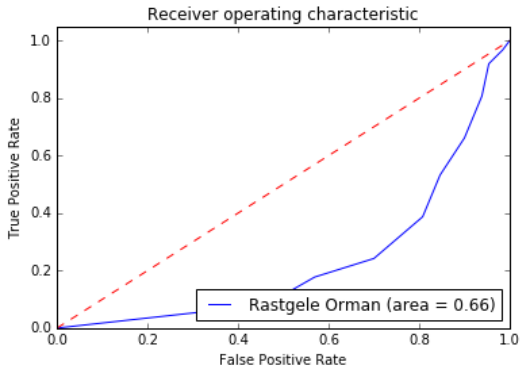
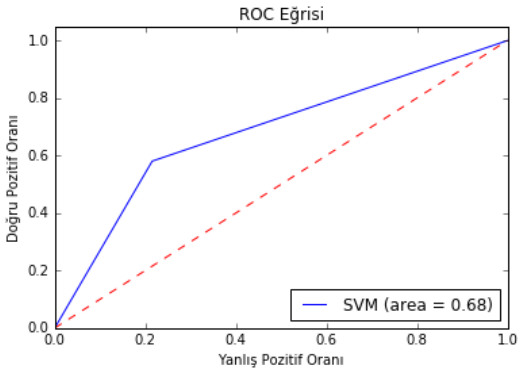
Eğrinin y eksenini ise duyarlılık vardır. Bir diğer ifade ile doğru pozitif oranı (True Positive Rate – TPR) oluşturur. TPR ise doğru tahmin edilen varların gerçek varlara oranıdır. Eşitlik de formülü verilmiştir. Bu iki değerin oluşturduğu eğrinin altında kalan alan (Area Under Curve – AUC) ne kadar büyük ise önerilen yöntem o kadar başarılıdır.

$$FPR = \frac{FP}{TN+FP}$$ Eşitlik 4.5.

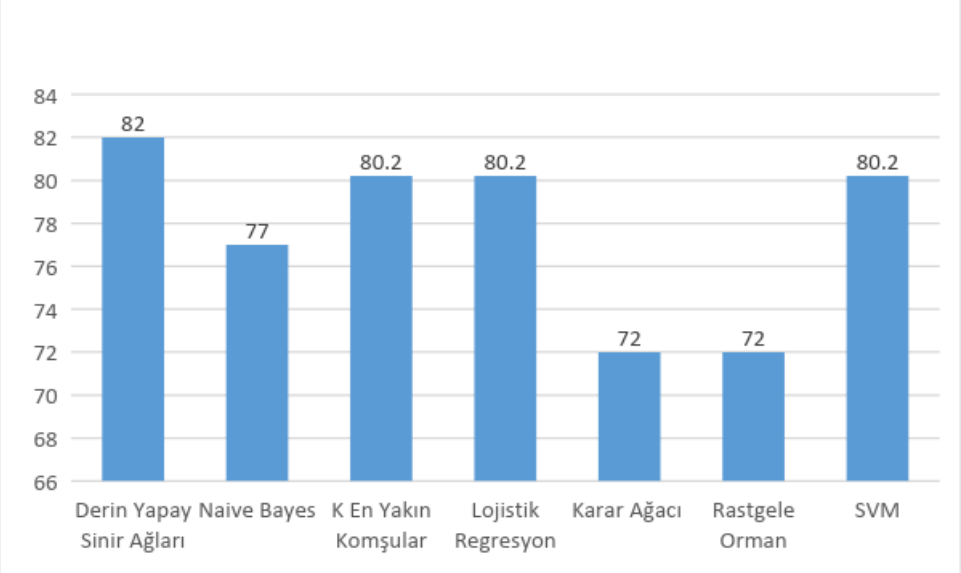
$$TPR = \frac{TP}{TP+FN}$$ Eşitlik 4.6.

Şekil 4.4.'de görüldüğü üzere Derin Yapay Sinir Ağları 0.88, Naive Bayes 0.76, Lojistik Regresyon 0.68, Karar Ağacı 0.68, Rasgele Orman 0.66, SVM 0.68 eğri altında kalan alan (AUC) değeri vermişlerdir. Bu değerler tez çalışmasında derin yapay sinir ağları ile gerçekleştirilen modelin başarımını ortaya koymaktadır. Test veri setine göre modellerin doğruluk (accuracy), hassasiyet (precission), duyarlılık (sensitivity), F1 skor değerleri grafikler halinde şekil 4.5., Şekil 4.6, Şekil 4.7 ve Şekil 4.8' de verilmiştir. Bu değerlendirme parametreleri de derin yapay sinir ağlarının diğer yöntemlerden başarılı olduğunu göstermiştir.

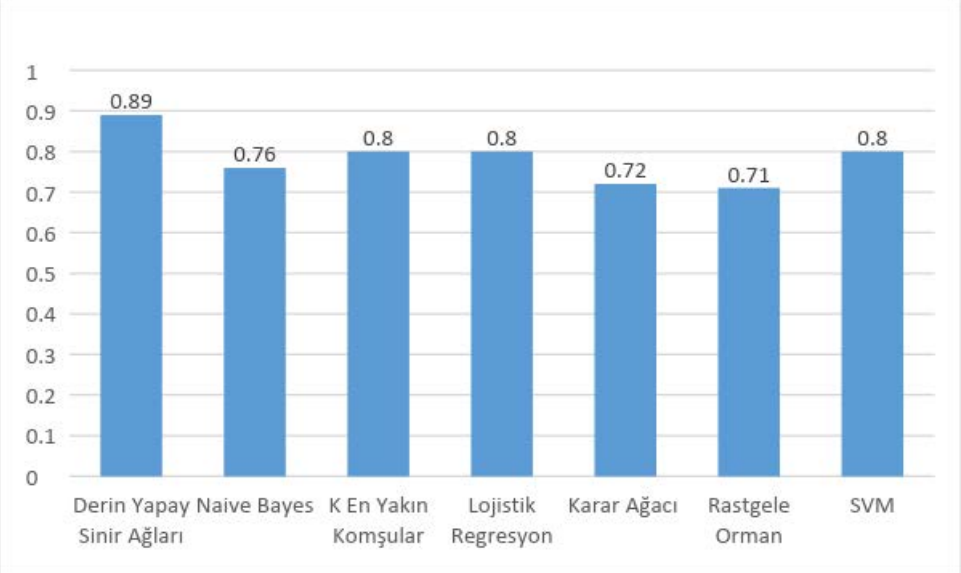


a- Derin yapay sinir ağları ROC eğrisi	b- Naive Bayes ROC eğrisi
	
c- Lojistik Regresyon ROC Eğrisi	d- Karar Ağacı ROC eğrisi
	
e- Rasgele Orman ROC eğrisi	f- SVM ROC eğrisi

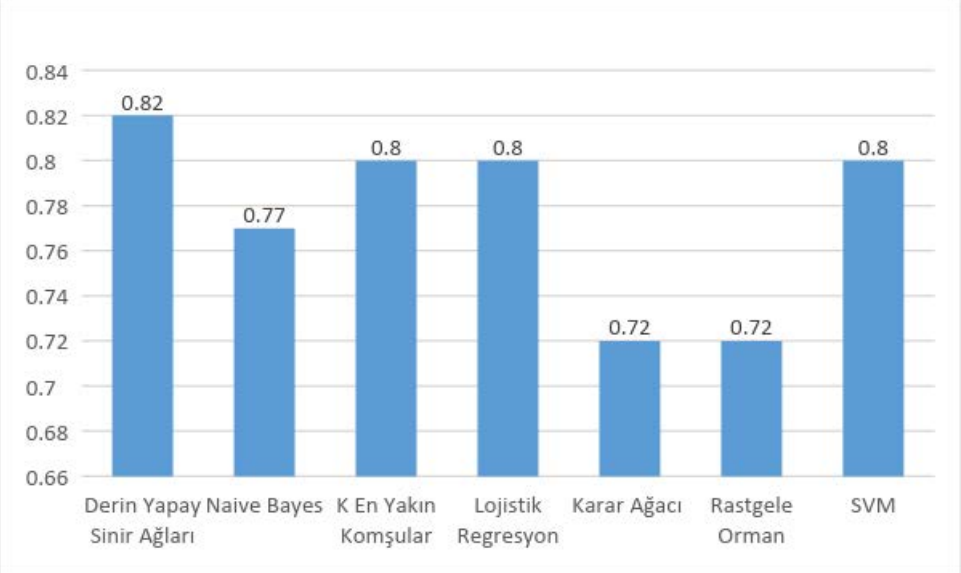
Şekil 4.4. ROC eğrileri



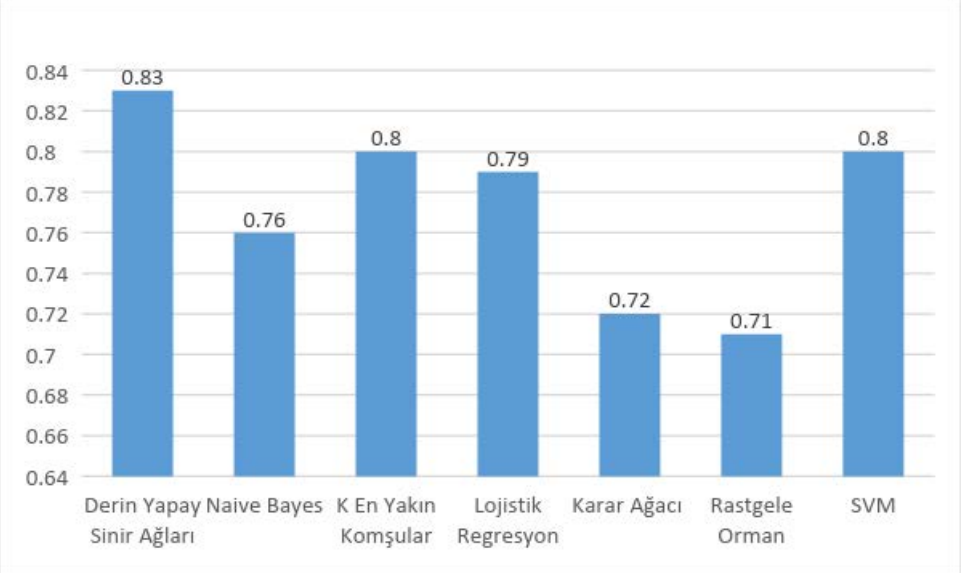
Şekil 4.5. Doğruluk ölçeğine göre yöntemlerin karşılaştırılması



Şekil 4.6. Hassasiyet ölçeğine göre yöntemlerin karşılaştırılması



Şekil 4.7. Duyarlılık ölçeğine göre yöntemlerin karşılaştırılması



Şekil 4.8. F1 skor ölçeğine göre yöntemlerin karşılaştırılması

KAYNAKÇA

1. Değişen Dünya Tıp ve Teknoloji. http://www.sisoft.com.tr/haber/page?SYF=Detay&hb=1197, Erişim Tarihi: 22 Mayıs, 2018
2. Lichman, M., 2013. UCI Machine Learning Repository. Erişim Adresi: http://archive.ics.uci.edu/ml , Erişim Tarihi: 22 Mayıs, 2018
3. Şeker (Diabet Hastalığı Nedir?. Erişim Adresi: http://tasovadh.saglik.gov.tr/TR,44432/seker--diyabet-hastaligi-nedir-ne-yapilmadir.html ,Erişim Tarihi: 22 Mayıs, 2018
4. I. MILLINGTON ve J. FUNGE, ARTIFICIAL INTELLIGENCE FOR GAMES, AMSTERDAM: Morgan Kaufmann Publishers is an imprint of Elsevier, 2009.
5. A. Gibson ve J. Patterson, Deep Learning, O'Reilly Media, Inc., 2017.
6. Anwer. A.M.O., Derin öğrenme ile Göğüs Kanseri Teçhizi, THK Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Eylül 2017
7. Şahin G.B., Gökhan T., Çetin A., Veri Madenciliği Yöntemleri İle Diyabet Hastalığına Sebep Olan Faktörlerin Tespiti, AB 2017, 2017
8. İnsan Beyninde Kaç Nöron Vardır?, Erişim Adresi: http://www.kuark.org/2014/01/insan-beyninde-kac-noron-vardir/ , Erişim Tarihi: 22 Mayıs, 2018
9. E. ÖZTEMEL, YAPAY SİNİR AĞLARI, PAPATYA YAYINCILIK EĞİTİM, 2006.
10. J. Ba ve D. Kingma, «Adam: A Method for Stochastic Optimization,» arXiv preprint arXiv:1412.6980, 2014.
11. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, pp. 436–444, 2015.
12. Y. Bengio, “Learning Deep Architectures for AI,” Found. trends® Mach. Learn., vol. 2, no. 1, pp. 1–127, 2009.
13. Understanding LSTM Networks (2017, HAZİRAN 7) http://colah.github.io/posts/2015-08-Understanding-LSTMs/
14. Y. LeCun ve Y. Bengio, «Convolutional Networks for Images, Speech, and Time,» %1 içinde Neural Information Processing: Research and Development, springer, 1995, pp. 276-279.
15. Machine Learning in python ,Erişim Adresi: https://machinelearningmastery.com/machine-learning-in-python-step-by-step/ Erişim Tarihi: 22 Mayıs, 2018
16. Veri Bilimi, Erişim Adresi: http://www.datascience.istanbul/, Erişim Tarihi: 22 Mayıs, 2018
17. Using LSTM in data, Erişim Adresi: https://www.kaggle.com/arjunsurendran/using-lstm-on-training-data/notebook , Erişim Tarihi: 22 Mayıs, 2018
18. G. L. F. d. Silva, O. P. d. S. Neto, A. C. Silva, A. C. d. Paiva ve M. Gattass, «Lung nodules diagnosis based on evolutionary convolutional neural network,» Multimedia Tools and Applications, pp. 1-17, 2017.
19. A. Gibson ve J. Patterson, Deep Learning, O'Reilly Media, Inc., 2017.
20. J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural Networks, vol. 61, pp. 85–117, 2015.
21. [15] C. Cortes and V. Vapnik, “Support-Vector Networks,” Mach. Learn., vol. 20, no. 3, pp. 273– 297, 199

22. Bozkurt M.R., Yurtay N,Yılmaz Z., Sertkaya C., Comparison of different methods for determining diabetes, Turkish Journal of Electrical Engineering & Computer Sciences, (2014) 22: 1044 – 1055, TUBİTAK, doi:10.3906/elk-1209-82
23. Onursal Ç. Temurtaş F, Gülgönül Ş., An application of multilayer neural network on hepatitis disease diagnosis using approximations of sigmoid activation function, Dicle Tıp Dergisi, Cilt 42, Sayı 2 (2015)
24. Lecture Notes. (2016, July 30). http://web.stanford.edu/class/cs224n/handouts/CS224N_DeepNLP_lecture1.pdf
25. Kanık E.A., Erden S., Tanı Testlerinin değerlendirilmesinde ROC (Receive Operating Characteristics) Eğrisinin Kullanımı, Mersin Üniversitesi Tıp Fakültesi Dergisi 2003;3:260-264
26. Deep Learning, (2018,mayıs,23) https://medium.com/deep-learning-turkiye