DOKUZ EYLÜL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME YAKLAŞIMIYLA FATURA GÖRÜNTÜLERİNDEN BİLGİ ÇIKARMA ALGORİTMALARI VE UYGULAMALARI

Adem AKDOĞAN

Mayıs, 2021
İZMİR

DERİN ÖĞRENME YAKLAŞIMIYLA FATURA GÖRÜNTÜLERİNDEN BİLGİ ÇIKARMA ALGORİTMALARI VE UYGULAMALARI

Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi
Bilgisayar Bilimleri Anabilim Dalı

Adem AKDOĞAN

Mayıs, 2021
İZMİR

YÜKSEK LİSANS TEZİ SINAV SONUÇ FORMU

ADEM AKDOĞAN, tarafından **DR. ÖĞR. ÜYESİ RESMİYE NASİBOĞLU** yönetiminde hazırlanan **"DERİN ÖĞRENME YAKLAŞIMIYLA FATURA GÖRÜNTÜLERİNDEN BİLGİ ÇIKARMA ALGORİTMALARI VE UYGULAMALARI"** başlıklı tez tarafımızdan okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

.....

Dr. Öğr. Üyesi Resmiye NASİBOĞLU

Yönetici

.....

Jüri Üyesi

.....

Jüri Üyesi

Prof.Dr. Özgür ÖZÇELİK

Müdür

Fen Bilimleri Enstitüsü

TEŞEKKÜR

Tez çalışmam süresince değerli bilgi ve önerileriyle beni sürekli destekleyen, karşılaştığım problemlerde bana her daim yardımcı olan tez danışmanım Dr. Ögr. Üyesi Resmiye Nasiboğlu'na,

Yapay zeka konusunda engin bilgi ve tecrübesiyle bana yol gösteren, fikirleriyle ufkumu açan, tezimin her aşamasında emeği olan değerli hocam Prof. Dr. Efendi Nasiboğlu'na,

Çalıştığım süre boyunca her çıkmaza girdiğimde yaratıcı fikirleriyle yolumu aydınlatan, yüksek lisansı bitirme sürecinde her türlü yardımı yapmaktan kaçınmayan Sayın Mustafa Özal'a,

Teknik konulardaki gelişimimde yardımlarını benden hiçbir zaman esirgemeyen ve Türkiye'nin en iyi yazılımcılarından birisi olan değerli müdürümüz Ömer Akbaş'a,

Her zaman olduğu gibi yüksek lisans eğitimimde de benden desteklerini ve anlayışlarını esirgemeyen annem Hatice Akdoğan'a, babam Şaban Akdoğan'a, kardeşlerim Merve Akdoğan ve Rana Akdoğan'a,

Sonsuz minnet ve teşekkürlerimi sunarım.

Adem AKDOĞAN

DERİN ÖĞRENME YAKLAŞIMIYLA FATURA GÖRÜNTÜLERİNDEN BİLGİ ÇIKARMA ALGORİTMALARI VE UYGULAMALARI

ÖZ

Fatura, fiş, dekont ve benzeri şirket dokümanlarının dijitalleştirilmesi, gelişen teknoloji ile yaygınlaşmıştır. Fiziksel evrakların muhafaza ve idare işlemleri dijital evraklara göre oldukça zor olduğu için şirketler evraklarını dijital ortama taşımaya başlamışlardır. Ayrıca muhasebe işlemlerinin yoğun olduğu dönemlerde klasik yöntemler ile faturaların işlenmesi yoğun insan gücü gerektiren, maliyetli bir sürece neden olmaktadır. Dijitalleşme süreciyle birlikte fatura işleme yöntemleri de makineler yardımıyla gelişmeye başlamıştır.

Bu tez kapsamında yapay zeka teknikleri kullanılarak faturaların hızlı, doğru ve daha az kaynak ile işlenme süreci ele alınmıştır. Öncelikle görüntü formatındaki faturalar, belirli görüntü işleme süreçlerinden sonra optik karakter tanıma motoru olan Tesseract ile yazı formuna çevrilir. Çoklu N-gram yapıları kullanılarak tahmin edilmek istenen etiketli verilerin oluşturulması sağlanır. Oluşturulan N-gramların öznitelikleri belirlenir. Bu aşamada eğitim için önemli olan kelime uzaklıklarını belirleme algoritmalarından Levenshtein ve Jaro-Winkler yöntemleri karşılaştırılmalı olarak değerlendirilir. Son olarak eğitim yaptırılarak elde edilen sonuçlar değerlendirilir. Eğitimde Rassal Orman (Random Forest), Gradyan Yükseltme Makinesi (Gradient Boosting Machine), Aşırı Gradyan Artırma (Extreme Gradient Boosting), K-En Yakın Komşu (K-Nearest Neighbors), AdaBoost, Karar Ağacı (Decision Tree) modelleri kullanılmıştır. Derin öğrenme kısmında ise evrişimsel sinir ağları kullanılmıştır. Toplamda 9910 adet fatura ile eğitim ve test işlemleri gerçekleştirilmiştir. Eğitimler neticesinde fatura numarası için 0,97, fatura tarihi için 0,97, ödeme tarihi için 0,88, teslimat tarihi için 0,76, toplam tutar için 0,93, net tutar için 0,89, vergi tutarı için 0,92, IBAN için 0,99 ve vergi numarası için 0,99 olarak F1 skor değerleri elde edilmiştir.

Anahtar Kelimeler: Bilgi çıkarımı, N-gram, optik karakter tanıma, Levenshtein uzaklığı, Jaro-Winkler uzaklığı, makine öğrenmesi

INFORMATION EXTRACTION ALGORITHMS AND APPLICATIONS FROM THE INVOICE IMAGES USING A DEEP LEARNING APPROACH

ABSTRACT

The digitization of invoices, receipts and similar company documents has become very common with the rapid development in technology. Since the preservation and management processes of physical documents are much more difficult than digital documents, companies have started to digitalize their documents. In addition, the processing of invoices using classical methods is a costly process that requires great manpower during periods of intensive accounting. With the digitalization process, invoice processing methods have also started to develop with the help of machines.

In this wok, how to process invoices faster, more accurately and with less resources by using artificial intelligence techniques is discussed. First of all, invoices in image format are converted to text with Tesseract that is an optical character recognition engine. The labeled data is created by using multiple N-gram structures. The attributes of the created N-grams are determined. At this stage, Levenshtein and Jaro-Winkler methods, which are important algorithms for determining word distances are evaluated comparatively. Finally, the results obtained by training are evaluated. The models were used in the training such as Random Forest, Gradient Boosting Machine, Extreme Gradient Boosting, K-Nearest Neighbors, AdaBoost and Decision Tree. The convolutional neural networks are used as a deep learning model. The training and testing carried out with a total of 9910 invoices. As a result of the trainings, the F1 score values were obtained such as 0.97 for the invoice number, 0.97 for the invoice date, 0.88 for the due date, 0.76 for the delivery date, 0.93 for the total amount, 0.89 for the net amount, 0.92 for the vat amount, 0.99 for the IBAN and 0.99 for the sender vat number.

Keywords: Information extraction, N-gram, optical character recognition, Levenshtein distance, Jaro-Winkler distance, machine learning

İÇİNDEKİLER

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

TABLOLAR LİSTESİ

Sayfa

BÖLÜM 1
GİRİŞ

Günümüzde hemen her alanda yapay zeka destekli sistemler kullanılmaya başlanmıştır. Sanal asistanlar, oyunlar, akıllı arabalar, doküman işleme, akıllı ev sistemleri, sağlık ve bankacılık sektörleri gibi birçok alanda yapay zekanın bu sistemlere entegrasyonu yapılmaktadır. Bu tez kapsamında dokümanlardan makine öğrenmesi ve derin öğrenme teknikleri kullanılarak bilgi çıkarılması üzerine çalışmalar yapılmıştır. Endüstri 4.0 ile birlikte özellikle büyük şirketlerin fatura, dekont, fiş ve benzeri şirket dokümanlarının dijitalleştirilmesi büyük önem arz etmektedir. Bunun yanı sıra birçok muhasebe firması da ellerindeki fiziksel dokümanları dijital ortama aktarıp bunların hem işlenme kolaylığını hem güvenliğini hem de ulaşılabilirliğini artırmayı hedeflemektedir. Özellikle bu süreçle birlikte faturalardan daha hızlı ve daha etkili şekilde bilgilerin nasıl elde edilebileceği üzerine yoğun araştırmalar yapılmıştır. Dokümanlardaki bilgilerin elde edilmesi ve yönetilmesi insan gücüne bağlı olan geleneksel yöntemlerle yapılması oldukça yoğun emek isteyen, pahalı bir süreçtir (Klein, Agne, ve Dengel, 2004). Bunun yanı sıra bu dokümanların manuel bir şekilde işlenmesi zaman olarak da ciddi bir kayıp oluşturmaktadır. Özellikle fatura gibi sürekli işlenmesi ve yönetilmesi gereken evraklar, bu süreç içerisindeki en kritik dokümanlardır.

Faturalardan bilgi çıkarımı için geçmiş yıllarda araştırmalar ve çalışmalar yapılmıştır. Burada faturaların değişken yapısı istenilen sonuçları elde etmeyi oldukça zorlaştırmıştır. Özellikle birçok şirketin farklı tarzda fatura kullandığı göz önüne alındığında, faturaların çok değişken bir yapıya sahip olduğunu rahatlıkla söyleyebiliriz. Belirli şablonlar için elimizde güçlü tahmin yöntemleri ve teknikleri olmasına rağmen şablonu belirli olmayan dokümanlarda bu klasik yöntemler işe yaramamaktadır.

Bu çalışmada faturalardan fatura numarası, fatura tarihi, teslimat tarihi, fatura tutarı, vergi tutarı, IBAN ve varsa vergi numarası gibi birçok spesifik bilginin makine öğrenmesi ve derin öğrenme teknikleri kullanılarak, herhangi bir şablona ihtiyaç

duyulmadan elde edilmesi amaçlanmıştır. Modelimizin eğitim ve test aşamaları için toplamda 9910 adet fatura kullanılmıştır. Hedeflenen bilgilerin tahmin edilmesi için genel bir eğitim modeli tasarlamak yerine her bir etiket için farklı eğitim modeli kullanılmıştır. Böylelikle tahmin edilmek istenen her bir etiketi en iyi şekilde yansıtan öznitelikler eğitime dahil edilirken diğerleri göz ardı edilebilir. Bu da eğitimin daha başarılı geçmesini sağlamıştır. Eğitimde makine öğrenmesi modeli olarak Rassal Orman (Random Forest), Gradyan Yükseltme Makinesi (Gradient Boosting Machine), Aşırı Gradyan Yükseltme (Extreme Gradient Boosting), K-En Yakın Komşu (K-Nearest Neighbors), AdaBoost, Karar Ağacı (Decision Tree) modelleri kullanılmıştır. Ayrıca kelimelerin birbirlerine olan uzaklıklarının ölçülebilmesi için hem Levenshtein hem de Jaro-Winkler değerleri hesaplanmıştır. Derin öğrenme kısmında ise evrişimsel sinir ağları kullanılmıştır.

BÖLÜM 2
LİTERATÜR TARAMASI

Görüntü formatındaki dokümanlardan bilgi elde edilmesi üzerine makine öğrenmesi, derin öğrenme gibi farklı yapay zeka teknikleri kullanılarak çalışmalar yapılmıştır. Bunun yanı sıra belirli şablonlara ait dokümanlar için kural bazlı çözüm yöntemleri de geliştirilmiştir. Yapılan bu çalışmalar ayrıntılı olarak aşağıda incelenebilir.

2.1 Kural Bazlı Çalışmalar

Çeşitli kurallara dayalı birçok çalışma literatürde mevcuttur. Farklı amaçlar için yapılan bu çalışmaların güçlü olduğu farklı yönler olmasına rağmen kural bazlı çalıştıkları için geniş çaplı çözümler sunamamaktadırlar.

Automatic Indexing isimli dokümantasyon okuma projesinde bilgi çıkarım işlemi şablonlar yardımıyla gerçekleştirilir (Esser, Schuster, Muthmann, Berger ve Schill, 2012). Şirketler genellikle önceden belirledikleri şablonları doldurarak fatura oluştururlar. Bu sayede her şirket kendi içinde tutarlı bir yapı oluşturmuş olur. Sisteme gelen her yeni doküman grafiksel olarak taranır. Kendisine en yakın şablona göre verileri indekslenir. Bu indeksleme sonucunda elde edilmek istenen bilgiler (tarih, tutar gibi) elde edilir. Yanlış şablon belirlenmesi ve elde edilen verilerin yanlış olması sonucunda geri dönüş (feedback) yapılarak şablon seçimi tekrarlanır. Şablon tanıma sırasında sistem, veri seti içerisindeki her bir şablona ait dokümanları tarayarak alt kümeler oluşturur. Şablonu belirlenmek istenen doküman için bu alt kümeler taranarak en çok benzediği dokümanlar tespit edilir. İşlemlerin hızlı olması için indeksleme ve tarama aşamalarında Apache Lucene kullanılmıştır. Benzer belgeleri bulmak için Lucene veya benzeri metin arama motorları kullanılarak basit bir şekilde sorgular gerçekleştirilir. Yapılan sorgular neticesinde benzer belgelerin benzerlik skorları yüksek olur. Burada sadece kelime benzerlikleri yeterli değildir. Belgelerin aynı şablona ait olup olmadıklarının tespit edilmesi için benzer kelimelerin doküman üzerindeki pozisyon bilgileri de kontrol edilir. Bu doğrultuda bilgi çıkarımı işlemleri

gerçekleştirilir. Yeni doküman sisteme girdi olarak verildikten sonra yukarıda anlatılan şablon tanıma ve bilgi çıkarımı işlemleri yapılır. Doğru sonuçlar elde edilebilse de faturalardaki bilgilerin pozisyonlarının değişmesi sonuçları olumsuz şekilde etkilemektedir. Dolayısıyla sadece izin verilen şablonlardaki faturalar iyi sonuçlar verebilmektedir.

Intellix isimli çalışmada da Automatic Indexing gibi faturaların şablonlarını temel alarak bilgi çıkarımı işlemi gerçekleştirilir (Schuster ve diğer., 2013). Öncelikle OCR (Optical Character Recognition) ile faturalardaki kelimeler ve bu kelimelere ait pozisyon bilgileri elde edilir. Sonrasında farklı şablonlara sahip olan faturalar görsel yapılarına, logolarına ve ayırt edici bazı kelimelere göre sınıflandırılır. Bu çalışmada kullanılan veri seti için en iyi sınıflandırma K-en yakın komşular (KNN) algoritması ile yapılmıştır. Bir önceki çalışmada olduğu gibi indeksleme ve tarama işlemleri için Lucene kullanılmıştır. Fatura şablonlarının benzerliklerinin ve farklılıklarının tespit edilebilmesi için ilk olarak fatura ızgaralarla eş parçalara bölünür. Sonrasında her bir kelimenin koordinat bilgilerine göre kelimeleri bölünmüş ızgaralara olan mesafeleri tespit edilir. Bu mesafelere ve mesafe oranlarına göre (faturaların kayma durumlarında mesafeler değişir ancak oranlarında bir değişiklik meydana gelmez) fatura şablonları bulunur. Her bir indeksleyici bileşen girdi olarak dokümanın kendisini ve şablon eşleşmesi için gereken listeyi alır. Çıktı olarak ise tespit edilmek istenen spesifik etikete ait veriler ve bu verileri ait olasılık değerleri yer alır. Tespit edilmek istenen 10 farklı alanın (tarih, tutar, müşteri numarası vb.) ortalama başarısı %85 olarak elde edilmiştir.

Kural bazlı bir diğer çalışma ise Receipts2Go isimli bilgi çıkarım projesidir (Janssen, Saund, Bier, Wall ve Sprague, 2012). Bu çalışmada makbuzlar, etiketler, biletler, faturalar ve fişler gibi birçok doküman kullanılır. Bu sistemi kullanabilmek için dokümanlar tek sayfa olmak zorundadır. Çalışma görüntü normalleştirme, metin çıkarımı, varlık çıkarımı, varlık anlama ve bilgi dağıtımı olmak üzere 5 ana kısımdan meydana gelmektedir. Veri kaynağı olarak cep telefonlarından çekilen resimler kullanılmaktadır. Ancak telefonlarla çeşitli çözünürlüklerde resimler çekildiği için elde edilen veriler farklı genişlik-boy değerlerine sahip olacaktır. Bu sebeple

normalizasyon işlemleri gerçekleştirilir. Görüntülerin arka planları temizlenir ve görüntü olabildiğince kirliliklerden arındırılır. Bu işlemler sonrasında oluşturulan görüntüden optik karakter tanıma (OCR) yardımı ile yazılar elde edilir. Sonrasında varlık çıkarımı için Regex kullanılarak bazı spesifik ifadeler (Toplam tutar, ödeme tarihi, vb.) elde edilmeye çalışır. Bu aşamada bulanık eşleme yöntemi kullanılır. Sonrasında sisteme giren dokümanların sınıflandırılması için bir dizi kural uygulanır. Sınıflandırılma işlemi tamamlandıktan sonra dokümanların dahil olduğu sınıfın varlık özelliklerine göre bilgi çıkarımı gerçekleştirilir. Sistemde kuralları belli olan dokümanlar için iyi sonuçlar elde edilirken, sisteme daha önce girmemiş doküman modellerinde çok sağlıklı çalışmamaktadır. Ayrıca bu çalışmada yalnızca tek sayfalık dokümanların kuralları tanımlandığı için birden fazla sayfaya sahip dokümanların bilgi çıkarım işlemi yapılamamaktadır.

2.2 Yapay Zeka Destekli Çalışmalar

Kural bazlı çalışmalar spesifik problemleri çözebilmesine rağmen geniş ölçekli çalışmalarda yetersiz kalabilmektedirler. Özellikle hem yapay zeka teknolojilerinin ilerlemesi hem de bu teknolojiler için ihtiyaç duyulan donanım özelliklerinin gelişmesiyle birlikte yapay zeka destekli bilgi çıkarım çalışmalarına başlanmıştır.

CloudScan çalışmasında Automatic Indexing ve Intellix çalışmalarından farklı olarak belirli kurallara dayalı bir model yerine LSTM (Uzun-Kısa Vadeli Hafıza) modeli kullanılarak amaçlanan bilgiler elde edilmiştir (Palm, Winther ve Laws, 2017). Bu çalışmada 2 farklı model kullanılmıştır. İlk modelde lojistik regresyon sınıflandırıcı ile tahminler yapılmıştır. İkinci model de ise LSTM ağırlıklı bir tahmin algoritması kullanılır. İlk olarak OCR ile görüntü formatındaki belgelerin bilgileri elde edilir. Elde edilen bilgiler N-gramlar şeklinde düzenlenir. Elde edilen her bir N-gramın uzunluğu, sayfadaki konumu, sağındaki ve solundaki kelimelere olan uzaklıkları gibi öznitelik bilgileri çıkartılır. N-gramların tamamının şablonları çıkartılır. Öngörülebilir OCR hatalarından kaynaklı problemler (“100,0o” gibi) regex işlemleri ile düzeltilir. Etiketlere değerler atamak için Macar algoritması kullanılmıştır. Temel işlemlerden sonra Lojistik Regresyon Sınıflandırıcısı ile tahmin işlemi gerçekleştirilir. LSTM

kısmında ise etiketleme IOB sistemine göre gerçekleştirilir. Ön işlemler tamamlandıktan sonra eğitim gerçekleştirilir. Sonuç olarak tahmin edilecek olan etiket için en yüksek olasılık değerine sahip olan N-gram seçilir. Bu çalışmada 326471 adet fatura kullanılmıştır. Eğitimde kullanılan faturalara benzer faturalar kullanılarak test işlemi yapıldığında ilk modelde 0,887, ikinci modelde ise 0,891 F1 skor değeri elde edilmiştir. Eğitimde kullanılan faturalardan tamamen farklı faturalar kullanıldığında ise ilk modelde 0,788, ikinci modelde ise 0,840 F1 skor değerine ulaşılmıştır.

Chargrid isimli bir çalışma, dokümanların 2 boyutlu yapılarının anlaşılması üzerine yapılmıştır (Katti ve diğer., 2018). Geleneksel dil işleme modellerinde doküman üzerindeki bilgiler 1 boyutlu düz yazı formatı haline getirilerek işlenir. Eğer işlenecek olan dokümanlar kitap, makale, dergi gibi düz yazılarsa bu durumda dokümanın 1 boyuta indirgenmesi sorun teşkil etmeyecektir. Ancak işlenecek belgelerin fatura dekont gibi yapısal şirket dokümanları olması halinde yazıların tek boyuta indirgenmesinde bilgi kaybı meydana gelecektir. Kelimelerin anlamları korunsa da bu kelimelerin yükseklik, genişlik, sayfa üzerindeki koordinatları gibi bilgiler kaybolacaktır. Dokümanların 2 boyutlu yapılarının korunması için kelime ve karakter vektörlerine koordinat bilgilerinin de eklenmesi gerekmektedir. Bu çalışmada vektörleştirme işlemi kelime odaklı değil, karakter odaklı gerçekleştirilmiştir. Eldeki bütün veri seti incelenerek kullanılan karakterlerin tamamı çıkartılır, sonrasında bu karakterlere indeksleme işlemi yapılır. Bunun yanı sıra karakterlerin her bir yükseklik, genişlik, sayfanın üstüne ve soluna olan mesafe bilgileri de eklenir. Böylece doküman üzerindeki her bir kelimenin her karakterinin hem indeks değeri hem de lokasyon bilgileri bulunmuş olur. Bu karakterlerin modele entegrasyonu için tekil sıcak kodlama tekniği kullanılır (One Hot Encoding). Eğitim için tasarlanan model bir adet kodlayıcıdan (encoder), anlamsal bölümleme ve sınırlayıcı kutu regresyonu (bounding box regression) olmak üzere iki adet kod çözücüden (decoder) meydana gelmektedir. Çalışmada fatura numarası, fatura tarihi, fatura tutarı, satıcı ismi ve satıcı adresi olmak üzere 5 ana bölüm tahmin edilmek istenmiştir. Eğitim ve test işlemleri için toplamda yaklaşık olarak 12 bin fatura kullanılmıştır. Fatura numarası için %80,48, fatura tutarı için %80,74, fatura tarihi için %83,78, satıcı ismi için %36,00, satıcı adresi için de %39,13 doğruluk oranları elde edilmiştir.

Liu, Wan ve Zhang (2016), çalışmalarında pdf formatındaki faturalardan makine öğrenmesi tekniklerini kullanarak bilgi çıkarım işlemini gerçekleştirmişlerdir. Öncelikle OCR işlemlerinin daha iyi sonuç verebilmesi için bazı görüntü optimizasyon işlemleri yapılır. Eldeki görüntülerin eğikliği tespit edilir ve belirlenen açıya göre görüntüler, optimum hizalama için döndürülür. OCR taraması sonucunda görüntülerden bilgiler elde edilir. Böylelikle eğik fatura görüntülerinden bile en iyi OCR performansı elde edilir. Sonrasında bu bilgilerde kök bulma (stemming) işlemi gerçekleştirilir. Akabinde etkisiz kelimelerin (stop words) temizliği yapılır. Çeşitli özellikler baz alınarak öznitelik seçiminin tamamlanmasının ardında eğitim aşamasına geçilir. Bu alanda Naive Bayes, Lojistik Regresyon ve Destek Vektör Makinesi (SVM) metotları kullanılmıştır. Toplamda 7 adet olmak üzere fatura numarası, fatura tutarı, tarih vb. etiketler tahmin edilmiştir. Çalışmada modellerin başarıları ölçüldüğünde Naive Bayes modelinin eğitim hatası %16,17 ve test hatası %16,43, Lojistik Regresyon modelinin eğitim hatası %10,95 ve test hatası %14,53, SVM modelinin eğitim hatası %8,81 ve test hatası %13,99 değerlerine ulaşılmıştır.

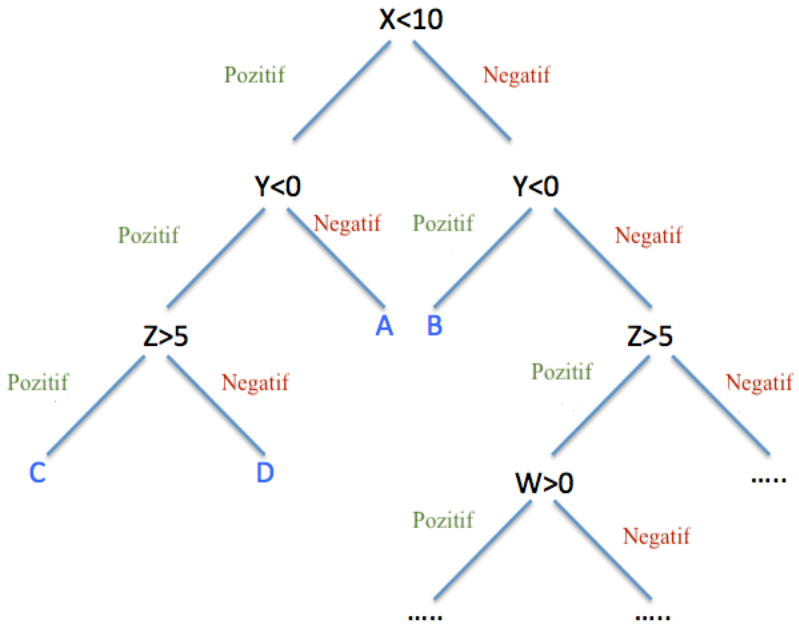
BÖLÜM 3
YAPAY ÖĞRENME MODELLERİ

Bu tez çalışmasında, etiketlerin tahmin edilmesinde kullanılan yapay öğrenme (makine öğrenimi) modelleri olarak Rassal Orman, Gradyan Yükseltme Makinesi, Aşırı Gradyan Yükseltme, K-En Yakın Komşu, AdaBoost ve Karar Ağacı modelleri ele alınmıştır. Bu modellerin her biri verilerin farklı dağılımlarında iyi sonuçlar verebilmektedir. Hem bu dağılımı kavrayabilmek hem de en iyi tahmin modelini belirlemek için bu yapay öğrenme modellerinin tamamı ile eğitim gerçekleştirilmiştir. Bu bölümde, kullandığımız modeller hakkında bilgiler verilmiştir.

3.1 Karar Ağaçları

Eğitim için ana model olarak Rassal Orman (Random Forest) modeli kullanılmıştır. Bu model, Karar Ağaçı (Decision Tree) yapısına dayanan bir modeldir (Quinlan, 1986). Şekil 3.1'de örnek bir karar ağacı yapısı görülmektedir. Örnekte görülen karar ağacında A-B-C-D sonuçları için farklı X-Y-Z koşullarının sağlanması gerekmektedir. Örnek olarak $X<10$ koşulu sağlanırken $Y<0$ koşulu sağlanmazsa A durumu meydana gelirken, $X<10$, $Y<0$ ve $Z>5$ koşullarının tamamı sağlandığında ise C durumu meydana gelmektedir.

Karar Ağaçlarının temelinde yatan fikir, eldeki bütün öznitelik değerlerini en verimli şekilde kullanarak karar yapısını oluşturmaktır (Mashat, Fouad, Yu ve Gharib, 2012). Bu sayede çok hızlı bir şekilde yüksek başarı oranları elde edilebilmektedir. Bunun için en çok kullanılan algoritmalardan birisi C4.5 algoritmasıdır (Xiaoliang, Jian, Hongcan ve Shangzhuo, 2009). C4.5 algoritmasında karar düğümlerinde kullanılan özniteliklerin seçiminde "bilgi kazancı" esas alınır. Bilgi kazancının belirlenmesinde entropi (belirsizlik düzeyi) kavramı kullanılmaktadır. Bu kavramların hesaplanması için aşağıdaki hesaplamalar kullanılmaktadır.



Şekil 3.1 Örnek karar ağacı yapısı (Cavaioni, 2017)

p_i : i. sınıftan olan etiket sayısının toplam etiket sayısına oranı,

D_j : Genel veri setinde belli bir özniteliğin sadece j. değerini içeren alt veri seti,

$info(D)$: Veri setindeki toplam entropi değeri,

$info_A(D)$: Her bir "A" özniteliğinin farklı değerlerine göre ayrıştırma sonrası toplam entropi değeri,

$Gain(A)$: A özniteliği için sağlanan kazanç,

olmak üzere;

$$info(D) = -\sum_{i=1}^{m} p_i \,.\, log p_i \tag{3.1}$$

$$info_A(D) = -\sum_{j=1}^{v} \frac{|\ D_j\ |}{|\ D\ |} . info(D_j) \tag{3.2}$$

$$Gain(A) = info(D) - info_A(D) \tag{3.3}$$

Belli bir özniteliğe göre bilgi kazancını hesaplamak için öncelikle Denklem 3.1 ile sistemin toplam entropi değeri hesaplanır. Daha sonra sistemdeki her bir özniteliğe ait entropi değeri ayrı ayrı Denklem 3.2 ile hesaplanır. Daha sonra Denklem 3.3 ile o öznitelik değerinin sisteme sağladığı bilgi kazancı elde edilir. Hesaplamalar tamamlandığında her bir öznitelik için hesaplanmış olan kazanç değerleri kıyaslanır ve en yüksek değeri sağlayan öznitelik, karar düğümü olarak belirlenir. Bu düğümler birleşerek optimal karar ağacı yapısı oluşturulur.

Karar Ağaçlarını oluşturabilmek için kullanılan bir diğer yöntem ise IBM tarafından geliştirilmiş olan Gini index yöntemidir (Gelfand, Ravishankar ve Delp, 1991). Bu yöntemde de hesaplamalar yukarıda gösterilen formüller kullanılarak, benzer şekilde yapılmaktadır. Sadece bilgi miktarı, entropiye dayanan Denklem 3.1 yerine Denklem 3.4 ile belirlenen Gini indeksi yardımıyla hesaplanmaktadır.

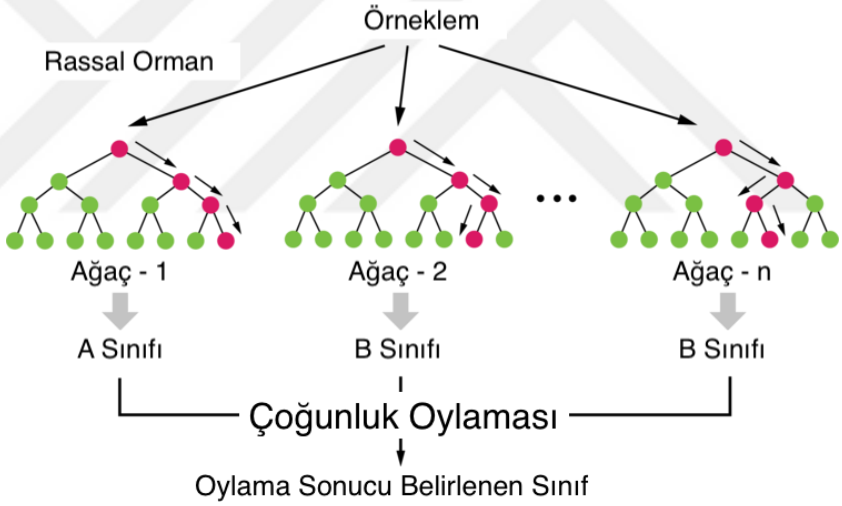
$$gini(D) = 1 - \sum_{i=1}^{m} p_i^2 \tag{3.4}$$

Makine öğreniminde en büyük problemlerinden birisi "overfitting" denen, sistemin aşırı eğitilmesi sonucu esnekliğini kaybederek ezber yapması durumudur. Karar ağacı, model eğitiminde sistem detaylarını dikkate alan özel yapı oluşturmaktadır. Modelin aşırı eğitilerek ezber yapması sonucu bu detaylara aykırı (outlier) gözlemler de dahil olabilir. Örnek olarak karar ağacında gerekli elemeler yapıldıktan sonra 100.000 verinin 99.990 tanesi pozitif değerini döndürürken kalan 10 tanesi negatif değerini döndürür. Sistemin ezber yapması ile aykırı olan bu 10 değer için ayrı bir dal oluşturulabilir ve bu katmanda kendi arasında dallara ayrılabilir. Bu da sistemin optimum sayıda dal oluşturmasını kötü yönde etkiler.

3.2 Rassal Orman

Kullanılan modellerden biri de hem regresyon hem de sınıflandırma problemleri için kullanılan ve temelini Karar Ağacı modelinden alan Rassal Orman modelidir (Breiman, 2001). Geleneksel yöntemlerden birisi olan karar ağaçlarının en büyük

problemlerinden birisi sistemin ezber yapmasıdır. Rassal Orman, karar ağaçlarının bu problemine çözüm oluşturmaktadır. Rassal Orman yöntemi, birbirinden farklı karar ağaçlarının rasgele olarak bir araya gelmesiyle oluşmaktadır. Rassal Orman'ı meydana getiren karar ağaçları oluşturulurken, yaprak düğümlerindeki sınıfların homojenliği başarının belirlenmesinde oldukça önemli rol oynamaktadır. Bu sebeple en uygun sınıflandırmayı yapabilmek için Karar Ağaçları konusunda anlatılan teknikler uygulanmaktadır. Örnek olarak karar ağaçlarını oluşturmada Gini indeks yönteminin kullanılacağını varsayarsak, (Denklem 3.4) $gini(D)$ indeksinin değeri sınıfın homojenliğini belirler. Düşük olan $gini(D)$ değeri sınıfın homojen bir yapıda olduğunu belirtirken, bu değerin yükselmesi sınıfların homojenliğinin bozulduğu anlamına gelmektedir (Menze ve diğer., 2009). Belirlenen Gini indeksine göre karar ağaçlarının yapısı belirlenerek model oluşturulur.



Şekil 3.2 Örnek Rassal Orman yapısı (Bailey, 2018)

Şekil 3.2'de N sayıdaki Karar Ağacı birleşerek Rassal Ormanı nasıl oluşturulduğu görülebilir. Yapılacak tahminlere göre genel bir tahmin değeri elde edilir. Böylece modelin nihai tahmini ortaya çıkar. Modelde her bir ağacın farklı dereceden ağırlıkları vardır. Ağırlıkları belirleyebilmek için Out-Of-Bag (OOB) hata oranı belirleme yaklaşımı kullanılır. Bu yöntemde veri seti, 2/3'ü eğitim ve 1/3'ü test olacak şekilde bölünür. İşlem sonunda en düşük hataya sahip olan ağaç en yüksek ağırlığa sahip olurken, en yüksek hataya sahip olan ağaç en düşük ağırlığa sahip olur. Sonuç olarak

ormandaki her bir ağaç ağırlıkları oranında bir tahminde bulunur. Çoğunluğa sahip olan tahmin değeri modelin genel tahmin değeri olarak belirlenir.

3.3 K-En Yakın Komşu Metodu

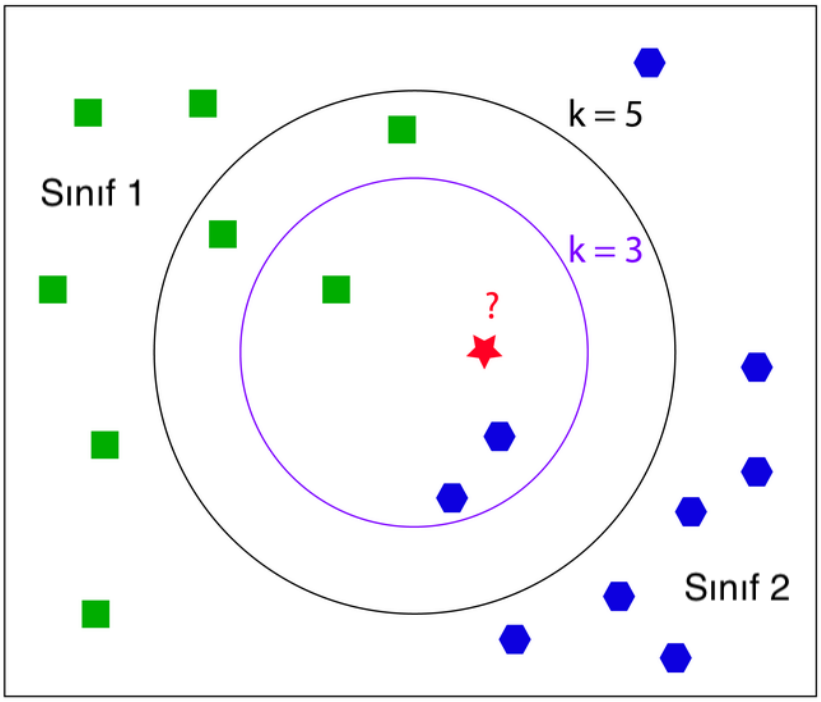
K-En Yakın Komşular algoritması (KNN) genellikle sınıflandırma problemlerinde kullanılmasına rağmen aynı zamanda regresyon problemlerinde de oldukça iyi sonuçlar vermektedir (Guo, Wang, Bell, Bi ve Greer, 2003). KNN algoritması, tahmin edilecek örnek verinin k sayıdaki en yakın komşularına olan uzaklıkları hesaplar. Sınıflandırılmak istenen örnek, bu hesap sonucunda belirlenen en yakın komşuların ait oldukları sınıfların çoğunluk durumuna göre belirlenir. Bu uzaklıkları hesaplamak için Denklem 3.5 ile gösterilen Euclidean, Denklem 3.6 ile gösterilen Manhattan ve Denklem 3.7 ile gösterilen Minkowski yöntemleri gibi farklı uzaklıklar kullanılabilir:

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \tag{3.5}$$

$$\sum_{i=1}^{k} | x_i - y_i | \tag{3.6}$$

$$\left(\sum_{i=1}^{k}(| x_i - y_i |)^q\right)^{1/q} \tag{3.7}$$

Farklı veri dağılımlarında farklı uzaklık hesaplama metrikleri optimum sonuçlar verir. Bu sebeple her veri dağılımı için en iyi sonucu verecek olan yöntemi belirlemek gerekir. Bu çalışma kapsamında Denklem 3.5 ile gösterilen Euclidean mesafesine dayalı hesap yöntemi kullanılmıştır.



Şekil 3.3 K-En Yakın Komşu metoduna göre sınıfı belirlenecek yeni eleman (Tang, 2017)

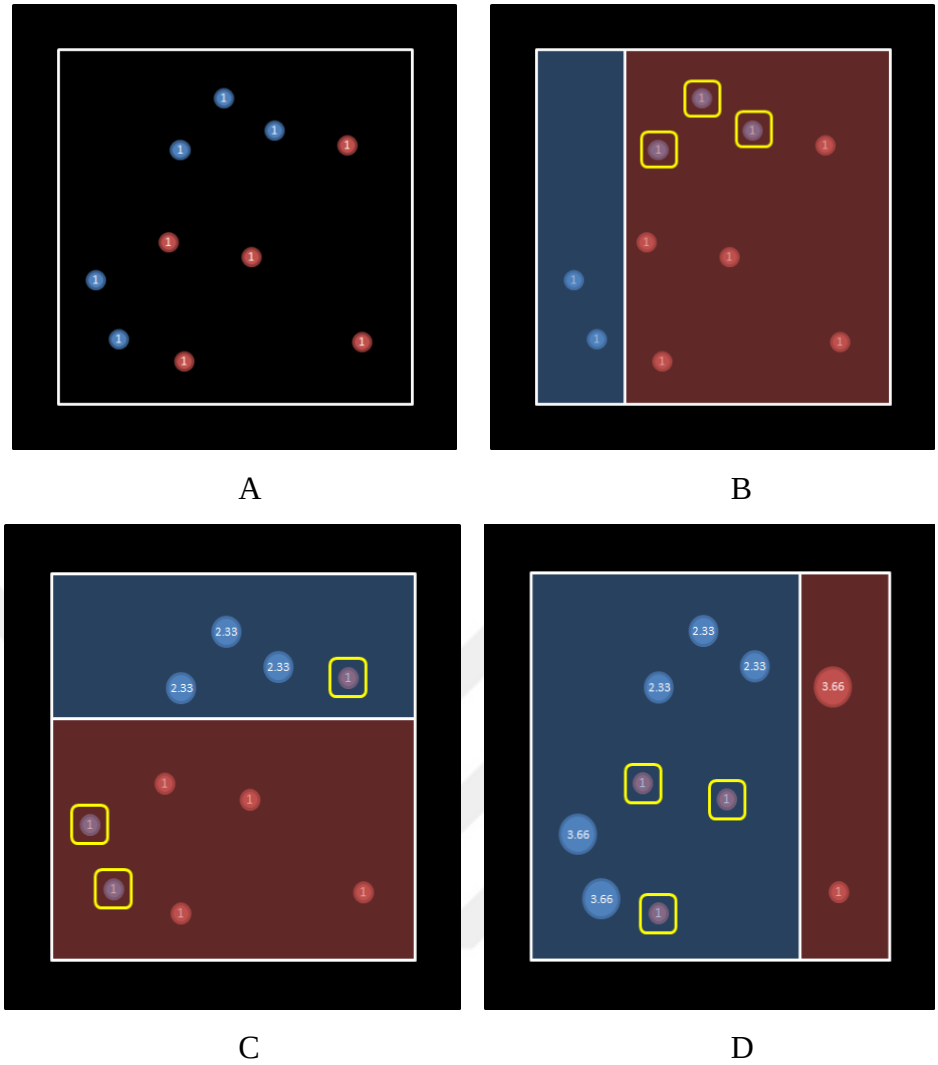
Bu algoritmanın en önemli parametrelerinden birisi k en yakın komşu sayısıdır. Şekil 3.3'teki örnekte görüldüğü üzere elimizde yeşil renkli kare elemanların dahil olduğu 1.Sınıf ve mavi renkli altıgen elemanların dahil olduğu 2.Sınıf ve sınıfını tahmin etmek istediğim kırmızı renkli yıldız elemanı vardır. Bu örnekte k parametresi 3 olarak seçilip en yakın komşularına bakıldığında 2 adet mavi ve 1 adet yeşil eleman olduğu görülür. Bu durumda kırmızı renkli eleman 2.Sınıf'a dahil olur. Ancak k parametresi 5 olarak belirlendiğine, en yakın komşularından 3 tanesi yeşil, 2 tanesi mavi olacaktır. Bu durumda da kırmızı eleman 1.Sınıf'a dahil olması gerekmektedir. Bu sebeple yüksek başarı elde etmek için eldeki veriler doğru analiz edilerek, k parametresini en doğru şekilde belirlemek gerekir.

KNN algoritması özellikle gürültü içeren verilere karşı çok etkili bir makine öğrenmesi modeli olmasına rağmen her yeni eklenen örnekte bu uzaklıkların yeniden hesaplanması büyük veriler için problem oluşturabilmektedir.

3.4 AdaBoost

AdaBoost algoritması 1995 yılında Robert Schapire ve Yoav Freund tarafından geliştirilmiştir. AdaBoost, her iterasyonda elde edilen tekil ve zayıf tahminleyicilerin birleştirilmesi sonucu güçlü bir tahminleyici elde edebilen sınıflandırıcı yöntemidir. Her iterasyonda oluşturulan tahminleyicilere katsayılar atanır. Bu adımlarda zayıf tahminleyicilerin ağırlıkları artırılır. İterasyonların tamamlanmasıyla birlikte birleştirilecek tahminleyiciler elde edilmiş olur (Kégl, 2009).

AdaBoost algoritması Şekil. 3.4'te görülen örnek üzerinden incelenecek olursa, öncelikle bütün noktaların ağırlıkları ilk iterasyonda eşit olacaktır. "B" durumunda sarı ile işaretlenen bölgede görüldüğü üzere 3 adet yanlış tahmin mevcuttur. Sistem doğru ve yanlış değerleri dengeleyebilmek adına (7/3 = 2,33) katsayı düzenlemesi yapar. Bu işlemden sonra bir diğer iterasyona geçiş yapılır. "C" durumunda görüldüğü üzere yine 3 adet yanlış tahmin mevcuttur. Bir önceki iterasyonda yapılan düzenleme işlemi burada da gerçekleştirilir. Doğru ve yanlış dengesinin kurulabilmesi için tekrar yanlış olan değerlerin katsayıları ((2,33+2,33+2,33+1+1+1+1) /3 = 3,66) düzenlenir. Sonrasında aynı işlemler "D" durumu için de yapılarak devam eder. İşlemler sonucunda belirlenen zayıf tahminleyiciler birleştirilerek nihai tahminleyici model elde edilir. Farklı veri yapıları için farklı iterasyon sayıları optimum sonucun elde edilmesini sağlayacaktır. Bu sebeple her yeni veri seti öncelikle analiz edilmelidir. Sonrasında uygun olabileceği öngörülen iterasyon aralığı belirlenmeli ve o aralık içerisinde optimizasyon çalışmaları yapılmalıdır.



Şekil 3.4 Örnek AdaBoost iterasyonu (Medium, 2019)

3.5 Gradyan Yükseltme Algoritması (GBM)

Yükseltme (boosting) işlemlerinde, tahminler Rassal Orman modelinde olduğu gibi bağımsız olarak gerçekleşmez. Bu yapılarda tahminler sıralı bir yapıda gerçekleşir. Her adımda tahmin edilen değer, kendinden bir önce yapılan tahminin hataları temel alınarak yapılır. Bu sebeple her yeni adımda gerçek değerlere daha yakın tahminler yapılır (Friedman, 2001).

y_i : Gerçek değer

y_{pred} : Tahmin edilen değer

a : Öğrenme derecesi (Learning rate)

$$y_i = y_{pred} - a \cdot 2 \sum (y_i - y_{pred}) \tag{3.8}$$

Yapılan tahminlerdeki hata değerlerini belirlemek için ortalama hata kareler (MSE) yönteminin kullanıldığı varsayıldığında, yeni tahminleri güncellemek için Denklem 3.8 elde edilir. Bu denkleme göre artıkların toplam değeri 0 veya 0'a çok yakın bir değer olduğunda modelde herhangi bir güncelleme gerçekleşmeyecektir.

3.6 Aşırı Gradyan Yükseltme (XGBoost)

Aşırı Gradyan Yükseltme metodu, temelini Gradient Boosting Machine (GBM) algoritmasından almakla beraber bu algoritmaya göre ciddi farklılıklara sahiptir. Bu farklılıklardan bir tanesi işlemlerini paralel şekilde yapabilmesidir. Klasik GBM modelinde tahminleme işlemi gerçekleştirilirken o anda yapılacak tahmin için ağaç oluşturulur. Tahmin aşaması tamamlandıktan sonra hata oranı belirlenir. Bu belirlenen hataya göre yeni yapılacak tahmin için tekrardan ağaç oluşturulur. Ancak XGBoost modelinde bir ağaç oluşumu yapılırken eş zamanlı bir şekilde özniteliklerin farklı koşullarına göre dallanmalar gerçekleştirilebilir. Bu yapı sayesinde birbirinden bağımsız sonuçlar üretecek aynı özniteliğin farklı durumlardaki değerleri için paralel işlemler gerçekleştirilebilir. Bu sebeple XGBoost daha hızlı bir yapıya sahiptir (Chen ve Guestrin, 2016).

Aşırı Gradyan Yükseltme modeli Out-Of-Core optimizasyonu ile büyük veriler için de çözümler üretebilmektedir. Ayrıca model hem Lasso hem de Ridge regularizasyon tekniklerini kullanarak ezberleme durumunu için önlem üretebilmektedir. Bunların yanı sıra model, kendi içerisinde çapraz doğrulama (cross validation) işlemi yapabilmektedir. Böylelikle harici bir çapraz doğrulama işlemine ihtiyaç duyulmamaktadır.

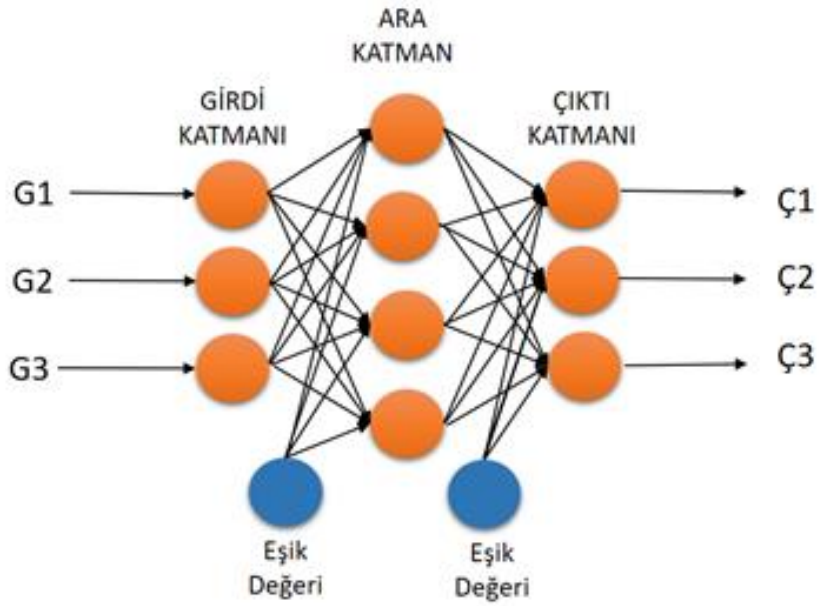
BÖLÜM 4
YAPAY SİNİR AĞLARI VE DERİN ÖĞRENME

4.1 Yapay Sinir Ağları

Basit bir yapay sinir ağı Şekil 4.1'de de görülebileceği gibi temel olarak giriş katmanı (input layer), gizli katman (hidden layer) ve çıkış katmanı (output layer) olmak üzere 3 bölümden meydana gelmektedir (Chollet, 2017).

İlk katman olan giriş katmanında veriler sisteme aktarılır. Bu katmanda kullanılan düğüm sayısı eldeki veriyi optimum şekilde temsil eden öznitelik sayısına eşittir. Giriş katmanındaki değerler gizli katmana aktarılır.

Gizli katman veya ara katman olarak da nitelendirilen bu katmanda giriş katmanından gelen değerlere belirli işlemler uygulanır. Başlangıçta bu işlemlerde kullanılan katsayı değerleri rastgele bir şekilde oluşturulur. Bazı durumlarda tamamen rastgele oluşturulmaz önceden belirlenen koşullara bağlı şekilde ortalama değerler atanarak ilk katsayı üretimine başlanır. Eğitim aşamasında bu değerler sürekli olarak güncellenir. Nihayetinde eldeki verilerle eğitim belirli bir başarıda tamamlanarak katsayılarını günceller. Bu katsayıların optimum şekilde elde edilebilmesinde eğitimin kaç adımda tamamlanacağı (epoch), öğrenme derecesi (learning rate) gibi birçok parametre etkilidir. Eldeki verilerin durumuna göre bu parametreleri doğru şekilde belirlemek, modelin başarısını oldukça yükseltmektedir. Genellikle gelişmiş modellerde birden fazla gizli katman kullanılmakla beraber bu katmanlarda da birden fazla düğüm kullanılır. Gelen değerlerin belirlenen katsayılarla işleme girmesi sonucunda elde edilen yeni değerler, belirli aktivasyon fonksiyonlarıyla işleme girer. Bu işlemler sonucunda elde edilen değerler bir diğer gizli katmana veya aktarılacak bir gizli katman yoksa çıkış katmanına aktarılır.



Şekil 4.1 Yapay sinir ağı modeli (Öztemel, 2012)

Çıkış katmanında ise bu katmanda bulunan her bir düğüme gelen verilere belirli toparlama (agregasyon) işlemleri uygulanır. Sonrasında Bölüm 4.3'te bahsedilen aktivasyon fonksiyonlarından ihtiyaca yönelik olan tespit edilerek kullanılır ve sonuç değerleri elde edilir. Sınıflandırma problemlerinde bu katmandaki düğüm sayısı tahmin edilecek olan etiket sayısına eşittir.

4.2 Derin Öğrenme

Bu başlık altında çalışmada kullanılan derin öğrenme modellerinin nasıl oluşturulduğu, eğitimin nasıl gerçekleştirildiği ve elde edilen sonuçların nasıl analiz edildiği konuları işlenmiştir. Elbette bunun için öncelikle derin öğrenmenin ne demek olduğu ve hangi bileşenlerden meydana geldiği bilinmelidir.

Başlangıçta yapay sinir ağlarının oluşturulmasındaki temel amaç, insandaki sinir ağlarına benzer bir yapıyı matematiksel olarak dizayn ederek, insan gibi düşünebilen makineler oluşturmaktı (McCulloch ve Pitts, 1943). İnsandaki sinir hücrelerinin çalışma prensipleri nöron aktivitelerinin incelenmesiyle anlaşılmıştır. Genel olarak nöronlar aksiyon potansiyeli adı verilen sinyalleri dendritleri vasıtasıyla alır. Soma

üzerinde aksiyon potansiyelinin şiddetine göre akson tepeciğinde bazı kimyasal değişimler meydana gelir. Aksonlarla bu değişimler yeni nöronlara aktarılır. Kısacası nöronlar aldıkları sinyalleri şiddetlerine göre değerlendirip yeni nöronlara aktaran yapılardır. Yapay sinir ağlarında da durum bundan pek de farklı değildir. Girdi olarak gelen değerler belirli ağırlıklarla çarpılır. Elde edilen değerler belirlenen fonksiyona gönderilir. Genellikle burada toplama fonksiyonu kullanılarak gelen bilgilerin tamamı toplanır. Ancak burada gelen değerlerin ortalamasını, maksimumunu veya minimumunu döndüren fonksiyonlar da mevcuttur. Fonksiyondan elde edilen sonuç belirli bir sınır değeri geçerse sinyal, katman içindeki ilgili nörondan tetiklenir aksi durumda ise tetikleme işlemi gerçekleşmez. Burada bahsedilen sınır değerlerin belirlenmesi aktivasyon fonksiyonları ile gerçekleştirilir. Seçilen aktivasyon fonksiyonuna göre çıkış değeri elde edilir. Basit bir nöronun çalışma mantığı bu şekilde ilerler. Literatürde nöron yerine sıkça düğüm (node) ifadesi kullanılmasından dolayı çalışmanın geri kalan bölümlerinde kullanım bu şekilde olacaktır.

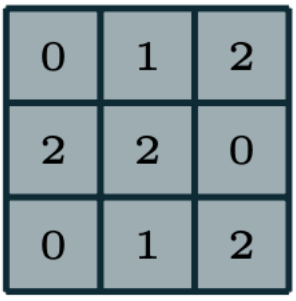
4.2.1 Evrişimsel Sinir Ağları

Derin öğrenme derken akla ilk gelen model Evrişimsel Sinir Ağı (Convolutional Neural Network-CNN) modelidir. Bu model özellikle görüntü tanıma problemlerinde sıklıkla kullanılan bir sinir ağı türüdür. Google ve Facebook gibi büyük şirketler başta olmak üzere birçok şirket tarafından kullanılmaktadır. Görüntü formatındaki bir veri gerekli ön işleme aşamalarından geçtikten sonra sisteme girdi olarak verildiğinde bu resim matrise dönüştürülür. Görüntü formatındaki verinin her bir piksel değeri bu matrisin elemanlarını temsil eder. Dolayısıyla matrisin boyutu girdi olarak gelen görüntünün boyutlarına eşit olacaktır. Evrişimsel sinir ağları geleneksel yapay sinir ağlarından farklı olarak bütün bir yapıyı eğitimde doğrudan kullanmak yerine farklılıklara odaklanarak, bu bölümleri yalnızca eğitimde kullanır (LeChun ve Bengio, 1995). Bir matrisin bütün elemanları kullanmak yerine farklılığa sebep olan elemanları kullanarak eğitim gerçekleştirmek, eğitim süresini çok büyük ölçüde kısaltacaktır. Büyük oranda görüntü tanıma ve sınıflandırma problemlerinde sıklıkla kullanılan evrişimsel sinir ağları günümüzde doğal dil işleme gibi farklı alanlarda da kullanılmaya başlanmıştır.

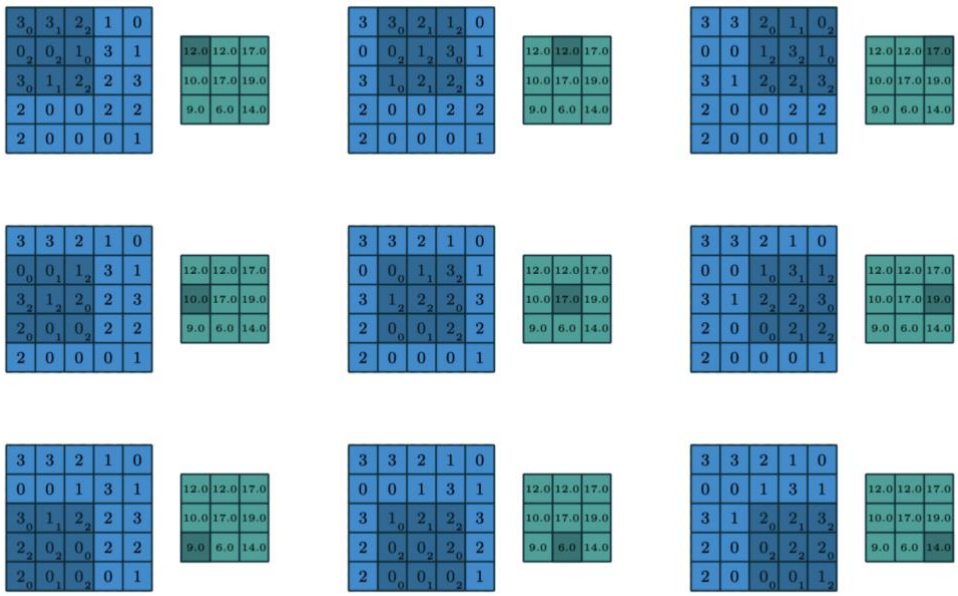
Evrişimsel sinir ağı yapısı evrişim ve havuzlama katmanından meydana gelir. Farklı kullanımlar mevcut olmakla beraber genel olarak bu yapıdan çıktı olarak gelen veriler düzleştirilip tam bağlaşımlı katmana aktarılır. Temel anlamda işleyiş bu şekilde olsa da buradaki katman sayıları, katmanların sırası, katmanlarda düğüm sayıları, kullanılan aktivasyon fonksiyonları gibi birçok parametreden dolayı modeller birbirlerinden çok farklı problemlere optimum çözümler sunabilmektedir.

4.2.1.1 Evrişim Katmanı

Evrişim katmanında, matris olarak gelen verilere bir filtre aracılığıyla evrişim işlemi uygulanır. Belirlenen filtreye göre, gelen matris başka bir matrise dönüştürülür. Burada elde edilecek olan yeni matrise öznitelik matrisi ismi verilir (Albawi, Mohammed ve Al-Zawi, 2017). Bu matriste, veri üzerindeki farklılıkların ve değişim noktalarının değeri diğer bölgelere göre daha fazladır. Özellikle görüntü formatındaki bir veri için bu durum analiz edildiğinde literatürde de sıkça kullanıldığı gibi kenar tanıma işlevini de görebilmektedir. Çünkü görüntülerde sabit olarak devam eden bir durumun değişim bölgeleri incelendiğinde bu değişimin gerçekleştiği sınır o durum için bir kenar oluşturur. Ancak özellikle son zamanlardaki çalışmalarda da görülebileceği üzerine evrişimsel sinir ağları yalnızca görüntüler için değil bu çalışmada da olduğu gibi birçok farklı dil işleme projelerinde de kullanılır. Dolayısıyla bu durumu basit bir şekilde kenar tanıma katmanı olarak değerlendirmek yerine farklılıklara odaklanan, matris üzerindeki değişimleri daha çok öne çıkartan bir filtreleme katmanı olarak görmek daha doğru olacaktır.



Şekil 4.2 Evrişim katmanında kullanılacak örnek bir filtre (Abinesh, 2020)



Şekil 4.3 Evrişim katmanında gerçekleşen matris işlemleri (Abinesh, 2020)

Şekil 4.2'de görülen 3×3 boyutunda bir filtre matrisi ile Şekil 4.3'te mavi ile görülen 5×5 boyutundaki ana matris işleme girer. Burada gerçekleşen işlem Şekil 4.3'te görüldüğü gibi filtre matrisinin ana matris üzerinde her adımda kayarak ilerlemesiyle gerçekleşir. İlk adımda gerçekleşen işlemi ayrıntılı olarak incelenirse

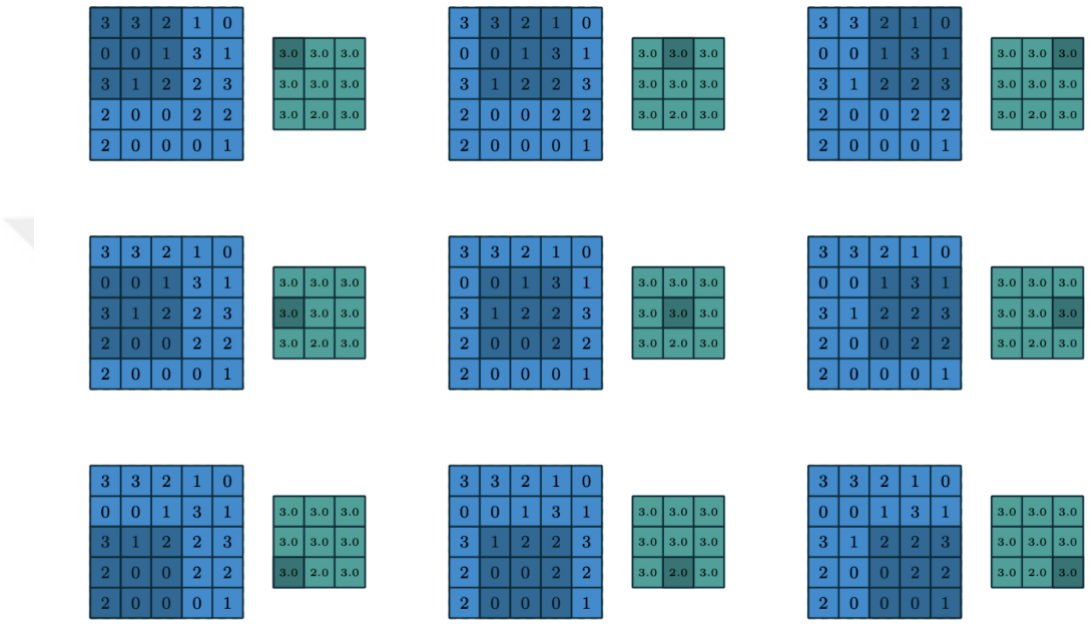
$(3 \times 0) + (3 \times 1) + (2 \times 2) + (0 \times 2) + (0 \times 2) + (1 \times 0) + (3 \times 0) + (1 \times 1) + (2 \times 2) = 12$

işlemleri sonucunda 12 değeri elde edilir. Bu değer Şekil 4.3'te yeşil renkle gösterilen öznitelik matrisinin ilk elemanı olarak matrise işlenir. Filtre matrisinin ilerleyerek her adımda ana matrisin farklı bölümleriyle işleme girmesi sonucunda öznitelik matrisinin kalan diğer elemanları da elde edilmiş olur. Ana matrisin boyutu $N \times N$, filtre matrisinin boyutu $F \times F$ olmak üzere yukarıdaki işlemler sonucunda oluşacak öznitelik matrisinin boyutu $(N - F + 1) \times (N - F + 1)$ olur.

4.2.1.2 Ortaklama (Pooling) Katmanı

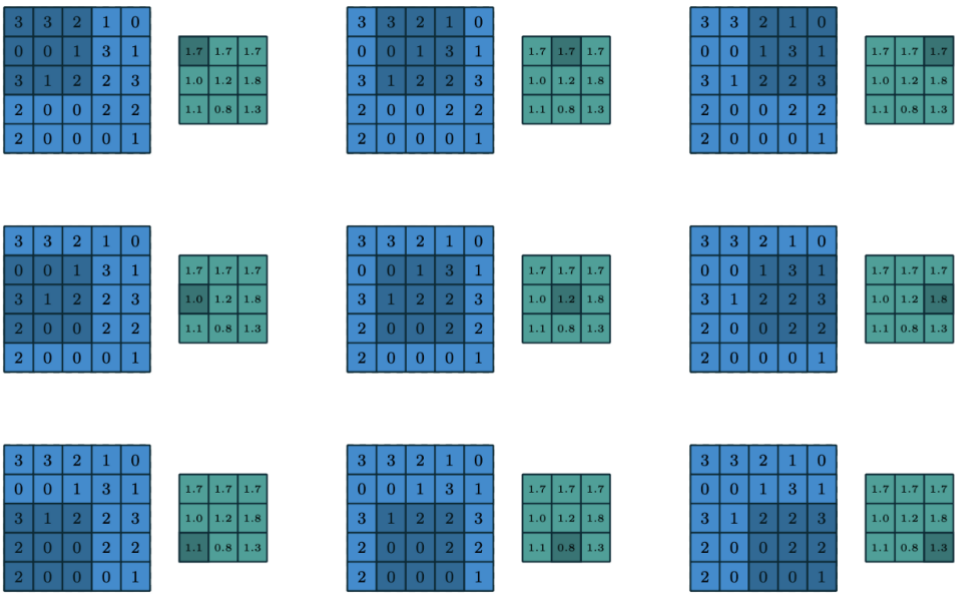
Ortaklama (pooling) katmanında girdi olarak gelen verilerin boyutlarını azaltılma işlemi yapılır. Evrişim katmanında kullanılan her bir filtre için ayrı öznitelik matrisi

oluşacaktır. Özellikle hem veri boyutu arttığında hem de evrişim katmanında kullanılacak filtre sayıları arttığında özellikle hataların geri yayılımı (backpropagation) aşamasında durum karmaşık bir hal alabilir. Bu problemin önüne geçebilmek adına toparlama katmanı kullanılmaktadır (Krizhevsky, Sutskever ve Hinton, 2012).



Şekil 4.4 Maksimum ortaklama işlemi örneği (Abinesh, 2020)

Ortaklama katmanı için öncelikle pencere boyutu belirlenmelidir. Belirlenen pencere, girdi matrisi üzerinde ilerletilerek her adımda belirli bir toparlama işlemi yapılır. Farklı problemler için, belirlenen pencere içinde kalan değerlerin maksimumlarını, minimumlarını, ortalamalarını veya toplamlarını almak gibi birçok ortaklama yöntemi mevcuttur. Literatürde genellikle maksimum ve ortalama değerin alındığı toparlama katmanı kullanılmaktadır (Scherer, Müller ve Behnke, 2010). Bu çalışma kapsamında da maksimum toparlama katmanı kullanılmıştır. Şekil 4.4'te görülen örnek maksimum toparlama işleminde 5x5 boyutundaki ana matris için pencere boyutu 3x3 olarak belirlenmiştir. Her bir adımda pencere içerisinde kalan değerlerin maksimumları alınır. Sonuç olarak maksimum değerlerden oluşan 3x3 boyutunda ortaklama matrisi oluşur.

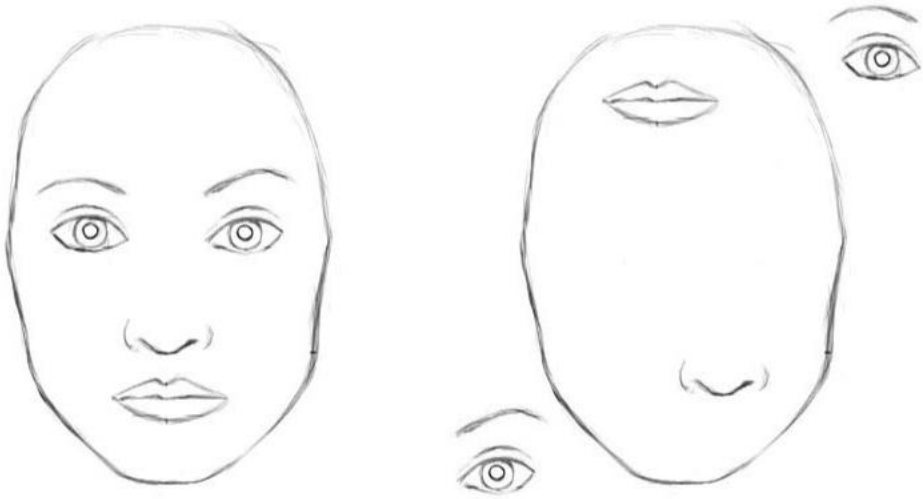


Şekil 4.5 Ortalama ortaklama işlemi örneği (Abinesh, 2020)

Sıklıkla kullanılan bir diğer ortaklama yöntemi olan ortalama ortaklama işleminde ise maksimum değerler yerine ortalama değerler kullanılır. Şekil 4.5'te her bir adımda girdi matrisi üzerindeki pencere içerisinde kalan değerlerin ortalamalarının alınarak 3×3 boyutlu ortaklama matrisinin oluşturulma aşamaları incelenebilir. Ortaklama matrisinin boyutu: o, girdi matrisinin boyutu: i, girdi matrisi üzerinde işlem yapılacak pencere boyutu: k ve adım sayısı: s olmak üzere ortaklama matrisinin boyutu Denklem 4.1 ile elde edilir.

$$o = \frac{i - k}{s} + 1 \tag{4.1}$$

Ortaklama katmanı genel olarak evrişimsel sinir ağlarında sıklıkla kullanılmaktadır. Ancak bu katmanda gerçekleştirilen ortaklama işlemlerinde bazı bilgi kayıpları da oluşmaktadır. Aynı zamanda klasik evrişimsel sinir ağlarında objelerin konumu, açısal değerleri ve pozisyon bilgileri gibi örnekleme parametreleri elde edilememektedir.

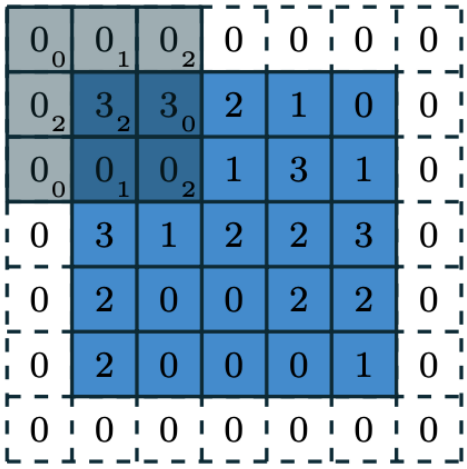


Şekil 4.6 Klasik evrişimsel sinir ağlarının problem oluşturduğu örnek durum (Sharenoesis., 2010)

Bazı bilgilerin kaybolması ve örnekleme parametrelerinin elde edilememesi gibi durumlar söz konusu olduğunda bazı problemler meydana gelmektedir. Şekil 4.6'da her iki şekli de klasik evrişimsel sinir ağları, insan yüzü olarak tanımaktadır. Bu tarz sorunlarla karşılaşıldığında kapsül ağları ve dinamik yönlendirme algoritmasını kullanmak daha doğru bir yaklaşım olacaktır (Sabour, Frosst ve Hinton, 2017). Çalışmamızda bu duruma benzer bir problem olmadığı için klasik evrişimsel sinir ağları kullanılmıştır.

4.2.1.3 Hiperparametreler

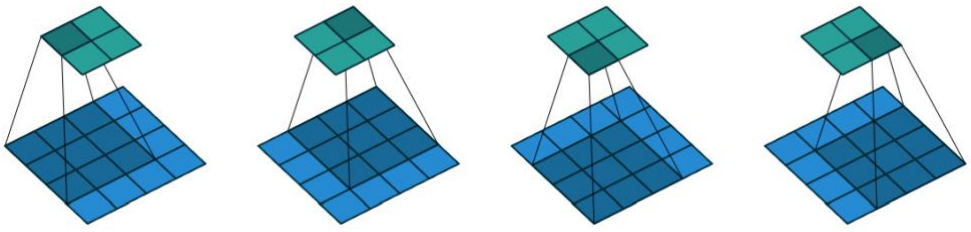
Evrişim katmanında kullanılan filtre sayısı birden fazla olabilir. Buradaki optimum filtre sayısı girdi olarak gelen veri üzerinde odaklanılmak istenilen özellik sayısına eşit olmalıdır. Girdi verisinin görüntü olduğu varsayılırsa, hem kenarların öne çıktığı bir filtre kullanılabilir hem de eğer bu görüntü üzerinde kirlilikle mevcut ise bulanıklaştırma işlevi görecek ilave bir filtre de kullanılabilir. Verinin türüne ve eğitimin amacına bağlı olarak belirlenmesi gereken filtre sayısı en önemli parametrelerden bir tanesidir (Krishnakumari, Elango ve Radhakrishnan, 2020).



Şekil 4.7 Örnek bir dolgu işlemi (Dumoulin ve Visin, 2016)

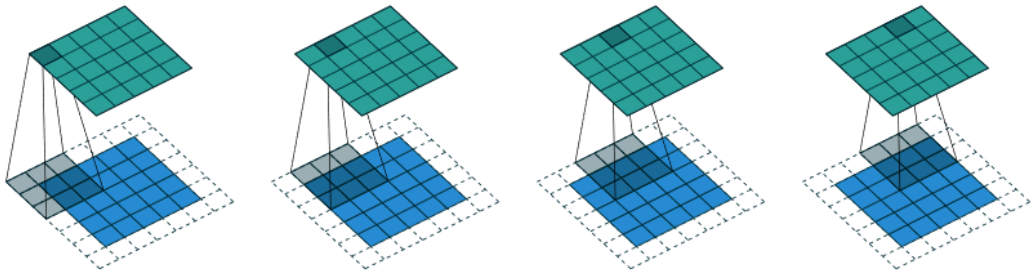
Filtrenin ana matris üzerinde her adımda kaç piksel kaydırılacağı da başarıyı etkileyen bir diğer parametredir. Şekil 4.3'teki örnekte her adımda 1 piksel kayarak ana matris üzerinde işlemler gerçekleştirilmiştir. Ancak gerekli görülmesi durumunda farklı değerler de seçilebilir. Özellikle çok büyük verilerin işlenmesinde donanımsal problemlerin yaşanması durumunda bu parametrenin artırılması çıktı matrisinin boyutunu düşürecektir.

Başarıyı etkileyen bir diğer hiperparametre ise dolgu genişliğidir. Özellikle girdi olarak gelen verilerin kenarlarının (matris sınırlarının) kritik öneme sahip olduğu durumlarda dolgu kullanılmalıdır. Çünkü bu bölgelerde evrişim katmanında veri kaybı olabilir. Bu kaybın önüne geçmek ve bu bölgelerin değerini artırabilmek için genellikle değeri 0 olan ve matrisin etrafını çevreleyen bir katman oluşturulur. Bu durum Şekil 4.7'de incelenebilir.



Şekil 4.8 Dolgusuz evrişim katmanı (Dumoulin ve Visin, 2016)

Şekil 4.8'de dolgusuz bir evrişim katmanı örneği görülebilir. Eğer ana matrisin üzerine derinliği 1 olan dolgu yapılmak istenirse bu defa Şekil 4.9'daki durum meydana gelir. Burada dikkat edilmesi gereken unsur dolgu değeri arttıkça çıktı matrisinin boyutunun da artacağıdır.

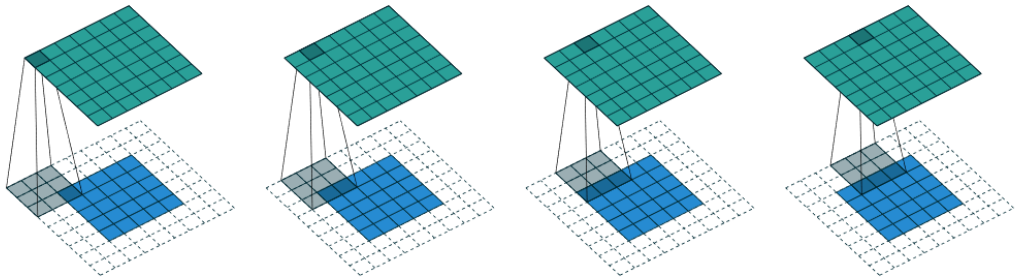


Şekil 4.9 Dolgu derinliği 1 olan evrişim katmanı (Dumoulin ve Visin, 2016)

Özellikle elde az ve değerli verinin olduğu ve sistemin donanımsal olarak bir problem oluşturmadığı durumlarda Şekil 4.7'de görülen dolguyu yapmak yeterli olmayabilir. Nitekim evrişim işleminin dolgusuz olarak gerçekleştiği duruma göre daha iyi sonuçlar elde edilse de matris sınırlarında az da olsa bir veri kaybı mevcuttur. Bu durumun da önüne geçilmek istenirse Şekil 4.10'daki gibi tam dolgu işlemi gerçekleştirilir. Tam dolgu işleminde en önemli unsur filtre boyutunun 1 eksiği kadar ana matris üzerine dolgu yapılmasıdır (Dumoulin ve Visin, 2016). Bu sayede sınır bölgelerinde oluşacak olan kayıp minimum seviyeye indirgenmiş olur.

$$K \times K = \left[\frac{N + 2.p - F}{s} + 1\right] \times \left[\frac{N + 2.p - F}{s} + 1\right] \quad (4.2)$$

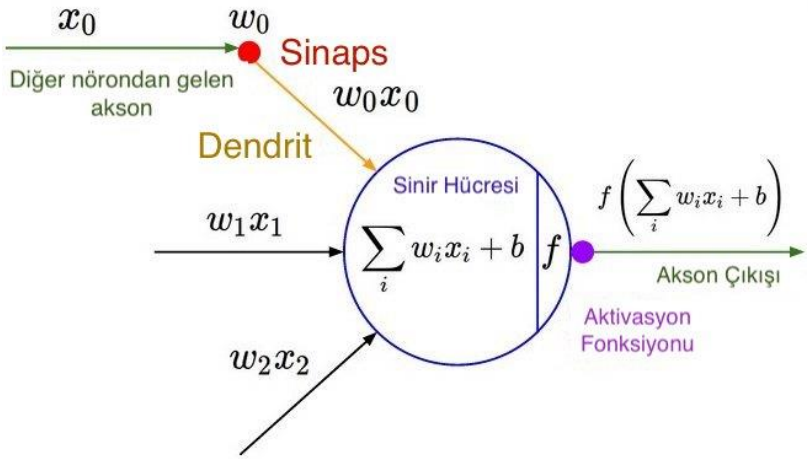
Bütün bu şartlar göz önünde bulundurulduğunda ana matrisin boyutu $N \times N$, filtre matrisinin boyutları $F \times F$, adım sayısı s ve dolgu derinliği p olmak üzere çıktı matrisinin boyutu $K \times K$ Denklem 4.2 ile elde edilir.



Şekil 4.10 Tam dolgu örneği (Dumoulin ve Visin, 2016)

4.3 Aktivasyon Fonksiyonları

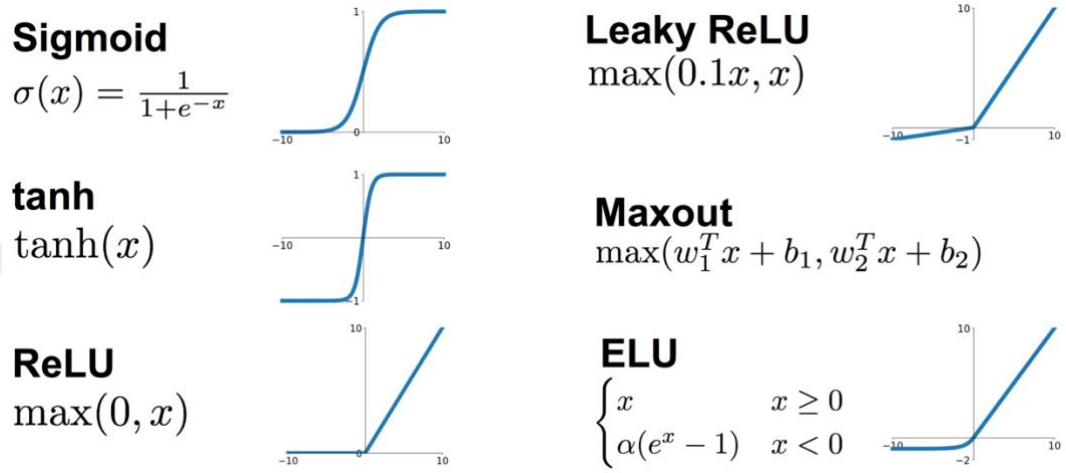
Aktivasyon fonksiyonları temel anlamıyla yapay sinir ağındaki düğümlerde girdilere belirli aktivasyon işlemleri uygulayan ve sonucunda çıktı verilerini elde etmemizi sağlayan fonksiyonlardır. Basit bir aktivasyon işlemi yapısı Şekil 4.11'de görülebilir. Burada girdi olarak gelen veriler belirli ağırlıklarla çarpılarak toplanır. Nihayetinde elde edilen değerler bir b sabiti ile toplanarak çıktı değeri elde edilir.



Şekil 4.11 Aktivasyon fonksiyonunun çalışması (Ariawan, 2018)

Her aktivasyon fonksiyonunda toplama işlemi yapılmak zorunda değildir. Aktivasyon fonksiyonlarının farklı durumlar için kritik öneme sahip özellikleri mevcuttur. Önemli olan bu aktivasyon fonksiyonlarından hangisinin, hangi durumlar için daha uygun olduğunu saptayıp, o doğrultuda kullanmaktır. Örnek olarak Relu gibi

bazı aktivasyon fonksiyonları sıklıkla katmanlar arasında kullanılırken Softmax tarzı fonksiyonlar ise genellikle sadece çıkış katmanında kullanılırlar. İlk durumda elde edilecek olan çıkış değerleri bir diğer katmanın giriş değerleri olacaktır. Diğer durumda ise fonksiyondan nihai çıkış değerleri elde edilir. Bunlar gibi literatürde en çok kullanılan aktivasyon fonksiyonları aşağıdakilerdir (Şekil 4.12).



Şekil 4.12 En sık kullanılan aktivasyon fonksiyonları (Endicott, 2017)

4.3.1 Sigmoid Aktivasyon Fonksiyonu

Sigmoid aktivasyon fonksiyonu Denklem 4.3'teki formülde görüldüğü gibi aldığı değerin [0,1] arasındaki karşılığını üretir. Özellikle olasılık temelli problemlerin çözümlerinde kullanılabilecek bir fonksiyondur (Ding, Qian ve Zhou, 2018). Ancak Sigmoid fonksiyonunun türevine bakıldığında belirli değerlerden sonra sıfıra çok yakın sonuçlar verdiği görülür. Bu da eğimin çok küçük olduğu bu bölgelerde gerçekleşecek değişimlerin yeterince iyi yansıtılamamasına sebebiyet verecektir. Sonuç itibariyle bu tarz durumları içinde barındıran veri setlerinde bu aktivasyon fonksiyonunu kullanmak modelden maksimum performansı almaya engel olacaktır.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.3}$$

4.3.2 Hiperbolik Tanjant Aktivasyon Fonksiyonu

Sigmoid fonksiyonuna benzer bir diğer fonksiyon hiperbolik tanjant fonksiyonudur. Sigmoid fonksiyonundan farklı olarak aldığı değerleri [0,1] aralığında değil [-1,1] aralığındaki karşılığını üretir (Sartin ve Silva, 2013). İki fonksiyonunda hem grafikleri hem türevleri benzer yapılara sahiptir. Sigmoid fonksiyonundaki gradyan kaybolması problemi bu fonksiyonda da mevcuttur. Hiperbolik tanjant fonksiyonunun grafiği Şekil 4.12'de görülmektedir.

4.3.3 Relu Aktivasyon Fonksiyonu

Relu fonksiyonu günümüzde en çok kullanılan aktivasyon fonksiyonudur. Bu fonksiyon Şekil 4.12'de da görüldüğü gibi aldığı değerleri [0, +∞) aralığında değerlere dönüştürür (Zeiler ve diğer., 2013). Yani aldığı negatif değerleri sıfır olarak döndürürken, pozitif değerleri doğrudan geri döndürmektedir. Relu fonksiyonunun günümüzde en çok kullanılan aktivasyon fonksiyonu olmasının sebeplerinden birisi hesaplama yükünün az olmasıdır. Bu da çok katmanlı yapay sinir ağlarında rahatlıkla kullanılmasına neden olmuştur. Şekil 4.12'de görüldüğü üzere fonksiyonun negatif bölgesinde türev sıfır olmaktadır. Bu da bu bölgede öğrenme işleminin gerçekleşmeyeceği anlamına gelir. Bu sebeple Relu fonksiyonunu pozitif değerlerle beslemek eğitim için daha uygun olacaktır.

4.3.4 Leakly Relu Aktivasyon Fonksiyonu

Leaky Relu aktivasyon fonksiyonu tıpkı Relu fonksiyonu gibi aldığı pozitif değerleri doğrudan çıktı olarak döndürmektedir. Relu fonksiyonundan farklı olarak negatif bölgede oluşturduğu çok küçük bir açı ile -∞ değerine ulaşmaktadır (Şekil 4.12). Burada eğimin sıfır olmaması sebebiyle türev değeri de sıfırdan farklı gelecek ve öğrenme işlemi bu bölgede de devam edecektir.

4.3.5 Softmax Aktivasyon Fonksiyonu

Softmax aktivasyon fonksiyonu özellikle çoklu sınıflandırma problemlerinde yoğun olarak kullanılmaktadır (Goodfellow, Bengio ve Courville, 2016). Girdi verileri $\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ denklemi ile işlenir. Neticesinde fonksiyon, toplamları 1 olacak şekilde çıktı değeri üretir. Özellikle olasılık temelli problemlerde sıklıkla bu fonksiyon kullanılmaktadır. Örnek olarak 3 sınıflı bir tahmin probleminin çözümü için oluşturulması gereken senaryoda modelin bize her bir sınıf için döndürdüğü tahmin değerlerinin toplamının 1 olması bizim için çoğu zaman en makul çözüm durumu demektir. Bu sebepten dolayı softmax aktivasyon fonksiyonu genellikle derin öğrenme modellerinin çıkış katmanında sıklıkla kullanılmaktadır.

4.3.6 Üstel Lineer Birim Aktivasyon Fonksiyonu

Üstel lineer birim (Exponential Linear Unit – ELU) aktivasyon fonksiyonu Relu fonksiyonuna oldukça benzerlik göstermektedir. Şekil 4.12'de de görüleceği üzere pozitif bölgede Relu aktivasyon fonksiyonu ile aynı işleve sahiptir. Farklılığa sebep olan negatif bölge incelendiğinde önceden belirlenen bir a değerine göre öteleme işlemi gerçekleştirilir. Bu öteleme işlemi sonucunda fonksiyon negatif bölgede (-a, 0] aralığından değerler alır.

4.3.7 Maxout Aktivasyon Fonksiyonu

Maxout aktivasyon fonksiyonunda girdi olarak gelen bilgiler k sayıdaki gruplara bölünür. Her gruptan gelen değerler analiz edilir. Sonrasında bu değerlerin, $max(w_1^T x + b_1,\ w_2^T x + b_2)$ denkleminde de görüldüğü üzere maksimum olanı alınır. Çıktı değeri böylece üretilmiş olur. Maxout fonksiyonu derin öğrenme modellerinde özellikle Dropout tekniğiyle sıklıkla kullanılmaktadır.

4.4 Kayıp Fonksiyonları

Yukarıda incelenen aktivasyon fonksiyonları, eldeki veri setine uygun olacak şekilde seçilir. Tahmin işlemi aşamasında belirlediğimiz bir aktivasyon fonksiyonuna göre çıktı değerleri elde edilir. Elde edilen tahmin değerlerinin yanı sıra bir de gerçek değerler mevcuttur. Çeşitli kayıp fonksiyonlarından ihtiyaca göre en uygun olan belirlenip gerçek değer ve tahmin değer arasındaki kayıp durumu analiz edilir. Bu analiz elde ettiğimiz eğitim modelinin başarısını anlamak adına oldukça önemlidir. Aynı zamanda bazı eğitim modellerinde eğitim adım adım gerçekleştiği için model kendini kayıp fonksiyonlarının döndürdüğü değere göre her bir adımda güncelleyerek eğitimi ilerletir. Bu durumlarda da eğitimin sağlıklı ve doğru bir şekilde gerçekleşmesinde kayıp fonksiyonlarının rolü büyüktür.

4.4.1 Ortalama Hata Kareleri Kare Kökü

Ortalama hata kareleri kare kökü (Root mean squared error - RMSE) yöntemi kayıp değerlerini analiz etmek için sıkça kullanılan tekniklerden birisidir (Chai ve Draxler, 2014). Denklem 4.4'teki formül ile veri seti içerisindeki her bir gerçek değer ile tahmin değeri arasındaki farkın karesi alınarak toplanır. Elde edilen bu değer, veri setindeki gerçek değer sayısına bölünerek kare kökü alınır. Sonuç olarak modelin ortalama hata kareleri karekökü değeri elde edilir. Aykırı gözlemlere karşı çok dayanıklı değildir.

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y_i - y_{pred,i}\right)^2} \tag{4.4}$$

4.4.2 Ortalama Mutlak Hata

Denklem 4.5'te görülen ortalama mutlak hata (Mean absolute error - MAE) yönteminde her bir gerçek değer ile tahmin değeri arasındaki fark alınır. Sonucun mutlak değeri alınır ve ortalaması elde edilir. Ortalama hata kareleri kare kökü

yöntemine kıyasla aykırı gözlemlere karşı çok daha dayanıklıdır. Veri setinde aykırı gözlemlerin etkisini azaltmak ve bu gözlemlere daha az önem vermek istediğimiz durumlarda ortalama mutlak hata yöntemini kullanmak daha iyi olacaktır (Willmott ve Matsuura, 2005). Ancak aykırı gözlemlerin kritik öneme sahip olduğu veri setlerinde ortalama hata kareleri kare kökü yöntemini kullanmak doğru olacaktır.

$$MAE = \frac{1}{m}\sum_{i=1}^{m} | y_i - y_{pred,i} | \tag{4.5}$$

Aradaki farkın daha iyi anlaşılması için örnek verecek olursak elimizde gerçek değeri [10,10,10,10] olan bir durum için ilk tahmin değerinin [9,9,9,9] olduğunu varsayalım. Bu tahmin durumu için her iki yöntemde de (Denklem 4.4 ve Denklem 4.5) kayıp değer 1 olarak bulunacaktır. Ancak tahmin değerinin [10,10,10,2] olması durumunda ortalama mutlak hata değeri $\frac{1}{4}(0+0+0+8) = 2$ olarak elde edilir. Aynı tahmin değeri için ortalama hata kareleri kare kökü değeri $\sqrt{\frac{1}{4}(0+0+0+8^2)} = 4$ değeri elde edilir. Buradan anlaşılacağı üzere gerçek değer ve tahmin değeri arasındaki fark arttığında aynı durum için RMSE değerindeki artış MAE değerindeki artışa oranla daha fazla olmaktadır.

4.4.3 Huber Fonksiyonu

Kayıp fonksiyonu olarak kullanılan bir diğer fonksiyon ise Huber fonksiyonudur (Denklem 4.6). Gerçek değer ve tahmin değer arasındaki farkın mutlak değeri, belirlenen eşik değerin altında olması durumunda kayıp değeri, gerçek değer ve tahmin değeri arasındaki farkın karesinin yarısı şeklinde elde edilir. Aksi durumlarda kayıp değeri belirlenen eşik değerinin karesinin yarısını, gerçek değer ve tahmin değeri arasındaki farkın mutlak değerinin eşik değeri ile çarpımından ortaya çıkan sonuçtan çıkartılmasıyla elde edilir (Huber, 1964). Bu durum sayesinde Huber kayıp fonksiyonu da aykırı gözlem değerlerine, ortalama hata kareleri kare kökü yöntemine göre daha az duyarlı olur.

$$L_{\delta}(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & for |y - f(x)| \leq \delta, \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & otherwise \end{cases} \quad (4.6)$$

4.4.4 İkili Çapraz Entropi

Denklem 4.7 ile gösterilen ikili çapraz entropi (binary cross entropy) yöntemi çıktı olarak iki seçenekten birini (evet-hayır veya 1-0 gibi) elde edilmek istendiğinde kullanılır. Genellikle sınıflandırma problemlerinde kullanılan bu yöntem özellikle ikili sınıflandırma durumlarında yaygın olarak kullanılmaktadır.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \,.\, log\big(p(y_i)\big) + (1 - y_i).\, log\big(1 - p(y_i)\big) \quad (4.7)$$

4.4.5 Çapraz Entropi

Çapraz Entropi (Cross Entropy) kayıp fonksiyonu (Denklem 4.8) genel olarak çoklu sınıflandırma problemlerinde sıklıkla tercih edilmektedir (Nasr, Badr ve Joun, 2002). Örnek üzerinden anlatmak gerekirse 3 sınıfın olduğu bir durumda gerçek değerin [1,0,0] olduğunu ve tasarladığımız modelin softmax katmanından gelen çıktısında [0,2, 0,5, 0,3] değerinin olduğunu varsayalım. Bu durumda Denklem 4.8'e göre elde edeceğimiz hata değeri $-(1.log(0{,}2) + 0.log(0{,}5) + 0.log(0{,}3) = 1{,}61$ olarak elde edilir. Yaptığımız optimizasyon işlemleri sonucunda yeni tahmin değerimizin [0,8, 0,1, 0,1] olduğunu düşünelim. Bu durumda $-(1.log(0{,}8) + 0.log(0{,}1) + 0.log(0{,}1) = 0{,}22$ olarak yeni hata değeri elde edilir. Yukarıda da görüldüğü üzere yapacağımız tahminin doğruluğu ne kadar büyük olursa kayıp değerimizde o oranda azalmış olacaktır.

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \,.\, log\big(y_{pred,i}\big) \quad (4.8)$$

4.4.6 Kullback Libler Iraksama Metodu

Kullback Libler ıraksama (KL divergence) yöntemi kayıp fonksiyonunu hesaplamak için kullanılan bir diğer yöntemdir (Press, Teukolsky, Vetterling ve Flannery, 2007). Burada amaç iki olasılık dağılımı arasındaki farkı bulmaktır. Denklem 4.9 ile elde edilen değer arttıkça olasılık dağılımlarının birbirlerinden farklı oldukları, değer azaldıkça ise bu dağılımların birbirlerine yakın oldukları anlamına gelir. KL ıraksama fonksiyonu minimum 0 değerini alır, negatif bir değer alamaz. Burada dikkat edilmesi gereken bir diğer husus ise eğer gerçek dağılım tekil sıcak kodlama sonucunda elde edildiyse veya bir diğer deyişle dağılımda bir elemanın değeri 1 iken diğerlerinin 0 olması durumunda (çapraz entropi için verilen çoklu sınıflandırma probleminde olduğu gibi [1,0,0]) denklemin entropi kısmı olan $\sum_i y_i \,.\, log\frac{1}{y_i}$ bölümünde y_i değerinin 1 olması sebebiyle bu kısım 0 değerini alacaktır. Dolayısıyla bu şartları sağlayan herhangi bir çoklu sınıflandırma probleminde KL ıraksama yöntemi kullanılmak istendiğinde elde edilecek sonuç çapraz entropi ile elde edilecek sonuca eşit olacaktır.

$$KL(y||y_{pred}) = \sum_i y_i \,.\, log\frac{1}{y_{pred,i}} - \sum_i y_i \,.\, log\frac{1}{y_i} = \sum_i y_i \,.\, log\frac{y_i}{y_{pred,i}} \qquad (4.9)$$

4.4.7 Karşıtsal Kayıp

Karşıtsal kayıp (Contrastive loss) fonksiyonu çapraz entropi gibi tahmin bazlı hata fonksiyonlarının aksine mesafe bazlı bir hata fonksiyonudur (Qi ve Su, 2017). Genellikle görüntüler için kullanılan bu fonksiyon, görüntülerin birbirleriyle olan farklılıklarından bilgi elde edilmesi üzerine kurulmuştur. Buradaki amaç girdi olarak gelen veri çiftleri arasındaki pozitif çiftler (birbirine benzer) arasındaki uzaklıkları minimize etmek, negatif çiftler (birbirlerinden farklı) arasındaki uzaklıkları maksimize etmektir. Denklem 4.10'da görülen formül ile karşıtsal kayıp değeri hesaplanır. D_w tahmin edilen uzaklık değeri, Y değeri 0 veya 1 olmak üzere, girdi olarak gelen görüntüler aynı sınıfa aitlerse 0, farklı sınıflara aitlerse 1 değerini alır.

$$L_{contrast} = (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}max(0, m - D_W)^2 \quad (4.10)$$

4.4.8 Menteşe Yitimi

Menteşe Yitimi (Hinge Loss) fonksiyonu makine öğrenmesinde sınıflandırıcıları eğitmek için kullanılan kayıp fonksiyonudur (Wu ve Liu, 2007). İkili sınıflandırma problemlerinde çapraz entropiye alternatif olarak kullanılabilir. Başlangıçta özellikle destek vektör makinesi modelleriyle kullanılmak üzere geliştirilmiştir ancak günümüzde kayıp fonksiyonunda yapılan ufak değişiklikler ile farklı modellerle de kullanılabilmektedir. Lojistik kayıp fonksiyonuyla kıyaslandığında aykırı gözlemlere karşı daha dayanıklı bir yapıda olduğu görülmektedir. Örnek olarak tahmin etmek istediğimiz etiketin gerçek değerinin 1 olduğunu varsayalım. Yapılan tahmin işleminden sonra elde edilen değerler ise [1, 0,8, 0,1] olsun. Bu durumda Denklem 4.11'e göre kayıp değerleri;

$max[0{,}1 - (1 \cdot 1)] = max[0, 0] = 0$
$max[0{,}1 - (1 \cdot 0{,}8)] = max[0, 0{,}2] = 0{,}2$
$max[0{,}1 - (1 \cdot 0{,}1)] = max[0, 0{,}9] = 0{,}9$

elde edilir. Burada gerçek değere yakın tahmin değerleri elde edildiğinde 0,2 gibi küçük bir kayıp değeri dönerken, gerçek değerden çok farklı bir tahmin değeri elde edildiği durumlarda 0.9 gibi oldukça yüksek kayıp değeri elde edilmektedir.

$$\sum max(0{,}1 - y.y_{pred}) \quad (4.11)$$

BÖLÜM 5
FATURALARDAN BİLGİ ÇIKARIMI

Görüntü formatındaki faturaların baştan sona eğitim süreci incelendiğinde özet olarak öncelikle faturalara belirli görüntü işleme teknikleri uygulanır. Sonrasında ".tiff" uzantılı faturalara OCR işlemi uygulanır. Bu işlem sonucunda görüntü formatındaki dokümanlardan işlem görebilecek ".tsv" çıktıları üretilir. Bu çıktılar yardımıyla eğitimin yaptırılacağı ".csv" dosyası üretilir. Bu aşamada Levenshtein ve Jaro-Winkler mesafeleri, Char2vec değerleri gibi eğitime faydalı olabilecek birçok özellik öznitelik sütunu olarak çıktıya eklenir. Hazırlığı tamamlanan veriler Bölüm 3 ve Bölüm 4'te anlatılan modeller ile eğitime girer. Eğitim sonuçlarının değerlendirilmesi akabinde en iyi model belirlenir.

5.1 Optik Karakter Tanıma (OCR)

Bu çalışmada kullanılan bütün faturalar görüntü formatında olduğu için ilk işlem olarak bu görüntülerden yazı formatında bilgiler elde edilir. Bunun için derin öğrenme tekniklerine dayalı Tesseract isimli açık kaynak kodlu optik karakter tanıma motoru kullanılmıştır (Smith, 2007). İlk olarak 1984 yılında, el yazısı ile hazırlanmış dokümanları elektronik ortama aktarma fikri ile geliştirmeye başlanmıştır. Özellikle teknolojinin ilerlemesine paralel olarak bilgisayarların donanımsal olarak güçlenmesiyle beraber zaman içinde derin öğrenme teknikleriyle de güçlendirilerek, doğruluk oranlarını artırmıştır. Günümüzde Tesseract, en çok kullanılan optik karakter tanıma motorlarının başında gelmektedir.

Optik karakter tanıma işleminin ilk adımında ikilileştirme (binarization) işlemi yapılır. Şekil 5.1'de görülebileceği gibi bu işlemin devamında sayfa analizi gerçekleştirilir. Sayfa analizi işlemi sonrasında doküman üzerindeki kelimeler ve bu kelimelerin satırları tespit edilir. Blok yapısı oluşturulur. Daha sonra kelimelerin tanınma işlemi gerçekleştirilir. Bu işlem 2 aşamadan oluşmaktadır. İlk aşamada statik sınıflandırıcı yardımıyla tanınma işlemi gerçekleştirilir. Sonrasında ise uyarlanabilir sınıflandırıcıyla ikinci aşama tamamlanır.

Şekil 5.1 Optik karakter tanıma aşamaları

5.1.1 İkilileştirme İşlemi

Sisteme gelen ham veriler içerisinde hem renkli hem de siyah-beyaz faturalar bulunmaktadır. Tesseract motorundan optimum verim alınabilmesi için renkli olan veriler öncelikle siyah–beyaz formata getirilir. Bu yapıdaki resimlerin piksel değerleri 0 ile 255 arasında değişmektedir. İkilileştirme yöntemi ile bu pikseller yalnızca 0 veya 255 değerini alacak hale getirilir. Bunun için Tesseract kendi bünyesinde Otsu ikilileştirme metodunu kullanmaktadır.

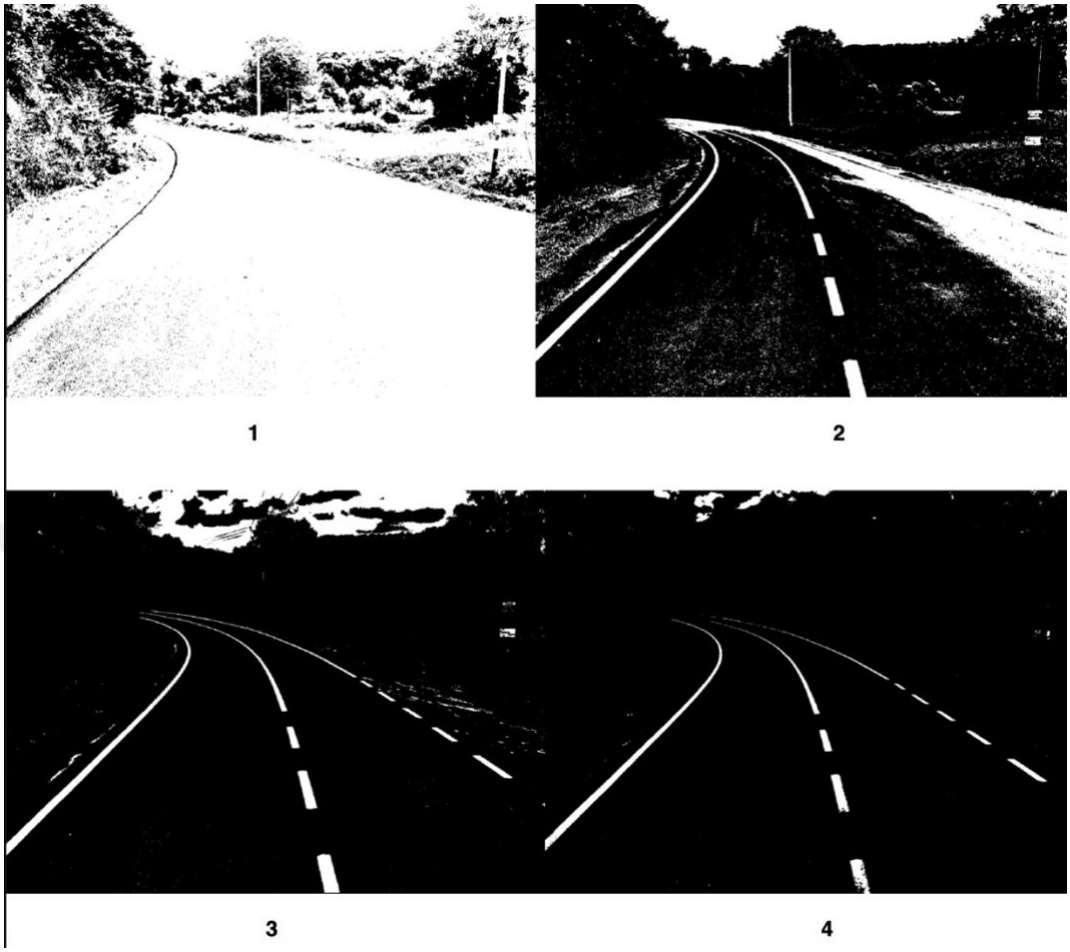
$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{5.1}$$

İkilileştirme işleminin yapılabilmesi için bir eşik değer belirlenmelidir. Başlangıçta resim üzerindeki her bir piksel potansiyel eşik değeri olarak görülür. Otsu metodundaki esas amaç ikili hale gelecek olan sınıfların kendi içerisinde varyans değerini minimum yapmaktır. ω_0 ve ω_1 sınıfların olasılıksal ağırlıklarını ($w(k),\ 1 - w(k)$), t değeri sınır değerini ve σ_1^2,σ_2^2 değerleri de sınıfların varyans değerleri olmak üzere sınıf içi varyans değeri Denklem 5.1'de görüldüğü gibi elde edilir. Sınıf içi varyans değerinin minimum olmasının yanı sıra Denklem 5.2'deki sınıflar arası varyans değerinin ise maksimum olması gerekir. Bu koşulu optimum derecede sağlayan t eşik değeri seçilerek, bu eşik değere göre ikilileştirme işlemi gerçekleştirilir (Nair, 2016).

$$\sigma_b^2(t) = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2 \tag{5.2}$$

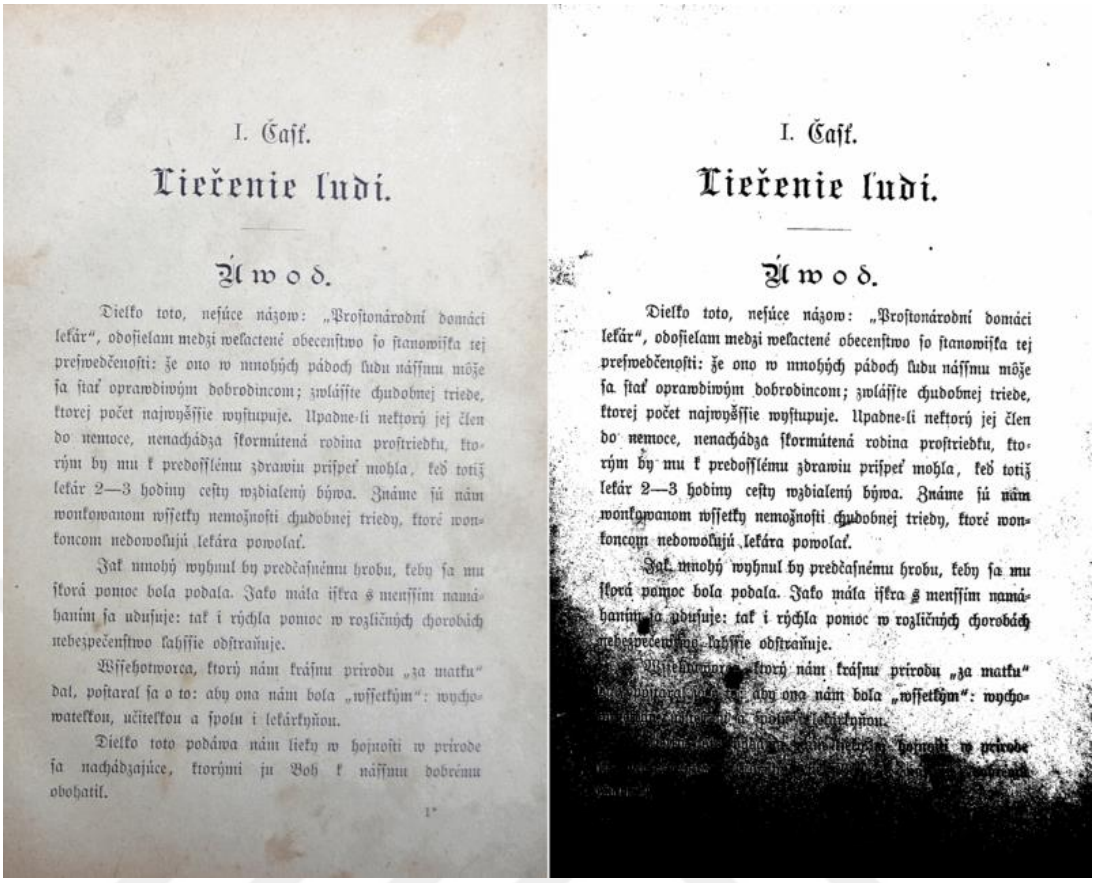
Eşik değerin doğru olarak belirlenmesi sadece dokümanlar için değil, genel olarak bütün görüntü işleme problemlerindeki en önemli adımlardan birisidir. Nitekim bu alanda belirlenecek parametreler, elde edilecek verilerin kalitesini doğrudan etkileyecektir. Örnek olarak Şekil 5.2'deki görüntü belirli eşik değerlerine göre ikilileştirilecek olsun.

Şekil 5.2 İkilileştirilecek görüntünün işlenmemiş hali (Sakaryadanhaber, 2019)



Şekil 5.3 Farklı eşik değerleri ile görüntünün ikilileştirilmesi

Şekil 5.3'te aynı görüntünün (Şekil 5.2) 4 farklı eşik değere göre ikilileştirilmesi görülebilir. Burada ilk durum için eşik değeri 20 olarak verilmiştir. Eşik değerin düşük olarak belirlenmesi Şekil 5.3'ün birinci durumunda da görüleceği üzere siyah ve siyaha çok yakın olan renkler 0 değerini almış, kalan bütün renkler de 255 değerini almıştır. İkinci durumda 80, üçüncü durumda 160 ve son olarak dördüncü durumda da 200 olacak şekilde eşik değer belirlenmiştir. Şekil 5.3'ün birinci ve dördüncü durumlarını inceleyecek bir kişi bambaşka iki görüntüyü inceliyor hissine kapılabilir. Bu sebeple bu tarz problemlerde eşik değerin doğru ve amaca yönelik olarak belirlenmesi oldukça önemlidir.



I. Časť.

Liečenie ľudí.

Úvod.

Dielko toto, nesúce názov: „Prostonárodní domáci lekár", odosielam medzi welactené obecenstwo so stanowiska tej preswedčenosti: že ono w mnohých pádoch ľudu nášmu môže sa stať opravdivým dobrodincom; zwlášte chudobnej triede, ktorej počet najwyššie wystupuje. Upadne-li nektorý jej člen do nemoce, nenachádza skormútená rodina prostriedku, ktorým by mu k predošlému zdraviu prispeť mohla, keď totiž lekár 2—3 hodiny cesty wzdialený býwa. Známe sú nám wonkajšnom wšetky nemožnosti chudobnej triedy, ktoré wonkoncom nedowoľujú lekára powolať.

Jak mnohý wyhnul by predčasnému hrobu, keby sa mu skorá pomoc bola podala. Jako mála iskra s menšim namáhaním sa udušuje: tak i rýchla pomoc w rozličných chorobách nebezpečenstwo ľahšie odstraňuje.

Wšehotworca, ktorý nám krásnu prirodu „za matku" dal, postaral sa o to: aby ona nám bola „wšetkým": wychowatelkou, učitelkou a spolu i lekárkyňou.

Dielko toto podáwa nám lieky w hojnosti w prirode sa nachádzajúce, ktorými ju Boh k náššmu dobrému obohatil.

1*

I. Časť.

Liečenie ľudí.

Úvod.

Dielko toto, nesúce názow: „Prostonárodní domáci lekár", odosielam medzi welactené obecenstwo so stanowiska tej preswedčenosti: že ono w mnohých pádoch ľudu nášmu môže sa stať opravdivým dobrodincom; zwlášte chudobnej triede, ktorej počet najwyššie wystupuje. Upadne-li nektorý jej člen do nemoce, nenachádza skormútená rodina prostriedku, ktorým by mu k predošlému zdraviu prispeť mohla, keď totiž lekár 2—3 hodiny cesty wzdialený býwa. Známe sú nám wonkajšnom wšetky nemožnosti chudobnej triedy, ktoré wonkoncom nedowoľujú lekára powolať.

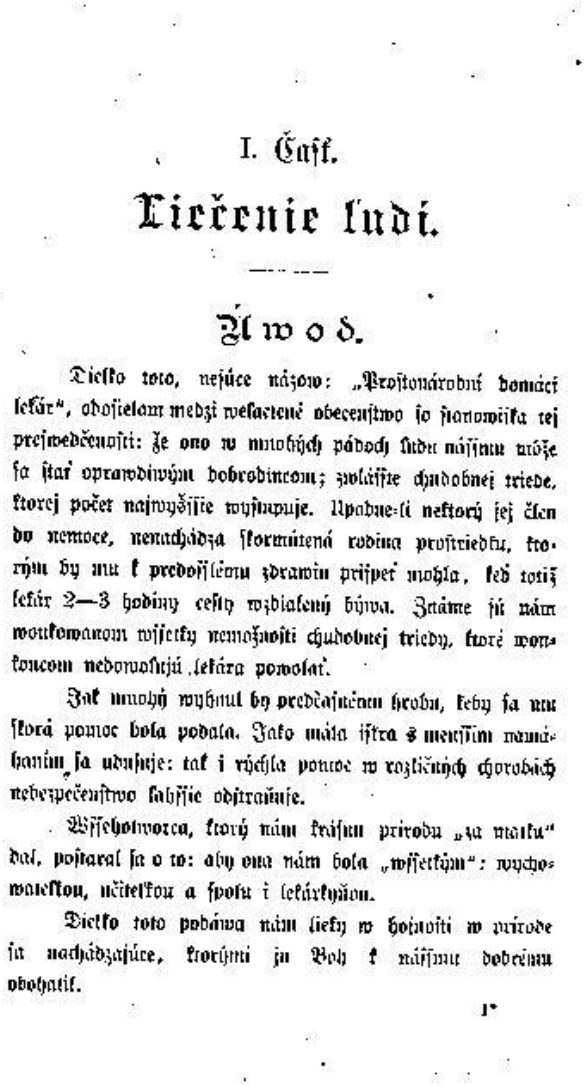
Jak mnohý wyhnul by predčasnému hrobu, keby sa mu skorá pomoc bola podala. Jako mála iskra s menšim namáhaním sa udušuje: tak i rýchla pomoc w rozličných chorobách nebezpečenstwo ľahšie odstraňuje.

Şekil 5.4 İkilileştirilmiş örnek doküman (Github, 2015)

Dokümanlar için de ikilileştirme adımı oldukça önemlidir. Tesseact'ın kendi bünyesinde gerçekleştirdiği ikilileştirme tekniğinin yanı sıra farklı teknikler de mevcuttur. Özellikle doküman üzerinde gölgelenmelerin olması veya sayfanın bir tarafının aydınlık, diğer tarafının karanlık olması gibi durumlarda global bir eşik değer belirlenmesi sağlıklı sonuç elde edilmesini engeller.

Şekil 5.4'te görülen örnek Tesseract'ın resmi Github sayfasından alınmıştır. Örnekteki doküman Tesseract'ın kendi bünyesindeki Otsu tekniği (Otsu, 1979) kullanılarak ikilileştirildiğinde özellikle sol alt bölge başta olmak üzere belge üzerinde problemler oluşmaktadır. Bu problemlerin esas sebebi yukarıda da bahsedildiği üzere alt kısımların belgenin geneline göre daha koyu renkte olmasıdır. Bu tarz durumlarda daha iyi sonuç elde edebilmek için veriyi Tesseract'a göndermeden önce ikilileştirme işlemini yapmak gerekir. Şekil 5.4'te görülen işlenmemiş belge, OpenCV kütüphanesi içerisindeki "adaptiveThreshold" fonksiyonu ile işleme girdiğinde ortaya Şekil

5.5'teki durum ortaya çıkmıştır. İlk durumlarla kıyaslandığında uyarlanabilir eşik tekniğiyle görüntünün çok daha düzgün şekilde ikilileştirildiği görülebilir. Özellikle ilk durumda sayfanın son çeyreğinin Tesseract tarafından okunmasının çok zor olduğu açıkça görülmektedir. Ancak Şekil 5.5'teki durumda bu tarz bir problem görünmemektedir. Bu işlemin temelinde, ikilileştirme için gerekli olan eşik değerin global olarak değil de lokal olarak belirlenmesi fikri vardır. Eşik değer belirli komşu piksellere göre belirlenir. Nihayetinde sayfa üzerinde farklı alanlarda farklı eşik değerler meydana gelir. Bu aşamada elde edilen eşik değerler bölgesel olarak değerlendirilerek kullanılır. Böylece daha okunabilir veriler elde edilebilir.



I. Časť.

Liečenie ľudí.

Úvod.

Dielko toto, nesúce názov: „Prostonárodní domáci lekár", odovzdáva medzi nevzdelané obecenstvo so stanovitka tej presvedčenosti: že ono v mnohých prípadoch ľudu nášmu môže sa stať opravdivým dobrodincom; zvlášte chudobnej triede, ktorej počet najvyššie vynikuje. Upadne-li niektorý jej člen do nemoce, nenachádza sformovaná rodina prostriedku, ktorým by mu k predošlému zdraviu prispieť mohla, keď totiž lekár 2—3 hodiny cesty vzdialený býva. Známe sú nám vonkoncom všetky nemožnosti chudobnej triedy, ktoré vonkoncom nedovolujú lekára povolať.

Jak mnohý vyhnul by predčasnému hrobu, keby sa mu skorá pomoc bola podala. Jako málo iskra s menším namáhaním sa udusuje: tak i všetka pomoc v rozličných chorobách nebezpečenstvo ľahšie odstraňuje.

Všemohúca, ktorý nám krásnu prírodu „za matku" dal, postaral sa o to: aby ona nám bola „všetkým": vychovateľkou, učiteľkou a spolu i lekárkyňou.

Všetko toto podáva nám lieky v hojnosti v prírode sa nachádzajúce, ktorými ju Boh k nášmu dobrému obohatil.

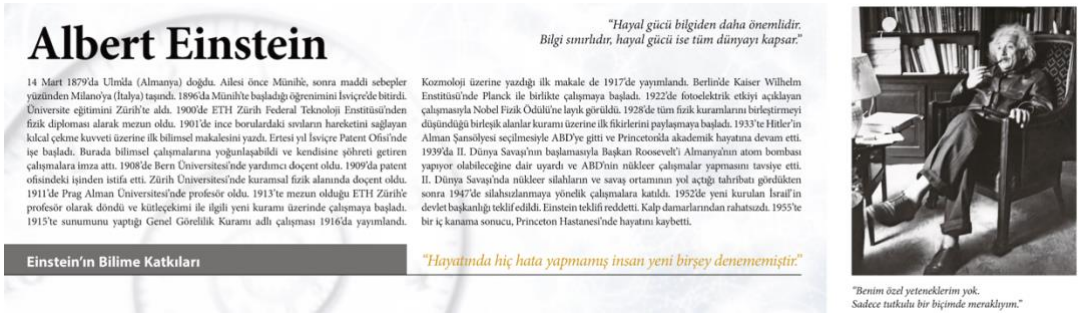
1*

Şekil 5.5 Uyarlanabilir eşik değere göre ikilileştirme

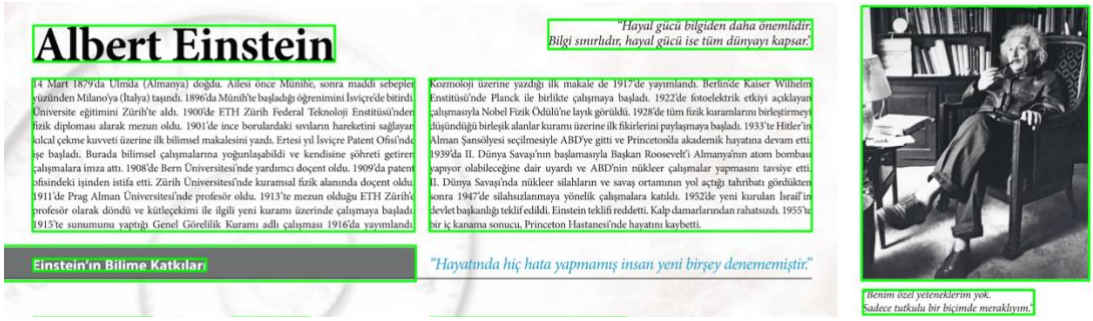
Sonuç olarak bu alan, Tesseract işlemi yapılan çalışmaların en kilit noktalarından birisidir. Tesseract içinde varsayılan olarak çalışan ikilileştirme tekniği birçok problem için yeterli sonuç üretebilmektedir. Ancak bazı spesifik durumların olduğu da unutulmamalıdır. Şekil 5.4'e benzer durumlar karşısında Tesseract'ın ikilileştirme tekniğini varsayılan olarak kullanmak kötü sonuçların alınmasına sebep olur. Burada yapılması gereken en doğru hareket, problemin ne olduğunu kavrayarak probleme göre çözüm üretmektir. Örnek olarak yukarıdaki durumlara benzer problemlerin yaşanması durumunda literatürde de sıkça kullanılan ikilileştirme tekniğinin özelleştirilerek global yapıya geçilmesiyle çözüm elde edilebilir.

5.1.2 Sayfa Analizi

Sayfa analizi işleminde öncelikle doküman içerisindeki yazılar ve resimler ayrılır. Yazı içerisindeki sekme duraklarından ve hizalamalardan faydalanarak gruplama işlemleri gerçekleştirilir. Bu işlem esnasında yukarıdan aşağıya ve soldan sağa doğru tarama işlemi yapılır (Patel, I., C., Patel, A. ve Patel, D., 2012). Benzer özelliklere sahip olan veriler aynı grupta toplanır. Şekil 5.6'daki görüntünün üzerine sayfa analizi yapıldığında Şekil 5.7'deki görünüm elde edilir. Buradan da anlaşılacağı üzere belge üzerindeki resim tek bir blok şeklinde alınır. Yazılarda ise sekme duraklarına ve boşluk değerlerine göre gruplama işlemi yapılır.



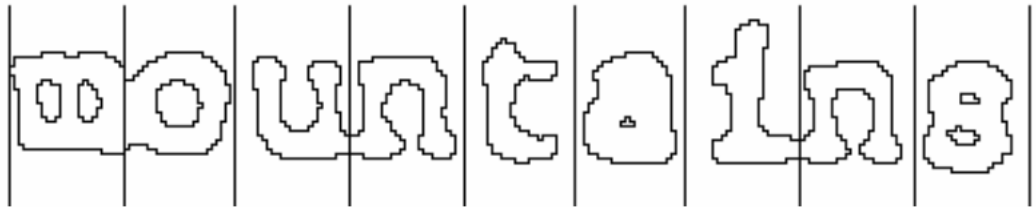
Şekil 5.6 Örnek Albert Einstein yazısı (Yazıcı, 2015)



Şekil 5.7 Örnek yazının sayfa analizi sonrası elde edilen çıktısı (Yazıcı, 2015)

5.1.3 Satır ve Kelimelerin Tespiti

Satır tespit işlemini sayfa analizine entegre bir şekilde düşünmek daha doğru olacaktır. Tesseract satır tespit sürecinde eğik olan satırları düzeltmeden satırları tespit etmeye çalışır. Burada eğikliği düzeltirken ortaya çıkacak olan bilgi kaybının engellenmesi amaçlanmıştır. Satır elemanlarını belirlerken damla adı verilen sürekli bileşenler üzerinden işlem yapılır. Metindeki bileşenlerin ortalama yüksekliği metnin yüksekliği olarak kabul edilir. Bu noktada ortalama yüksekliğin belirli bir kısmından daha küçük olan bileşenler filtrelenerek satır hesabından atılır. Bu bileşenler genellikle gürültü veya noktalama işaretleridir. Filtreleme sonucunda atılmayan bileşenler birbirleriyle çakışmayan, paralel çizgilere oturtulurlar. Bu çizgiler dokümanın eğik olması durumuna göre eğimli de olabilmektedirler. Bu işlem sonrasında her bir bileşenin x koordinatlarına göre satır içerisindeki dizilimleri gerçekleştirilir (Smith, 2007).

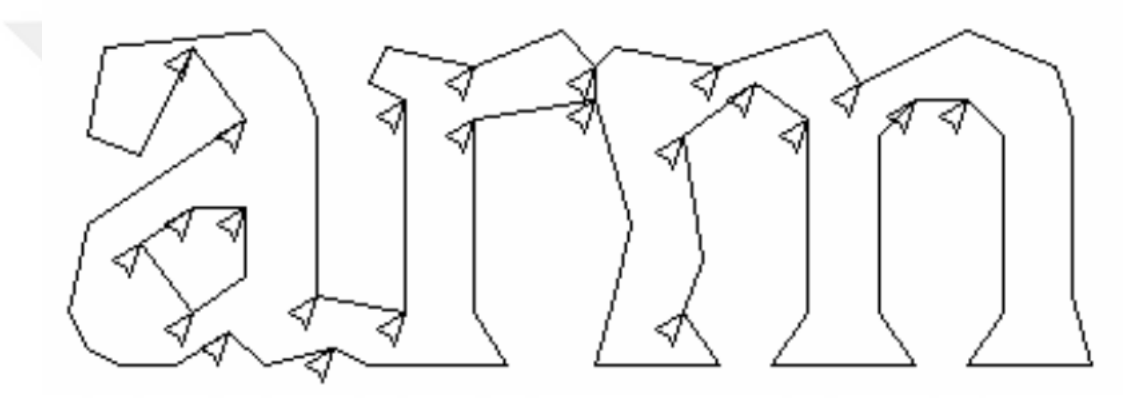


Şekil 5.8 Kelimenin harflerine ayrılma aşaması (Smith, 2007)

Tesseract kelime tanıma adımından önce son işlem olarak kelimelerin tespiti için satır üzerinde tarama yapar. Bu tarama işleminde Şekil 5.8'de görüldüğü gibi sabit aralıklı metin bulduğu anda bu kelimeyi karakterlerine ayırır.

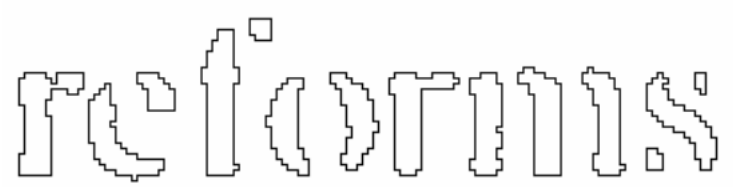
5.1.4 Kelime Tanıma

Kelime tanımanın en kritik adımlarından birisi karakterlerin düzgün şekilde bölünme işlemidir. Tesseract bu işlemi yaparken bir önceki adımda da anlatıldığı üzere sabit aralıklara göre bölme işlemi yapar.



Şekil 5.9 Birleşik olarak yazılmış yazı örneği (Smith, 2007)

Şekil 5.9'daki gibi durumlarda tanınması gereken kelimler birleşik olarak yazılmış olabilir. Bu tarz durumlarda birleşik karakterlerin ayrılacağı noktayı bulmak için karakterlerin ana hatları boyunca oluşturduğu poligonal iç bükey köşelere bakılır. Zıt iç bükey bir tepe noktası veya karşısında farklı bir satır segmenti olması durumunda bu nokta parçalanma noktası olarak belirlenir. Eldeki verinin durumuna göre parçalanma işleminin sağlıklı olarak yapılabilmesi için 3 farklı parçalanma noktasına ihtiyaç duyulabilir. Tespit edilen parçalanma noktalarına göre parçalama işlemi yapılıp kelimenin tanınması gerçekleştirilir.



Şekil 5.10 Bozulma meydana gelmiş örnek kelime (Smith, 2007)

Bazı durumlarda kelimelerin yapılarından ufak bozulmalar meydana gelebilir. Tesseract Şekil 5.10'daki gibi bozulmalara sahip olan kelimeleri okumayı başarmaktadır. Öncelikle kelimeler olağan şekilde karakterlerine ayrılır. Eğer kelime yeterince iyi değilse bu defa Tesseract içerisindeki ilişkilendiriciyi devreye alır. Kelime bu ilişkilendiriciye gönderilir. Maksimum sayıda parçalara ayrılan bu karakterlerin kombinasyonları, ilişkilendirici içerisinde değerlendirilerek en uygun durum saptanır.

Kelimelerin tanınma işlemi için özniteliklerin oluşturulması gereklidir. Tesseract'ın önceki versiyonlarında kelimelerin font ve boyutları dikkate alınmadan yalnızca topolojik özellikleri kullanılırdı. Sonrasında bu fikir karakterlerin çokgen özelliklerini kullanarak sınıflandırma işleminin yapılması fikrine evrilmiştir. Eğitim için 4 boyutlu (x, y, açı, uzunluk) öznitelik vektörü kullanılmıştır. Tanıma işleminde her bir element küçük, eşit uzunluklu parçalara bölünür. Uzunluğun eşit olması sebebiyle bu değer öznitelik vektöründen çıkartılır. Statik sınıflandırıcıdan elde edilen bilgiler uyarlanabilir sınıflandırıcıya eğitim verisi olarak gönderilir. Uyarlanabilir sınıflandırıcı statik sınıflandırıcıya göre font ve yazı tipi özelliklerine karşı daha duyarlıdır. Böylelikle her belgenin farklı yanlarını daha iyi tespit ederek bu farklılıklara göre tahminde bulunabilir. Eğitim sayfanın başlayarak sonuna doğru kümülatif bir şekilde gerçekleştirilir. Buradaki en büyük problem sayfanın başındayken henüz yeterince iyi eğitilmemiş olan uyarlanabilir sınıflandırıcı sayfanın sonuna geldiğinde istenilen seviyeye ulaşmış olur. Bu durumda sayfanın üst bölgelerinde sınıflandırıcıdan yeterince iyi faydalanılmamış olur. Bu problemin önüne geçmek adına işlem tamamlandıktan sonra sayfa tekrar en baştan, eğitimi tamamlanmış sınıflandırıcı ile analiz edilir (Nair, 2016).

5.1.5 Faturaların Optik Karakter Tanıma İşlemi

Görüntü formatındaki faturaların optik karakter tanıma işlemi yukarıda anlatılan tekniklere göre Tesseract yardımıyla gerçekleştirilir. Yapılan bu işlem sonucunda işlenmiş fatura özel ".tsv" dosyası üretilir.

Tablo 1.1 Optik karakter tanıma sonucunda elde edilen bilgiler

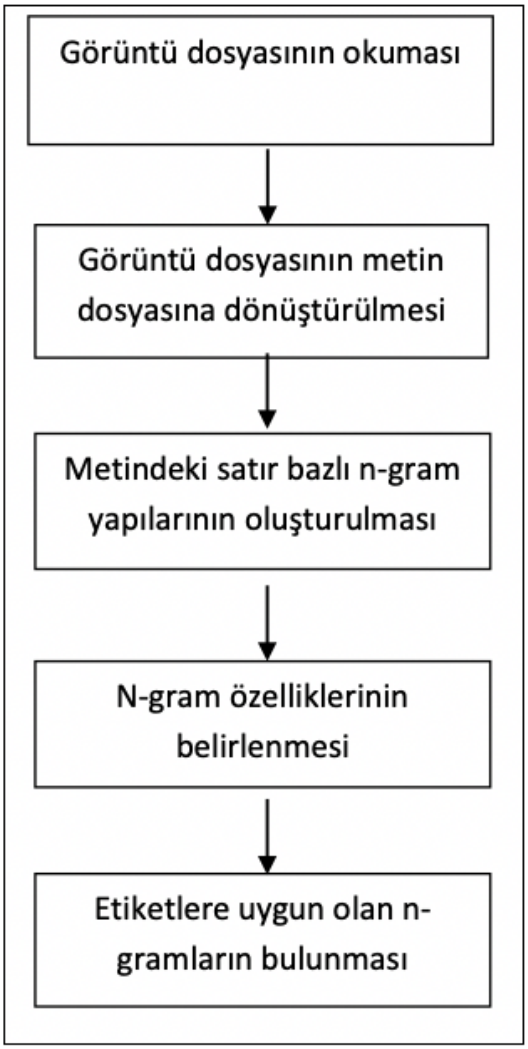
Öznitelik adı	**Anlamı**
level	Algılanan birimin düzeyi
page_num	Algılanan birimin sayfa numarası
block_num	Algılanan birimin sayfadaki blok numarası
par_num	Algılanan birimin bloktaki paragraf numarası
line_num	Algılanan birimin paragraftaki satır numarası
word_num	Algılanan birimin satırdaki söz numarası
left	Algılanan birimin sol üst köşe X koordinatı
top	Algılanan birimin sol üst köşe Y koordinatı
width	Algılanan birimin genişliği
height	Algılanan birimin yüksekliği
conf	Algılanan birimin tanınma doğruluk yüzdesi
text	Algılanan birimin metni

Üretilen bu dosya içerisinde Tablo 5.1'de görülen özellikler mevcuttur. Bu ".tsv" dosyası ile eğitim için gerekli olan ".csv" dosyasının üretimi gerçekleştirilir. Bu özelliklerin birçoğu ve bu özelliklerin farklı kombinasyonları ile yapılmış hesaplamalar öznitelik olarak eğitim dosyasına eklenir.

5.2 N-gram Yapısı

Optik karakter tanıma sonucunda algılanan her bir birimin metin, koordinat, satır numarası gibi bilgileri çıkartılır. Ancak burada bazı problemler mevcuttur. Algılanan bütün birimler aralarındaki boşluklara göre ayrılır. Bu birimlerin arasında boşluk

kalmayana kadar bölünme işlemi devam eder. Nihayetinde bölünme işlemi yapılacak boşluk kalmadığı anda ".tsv" dosyasına "text" olarak eklenir. Bu noktada problem, fatura üzerinde elde etmek istediğimiz bilgilerin her zaman tek kelimeden veya tek bir ifadeden meydana gelmemesidir. Bu sebepten dolayı elde edilen metinlerin N-gram değerlerinin çıkartılması gerekir (Watanabe, Tsukada ve Isozaki, 2009).



Şekil 5.11 Örnek N-gram yapısının oluşturulması

Tez kapsamında kullanılan faturalar genellikle Almanca olduğu için birçok değerli bilgi birleşik olarak yani tek bir ifade şeklinde bulunmaktadır. Ancak IBAN, vergi numarası ve hatta zaman zaman tarih ve fatura numaralarında da birden fazla ifadenin birleşimi şeklinde karşımıza çıktığını görebiliriz. Örnek olarak "DE 0000 1111 1111 0000 00000000" şeklinde bir IBAN numarasını bulmak istediğimizi varsayalım. Bu

durumda boşluklarla ayrılan her bir ifade metin olarak ".tsv" dosyasına ekleneceği için "DE", "0000", "1111",.. ifadeleri farklı elemanlar olarak eklenir. Aynı durum zaman zaman fatura tarihi için de geçerlidir. Örnek olarak birçok faturada, fatura tarihi "21.08.2020" gibi tek bir ifade olarak geçerken bazı faturalarda da "13 Eylül 2017" gibi 3 ifadeli bir yapı olarak karşımıza çıkmaktadır. Bu problemin çözümü için kullanılan çoklu N-gram modeliyle satır üzerindeki ifadelerin sadece kendilerinin değil aynı zamanda diğer kelime veya ifadelerle olan kombinasyonlarının da eklenmesi sağlanmıştır (Flor, 2013). Görüntü formatındaki faturalardan N-gram bilgilerinin çıkartılma süreci Şekil 5.11'de görülebilir.

N-gram sistemin nasıl çalıştığının daha iyi anlaşılması içi "Dokuz Eylül Üniversitesi" kelime dizisinin N-gramlarını elde edelim.

Tablo 5.2 Örnek kelime dizisinin N-gram yapısı

1-gramlar	"Dokuz","Eylül","Üniversitesi"
2-gramlar	"Dokuz Eylül","Eylül Üniversitesi"
3-gram	"Dokuz Eylül Üniversitesi"

Bu dizilimde ilk olarak 1-gram'lar elde edilir. Yani her bir kelime tek tek alınır. Sonrasında kelimelerin ikili kombinasyonlar, bulundukları sıraya göre yapılır. "Dokuz Eylül" ve "Eylül Üniversitesi" şeklinde 2 adet kombinasyon mevcuttur. Son olarak Tablo 5.2'de de görüldüğü gibi 3-gram olarak kelime dizisinin kendisi alınmıştır. Böylelikle 3 ifadeli bir kelime dizisinden 6 farklı N-gram durumu hesaplanmış olur. Bu işlemle birlikte herhangi bir birleşim kombinasyonu kaçırılmaz. k sayıda kelimeden oluşan bir diziden en fazla w uzunluklu oluşturulabilecek N-gramların toplam sayısını hesaplamak için Denklem 5.3 kullanılabilir.

$$C_{n-gram} = \sum_{i=1}^{w} \big(k - (i-1)\big) = k \cdot w - \frac{(w-1) \cdot w}{2} = \frac{2kw - w^2 + w}{2} \qquad (5.3)$$

Tablo 5.2'deki örnek üzerinden örnek bir hesaplama yapılacak olursa,

(1-gram + 2-gram + 3-gram) = $\frac{2.3.3-3^2+3}{2} = 6$

(1-gram + 2-gram) = $\frac{2.3.2-2^2+2}{2} = 5$

(1-gram) = $\frac{2.3.1-1^2+1}{2} = 3$

Çoklu N-gram yöntemini kullanarak hem ara dizilimlerin tamamı hem de ana dizilim elde edilmiş olacaktır. Bu yöntemin sağlıklı olarak çalışabilmesi için belirlenmesi gereken en kritik parametre maksimum N-gram uzunluğudur. Faturalardan elde etmek istediğimiz bütün alanlar analiz edildiğindi en uzun N-gram dizilimine IBAN alanında ihtiyaç olduğu görülmüştür. Farklı farklı durumlar olsa da maksimum 6-gram dizilimiyle IBAN değerine ulaşılmıştır. Bu sebepten dolayı w parametresi 6 olarak belirlenmiştir. Ancak bazı durumlarda satırdaki toplam kelime sayısı, belirlediğimiz maksimum N-gram sayısından daha düşük olabilir. Bu gibi durumlarda $w = k$ olarak kabul edilerek N-gram üretimine devam edilir.

Maksimum N-gram değerinin elde edilmek istenen alanlara göre değişiklik göstereceği elbette unutulmamalıdır. Nitekim tez kapsamında yalnızca fatura tarihi ve fatura numarası tahmin edilecek olsaydı w değeri 3 olarak belirlenirdi. Bu değeri olabilecek en düşük seviyede tutarak N-gram yapılarını oluşturmak, özellikle ".tsv" dosyasından ".csv" dosyası oluşturma ve akabinde bunların işlenerek tahmin işlemi için modele gönderilmesi aşamalarında zaman kaybını ciddi şekilde önleyecektir.

5.3 Özniteliklerin Oluşturulması

N-gram oluşturma işlemi tamamlandıktan sonra eğitimde kullanılacak verilerin öznitelikleri belirlenir. Bu aşama için öncelikle optik karakter tanıma işlemi sonucunda elde edilen ".tsv" dosyası içindeki özellikler kullanılır. Tablo 5.1'de görülen özelliklerden her bir ifadenin koordinat bilgileri, satır sayısı, sayfa sayısı, genişlik ve yükseklik bilgileri alınarak N-gramların olduğu ".csv" dosyasına öznitelik bilgisi olarak eklenir. Sonrasında Tablo 5.3'te görülen, eğitim için gerekli olabilecek farklı öznitelikler hesaplanarak sisteme dahil edilir.

Tablo 5.3 N-gram yapısı için oluşturulan öznitelikler

N-gramın özniteliği	**Açıklaması**
RawText	N-gramın metni
TextPattern	N-gramın şablonu
IsFirstStr	N-gramın ilk harfinin String olma durumu
IsFirstInt	N-gramın ilk harfinin Integer olma durumu
IsFirstSpc	N-gramın ilk harfinin özel karakter olma durumu
IsLastStr	N-gramın son harfinin String olma durumu
IsLastInt	N-gramın son harfinin Integer olma durumu
IsLastSpc	N-gramın son harfinin özel karakter olma durumu
StrCount	N-gram içerisindeki toplam String sayısı
IntCount	N-gram içerisindeki toplam Integer sayısı
SpcCount	N-gram içerisindeki toplam özel karakter sayısı
WordCount	N-gramın toplam kelime sayısı
CharCount	N-gramın toplam harf sayısı
Left	N-gramın sol mesafesi (X koordinatı)
Top	N-gramın üst mesafesi (Y koordinatı)
LeftMargin	N-gramın sayfaya oransal sol mesafesi
TopMargin	N-gramın sayfaya oransal üst mesafesi
HasDigit	N-gram içerisinde Integer olma durumu
FirstQuarter	N-gramın sayfanın ilk çeyreğinde olma durmu
SecondQuarter	N-gramın sayfanın ikinci çeyreğinde olma durmu
ThirdQuarter	N-gramın sayfanın üçüncü çeyreğinde olma durmu
FourthQuarter	N-gramın sayfanın dördüncü çeyreğinde olma durmu
LineNo	N-gramın kaçıncı satırda olduğu
PageNo	N-gramın kaçıncı sayfada olduğu

Öznitelik olarak Tablo 5.3'teki özellikler doğrudan ilgili N-gram değerlerine göre oluşturulur. Bu öznitelikler dışında ilgili N-gramı dolaylı olarak yansıtan bazı özellikler (Levenshtein ve Jaro-Winkler benzerlik sonuçları) ve genel bilgiler Tablo 5.4'te gösterilmiştir.

Tablo 5.4 Eğitimde kullanılan genel öznitelikler

Öznitelik	**Açıklaması**
PageWidth	Sayfa genişliği
PageHeight	Sayfa yüksekliği
InvoiceDateLevDistance	Fatura tarihi için kullanılan Levenshtein Uzaklığı
DeliveryDateLevDistance	Teslimat tarihi için kullanılan Levenshtein Uzaklığı
DueDateLevDistance	Vade tarihi için kullanılan Levenshtein Uzaklığı
InvoiceNoLevDistance	Fatura numarası için kullanılan Levenshtein Uzaklığı
TotalGrossLevDistance	Toplam tutar için kullanılan Levenshtein Uzaklığı
TotalNetLevDistance	Net tutar için kullanılan Levenshtein Uzaklığı
VatAmountsLevDistance	Vergi tutarı için kullanılan Levenshtein Uzaklığı
IBANLevDistance	IBAN için kullanılan Levenshtein Uzaklığı
SenderVatNoLevDistance	Vergi numarası için kullanılan Levenshtein Uzaklığı
InvoiceDateJWDistance	Fatura tarihi için kullanılan Jaro-Winkler Uzaklığı
DeliveryDateJWDistance	Teslimat tarihi için kullanılan Jaro-Winkler Uzaklığı
DueDateJWDistance	Vade tarihi için kullanılan Jaro-Winkler Uzaklığı
InvoiceNoJWDistance	Fatura numarası için kullanılan Jaro-Winkler Uzaklığı
TotalGrossJWDistance	Toplam tutar için kullanılan Jaro-Winkler Uzaklığı
TotalNetJWDistance	Net tutar için kullanılan Jaro-Winkler Uzaklığı
VatAmountsJWDistance	Vergi tutarı için kullanılan Jaro-Winkler Uzaklığı
IBANJWDistance	IBAN için kullanılan Jaro-Winkler Uzaklığı
SenderVatNoJWDistance	Vergi numarası için kullanılan Jaro-Winkler Uzaklığı

5.3.1 N-gram Özniteliklerinin Bulunması

Tablo 5.3'te gösterilen özniteliklerin elde edilmesi için ilk olarak N-gramların şablonlarının elde edilmesi gerekmektedir. Her bir N-gramın içinde geçen büyük harf

için “X”, küçük harf için “x”, rakam için “0” ve özel karakter için “?” ifadeleri kullanılarak şablon oluşturulur. Yani elimizde “InvoiceNo: 101” değerine sahip 2-gram ifadesini şablonunu çıkartmak istersek, şablon değeri “XxxxxxxXx? 000” şeklinde olacaktır. N-gram şablonlarının çıkartılmasının nedeni bu ifadelerin yapılarını modelimize en iyi şekilde öğretmektir. Özellikle bilgilerini tahmin etmek istediğimiz fatura tarihi gibi bazı alanların sabit olmasa da belirli şablon yapıları mevcuttur. Bazı faturalarda yalnızca sayı ve özel karakterden oluşan fatura numaraları olurken (örnek olarak 02.05.2020) bazılarında küçük veya büyük harf olabilmektedir (8 Ekim 1993). Ancak bütün fatura yapıları analiz edildiğinde görülmüştür ki içerdiği toplam karakter sayısı en az 6 olurken, mutlaka içerisinde de sayı geçmektedir. Modelin en azından bu durumları sağlıklı olarak öğrenebilmesinde bu şablonların önemi büyüktür.

N-gram şablonlarının çıkartılması tamamlandıktan sonra bu şablonların öznitelik olarak nasıl ekleneceğinin belirlenmesi gerekir. Bu kararın verilmesinde verilerin sayı ve tür çeşitliliği oldukça önemli bir etkendir. Eldeki veri sayısı ve tür çeşitliliği fazla olduğunda elde edilen bu şablonları doğrudan kullanmak fazla maliyetli bir işlem olabilir. Onun yerine şablonda gerekli olan bilgilerin çıkartılarak, öznitelik olarak eklenmesi daha doğru ve efektif bir çözüm olacaktır. Şablonun ilk ve son karakterlerinin durumu (“X”,”x”,”0”, “?”), şablon içerisindeki toplam string, integer ve özel karakter sayıları, toplam karakter sayısı ve şablonun kaç kelimeden meydana geldiği gibi kritik bilgiler elde edilerek öznitelik olarak belirlenir.

Az sayıda veri ile veya çeşitliliği az olan bir veri seti ile çalışılacak olursa elde edilen bu şablon değerleri doğrudan modele tekil sıcak kodlama (One Hot Encoding) ile aktarılabilir (Potdar, Pardawala ve Pai, 2017). Ancak bu çalışmada veri çeşitliliği çok fazla olduğu için bu yöntemi kullanmak doğru olmaz. Bunun yerine şablon içerisindeki her bir karakterin potansiyel durumlarından yola çıkarak belirli bir şablon uzunluğu için karakter bazlı tekil sıcak kodlama işlemi yapılabilir. Şablon içerisindeki karakterler potansiyel olarak “X”, “x”, “0”, “?” ve boşluk olma durumları olmak üzere 5 farklı durumda bulunabilirler. Belirlenecek maksimum şablon uzunluğuna göre her karakter için 5 farklı durum olacak şekilde işlenir. Örnek olarak maksimum şablon uzunluğu 40 olarak alındığında toplamda 200 adet öznitelik sütunu, şablonu temsil

etmek adına oluşturulmuş olur. Bu çalışmada iki farklı durum için yapılan denemeler sonrasında elde edilen başarı değerlerinin birbirlerine yakın olmasından dolayı daha az maliyetli olan ilk yöntem tercih edilmiştir. Ancak farklı çalışmalarda farklı durumların daha efektif sonuçlar verebileceği unutulmamalıdır.

Yukarıda anlatılan tekil sıcak kodlama genel olarak kategorik değişkenlerin ikili (binary) vektör kullanılarak temsil edilmesi anlamına gelmektedir. Bu işlem, ilk önce kategorik değişkenlerin sayısı kadar 0 veya 1 değerli bileşenden oluşan vektörler oluşturulmasını gerektirir. Bu vektörlerde, her bir kategori için yalnızca o indekse karşılık gelen değer 1 olurken diğer değerler 0 olur. Örnek olarak "No: 101" ifadesinin karakter bazlı tekil sıcak kodlama işlemi Tablo 5.5'te görülmektedir.

Tablo 5.5 Örnek tekil sıcak kodlama

	X	x	0	?	" "
X (N)	1	0	0	0	0
X (o)	0	1	0	0	0
? (:)	0	0	0	1	0
" " ( )	0	0	0	0	1
0 (1)	0	0	1	0	0
0 (0)	0	0	1	0	0
0 (1)	0	0	1	0	0

Bu özellikler dışında her N-gramın sol dıştan boşluğu (left margin), sağ dıştan boşluğu (right margin) değerleri, n-gramların sayfanın sol ve sağına göre hizalama değerleri, bu hizalama değerlerine göre oluşturdukları grup sayıları, N-gram dizilimlerinin soldaki, sağdaki, üstteki ve alttaki en yakın tekli N-gram dizilimi gibi değerler öznitelik olarak eklenmiştir. Fatura üzerinden elde edilmesi amaçlanan bölgeler incelendiğinde, bu değerlerin büyük oranda anahtar kelimelere sahip olduğu görülmüştür. Özellikle hedef N-gramın yani etiketli verilerin özellikle en yakın sol N-gram ve en yakın üst N-gram değerinin büyük olasılıkla bu anahtar kelimelere denk geldiği yapılan analizler sonucunda elde edilmiştir. Bahsedilen anahtar kelimeler

hedef N-gramı niteleyen, açıklayan sözler olarak düşünülmelidir. Örnek olarak "Rechnungnummer : 1010" N-gram dizilimi incelenecek olursa, burada hedef N-gram "1010" ifadesidir. Bu ifadeyi açıklayan anahtar kelime "Rechnungnummer" N-gramı olacaktır.

Bu çalışmada kullanılan faturalar çoğunlukla Almanca olduğu için potansiyel anahtar kelimelerin belirlenmesi için 1-gram yapısını kullanmak yeterli gözükmektedir. Veri seti dilinin Türkçe veya İngilizce olması durumunda 2-gram kullanımı şarttır. Bunun nedeni, bu dillerde hedeflenen potansiyel anahtar kelimeler çoğunlukla 2 kelimeden meydana gelmesidir. "Fatura Numarası", "Fatura Tarihi" ya da "Invoice Number", "Invoice Date" gibi örneklerle anlaşılacağı üzere anahtar kelimeler için 2-gram yapısı gereklidir. Ancak Almanca'da bu yapılar "Rechnungsnummer", "Rechnungsdatum", "Belegdatum" gibi örneklerle görülebileceği gibi tek kelime ile ifade edilebilmektedir. Buna rağmen aykırı durumlar da mevcuttur.

Zwischensumme	**EUR**	**22,00**
Porti & Papiere	EUR	1,50
Nettobetrag	**EUR**	**23,50**
19,00 % MwSt	**EUR**	**4,47**
Rechnungsbetrag	**EUR**	**27,97**
Gesamtzahlbetrag	**EUR**	**27,97**

Şekil 5.12 Aykırı durum örneği – 1 (Kişisel arşiv, 2020)

Nettobetrag:	€	3.399,64
USt 21%:	€	0,00
Gesamtbetrag:	€	3.399,64

Şekil 5.13 Aykırı durum örneği – 2 (Kişisel arşiv, 2020)

Şekil 5.12 ve Şekil 5.13'te aykırı durumlar görülebilir. Şekil 5.12'de potansiyel anahtar kelime olması gereken "Nettobetrag" ifadesinin önüne "EUR" yazısı gelmiş, Şekil 5.13'te ise para birimi işareti gelmiştir. Genellikle bu ifadeler tutarların sağında yer almasına rağmen bu tip örnekler de göz ardı edilemez. Ayrıca bazı durumlarda görüntü kirliliklerinden dolayı Tesseract, anahtar kelime ve etiketli veri arasında gerçekte olmayan bir noktalama işaretini veya karakteri varmış gibi sonuç döndürebilmektedir. Bu sebeplerden dolayı en yakın soldaki 2 kelimeyi de potansiyel anahtar kelime olarak değerlendirmek daha doğru bir yaklaşım olacaktır.

5.3.2 Kelimeler Arası Uzaklıkların Bulunması

N-gramların etrafındaki sol, sağ, üst ve sol-2 N-gram değerleri potansiyel anahtar kelime olarak listeye eklenmesinin ardında çözülmesi gereken bazı problemler ortaya çıkmıştır. Öncelikle veri seti çeşitliliği çok fazla olduğu için farklı farklı kelimeler, anahtar kelime olarak kullanılmaktadır. Örnek verilecek olursa, fatura numarasının tahmin edilmek istendiği bir faturada anahtar kelime "invoicenummer" olurken diğerinde "belegnummer" bir diğerinde ise "rechnungnummer" olabilmektedir. Bu yüzden sabit bir kelime belirleyerek potansiyel anahtar kelimeleri bu kelimeyle kıyaslayarak hareket etmek doğru bir çözüm yöntemi olmayacaktır. Veri setinin geneline göre hareket etmek için eldeki etiketli verilerin tamamının potansiyel anahtar kelimelerinin frekanslarına bakılır. Belirli eşik değeri geçen kelimeler kontrol anahtar kelimesi olarak seçilir. Bu kelimelerin birleştirilmesi sonucunda da kontrol anahtar kelime listesi oluşturulur. Böylelikle veri setine göre dinamik bir yapı oluşturulmuş olur. Ancak hala daha çözülmesi gereken bazı problemler mevcuttur. Bazı faturalarda potansiyel anahtar kelimelerin kısaltılmış halleri yazılmaktadır. Fatura üzerinde "InvoiceNo:" şeklinde kısaltmalar yapılabilmektedir. Potansiyel anahtar kelimelerin ne olabilecekleri her ne kadar öngörülebilse de bu kısaltmaların ne olabileceğini öngörmek mümkün değildir. Yine fatura numarası üzerinden örnek verilecek olursa "InvoiceNum", "InvoiceNo : ", "InvoiceNo.:", "Inv.Nummer:" ve daha sayılabilecek birçok varyasyon mevcuttur. Ayrıca problemlerden bir diğeri ise anahtar kelimelerin tam olarak doğru yazılsa bile potansiyel optik karakter tanıma işlemi hatalarından ötürü bir harf yanlış olarak okunabilir ("1nvo1ceNummer") veya olmayan bir

noktalama işareti kirlilikten dolayı sisteme eklenmiş olabilir. Anahtar kontrol listesi dinamik olarak frekansa göre belirlense bile hala daha yeterince esnek olmadığı ortadadır. Yapıya bu esnekliği kazandırabilmek için kelimeler arası uzaklık ölçüm yöntemlerinin kullanılması gerekmektedir.

Kelimeler arasında kesin bir eşleşme aramak yerine kelime uzaklık ölçüm yöntemleri ile iki kelimenin birbirlerine ne kadar benzer olduğu belirlenebilir. Burada bahsedilen kelimelerden ilki zaten N-gram yapısından elde ettiğimiz potansiyel anahtar kelimelerdir. İkinci kelime ise veri setinin tamamı baz alınarak frekansa göre bulunan kontrol anahtar kelimelerdir. Kontrol listesindeki her bir kelimenin potansiyel anahtar kelime ile arasındaki benzerlik ölçülür. Bu benzerliğin maksimum olduğu değer, yapıya öznitelik olarak eklenir. Kelimelerin uzaklık ve benzerlik değerleri aynı yöntemle elde edilir. 0 ile 1 arasındaki uzaklık değerinin 1'den çıkartılması sonucunda benzerlik değeri elde edilir. Bu çalışmada kelimeler arasındaki mesafeyi ölçebilmek için Levenshtein ve Jaro-Winkler olmak üzere 2 farklı yöntem kullanılmıştır.

5.3.2.1 Levenshtein Yöntemi

Kelimeler arasındaki mesafeyi tespit etmek için kullanılan yöntemlerden ilki Levenshtein yöntemidir (Haldar ve Mukhopadhyay, 2011). Levenshtein uzaklığının hesaplanmasında değiştirmek (replace), eklemek (insert) ve silmek (delete) olmak üzere 3 temel işlem vardır. Buradaki amaç, karşılaştırılan bir kelimeden diğer kelimeye yukarıdaki temel işlemleri en az sayıda yaparak ulaşabilmektir. Herhangi a ve b stringlerinin arasındaki Levenshtein mesafesi Denklem 5.4'teki gibi özyinelemeli fonksiyon olarak hesaplanır (Schulz ve Mihov, 2002).

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j), & if\ \min(i,j) = 0, \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & ,aksi\ durumda. \end{cases} \quad (5.4)$$

Denklem 5.4'teki $lev_{a,b}(i,j)$; a dizisinin ilk i karakteri ile b dizisinin ilk j karakteri arasında Levenshtein mesafesini gösteriyor. $1_{(a_i \neq b_j)}$ belirteç fonksiyonudur, yani $a_i =$

b_j olduğunda 0'a, aksi durumda 1'e eşit olur. Formül içindeki min altında ilk satır silme işlemine ikinci satır ise ekleme işlemine karşılık gelmektedir.

Adım adım iki kelimenin Levenshtein uzaklıklarını hesaplamak için "UZAK" ve "KAZAK" kelimelerini örnek olarak alalım (Tablo 5.6).

Tablo 5.6 "Kazak" ve "Uzak" kelimelerinin Levenshtein uzaklık hesabı

	"	K	A	Z	A	K
"	0	1	2	3	4	5
U	1	1	2	3	4	5
Z	2	2	2	2	3	4
A	3	3	2	3	2	3
K	4	3	3	3	3	2

"KAZAK" kelimesini "UZAK" kelimesine dönüştürerek, kelimelerin arasındaki Levenshtein mesafesi bulunacak olursa ilk olarak "KAZAK" kelimesinden "K" harfinin çıkartılması gerekir. Bunun için temel işlemlerden silme işlemi gerçekleştirilir. Sonrasında "A" harfinin "U" harfine dönüşüm işlemi gereklidir. Kalan "Z", "A" ve "K" harfleri her iki kelimede de aynı olduğu için bu harflere herhangi bir işlem uygulanmaz. Böylelikle iki kelime arasındaki Levenshtein uzaklığı 2 olarak bulunur.

Hedef N-gramın çevresindeki anahtar kelimelerin (özellikle sol, sol-2 ve üst 1-gram ifadeleri) elde edilmesinin ardından kontrol anahtar kelimelerle karşılaştırılarak Levenshtein uzaklıklarının bulunması için kullanılan algoritmanın kaba kodu Şekil 5.14'te gösterilmiştir.

```
def (pot_keys, control_keys):
    result = {}
    for i in pot_keys:
        dist_list = []
        for j in  control_keys:
            dist = calc_levenshtein(i, j)
            dist_list.append(dist)
        result[i] = min(dist_list)
    return result
```

Şekil 5.14 Levenshtein uzaklıklarının hesaplanması

5.3.2.2 Jaro-Winkler Yöntemi

İki kelime arasındaki mesafeyi ölçmek için kullanılan bir diğer metot Jaro-Winkler yöntemidir. Jaro-Winkler uzaklık değerinin bulunabilmesi için öncelikle Jaro değerinin elde edilmesi gerekmektedir (Jaro, 1989).

$$sim_j = \begin{cases} 0 & ,if\ m = 0, \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & ,otherwise \end{cases} \quad (5.5)$$

Jaro değerinin hesaplanması için Denklem 5.5 kullanılır. Burada $|s_i|$ ifadesi kelimelerin uzunlukları, m kelimelerin karşılaştırılması sonucunda eşleşen karakter sayısı ve t transpoz değerinin yarısı olmak üzere sim_j ile Jaro mesafesi elde edilir.

$$\left\lfloor \frac{\max(|s_1|, |s_2|)}{2} \right\rfloor - 1 \quad (5.6)$$

Eşleşen karakter sayısının doğru olarak bulunabilmesi için öncelikle eşleşme mesafesinin tespit edilmesi gerekir. Kelime içindeki hedef karakterin indeksi bulunur. Sonrasında mesafe ölçümü yapılacak diğer kelimenin o indeksi merkez olacak şekilde eşleşme mesafesine göre karakter taraması yapılır. Uygun karakterin bulunması durumunda eşleşme gerçekleşir.

Örnek olarak 6 karakterli “BXXXXX” ile “XBXXXX” ve “XXXXBX” kelimeleri arasında “B” harfi için bir eşleşme durumu olup olmayacağını kontrol etmek isteyelim. Bu durumda öncelikle Denklem 5.6 ile eşleşme mesafesi 2 olarak bulunur. İlk durumda “XBXXXX” kelimesinde 1 birimlik eşleşme mesafesi içinde “B” karakteri tespit edildiği için iki kelime arasında bu karakterin eşleşmesi gerçekleşir. Ancak “XXXXBX” ifadesine bakıldığında “B” karakteri eşleşme mesafesinin dışında kaldığı için herhangi bir eşleşme işlemi gerçekleşmez. “CRATE” ve “TRACE” kelime arasında eşleşen karakterlere genel olarak bakıldığında “R”, “A” ve “E” karakterlerinin eşleşmesi gerçekleşir. İki kelimede de “C” ve “T” harfleri olmasına rağmen eşleşme mesafesi içinde kalmadığından dolayı bu karakterler için eşleşme olmaz.

Kelimeler arasındaki karakterlerin eşleşme mantığının anlaşılmasının ardından “BAKER” ve “BAKRE” kelimelerinin Jaro değeri hesaplanacak olsun. İlk 3 harf olan “B”, “A” ve “K” karakterleri doğrudan eşleşmiştir. Bunun dışında “E” ve “R” karakterleri de eşleşme mesafesi içerisinde kaldığı için eşleşme gerçekleşmiştir. Böylece m değeri 5 olurken t değeri de 1 (2/2 = 1) olarak elde edilir. Her iki kelimenin de uzunluğu 5 birimdir. Bu bilgiler ile Denklem 5.5 kullanıldığında elde edilecek Jaro değeri $\frac{1}{3}.\left(\frac{5}{5}+\frac{5}{5}+\frac{5-1}{5}\right) = 0{,}9333$ olarak elde edilir. Bu değerin 1’e yaklaşması kelimeler arasındaki benzerliğin arttığını gösterir. Jaro değerinin bulunmasının ardından Denklem 5.7 kullanılarak Jaro-Winkler benzerlik değerine geçiş yapılır.

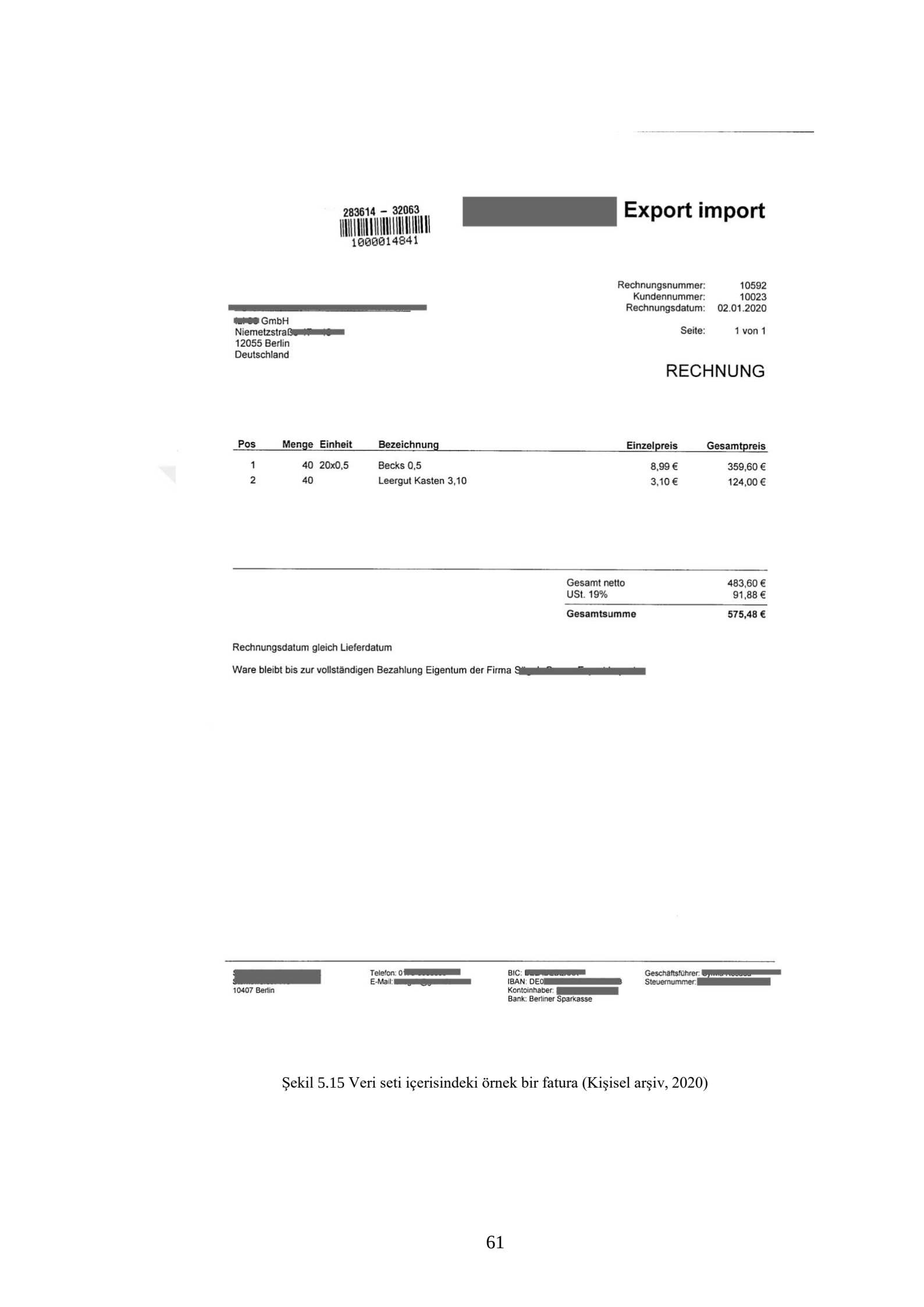
$$sim_w = sim_j + l \cdot p(1 - sim_j) \tag{5.7}$$

Denklem 5.7’de sim_j ile ifade edilen değer Jaro benzerlik değeri, l en fazla 4 olacak şekilde iki kelimenin ön eklerinin eşleşme sayısı, p ölçekleme faktörü olmak üzere sim_w Jaro-Winkler benzerlik değeri elde edilir. Burada ölçekleme faktörü Jaro-Winkler hesaplamalarında varsayılan olarak 0,1 değerinde kullanılır. Bu çalışma kapsamında kelimelerin Jaro-Winkler benzerlik mesafesi hesaplamalarında p değeri 0,1 olarak alınmıştır. Bu değer 0,25’ten büyük olmamalıdır. Aksi durumda benzerlik değeri 1’den büyük olur.

Yukarıda Jaro benzerlik mesafesi hesaplanan örneğin Jaro-Winkler hesabı da yapılacak olursa Denklem 5.7 kullanılır. Jaro benzerlik değeri 0,9333 elde edilmişti. Ölçekleme faktörü ise 0,1 olarak alınmıştır. İki kelime karşılaştırıldığında "B", "A" ve "K" olmak üzere ilk 3 karakter eşleşmiştir. Dolayısıyla l değeri 3 olur. Bu veriler Denklem 5.7'de yerine konulursa sim_w değeri 0,9533 olarak bulunur. Bu değer iki kelime arasındaki benzerliği ifade etmektedir. İki kelime arasındaki uzaklık hesaplanmak istenirse $d_w = 1 - sim_w$ formülü kullanılır. Bu durumda uzaklık değeri 0,0467 olur.

5.3.2.3 Uzaklık Hesaplama Yöntemlerinin Faturalarda Kullanımı

Faturalar üzerinde, yukarıda anlatılan her iki yöntem ile mesafe uzaklıkları hesaplanmıştır. Şekil 5.15 ile eğitimlerde kullanılan örnek bir fatura görülmektedir. Bu faturadaki fatura numarası için potansiyel anahtar kelimelerin hesaplamasını yapacak olursak, ilk olarak bu etiket için bir kontrol anahtar kelime listesinin frekansa göre hazırlanması gerekir. Ardından faturadaki her bir N-gram değeri için kontrol anahtar kelimelerle, bu N-grama ait potansiyel anahtar kelimelerin arasındaki benzerlikler hesaplanır. Şekil 5.16 soldaki görselde ilk olarak N-gramın etrafındaki 1-gram ifadelerinin çıkartılması gözükmektedir. Bu aşamada henüz bir benzerlik hesabı yoktur. Bu aşamada yalnızca bütün potansiyel anahtar kelimelerin 1-gram değerleri ve koordinat bilgiler mevcuttur. Dolayısıyla sistem için henüz bu kelimeler anlamlandırılmamıştır.



283614 – 32063
1000014841

[redacted] Export import

Rechnungsnummer: 10592
Kundennummer: 10023
Rechnungsdatum: 02.01.2020

Seite: 1 von 1

[redacted]
[redacted] GmbH
Niemetzstraß[redacted]
12055 Berlin
Deutschland

RECHNUNG

Pos	Menge	Einheit	Bezeichnung	Einzelpreis	Gesamtpreis
1	40	20x0,5	Becks 0,5	8,99 €	359,60 €
2	40		Leergut Kasten 3,10	3,10 €	124,00 €

Gesamt netto	483,60 €
USt. 19%	91,88 €
Gesamtsumme	**575,48 €**

Rechnungsdatum gleich Lieferdatum

Ware bleibt bis zur vollständigen Bezahlung Eigentum der Firma S[redacted]

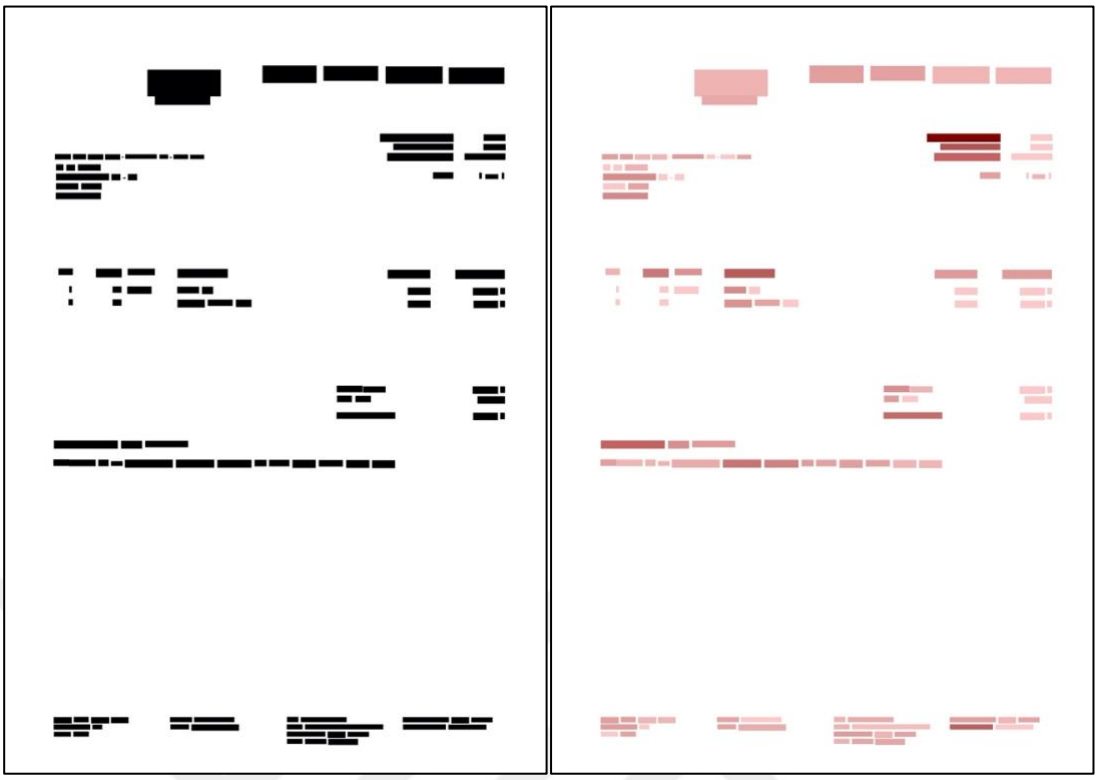
[redacted]
10407 Berlin

Telefon: 0[redacted]
E-Mail: [redacted]

BIC: [redacted]
IBAN: DE0[redacted]
Kontoinhaber: [redacted]
Bank: Berliner Sparkasse

Geschäftsführer: [redacted]
Steuernummer: [redacted]

Şekil 5.15 Veri seti içerisindeki örnek bir fatura (Kişisel arşiv, 2020)



Şekil 5.16 Örnek fatura için N-gram ve potansiyel anahtar kelime dağılımı

Şekil 5.16 sağdaki görselde ise hesaplamalar sonucunda benzerliklerin elde edilmesi görülmektedir. Görseldeki renkler koyulaştıkça potansiyel anahtar kelimelerin, kontrol anahtar kelime listesindeki kelimelerle olan benzerliklerin artmasını, rengin soluklaşması ise aradaki benzerliğin azalmasını göstermektedir.

5.4 Eğitim Aşaması

Bütün öznitelikler hazırlandıktan sonra eğitim aşamasına geçilir. Eğitimde tek bir ana model kullanılarak bütün etiketli veriler tek bir defa da tahmin edilebilir. Ancak bu yöntemin bazı dezavantajları vardır. Örnek olarak fatura numarası için sayfanın hangi bölgesinde bulunduğunun önemi büyüktür. Nitekim hemen hemen bütün faturalarda fatura numarası sayfanın üst bölgelerinde yer alır. Ancak aynı kritik durum fatura tutarı için geçerli değildir. Fatura tutarı sayfanın en alt kısmında bulunabileceği gibi faturanın 2 veya daha fazla sayfa içermesi durumunda sayfanın ortasında veya en üst bölgelerinde yer alabilir. Haliyle fatura numarası için y koordinat değerinin önemi fazlayken fatura tutarı için aynı durum geçerli olmaz. Aynı şekilde fatura numarası

sayfanın üst bölgesinde sağda, ortada veya solda bulunabilir. Ancak fatura tutarı çok büyük ihtimalle sayfanın sağ tarafında bulunmaktadır. Dolayısıyla fatura tutarı için *x* koordinat bilgisi fatura tarihine göre çok daha fazladır. Bu sebeplerden dolayı tek bir modelde bütün öznitelikleri kullanarak eğitim yaptırmak, maksimum başarıya ulaşmayı engelleyecektir. Ayrıca her model için farklı eğitim gerçekleştirmek her bir etiketli veriyi farklı farklı yöntemlerle tahmin etmeyi mümkün kılar. Bu sebeplerden dolayı her bir etiketli veri için farklı eğitimler gerçekleştirilmiştir. Ana eğitim modeli için Rassal Orman, Gradyan Yükseltme Makinesi, Aşırı Gradyan Yükseltme, K-En Yakın Komşu, AdaBoost ve Karar Ağacı algoritmaları kullanılmıştır. Eğitimde kullanılan modellerin tamamı Bölüm 3'te ayrıntılı olarak anlatılmıştır.

Ana eğitim modelleri hazırlandıktan sonra bazı etiketli verilerin tahmini için 2.katman oluşturulmuştur. Fatura üzerindeki bazı bilgiler mutlaka faturada bulunmak zorundadırlar. Bütün veri seti üzerinde analiz yapıldığında fatura numarası ve fatura tarihi bilgilerinin bulunmadığı hiçbir faturanın olmadığı görülmüştür. Bu durumda fatura tarihi ve fatura numarası için eğitilen model bir tahmin yaptığında ve bu tahmindeki olasılık değerlerinin tamamı %50'den düşük olduğunda aslında modelin hatalı bir tahminde bulunduğu anlaşılır. Çünkü her faturada bu değerlerin mutlaka olacağını bildiğimiz için tahmin edilen N-gram olasılıklarının en azından birinin değeri %50'den büyük olmalıdır. Haliyle bu gibi durumlarda modelin hatalı tahmin yaptığını anlayabiliriz. Diğer etiketli veriler için aynı kontrolü yapmak bu yöntemle mümkün olmaz. Modelin tahmin olasılık değeri düşük olduğunda, faturada o etiketli verinin olmadığı anlamına gelebilir. Bizim için modelin faturadaki etiketli veriyi doğru tahmin etmesi kadar, faturada bu verinin olmadığını tahmin etmesi de oldukça önemlidir. Ancak fatura tarihi ve fatura numarası gibi alanlar için öngörülebilir hata durumları karşısında (olasılık değerinin %50 altında olması durumu) ikinci katman oluşturulabilir.

İkinci katmanın sağlıklı olarak oluşturulabilmesi için öncelikle eğitimin nasıl gerçekleştirildiğinin anlaşılması gerekir. Özellikle büyük veri setleri ile çalışıldığında, model tarafından aykırı gözlem olarak nitelendirilebilecek fazla sayıda veri olabilir. Birçok model buna karşı önlemlerini kendi içlerinde barındırsalar da özellikle aşırı

öğrenme veya ezberleme (overfitting) olarak nitelendirilen duruma yakalanmamak için aykırı gözlemlere verilen önemi azaltırlar. Haliyle genel başarı oranı yüksek olsa da görece az sayıdaki farklı fatura içi başarısız tahminler gerçekleştirilir. Bazı durumlarda bu tahminler göz ardı edilebilir. Ancak modelimizin yanlış tahminlerde bulunduğunu öngörebileceğimiz hallerde ikinci bir tahmin mekanizması oluşturabiliriz. Bunun için ilk eğitim sonucunda yanlış tahmin edilen veriler tespit edilir. Bu veriler ile yeni belirlenecek olan ikinci katman model eğitilir.

Yapılan analizler sonrasında bu problem için makine öğrenmesi teknikleri, standart yapay sinir ağları ve evrişimsel sinir ağları arasında en uygun modelin Bölüm 4'te de anlatılan evrişimsel sinir ağları olduğu görülmüştür. İkinci eğitim daha çok aykırı gözlem verileriyle gerçekleştirildiği için bu modelin ana model gibi kullanılması doğru olmaz. Bu modelin ne zaman kullanılıp ne zaman kullanılmayacağını anlamak için ilk modelin tahmin değerlerini analiz etmek gerekir. İlk modelin tahmin ettiği N-gramın olasılık değeri belirlenen eşik değerin altında kaldığında 2.modelin devreye girerek tahmin yapar. Böylelikle aykırı gözlem verilerine göre ekstra tahmin işlemi yapılmış olur.

Fatura üzerinde elde etmek istediğimiz alanlardan vergi numarası ve IBAN alanları için de 2.katman oluşturulabilir. Bu alanların yapıları çok spesifik olduğu için tahmin başarıları oldukça yüksektir. Ancak özellikle bu alanlarda sıklıkla optik karakter tanıma sürecinde hatalar meydana gelmektedir. Örnek olarak fatura üzerindeki IBAN ifadesi "DE 50 1000 …" şeklinde olduğunu varsayalım. Ancak OCR işlemleri sonucunda bu ifade "DE SO 1000 …" olarak elde edilebilmektedir. Bu tarz durumlarda yapay zekanın tahmin ettiği N-gram teknik olarak doğru olsa da gerçek IBAN ifadesini yansıtmamaktadır. Özellikle yüksek tahmin oranına sahip değerlerin herhangi bir OCR hatasına sahip olup olmadığını kontrol etmek için 2.katman kullanılabilir. Buradaki validasyon işlemi sonucunda tahmin edilen ifadelerin gerçek vergi numarası veya IBAN olup olmadığı ortaya çıkar.

BÖLÜM 6
HESAPLAMA SONUÇLARI VE DEĞERLENDİRMELER

6.1 Değerlendirme yöntemleri

Eğitim sonucunda elde edilen modeller test için ayrılan veriler kullanılarak sınanır. Ekstrem durumlar olmakla beraber genellikle eldeki veri seti %80-%70 aralığında eğitim verisi, %20-%30 aralığında ise test verisi şeklinde ayrılır. Eğitim verisi ile model eğitimi gerçekleştirilir. Sonrasında ise modelin başarısı test verileri kullanılarak ölçülür. Sınıflandırma problemlerinde model başarı değerlendirmesi için genel olarak ilk önce karmaşıklık matrisi (confusion matrix) hesaplanmaktadır (Fawcett, 2006).

Tablo 6.1 Karmaşıklık Matrisi

	Gerçek Değer	
Tahmin Değeri	TP	FP
	FN	TN

- TP : Gerçek değeri 1 olan durumun 1 olarak tahmin edilmesi (True Positive)
- FP : Gerçek değeri 0 olan durumun 1 olarak tahmin edilmesi (False Positive)
- FN : Gerçek değeri 1 olan durumun 0 olarak tahmin edilmesi (False Negative)
- TN : Gerçek değeri 0 olan durumun 0 olarak tahmin edilmesi (True Negative)

Modelin tahminleme başarısının değerlendirmesi için en basit ölçüm değeri Denklem 6.1'de tanımlanan doğruluk (accuracy) değeridir:

$$Doğruluk = \frac{TP + TN}{TP + TN + FN + FP} \tag{6.1}$$

Fakat veri setinde farklı sınıflardan olan örneklerin sayısı dengesiz olduğunda doğruluk değerini tek başına kullanmak zaman zaman yeterli olmamaktadır. Bu durumlarda kesinlik (precision), duyarlılık (recall) ve F1 skor değerleri gibi ölçümler kullanılabilir.

Kesinlik, duyarlılık ve F1 skor değerleri de Tablo 6.1 kullanılarak elde edilir. Bu değerler kullanılarak modelin genel başarısı ölçülebildiği gibi aynı zamanda gerçek durumun 1 ve 0 olması durumuna özel değerlendirmelerde yapılabilir. Özellikle eldeki verilerin eşit bir şekilde dağılmayıp, bir tarafa yoğunlaşması durumunda (verilerin büyük oranda gerçek değerlerinin 1 veya 0 olması gibi) modelin genel başarısını F1 skora bakarak değerlendirmek de doğru sonuca ulaştırmayacaktır. Bu gibi durumlarda kesinlik ve duyarlılık değerlerini ayrı ayrı analiz etmek daha doğru olacaktır. Denklem 6.2 ile duyarlılık değeri, Denklem 6.3 ile kesinlik değeri ve Denklem 6.4 ile de F1 skor değeri elde edilir.

$$Duyarlılık = \frac{TP}{TP + FN} \tag{6.2}$$

$$Kesinlik = \frac{TP}{TP + FP} \tag{6.3}$$

$$F1 = 2\frac{Duyarlılık.Kesinlik}{Duyarlılık + Kesinlik} \tag{6.4}$$

Bu tez çalışmasında modellerin başarılarının karşılaştırılması için F1 skor değerleri kullanılmıştır.

6.2 Hesaplama sonuçları

Görüntü formatındaki faturaların optik karakter tanıma, öznitelik oluşturma, kullanılacak modellerin belirlenmesi ve eğitim adımları gerçekleştirildikten sonra elde edilen sonuçlar değerlendirilerek optimum model ve öznitelik seçimi yapılır. Makine öğrenmesi bölümünde üzerinden geçilen bütün modeller aynı veri ve özniteliklerle

eğitilir. Böylelikle bu veri seti için en uygun model belirlenir. Ayrıca kelime uzaklıklarının hesaplanması için kullanılan Levenshtein ve Jaro-Winkler yöntemlerinin her ikisi de sisteme eklenmiştir. Hangi yöntemin bizim için en uygun olduğunu anlamak için iki yöntem karşılaştırmalı olarak değerlendirilmiştir. Fatura üzerinde elde etmeyi amaçladığımız bütün alanlar ayrı ayrı farklı makine öğrenme modelleri ile eğitilmiştir. Tablo 6.2 – Tablo 6.19 tablolarında eğitim sonucundaki başarılar görülmektedir.

Tablo 6.2 Fatura numarası için kesinlik, duyarlılık ve F1 skor değerleri

Model	**Kesinlik Lev.**	**Duyar. Lev.**	**F1 Lev.**	**Kesinlik J-Win**	**Duyar. J-Win.**	**F1 J-Win**
Rassal Orman	0,99	0,96	0,97	0,99	0,96	0,97
Gradyan Yükseltme Makinesi	0,96	0,91	0,93	0,96	0,90	0,93
Aşırı Gradyan Yükseltme	0,98	0,97	0,97	0,98	0,97	0,97
K-En Yakın Komşu	0,93	0,95	0,94	0,90	0,94	0,92
AdaBoost	0,94	0,89	0,92	0,94	0,88	0,91
Karar Ağacı	0,95	0,96	0,95	0,95	0,96	0,95

Tablo 6.3 Fatura numarası için karmaşıklık matrisi

Model	TP Lev.	FP Lev.	FN Lev.	TN Lev.	TP J-Win	FP J-Win	FN J-Win	TN J-Win
Rassal Orman	1718	26	70	19504	1715	26	73	19504
Gradyan Yükseltme Makinesi	1632	75	156	19455	1614	70	174	19460
Aşırı Gradyan Yükseltme	1735	38	53	19492	1726	36	62	19494
K-En Yakın Komşu	1701	135	87	19395	1679	181	109	19349
AdaBoost	1599	106	189	19424	1577	96	211	19434
Karar Ağacı	1710	99	78	19431	1710	86	78	19444

Tablo 6.4 Fatura tarihi için kesinlik, duyarlılık ve F1 skor değerleri

Model	Kesinlik Lev.	Duyar. Lev.	F1 Lev.	Kesinlik J-Win	Duyar. J-Win.	F1 J-Win
Rassal Orman	0,97	0,97	0,97	0,97	0,97	0,97
Gradyan Yükseltme Makinesi	0,93	0,95	0,94	0,93	0,94	0,94
Aşırı Gradyan Yükseltme	0,96	0,97	0,96	0,96	0,97	0,96
K-En Yakın Komşu	0,91	0,93	0,92	0,92	0,93	0,93
AdaBoost	0,90	0,94	0,92	0,90	0,93	0,92
Karar Ağacı	0,97	0,96	0,96	0,97	0,96	0,96

Tablo 6.5 Fatura tarihi için karmaşıklık matrisi

Model	TP Lev.	FP Lev.	FN Lev.	TN Lev.	TP J-Win	FP J-Win	FN J-Win	TN J-Win
Rassal Orman	2664	78	79	41029	2657	76	86	41031
Gradyan Yükseltme Makinesi	2600	191	143	40916	2588	183	155	40924
Aşırı Gradyan Yükseltme	2660	111	83	40996	2664	123	79	40984
K-En Yakın Komşu	2558	242	185	40865	2563	219	180	40888
AdaBoost	2567	287	176	40820	2553	280	190	40827
Karar Ağacı	2626	86	117	41021	2627	78	116	41029

Tablo 6.6 Ödeme tarihi için kesinlik, duyarlılık ve F1 skor değerleri

Model	Kesinlik Lev.	Duyar. Lev.	F1 Lev.	Kesinlik J-Win	Duyar. J-Win.	F1 J-Win
Rassal Orman	0,97	0,81	0,88	0,95	0,81	0,87
Gradyan Yükseltme Makinesi	0,90	0,64	0,75	0,87	0,65	0,74
Aşırı Gradyan Yükseltme	0,91	0,80	0,85	0,91	0,81	0,86
K-En Yakın Komşu	0,66	0,61	0,63	0,64	0,56	0,60
AdaBoost	0,85	0,53	0,65	0,69	0,56	0,61
Karar Ağacı	0,78	0,80	0,79	0,74	0,78	0,76

Tablo 6.7 Ödeme tarihi için karmaşıklık matrisi

Model	TP Lev.	FP Lev.	FN Lev.	TN Lev.	TP J-Win	FP J-Win	FN J-Win	TN J-Win
Rassal Orman	126	4	30	3241	126	7	30	3238
Gradyan Yükseltme Makinesi	100	11	56	3234	101	15	55	3230
Aşırı Gradyan Yükseltme	125	12	31	3233	127	12	29	3233
K-En Yakın Komşu	95	49	61	3196	87	48	69	3197
AdaBoost	82	15	74	3230	87	40	69	3205
Karar Ağacı	125	35	31	3210	121	42	35	3203

Tablo 6.8 Teslimat tarihi için kesinlik, duyarlılık ve F1 skor değerleri

Model	Kesinlik Lev.	Duyar. Lev.	F1 Lev.	Kesinlik J-Win	Duyar. J-Win.	F1 J-Win
Rassal Orman	0,93	0,64	0,76	0,89	0,63	0,74
Gradyan Yükseltme Makinesi	0,83	0,44	0,58	0,76	0,52	0,62
Aşırı Gradyan Yükseltme	0,84	0,61	0,71	0,81	0,57	0,67
K-En Yakın Komşu	0,65	0,50	0,56	0,70	0,57	0,63
AdaBoost	0,65	0,34	0,45	0,83	0,34	0,48
Karar Ağacı	0,63	0,64	0,64	0,64	0,62	0,63

Tablo 6.9 Teslimat tarihi için karmaşıklık matrisi

Model	TP Lev.	FP Lev.	FN Lev.	TN Lev.	TP J-Win	FP J-Win	FN J-Win	TN J-Win
Rassal Orman	64	5	36	2226	63	8	37	2223
Gradyan Yükseltme Makinesi	44	9	56	2222	52	16	48	2215
Aşırı Gradyan Yükseltme	61	12	39	2219	57	13	43	2218
K-En Yakın Komşu	50	27	50	2204	57	25	43	2206
AdaBoost	34	18	66	2213	34	7	66	2224
Karar Ağacı	64	37	36	2194	62	35	38	2196

Tablo 6.10 Toplam tutar için kesinlik, duyarlılık ve F1 skor değerleri

Model	Kesinlik Lev.	Duyar. Lev.	F1 Lev.	Kesinlik J-Win	Duyar. J-Win.	F1 J-Win
Rassal Orman	0,97	0,88	0,93	0,98	0,88	0,93
Gradyan Yükseltme Makinesi	0,93	0,69	0,79	0,93	0,68	0,79
Aşırı Gradyan Yükseltme	0,96	0,88	0,92	0,96	0,88	0,92
K-En Yakın Komşu	0,87	0,78	0,82	0,84	0,77	0,80
AdaBoost	0,86	0,67	0,76	0,85	0,64	0,73
Karar Ağacı	0,86	0,88	0,87	0,85	0,88	0,86

Tablo 6.11 Toplam tutar için karmaşıklık matrisi

Model	TP Lev.	FP Lev.	FN Lev.	TN Lev.	TP J-Win	FP J-Win	FN J-Win	TN J-Win
Rassal Orman	1773	52	232	29617	1766	44	239	29625
Gradyan Yükseltme Makinesi	1386	105	619	29564	1367	102	638	29567
Aşırı Gradyan Yükseltme	1767	67	238	29602	1771	77	234	29592
K-En Yakın Komşu	1558	234	447	29435	1541	294	464	29375
AdaBoost	1353	221	652	29448	1292	222	713	29447
Karar Ağacı	1758	293	247	29376	1758	312	247	29357

Tablo 6.12 Net tutar için kesinlik, duyarlılık ve F1 skor değerleri

Model	Kesinlik Lev.	Duyar. Lev.	F1 Lev.	Kesinlik J-Win	Duyar. J-Win.	F1 J-Win
Rassal Orman	0,96	0,83	0,89	0,96	0,83	0,89
Gradyan Yükseltme Makinesi	0,90	0,55	0,68	0,89	0,55	0,68
Aşırı Gradyan Yükseltme	0,93	0,81	0,86	0,93	0,81	0,86
K-En Yakın Komşu	0,79	0,70	0,74	0,76	0,69	0,73
AdaBoost	0,78	0,52	0,62	0,76	0,52	0,62
Karar Ağacı	0,80	0,83	0,82	0,81	0,82	0,82

Tablo 6.13 Net tutar için karmaşıklık matrisi

Model	TP Lev.	FP Lev.	FN Lev.	TN Lev.	TP J-Win	FP J-Win	FN J-Win	TN J-Win
Rassal Orman	1628	70	346	28973	1625	71	341	28980
Gradyan Yükseltme Makinesi	1081	115	885	28936	1083	130	883	28921
Aşırı Gradyan Yükseltme	1585	127	381	28924	1584	113	382	28938
K-En Yakın Komşu	1373	357	593	28694	1361	420	605	28631
AdaBoost	1021	295	945	28756	1018	322	948	28729
Karar Ağacı	1636	399	330	28652	1620	382	346	28669

Tablo 6.14 Vergi tutarı için kesinlik, duyarlılık ve F1 skor değerleri

Model	Kesinlik Lev.	Duyar. Lev.	F1 Lev.	Kesinlik J-Win	Duyar. J-Win.	F1 J-Win
Rassal Orman	0,96	0,88	0,92	0,96	0,87	0,92
Gradyan Yükseltme Makinesi	0,91	0,63	0,75	0,91	0,62	0,74
Aşırı Gradyan Yükseltme	0,95	0,88	0,92	0,94	0,87	0,91
K-En Yakın Komşu	0,85	0,80	0,82	0,85	0,78	0,81
AdaBoost	0,80	0,59	0,68	0,82	0,57	0,67
Karar Ağacı	0,85	0,85	0,85	0,82	0,85	0,83

Tablo 6.15 Vergi tutarı için karmaşıklık matrisi

Model	TP Lev.	FP Lev.	FN Lev.	TN Lev.	TP J-Win	FP J-Win	FN J-Win	TN J-Win
Rassal Orman	1298	51	184	23816	1296	52	186	23815
Gradyan Yükseltme Makinesi	934	89	548	23778	923	90	559	23777
Aşırı Gradyan Yükseltme	1311	66	171	23801	1295	76	187	23791
K-En Yakın Komşu	1187	212	295	23655	1156	211	326	23656
AdaBoost	868	214	614	23653	847	189	635	23678
Karar Ağacı	1260	219	222	23648	1259	281	223	23586

Tablo 6.16 IBAN için kesinlik, duyarlılık ve F1 skor değerleri

Model	Kesinlik Lev.	Duyar. Lev.	F1 Lev.	Kesinlik J-Win	Duyar. J-Win.	F1 J-Win
Rassal Orman	0,99	0,99	0,99	0,99	0,99	0,99
Gradyan Yükseltme Makinesi	0,98	0,97	0,98	0,98	0,97	0,98
Aşırı Gradyan Yükseltme	0,99	0,99	0,99	0,99	0,99	0,99
K-En Yakın Komşu	0,97	0,99	0,98	0,98	0,99	0,99
AdaBoost	0,97	0,97	0,97	0,97	0,97	0,97
Karar Ağacı	0,99	0,99	0,99	0,99	0,99	0,99

Tablo 6.17 IBAN için karmaşıklık matrisi

Model	TP Lev.	FP Lev.	FN Lev.	TN Lev.	TP J-Win	FP J-Win	FN J-Win	TN J-Win
Rassal Orman	1630	10	3	31987	1629	18	4	31979
Gradyan Yükseltme Makinesi	1587	25	46	31972	1588	27	45	31970
Aşırı Gradyan Yükseltme	1625	12	8	31985	1629	16	4	31981
K-En Yakın Komşu	1627	49	6	31948	1629	39	4	31958
AdaBoost	1584	55	49	31942	1580	54	53	31943
Karar Ağacı	1620	19	13	31978	1612	22	21	31975

Tablo 6.18 Vergi numarası için kesinlik, duyarlılık ve F1 skor değerleri

Model	Kesinlik Lev.	Duyar. Lev.	F1 Lev.	Kesinlik J-Win	Duyar. J-Win.	F1 J-Win
Rassal Orman	0,99	0,99	0,99	0,99	0,99	0,99
Gradyan Yükseltme Makinesi	0,98	0,97	0,98	0,98	0,97	0,98
Aşırı Gradyan Yükseltme	0,99	0,99	0,99	0,99	0,99	0,99
K-En Yakın Komşu	0,98	0,98	0,98	0,98	0,98	0,98
AdaBoost	0,97	0,97	0,97	0,97	0,97	0,97
Karar Ağacı	0,99	0,99	0,99	0,99	0,99	0,99

Tablo 6.19 Vergi numarası için karmaşıklık matrisi

Model	TP Lev.	FP Lev.	FN Lev.	TN Lev.	TP J-Win	FP J-Win	FN J-Win	TN J-Win
Rassal Orman	1632	14	1	31983	1630	15	3	31982
Gradyan Yükseltme Makinesi	1590	26	41	31973	1592	24	40	31974
Aşırı Gradyan Yükseltme	1622	20	5	31979	1619	22	6	31979
K-En Yakın Komşu	1592	34	26	31943	1590	35	27	31943
AdaBoost	1581	45	39	31930	1583	44	36	31934
Karar Ağacı	1630	10	12	31978	1612	18	15	31975

Tablolara dikkat edildiğinde, kesinlik duyarlılık ve F1-skor değerleri analiz edildiğinde Rassal Orman modelinin bu veri seti için en başarılı model olduğu rahatlıkla anlaşılır. Zaman zaman Aşırı Gradyan Yükseltme ve Karar Ağaçları modelleri Rassal Orman modelinin başarı değerine yaklaşsa da geçememişlerdir. Diğer modellerin ise bariz bir şekilde geride kaldığı görülmektedir. Faturadaki her bir hedef alanının ayrı ayrı eğitilmesinin bir faydası da her bir alanı farklı modelle tahmin edebilme imkanının olmasıdır. Sonuçlara bakıldığında böyle bir ihtiyaç olmadığı görülmektedir. Ancak bu tarz durumlarda farklı modellerin kullanılmasıyla maksimum başarıya ulaşılan çalışmalarda mevcuttur (Nasiboğlu ve Akdoğan, 2020). Dolayısıyla farklı veri setleri ve farklı problemler için bu çözümü de göz önünde bulundurmak doğru olacaktır.

Tablo 6.2 ve Tablo 6.19 arasındaki tablolarda Levenshtein ve Jaro-Winkler gibi iki farklı kelime benzerlik hesaplarının başarılar üzerindeki etkileri karşılaştırmalı olarak görülmektedir. Zaman zaman Jaro-Winkler kullanılan modellerdeki başarı daha

yüksek gözükse de genel yapıya bakıldığında Levenshtein yönteminin daha üstün olduğu söylenebilir. Ancak bu üstünlük model seçimindeki kadar bariz değildir. Hatta bazı durumlarda kesinlik, duyarlılık ve F1-skor tablolarındaki değerlerin eşitliğini bozamayacak kadar ufak farklar meydana gelebilmektedir. Bu sebeple aradaki farkları analiz edebilmek için karmaşıklık matrisi kullanılmıştır.

Tablo 6.20 Fatura numarası için 2.seviye model eğitimi

Model	**TP**	**FP**	**FN**	**TN**
Rassal Orman	185	20	36	4993
Gradyan Yükseltme Makinesi	159	19	52	5004
Aşırı Gradyan Yükseltme	180	22	32	5000
K-En Yakın Komşu	167	14	54	4999
AdaBoost	137	24	84	4989
Karar Ağacı	187	34	46	4967
Yapay Sinir Ağları	160	30	51	4993
Evrişimsel Sinir Ağları	194	19	27	4994

Fatura numarasında kullanılacak olan 2.modelin belirlenmesi için eğitimler gerçekleştirilmiş ve elde edilen sonuçlara göre model seçimi yapılmıştır. Tablo 6.20'de görüldüğü üzere en iyi sonuç evrişimsel sinir ağları modeli ile edilmiştir. Bölüm 4'te de anlatıldığı üzere ara katmanlarda (1 evrişimsel + 2 lineer) Relu aktivasyon fonksiyonu, son katmanda (lineer) ise Sigmoid aktivasyonu kullanılmıştır. Öğrenme oranı 0,001 ve adım sayısı parametresi 500 olarak belirlenmiştir. Bu işlem

uygulanmadan önce sisteme verilen 2000 adet farklı faturanın 1879 tanesi (0,939) doğru olarak tahmin edilmiştir. Oluşan yeni hibrit model ile 1902 adet faturada (0,951) doğru sonuç elde edilmiştir. Bu sonuç doğrultusunda aynı işlemler toplam tutar, net tutar ve vergi tutarı içinde uygulanmıştır. Aynı 2000 fatura ile test edildiğinde toplam tutar için klasik model 1812 faturayı (0,906) doğru tahmin ederken hibrit model 1871 faturayı (0,935) doğru tahmin etmiştir. Net tutar için klasik model 1764 (0,882), hibrit model 1822 (0,911) ve vergi tutarı için klasik model 1851 (0,925), hibrit model 1891 (0,945) faturayı doğru tahmin etmiştir. Buradaki başarı oranını daha fazla artırmak ve diğer modellerde de bu yöntemin uygulanabilmesi için daha kapsamlı teknik bir çalışma başlatılmıştır. Bu tez kapsamında diğer etiketler için yeterince aykırı gözlem verisi bulunmadığından dolayı bu teknik fatura numarası, toplam tutar, net tutar ve vergi tutarı için uygulanmıştır.

Tablo 6.16-6.19'daki tablolarda görüldüğü üzere eğitilen model IBAN ve vergi numaralarını tahmin etmekte oldukça iyidir. Buradaki az sayıda hatanın sebebi ise genel olarak faturalardaki OCR hatalarıdır. Bu hataların da önüne geçebilmek adına Bölüm 5'te bahsedilen validasyon katmanı kullanılmıştır. Böylelikle faturadan yapay zeka yardımıyla elde edilen IBAN ve vergi numarası değerlerinin OCR hatalı olup olmama durumları tespit edilmiştir. Bu aşamada fatura görüntüleri üzerinde yapılan görüntü işleme süreçleri her faturada sabit tutulmuştur. Yeterince veri toplandığında, her bir faturaya hangi görüntü temizleme işlemlerinin hangi parametre değerleriyle yapılacağını belirleyecek yapay zekanın oluşturulması üzerine çalışma başlatılacaktır. Böylelikle farklı tip faturalarda farklı tip düzenlemeler yapılarak OCR hataları minimuma indirilecektir. Bu sebeple, eklenen validasyon katmanları hem tahmin edilen değerin doğruluğunu test etmemize yardımcı olurken hem de farklı bir proje için veri elde edilmesini sağlayacaktır.

Veri setinde bulunan faturalar içinde çok uzun veya tek sayfalık çok kısa faturalar olmakla birlikte ortalama fatura başına yaklaşık 1500-2000 arası N-gram oluşturulmaktadır. Bu durum eğitime girecek olan etiketli ve etiketsiz veriler arasında dengesizlik oluşmasına sebep olur. Örnek olarak fatura üzerinde 1 adet fatura numarası (etiketli veri) bulunurken 1800 adet etiketsiz veri bulunabilmektedir. Bu şartlarda

eğitim yaptırıldığında öğrenme işlemi gerçekleşmeyecek ve model her veriyi etiketsiz veri olarak tahmin edecektir. Bu sebeple az sayıda olan sınıftaki verilerin tamamı alınırken (etiketli veriler), sayıca üstün olan sınıftaki verilerin yalnızca belirli bir kısmı alınır r (Chawla, Bowyer, Hall ve Kegelmeyer, 2002). Böylelikle öğrenme işleminin sağlıklı bir şekilde gerçekleşmesi sağlanır. Yapılan eğitim denemelerinin sonucunda her bir etiketli veri durumu için farklı sonuçlar elde edilse de 12-17 kat arasında etiketsiz veri seçimi yapılması en iyi sonuçların alınmasını sağlamaktadır. Eğitim için gerekli olan etiketsiz veriler, etiketli verilerin özellikleri göz önüne alınarak seçilmelidir. Örnek olarak toplam tutar ifadesinin eğitimi yaptırılırken, etiketsiz veri olarak içerisinde çok sayıda harf geçen string ifadeler kullanmak öğrenme işleminin sınırlı kalmasına neden olacaktır. Bu sebeple tutar ifadesine en çok benzeyen etiketsiz verileri kullanarak eğimi yaptırmak en iyi sonucun elde edilmesini sağlar. Belirli koşullara göre oluşturulmuş etiketsiz veri havuzundan, belirlenen etiketsiz veri sayısı kadar (12-17 kat) seçim yapılarak eğitim gerçekleştirilir. Seçim işlemi 500 tekrar ile havuz içerisinden rastgele şekilde yapılır. Böylelikle eğitim için gereken en faydalı etiketsiz veri seti elde edilir.

BÖLÜM 7
SONUÇ

Tez çalışmasında görüntü formatındaki faturalardan yapay öğrenme ve derin öğrenme tekniklerine dayalı bilgi çıkarımı süreci ele alınmıştır. Eğitim için toplamda 9910 adet fatura kullanılmıştır. Görüntü şeklindeki faturaların metin şekline dönüştürülmesi için derin öğrenme destekli Tesseract optik karakter tanıyıcı kullanılmıştır. Faturalardaki etiketlenmiş verilerin tahmin edilmesinde çeşitli yapay öğrenme modelleri kullanılmıştır. Ayrıca farklı kelime benzerlik teknikleri olarak Levenshtein ve Jaro-Winkler uzaklıkları kullanılmıştır. Kullandığımız veri setinde, Rassal Orman modelinin diğer modellere karşı üstünlüğü görülmüştür. Modellerin karşılaştırılmasında F1-skor değerleri kullanılmıştır. Tablo 6.2-6.19 arasındaki tablolarda modellerin tahminleme sonuçları verilmiştir. Fatura numarası için 0,97, fatura tarihi için 0,97, ödeme tarihi için 0,88, teslimat tarihi için 0,76, toplam tutar için 0,93, net tutar için 0,89, vergi tutarı için 0,92, IBAN için 0,99 ve vergi numarası için 0,99 olarak F1-skor değerleri elde edilmiştir. Modellerde farklı kelime benzerlik teknikleri sonucuna göre Levenshtein ve Jaro-Winkler yöntemleri birbirlerine çok yakın sonuçlar sergilemiştir.

Gelecek çalışmalarda faturalardaki şirket ismi, adres gibi diğer bilgilerin de tahmin edilmesi ilgi çekebilir. Ayrıca, bozuk görüntülü ve daha karmaşık yapılı belgelerin görüntülerinden bilgilerin çıkarılabilmesi için daha ileri yapay zeka ve esnek hesaplama teknolojilerinin kullanılması hedeflenebilir.

KAYNAKLAR

Abinesh, B. (2020). *Convolutional Neural networks for computer vision applications*. 28 Ekim 2020, https://medium.com/analytics-vidhya/convolutional-neural-networks-for-computer-vision-applications-4808a6d984c2s.

Albawi, S., Mohammed, T., A. ve Al-Zawi, S. (2017). Understanding of a convolutional neural network. *International Conference on Engineering and Technology (ICET)*, 1-6.

Ariawan, E. (2018). *Predictive analysis of employee loyalty: a comparative study using logistic regression model and artificial neural network*. 23 Ekim 2020, https://www.researchgate.net/figure/Activation-function-of-neuron_fig1_329799984.

Bailey, J. (2018). *Inventing the future of art analytics*. 18 Ekim 2020, https://www.artnome.com/news/2018/11/8/inventing-the-future-of-art-analytics.

Boosting algoritmaları nasıl çalışır. (2019). 25 Ekim 2020, https://medium.com/@sertacozker/boosting-algoritmalar%C4%B1-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-edac1174e971.

Boosting in machine learning | boosting and adaboost. (2019). 25 Ekim 2020, https://media.geeksforgeeks.org/wp-content/uploads/20190324085500/adaboost.jpg.

Breiman, F. (2001). Random forest. *Machine Learning*, *45* (1), 5–32.

Cavaioni, M. (2017). *Machine learning: decision tree classifier*. 17 Ekim 2020, https://medium.com/machine-learning-bites/machine-learning-decision-tree-classifier-9eb67cad263e.

Chai, T. ve Draxler, R.R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)- arguments against avoiding RMSE in the literature. *Geoscientific Model Development, 7* (3), 1247-1250.

Chawla, N., V., Bowyer, K., Hall, L., O. ve Kegelmeyer, W., P. (2002). Smote: synthetic minority oversampling technique. *Journal of Artificial Intelligence Research 16* (1), 321–357.

Chen, T. ve Guestrin , C. (2016). Xgboost: a scalable tree boosting system. *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.

Chollet, F. (2017). Anatomy of a neural network. *Deep Learning with Python* (1.Baskı) içinde (58-61). Amerika Birleşik Devletleri: Manning Yayınevi.

Ding, B., Qian, H. ve Zhou, J. (2018). Activation functions and their characteristics in deep neural networks. *Chinese Control And Decision Conference (CCDC),* 1836-1841.

Dumoulin, V. ve Visin, F. (2016). *A guide to convolution arithmetic for deep learning.* 06 Aralık 2020, https://arxiv.org/pdf/1603.07285.pdf.

Endicott, S. (2017). *Game applications of deep neural networks.* 20 Kasım 2020, https://www.semanticscholar.org/paper/Game-Applications-of-Deep-Neural Networks-Endicott/8993743c75e2b611afb3b0ee7629c2bc67326a39/figure/4.

Esser, D., Schuster, D., Muthmann, K., Berger, M. ve Schill, A. (2012). Automatic indexing of scanned documents: a layout-based approach. *In Document Recognition and Retrieval XIX, 8297*, 82970H.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27* (8), 861–874.

Flor, M. (2013). A fast and flexible architecture for very large word n-gram datasets. *Natural Language Engineering, 19* (1), 1-33.

Freund, Y. ve Schapire, R., E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55* (1), 119–139.

Friedman, J., H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics, 29* (5),1189-1232.

Gelfand, S., Ravishankar, C., S. ve Delp, E., J. (1991). An iterative growing and pruning algorithm for classification tree design. *IEEE Transaction on Pattern Analysis and Machine Intelligence, 13* (2), 163-174.

Goodfellow, I., Bengio, Y. ve Courville, A. (2016). Softmax units for multinoulli output distributions. *Deep Learning. MIT Press*, 180–184.

Guo, G., Wang, H., Bell, D., Bi, Y. ve Greer, K. (2003). KNN model-based approach in classification. *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", LNCS 2888,* 986-996.

Haldar, R. ve Mukhopadhyay, D. (2011). *Levenshtein distance technique in dictionary lookup methods: an improved approach.* 28 Kasım 2020, https://arxiv.org/pdf/1101.1232.pdf.

How to draw faces: female. (2010). 10 Aralık 2020, http://sharenoesis.com/wp-content/uploads/2010/05/7ShapeFaceRemoveGuides.jpg.

Huber, P., J. (1964). Robust estimation of a location parameter. *Annals of Statistics*, *53* (1), 73–101.

Improving the quality of the output, (2015). 20 Kasım 2020, https://github.com/tesseract-ocr/tessdoc/blob/master/ImproveQuality.md.

Janssen, B., Saund, E., Bier, E., Wall, P. ve Sprague, M. (2012). Receipts2Go: the big world of small documents. *Proceedings of the 2012 ACM Symposium on Document Engineering, 12*, 121-124.

Jaro, M., A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa. *Journal of the American Statistical Association, 84* (406), 414-420.

Katti, A., Reisswig, C., Guder, C., Brarda, S., Bickel, S., Höhne, J. ve diğer. (2018). Chargrid: towards understanding 2d documents. *Association for Computational Linguistics, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4459-4469.

Kégl, B. (2009). *Introduction to adaboost*, 23 Ekim 2020, https://users.lal.in2p3.fr/kegl/teaching/stages/notes/tutorial.

Klein, B., Agne, S. ve Dengel, A. (2004). Results of a study on invoice- reading systems in germany. *In Document Analysis Systems VI ser. Lecture Notes in Computer Science, 3163*, 451–462.

Krishnakumari, K., Elango, S. ve Radhakrishnan, S. (2020). Hyperparameter tuning in convolutional neural networks for domain adaptation in sentiment classification (HTCNN-DASC). *Soft Computing, 24* (5),1-17.

Krizhevsky, A., Sutskever, I. ve Hinton, G., E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25* (2), 1097-1105.

LeCun, Y. ve Bengio, Y. (1995). Convolutional networks for images, speech, and time series. Arbib, M., A. (Ed.), *The Handbook of Brain Theory and Neural Networks* (2.Baskı) içinde (276-278). Amerika Birleşik Devletleri: The MIT yayınları.

Liu, W., Wan, B. ve Zhang, Y. (2016). *Unstructured document recognition on business invoice*. 12 Ekim 2016, http://cs229.stanford.edu/proj2016/report/LiuWanZhang-UnstructuredDocumentRecognitionOnBusinessInvoice-report.pdf.

Mashat, A., Fouad, M., Yu, P. ve Gharib, T. (2012). A decision tree classification model for university admission system. *International Journal of Advanced Computer Science and Applications*, *3* (10), 17–21.

McCulloch, W., S. ve Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics, 5*, 115–133.

Menze, B., H., Kelm, B., M., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W. ve Hamprecht, F., A. (2009). A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC Bioinformatics, 10* (1), 213.

Nair, A. (2016). *Overview of tesseract OCR engine.* 06 Aralık 2020, https://www.researchgate.net/publication/315834331_Overview_of_Tesseract_OCR_engine.

Nasiboğlu, R. ve Akdoğan, A. (2020). Estimation of the second hand car prices from data extracted via web scraping techniques. *Journal of Modern Technology and Engineering, 5* (2), 157-166.

Nasr, G., E., Badr, E., A. ve Joun, C. (2002). Cross entropy error function in neural networks: forecasting gasoline demand. *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference*, 381-384.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics, 9* (1), 62-66.

Öztemel, E. (2012). *Yapay sinir ağları* (3. Baskı). Ankara : Papatya Yayıncılık.

Palm, R., Winther, O. ve Laws, F. (2017). Cloudscan - a configuration-free invoice analysis system using recurrent neural networks. *International Conference on Document Analysis and Recognition (ICDAR), 14*, 406-413.

Patel, I., C., Patel, A. ve Patel, D. (2012). Optical character recognition by open source OCR tool tesseract: a case study. *International Journal of Computer Applications 55* (10), 50-56.

Potdar, K., Pardawala, T. ve Pai, C. (2017). A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications, 175* (4), 7-9.

Press, W.,H., Teukolsky, S., A., Vetterling, W.,T. ve Flannery, B., P. (2007). 14.7.2. Kullback–Leibler distance. *Numerical Recipes: The Art of Scientific Computing* (3.Baskı) içinde (756-758). İngiltere: Cambridge Üniversitesi Yayınları.

Qi, C. ve Su, F. (2017). Contrastive-center loss for deep neural networks. *International Conference on Image Processing (ICIP)*, 2851-2855.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning, 1* (1), 81–106.

Sabour, S., Frosst, N. ve Hinton, E., G. (2017). Dynamic routing between capsules. *31st Conference on Neural Information Processing Systems (NIPS'17),* 3856-3866.

Sartin, M., A. ve Silva, A., C. (2013). Approximation of hyperbolic tangent activation function using hybrid methods. *In 2013 8th International Workshop on Reconfigurable and Communication-Centric Systems-onChip (ReCoSoC)*, 1–6.

Scherer, D., Müller, A. ve Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. *Artificial Neural Networks - ICANN 2010 - 20th International Conference*, 92-101.

Schulz, U., K. ve Mihov. S. (2002). Fast string correction with Levenshtein automata. *International Journal on Document Analysis and Recognition, 5,* 67-85.

Schuster, D., Muthmann, K., Esser, D., Schill, A., Berger, M., Weidling, C. ve diğer. (2013). Intellix– end-user trained information extraction for document archiving. *International Conference on Document Analysis and Recognition*, *12*, 101– 105.

Smith, R. (2007). An overview of the tesseract OCR engine. *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, 629-633.

Tang, S. (2017). *Predicting the ambiance of restaurants using only the wording of Yelp reviews*. 24 Ekim 2020, https://medium.com/fun-with-data-and-stats/predicting-the-ambiance-of-restaurants-using-only-the-wording-of-yelp-reviews-954413b6d490.

Watanabe, T., Tsukada, H. ve Isozaki, H. (2009). A succinct n-gram language model. *International Joint Conference on Natural Language Processing (IJCNLP),* 341–344.

Willmott, C., J. ve Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research, 30*, 79–82.

Wu, Y. ve Liu, Y. (2007). Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association, 102* (479), 974–983.

Xiaoliang, Z., Jian, W., Hongcan, Y. ve Shangzhuo, W. (2009). Research and application of the improved algorithm C4.5 on decision tree. *International Conference on Test and Measurement (ICTM), 2*, 184-187.

Yazıcı, E. (2015). *Tüm fiziği tek bir kuramda birleştirmeye çalışacak kadar iddialı bir adam.* 25 Kasım 2020, https://bilimteknik.tubitak.gov.tr/sites/default/files/posterler/albert_einstein.pdf.

Yol çizgileri trafik güvenliğini arttırıyor. (2019). 25 Ekim 2020, https://www.sakaryadanhaber.com/haber/3019569/yol-cizgileri-trafik-guvenligini-arttiriyor.

Zeiler, D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. ve Hinton, G., E. (2013). On rectified linear units for speech processing. *International Conference on Acoustics - Speech and Signal Processing*, 3517–3521.