T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİ YÖNTEMLERİYLE EL YAZISI TANIMA

YÜKSEK LİSANS TEZİ

Rabia KARAKAYA

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Dr. Öğr. Üyesi Serap ÇAKAR**

Eylül 2020

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİ YÖNTEMLERİYLE EL YAZISI TANIMA

YÜKSEK LİSANS TEZİ

Rabia KARAKAYA

Enstitü Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Bu tez 11/09/2020 tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Rabia KARAKAYA

11.09.2020

TEŞEKKÜR

Yüksek lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Dr. Öğr. Üyesi Serap ÇAKAR'a teşekkürlerimi sunarım.

Eğitimim ve tez yazım sürecim boyunca her türlü desteği sağlayan, yardımlarını esirgemeyen Sakarya Üniversitesi Bilgisayar Mühendisliği Bölüm hocalarıma teşekkür ederim.

Ve tüm eğitim hayatım boyunca benden maddi ve manevi desteklerini esirgemeyen her zaman yanımda olan sevgili eşime ve aileme teşekkürlerimi bir borç bilirim.

İÇİNDEKİLER

TEŞEKKÜR i
İÇİNDEKİLER ii
SİMGELER VE KISALTMALAR LİSTESİ v
ŞEKİLLER LİSTESİ vi
TABLOLAR LİSTESİ vii
ÖZET vii
SUMMARY ix

BÖLÜM 1.
GİRİŞ 1
1.1. Literatür Tarama..... 2

BÖLÜM 2.
KAYNAK ARAŞTIRMASI 3
2.1. El Yazısı Tanıma..... 3
2.1.1. Etkileşimli el yazısı tanıma..... 3
2.1.2. Etkileşimsiz el yazısı tanıma..... 4
2.1.2.1. Ön işleme..... 5
2.1.2.2. Dilimleme..... 7
2.1.2.3. Öznitelik çıkarımı..... 7
2.1.2.4. Sınıflandırma..... 8
2.1.2.5. Son işleme..... 8
2.2. Makine Öğrenmesi 8
2.2.1. Gözetimli öğrenme..... 9
2.2.2. Gözetimsiz öğrenme 9

2.3. El yazısı Tanıma Sistemlerinde Kullanılan Bazı Algoritmalar…….. 10
2.3.1. Destek vektör makinesi …… 10
2.3.2. Yapay sinir ağı ……..…… 11
2.3.3. K-en yakın komşu algoritması ……. 15
2.3.4. Naive bayes sınıflandırıcı ……..…….. 17
2.3.5. Saklı markov modelleme …….……. 17
2.3.6. Karar ağaçları ……...……. 18
2.3.7. K-ortalama algoritması ……...……. 19

BÖLÜM 3.

MATERYAL VE YÖNTEM …….……..…… 21
3.1. MNIST ……..….. 21
3.2. Scikit Learn ……. 22
3.3. Geliştirilen Uygulama..……..….. 23

BÖLÜM 4.

ARAŞTIRMA BULGULARI …….……..….. 25
4.1. Karşılaştırılan El Yazısı Tanıma Yöntemleri ……..…….…… 25
4.1.1. Destek vektör makinesi ……..….. 25
4.1.1.1. Doğrusal destek vektör makinesi……..….. 26
4.1.1.2. Doğrusal olmayan destek vektör makinesi..…….. 26
4.1.2. Karar ağacı …… 28
4.1.3. Rastgele orman ……. 31
4.1.4. Yapay sinir ağı ……..…… 32
4.1.5. K-en yakın komşu algoritması …… 34
4.1.6. K-ortalama algoritması …… 35

BÖLÜM 5.

SONUÇLAR VE RAPOR ……..... 38
5.1. Algoritmaların Sonuçlarının Değerlendirilmesi…… 38
5.2. Çalışmanın Faydaları ……... 40

KAYNAKLAR ………………………………………………………………………. 41
ÖZGEÇMİŞ ………………………………………………………………...... 44

SİMGELER VE KISALTMALAR LİSTESİ

DVM	: Destek vektör makinesi
MNIST	: Modified National Institute of Standards and Technology
SVM	: Support vector machine
YSA	: Yapay sinir ağı

ŞEKİLLER LİSTESİ

Şekil 2.1. El yazısı tanıma adımları ….…….……..… 4
Şekil 2.2. Eşikleme ile arka plan renginin değiştirilmesi ..……..…….…..... 6
Şekil 2.4. El yazısında eğrilik ve eğim ……...…….. 7
Şekil 2.5. Gözetimli ve gözetimsiz öğrenme …….……..…….. 9
Şekil 2.6. YSA hiyerarşik yapısı …….…….. 11
Şekil 2.7. Yapay sinir hücresinin yapısı ……..…..……. 12
Şekil 2.8. K-en yakın komşu sınıflandırması ……..…….. 15
Şekil 2.9. K değerlerine göre küçük harfler ve büyük harfler için tanıma oranları... 16
Şekil 2.10. Bayes teoremi formülü ……...… 17
Şekil 2.11. Saklı Markov modelleme olasılık parametreleri ……..….. 18
Şekil 2.12. Düğüm, dal ve yapraklardan oluşan basit bir karar ağacı yapısı …..…. 19
Şekil 2.13. K-Ortalama kümeleme adımları ……..…….. 20
Şekil 3.1. MNIST 1 rakam verisinin matris formunda gösterimi…….…….….. 24
Şekil 4.1. DVM için 2 sınıflı problem örneği ……..… 26
Şekil 4.2. Doğrusal olarak sınıflandırılamayan girdi uzayının bir üst boyuta çekirdek fonksiyonu ile haritalanması ……..……... 27
Şekil 4.3. Scikit Learn destek vektör makinesi raporu …… 28
Şekil 4.4. Scikit Learn karar ağaçları önemli piksel değerleri ……..... 30
Şekil 4.5. Scikit Learn karar ağaçları raporu…… 30
Şekil 4.6. Scikit Learn rastgele orman önemli piksel değerleri …….. 31
Şekil 4.7. Scikit Learn rastgele orman raporu ……. 32
Şekil 4.8. Scikit Learn yapay sinir ağı raporu ……. 34
Şekil 4.9. K-En yakın komşu algoritması raporu ……..... 35
Şekil 4.10. MNIST veri seti K-ortalama rakam kümeleri …… 36
Şekil 4.11. 5 rakamının görsellerini içeren küme …….…… 37

TABLOLAR LİSTESİ

Tablo 2.1. Sinir sistemi ile YSA'nın benzerlikleri ……...…….. 11
Tablo 2.2. K=5 değeri için örnek veri tablosu ……..……. 16
Tablo 3.1. MNIST veri seti ……... 21
Tablo 4.1. MNIST veri seti ile yapılan test değerleri ……..……. 27
Tablo 5.1. Karşılaştırılan algoritmaların verimleri …….…….. 38

ÖZET

Anahtar kelimeler : El yazısı tanıma, makine öğrenmesi, python, mnist veri seti

El yazısı tanıma problemi makine öğrenmesi çalışmalarında her zaman kendisine yer bulmuştur. El yazısı tanıma işlemi bir çok aşamadan oluşmaktadır. Bunlar; ön işleme, dilimleme, öz nitelik çıkarımı, sınıflandırma ve son işleme aşamalarıdır. Bu aşamaların her biri kendi özelinde ayrı birer çalışmaya konu olabilecek kapsamdadırlar. Bu tez çalışmasında sınıflandırma aşamasında yapılan; ön işlemesi, dilimlemesi yapılmış veri setini tanıma işlemi üzerinde çalışılmıştır.

Çalışmada MNIST veri seti kullanılmıştır. Bu veri seti 250 farklı kişiden alınan 60.000 numune içerir. 0-9 arası rakamların el yazısı görsellerini kullanıma sunar. Akademik çalışmalarda sıklıkla tercih edilmektedir.

MNIST veri seti üzerinde bir çok makine öğrenmesi yöntemi çalıştırılmıştır. Her bir yöntem için verim hesaplanmış, sonuçlar raporlanmıştır. Literatür taramasında en çok karşılaşılan ve el yazısı tanıma alanında en çok tercih edilen makine öğrenmesi yöntemlerine öncelik verilmiştir. Bu araştırmalar ve testler sonucunda karşılaştırılan yöntemlerden en yüksek verimi sağlayan yöntemler K-En Yakın Komşu algoritması ve K-Ortalama algoritması olmuştur. Elbette çalışma süresi, veri seti gibi bir çok etkenin de varlığı bu alanda çalışmak isteyenler için kıstas olarak alınmalıdır. Bu tez çalışması el yazısı tanıma alanına temelden bir giriş yapmış ve makine öğrenmesi alanında farklı bir çok alt yöntemi el yazısı rakam tanıma için test etmiştir. Bu açıdan yol gösterici olması hedeflenmiştir.

HANDWRITING RECOGNITION USING MACHINE LEARNING

SUMMARY

Keywords: handwriting recognition, machine learning, python, mnist data set

The problem of handwriting recognition has always been involved in machine learning studies. There are many stages of handwriting recognition. These are; pre-processing, segmentation, feature extraction, classification and post-processing stages. Each of these stages can be subject to a separate academic study. The subject of this study is the classification of the pre-processed and segmented data set.

MNIST data set was used in this study. This data set contains 60,000 samples from 250 different people. Includes handwritten images of numbers 0 to 9. It is frequently preferred in academic studies.

Many machine learning methods were run on the MNIST data set. Accuracy score was calculated for each method and results were reported. For the study, the most common machine learning methods in the field of handwriting recognition were selected.

As a result of these researches and tests; the methods providing the highest efficiency from the methods compared were K-Nearest Neighbor algorithm and K-Means algorithm. Of course, the existence of many factors such as working time and data set should be taken as a criterion for those who want to work in this field. This thesis study has made a basic introduction to the field of handwriting recognition and has tested many different machine learning methods for handwriting digit recognition. It is aimed to be a guide in this study area.

BÖLÜM 1. GİRİŞ

Günümüzde el yazısı tanıma sistemleri birçok alanda kullanılmaktadır. Örneğin form veya dökümanların taranıp elektronik ortama aktarılmasında el yazısı tanıma sistemleri kullanılır. Bu sistemler özellikle geçmiş zamanlardan kalma dökümanların aktarımını büyük ölçüde kolaylaştırmaktadırlar. Günümüzde de özellikle bankalar gibi yazışmaların, bireysel işlemlerin yoğun olduğu iş alanlarında bilgi girişi maliyetini büyük oranda düşürebilmektedirler [1].

Bu örneklere ek olarak çevrimiçi(anlık) tanıma yapan sistemler de yaygın olarak kullanılmaktadır. Özellikle eğitim alanında bu yöntemlerden faydalanılmaktadır. Öğrencilerin yazmayı öğrenmesine katkı sağlayan çokça uygulama bulunmaktadır. Yine el yazısı tanıma tekniklerinden, bedensel veya zihinsel engeli olan kişiler için özelleştirilmiş eğitim-öğretim uygulamalarında yararlanılmaktadır [7].

Yazı tanıma alanında özellikle matbaa baskısı, bilgisayar veya daktilo çıktısı olan metinlerde, tanıma işlemi yüksek başarı oranına sahiptir ve hızlıdır. Bu teknoloji yine birçok iş kolunda fayda sağlarken günlük işlerimizi de büyük oranda kolaylaştırmaktadır. Örneğin artık her an elimizin altında olan akıllı telefonlarımızda da bu teknolojiyi çokça kullanabilmekteyiz. El yazısı veya dijital baskıları tanıyıp ilgili yazıyı farklı dillere çevirebilen, internet üzerinde aramalar yapabilen pek çok mobil uygulama mevcuttur.

Bu çalışmada ise tüm bu sistemlerde ve uygulamalarda en çok kullanılan el yazısı tanıma algoritmaları tanıtılmıştır. Bu algoritmalar üzerinde belirli bir veri setiyle testler yapılmış ve sonuçları raporlanmıştır. Bu şekilde algoritmalar karşılaştırılmıştır.

Bu çalışmanın el yazısı tanıma alanında çalışmak isteyenler için, algoritma seçimi aşamasında bir kaynak teşkil etmesi amaçlanmıştır.

1.1. Literatür Tarama

Engin Dağdeviren, MNIST veri tabanını kullanarak yaptığı el yazısı rakam tanıma çalışmalarında; destek vektör makineleri ve yapay sinir ağlarını elde edilen doğruluk oranlarına göre karşılaştırmıştır. MATLAB üzerinden gerçekleştirdiği testlerinde destek vektör makineleri için 10000 verilik bir veri setinde %99,97'lik bir başarı oranına erişmiştir. YSA kullanarak geliştirdiği el yazısı tanıma sisteminde ise yine 10000 verilik bir veri setinde %80,39'luk bir başarı sağlamıştır [5].

Murat ve arkadaşları ise 172 kişiden toplanan verilerle toplamda 9976 karakterlik bir veri seti oluşturmuşlardır. Karakter tanıma aşamasında K-en yakın komşu algoritmasını kullanmışlardır. Doğrulama aşamasında sınırlı sayıda sözcük içeren bir sözlük kullanmışlardır. Bu şekilde anlamsız kelimeler düzeltilmiştir. Başarı oranları ise küçük harfler için %90,5, büyük harfler için %91,2 doğru tanıma şeklindedir. Sistemin genelinde ise küçük harflerle yazılmış 152 kelime ve büyük harflerle yazılmış 62 kelime için test yapılmış ve %100 başarı görülmüştür [6].

Murat Şekerci, Türkçe el yazısı tanıma üzerine yaptığı çalışmada, karakter tanıma aşamasında korelasyon yöntemini kullanmış, K-en yakın komşuluk algoritmasıyla sistemlerini güçlendirmiştir. Veri seti 172 farklı kişiden alınan yazı örnekleriyle oluşturulmuştur. Bu çalışma ile rakamlarda %93, küçük harflerde %90.4, büyük harflerde ise %91.2 tanıma oranlarına ulaşılmıştır. Bu oranlar birleşik yazılan kelime

ve rakamlarda %10 oranında düşmüştür. Ayrık yazılan kelime ve rakamlarda ise %100 başarı sağlanmıştır. Ayrıca sözlük kullanılarak tanıma sistemi güçlendirilmiştir [1].

Uğur ve arkadaşları rakamlar ve ingiliz alfabesindeki büyük karakterler ile toplam 36 karakter ve her karaktere ait 100 örneği içeren bir veri seti oluşturmuşlardır. Bu veri seti üzerinde karakter tanıma işlemi uygulamışlardır. Öznitelik çıkarımı aşamasında Mesafe Hesaplama Algoritması ile her örnek için 8 adet öznitelik çıkartmışlardır. Elde edilen öznitelikler; Naive Bayes, Lineer çekirdek fonksiyonlu LibSVM ve Öklid bağıntılı k-NN algoritmaları yardımıyla, 10-Kat Çapraz Doğrulama (Ten Fold Cross Validation) – Birini Dışarıda Bırak (Leave One Out, LOO) yöntemi kullanılarak sınıflandırılmıştır. Elde edilen doğruluk oranları ise Naive Bayes için %65, LibSVM için %81, K-en Yakın Komşu Algoritması için %89 şeklindedir [10].

Tsehay Admassu Assegie yaptığı çalışmada 42000 satır ve 720 sütundan oluşan bir veri seti üzerinde karar ağaçları algoritması ile rakam tanıma uygulaması geliştirmiştir. %83.4'lük bir başarı sağlanmıştır [12].

BÖLÜM 2. KAYNAK ARAŞTIRMASI

2.1. El Yazısı Tanıma

Bu bölümde el yazısı tanıma sistemleri ve bu sistemlerde kullanılan makine öğrenmesi algoritmaları tanıtılmıştır.

El yazısı tanıma sistemleri birçok alanda ihtiyaç duyulan sistemlerdir. Günümüzde hemen hemen tüm işlemler dijital ortama taşınmış durumdadır. Bu aşamada el yazısı tanıma sistemlerine büyük görev düşmektedir. Bu sistemlerle el yazısı ile yazılmış mevcut dökümanların dijital ortama aktarılması sağlanabilmektedir. Bunun dışında da eğitim, sağlık gibi bir çok alanda el yazısı tanıma sistemleri büyük önem arz etmektedir.

Yine banka sistemlerinde çeklerin okunması, gelen postaların okunup sıralanması gibi alanlarda el yazısı tanıma sistemlerinden faydalanılır [4].

El yazısı tanıma yöntemleri iki grupta toplanabilir. Bunlar etkileşimsiz (çevrimdışı) ve etkileşimli (çevrimiçi) yöntemlerdir.

2.1.1. Etkileşimli el yazısı tanıma

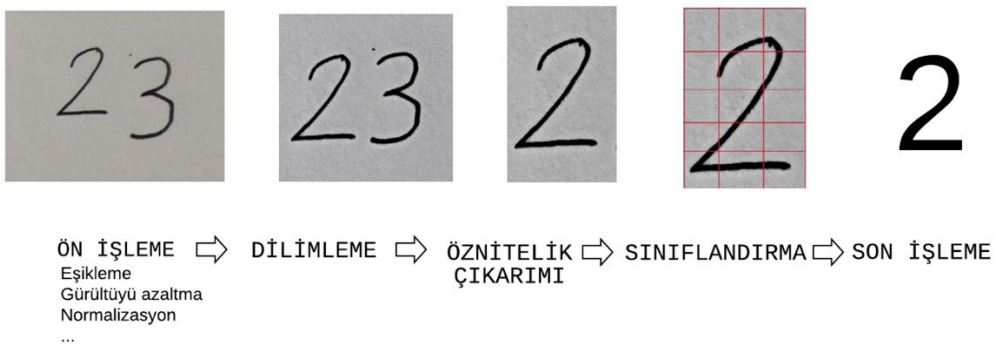
Etkileşimli yöntemler el yazısını yazıldığı sırada tanımaya yönelik tasarlanmışlardır. Genelde dokunmatik özelliği olan kalemler ve tabletlerden faydalanılarak kullanılırlar. Bu sistemlerde kalemin her hareketi, dokunuşu, devamlılığı izlenir. Elektronik ajandalar gibi günlük birçok alanda kullanılmaktadırlar. Etkileşimli sistemlerin kullanım alanı her geçen gün genişlerken bu alandaki ihtiyaç artmaktadır. Özellikle

kalem algılayan veya dokunmatik tabletlerin kullanımının artması bu alandaki ihtiyacın artmasına sebep olmuştur [4].

2.1.2. Etkileşimsiz el yazısı tanıma

Etkileşimsiz sistemler genelde önce kağıt üzerine el yazısının yazılması ve sonrasında dijital ortama aktarılması şeklindedir. Anlık çıktı beklenmediğinden etkileşimli sistemlere göre daha yavaş çalışabilirler. Öte yandan etkileşimli sistemlerde kullanıcıyla anlık etkileşim sağlanabileceği için her harfi tek tek algılayabilme, gerektiğinde tekrar girdi isteyebilme gibi avantajları vardır. Etkileşimsiz sistemlerde bu imkanların olmaması işi biraz daha zorlaştırabilmektedir. Özellikle geçmişte yazılmış dökümanların okunaklı ve temiz olmaları her zaman mümkün olmamaktadır. Bu sebeplerden dolayı daha fazla ön işlemeye ihtiyaç duyulmaktadır [2].

El yazısı tanıma işlem adımları şu şekilde sıralanabilir;



Şekil 2.1. El yazısı tanıma adımları

2.1.2.1. Ön işleme

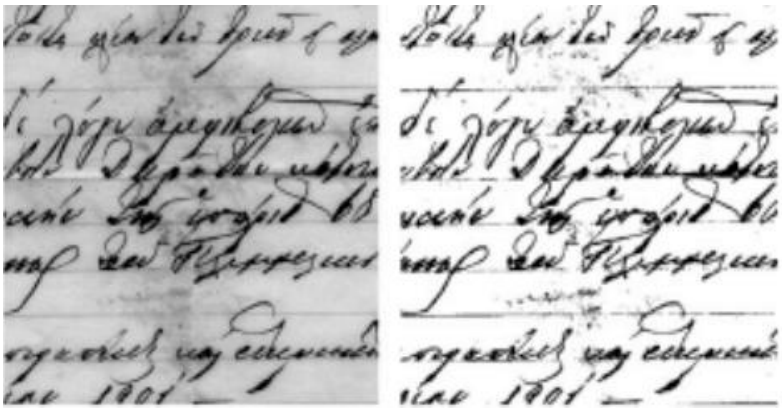
Bu aşamada gürültü azaltılması, normalleştirme, referans çizgisinin bulunması gibi işlemler yapılır. Normalleştirme işlemi yazı fontlarından kaynaklı farklılıkların ortadan kaldırılarak standart bir hale getirilmesidir. Yazının eğimi varsa veya yukarıdan aşağıya, çapraz yazılmışsa bu aşamada yine düzenlemesi yapılır. Bir başka normalleştirme işlemi ise yazı karakterlerinin boyutlarının birbirlerine eşitlenmesidir. Referans çizgisinin belirlenmesi ile de birbirine benzeyen karakterlerin ayrımı sağlanır. Örneğin '9' rakamı ile 'g' karakterleri birbirine benzer, ancak referans çizgisi ile ayırt edilebilirler [1].

Yine ön işleme adımında yapılan bir diğer işlem ise el yazısı verisinin siyah beyaz(binary) bir veriye dönüştürülmesidir. Bu işleme eşikleme denmektedir. Eşikleme, siyah beyaz bir görüntünün grinin tonlarına çevrilmesiyle yapılır [7].

Ön işleme aşamasında yapılan işlemler şöyle sıralanabilir :

Eşikleme : Eşikleme işleminin mantığında arka plandan ön plana çıkarma vardır. Bir dökümanın gri skalalı histogramında iki tepe noktası vardır. Bu tepe noktaları gri skalanın beyaz veya siyaha olan yakınlığına göre değer belirlenir.

Arka plan ve ön plan noktalarının ayrımı gerçekleştirilir ve bu şekilde daha net işlenebilecek karakterler ortaya çıkartılır [7].

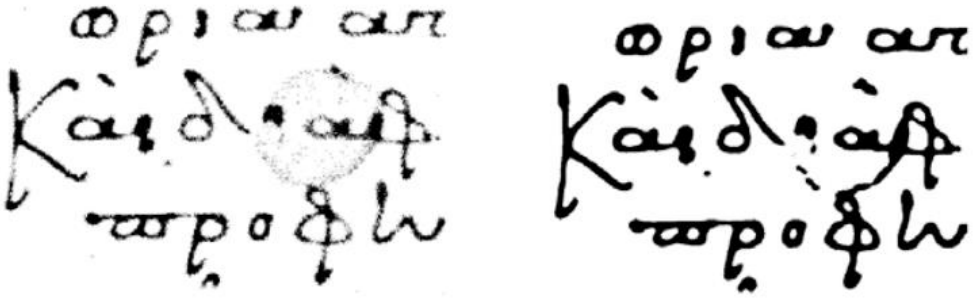


Şekil 2.2. Eşikleme ile arka plan renginin değiştirilmesi [7]

Gürültüyü azaltma : El yazısı dökümanlarda yazının vurguları korunurken dijital görüntülerde ya da taratılmış metinlerde gürültüler oluşabilir. Gürültü azaltma yöntemi bu aşamada yine temiz bir görüntüye ulaşılmasına yardımcı olmaktadır.

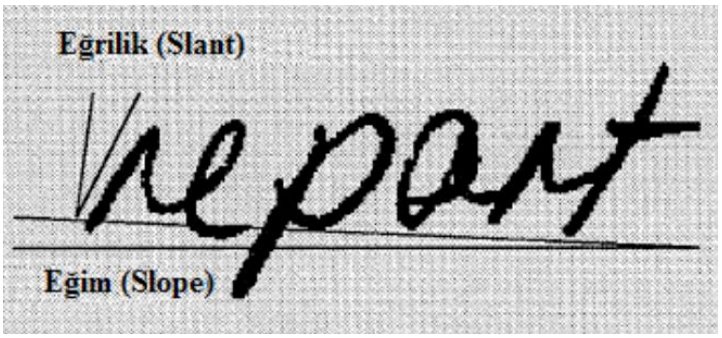
Örneğin taratılmış bir sözleşme dökümanında metnin üzerinde damga gibi gürültü oluşturabilecek unsurlar bulunabilir. Görseldeki örnekte görülebileceği üzere gürültü azaltma yöntemiyle bu durum temizlenebilir.

Gürültüyü azaltırken gürültüyü içeren pikseller üzerinde filtreleme yapılır. Maskeleme yöntemiyle komşuluk sayısı eşik değerinin altında kalan pikseller silinerek gürültü giderilmiş olur [7].



Şekil 2.3. Gürültüsü azaltılmış metin [7]

Normalizasyon(Eğim Düzeltme) : Ön işleme aşamasının bir diğer işlemi ise normalizasyondur. Bu işlemle yazıdaki eğim düzeltilir, eğrilikler giderilir. Özellikle banka çeklerindeki el yazısı metinleri tanımada sıklıkla kullanılan bir işlemdir. Dikey eksendeki sapmaya eğrilik(slant), yatay eksendeki sapmaya ise eğim(slope) denir [7].



Şekil 2.4. El yazısında eğrilik ve eğim [7]

Şekilde görüldüğü gibi eğrilik ve eğimlerin tespit edilmesi ve giderilmesi aşamasında histogramlardan faydalanılır. Bu aşamada genellikle Bosinovic ve Shrihari Metodu kullanılır [7].

2.1.2.2. Dilimleme

Bu aşamada sözcükler harflere ya da rakamlara karşılık gelecek parçalara bölünür. Bu işlem harfler arası boşluklara göre yapılabilir. Ancak bu yöntemde harflerde parçalanmalar olabilir. Bu yüzden alternatif bazı yöntemler önerilmiştir. Örneğin; tanıma ve dilimleme aşamalarının birleştirilmesi veya metnin önce çok küçük parçalara bölünmesi ve sonrasında genellikle Hidden Markov Modeli kullanılarak birleştirilmesi şeklindedir [2].

2.1.2.3. Öznitelik çıkarımı

Bu aşamanın hedefi verinin işlenerek daha kısıtlı bir uzayda tanımlanması ve tanıma aşamasına zemin hazırlanmasıdır. Fourier, Wavelet dönüşümleri, histogram yada izdüşüm bazlı yöntemler bu aşamada kullanılabilecek yöntemlere örnektir [2].

Öznitelik çıkarımı aşamasında elde edilen veriler önemli veriler olup, tanıma aşamasında yardımcı olmaktadırlar.

2.1.2.4. Sınıflandırma

Tanıma aşaması da diyebileceğimiz bu aşama genel anlamda bakıldığında sadece karakterin sembolik sınıfıyla eşleştirilmesinden ibarettir. Bu sebeple makine baskısı bir yazıda bu aşama çok daha kolaydır.

Tanıma aşamasında kullanılabilecek çok sayıda yöntem mevcuttur. Yapay sinir ağları, sınıflandırma algoritmaları bunlardan bazılarıdır. Bu aşamada kullanılacak veri seti büyük öneme sahiptir. Algoritmanın eğitildiği veri setinin çeşitliliği, zenginliği başarı oranını artıracaktır [2].

Sınıflandırma aşamasında kullanılabilecek algoritmalar makine öğrenmesi başlığı altında detaylandırılmıştır.

2.1.2.5. Son işleme

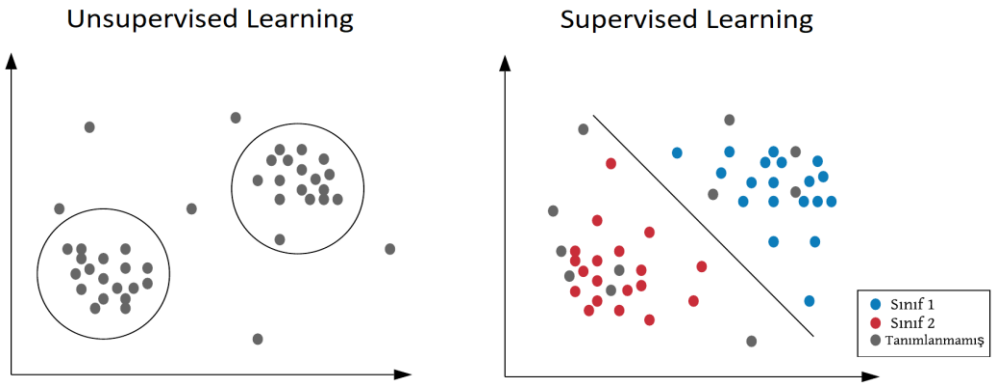
Bu aşamada tanıma işleminden sonra olası hataların giderilmesi hedeflenir. Bazı sistemler bu aşamada sözlüklerden yararlanırlar. Böylece tanıma işleminden sonra tanınan metin, sözlükteki olası kelimelerle karşılaştırılarak tekrardan kontrol edilmiş olur [4].

2.2. Makine Öğrenmesi

Yapay zekanın bir alt dalı olan Makine Öğrenmesi bilgisayarların bir takım verileri kullanarak öğrenme işlemini gerçekleştirmelerini amaçlayan algoritma ve tekniklerin gelişimi ile ilgili bir çalışma alanıdır [8].

Doğal dil işleme, el yazısı tanıma, görüntü işleme, medikal işlemler, finans işlemleri gibi birçok alanda makine öğrenmesi kullanılmaktadır.

Makine öğrenmesi gözetimli ve gözetimsiz öğrenme olarak iki yönteme sahiptir.



Şekil 2.5. Gözetimli ve gözetimsiz öğrenme

2.2.1. Gözetimli öğrenme (Supervised learning)

Gözetimli öğrenme yönteminde sisteme hem girdiler(etiketlenmemiş veri) hem de çıktılar(etiketlenmiş veri) verilir. Sistem bu veriler arasında eşleme yapan bir fonksiyon üretir. Bu yöntemle öğrenme işlemini gerçekleştirir [8].

Gözetimli öğrenme yöntemlerinde Sınıflandırma ve Regresyon yöntemleri çokça kullanılmaktadır. Sınıflandırma yöntemi, günlük yaşantımızda da en sık kullandığımız karar verme yöntemlerindendir. Bir sınıflandırma problemi herhangi bir nesnenin, önceki gözlemler sonucu elde edilmiş gruplardan birine ait olmasının belirlenmesi işlemidir. Örneğin hava tahmini, tıbbi tanılar, karakter tanıma, ses tanıma gibi birçok alanda kullanılmaktadır [13].

2.2.2. Gözetimsiz öğrenme (Unsupervised learning)

Gözetimsiz öğrenme yönteminde sistemi besleyen bir eğitim veri seti bulunmamaktadır. Sistem, çıktı verileri olmadan girdiler üzerinden öğrenme işlemini gerçekleştirir [8].

Gözetimsiz öğrenme algoritmaları genellikle örüntüler arasında benzerliklerden faydalanarak kümeleme yöntemini kullanırlar [14].

2.3. El Yazısı Tanıma Sistemlerinde Kullanılan Bazı Algoritmalar

El yazısı tanıma sistemlerinde kullanılan algoritmalar çok çeşitlilik gösterebilmektedirler. Üzerinde çalışılacak el yazısı verisinin karakteristiği, yapısı, başarı ve zaman beklentileri gibi kıstaslara göre farklı algoritmalardan faydalanılabilir.

Ortaya çıkış dönemleri oldukça eski olan ama halen kullanılan el yazısı tanıma teknikleri Hidden Markov modeli ve yapay sinir ağlarıdır. Son dönemlerde ise özellikle derin öğrenme ile bu alanda ciddi çalışmalar yapılmıştır. Her algoritmanın kendi özelinde birçok güçlü yanı vardır. Bu durum hibrit sistemlere yönelinmesine sebep olmuştur. Bu sistemlerde derin öğrenme veya yapay sinir ağları ile Hidden Markov modeli bir arada kullanılmıştır. Bu şekilde maksimum verim amaçlanmıştır [3].

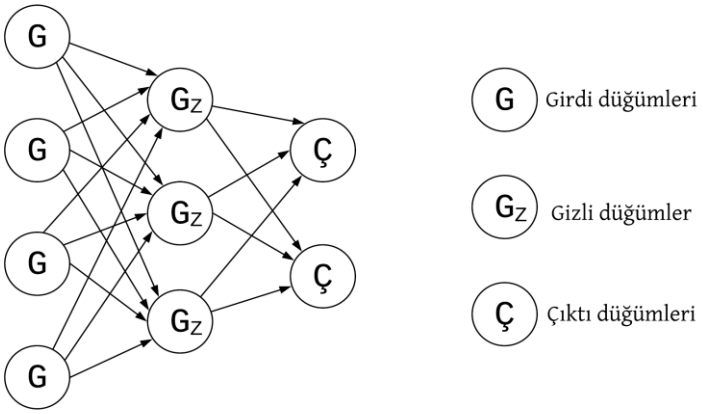
El yazısı tanıma sistemlerinde en çok faydalanılan algoritmalardan bazıları aşağıda detaylandırılmıştır.

2.3.1. Destek vektör makinesi

İlk kez 1960'larda sınıflandırma problemlerinin çözümü için önerilen Destek Vektör Makinesi(DVM), istatistiksel bir öğrenme teorisi yöntemidir. DVM yöntemi makine öğrenmesi alanında yakın dönemlerde kullanılmaya başlanmıştır. Özellikle yüz tanıma, ses tanıma, görsel sınıflandırma ve el yazısı tanıma alanları DVM'nin kullanım alanlarından en başta gelenlerdendir. DVM aynı zamanda gözetimli bir sınıflandırma yöntemidir. Bu sebeple daha önceden oluşturulmuş bir eğitim seti ile işlemlere başlar. DVM eğitim süresince hali hazırda varolan eğitim setindeki her bir veri ve etiketinin ilişkisini öğrenir. Bu ilişkilerle destek vektörlerini oluşturur [5].

2.3.2. Yapay sinir ağı

Yapay Sinir Ağı(YSA); sınıflandırma, gruplama tahminleme gibi bir çok problemin çözümünde kullanılan bir yöntemdir. Birden çok girdi alabilen ve birden çok çıktı verebilen matematiksel bir algoritmaya sahiptir. İlk kez 1943'lerde ortaya atılmıştır. Doğrusal olmayan problemlerin çözümünde yaygın olarak kullanılırlar [5].



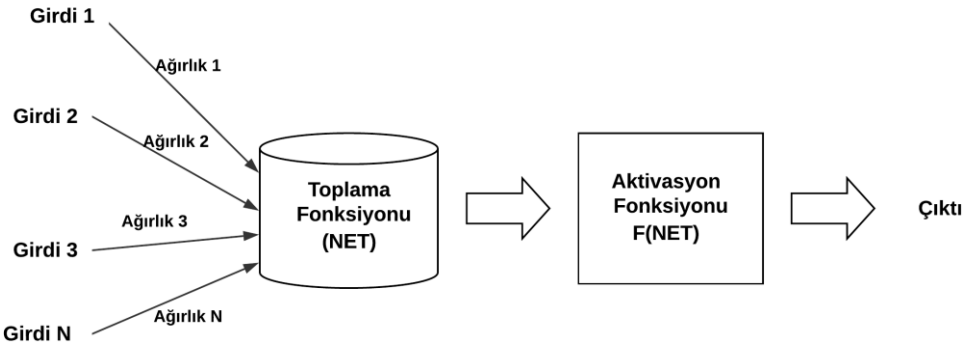
Şekil 2.6. YSA hiyerarşik yapısı

YSA modeli bilgisayar mimarisi yerine insan sinir sisteminin iletim modelini ve insan beyninin paralel hesaplama yeteneğini model alır. İnsan sinir sisteminde temel birim olan nöronların her biri belli verileri alır ve ikili bir çıktı üretirler. Benzer mantık YSA'lar için de geçerlidir [5].

Tablo 2.1.'de insan sinir sistemi ile YSA hücresinin benzerlikleri gösterilmiştir.

Tablo 2.1. Sinir sistemi ile YSA'nın benzerlikleri [23]

Sinir Sistemi	YSA Sistemi
Nöron	İşlem elemanı
Dentrit	Toplama fonksiyonu
Hücre gövdesi	Transfer fonksiyonu
Aksonlar	Eleman çıkışı
Sinapslar	Ağırlıklar



Şekil 2.7. Yapay sinir hücresinin yapısı [24]

Şekil 2.7.'de bir yapay sinir hücresinin yapısı gösterilmiştir. Bu yapılar kısaca şu şekilde açıklanabilir.

Girdiler: Bir yapay sinir hücresine dış dünyadan gelen bilgilerdir. Bunlar ağın öğrenmesi için verilen veriler tarafından belirlenir. Yapay sinir hücresine dış dünyadan olduğu gibi başka hücrelerden veya kendi kendisinden de bilgiler gelebilir.

Ağırlıklar: Yapay hücreye gelen bilginin önemini ve hücre üzerindeki etkisini gösterir. Pozitif, negatif, sabit veya değişken değerler olabilirler.

Toplama fonksiyonu: Bir hücreye gelen net girdiyi hesaplar. Girilen değerlere toplama fonksiyonları uygulanır ve her bir işlem elemanının net çıktı değeri bulunur.

Aktivasyon fonksiyonu: Hücrenin çıktısını belirler. Lineer fonksiyon, sinüs fonksiyonu, eşik değer fonksiyonu gibi örnekleri bulunmaktadır.

Çıktı: Aktivasyon fonksiyonu tarafından üretilen çıktı, dış dünyaya veya başka bir hücreye gönderilir. Hücre kendi çıktısını kendisine girdi olarak da gönderebilir.

YSA yapısı daha çok matematiksel olmayan problemlerin çözümü için kullanılabilir. Örneğin bir toplama işlemi için sinir ağı tasarlamak, eğitmek gereksiz maliyettir. Ancak öte yandan desen tanıma gibi işlemler için oldukça kullanışlıdır.

YSA'ların sınıflandırma problemini çözme becerisinin yüksek olması için birden fazla sinir yapısı kullanılmalıdır. YSA'larda bir veya birden fazla katman kullanılabilir. Giriş katmanında veri grupları ağa sunulur. Her bir nöron bir veri alır. Daha sonra veri, işlenmeden bir sonraki katman olan gizli katmana geçer. Gizli katman, ağın temel işlevini gören fonksiyonlarla veriyi işler ve sonraki katmana gönderir. Çıkış katmanı YSA'nın en uç katmanıdır. Gizli katmandan aldığı veriyi işleyerek çıktıyı verir [5].

YSA'ların diğer bilgi işlem yöntemlerinden farklılıkları şunlardır;

Paralellik: Alışılagelmiş yöntemlerde genelde süreç doğrusal olarak işler. Bu durum doğrusal olmayan problemler çözülmek istendiğinde yavaşlığa sebep olur. Ancak YSA insan sinir sistemini model aldığı için tıpkı onun gibi eş zamanlı çalışma özelliğine sahiptir. Bu durum hızı artırmakta ve karmaşık problemlerin çözümüne olanak sağlamaktadır.

Öğrenebilirlik: Mevcut bilgi işlem yöntemleri belirli bir algoritma kapsamında çalışarak kendileri ağırlık yada verileri yenileyememektedirler. Ancak YSA örneklerle kendini eğiterek gerekli verileri oluşturabilir.

Hata Toleransı: YSA'larda bir veya birkaç hücrenin kaybı sistemi bozmaz, ancak diğer sistemlerde herhangi bir elemanın kaybı sistemin çalışmasını durdurabilmektedir.
Uyarlanabilirlik: YSA'larda ağırlıklar yeniden yapılandırılabilir. Bu sayede belli bir problemi çözmek için eğitilen yapay sinir ağı problemdeki değişikliklere göre yeniden eğitilebilir ve farklı koşullara uyarlanabilir.

Genelleme: YSA eğitiminden sonra eğitim sırasında karşılaşılmayan test örnekleri de değerlendirilip istenilen tepkiler üretilebilir. Örneğin karakter tanımada hatalı bir karakter için doğru tanıma yapması için sistem eğitilebilir.

Yerel Bilgi İşleme: YSA'da karmaşık bir problemin tümüyle çalışılması yerine problem parçalar halinde işlenir. Bu durum karmaşık ve zor problemleri çözülebilir hale getirir.

Gerçekleme Kolaylığı: Karışık fonksiyonların yerine basit işlemlerin kullanılması gerçekleme kolaylığı sağlamaktadır.

Donanım ve Hız: YSA paralel yapısı sebebiyle hızlı bilgi işleme yeteneğine sahiptir [11].

YSA'da en uygun ağırlık setinin belirlenmesi için yapılan ağ hesaplamaları 2 aşamadan oluşur. Bunlar öğrenme ve test etme(hatırlama)dir.

Öğrenme: Ağlar örneklerle eğitilirler. Gözetimli ve gözetimsiz olarak iki öğrenme stratejisi de YSA için söz konusudur. Gözetimli öğrenmede ağa giriş-çıkış vektörleri şeklinde ayrıntılı eğitim değerleri verilmektedir. En yaygın öğrenme biçimidir.

Gözetimsiz öğrenmede ise ağa sadece giriş grubu sunulur ve bu girişe uygun bir çıkış değeri üretecek şekilde ağın kendisinin uygun ağırlıkları düzenlemesi beklenir.

Test Etme(Hatırlama): Ağın ilk aşamada öğrendiği ağırlık grubunu kullanarak kendisine verilen yeni giriş değerlerine doğru çıktılar üretmesi beklenir.

YSA'lar yapısal olarak da ileri beslemeli ve geri beslemeli olarak gruplandırılabilirler. Ayrıca hem ileri besleme hem geri besleme olarak tanımlanan ağlar da mevcuttur.

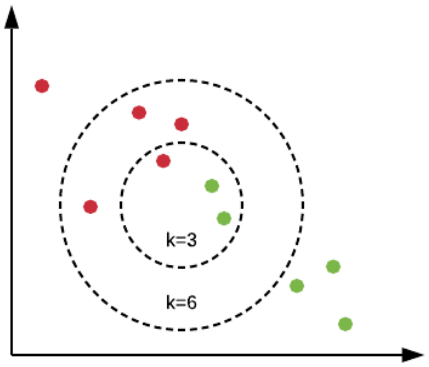
İleri Beslemeli YSA: Giriş, çıkış ve gizli olmak üzere 3 katmanı bulunur. Ağdaki bilgi akışı giriş katmanından çıkış katmanına göre yalnızca bir yönde ilerler. Yani nöronlar arka arkaya beslenirler.

Geri Beslemeli YSA: Bu tür ağlar geri besleme bağlantılarına sahiptir. Herhangi bir sinirin çıkışından girişine doğru bilgi akışı olmaktadır [22].

YSA uygulamalarında istatistiksel bağlantıların büyük önemi vardır. YSA ve istatistiksel yöntemler arasında büyük bir kesişme vardır. Örneğin en küçük kareler yöntemi YSA modellerinde de sıkça kullanılmaktadır. Veya bir aritmetik ortalama basit bir geri yayılma ağ ile kolayca hesaplanabilir.

İleri beslemeli YSA sınıflandırma modellerinin ve doğrusal olmayan regresyonun bir alt sınıfı haline gelmiştir. YSA'nın modern ve istatistiksel yöntemlere ek olarak kullanımı yaygınlaşmaya devam etmektedir. YSA bu alanlarda kullanılarak kendi avantajlarını da dahil etmekte ve daha yüksek verimde sonuçlar alınabilmektedir. Ancak YSA'nın görece daha karmaşık yapısı hesaplama yükünü artırması veya basit modelleri çok karmaşık modellemesi gibi dezavantajlar da doğurabilmektedir [22].

2.3.3. K-en yakın komşu algoritması



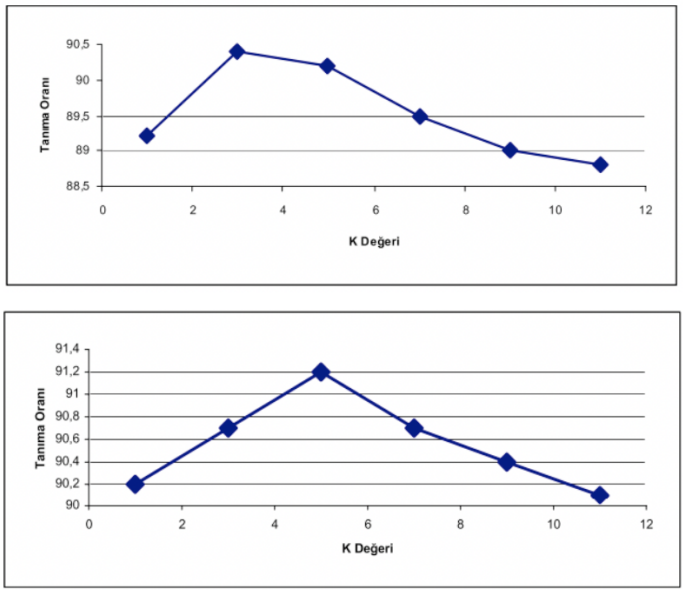
Şekil 2.8. K-en yakın komşu sınıflandırması [8]

K-en yakın komşuluk, sınıflandırma yöntemlerinden birisidir. Sınıfları belli olan bir örnek kümesindeki gözlem değerlerinden örneğe katılacak yeni bir gözlemin hangi sınıfa ait olduğunu belirlemek amacı ile K-en yakın komşu algoritması kullanılır [8].

Tablo 2.2. k=5 değeri için örnek veri tablosu [6]

Veritabanındaki örnek	Benzeme değeri
d	196
b	194
b	185
b	170
b	167

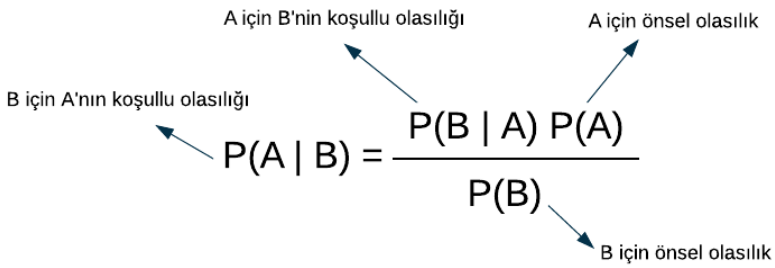
Tablo 2.2'deki 'k' değeri 1'den büyük ve genelde tek sayı olarak seçilen bir tam sayıdır. Bu yöntemde k tane en büyük benzeme değerine bakılarak sonuca ulaşılır. Aşağıdaki b harfinin tanınması için oluşturulmuş, k=5 değerine sahip örnek tablo incelendiğinde en yüksek benzeme değerinin veri setindeki bir d harfine ait olduğu görülür. Ancak K-en yakın komşuluk algoritması diğer sonuçlarla beraber değerlendirme yaparak en çok benzeme değerine sahip karakteri seçecektir [6]. K değeri mevcut veri setinin üzerinde yapılan testlerle küçük harfler ve büyük harfler için ayrı olacak şekilde belirlenir.



Şekil 2.9. K değerlerine göre küçük harfler ve büyük harfler için tanıma oranları [6]

2.3.4. Naive bayes sınıflandırıcı

Bayes karar yapısı, sınıflandırma hatalarını en aza indirgeyen bir sistemdir. Bunu, sınıflandırılacak veri ile ilgili ön bilgiye sahip olması ile sağlamaktadır. Bu algoritmada sınıflandırma için normalize edilmiş öznitelik vektörleri sınıflandırıcı uzayda kullanılır. Bu algoritma verilen örneklerin en büyük olasılıklarını hesaplar. Olasılık temelli çalışan algoritmada kısa sürede verimli sonuçlar elde edilebilmektedir [10].



Şekil 2.10. Bayes teoremi formülü

Naive Bayes, temeli belli bir ihtimalin var olduğu durumda bir diğer ihtimalin hesaplanmasına dayalıdır. Hesaplamayı doğru yapabilmesi için ciddi miktarda bir eğitime ihtiyaç duyar. Ayrıca her hesaplamada verileri tekrar tekrar tarar. Bu durum bu yöntemin maliyetli olmasına sebep olmaktadır.

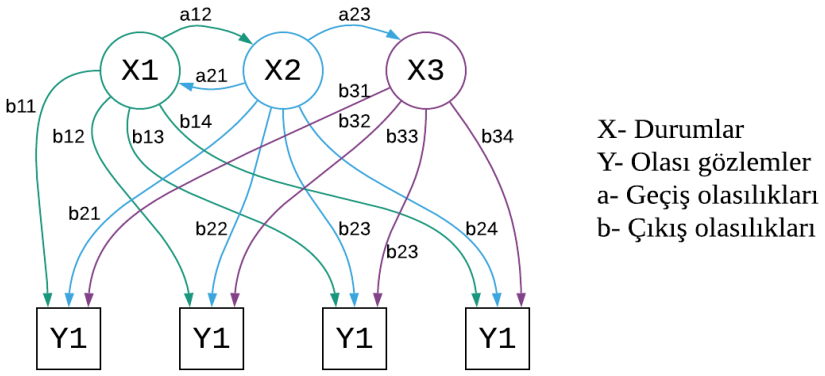
Ancak özellikle metin tanıma gibi işlemlerde yüksek verimle çalışabilmektedir. Bu gibi işlemlerde eğitim serisinin statik olması naive bayes sınıflandırıcının bir kez sistemi öğrendikten sonra hızlı ve verimli çalışmasını sağlar.

2.3.5. Saklı markov modelleme

Saklı Markov sınıflandırma işleminde verilerin birbirinden farklı oldukları varsayılır. Her bir elemanın maksimum olabilirlik derecesine göre hangi sınıfa ait olduğu

hesaplanır. Bu algoritmada amaç, belirli bir sınıfın olma olasılığını bulmak ve buna göre verileri sınıflandırmaktır [7].

Markov modelinde bir durum sadece mevcut geçmiş durumlara göre belirlenir, aynı şekilde gelecek durumlar sadece mevcut durumlara göre belirlenir. Yani t zamanında Q durumunda olma olasılığı sadece n-1 zamanındaki gözlemlere bağlıdır.



Şekil 2.11. Saklı Markov modelleme olasılık parametreleri

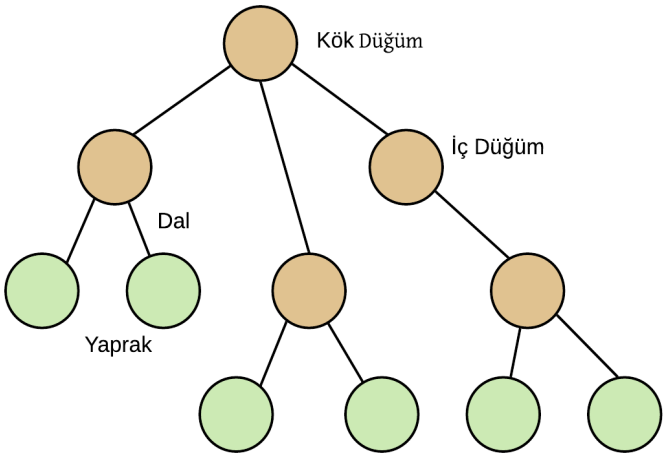
Bir Saklı Markov Modelleme yapısında temel olarak:

- Bir durumlar dizisi; Q = S1, … , SN,
- Her bir durum için ihtimallerin belirtildiği geçişler dizisi;
  P(Qt = Si | Qt-1 = Sj)(Burada Sj durumu mevcutken Si olasılığı belirtilmiştir.)
- Başlangıç durumlarının olasılık dağılımları; P(Q1 = S), ∀S ∈ Q,

bulunmalıdır [3].

2.3.6. Karar ağaçları

Karar ağaçları, bir problemin çözümü için problem yapısına göre bir ağaç yapısı şeklinde sınıflandırma modeli oluşturmaktadır. Ağaç yapısının ve kuralların anlaşılabilir olması algoritmanın uygulanmasını kolaylaştırır. Karar ağaçları yöntemi ardışık basit karar verme işlemlerinden oluşur.



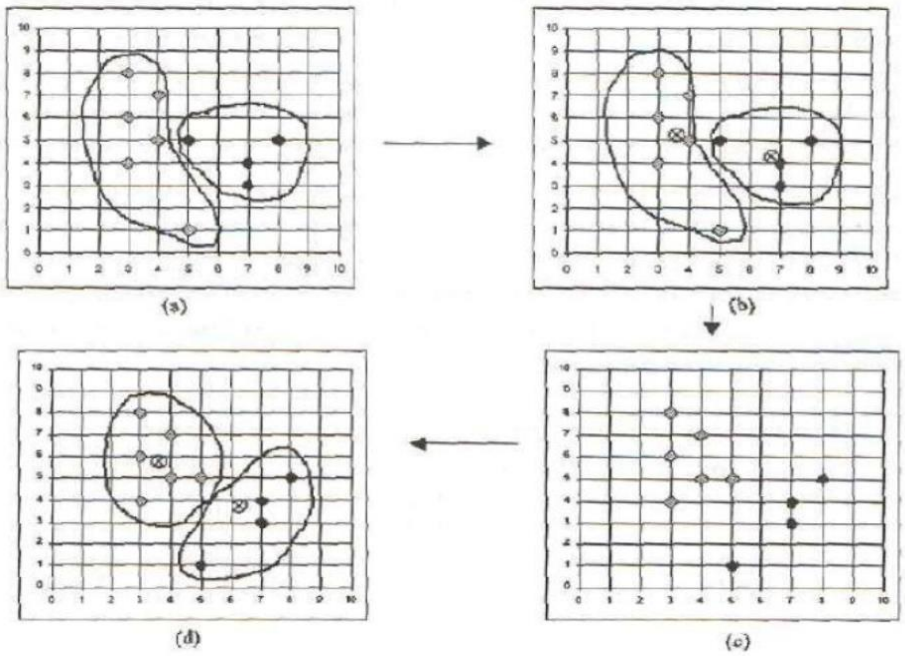
Şekil 2.12. Düğüm, dal ve yapraklardan oluşan basit bir karar ağacı yapısı [9]

Ağaç yapısında şekilde gösterildiği gibi düğümler ve dallar bulunur. Algoritma çalışırken ilk düğümden başlanarak sorular sorulmaya başlanır ve son eleman olan yapraklara ulaşana kadar ağacın büyümesi ve dallanması devam eder [9].

2.3.7. K-ortalama algoritması

Bu algoritma en çok tercih edilen gözetimsiz öğrenme algoritmalarından biridir. Verilerin benzerlik oranlarına bakarak kümeler oluşturulur. Algoritmada oluşturulacak küme sayısı önceden belirlenir. Küme sayısı “k” ile ifade edilir. Algoritma temel olarak 4 aşamadan oluşur:

- Küme merkezlerinin belirlenmesi
- Küme merkezleri dışındaki örneklerin mesafelerine göre sınıflandırılması
- Yapılan sınıflandırmaya göre yeni merkezlerin belirlenmesi veya eski merkezlerin güncellenmesi
- Kararlı hale gelinene kadar 2. ve 3. adımların tekrarlanması



Şekil 2.13. K-Ortalama algoritması kümeleme adımları [27]

BÖLÜM 3. MATERYAL VE YÖNTEM

3.1. MNIST

Bu çalışmada sistemin eğitimi ve testler için kullanılan veri seti olarak "Modified National Institute of Standarts and Technology"(MNIST) kullanılmıştır. MNIST veritabanı uluslararası alanda yaygınca kullanılmaktadır. El yazısı rakamlardan oluşur. MNIST veritabanında 60.000 eğitim verisi, 10.000 test verisi bulunmaktadır. Rakamların her biri 28x28 piksel ölçüsündeki çerçevede 20x20 piksel ölçüsündedir. Detaylı veriler Tablo 3.1.'de gösterilmiştir [20].

Tablo 3.1. MNIST veri seti [20]

Rakam	Eğitim	Test
0	5.923	980
1	6.742	1.135
2	5.958	1.032
3	6.131	1.010
4	5.842	982
5	5.421	892
6	5.918	958
7	6.265	1.028
8	5.851	974
9	5.949	1.009

3.2. Scikit Learn

Scikit Learn kullanıma hazır bir Python kütüphanesidir. Bir çok makine öğrenmesi algoritmasını etkin bir şekilde uygulama imkanı sunar. Yüksek seviyeli dillerle makine öğrenmesi algoritmalarının hızlı bir şekilde eğitilmesi ve testine imkan sağlar. Minimum bağımlılıkla en iyi performansı sunmayı amaçlar. Scikit Learn, detaylı içeriğe sahip dökümantasyon ve örnek projeleri kullanıma açık olarak sunmaktadır. Destek vektör makineleri, rastgele orman, gradyan arttırma, K-ortalama dahil çeşitli sınıflandırma, regresyon ve kümeleme algoritmalarına sahiptir [21].

Scikit Learn kütüphanesinin temelini oluşturan teknolojiler şunlardır;
Numpy: Veri ve modelleri saklayan temel kütüphanedir. Girdi verileri numpy dizileri olarak tutulurlar. Bu durum verilerin bir çok bilimsel kütüphanenin kullanabileceği formatta olmasını sağlar. Ayrıca temel aritmetik işlemleri de sağlar.

Scipy: Bilimsel hesaplamalar yapmak için kullanılan bir kütüphanedir. Lineer cebir, matris ve bir çok özel istatistiksel algoritmayı barındırmaktadır.

Cython: Python içerisinde C programlama diliyle kod yazabilmek için kullanılır.

Scikit Learn gözetimli ve gözetimsiz bir çok makine öğrenmesi algoritmasını kullanıma sunmaktadır. Fit isimli bir metod ile algoritmaya veri seti tanıtılır, eğitim aşaması gerçekleştirilir. Gözetimli makine öğrenmesi yöntemlerinde predict isimli metod kullanılır. Bu metod ile eğitilmiş olan model üzerinden test gerçekleştirilir. Bir diğer önemli yapı ise çapraz doğrulamadır.

Çapraz doğrulama (Cross validation): Veriler belli parçalara ayrıldıktan sonra bir kısmı test, diğerleri eğitim için ayrılır. Daha sonra bu gruplama her seferinde farklı olacak şekilde tekrarlanır. Böylece elde edilen sonuçların ortalaması alınarak bir sonuç üretilir.

Scikit Learn kütüphanesinin sağlamış olduğu Metrics modülünün verdiği raporlardaki değerlerin tanımlamaları aşağıdaki gibidir:

Precision(Hassasiyet): TP/(TP+FP) Pozitif tahminlerin doğruluk oranıdır. Bu değerin %100 olması tüm değerlerin doğru olarak tahmin edildiğini gösterir. Örneğin 5 rakamı için bu değerin %100 olması tüm 5'lerin doğru bilindiğini gösterir. 5 olmayan hiçbir değer için 5 tahmini yapılmaz ancak 5 olan bir değere 5 değil tahmini yapılabilir. Yani sistemin 5 dediği tüm değerler 5'tir ancak tanıyamadığı 5'ler olabilir.

Recall: TP/(TP+FN) Bu değere duyarlılık veya gerçek pozitif oran(true positive rate(TPR)) da denilir. Örneğin 5 rakamı için recall değerinin yükselmesi daha fazla 5'in doğru tahmin edildiğini ancak aynı zamanda başka rakamların da 5 olarak algılanabildiğini gösterir.

F1 Score: Bu değer precision ve recall değerlerinin harmonik ortalamasıdır.

Support: ((TP+TN)/(TP+TN+FP+FN)) Support değeri verinin ne kadar sık görüldüğünü gösteren değerdir. Verinin tüm veri seti içinde görülme oranını gösterir.

Accuracy: İsabetlilik de denilebilecek bu değer doğruluk oranını gösterir. Doğru olan tahminlerin yapılan tüm tahminlere oranı ile hesaplanır. Örneğin 5 rakamı için accuracy değeri %100 ise 5 rakamı için yapılan tüm tahminlerin doğru olduğunu gösterir.

3.2. Geliştirilen Uygulama

Python'da Scikit Learn kütüphanesi yardımıyla gerçekleştirilen işlem adımları aşağıdaki gibidir;

- Proje dosyalarına indirilmiş olan MNIST veri seti projeye dahil edilir.

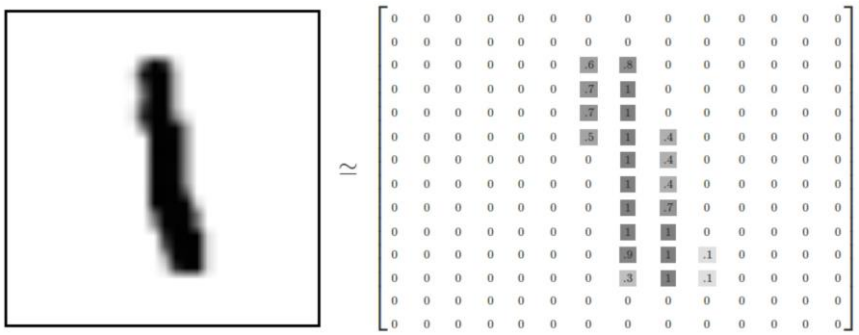
- Veri seti eğitim görselleri ile etiketleri ve test görselleri ile etiketleri şeklinde ayrılarak değişkenlerde tutulur.

- Scikit Learn kütüphanesinin yardımıyla kullanılacak algoritma tanımlanır.

- Tanımlanan algoritma Scikit Learn "fit" metoduyla MNIST veri seti eğitim verileri ile eğitilir. Eğitim aşamasında normalizasyonu gerçekleştirilmiş karakterlerin öznitelikleri ile öğrenme gerçekleştirilir. Her bir eğitim verisi temsil ettiği rakamın görselini ve görseli tanımlayan etiketi içerir. Her bir rakam 764 öz niteliğe sahiptir. Şekil 3.1.'de örnek görsel mevcuttur.

- Tanımlanan algoritma Scikit Learn "predict" metoduyla, MNIST veri seti test verileri ile test edilir. Test işleminde test görselleri algoritmaya verilir. Algoritmanın ürettiği çıktıların doğruluğu mevcut test etiketleri yardımıyla hesaplanır.

- Scikit Learn metrics modülü yardımı ile rapor oluşturulur.



Şekil 3.1. MNIST 1 rakam verisinin matris formunda gösterimi

BÖLÜM 4. ARAŞTIRMA BULGULARI

4.1. Karşılaştırılan El Yazısı Tanıma Yöntemleri

Bu bölümde karşılaştırılmak üzere kullanılacak algoritmalara detaylı örneklerle değinilmiştir. Algoritmalar belirlenirken farklı makine öğrenmesi alt kategorilerine ait algoritmalar seçilmeye çalışılmıştır. Bu şekilde daha kapsamlı bir karşılaştırma ve sonuçlar üzerinden analiz yapılabilmektedir.

Yapılan testlerde MNIST veri setinin 60.000 verisinin tamamı kullanılmıştır. Tüm algoritmalarda bu değer aynı tutulmuştur.

4.1.1. Destek vektör makinesi

DVM en kısa tanımıyla; eğitim veri setindeki verilerin ve etiketlerin ilişkilerini öğrendikten sonra bu değerlerin yardımıyla destek vektörlerini oluşturarak yeni verinin sınıflandırılmasını sağlayan makine öğrenme yöntemidir.

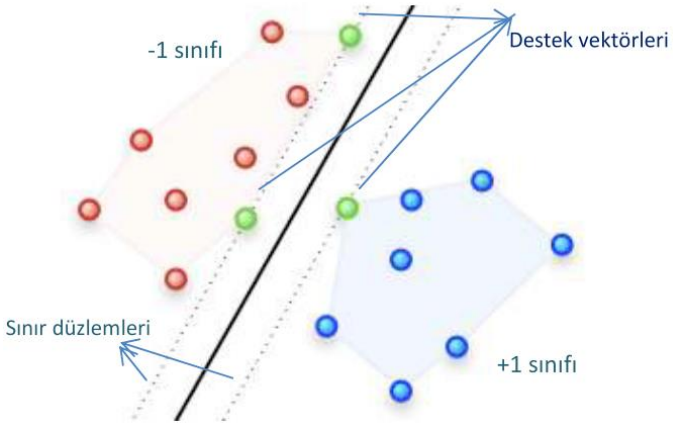
El yazısı tanıma problemi özelinde makine öğrenmesi algoritmaları incelendiğinde destek vektör makinelerinden çokça faydalanıldığı görülmektedir.

DVM'lerin sınıflandırma problemlerinin çözümünde diğer bir çok tekniğe göre başarılı sonuçlar verdiğinin kanıtlanmış olması; bu tekniğin el yazısı tanıma problemi için test edilecek algoritmalardan biri olarak belirlenmesinin sebeplerindendir. DVM'lerin bir problemi çözmeleri sürecinde çekirdek fonksiyonu seçimi önemli aşamalardandır [15].

DVM verinin doğrusal ve ya doğrusal olmaması şeklinde iki grupta incelenir [16].

4.1.1.1. Doğrusal destek vektör makinesi

Veri setinin doğrusal olarak ayrılabildiği durumlarda doğrusal DVM kullanılır [16].



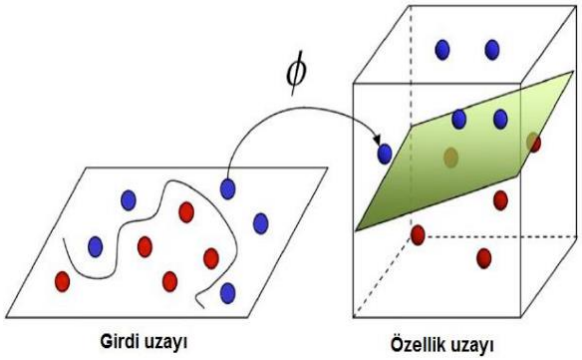
Şekil 4.1. DVM için 2 sınıflı problem örneği [15]

Şekil 4.1.'deki örnekte doğrusal bir DVM örneği görülmektedir. Görüldüğü üzere farklı sınıflara ait örnekleri birbirinden ayıran bir çok doğrusal düzlem bulunabilir. Ancak burada asıl amaç hiper düzlemi tespit etmektir. Hiper düzlem ise farklı sınıflara ait destek vektörleri arasındaki uzaklığı maksimize eden düzlemdir [15].

4.1.1.2. Doğrusal olmayan destek vektör makinesi

Doğrusal olmayan bir veri kümesinde DVM doğrusal bir hiper düzlem çizemez. Bu nedenle çekirdek fonksiyonları denilen kernel trickler kullanılır. Çekirdek fonksiyonları doğrusal olmayan DVM'lerde öğrenimi yüksek oranda arttırmaktadır.

Doğrusal olmayan DVM yönteminde çekirdek fonksiyonlarının kullanıldığı algoritma doğrusal DVM yöntemindeki ile benzerdir. Ancak her iç çarpımı doğrusal olmayan bir çekirdek fonksiyonu ile değiştirilmiştir. Böylelikle maksimum aralıklı hiper düzlem dönüştürülmüş örnek uzayına yerleşir. Bu dönüşüm doğrusal yapıda olmayabilir. Dönüştürülmüş örnek uzayı yüksek boyutlu olabilir. Şekil 4.2.'te doğrusal olarak sınıflandırılamayan girdi uzayı çekirdek fonksiyonu ile bir üst boyuta haritalanmıştır [16].



Şekil 4.2. Doğrusal olarak sınıflandırılamayan girdi uzayının bir üst boyuta çekirdek fonksiyonu ile haritalanması [16]

Doğrusal olmayan veri kümelerine sahip DVM'lerde çekirdek fonksiyonları kullanılır. Bu çalışmada değişik DVM modelleri oluşturmak ve test edebilmek amacıyla en sık kullanılan bir kaç çekirdek fonksiyonu kullanılmıştır. Tablo 4.1.'de MNIST el yazısı rakam veri seti ile yapılan testlerin sonuçları yer almaktadır. Karşılaştırılan 5 farklı çekirdek fonksiyonu arasında en yüksek verim polinomiyal çekirdek fonksiyonu ile alınmıştır.

Tablo 4.1. MNIST veri seti ile yapılan test değerleri

Çekirdek fonksiyonu	Eğitim süresi (sn)	Test süresi (sn)	Verim (%)
Polinomiyal	0.7	6.3	90
Radyal Tabanlı Fonksiyon(RBF)	0.8	7.7	89
Doğrusal (Linear)	0.4	5.8	87
Sigmoid	0.6	7.8	84

Polinomiyal çekirdeği tüm eğitim verilerinin normalleştirildiği problemler için çok uygun olduğundan söz konusu veri seti ile yapılan testlerde yüksek verim sağlaması beklenilesi bir durumdur [16].

Tablo 4.1.'deki testlerde 1000 adet test verisi kullanılmıştır. Bu testlerin verdiği sonuçların, bu alanda yapılacak bir çalışma için çekirdek fonksiyonu seçiminde yol gösterici olması beklenmektedir.

Scikit Learn kütüphanesi DVM'lerin gerçekleştirimi için ilgili algoritmalarını içeren "sklearn.svm" modülünü sunmaktadır. Bu modülün SVC(Support Vector Classification) isimli bir alt modülü bulunmaktadır. Yapısı libsvm kütüphanesine dayalı olan bu alt modül ile DVM oluşturulmuştur. Oluşturulan bu makine 3.2.2. Scikit Learn başlığında detaylandırılmış olan 'fit' metodu ile eğitilmiştir.. Daha sonra testler gerçekleştirilerek yine Scikit Learn kütüphanesinin metrics modülü ile rapor elde edilebilir. İlgili rapor Şekil 4.3.'te görülebilmektedir.

	precision	recall	f1-score	support
0	0.9789	0.9918	0.9853	980
1	0.9825	0.9912	0.9868	1135
2	0.9777	0.9777	0.9777	1032
3	0.9802	0.9782	0.9792	1010
4	0.9728	0.9817	0.9772	982
5	0.9797	0.9720	0.9758	892
6	0.9802	0.9802	0.9802	958
7	0.9736	0.9689	0.9712	1028
8	0.9733	0.9743	0.9738	974
9	0.9747	0.9564	0.9655	1009
accuracy			0.9774	10000

Şekil 4.3. Scikit Learn destek vektör makinesi raporu

Şekil 4.3.'te Scikit Learn kütüphanesinin sağlamış olduğu rapor görülmektedir. Testte polinomiyal çekirdek fonksiyonu kullanılmıştır. Raporda verilen değerlerin tanımları 3.2. Scikit Learn başlığında detaylandırılmıştır.

4.1.2. Karar ağacı

Karar ağacı yapısı; bir veri kümesini kullanarak belli kuralları ve bu kurallar çerçevesinde ve bir ağaç yapısı oluşturan sınıflandırma yöntemidir. Karar ağaçlarının oldukça güçlü olmalarının ve çokça tercih edilmelerinin sebeplerinin başında; anlaşılabilir ve basit kurallar çerçevesinde sınıflandırma gerçekleştirebilmeleri gelir.

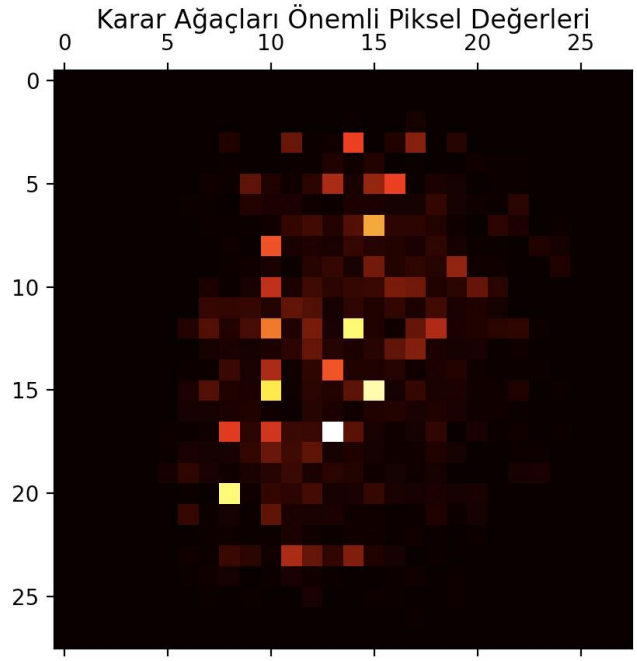
Ayrıca hesaplamaların yoğun işlemler gerektirmemesi de bu yöntemin güçlü yanlarından biridir.

Karar ağaçlarının zayıf yönlerinden biri eğitim aşamasının ve ağacın oluşturulmasının uzun sürebilmesidir. Yine ağaç yapısının getirdiği bir diğer olumsuz özellik ise küçük boyutlu eğitim verileriyle çalışıldığında hata verme oranının diğer sınıflandırma algoritmalarına göre yüksek olmasıdır [17].

Ağaç yapısı oluşturulurken en önemli aşamalardan biri ağaçtaki dallanmanın belirleneceği öz nitelik değerinin seçilmesidir. Bu problemin çözümü için literatürde mevcut olan yaklaşımlardan bazıları; Twoing Kuralı, Ki-Kare Olasılık Tablo İstatistiği, Gini İndeksi'dir [9]. Scikit Learn kütüphanesinde karar ağaçları için bölme ölçütü Gini İndeksi'dir.

Bu çalışmada dallanma faktörü olarak Gini indeksi kullanılmıştır. Testlerde Scikit Learn kütüphanesinin sağlamış olduğu karar ağacı yöntemi kullanılmıştır. Scikit Learn kütüphanesi binary ağaçlar oluşturan CART algoritmasını kullanmaktadır. Bu algoritmada her düğümün sadece iki alt düğümü veya uç yaprağı vardır. ID3 gibi farklı algoritmalar kullanılarak daha çok sayıda alt düğüme sahip ağaç yapıları oluşturulabilir [18].

Rakamların ayrılmasını sağlayan öznitelikler Scikit Learn kütüphanesinin sağlamış olduğu feature_importances özelliği ile incelenebilir. Şekil 4.4.'te 60.000 veriyle eğitilen karar ağacı yapısının piksel değerlerinin önem grafiği gösterilmiştir. Grafikte piksellerin önem değerleri kırmızıdan sarıya gidildikçe artar.



Şekil 4.4. Scikit Learn karar ağaçları önemli piksel değerleri

Karar ağaçları algoritması üzerinde çalışılırken Scikit Learn kütüphanesinin "sklearn.tree" modülünden faydalanılmıştır. Bu modül sınıflandırma ve regresyon için karar ağacı tabanlı modelleri içermektedir. Modülün "Decision Tree Classifier" isimli alt modülü kullanılmıştır. Daha sonra bu modül ile oluşturulan karar ağacı MNIST verileriyle eğitildikten sonra test edilmiştir.

	precision	recall	f1-score	support
0	0.9151	0.9347	0.9248	980
1	0.9571	0.9639	0.9605	1135
2	0.8571	0.8488	0.8530	1032
3	0.8350	0.8465	0.8407	1010
4	0.8710	0.8798	0.8754	982
5	0.8356	0.8430	0.8393	892
6	0.8930	0.8883	0.8906	958
7	0.9142	0.9018	0.9079	1028
8	0.8354	0.8183	0.8268	974
9	0.8516	0.8414	0.8465	1009
accuracy			0.8781	10000

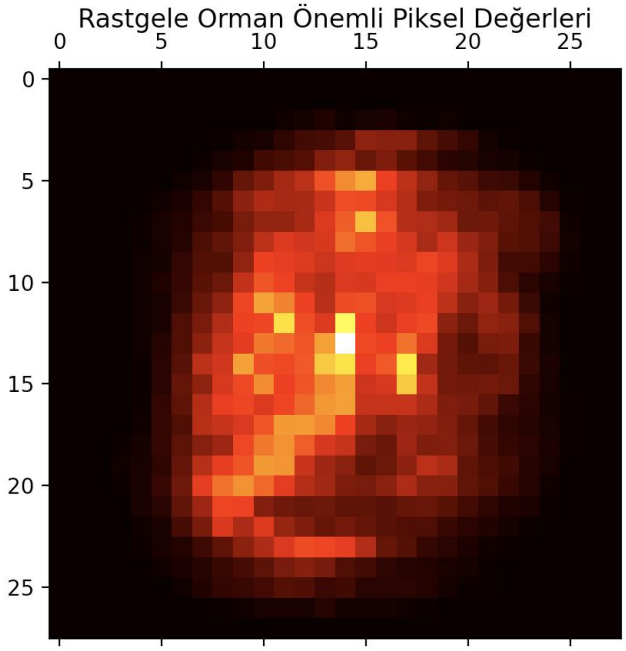
Şekil 4.5. Scikit Learn Karar Ağaçları Raporu

Şekil 4.5.'te Scikit Learn kütüphanesinin metrics modülünün sağlamış olduğu rapor görülmektedir. Raporda verilen değerlerin tanımları 3.2. Scikit Learn başlığında detaylandırılmıştır.

4.1.3. Rastgele orman

Rastgele orman algoritması karar ağaçları gibi hem sınıflandırma hem regresyon problemlerinde kullanılabilir. Çalışma mantığı birden fazla karar ağacı oluşturup bu ağaçların yardımıyla ortalama sonuçlar üretmesi şeklindedir. Bu algoritmaya rastgele denmesinin sebebi ağaç yapısının oluşturulması esnasında ekstra bir rastgelelik sunmasıdır. Bir düğümü bölerken direkt en iyi öz niteliği aramak yerine, bir rastgele öz nitelikler alt kümesinde en iyi öz niteliği arar. Bu durum daha çeşitli ağaçların oluşmasını sağlar [18].

Rakamların ayrılmasını sağlayan öznitelikler Scikit Learn kütüphanesinin sağlamış olduğu feature_importances özelliği ile incelenebilir. Şekil 4.6.'da 60.000 veriyle eğitilen rastgele orman yapısının piksel değerlerinin önem grafiği gösterilmiştir. Grafikte piksellerin önem değerleri kırmızıdan sarıya gidildikçe artar.



Şekil 4.6. Scikit Learn rastgele orman önemli piksel değerleri

Rastgele orman algoritmasını Scikit Learn ile gerçekleştirmek için “sklearn.ensemble” isimli modül bulunmaktadır. Bu modülün bir alt modülü olan “RandomForestClassifier” ile rastgele ağaç yapısı tanımlanır. Bu testte de karar ağaçlarındaki gibi dallanma faktörü olarak Gini indeksi kullanılmıştır. Ağaç sayısı 150 olarak belirlenmiştir. MNIST verileri ile yapılan test sonuçları Şekil 4.7.’de listelenmiştir.

	precision	recall	f1-score	support
0	0.9729	0.9908	0.9818	980
1	0.9894	0.9885	0.9890	1135
2	0.9607	0.9700	0.9653	1032
3	0.9710	0.9604	0.9657	1010
4	0.9706	0.9756	0.9731	982
5	0.9707	0.9664	0.9685	892
6	0.9730	0.9791	0.9761	958
7	0.9735	0.9660	0.9697	1028
8	0.9657	0.9538	0.9597	974
9	0.9573	0.9544	0.9558	1009
accuracy			0.9707	10000

Şekil 4.7. Scikit Learn rastgele orman raporu

4.1.4. Yapay sinir ağı

İnsan beyninin çalışma prensibinin taklit edilmesi üzerinde kurulmuş olan yapay sinir ağları; düşünen, öğrenen, çıkarımlar yapan bir yapıya sahiptir. İnsan beynini modellemeye çalışan bu makineler, sinir hücresi veya işlem birimi adı verilen hücreler arasındaki bağlantıyı kullanmakta ve bu işlemler esnasında öğrenme adı verilen bir süreç ile performansını arttırabilmektedir.

YSA yapısı bilgiyi saklama ve kullanma yeteneğine sahip basit işleme birimlerinden oluşan, yoğun, paralel ve dağıtılmış düzende çalışan bir işlemci şeklinde

tanımlanabilir. Bilgiyi öğrenmesi ve depolama için sinir hücreleri arası bağı kullanması insan beyni ile benzerliğini gösterir.

Doğrusal programlamadan farklı olarak YSA; sinir hücreleri arasındaki bağlantıları, eşik değerleri ve ağ yapısı gibi parametreleri kullanarak ayarlar yapıp eğitim gerçekleştirir. YSA'da bellek ve işlemci paralel olarak çalışır. Bilgi sinir hücreleri arasındaki bağlarda da saklanabilir ve değiştirilebilir [11].

YSA ile bir el yazısı karakter tanıma uygulaması tasarlanırken ilk aşama tanınması beklenen metindeki bir karakterin sinir ağı için girişe uygun şekilde bir matrisinin oluşturulmasıdır. Bu şekilde YSA eğitilebilir. Her bir karakterin matrisi için ağ çıkışları oluşturulur. Öğrenme işleminde gizli katmanlardaki nöron sayısının fazlalığı eğitim işleminin fazla zaman almasına neden olurken ağın hatırlama kabiliyetini de aynı oranda artırmaktadır.

Öğrenmeden sonra doğrulama işlemi yapılır. Bu işlem birçok yöntemle yapılabilir. Bu yöntemlerden biri 2 veya 3 harften oluşan kombinasyonları içeren veritabanları kullanmaktır. Böylece harf dizisi imkansız kombinasyonlar içermeyecektir. (örn 'fdr') Bir başka yol ise kelime sözlüğü kullanmaktır. Böylece harf dizisinin gerçek bir kelime olup olmadığı anlaşılabilir [1].

YSA testleri için Scikit Learn kütüphanesinin sinir ağları modellerini içeren "neural_network" modülü kullanılmıştır. Bu modüle ait bir alt modül olan "MLPClassifier" ile bir YSA yapısı kurulmuştur.

MLPClassifier, Multi Layer Perceptron yani çok katmanlı algılayıcılı sınıflandırma anlamına gelmektedir. DVM yada Naif Bayes gibi diğer sınıflandırma algoritmalarının aksine MLPClassifier sınıflandırma görevini gerçekleştirmek üzere bir sinir ağına sahiptir.

MLPClassifier modeli oluşturulduktan sonra MNIST verileriyle eğitilmiş ve sonrasında testler gerçekleştirilmiştir. Sonuçlar Şekil 4.8.'de listelenmiştir.

	precision	recall	f1-score	support
0	0.9653	0.9949	0.9799	980
1	0.9783	0.9938	0.9860	1135
2	0.9841	0.9612	0.9725	1032
3	0.9831	0.9802	0.9817	1010
4	0.9865	0.9684	0.9774	982
5	0.9691	0.9843	0.9766	892
6	0.9843	0.9812	0.9827	958
7	0.9852	0.9737	0.9795	1028
8	0.9733	0.9743	0.9738	974
9	0.9722	0.9693	0.9707	1009
accuracy			0.9782	10000

Şekil 4.8. Scikit learn yapay sinir ağı raporu

Şekil 4.8.'deki raporda verilen değerlerin tanımları 3.2. Scikit Learn başlığında detaylandırılmıştır.

4.1.5. K-en yakın komşu algoritması

K-en yakın komşu yöntemi denetimli öğrenme yöntemlerinden biridir. Bu yöntemde öncelikle sınıflandırma yapılacak test verisinin, eğitim verileriyle benzerlikleri hesaplanır. En yakın olduğu görülen k verinin ortalamasıyla, belirlenen eşik değere göre sınıflandırma yapılır. Yöntemin performansını k en yakın komşu sayısı, eşik değeri, benzerlik ölçümü ve öğrenme kümesindeki normal davranışların yeterli sayıda olması etkilemektedir [25].

K-en yakın komşu algoritması gerçekleştirilirken Scikit Learn kütüphanesinin neighbors modülüne ait "KNeighborsClassifier" alt modülünden faydalanılır. Testlerde MNIST veri setini eğitim ve test verisi olarak bölümlendirmek için Scikit Learn kütüphanesinin "model_selection" modülüne ait "train_test_split" sınıfı kullanılmıştır. Bu özelliği kullanarak veri seti kolaylıkla istenilen oranlarda bölünebilir. Önemli parametreler şunlardır:

test_size : Test verisi olarak belirlenecek verinin boyutunu belirleyen parametredir. Örneğin bu parametreye 0.5 değeri atanırsa tüm veri setinin %50'si test verisi olarak ayrılacak demektir. Bu parametreye değer atanması durumunda train_size parametresi boş geçilebilir.

train_size : "test_size" parametresi boş ise bu parametreye değer atanmalıdır. "test_size" parametresine benzer şekilde bu parametre de eğitim verisi olarak belirlenecek verinin boyutunu gösterir.

random_state : Bu değer integer olmalıdır. Boş bırakılırsa rastgele değer alır. Belli bir değer verilirse test ve eğitim verileri her seferinde aynı olacaktır [26].

	precision	recall	f1-score	support
0	0.9634	0.9939	0.9784	980
1	0.9545	0.9982	0.9759	1135
2	0.9822	0.9603	0.9711	1032
3	0.9644	0.9663	0.9654	1010
4	0.9762	0.9613	0.9687	982
5	0.9653	0.9664	0.9658	892
6	0.9813	0.9864	0.9839	958
7	0.9611	0.9611	0.9611	1028
8	0.9881	0.9374	0.9621	974
9	0.9563	0.9534	0.9548	1009
accuracy			0.9688	10000
macro avg	0.9693	0.9685	0.9687	10000
weighted avg	0.9690	0.9688	0.9687	10000

Şekil 4.9. K-en yakın komşu algoritması raporu

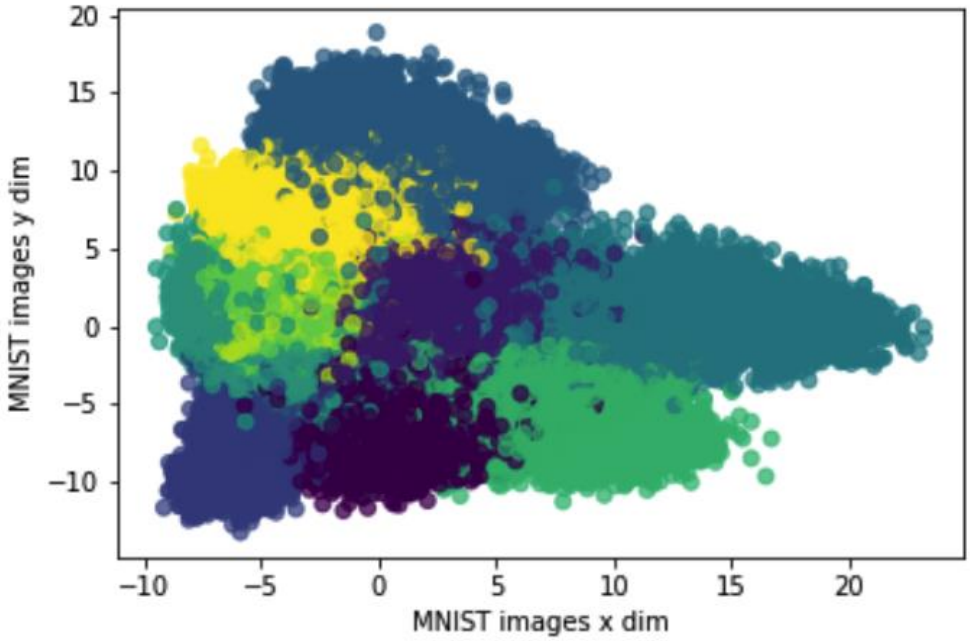
Şekil 4.9.'daki raporda verilen değerlerin tanımları 3.2. Scikit Learn başlığında detaylandırılmıştır.

4.1.6. K-ortalama algoritması

Çalışma kapsamında karşılaştırılan algoritmalar arasında gözetimsiz olan tek algoritma K-ortalama algoritmasıdır. Bu algoritmanın testleri için skelarn.cluster

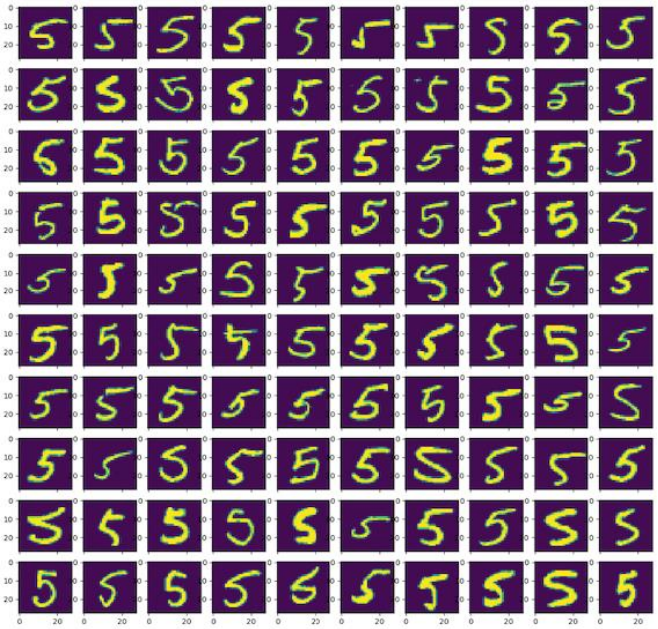
modülünün KMeans alt modülü kullanılmıştır. Veri seti olarak yine MNIST kullanılmıştır.

Çalışma sonucunda elde edilen verim %98.12 olarak ölçülmüştür. MNIST verileriyle oluşturulan kümelerin temsili gösterimi Şekil 4.10.'da gösterilmiştir.



Şekil 4.10. MNIST veri seti K-ortalama rakam kümeleri [28]

Oluşturulan kümelerden 5 rakamına ait görselleri içeren küme Şekil 4.11.'de gösterilmiştir. Diğer rakamların kümelerinde çok düşük oranda yanlış kümelenmiş görseller mevcuttur. Özellikle 4-6, 3-8 gibi birbirine benzer rakamlarda hatalar bulunmaktadır. Ancak oranı çok düşük olduğundan verimi kötü anlamda etkilememiştir.



Şekil 4.11. 5 rakamının görsellerini içeren küme

BÖLÜM 5. SONUÇLAR VE RAPOR

5.1. Algoritmaların Sonuçlarının Değerlendirilmesi

Bu çalışmada 6 farklı makine öğrenmesi algoritması ile el yazısı tanıma işlemi gerçekleştirilmiştir. Tüm testlerde MNIST el yazısı rakamlar veri seti kullanılmıştır. El yazısı tanıma problemi kendi içinde bir çok alt başlıkta incelenebilecek bir problemdir. Ön işleme, özellik çıkarımı, karakter ayırma, düzenleme gibi bir çok adımı mevcuttur. Çalışma kapsamında mevcut, üzerinde çalışılmaya elverişli bir veri tabanının bulunması durumunda rakam tanıma işlemi gerçekleştirilmiştir. Bu bağlamda elde edilen sonuçlar bir çok farklı durumdan etkilenebilir. Bu tez kapsamında el yazısı tanıma alanının spesifik, küçük bir kısmı üzerinde araştırma ve çalışmalar yapılmıştır.

Karşılaştırılan algoritmaların verimleri Tablo 5.1.'de verilmiştir. Bu çalışmanın ve tablonun hedefi; el yazısı tanıma alanında çalışacak kişilere ilerleyecekleri alanda ve seçecekleri makine öğrenmesi yönteminde yol göstermektir.

Tablo 5.1. Karşılaştırılan algoritmaların verimleri

Algoritma	Verim	Eğitim Süresi	Test Süresi
Destek Vektör Makinesi	%90	372.294 sn	97.769 sn
Karar Ağaçları	%87	21.142 sn	0.645 sn
Rastgele Orman	%97	63.055 sn	1.106 sn
Yapay Sinir Ağı	%97	178.390 sn	1.014 sn
K-En Yakın Komşu Algoritması	%96	36.396 sn	865.932 sn
K-Ortalama Algoritması	%98	8.350 sn	0.569 sn

DVM'lerin el yazısı tanıma alanında kullanımı oldukça yaygındır. İstatistiksel öğrenme yöntemini temel alan DVM'ler diğer sınıflandırma algoritmalarıyla karşılaştırıldıklarında bazı avantajlara sahiptirler. Eğitim verisinin az olduğu durumlarda da güçlü ve verimli çalışabilirler. Daha iyi sınıflandırıcılar üretme ve genelleme yeteneği de yüksektir. Bu çalışma kapsamında DVM'lerin el yazısı tanıma alanında çokça kullanıldığını ve hız, verim gibi değerlerinin de gösterdiği üzere el yazısı tanıma alanında kullanıma uygun olduğu görülmüştür.

Karar ağaçları ve rastgele orman algoritmaları aslında aynı temel üzerine kurulmuşlardır. Rastgele orman algoritmaları eğitilen karar ağaçları algoritmalarından oluşur. Bu iki algoritma da el yazısı tanıma alanında kullanıma oldukça uygundur. Karar ağaçları mevcut veri seti üzerinden çıkarımlar yapıp ağaç yapısını oluşturmaktadır. Rastgele ormanlarda ise biraz daha tahmin ve rastgelelik baskındır. Rastgele ormanlar çalışmada da görülebileceği üzere çok daha yavaş eğitilmelerine karşın daha yüksek verim sağlamaktadırlar. Karar ağaçlarında ise düşük verime karşın hızlı bir eğitim süresi söz konusudur.

YSA günümüzde el yazısı tanıma dışında da bir çok alanda kullanılmaktadır ve her geçen gün adını biraz daha duyurmaktadır. Yüksek verimiyle en iyi sonuç veren algoritmalar arasındadır. El yazısı tanıma gibi daha çok insan karar verme mekanizmasını taklit etmesi amaçlanan problemlerde iyi sonuçlar vermektedir.

K-en yakın komşu algoritması sınıflandırma alanında çokça tercih edilen temel algoritmalardandır. Basit bir yapısı olmasına karşın hızlı ve yüksek verimli sonuçlar vermiştir.

Çalışma kapsamında karşılaştırılan algoritmalar arasında gözetimsiz olan tek algoritma k-ortalama algoritmasıdır. Yapılan testlerde oldukça yüksek verim sağlamış ve çok düşük oranda hatalı kümeleme göstermiştir.

Bu tez çalışmasında sadece bir veri seti üzerinde çalışılmıştır. Farklı veri setleri üzerinde testler yapılarak algoritmaların çalışma durumları incelenebilir. Bu şekilde çalışmalar genişletilebilir ve daha net sonuçlar elde edilebilir.

5.2. Çalışmanın Faydaları

Yapılan tez çalışmasının faydaları şu şekilde sıralanabilir:

- Tez çalışmasının 4. bölümünde karşılaştırılan makine öğrenmesi algoritmalarının el yazısı rakam tanıma üzerine davranışları gözlemlenmiştir.

- El yazısı rakam tanıma alanında araştırmalar yapılmış ve bu alanda kaynak olabilecek bir çalışma hazırlanmıştır.

- Çalışma sonucunda bir tek makine öğrenmesinin diğerlerinden iyi olduğunu söylemek doğru olmayacaktır. Makine öğrenmesinin kullanılacağı uygulamanın hız, verim gibi alanlardaki beklentilerine göre en iyi algoritmanın hangisi olacağına karar verilmesi gerekmektedir.

KAYNAKLAR

[1] Şekerci, M., Birleşik ve eğik Türkçe el yazısı tanıma sistemi. Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi, 2007.

[2] Duygulu, P., El Yazısı Tanıma, Bilişim Ansiklopedisi, Papatya Yayıncılık, 2006.

[3] Bilgin Taşdemir, E. F., A large vocabulary online handwriting recognition system for Turkish, Sabancı University, Graduate School of Engineering and Natural Sciences, Doctor of Philosophy, 2018.

[4] MacKenzie, S., Tanaka-Ishii, K., Text Entry Systems, Morgan Kaufmann Yayıncılık, 123-137, 2007.

[5] Dağdeviren, E., El yazısı rakam tanıma için destek vektör makinelerinin ve yapay sinir ağlarının karşılaştırması, İstanbul Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi, 2013.

[6] Şekerci, M., Kandemir, R., Sözlük kullanarak Türkçe el yazısı tanıma, Edirne, 2006.

[7] Yılmaz, B., Öğrenme güçlüğü çeken çocuklar için el yazısı tanıma ile öğrenmeyi kolaylaştırıcı bir mobil öğrenme uygulaması tasarımı. Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi, 2014.

[8] Kiani, F., Kutlugün, M. A., Çakır, M. Y., Yapay sinir ağları ve K-en yakın komşu algoritmalarının birlikte çalışma tekniği (ensemble) ile metin türü tanıma. İstanbul, 2017.

[9] Şengür, D., Öğrencilerin akademik başarılarının veri madenciliği metotları ile tahmini. Fırat Üniversitesi, Eğitim Bilimleri Enstitüsü, Bilgisayar ve Öğretim Teknolojileri Eğitimi, Yüksek Lisans Tezi, 2013.

[10] Bektaş, B., Babur, S., Turhal, U., Köse, E., Makine öğrenmesi yardımıyla optik karakter tanıma sistemi. 5.Uluslar Arası Matbaa Teknolojileri Sempozyumu, İstanbul, 2016.

[11] Tosun, S., Sınıflandırmada yapay sinir ağları ve karar ağaçları karşılaştırması : Öğrenci başarıları üzerine bir uygulama. İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Bölümü, Yüksek Lisans Tezi, 2007.

[12] Assegie, T. A., Nair, P. S., Handwritten digits recognition with decision tree classification: a machine learning approach. International Journal of Electrical and Computer Engineering (IJECE), Endonezya, 446-4451, 2019.

[13] Sathya, R., Abraham, A., Comparison of supervised and unsupervised learning algorithms for pattern classification. International Journal of Advanced Research in Artificial Intelligence(IJARAI), 2013.

[14] Nilsson, N. J., Introduction to machine learning an early draft of a proposed textbook. Stanford, 119 , 1998.

[15] Abdalla, S., Erdoğmuş, Ş., Destek Vektör Makineleriyle Sınıflandırma Problemlerinin Çözümü İçin Çekirdek Fonksiyonu Seçimi. Eskişehir, 2014.

[16] Güldoğan, E., Çeşitli çekirdek fonksiyonları ile oluşturulan destek vektör makinesi modellerinin performanslarının incelenmesi : Bir klinik uygulama. İnönü Üniversitesi ve Mersin Üniversitesi, Biyoistatistik ve Tıp Bilişimi Anabilim Dalı Ortak Doktora Programı, Doktora Tezi, 2017.

[17] Sastry, P. N., Krishnan, R., Sanker Ram, B. V., Classification and identification of telugu handwritten characters extracted from palm leaves using decision tree approach. ARPN Journal of Engineering and Applied Sciences, 2010.

[18] Géron, A., Hands-On machine learning with Scikit-Learn and TensorFlow concepts. O'Reilly Yayıncılık, 2017.

[19] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 2011.

[20] Güneş, A., Yiğit, T., Hızlandırılmış destek vektör makineleri ile el yazısı rakamların tanınması. 20th Signal Processing and Communications Applications Conference, Muğla, 2012.

[21] Düzgün, A., Yaygın sınıflandırıcıların Scikit-Learn, Weka ve Matlab araçları ile Twitter spam tespitinde karşılaştırılması. Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi, 2019.

[22] Yazıcı, A. C., Öğüş, E., Ankaralı, S., Canan, S., Ankaralı, H., Akkuş, Z., Yapay sinir ağlarına genel bakış. Türkiye Klinikleri Tıp Bilimleri Dergisi, 2007.

[23] Özkan, İ. A., Tornalamada kesme kuvvetlerinin ve takım ucu sıcaklığının bulanıl mantık ve yapay sinir ağı teknikleriyle tahmin edilmesi. Selçuk Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Bölümü, Yüksek Lisans Tezi, 2006.

[24] Öztemel E., Yapay sinir ağları. Papatya Yayıncılık, 48 , 2012.

[25] Kırmızıgül Çalışkan, S., Soğukpınar, İ., KxKNN : K-ortalama ve K-en yakın komşu yöntemleri ile ağlarda nüfuz tespiti. Electronic Journal of Vocational Colleges(Ejovoc) , Kırklareli , 2015.

[26] www.medium.com/@contactsunny/how-to-split-your-dataset-to-train-and-test-datasets-using-scikit-learn-e7cf6eb5e0d., Erişim Tarihi : 15.07.20.

[27] Sarıman, G., Veri madenciliğinde kümeleme teknikleri üzerine bir çalışma: K-Means ve K-Medoids kümeleme algoritmalarının karşılaştırılması. Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü Dergisi, 192-202, 2011.

[28] https://github.com/Mahanteshambi/Clustering-MNIST., Erişim Tarihi: 23.07.2020.

ÖZGEÇMİŞ

Rabia Karakaya ilk, orta ve lise eğitimini Kocaeli'de tamamladı. 2012 yılında İzmit Anadolu İmam Hatip Lisesi'nden mezun oldu. 2013 yılında başladığı Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nü 2017 yılında bitirdi. 2017 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde yüksek lisans eğitimine başladı. Yüksek lisans eğitimine devam ettiği süreçte part time olarak bir e ticaret sitesinde web programlama üzerine çalıştı. Daha sonra ileri programlama eğitimleri aldı. Özel bir bankada bir süre çalıştı. Şu anda çevrimiçi yazılım eğitimi vererek evden çalışmaya devam etmektedir.