T.C.
NECMETTİN ERBAKAN NİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARI İLE BANKACILIK DÖKÜMANLARININ SINIFLANDIRILMASI

Ali ESER

YÜKSEK LİSANS TEZİ

Endüstri Mühendisliği Anabilim Dalı

Temmuz-2021
KONYA

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

İmza

Ali ESER

Tarih: 14/ 07/ 2021

ÖZET

YÜKSEK LİSANS TEZİ

YAPAY SİNİR AĞLARI İLE BANKACILIK DÖKÜMANLARININ SINIFLANDIRILMASI

Ali ESER

Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü
Endüstri Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Abdullah Erdal TÜMER

2021, 72 Sayfa

Jüri
Prof. Dr. Sabri KOÇER
Doç. Dr. Abdullah Erdal TÜMER
Dr. Öğretim üyesi Vahit TONGUR

Teknolojide yaşanan gelişmelerle birlikte gün geçtikçe dijital ortamdaki belgeler artmaktadır. Artan belgelerin belli bir düzen içinde sınıflandırmak için insan gücüne her geçen gün daha fazla ihtiyaç duyulmaktadır. Özellikle şirketlerin müşteri verilerini düzgün kayıt edilmesi, müşteri taleplerine hızlı bir şekilde cevap verilmesi, şirketin hem yasal olarak hem de özel işlerinin düzenli takip edilebilmesi oldukça önemlidir. Bu gibi insan gücüne ihtiyacın arttığı süreçlerde özellikle kurumsal şirketler müşterilerine sunduğu hizmetlerde yazılım gücünü etkin bir şekilde kullanmaya çalışmaktadır.

Bu tezde, taranan dokümanların sınıflandırılması değerlendirilmektedir. Tezin başlıca amacı başarılı sayılabilecek bir doğrulukla resimden metine çevirme işlemi olan Optical Character Recognation (OCR) işleminin yapılması ve yapay zekâ algoritmalarından en uygun algoritma ile de metinlerin sınıflandırılması işlemidir.

Taranan dokümanların resim formatından metin haline dönüştürülmesi için Windows OCR kütüphanesi kullanılmıştır. Bu kütüphane sayesinde hem hız olarak hem de doğruluk anlamında başarı elde edilmiştir. El yazısı harici bilgisayar yazısı olan dokümanlarda bu başarı oranı daha da yükselmektedir. Metin haline dönüştürülmüş dokümanların sınıflandırma işlemi .NET platformuna 2018 yılında dahil olan açık kaynak kodlu olan ML.NET makine öğrenmesi kütüphanesidir. ML.NET kütüphanesi açık kaynak kodlu olması, yeni araştırmalara açık olması, kullanıldığı makine de internet bağlantısı gerektirmemesi, hız ve başarı oranı yüksek olduğu için seçilmiştir. Araştırmanın kodları ile farklı sürümlerle güncellenebilmesi, farklı ortamlarda test edilebilmesi ve bu çalışma üzerine ileride daha yeni geliştirmelerin başkaları tarafından da yapılabilmesi için Github üzerinde tutulmuştur.

Geliştirilen yazılım ile taranan dokümanlar %97 başarı ile metin haline dönüştürülmüştür. Metin halindeki dokümanların sınıflandırılması işlemindeki başarı oranı ise %92'dir. Dokümanları temsil eden özelliklerin oluşmasında belgede geçen kelimelerin tekrarı önem arz etmektedir. Kimlik, ehliyet, ruhsat gibi aynı türdeki dokümanların belli kalıptaki kelimeleri barındırdığından, sınıflandırılma işlemi de başarılı olmuştur. Ayrıca tarama kalitesinin iyi olması ve taranan dokümanların bilgisayar yazısı ile hazırlanmış olması OCR işleminin daha başarılı olmasını sağlamıştır. Başarı oranını artırmıştır. Bu yazılımın özellikle kullanıcı/müşteriden evrak istendiği bankacılık vb. sektörlerde başarılı bir şekilde kullanılabileceği düşünülmektedir. Geliştirilen yazılıma patent alınması ve ticarileşmesi için de çalışmalar yapılacaktır.

Anahtar Kelimeler: Makine Öğrenmesi, OCR, Sınıflandırma

ABSTRACT

MS THESIS

CLASSIFICATION OF ARTIFICIAL NEURAL NETWORKS AND BANKING DOCUMENTS

Ali ESER

NECMETTİN ERBAKAN UNIVERSITY INSTITUTE OF SCIENCE OF SCIENCE

Advisor: Assoc. Prof. Abdullah Erdal TÜMER

2021, 72 Pages

Jury
Prof. Sabri KOÇER
Assoc. Prof. Abdullah Erdal TÜMER
Lecturer Vahit TONGUR

With the developments in technology, the documents in the digital environment are increasing day by day. Manpower is needed more and more every day in order to classify the increasing documents in a certain order. It is especially important for companies to properly record customer data, to respond quickly to customer requests, and to be able to follow up both legal and private affairs of the company regularly. In such processes where the need for manpower increases, especially corporate companies try to use the software power effectively in the services they provide to their customers.

In this thesis, the classification of the scanned documents is evaluated. The main purpose of the thesis is the process of Optical Character Recognation (OCR), which is the process of converting from image to text with an accuracy that can be considered successful, and the process of classifying the texts with the most appropriate algorithm from artificial intelligence algorithms.

Windows OCR library is used to convert scanned documents from image format to text. Thanks to this library, success has been achieved in terms of both speed and accuracy. This success rate increases even more for documents with handwritten external computer writing. The classification process of documents converted into text is the open source ML.NET machine learning library that was included in the .NET platform in 2018. ML.NET library was chosen because it is open source, open to new researches, does not require internet connection, and has a high speed and success rate. It is kept on Github in order to be updated with different versions with the codes of the research, to be tested in different environments and to be able to make new developments on this work by others in the future.

The documents scanned with the developed software were converted into text with 97% success. The success rate in classifying documents in text form is 92%. The repetition of the words in the document is important in the formation of the features that represent the documents. The classification process was also successful, as documents of the same type, such as identity, driver's license, and license, contain certain key words. In addition, good scanning quality and the fact that the scanned documents were prepared with computer text made the OCR process more successful. It has increased the success rate. It is thought that this software can be used successfully in banking etc. sectors where documents are requested from the user / customer. Efforts will also be made to obtain patents and commercialize the developed software.

Keywords: Classification, Machine Learning, OCR

ÖNSÖZ

Kariyerimin her basamağında yanımda olan ve beni bugünlere getiren anneme ve babama, yüksek lisans çalışmam boyunca manevi destekleri ile varlığını hissettiren ve her daim yanımda olan nişanlım Nurdan Karataş'a teşekkür ederim. Yüksek lisans sürecimde bilimsel katkıları ile yardımlarını benden esirgemeyen, gösterdiği yakınlık, özveri ve samimiyetle çalışmalarımın her aşamasında danışmanlığı ile yol gösteren değerli hocam ve danışmanım Sayın Doç. Dr. Abdullah Erdal Tümer'e teşekkür eder ve saygılarımı sunarım.

Ali ESER

KONYA-2021

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Kısaltmalar

OCR:	Optical Character Recognation
DVM:	Destek Vektör makinesi
ML.Net:	Machine Learning .Net
UWP:	Evrensel Windows Platformu
SVM:	Support Vector Machine
NB:	Naive Bayes
CNN:	Convolutional Neural Network
RVL-CDIP:	Ryerson Vision Lab Complex Document Information Processing

1. GİRİŞ

Teknolojide yaşanan gelişmelerle birlikte özellikle kurumsal şirketler müşterilerine sunduğu hizmetlerde yazılım gücünü etkin bir şekilde kullanmaya çalışmaktadır. Şirketler yeni müşteri kayıt ederken kanunlar tarafından müşteriden zorunlu olarak bazı belgeler almaktadır. Ayrıca özel ihtiyaç duyduğu belgeleri de aday müşteriden talep etmektedirler. Bu alınan belgeler tarayıcılar aracılığı ile taratılıp sisteme kayıt edilmektedir. Sisteme kayıt edilen dosyalar şirketin kullanmış olduğu yazılım çeşidine göre sunucuya kayıt edilmektedir. Müşteri verilerinin düzgün olması şirketin hem yasal olarak hem de özel işlerinin düzenli takip edilebilmesi açısından oldukça önemlidir. Bu aşamada müşteri kaydı yapan personelin taranan belgeleri düzenli bir şekilde tarayıcıda taraması ve ilgili klasörlere ataması gerekmektedir. Yanlış dokümanların yanlış klasörlere atılması ileride büyük sıkıntılara yol açabilmektedir. Örneğin teftiş gibi yasal denetim zamanında şirkete yaptırım cezalarına sebep olabilecek zararlara yol açabilmektedir. Bundan dolayı şirket yazılımlarında taranan dokümanları öğrenebilen ve sınıflandırabilen bir yazılım gücüne ihtiyaç duyulmaktadır. Böyle bir yazılım personelin ekstra zaman harcamandan dokümanları ilgili klasöre otomatik yerleştirilmesini sağlayacaktır. Gereksiz zaman kaybının önüne geçilebilecek ve personeller daha verimli çalışabileceklerdir.

Hızla değişen işletme ihtiyaçları ve rekabetler teknolojik yeni yöntemlerin araştırılmasını zorunlu hale getirmiştir. Bu amaçla pek çok kamu ve özel sektör ar-ge bölümleri kurmakta ve ilgili elamanları istihdam etmektedir. Bu amaçla ihtiyaca yönelik yazılımlar geliştirilmektedir. Üretilen yöntemler kurumların organizasyon kültürü, rekabetleri, müşteri portföyü, müşterinin ihtiyaçları gibi etkenlere göre şekillenmektedir. Farklı yöntemlerin ortaya çıkması ile kurumların rekabete uyum sağlaması hedeflenmektedir. Yazılım projelerinde yapay zekâ içerikli yöntemlerin kullanılmaya başlanması ile kurumlara insan gücünü etkin kullanımının azaltılması, hata paylarının azaltılması, müşteri taleplerinin daha doğru ve etkin bir şekilde belirlenmesi, işlerin akıllı bir şekilde yapılarak zaman kazanılması amaçlanmıştır.

Şirketler yeni müşteri kayıt ederken kanunlar tarafından müşteriden zorunlu alınması gereken belgelere artı olarak özel ihtiyaç duyduğu belgeleri aday müşteriden talep etmektedirler. Bu alınan belgeler tarayıcılar aracılığı ile taratılıp sisteme kayıt edilmektedir. Sisteme kayıt edilen dosyalar şirketin kullanmış olduğu yazılım çeşidine göre sunucuya kayıt edilmektedir. Müşteri verilerinin düzgün olması şirketin hem yasal

olarak hem de özel işlerinin düzenli takip edilebilmesi açısından oldukça önemlidir. Örneğin, bankalar şube aracılığı ile müşteri kayıt işleminde oldukça fazla doküman talep etmektedirler. Müşteriden alınan nüfus cüzdanı, kimlik kartı, pasaport gibi belgeler ve banka tarafından yazıcıdan alınan çıktıların imzalatılarak hepsi birlikte tarayıcı aracılığı ile taranması ve müşterinin klasörüne girip, tek tek ilgili klasörlere taranan dokümanların eklenmesi gerekmektedir. Bu işlem süresince hem müşteri şube de daha çok zaman harcamaktadır, hem de personelin müşteri ile ilgilenmesi gereken zamanda taranan dokümanları sınıflandırma ile uğraşması gerekmektedir. Özellikle pandemi döneminde hem müşteri hem de banka memuru açısndan aynı ortamın teneffüs edilmesi de ayrı bir sorun olarak karşımızda durmaktadır. Bu sebeplerden ötürü müşteri memnuniyeti sağlamak, şubede personel kaynağını/iş gücünü verimli kullanmak adına dokümanların otomatik olarak sınıflandırılması bahsedilen problemlerin önlenmesi açısından bir çözüm oluşturacaktır.

Bu çalışmada yapay zekâ algoritmaları kullanarak şirketin taradığı dokümanların Optical Character Recognation (OCR) işlemi ile metin haline getirilmesi ve bu metinlerin sisteme öğretilmesi sağlanmıştır. OCR işlemi Microsoft OCR kütüphanesi görüntü işlemi desteği ile sağlanmıştır. Microsoft OCR Kütüphanesi, evrensel Microsoft platformunun (UWP-Universal Windows Platform) bir parçasıdır (Josipovic, 2020). Yapay sinir ağının eğitilmesi .net tabanlı ML.NET yapay zekâ kütüphanesi ile yapılmıştır. Sistemin öğrenmesinden sonra sınıflandırma algoritmaları ile dokümanların ilgili klasörlere atılması sağlanmıştır. Bu proje ile amaç sistem tarafından en az hata ile en doğru yönlendirme işleminin gerçekleştirilmesidir. Bu yönlendirme ile şirketin ihtiyaç duyduğu müşteri dokümanları doğru bir şekilde sınıflandırabilecektir.

Bu çalışma ile bir şirketin ihtiyaç duyduğu dokümanların makine öğrenmesi ile sınıflandırılması sağlanmıştır. Sınıflandırma için üretilen model farklı uygulamalarda kullanılabilecek şekilde tasarlanmıştır. Çalışmanın veri setinin düzenlenmesi, veri elde edilmesi, eğitim, test ve doğrulama veri setinin oluşturulması ve makine öğrenimi aşaması sonraki aşamalarda anlatılmıştır. Sınıflandırma başarısı sonuçları çalışma için tasarlanan özel uygulama ara yüzü ile kullanıcı ile paylaşılmıştır. Ara yüz çalışması Microsoft Windows form uygulama ile bir masaüstü yazılım geliştirilmiştir.

Bu çalışmada kullanılan tüm kodlar Github üzerinde depolanmıştır. Github, git kullanarak sürüm kontrolü ile kod depolama sağlayan Microsoft yan kuruluşudur (Github, 2020). Bu sayede geçmiş sürümler kontrol altına alınmıştır. Açık kaynak kodlu geliştirme sağlanmış oldu.

2. KAYNAK ARAŞTIRMASI

Bu bölümde yapay zekâ, sınıflandırma, görüntü işleme ile görüntüden yazı elde edilmesi gibi konular tanıtılarak bu alanda yapılmış çalışmalara ait bazı literatür ve patentler sunulmuştur.

2.1. Doküman Sınıflandırma ile İlgili Literatür Çalışmaları

Optical Character Recognition (OCR) olarak bildiğimiz teknoloji Türkçe de Optik Karakter Tanımlama olarak adlandırılmaktadır. OCR işlemi doküman veya taranan her türlü belgenin resim halinin OCR programları aracılığı ile metne dönüştürülme işlemidir.

Yapay zekâ teknikleri ve istatiksel yöntemler kullanılarak çeşitli doküman sınıflandırma modelleri geliştirilmiştir. Doküman sınıflandırmayı metinleri sınıflandırarak NB (Naive Bayes), SVM (Support Vector Machine), RF (Random Forest) gibi sınıflandırma yöntemleri kullanarak elde edilen çalışmalar incelenmiştir.

(Sarı, 2018), dokümanların yazarına ve konusuna göre sınıflandırılması için çalışılmıştır. Bir gazete yazısının vektörleri oluşturulmuş ve analizi yapılmıştır. Yazarların stilleri gruplandırarak yazar profilleri oluşturulmuştur. Konusu bilinmeyen bir yazının konusunun tahmin edilmesi üzerine araştırmalar yapılmıştır. Bu araştırmaları DeepLearning4J Java kütüphanesi kullanılmıştır. Elde ettikleri sonuçlara göre yazarlar bazı yazarlar diğer yazarlardan belirgin bir şekilde ayrılmaktadır. En başarılı sonucu model sayısında 5 yazar olduğunda 0,88 oranında başarım ile sağlanmıştır. Yazar sayısı artırıldığında başarım oranı 0,69 seviyesine kadar düşmüştür.

(Çobanoğlu, 2015), Türkçe metinler ile yazılmış dokümanların sınıflandırma yaklaşımları üzerine çalışmışlardır. Hedefleri dokümanları ön işleme adımları ve sınıflandırma algoritmaları ile en iyi kombinasyonu oluşturmaya çalışmışlardır. Dokümanların esas özelliklerini belirlemek için kelimelerin kendileri, kökleri, bi-gram, tri-gram formları kullanılmıştır. Bi-gram 2 kelimelik gruplar, tri-gram 3 kelimelik gruplardır. Elde edilen sonuçlara göre C4.5 Decision Tree sınıflandırma algoritması ile %95 doğru sınıflandırmaya ulaşmışlar. SVM algoritması ile C4.5 Decision Tree sınıflandırma algoritmasına yakın sonuçlar elde edilmiş. NB algoritması ile yapılan sınıflandırmalar ise bu üç algoritmadan en düşük doğruluk sonucunu elde etmişler.

(Fidan, 2013), bu çalışmada 20 veri seti sınıflandırılmıştır. Veri setinde 18774 doküman ve her dokümanın içinde 61188 özellik bulunmaktadır. Özellik sayısı probleme matematiksel anlamda boyut kavramını etkilemektedir. En yüksek 250000 veri ile yapılan eğitim ve test yapıldığında elde edilen sonuçlarda eğitim veri setinin yaklaşık 250000 / 352463076 ~= 0.0007 örnek kullanılmış. Tüm deneylerde rastgele örnekler seçilmiştir. Destek vektör makineleri (DVM) kullanarak doküman sınıflandırma çalışması yapılmıştır. Çalışmanın deneysel sonucunda 20 farklı veri seti sınıflandırılmıştır. 18774 doküman ve her doküman için 61188 özellik bulunmaktadır. Eğitim verisi arttıkça başarı oranı artan sonuçlar elde etmişler.

(Amasyalı & Diri, 2006), bu çalışmada Türkçe için n-gram modeli kullanılarak ilk kapsamlı bir metin sınıflandırması gerçekleştirilmiştir. Türkçe belgenin yazarının kimliğinin belirlenmesi, belgelerin metnin türüne göre sınıflandırılması ve bir yazarın cinsiyetinin otomatik olarak belirlenmesi gibi 3 farklı alanda çalışmışlar. Çalışmada yazarı bulunması istenen dokümanlardan istatiksel veriler, sık geçen kelimeler, n-gram'lar elde edilmiştir. Türkçe için n-gram modeli kullanılarak ilk kapsamlı bir metin sınıflandırması gerçekleştirilmiştir. Sınıflandırma yöntemi olarak NB, SVM, C 4.5 ve RF kullanılmış ve sonuçlar karşılaştırmalı olarak verilmiştir. Metnin yazarını, metnin türünü ve cinsiyetini belirlemedeki başarı sırasıyla %83, %93 ve %96 olarak elde edilmiştir.

(Doğan & Diri, 2010), bu çalışmanın amacı Türkçe içerikli bir dokümanın yazarı, türü ve yazarının cinsiyeti n-gram modeli kullanılarak belirlenmeye çalışılmıştır. N-gram modeli ile 2,4,4 gramlar kullanılmış ve 3 farklı veri seti ile toplam altı adet özellik vektörü oluşturulmuştur. DVM, Rastgele Orman (RO), K-En yakın komşuluk (K-EYK) gibi sınıflandırıcılar kullanılmıştır. Ng-ind ismini verdikleri yöntem geliştirilmiştir. Bu yöntem ile cinsiyet ve tür sınıflandırmada daha iyi sonuçlar alınmıştır.

(Byun & Lee, 2000), form tipindeki dokümanların sınıflandırılmasını DP matching algoritması üzerine araştırma yapmışlardır. Çalışmayı formlar ve evrak veri tipleri üzerine yapmışlardır. Doküman sınıflandırma için görüntü işleme yöntemini araştırmışlardır. Görüntülerde belirgin özelliklerini kullandılar. Örneğin bazı dokümanlarda bulunan belirgin olan tabloların görüntü özellikleri kullanılmıştır. Sonuç olarak sınıflandırma açısından görüntü sınıflandırması olduğundan dolayı yavaşlık dezavantajı olmuştur.

(Hu, Kashi, & Wilfong, 2000), Layout Similarity algoritması ile dokümanları sınıflandırma üzerine çalışmışlardır. Bu çalışmada OCR işlemi yapılmadan görüntüler sınıflandırılmıştır. Taranan dokümanların resim formatları ile sistem eğitilmiştir. Bu çalışmada görüntü sınıflandırma üzerinden doküman sınıflandırılması yapılmıştır. Sonuç olarak görüntülerin sınıflandırılması için her belge türünden olabildiğince fazla doküman ile eğitim yapılması gerektiği sonucunu çıkarmışlardır. Görüntü işleme ile doküman sınıflandırıldığı için dokümanların taranma kaliteleri de başarı oranında büyük önem arz ettiği sonucunu elde etmişlerdir.

(Csurka, Dance, Fan, Willamowski, & Bray, 2004), farklı bir yöntem ile görüntüleri bütünsel olarak ele alıp dokümanların görüntülerinde ayırt edici işaretlerler arayarak sınıflandırma yapmaktadır. Dokümanların görüntülerinin sınıflandırmasındaki karşılaşılan durumları üç ana başlığa ayırmışlardır. Bu başlıklar; tanıma, içeriğe dayalı görüntü erişimi, algılamadır. Sonuç olarak, görüntü işleme ile doküman sınıflandırma işlemine Naïve Bayes ve SVM sınıflandırma algoritmaları yöntemleri sunmuştur.

(Taylor, Lipshutz, & Nilson, 1995), doküman sınıflandırma işlemi için iki aşama üzerinde çalışmışlardır. Dokümanları sınıflandırmak için önce belgenin sütun sayısına göre sınıflandırılması ikinci olarak dokümanlar arasında bulunan işlevsel farklılıkları sınıflandırma üzerine çalışmışlardır. Sonuç olarak belgenin sütun sayısına göre bölünmesi sınıflandırmada başarıyı artırmıştır.

(Lazebnik, Schmid, & Ponce, 2006), bu çalışmada geometrik uygunluğa bağlı olarak sınıflandırma üzerine çalışılmıştır. Bu çalışmadaki yöntem ile görüntü içinde bulunan resimlerin daha ince alt bölgelere bölerek her bir alt bölgenin sahip olduğu özelliklerin histogramlarını hesaplayarak çalışır. Ortaya çıkan görüntü uzayının sınıflandırılması üzerine çalışılmıştır. Bu şekilde sınıflandırma da başarılı sonuçlar elde etmişlerdir.

(Kumar, Ye, & Doermann, 2014), belge görüntüsünün yapısal benzerliğini tanımlaya yönelik bir yaklaşım sunmaktadır. İlk olarak eğitim verisinden çıkarılan bir kod kümesi oluşturulmuştur. Daha sonra her bir belge tipi kodlanıyor. Görüntüler yinelemeli olarak bölümlere ayrılarak kod sözcüklerinin histogramları hesaplanmıştır. Bu hesaplar ile modeller oluşturulmuştur. Belgenin yapısal benzerliği tanımlama ile başarılı bir şekilde doküman sınıflandırma işlemi yapılmıştır.

(Kang, Kumar, Ye, Li, & Doermann, 2014), Convolutional Neural Network (CNN) algoritması ile belge görüntü sınıflandırması yapılmıştır. CNN algoritması derin öğrenme algoritmasıdır. Dokümanların görüntüleri piksellerinden özellikleri öğrenme üzerine çalışmışlardır. Belgenin hiyerarşik doğasını algoritmanın doğruluğunu desteklemektedir. Belge görüntülerinde CNN performansını etkileyen faktörleri incelemişlerdir. Eğitim kümesi ne kadar büyük olursa kazanımlarında o kadar büyük olduğu sonucuna ulaşmışlar. RVL-CDIP isimli veri seti ile yapılan deneysel sonuçlarda % 90,8 doğruluk başarısı elde etmişler.

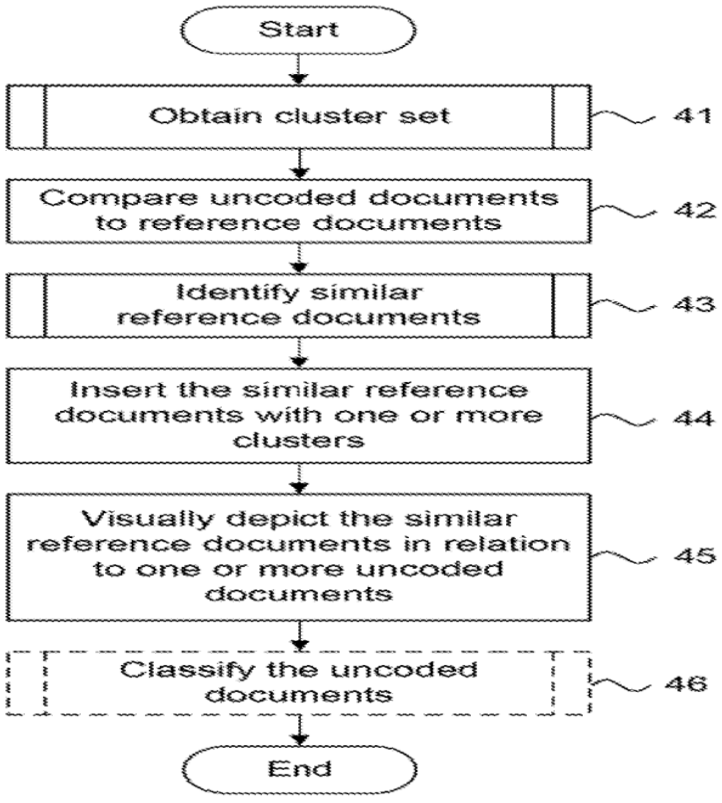
(Harley, Ufkes, & Derpanis, 2015), CNN algoritması ile dokümanların görüntü olarak sınıflandırılması üzerine çalışmışlardır. Görüntülerin öğrenme işlemi yaklaşık 400.000 dokümanın bulunduğu 16 kategorideki veri üzerinden yapılmıştır. CNN algoritması ile görüntü sınıflandırması ile başarılı sonuçlar elde edilmiştir. Şekil 1'de sonuçlar paylaşılmıştır.

Approach	**SmallTobacco**	**BigTobaccco**
Holistic BoW	.645	.446
H0V3 BoW	.679	.483
H2V0 BoW	.652	.461
H2V3 BoW	.681	.493
Pyramid BoW	.687	.491
Small holistic CNN (random init.)	.643	.851
Header CNN	.710	.849
Left body CNN	.667	.827
Right body CNN	.708	.795
Footer CNN	.622	.794
Holistic CNN	.756	**.898**
Holistic CNN (random init.)	.634	.878
Ensemble of CNNs	**.799**	.893

Şekil 1 CNN algoritması ile doküman sınıflandırma sonuçları(Harley et al., 2015)

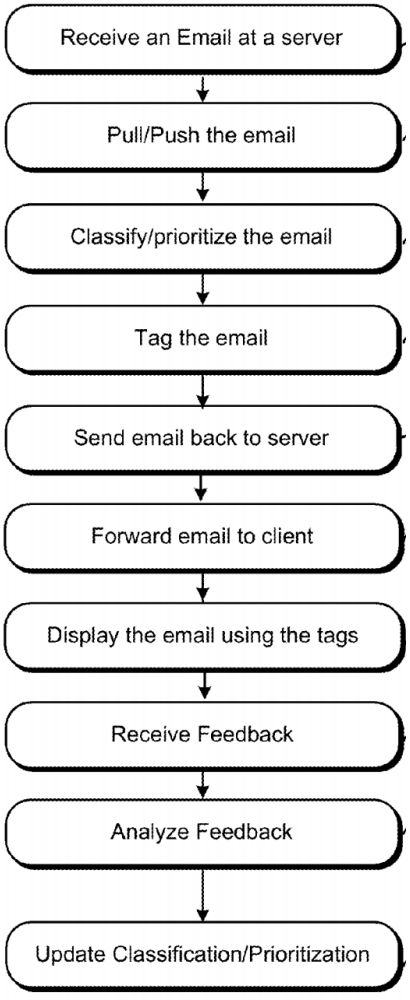
2.2. Doküman Sınıflandırma Patentli Çalışmalar

(Knight, 2017), US20170277998A1 numaralı patent çalışmasında şekil 2'deki algoritmanın patentini almışlardır. Patent içeriği; öncelikle veri seti oluşturulur. Bu oluşturulan veri setinden referans atanmış dokümanlar ile kodlanmamış dokümanlar karşılaştırılır. Benzer referanslar belirlenir. Benzer referansa sahip dokümanlar tekrar gruplanır. Bir veya daha fazla kodlanmamış belgeyle ilgili olarak benzer referans belgelerini görsel olarak tasvir edilir, böylelikle kodlanmamış dokümanlar sınıflandırılmış olur. Bu patentte kodlanmış kümeleri yöntemi kullanılmıştır. Kodlanmış belgeler bir dizi referans belge ile karşılaştırılarak doküman sınıflandırma yapılmıştır.



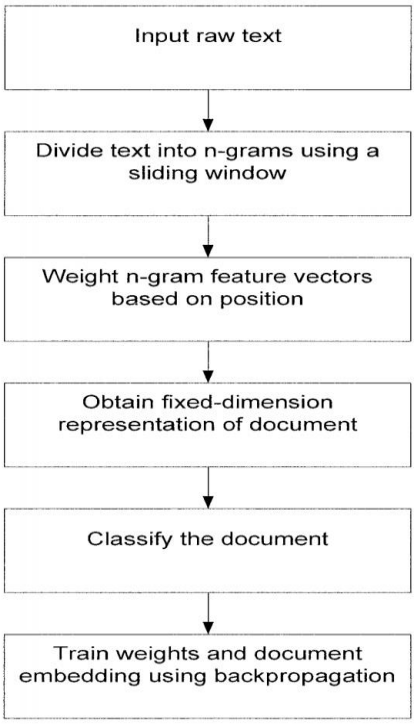
Şekil 2 Belge Sınıflandırma Önerileri Sağlama Sistemi ve Yöntemi

(Mchenry, 2016), US9378265B2 numaralı patentte elektronik doküman sınıflandırma üzerinedir. Elektronik belgeler analiz edilir ve daha sonra bir dizi oluşturulup sınıflandırılır. Yeni bir elektronik doküman tipi eklendiğinde diziler güncellenir. Şekil 3'te görüldüğü üzere patentin algoritması bulunmaktadır. Bu algoritmada e-mail sunucuları üzerinde sınıflandırma çalışmasıdır. Dokümanları analiz etmek için doğal dil işleme yöntemi kullanılmıştır. Doğal dil işleme yöntemleri ile kelimenin köklerine ayrılıp, derin öğrenme yöntemi ile doküman sınıflandırma yapmışlardır.



Şekil 3 Elektronik belge sınıflandırma patenti (Mchenry, 2016)

(Qi, 2014) US8892488B2 numaralı patent çalışmasında n-gram algoritması ile dokümanda metinlerin ardışık olarak bulunması üzerine doküman sınıflandırma yapılması üzerinedir. Öncelikle veri seti oluşturulur. Dokümanlar metinlerine n-gram algoritması ile ayrılır. Metinler ağırlıklarına göre konumlandırılır. Doküman sınıflandırılır. Geri yayılım algoritması ile doküman sınıflandırma öğrenimi sağlanır. Şekil 4'te patent belgesindeki algoritması verilmiştir.



Şekil 4 Ağırlıklı denetimli belge sınıflandırması (Qi, 2014)

Bu çalışmada literatürdeki çalışmalardan farklı olarak yakın dönemde ortaya çıkmış Microsoft ürünü olan .NET platformuna dahil olan açık kaynak kodlu makine öğrenmesi kütüphanesi ile çalışılmıştır. Literatürdeki çalışmalarda özetle iki yöntem kullanılmıştır. Birinci yöntem dokümanlar metin halde kabul edilip metin hali sınıflandırılmıştır. İkinci yöntem dokümanların taranmış hali olan resimleri görüntü işleme algoritmaları yardımı ile sınıflandırma çalışmaları yapılmıştır. Şirketler genellikle dokümanlarının taranmış hallerini resim formatında saklamaktadır. Dokümanları sadece metin formatında kabul edip doğrudan metin sınıflandırma algoritmaları kullanan çalışmalar bu noktada ihtiyacı karşılayamamaktadır. Ayrıca resim sınıflandırma algoritmaları görüntü işleme noktasında daha fazla kaynağa ihtiyaç duyduğundan özellikle zaman açısından maliyetli bir işlem olabilmektedir. Bu çalışmada, taranmış dokümanlar önce metin haline dönüştürülmüştür. İkinci aşamada metinlerin

sınıflandırılması sağlanmıştır. Bu yöntemle taranmış halde bulunan resim formatındaki dokümanların sınıflandırılması için yeni bir yöntem sunulmuştur. OCR kütüphanesi olarak Windows OCR ve sınıflandırma kütüphanesi olarak ML.Net kullanılmıştır. ML.Net kütüphanesi açık kaynak kodlu bir kütüphanedir. Bu sayede sürekli gelişim içinde olan bir kütüphane aracılığı ile başarı oranı da doğrusal olarak yeni versiyonlarda artması beklenmekledir.

3. MATERYAL VE YÖNTEM

Bu bölümde çalışmada kullanılan yöntemler ve araştırma sonucunda oluşturulan uygulama hakkında bilgi verilmiştir. Bu çalışma da kaynak araştırması bölümünde sunulan benzer yöntemlerden farklı bir yöntem sunularak sınıflandırma yapılmaya çalışılmıştır. Ayrıca araştırmaları somutlaştırmak için Windows üzerinde çalışabilecek bir uygulama yazılarak araştırmanın faydalı olabilmesi ve sonuçların daha net görülebilmesi amaçlanmıştır. Araştırma için yazılan kodlar ve ara yüz uygulaması için yazılan tüm kodlar Github üzerinde açık kaynak platformu kullanıcılarına sunulacaktır. Bu tezde hem yeni yöntemler bulmak hem de faydalı bir ürün ortaya koymak amaçlanmıştır.

Operating System	Windows 10 Pro 64-bit (10.0, Build 19042) (19041.vb_release.191206-1406)
Language	Turkish (Regional Setting Turkish)
System Manufacturer	Parallels Software International Inc.
System Model	Parallels Virtual Platform
BIOS	16.5.0 (49183) (type UEFI)
Processor	Intel(R) Core(TM) i9-9980HK CPU @ 2.40GHz (8 CPUs), ~2.4GHz
Memory	16384MB RAM
Available OS Memory	16362MB RAM
Page File	15648MB used, 8649MB available
Windows Dir	C\WINDOWS
DirectX Version	DirectX 12
DX Setup Parameters	Not found
User DPI Setting	192 DPI (200 percent)
System DPI Setting	192 DPI (200 percent)
DWM DPI Scaling	Enabled
Miracast	Not Available
Microsoft Graphics Hybrid	Not Supported
DirectX Database Version	1.0.8
DxDiag Version	10.00.19041.0928 64bit Unicode

Şekil 5 Bu çalışma da kullanılan bilgisayar özellikleri

Tez çalışması boyunca kullanılan bilgisayarın özellikleri şekil 5 de gösterilmiştir. Çalışma boyunca yüksek miktarda veri işlendiğinden dolayı performans gerektiren bir bilgisayar olması gerekmektedir. Alt bölümlerde araştırma aşamasında mimari, kullanılan yöntemler ve yazılan uygulama hakkında bilgi verilmiştir.

3.1. Sistem Mimarisi

Bu çalışmanın amacı taranan dokümanlardan sınıflandırma yapmaktır. Bunun için ilk önce çalışmada kullanılacak veri seti bulunmuştur. Sonraki alt başlıklarda bu aşamalar hakkında detaylı bilgi verilmiş olup veri seti referansı da eklenmiştir. Veri setinde 400.000 adet taranmış resim halinde doküman bulunmaktadır. Bu 400.000 verinin 320.000'i eğitim, 40.000'i doğrulama ve 40.000 de test amaçlı kullanılmıştır. Şekil 6 da görüldüğü üzere veri setindeki dokümanlardan sınıflandırma sonucunda uygulamada kullanılmak üzere model oluşturulmuştur.

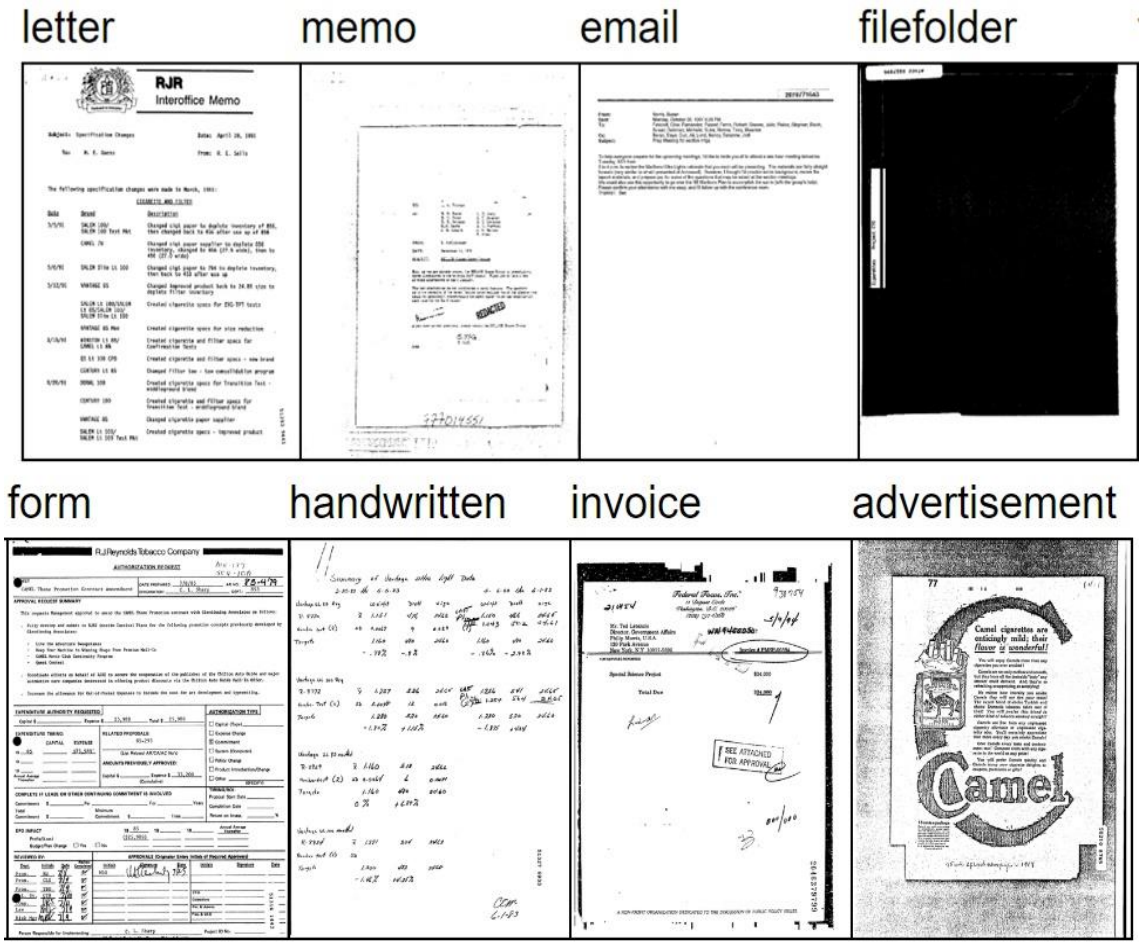
Şekil 6 Model oluşturulma aşaması

3.2. Verilerin Toplanması

Bu bölümde tez projesinde kullanılan veri setinin oluşturulması ve yapılan ön hazırlıklar hakkında bilgiler verilerek uygulanacak adımlardan bahsedilmiştir. Doğru bir eğitim seti oluşturmak yapay zekâ çalışmalarında başarı oranını yükseltmektedir. Eğitim seti üzerinde yapılan işlemler ve yapay zekâ algoritmaları hakkında detaylı araştırmalar ve elde edilen sonuçlar paylaşılmıştır.

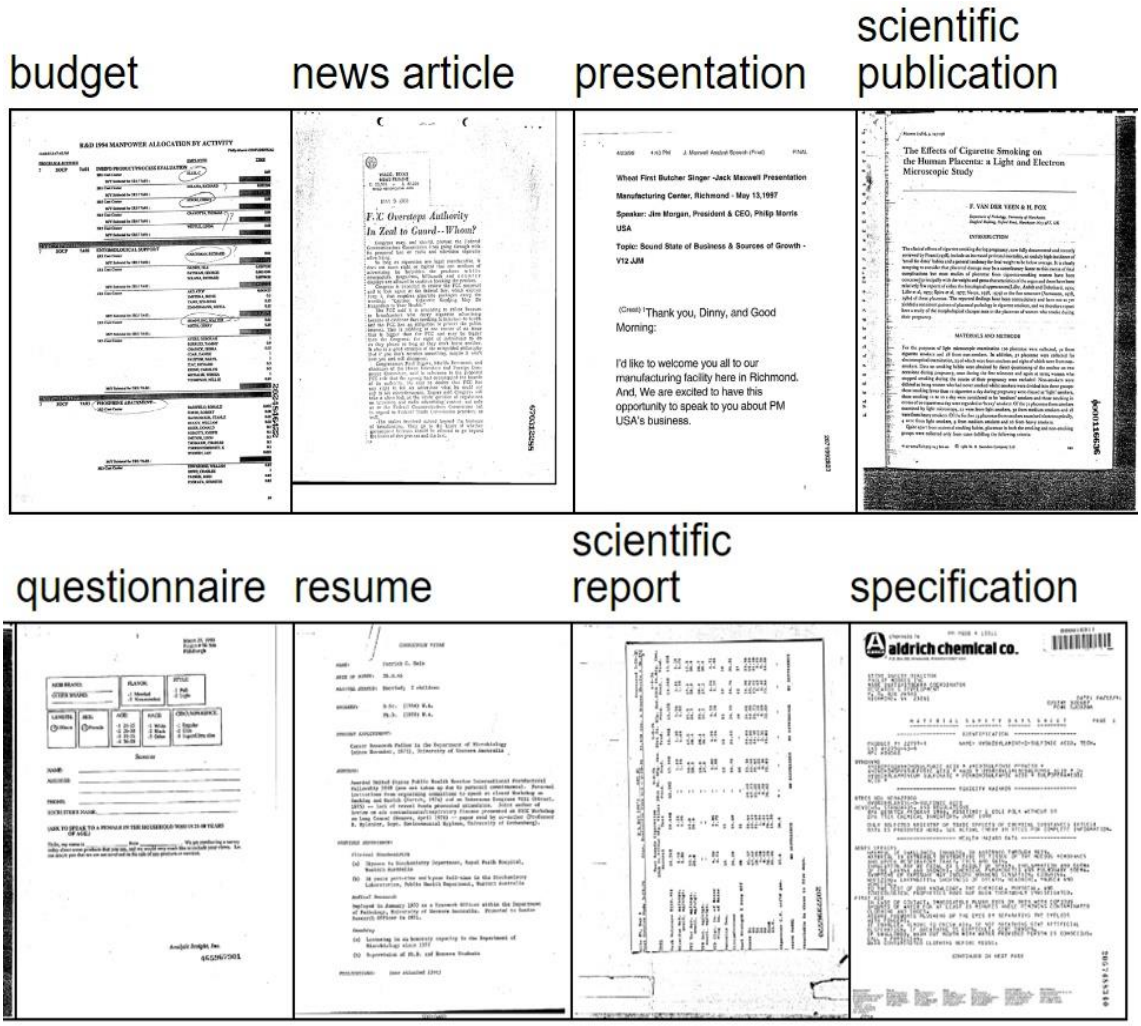
3.2.1. Veri Setinin Düzenlenmesi

Bu çalışmada kaynak veri olarak Carnegie Mellon Üniversite'sinin RVL-CDIP (Ryerson Vision Lab Complex Document Information Processing) (Adam W. Harley 2020) veri seti kullanılmıştır. Veri setinde 16 farklı tür bulunmaktadır. Her bir türde 25.000 gri tonlamalı görüntüden oluşan 400.000 adet resim verisi kullanılmıştır. Bu verilerin 320.000 adeti eğitim verisi, 40.000 adet doğrulama verisi ve 40.000 adet test verisi olarak ayrılmıştır. Doküman türleri harf, form, e-posta, el yazısı, reklam, bilimsel, rapor, bilimsel yayın, şartname, dosya klasörü, haber makalesi, bütçe, fatura, sunum, anket, özgeçmiş ve not olarak 16 tür bulunmaktadır. Bu resim dosyaları tek tek OCR işlemine tabi tutulmuş ve metin haline getirilmiştir. Metin haline getirilen dosyalar 320.000 adet dosya tek bir dosya içinde metin içeriği ve metin türü olmak üzere iki kolondan oluşacak şekilde düzenlenmiştir. Bu tek metin dosyası eğitim verisinin düzenlenmiş halidir. Eğitim setinin birinci kolonunda dokümanın içeriği ikinci kolonunda ise dokümanın tipi bulunmaktadır. Bu iki kolonlu eğitim verisini ML.NET kütüphanesi makine öğrenmesi yapılmıştır. Öğrenme sonucunda model dosyası üretilmiştir. Eğitim verisinde doküman içeriği ve tipi olduğu için hangi tür dokümanlar hangi tür içeriklere sahip olur bilgisi ve sınıflandırma sonucu model dosyasına yazılmıştır. Model dosyası yeni gelen doküman tiplerinde dokümanın türünü belirlemede kullanma amacıyla oluşturulmuştur.



Şekil 7 Veri setinde bulunan dokümanlardan ilk 8 doküman tipi (Adam W. Harley 2020)

Şekil 7 de 16 farklı doküman tipinden 8 tanesine yer verilmiştir. Bu doküman tipleri şirketlerin kullandığı evrak, not, e-mail, dosya klasörleri, formlar, el yazıları, faturalar ve ilandır. Şirketler tarafından sıklıkla bu dokümanlardan metin haline dönüştürülmesi en zor olan el yazısı olan dokümanlardır. Bu çalışmada kullanılan Microsoft OCR kütüphanesi bilgisayar yazı tiplerinde daha başarılı olduğu yapılan testlerde açık bir şekilde görülmüştür. El yazısı tanıma işleminin başarısı; çok sayıda el yazısı ile eğitilmesine resmin netliğine, yazan kişinin yazısının okunaklı ve düzgün olması gibi etkenlere bağlıdır.



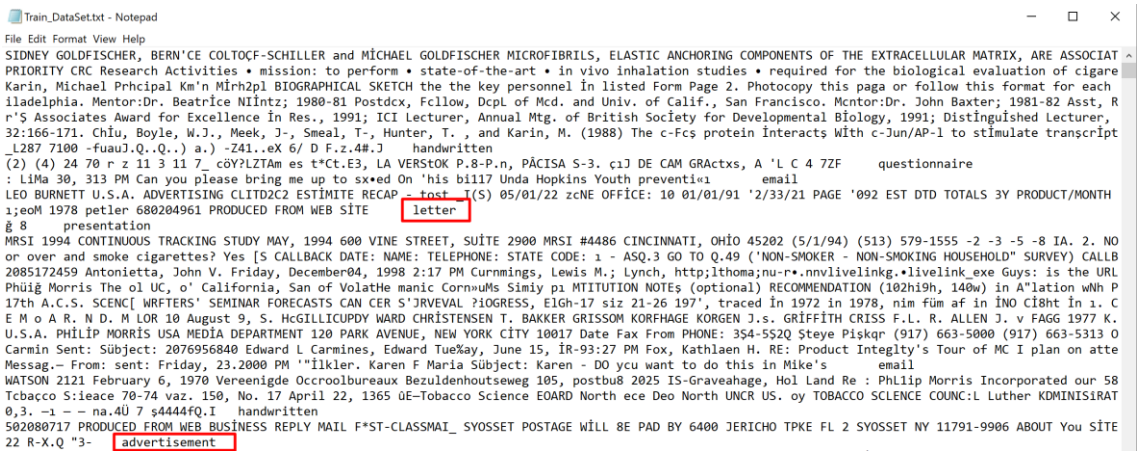
Şekil 8 Veri setinde bulunan 16 doküman tipinden son 8 doküman tipi

Şekil 8 de veri setinde bulunan diğer 8 doküman tipi gösterilmektedir. Bu doküman tipleri; bütçe dosyaları, haber makaleleri, sunum, bilimsel yayınlar, anket, özet, bilimsel rapor ve şartname dokümanlarından oluşmaktadır. Bu veri setindeki dokümanlar yinelemeli bir şekilde taranmıştır. Taranan dokümanların yazı içerikleri iki kolan olacak şekilde tek bir dosyada birleştirilmiştir. Birinci kolonda dokümanın OCR işleminden sonra okunan içeriği, ikinci kolonunda bu dokümanın tipi bulunmaktadır. Bu şekilde dokümanın eğitim verisi oluşturulmuştur.

3.2.2. Veri Setinden Eğitim Verisi Oluşturma

Dokümanların taranmış hali olan resim formatında bulunan veri setinden eğitim ve test verisi oluşturulmuştur. 400.000 adet veriden 320.000 adet resim eğitim verisi olarak ayrılmıştır. Eğitim verisinin OCR işlemi ile metin türüne dönüştürülmesi için C# dilinde console uygulaması yazılmıştır. Console uygulaması ara yüz gerektirmeyen arka planda çalışan kullanıcı etkileşimi console ekranından olan bir uygulamadır (A, 2020). Console uygulaması yazılmasının sebebi, eğitim verisi oluşturmak için herhangi bir ara yüze gerek olmamasıdır.

Eğitim veri seti iki kolon olacak şekilde tasarlanmıştır. Birinci kolonda taranan dokümanın OCR ile taranmış içeriği, ikinci kolonda ise dokümanın türü yer almaktadır.



Şekil 9 İki kolonlu eğitim verisinin örnek görüntüsü

Şekil 9 da resim halindeki dokümanlardan elde edilen eğitim verisinin içeriği görülmektedir. İki kolon halinde bulunan eğitim verisinde ilk kolon resmin içinde bulunan metnin, ikinci kolon ise işaretli olarak dokümanın türü gösterilmektedir. Makine öğrenimi aşamasında bu iki kolonlu eğitim verisi, sisteme şekil 5'teki gibi yüklenecektir. Kolon ayracı olarak tab karakteri konulmaktadır. Bu ayraç sayesinde txt veri tipinde olan doküman kolayca kolonlarına ayrılmaktadır.

```
1 reference | Ali Eser, 165 days ago | 1 author, 1 change
private static void PrepareDataAsync(int index, int count)
{
    var basePath = @"G:\TEZ\rvl-cdip\labels\train.txt";

    var training = new List<LetterData>();
    var test = new List<LetterData>();

    /* label çekilir */
    string[] lines = File.ReadAllLines(basePath);
    lines = lines.ToList().GetRange(index, count).ToArray();
    var texts = new List<string>();

    int counter = 0;
    ConcurrentBag<LetterData> returnList = new ConcurrentBag<LetterData>();
    Parallel.ForEach(lines, new ParallelOptions() { MaxDegreeOfParallelism = Environment.ProcessorCount }, line =>
    {
        var textParts = line.Split(' ').ToList();
        textParts.RemoveAll(s => string.IsNullOrEmpty(s));
        // train
        var result = GetContent(_basePath + textParts[0].Replace("/", @"\"));
        if (!string.IsNullOrWhiteSpace(result))
        {
            returnList.Add(new LetterData { Text = GetContent(_basePath + textParts[0].Replace("/", @"\")), Label = _categoriesLabel.Find(x => x.Item2 == textParts[1]).Item1 });
        }

        counter++;
        Console.WriteLine(counter.ToString() + " ---- " + lines.Count().ToString());
    });

    // Doküman içeriği ve doküman tipi arasına tab konulmuştur
    File.AppendAllLines(_trainingSet, returnList.Select(s => $"{s.Text}\t{s.Label}"));
}
```

Şekil 10 Eğitim verisi oluşturulması

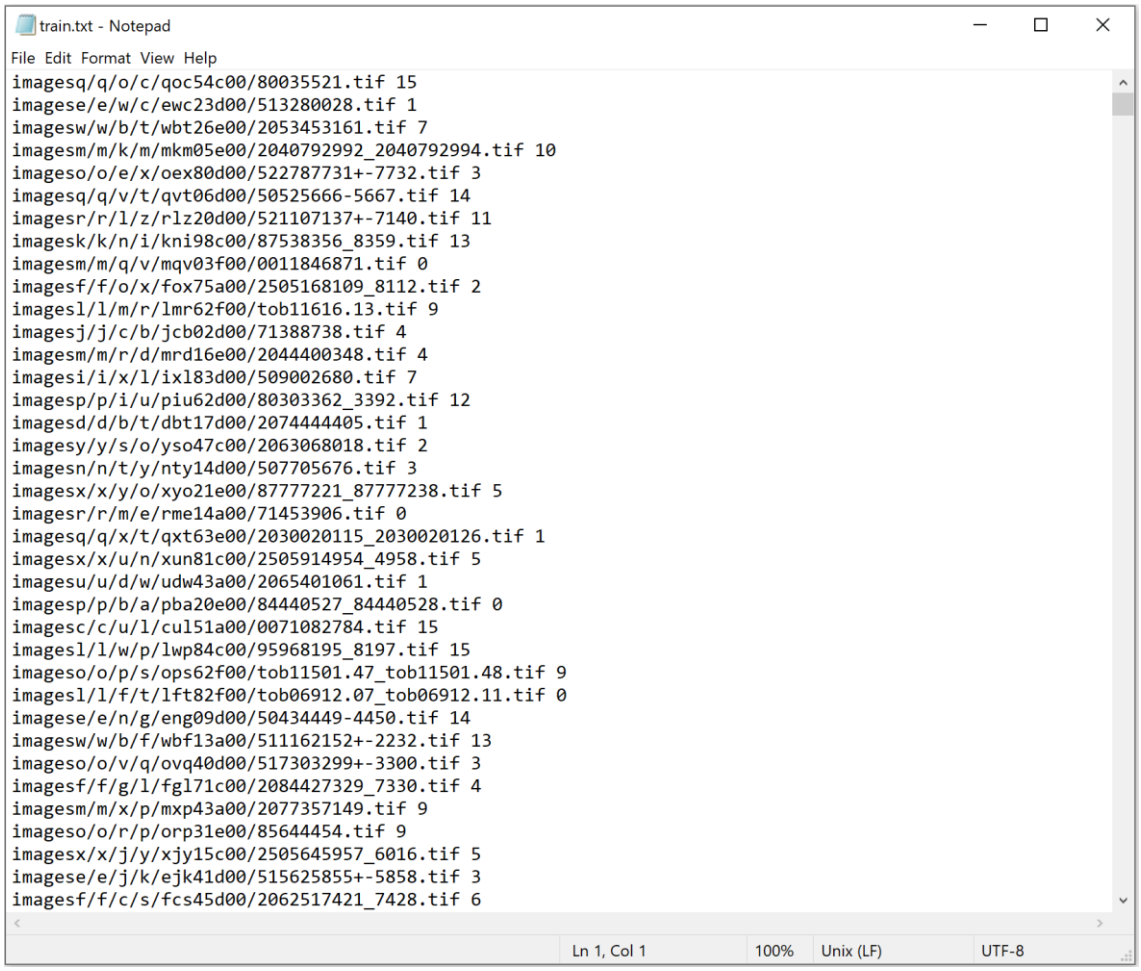
Şekil 10 da bulunan metot ile doküman içeriklerinden veri seti hazırlanmıştır. RVL-CDIP veri seti beraberinde hangi dokümanın hangi tipte olduğu da ayrıca bir metin dosyası halinde sunulmuştur. Bu metot OCR ile alınan doküman içeriklerinin ve doküman tiplerinin tek bir dosya da birleştirilmesini sağlamaktadır. RVL-CDIP veri seti aracılığı ile indirilen doküman haritasına göre metin dosyası satır halinde okunmuştur. Satırlarda hangi dokümanın hangi klasörde olduğu ve dokümanın tipi bulunmaktadır.

where the categories are numbered 0 to 15, in the following order:

0. letter
1. form
2. email
3. handwritten
4. advertisement
5. scientific report
6. scientific publication
7. specification
8. file folder
9. news article
10. budget
11. invoice
12. presentation
13. questionnaire
14. resume
15. memo

Şekil 11 RVL-CDIP Veri setinde doküman türlerinin numaralandırılması

Şekil 11 de görüldüğü gibi RVL-CDIP veri setinde doküman türlerinin numaraları gösterilmiştir. Doküman türlerinde veri setinde metin olarak yazmak yerine bir kerelik şekil 11'deki gibi verilen numaralar yazılmıştır. Bu durum programlama açısından hız kazandırmaktadır.



Şekil 12 RVL-CDIP Veri setinde doküman türlerinin klasör adresleri

Şekil 12 de görüldüğü gibi RVL-CDIP veri setinde dokümanların bulunduğu klasörlerin adres bilgisi ve o klasörlerdeki dokümanların türü bulunmaktadır. Bu çalışma için yazılan console uygulamasında bu satırlar okunarak döngü ile klasörlerdeki dokümanlar okunmuştur.

```
private static List<(string, string)> _categoriesLabel = new List<(string, string)> {
    ("letter","0"), ("form","1"), ("email","2"), ("handwritten","3"), ("advertisement","4"),("scientific report","5"),
    ("scientific publication","6"),("specification","7"), ("file folder","8"), ("news article","9"),
    ("budget","10"),("invoice","11"),("presentation","12"),("questionnaire","13"),
    ("resume","14"),("memo","15")};
```

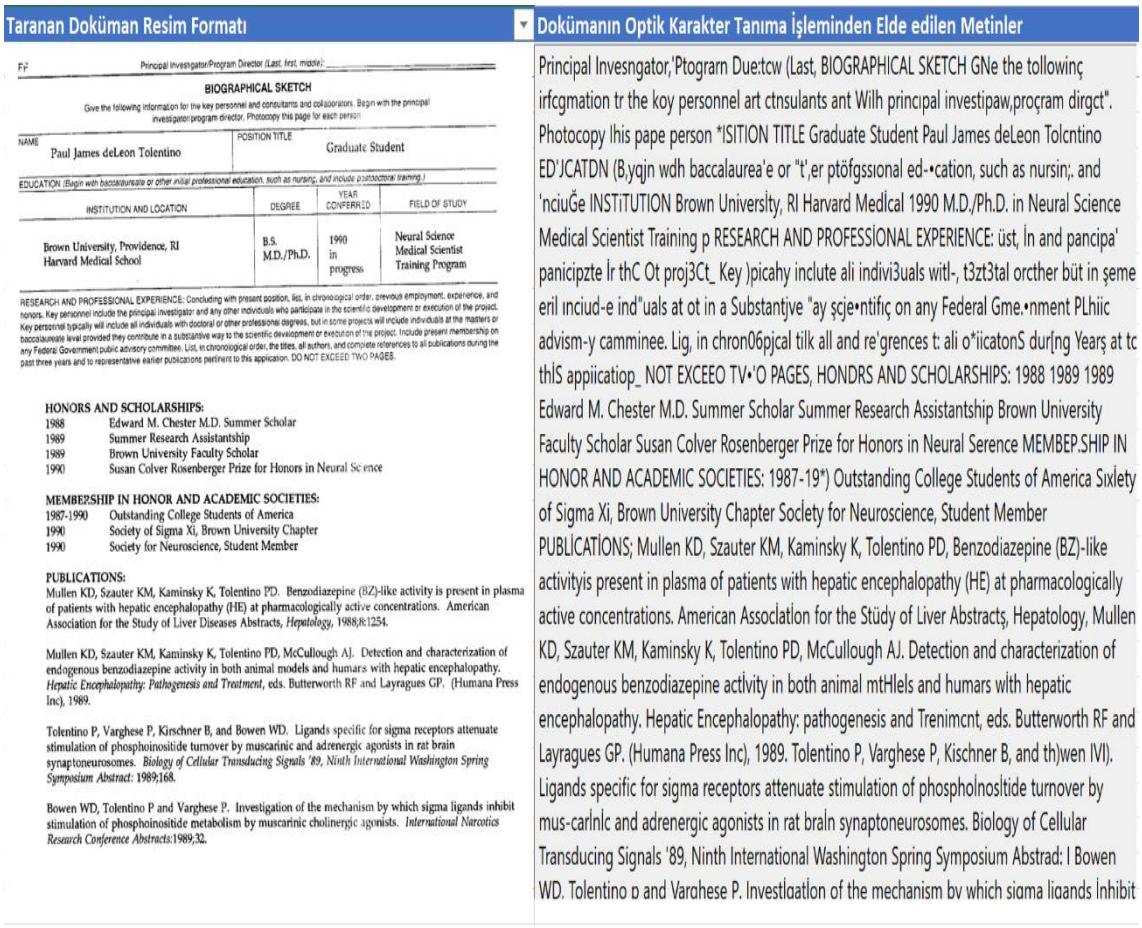
Şekil 13 Doküman isim ve numara eşleştirilmesi

Şekil 13'teki gibi dokümanların isimleri ve atanan numaraları bu çalışma için yazılan uygulamada eşleştirilmiştir.

3.3. Optik Karakter Tanıma (Optical Character Recognition OCR)

Yapay sinir ağların Optik karakter tanıma işlemi basılı veya her türlü bilgi kaynağının tarayıcılar yardımıyla tarama işlemi yapılarak bilgisayar ortamına aktarılması ile oluşan resim formatındaki dosyaların metin olarak yazıya dönüştürülmesi işlemidir. Optik karakter tanıma işleminde resim formatındaki dosayalar pikseller taranarak anlamlı bir harf aranır.

Bu çalışmada resim formatında olan taranmış şirket dokümanları optik karakter tanıma işlemi ile dokümanların her biri metin karşılığına çevrilmiştir. Metin haline çevrilme işlemi Windows OCR kütüphanesi ile yapılmıştır. Windows OCR Windows 10 ile birlikte gelen resimden yazı elde edilmesine olanak sağlayan .NET kütüphanesidir. Bu kütüphane yardımı ile offline olarak resimler optik karakterler olarak tanımlanabilir. Metin haline dönüştürme işleminde okunamayan karakterlerin kontrolleri yapılmıştır. Veri temizliği ne kadar düzgün olursa sonuçlar o kadar belirgin olacaktır.



Şekil 14 Orijinal Taranan doküman ve bu dokümanın OCR işlemi sonrası

Şekil 14 de görüldüğü üzere şirketin taranan dokümanı resim formatında bulunmaktadır. Bu doküman OCR işleminden sonra elde edilen metin şeklin sağ tarafında görüldüğü gibi çıkarılmaktadır. Çalışmada Carnegie Mellon Üniversite'sinin RVL-CDIP veri seti dosyaları tek tek OCR işleminden geçirilmiştir. Elde edilen verilen düzenli ve sıralı bir şekilde kayıt edilmiştir. Metin halinde kayıt edilmiş veriler makine öğrenmesi için hazır hale getirilmiştir. Eğitim verisi ve test verisi olarak iki grup halinde kayıt edilmiştir. Eğitim veri setinde doküman içeriği ve doküman tipi olarak iki kolon halinde kayıt edilmiştir. Test veri setinde ise tek kolon halinde doküman içeriği olarak kayıt edilmiştir.

3.3.1 OCR İşlemi Yöntemleri

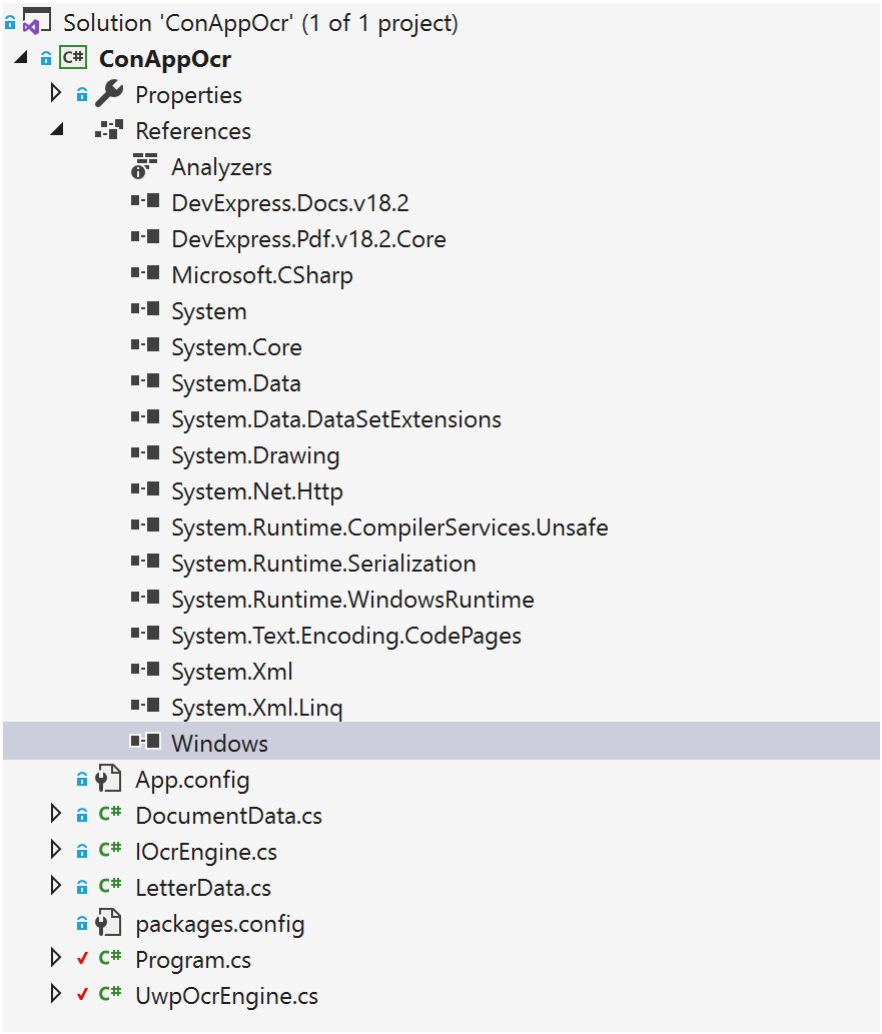
Taranmış dokümanların metin haline dönüştürülmesi işleminde kullanılan Windows OCR kütüphanesi offline olarak çalışmaktadır. Kütüphaneye yayıncı kuruluş olan Microsoft tarafından güncelleme geldikçe uygulamanın güncellemesine dahil ederek versiyon artışı yapılacaktır. Windows OCR kütüphanesinin avantajları;

i. Kullanıcının uygulamayı kullanırken internete bağlı olma zorunluluğuna olmaması
ii. Offline çalışması sayesinde yeni versiyonlarda uygulamanın herhangi bir problemle karşılaşmasının önüne geçilmesi
iii. OCR kütüphanesinde yoğun bir şekilde karşılaştırma ve veri tabanına ihtiyaç duyacağı için offline olmasından dolayı yoğun internet bağlantısı tüketmeye ihtiyaç olmaması
iv. İnternete bağlanma ihtiyacı olmadığı için daha hızlı olması
v. Daha hızlı olduğu için müşteri bekleme süresinin kısaltılması
vi. Kütüphaneyi sağlayıcı firma Microsoft olduğu için Windows tabanlı işletim sistemlerinde yüksek kararlılık olması olarak sıralanabilir.

Bu avantajları dikkate alındığında uygulamanın çalışma kararlılığı ve hızlı olmasına değerlendirilmiştir. Şirketler içinde kullanılan yazılımlarda hız ve çalışma kararlılığı son derece önemlidir. Bu uygulama özellikle banka şubelerinde kullanılırken müşterinin bekleme aşaması düşünülerek planlanmıştır. Müşteri bekleme süresinin düşürülmesiyle memnuniyetlerinin artması beklenmektedir.

3.3.2. OCR Uygulaması

Bu tez çalışması için Windows OCR kütüphanesi kullanılarak OCR uygulaması yazılmıştır. Uygulamanın amacı RVL-CDIP veri seti için verilen klasör yollarından dosyaların resim hallerini işleyip metin haline dönüştürmektir.



Şekil 15 OCR uygulaması referans listesi

Şekil 15 de bu çalışma için yazılmış olan OCR uygulamasının kütüphaneleri bulunmaktadır. OCR işlemini yapan asıl kütüphanesi ise şekildeki referansların en altında bulunan Windows.winmd referansıdır. Bu referans sayesinde OCR işlemi yapılmaktadır. Bu referansın çalışması için işletim sistemi minimum Windows 10 olması gerekmektedir

```
22      /// <summary>
23      /// Mains the asynchronous.
24      /// </summary>
25      /// <returns></returns>
        1 reference | Ali Eser, 171 days ago | 1 author, 1 change
26      private static async Task<string> RunOcrAsync(byte[] imageBytes)
27      {
28          OcrEngine ocrEngine = OcrEngine.TryCreateFromLanguage(new Language("en"));
29          if (ocrEngine == null) return null;
30
31          List<string> pageTexts = new List<string>();
32
33          using (var imgStream = new MemoryStream(imageBytes))
34          {
35              using (var image = Image.FromStream(imgStream))
36              {
37                  int pageCount = image.GetFrameCount(FrameDimension.Page);
38                  for (int i = 0; i < pageCount; i++)
39                  {
40                      image.SelectActiveFrame(FrameDimension.Page, i);
41                      using (var imgPageStream = new MemoryStream())
42                      {
43                          image.Save(imgPageStream, ImageFormat.Tiff);
44                          using (var randomAccessStream = await ConvertToRandomAccessStream(imgPageStream))
45                          {
46                              /* İşleme */
47                              BitmapDecoder decoder = await BitmapDecoder.CreateAsync(randomAccessStream);
48                              var ocrResult = await ocrEngine.RecognizeAsync(await decoder.GetSoftwareBitmapAsync());
49                              pageTexts.Add(string.Join(" ", ocrResult.Lines.Select(l => l.Text)));
50                          }
51                      }
52                  }
53              }
54          }
55
56          return string.Join(" ", pageTexts);
57      }
```

Şekil 16 OCR uygulaması ana fonksiyonu

Şekil 16 da bu çalışma için yazılan uygulamanın C# ile yazılmış ana fonksiyonu görülmektedir. Asenkron olarak çağrılacak şekilde yazılmıştır. Asenkron olarak threadler ile çalışmasına olanak sağlayan bu metot ile aynı anda birden fazla dokümanın işlenmesi sağlanmıştır. Threadli fonksiyonların yanında uygulamanın performanslı çalışması için usign ifadeleri kullanılarak çalışma anında üretilen nesneler dispose edilmiştir. Nesnelerin dispose edilmesi, .net tarafından geliştiricinin tercihine bırakılan bir işlemdir. Dispose; çalışma anından üretilen nesnelerin bellekten silinmesi işlemidir. Bu işlemin yapılması aynı zamanda uygulamaya performans kazandırması amaçlanmıştır. Çalışma için kullanılan 400.000 adet dokümanın taranması gerektiğinden dolayı bu şekilde performans kazandırabilecek yöntemlere de dikkat edilmiştir.

3.4. Yapay Sinir Ağları

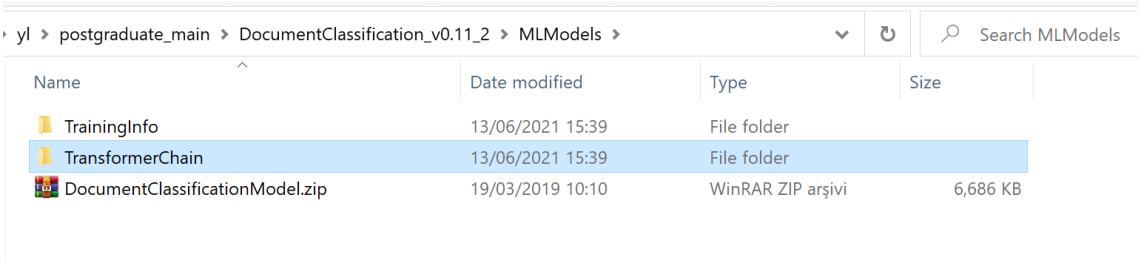
Yapay sinir ağları (YSA) insan sinir sistemi gibi davranır ve insan sinir sistemi gibi öğrenme işlemlerini gerçekleştirir. YSA giriş, gizli ve çıkış katmanlarından oluşmaktadır. Sinir hücreleri bir araya gelerek bir katmanı oluşturmakta ve her bir sinir hücresi sonraki katmandaki sinir hücrelerine ağırlık değerleri denilen parametre değerleri ile bağlanmaktadır. 1980'li yıllarda ortaya atılan makinanın insan gibi düşünebilmesi fikri ortaya atılmış, 1990'lı yıllara gelindiğinde ise Yapay Sinir Ağları teknolojisi iyiden iyiye hızlanmış ve büyük bir gelişme görülmüştür. Yapay sinir ağları örneklerle ilgili bilgiler toplamakta, genellemeler yapmakta ve daha sonra hiç görmediği örnekler ile karşılaştırılınca öğrendiği bilgileri kullanarak o örnekler hakkında karar verebilmektedir. Yapay sinir ağları bu öğrenebilme ve genelleme özellikleri nedeniyle günümüzde birçok bilim alanında geniş uygulama olanağı bulmakta ve karmaşık problemleri başarı ile çözebilme yeteneğini ortaya koymaktadır (Ergezer, 2003). Bu çalışmada sınıflandırma işlemi için ML.NET kütüphanesi kullanılmıştır. ML.NET kütüphanesinin barındırdığı Stochastic Dual Coordinate Ascent metodu ile eğitim yapılmıştır.

```
0 references | Ali Eser, 203 days ago | 1 author, 1 change
50  public static void BuildAndTrainModel(string DataSetLocation, string ModelPath, MyTrainerStrategy selectedStrategy)
51  {
52      // MLContext oluşturulması
53      var mlContext = new MLContext(seed: 1);
54
55      // STEP 1: Veri setinin yüklenmesi
56      // Veri seti yüklenirken satırlardaki kolonların ayrılması seperatör olarak tab kullanılmıştır
57      var trainingDataView = mlContext.Data.LoadFromTextFile<DocumentIssue>(DataSetLocation, hasHeader: true, separatorChar: '\t', allowSparse: false);
58
59      // STEP 2: Veri işleme için pipe oluşturulması
60      var dataProcessPipeline = mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: DefaultColumnNames.Label, inputColumnName: nameof(DocumentIssue.DocumentType))
61                      .Append(mlContext.Transforms.Text.FeaturizeText(outputColumnName: "DocumentContentFeaturized", inputColumnName: nameof(DocumentIssue.DocumentContent)))
62                      .Append(mlContext.Transforms.Concatenate(outputColumnName: DefaultColumnNames.Features, "DocumentContentFeaturized"))
63                      .AppendCacheCheckpoint(mlContext);
64
65      // Veri setinin özet bilgisinin console ekranında gösterilmesi
66      Common.ConsoleHelper.PeekDataViewInConsole(mlContext, trainingDataView, dataProcessPipeline, 2);
67
68      // STEP 3: Algoritma seçimi
69      IEstimator<ITransformer> trainer = null;
70      trainer = mlContext.MulticlassClassification.Trainers.StochasticDualCoordinateAscent(DefaultColumnNames.Label, DefaultColumnNames.Features);
71
```

Şekil 17 ML.NET kütüphanesi ile sınıflandırma metodu

Şekil 17 de bu tez için yazılan uygulamada kullanılan sınıflandırma metodu görülmektedir. Şekilde 70.satırda Stochastic Dual Coordinate Ascent metodu çağrılmıştır. Bu metot ile dokümanların sahip olduğu kelimeler ile ağırlıklar oluşturulmuştur. Yapay sinir ağının eğitilmesi için sonuçları belli olan eğitim veri seti ile eğitilmesi sağlanmıştır. 320.000 adet eğitim verisi ile yapay sinir ağı eğitimi sonucu model üretilmiştir. Bu model bilgisinde hangi kelime içeriklerinin hangi dokümanlara ait olduğu ve ağırlıkları bilgisi ile yapay sinir ağının öğrendiği bilgi yer almaktadır.

Yapay sinir ağı sonucu üretilen model kullanılarak offline bir şekilde uygulama aracılığı ile bir dosya türünün hangi türe ait olduğu belirlenmesi sağlanmıştır. Bu model aracılığı ile her bir işlem için tekrar bir öğrenmeye gerek kalmadan, model üzerinden tahmin işlemi gerçekleşmesi sağlanmıştır.



Şekil 18 Sınıflandırma sonunda üretilen model dosyası

Şekil 18 de sınıflandırma eğitimi sonunda üretilen model dosyası bulunmaktadır. Bu model dosyasında “TrainingInfo” ve “TransformerChain” adında iki klasör bulunmaktadır. Bu model tamamen ML.Net kütüphanesinin kendi düzenine göre oluşturulmuş ve manuel müdahale edilmemiştir. Bu model dosyası ile yeni dokümanların tipi tahmin edilebilecektir.

```
56
57      public void TestPredictionForSingleIssue()
58      {
59          var labeler = new Labeler(modelPath: ModelPath);
60          labeler.TestPredictionForSingleIssue();
61
62          var lines = File.ReadAllLines(@"D:\Tez Çalışması\Data\TestData.txt");
63          var line = lines[Convert.ToInt32(new Random().Next(lines.Count() - 1))];
64
65          DocumentIssue singleIssue = new DocumentIssue()
66          {
67              DocumentContent = line.Split('\t').First()
68          };
69
70          //Predict labels and scores for single hard-coded issue
71          var prediction = _predEngine.Predict(singleIssue);
72
73          _fullPredictions = GetBestThreePredictions(prediction);
74
75          Console.WriteLine("==== Displaying prediction of Issue with Title = {0} and Description = {1} ====", singleIssue.DocumentType, singleIssue.DocumentContent);
76
77          Console.WriteLine("1st Label: " + _fullPredictions[0].PredictedLabel + " with score: " + _fullPredictions[0].Score);
78          Console.WriteLine("2nd Label: " + _fullPredictions[1].PredictedLabel + " with score: " + _fullPredictions[1].Score);
79          Console.WriteLine("3rd Label: " + _fullPredictions[2].PredictedLabel + " with score: " + _fullPredictions[2].Score);
80
81          Console.WriteLine($"=============== Single Prediction - Result: {prediction.DocumentType} ===============");
82      }
83
```

Şekil 19 Model dosyası ile yeni doküman türlerinin test edilmesi

Şekil 19 da sınıflandırma sonucunda üretilen model dosyasından yeni doküman türlerinin test edilmesi metodu gösterilmiştir. Eğitim aşamasından sonra üretilen model dosyası uygulamaya yüklenerek sonraki yeni dokümanların sınıflandırılması tahmin edilmiştir. Resimdeki metot ile yeni taranan dokümanın olabileceği tür tahmin edilmektedir. Üç adet tahmin üretilmektedir. Bu üç tahmin içinde puan bilgisi de belirtilmiştir.

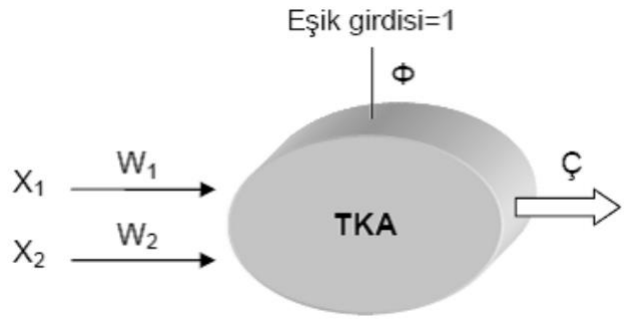
3.4.1. Yapay Sinir Ağ Modelleri

Yapay sinir ağ modellerinde bulunan hücrelerin bağlantı şekillerine ve aktivasyon fonksiyonlarına göre yapay sinir ağ modelleri geliştirilmiştir. Bu modeller temel olarak iki grupta toplanır.

- Tek katmanlı yapay sinir ağı mimarisi
- Çok katmanlı yapay sinir ağı mimarisi

3.4.1.1. Tek Katmanlı Yapay Sinir Ağı Mimarisi

Tek katmanlı yapay sinir ağları sadece giriş katmanı ve çıkış katmanından oluşmaktadır. Her bir bağlantının bir ağırlığı vardır.

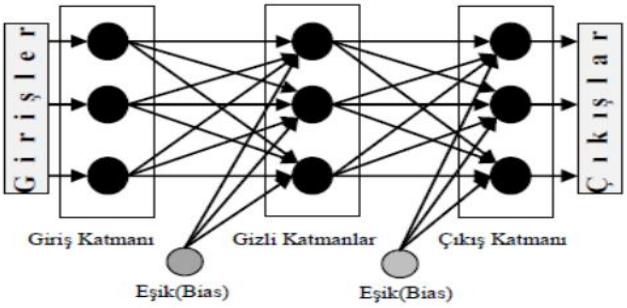


Şekil 20 Tek katmanlı yapay sinir ağı basit model (Öztemel, 2003)

İki adet girdisi ve bir çıktıdan oluşan tek katmanlı bir yapay sinir ağı Şekil 20 de verilmiştir. Şekilde görülen eşik girdisi bir olarak görülmektedir. Eşik değeri tek katmanlı yapay sinir ağında çıktı değerinin sıfır olmasını önler. Ağın çıktı değeri ağırlıklandırılmış girdi verisinin eşik değeri ile toplanması sonucu bulunur. Ağın girdi değeri aktivasyon fonksiyonundan geçirilerek çıktı hesaplanır.

3.4.1.2. Çok Katmanlı Yapay Sinir Ağı Mimarisi

Mimari olarak doğrusal olmayan aktivasyon fonksiyonuna sahip birçok nöronun birbirine hiyerarşik olarak bağlanması sonucu oluşturulan yapay sinir ağı modelidir. İleri beslemeli ağlar ve geri beslemeli ağlar olmak üzere ikiye ayrılmaktadır.



Şekil 21 Çok katmanlı yapay sinir ağı mimarisi

Şekil 21 de çok katmanlı yapay sinir ağı modeli gösterilmiştir. Giriş katmanı gizli katmanlar ve çıkış katmanları bulunmaktadır.

3.4.1.2.1 İleri Doğru Beslemeli Ağlar

Ağa gelen girdiler hiçbir değişiklik olmadan ara katmana gönderilir. Ara katmandaki her işlem elemanı girdi katmanındaki bütün işlem elemanlarından gelen bilgileri bağlantı ağırlıklarının etkisi ile alır.

3.4.1.2.2 Geriye Doğru Beslemeli Ağlar

Ağa sunulan girdi için ağın ürettiği çıktı ağın beklenen çıktıları ile karşılaştırılır. Bunların arasındaki fark hata olarak kabul edilir. Amaç hatanın minimuma indirilmesidir. Bu hata ağın değerlerine dağıtılarak bir sonraki iterasyonda hatanın azaltılması sağlanır. Toplam hatayı en aza indirmek için bu hatanın kendisine neden olan işlem elemanlarına dağıtılması gerekmektedir.

3.5. ML.NET Makine Öğrenimi Kütüphanesi

ML.NET, .NET platformuna dahil olan açık kaynaklı makine öğrenmesi kütüphanesidir. Farklı makine öğrenmesi ihtiyaçlarını eğitebilir, derleyebilir ve model oluşturarak kullanıma hazır hale getirilebilir bir yapay zekâ altyapıdır. Geliştiricilerin karmaşık makine öğrenimi oluşturmasına olanak tanır (Ahmed et al., 2019). Verilerden yapay zekâ eğitim işlemlerini C# ve F# dilleri ile yapılabilir. Bu çalışmada kodlar C# ile yazılmıştır.

```
var loader = new TextLoader().From<SentimentData>();
var featurizer = new TextFeaturizer("Features", "Text");
var learner = new FastTreeBinaryClassifier() { /* Some parameters */ };

var pipeline = new LearningPipeline()
    .Add(loader)
    .Add(featurizer)
    .Add(learner);
```

Şekil 22 Metin Okuma ve Binary Sınıflandırma Örneği

Şekil 22 de ML.NET kütüphanesi ile doküman sınıflandırmanın C# dilinde yazılmış bir kod örneği bulunmaktadır. Bu örnekte önce TextLoader sınıfı ile eğitim datasından okunacak olan kolonların sayısı ve tipleri belirtiliyor. Örnekte "Features", "Text" olmak üzere eğitim datasında dokümanın özelliği ve metin bilgisi yer alacağı bildirilmektedir. Hangi algoritma uygulanacağı bilgisi de "learner" objesi ile gösterilmiştir. Gerekli parametreleri bu şekilde vererek model oluşturma işlemi en temel seviyede başlatılabilir.

ML.NET yedi temel adımdan oluşmaktadır.

Birinci adım: ML.NET contexti oluşturulur. Bu adım tüm işlemler için başlangıç noktasıdır. Context içinde temel olarak makine öğrenimi için gerekli olan veriler ilk başlangıç değerleri set edilir. Context içinde eğitim verisinden okunacak olan kolonların bilgisi bulunmaktadır. Kolonların hangi ayraç ile ayrıldığı bilgisi verilmesi gerekmektedir.

İkinci adım: Analiz edilecek veriler yüklenir. Örneğin bu çalışmada verilerin içeriği ve etiketleri olmak üzere iki kolondan oluşacak şekilde veriler yüklenmiştir.

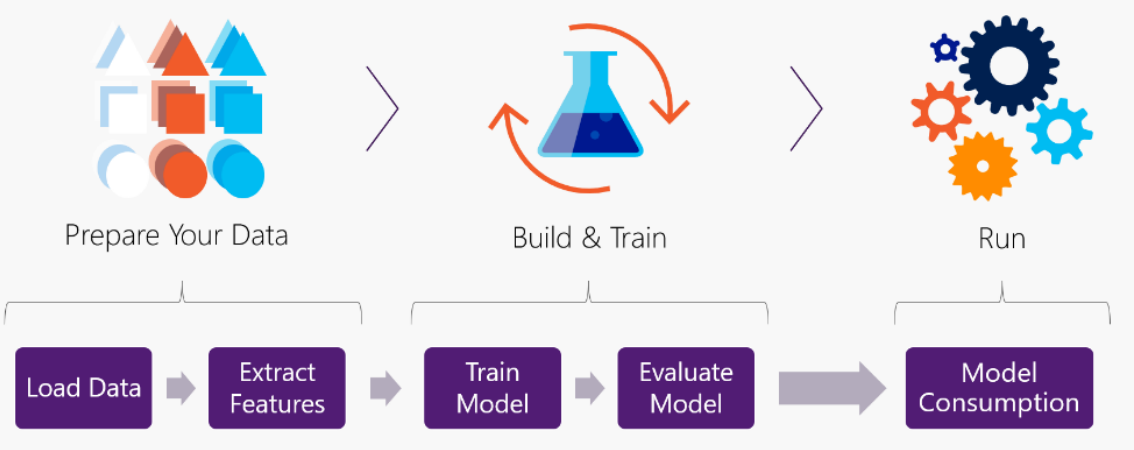
Üçüncü adım: Yüklenen ham veriler dönüştürülür.

Dördüncü adım: Dönüştürülen verinin makine öğrenimi için uygun algoritma seçilir.

Beşinci adım: Uygun algoritmaya göre ve girdi verilerine göre makine öğrenimi gerçekleştirilir. Makine öğrenimi için öğrenme algoritması seçilir ve öğrenme için pipe adımı başlatılır. Bu çalışmada doküman sınıflandırma için ML.NET sınıflandırma kütüphanesi algoritmalarından SdcaMultiClassTrainer etiketli sınıflandırma algoritması seçilmiştir. Makine öğreniminden sonra model oluşturulur.

Altıncı adım: Üretilen model değerlendirilir.

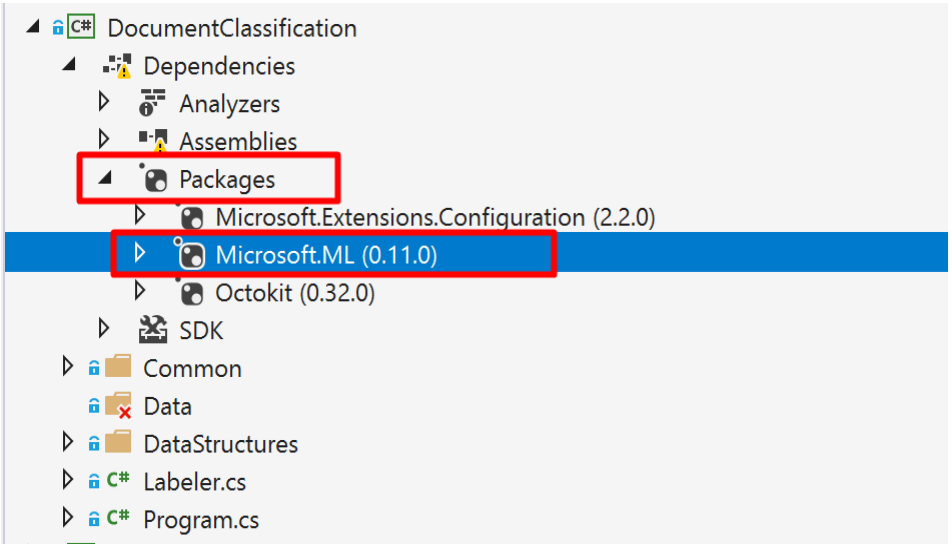
Yedinci adım: Eğitilen ve değerlendirilen model .NET uygulaması olarak entegre edilir.



Şekil 23 ML.NET makine öğrenimi adımları (Torre, 2020)

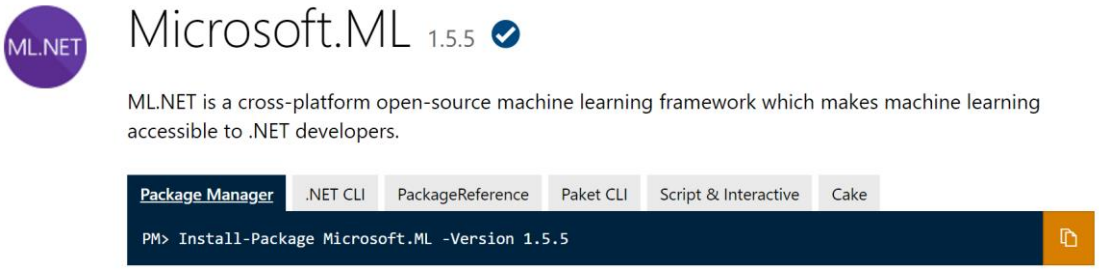
Şekil 23 de görüldüğü üzere ML.NET kütüphanesi aracılığı ile bir eğitim verisinin yüklenmesi, eğitim aşaması ve model üretilmesi aşaması gösterilmiştir. Son adımda üretilen model kullanılması gösterilmiştir. MLContext yapısı veri hazırlama, özellik belirleme, eğitim ve tahmin için bileşenler oluşturmamıza olanak sağlar. MLContext verileri yükleyebilir, dönüştürebilir, modeli eğitebilir, eğitilen model üzerinden testler yapılabilir.

ML.NET yapay zekâ kütüphanesi offline ve online olarak iki seçenekte kullanılabilir. Proje de offline olarak kullanılmıştır. Offline olarak kullanılması için proje de nuget referansı eklenmesi gerekmektedir.



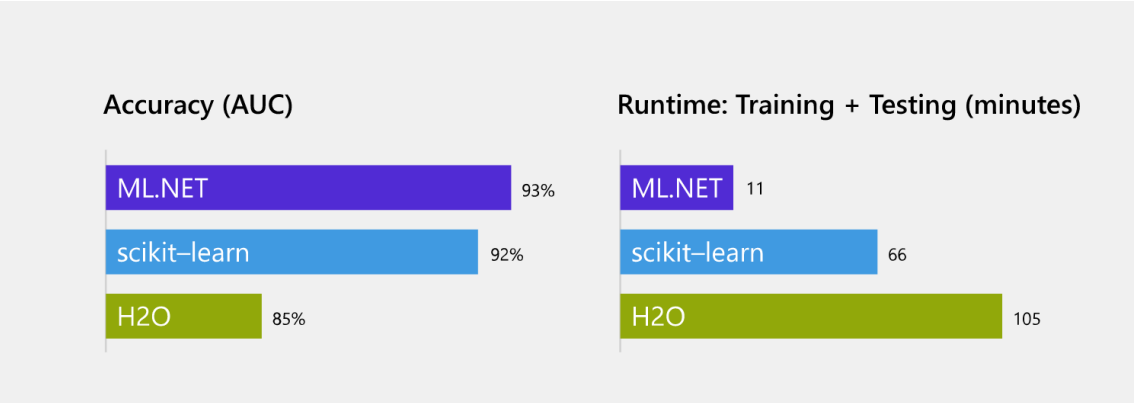
Şekil 24 ML.NET offline kullanım

Şekil 24 de ML.NET kütüphanesinin offline olarak eklenmesi gösterilmiştir. Nuget, .NET platformu ile yazılım geliştirirken kullanılan harici paketin yönetimi sağlanmaktadır. Özel bir alana hitap eden uygulama geliştirilmek istendiğinde .NET içindeki sınıflar bu ihtiyacı karşılamayabilir. Bu durumda Nuget, ihtiyaca uygun kütüphanenin projeye eklenmesini, güncellenmesini sağlayan paket yöneticisi olarak ihtiyacı karşılamaktadır.



Şekil 25 ML.NET paketi yüklenmesi

Şekil 25 de ML.NET kütüphanesinin nuget.org sitesinden alınmış görüntüsü bulunmaktadır. Proje eklenmesi şekildeki komutla yapılabilmektedir. Nuget.org aracılığı ile önceki versiyonları görülebilir. Projede istenilen versiyon ile çalışılabilmektedir.



Şekil 26 ML.NET ve diğer popüler yapay zekâ kütüphaneleri doğruluk ve performans göstergesi (Microsoft, 2020)

Şekil 26 da ML.NET yapay zekâ kütüphanesi ile diğer popüler yapay zekâ kütüphanelerinin karşılaştırılması yapılmıştır. Bu çalışmada ML.NET yapay zekâ kütüphanesinin seçilmesinin nedenlerinden biri de doğruluk başarısı ve performans değerlerinin yüksek olmasıdır. Çünkü şirketler için doğruluk başarısı ve yüksek performans oldukça önemlidir.

Tablo 1 Tablo 1 ML.NET kütüphanesi metotları ve karşılıkları (Microsoft, 2020)

ML.NET Kütüphanesi		
		ML.NET Metot Karşılığı
Veri Yükleme ve Kayıt Etme	Veri İşlemleri	DataOperationsCatalog
Veri Hazırlama	Dönüşüm İşlemleri	StochasticDualCoordinateAscent
Eğitim Algoritmaları	Sınıflandırma İşlemleri	StochasticDualCoordinateAscent
	Sıralama İşlemleri	RankingCatalog
	Kümeleme İşlemleri	ClusteringCatalog

Tablo 1 de ML.NET kütüphanesi ile eylemleri ve metotları gösterilmiştir. Veri yükleme, kayıt edilmesi, hazırlanması ve eğitim algoritmaları gösterilmiştir. Eğitim algoritmaları birden çok class sınıflandırması, Anormallik algılama, Kümeleme, Tahmin etme, Sıralamasına, Regresyon, Öneri ve Zaman serisi metotları da bulunmaktadır.

3.5.1. ML.NET Model Oluşturma

ML.NET makine öğrenimi kütüphanesi kullanılabilmesi için uygulama katmanında uygun bir sınıf oluşturulması gerekmektedir. Bu çalışmada iki kolonlu bir veri yapısı oluşturulmuştur.

```
namespace DocumentClassification.DataStructures
{
    12 references | Ali Eser, 44 days ago | 1 author, 1 change
    internal class DocumentIssue
    {
        [LoadColumn(0)]
        public string DocumentContent;

        [LoadColumn(1)]
        public string DocumentType;
    }
}
```

Şekil 27 ML.NET doküman yapısı veri seti sınıfı

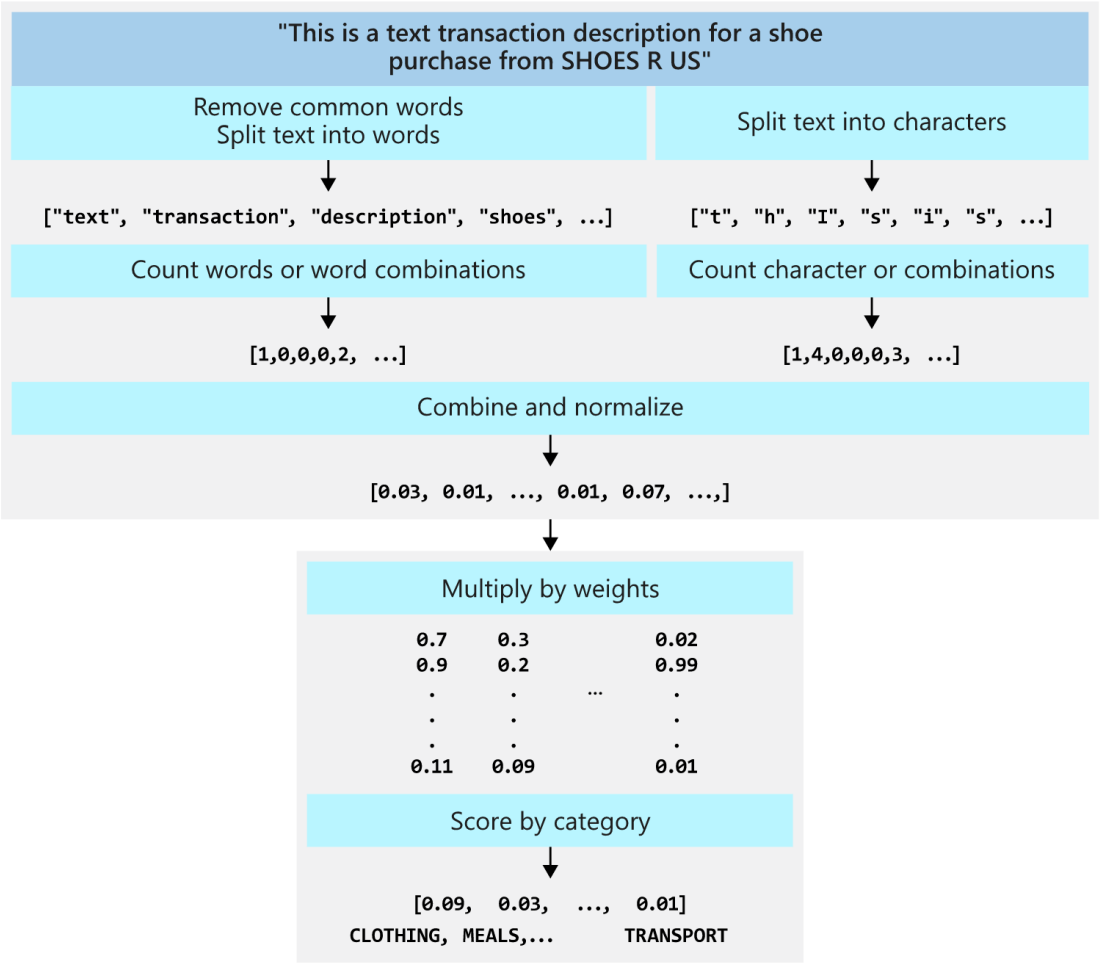
Şekil 27 de bu çalışmada kullanılan doküman yapısı için veri seti sınıfı bulunmaktadır. Veri seti uygulama katmanında kullanılması için iki kolonlu veri yapısı oluşturulmuştur. Model oluşturulması için ilgili senaryo için algoritma kurulması gerekmektedir. Bu projede sınıflandırma algoritması seçimi yapılmıştır. Sınıflandırma yapılabilmesi için ML.NET context yapısı veri seti yüklendikten sonra sonuçları bilinen doküman tipleri ile eğitilmiştir.

Doküman sınıflandırma için metin halinde bulunan veri tipi LoadColumn etiketi altında bulunan veri tipleri için her bir veri doldurulmaktadır. LoadColumn etiketi ile tanımlanan sınıflarda kelimeler doküman içerikleri indexlenmektedir. İndexlenen metinler içinde öğrenme ve model oluşturma işlemleri hızlı bir şekilde yapılmasına olanak sağlamaktadır. ML.NET kütüphanesi bu indexleme sayesinde popüler yapay zekâ eğitim kütüphanelerine daha hızlı bir şekilde eğitim yapılmasına olanak sağlamaktadır.

Eğitim sonucunda üretilen model zip dosyası halinde şifrelenmiş bir şekilde bulunmaktadır. Model dosyası bir kere üretildikten sonra uygulamadan bu üretilen modeli kullanarak sonuç elde edilmesi mümkündür. Model uygulama katmanına yüklendikten sonra testleri yüksek performansla gerçekleştirmek mümkündür.

3.5.2. ML.NET Eğitim Algoritmaları

ML.NET kütüphanesi bünyesinde sınıflandırma, anormallik algılama, kümeleme, tahmin etme, regresyon, öneri, sınıflandırma algoritmalarını barındırmaktadır. Model oluşturma aşamasında bu algoritmalar kullanılabilir.



Şekil 28 ML.NET Metin sınıflandırma örneği (Microsoft, 2020)

Şekil 28 de ML.NET sınıflandırma örneği gösterilmiştir. Şekil 28 de bir metin içindeki kelimeler birinci aşamada önce parçalanır. Her bir metinin ağırlıkları bulunur. Metinlerdeki karakterlerin sayıları ile veya ilişkileri hesaplanır. Çoklu olarak ağırlıkları alınır ve sınıflandırılması yapılır. Bu tez çalışmasında kütüphane de bulunan sıralama algoritması sayesinde doküman içinde bulunan metinler bu şekilde ağırlıkları ile sınıflandırılmıştır.

3.6. Sınıflandırma

Sınıflandırma, bir nesnenin sahip olduğu özelliklerine göre hangi gruba ait olduğunu belirlemektir. Makine öğrenimi algoritmaları ile önceden belirlenen veriler ile nesnelerin sınıfları öğretilerek model oluşturulur. Bu modele göre yeni eklenecek olan nesnelerin sınıfları tahmin edilir. Bu tahminlerin sağlıklı olması, sınıflandırmanın iyi analiz edilmesine, verilerin doğruluğuna, verilerin çok olmasına ve algoritmanın veri ile uyumuna bağlıdır. Veriler ne kadar birbirinden keskin ayrışıyorsa, veriler düzgün ise ve algoritma uygun seçilmiş ise sınıflandırma başarısı da yüksek olur. Sınıflandırma algoritmalarının en bilinenleri;

Karar ağaçları: Sınıflama, özellik ve hedefe göre karar düğümleri (decision nodes) ve yaprak düğümlerinden (leaf nodes) oluşan ağaç yapısı formunda bir model oluşturan bir sınıflandırma yöntemidir. Karar ağacı algoritması, veri setini küçük parçalara bölerek geliştirilir. Bir karar düğümü bir veya birden fazla dallanma içerebilir. İlk düğüme kök düğüm denir. Bir karar ağacı hem kategorik hem de sayısal verilerden oluşabilir.

Lineer diskriminant analizi: 1936 yılında R. A. Fischer tarafından geliştirilen bir sınıflama metodudur. Basit olmasına rağmen kompleks problemlerde iyi sonuçlar üreten bir modeldir. Lineer diskriminant analizi, iyi sınıf (hedefler) arasında en iyi şekilde ayıran değişkenleri lineer bir kombinasyonunu aramaya dayanır.

Lojistik regresyon: Lojistik regresyon, bağımlı değişkenin kategorik bir değişken olduğu regresyon problemi gibidir. Doğrusal sınıflandırma problemlerinde yaygın bir biçimde kullanılır. Lojistik regresyon, bir sonucu belirleyen bir veya daha fazla bağımsız değişken bulunan bir veri kümesini analiz etmek için kullanılan istatistiksel bir yöntemdir.

KNN (k-en yakın komşu): K-en yakın komşuluk (KNN) algoritması, uygulaması kolay gözetimli öğrenme algoritmalarındandır. Hem sınıflandırma hem de regresyon problemlerinin çözümünde kullanılıyor olmakla birlikte, endüstride çoğunlukla sınıflandırma problemlerinin çözümünde kullanılmaktadır.

Destek Vektör Makineleri: Genellikle sınıflandırma problemlerinde kullanılan gözetimli öğrenme yöntemlerinden biridir. Bir düzlem üzerine yerleştirilmiş noktaları ayırmak için bir doğru çizer. Bu doğrunun, iki sınıfının noktaları için de maksimum uzaklıkta olmasını amaçlar. Karmaşık ama küçük ve orta ölçekteki veri setleri için uygundur.

Bu çalışmada ML.NET sınıflandırma kütüphanelerinden Stochastic Dual Coordinate Ascent metodu ile sınıflandırma yapılmaktadır. Stochastic Dual Coordinate Ascent metodu makine öğrenimi için ağırlıkları bulmak ve hatayı en aza indirmek için Calculus türevini kullanır (McCaffrey, 2020). Bu sınıflandırma algoritması ile dokümanların içerikleri türlerine göre sınıflandırılmıştır. Metinlerin içerdiği ağırlıklı kelimeler ile türlerin eşleştirilmesi ile makine öğrenimi gerçekleşmiştir. Sınıflandırma sonucunda taranan bir dokümanın hangi doküman tipi olduğu sisteme öğretilmiştir. ML.NET ile yapılan sınıflandırma sonucunda model üretilmiştir.

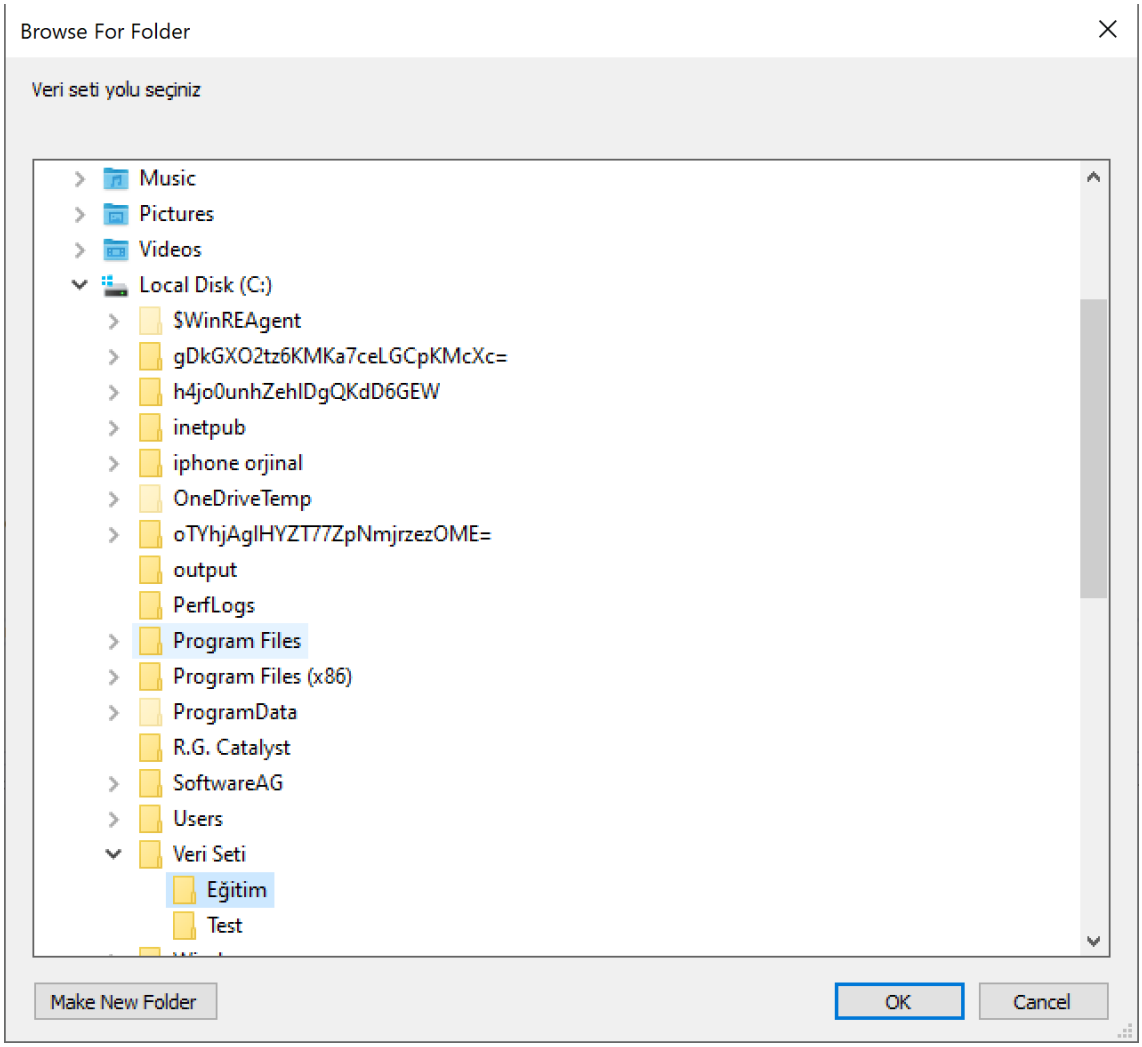
```
1 reference | Ali Eser, 35 days ago | 1 author, 1 change
public static void BuildAndTrainModel(string DataSetLocation, string ModelPath, MyTrainerStrategy selectedStrategy)
{
    // Create MLContext to be shared across the model creation workflow objects
    // Set a random seed for repeatable/deterministic results across multiple trainings.
    var mlContext = new MLContext(seed: 1);

    // STEP 1: Common data loading configuration
    var trainingDataView = mlContext.Data.LoadFromTextFile<DocumentIssue>(DataSetLocation, hasHeader: true, separatorChar: '\t', allowSparse: false);

    // STEP 2: Common data process configuration with pipeline data transformations
    var dataProcessPipeline = mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: DefaultColumnNames.Label, inputColumnName: nameof(DocumentIssue.DocumentType))
                  .Append(mlContext.Transforms.Text.FeaturizeText(outputColumnName: "DocumentContentFeaturized", inputColumnName: nameof(DocumentIssue.DocumentContent)))
                  .Append(mlContext.Transforms.Concatenate(outputColumnName: DefaultColumnNames.Features, "DocumentContentFeaturized"))
                  .AppendCacheCheckpoint(mlContext);
    // Use in-memory cache for small/medium datasets to lower training time.
    // Do NOT use it (remove .AppendCacheCheckpoint()) when handling very large datasets.

    // (OPTIONAL) Peek data (such as 2 records) in training DataView after applying the ProcessPipeline's transformations into "Features"
    Common.ConsoleHelper.PeekDataViewInConsole(mlContext, trainingDataView, dataProcessPipeline, 2);

    // STEP 3: Create the selected training algorithm/trainer
    IEstimator<ITransformer> trainer = null;
    switch (selectedStrategy)
    {
        case MyTrainerStrategy.SdcaMultiClassTrainer:
            trainer = mlContext.MulticlassClassification.Trainers.StochasticDualCoordinateAscent(DefaultColumnNames.Label,
                                                                                                  DefaultColumnNames.Features);
            break;
```

Şekil 29 ML.NET aracılığı ile sınıflandırmanın yapılması, ekran görüntüsü bu çalışma için yazılan projeden alınmıştır

Şekil 29 de bu tez çalışması için hazırlan uygulamanın kodlarından bir görüntü paylaşılmıştır. BuildAndTrainModel metodunda veri setindeki eğitim verisi ile sınıflandırma yapılır ve model üretilir. Sınıflandırma için adımlar gösterilmektedir.

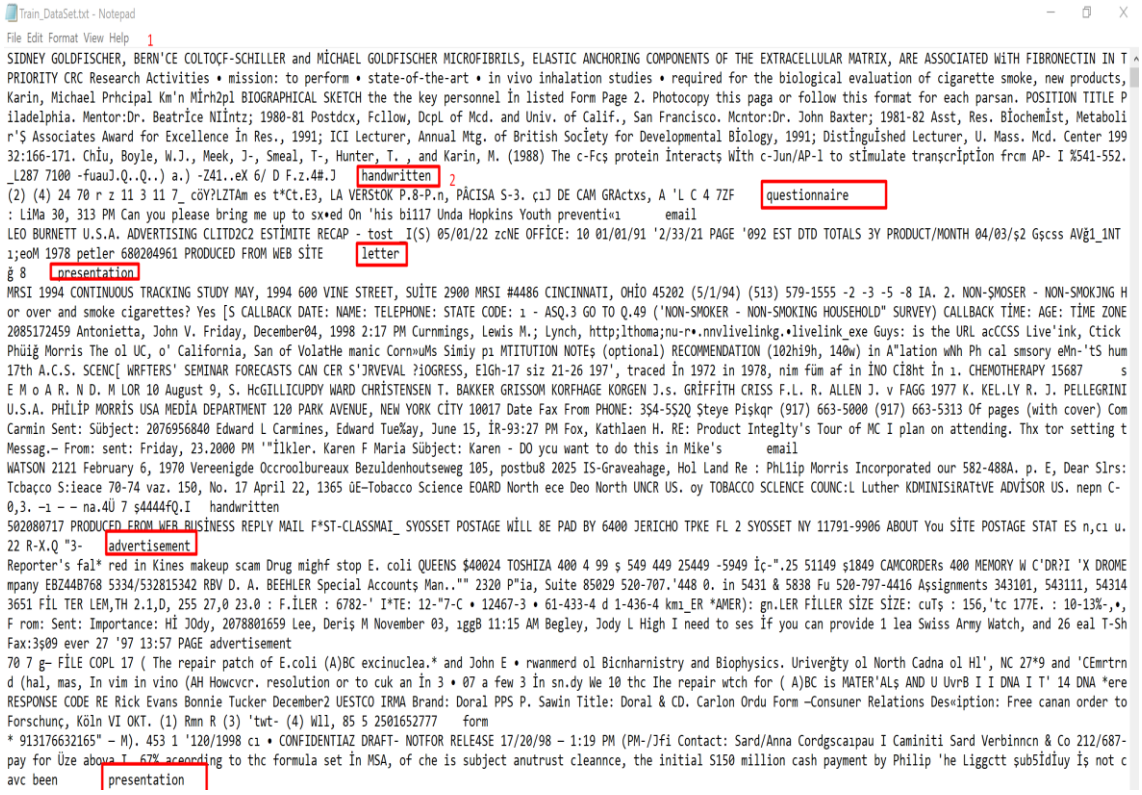
ML.Net ile sınıflandırma adımlarında kullanılacak olan MLContext oluşturulur. Sonra sırası ile;

i. Eğitim datası hangi ayraç ile ayrılacak, eğitim verisinin bilgisi gibi temel veriler set edilir ve view oluşturulur. Bu view aracılığı ile uygulamaya txt dosya türünde olan eğitim verisi yüklenir. Şekil 30 da gösterilen ekran ile eğitim verisi yükleme işlemi yapılmaktadır. Uygulama ilk açıldığında kullanıcıdan veri seti yolu seçmesini istemektedir.



Şekil 30 Tez uygulaması veri seti seçimi

ii. İkinci adım olarak veri dönüşümü ve eşleştirme yapılır. Şekil 31 de görüldüğü üzere txt dosyasında bulunan metin içeriği ve metin türü şeklinde olan iki kolon uygulama katmanında karşılığı ve veri tipi eşleştirmesi yapılmıştır. Verilerin yüklenmesi ve uygulama katmanında eşleştirilmesinden sonra makine öğreniminin gerçekleştirilmesi için ML.NET kütüphanesinde yer alan StochasticDualCoordinateAscent metodu seçilmiştir. Eğitim sonucunda model üretilmesi sağlanmıştır.



Şekil 31 Uygulama veri seti biçimi

3.6. Açık Kaynak Kod

Açık kaynak kod yazılım, uygulamanın kaynak kodlarının isteyen herkesin ulaşabildiği, yararlanabildiği gerektiğinde değişiklik yapabildiği yazılım demektir. Açık kaynak yazılımı bilgisayar programlama dilleri ortaya çıktığından itibaren var olan paylaşma olgusuna temelleri dayanmaktadır. Açık kaynak yazılım dünyanın her yer yerinden yazılım bilgisine sahip insanların bilgi paylaşımı ile ortak bir platformda iş birliği yapmasına olanak tanımaktadır. Açık kaynak yazılımı ile özgür yazılım ile temelde aynı fikre sahiptir ama eşit değildir. Özgür yazılım özetle lisans bedeli gibi kavramlar olmadan geliştirilen yazılımlardır. Özgür yazılımı ilk ortaya atan kişi Richard Stallman'dır (Perens, 1999).

Açık kayna kodların lisanslama için açık kaynaklı yazılımı tanıtmak ve terimin kullanımını normalleştirmek amacıyla 1998 yılında kurulan Açık Kaynak Girişimi Kurumu, 80'den fazla açık kaynak lisansı onaylamıştır (Kucukgultekin, 2020). Örnek lisanslama sistemleri aşağıdaki gibidir;

- BSD (Berkeley Software Distribution)
- MIT
- Apache 2
- Affero GPL (AGPL)
- GPL
- Lesser GPL (LGPL)
- Mozilla Public License (MPL)
- Eclipse Public License (EPL)
- Common Development and Distribution License (CDDL)

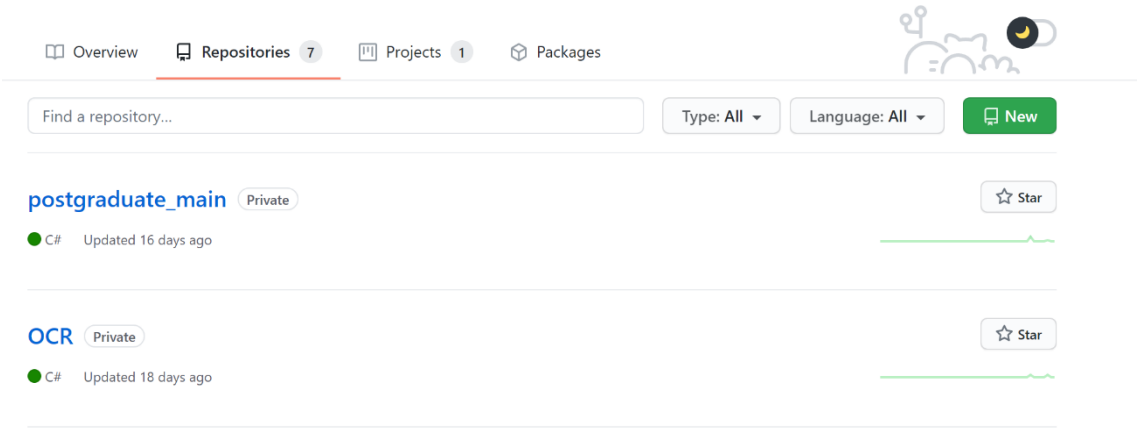
Bu sistemler aracılığı ile izin verilen sistemler, yazılımı olduğu gibi kabul eder ve sonucunda hiçbir garanti vermez. Bu lisanslama sistemi projeye katkı sunanlar ve geliştirme yapanlar için ortak bir yapıdır. Bu lisanslama sistemi açık kaynak kod paylaşımının gelişmesi ve değişmesi ile yıllar içinde tekrar yapılandırılabilir. Bu çalışmada kodlar açık kaynak kod prensipleri çerçevesinde geliştirilmiştir. Bu sayede ileride bu projeye destek vermek isteyenler sayesinde proje daha geliştirilebilir olacaktır. Erişim ve depolama kolaylığı sağlanmıştır. Versiyon sistemi sayesinde önceki versiyonlardaki değişiklikler kontrol altında tutulmuştur.

Açık kaynak kodlu yazılımların depolanması, yedeklenmesi, erişim kolaylığı gibi sebeplerden versiyon kontrol sistemleri kullanılabilir. Aşağıda açık kaynak kod sistemlerine hizmet veren sistemlerinden popüler olanları sıralanmıştır;

- BitBucket
- GitHub
- GitLab

Açık kaynak kod sistemlerine hizmet veren bu sistemler sayesinde projeler uzak sunucuda depolanmaktadır. Projelere erişim çoklu ve tekli olarak çalışma imkânı sunmaktadır. Ekip halinde bir proje üzerinde aynı anda çalışılabilir. Proje üzerinde çalışan her bir geliştirici kendine özel branch oluşturup çalışmasını tamamlayıp main branch proje deposuna kodlarını merge edebilir. Bu sayede projedeki çalışmaların hepsi kontrol altında tutulmuş olur.

Projede OCR için ve makine öğrenimi projesi için iki ayrı depo (resository) oluşturulmuştur. OCR projesinde taranmış dokümanların resim halinden metin elde edilmesini sağlayan projedir. Ana projede ise uygulamanın UX ekranı, makine öğrenimi sağlayan projeleri bulunmaktadır.

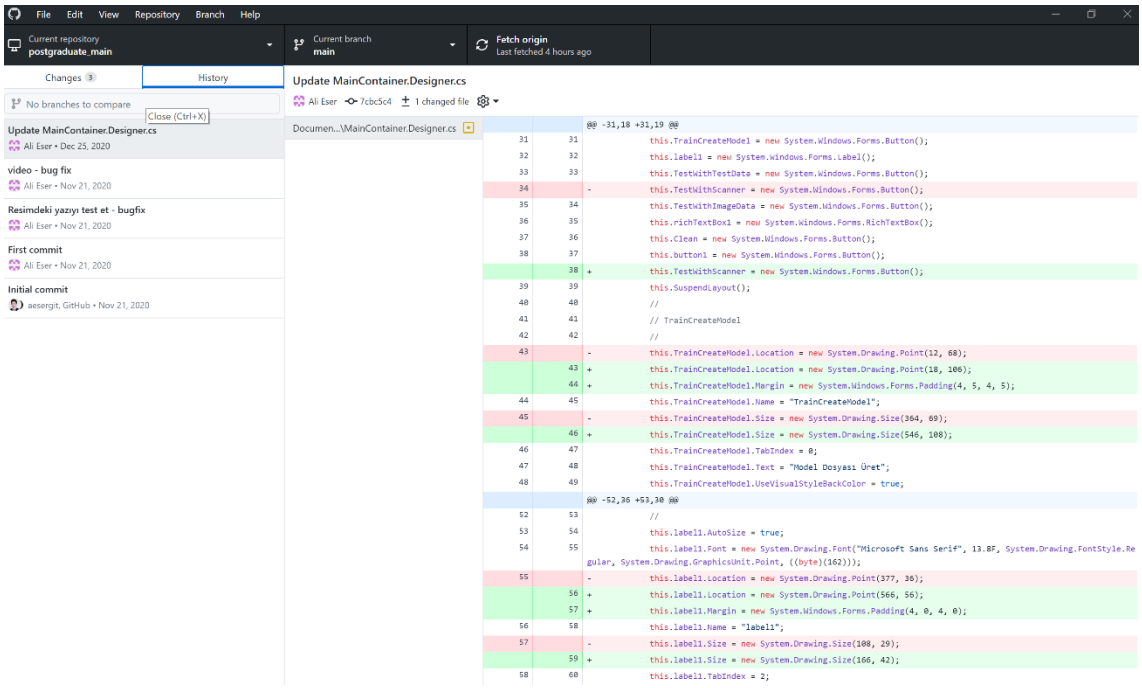


Şekil 32 Projenin açık kaynak kod geliştirmesi ekran görüntüsü

Şekil 32 de tez çalışması için oluşturulmuş açık kaynak kod sistemi görülmektedir.

3.6.1. Git

Git versiyon sistemi bir projenin aynı anda birçok kişi ile geliştirilmesine olanak sağlayan versiyon kontrol sistemidir. Git kullanılmadan önce yazılım geliştiriciler kodlarını depolama sorunun yanında ortak geliştirme kodlarını merge etme sorunları yaşamışlardır. Git versiyon sistemi sayesinde yazılan kodların geçmiş versiyonları bulunmaktadır. Bu sayede bir sorun olduğunda o kodu kim yazdı, kim değişiklik yaptı, ne gibi değişiklik yaptı görülebilir. Git komutları ile kodlar kendi ortamımıza çekme, merge etme, yaptığımız değişiklikleri geri alma, son değişiklikleri alma gibi işlemler yapılabilir.



Şekil 33 Proje git kullanımı kod geçmişi fark gösterimi

Şekil 33 de bu tez çalışmasında bulunan kod geçmişinin ve değişikliklerin örnek görüntüsü bulunmaktadır. Git sayesinde kod geçmişinin ve farklarının görüntülenmesi ile olabilecek hatalarında önüne geçilmesi sağlanabilmektedir. Ana projede değişiklik yapmak isteyen geliştiricilerin öncelikle kendilerine ana proje referans alarak branch oluşturması gerekmektedir. Branch oluşturulmasından sonra geliştirici kendi branch sisteminde değişikliklerini ana branch sistemine geçirebilir. Bu geçiş için istenirse onay sistemi kullanılabilir. Onay sistemi koda katkı sağlamak isteyenlerin kodların asıl sahiplerine göndermiş olduğu kod değişikliklerin yürütüldüğü sistemdir. Kod onay sisteminde onaya gönderilecek kişiler veya gruplar seçilebilir. Örneğin bu proje için OCR için görüntü işleme ekibi adında bir grup oluşturup onayların görüntü işleme ekibine

düşmesi sağlanabilir. Veya kodlar onaya gönderilmeden önce kod gözden geçirme sistemlerinde yeterli puan alması beklenebilir. Aşağıda kod gözden geçirilmesi için otomasyonlar bulunmaktadır.

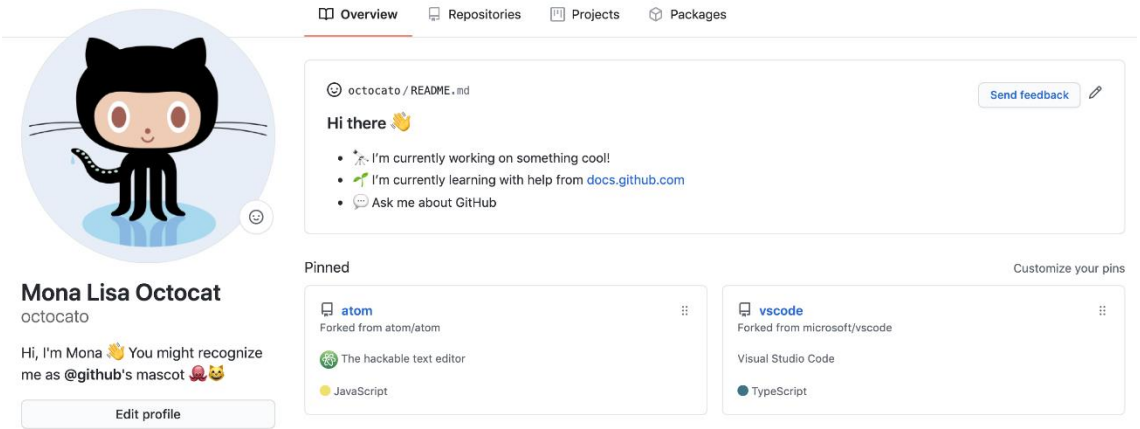
• Codacy
• CodeClimate
• QuantifiedCode
• Landscape
• Sonar Cloud

Bu sistemler sayesinde onaya gönderilen kodlar önce bu sistemlerde analiz edilmektedir. Gereksiz kullanılan değişkenler, yanlış kullanılan sistem metotları gibi bulunan analizler paylaşılmaktadır.

3.6.2. Github

Github, Git sürüm kontrol sistemi kullanan yazılım geliştirme projeleri için web tabanlı bir depolama sistemidir (Wikipedia, 2021). 2008 de kurulan Github bir Microsoft kuruluşudur. Ücretsiz bir şekilde kod depolama ve versiyon yapma imkânı sunmaktadır. Projenin kodları Github üzerinde depolanmıştır. Proje için iki adet depo oluşturulmuştur. Görüntü işleme için OCR deposu ve ana proje için main deposu oluşturulmuştur.

Github üzerinde kodlara erişim kolaylığı sağlanmaktadır. Versiyon kontrol sistemi sayesinde adım adım proje geliştirilmesi sunucuya gönderilmiştir. Projede başka kullanıcılarında geliştirme yapabilecek ortam sağlanmıştır.



Şekil 34 Github örnek ana sayfa

Şekil 34 de Github örnek ekran görüntüsü görülmektedir. Kullanıcıların bu sayfadaki bilgileri bilirliği etkinlikleri ve tecrübesinin görülmesi bakımından önemlidir.

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Bu bölümde tez boyunca yapılmış olan çalışmalar ve çalışmaların sonucuna ait bilgiler paylaşılmıştır. Bu tez kapsamında;

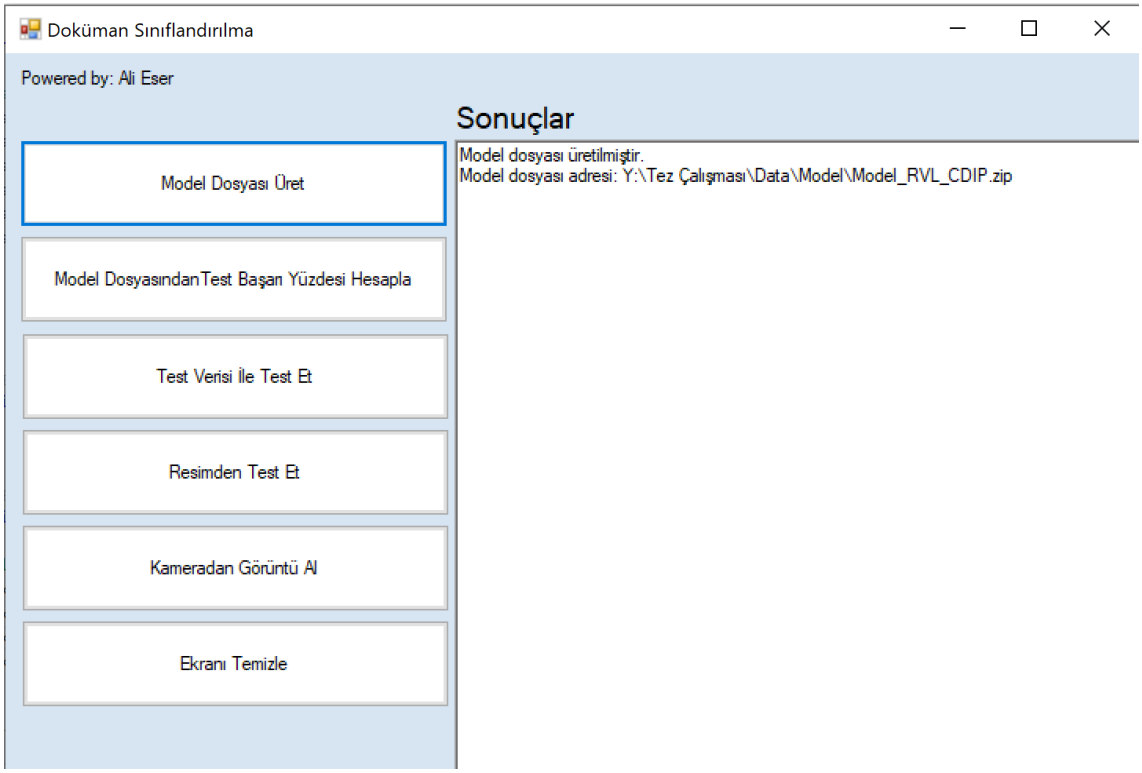
- Taranmış dokümanların bulunduğu veri kaynağı aracılığı ile veri setinin oluşturulması
- Oluşturulan veri setinin OCR işlemi ile metin haline getirilmesi
- Eğitim verisi, test verisi halinde parçalara ayrıldı
- ML.NET makine öğrenimi kütüphanesi ile model oluşturulması
- Test verisi ile modelin test edilmesi
- ML.NET makine öğrenimi kütüphanesi ile model oluşturulması
- Görsel kullanım için uygulamanın yazılması

Adımları uygulanmıştır.

4.1. Çalışma Ortamı ve Uygulamanın Geliştirilmesi

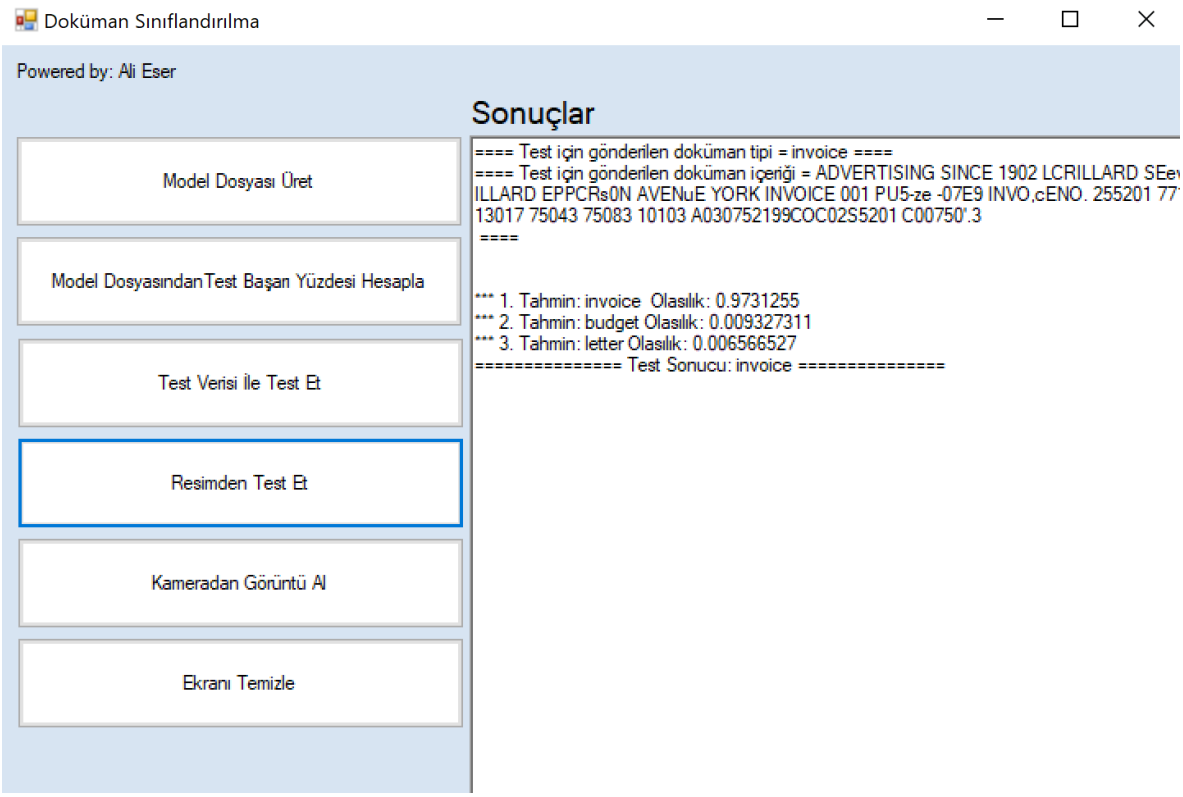
Çalışmada 320.000 adet veri seti kullanılmıştır. Bu veri setinden elde edilen eğitim verisi ile ML.NET kütüphanesi aracılığı ile doküman sınıflandırma yapılmıştır. Elde edilen sonuçlar aşağıdaki gibidir. Eğitimden sonra üretilen model ile 1000 doküman test edilmiştir. Test aşaması için ayrılan 1000 doküman tek tek OCR işlemi ile metin haline getirilip modele göre tahmin üretilmesi sağlanmıştır. 1000 dosyadan tahmin başarısı %92 olarak sonuçlanmıştır.

Çalışmayı son kullanıcının kullanabilmesi için ara yüz hazırlanmıştır. Hazırlanan ara yüz C# ile yazılmıştır. Bu uygulama ile kullanıcı eğitim verisinden model dosyası üretebilir, model dosyasından test yapılabilir, tekli olarak test yapılabilir. Ayrıca direkt olarak taranmış doküman üzerinden test yapılabilir. Taranmış dosyanın adresi verildiğinde uygulama önce OCR kütüphanesi yardımı ile metin haline dönüştürüp üretilen metinleri model dosyası ile karar vermektedir.



Şekil 35 Doküman sınıflandırma için hazırlanan görsel ara yüz

Şekil 35 de uygulamanın görsel ara yüzü görülmektedir. Bu ara yüz sayesinde kullanıcının dokümanları sınıflandırması ve test sonuçlarını kolay bir şekilde görmesi sağlanmıştır. Model dosyası üretme aksiyonu ile kullanıcı taranan resim halindeki verilerin tek bir dosya haline getirilmiş eğitim verisi ile model dosyası üretilmektedir. Bu model dosyasını üretme aşamasında eğitim verisi metin içeriği ve metin tipi olmak üzere iki kolon içermektedir. Test verisi ile test etme aksiyonu ile kullanıcı üretilmiş model üzerinden yeni gelen doküman tipini tahmin etmektedir.



Şekil 36 Tek doküman türü testi

Şekil 36 da görüldüğü gibi tek doküman türü testi ve üç adet tahmin bulunmaktadır. Bu tahminlerin yüzdeleri ile birlikte görülmesi mümkündür. Bu şekilde "invoice" doküman türü test edilmiştir. Şekil 7'deki gibi test için gönderilen doküman ve OCR ile okunmuş içeriği görülmektedir. El yazısı olmayan doküman türlerinde başarı oranı daha yüksek olduğu belirlenmiştir. Test için gönderilen "invoice" doküman tipi için 1. sıradaki tahmin 0.9731255 olasılıkla "invoice" 2. sıradaki tahmin 0.009327311 olasılıkla "budget" ve 3. sıradaki tahmin 0.006566527 olasılıkla "letter" sonucu çıkmıştır. Test sonucu ile 0.97 ile invoice olarak doğru tahmin yapılmıştır.

Resimden test et aksiyonu ile bilgisayarda bulunan taranmış resim halindeki doküman önce metin haline getirilip daha sonra üretilmiş olan model dosyası üzerinden test ediliyor. Kameradan görüntü al aksiyonu ile bilgisayar kamerasından alınan görüntünün OCR işlemi ile metin haline getirilip, model dosyası aracılığı ile türünün belirlenmesi sağlanmıştır. Araştırma sonuçlarına göre yüzde doksan üzeri bir başarı ile tahmin sağlanmıştır. Bu araştırma sayesinde şirketler taranan dokümanların türlerini belirleyip ilgili klasörlere otomasyon sayesinde atılmasını ve düzenli bir şekilde müşteri dokümanlarının kayıt edilmesi

sağlanacaktır. Örneğin müşteri kimlikleri, kimlikler klasöründe, ehliyetleri ehliyetler klasöründe olacak. Bu sayede eksik bir dokümanı var ise kolayca tespiti sağlanacak, personelin taranan dokümanı ilgili klasöre atmak için zaman kaybından kurtarması amaçlanmaktadır.

4.2. Sınıflandırma Sonuçları

Çalışmanın veri setinde bulunan 320.000 adet veri kullanılmıştır. Bu verilerle makine öğrenimi sağlanmıştır. Makine öğrenimi eğitimi sonucunda model üretilmiştir. Üretilen model üzerinden test yapılmıştır. Test veri setinden ilk 50 kayıt tablo 2'de gösterilmiştir. Dosya adı, tahmin edilen doküman türü, benzerlik oranı, hedef doküman kaynağı ve sonuç paylaşılmıştır.

Tablo 2 Test veri setinden ilk 50 kayıt test sonuçları

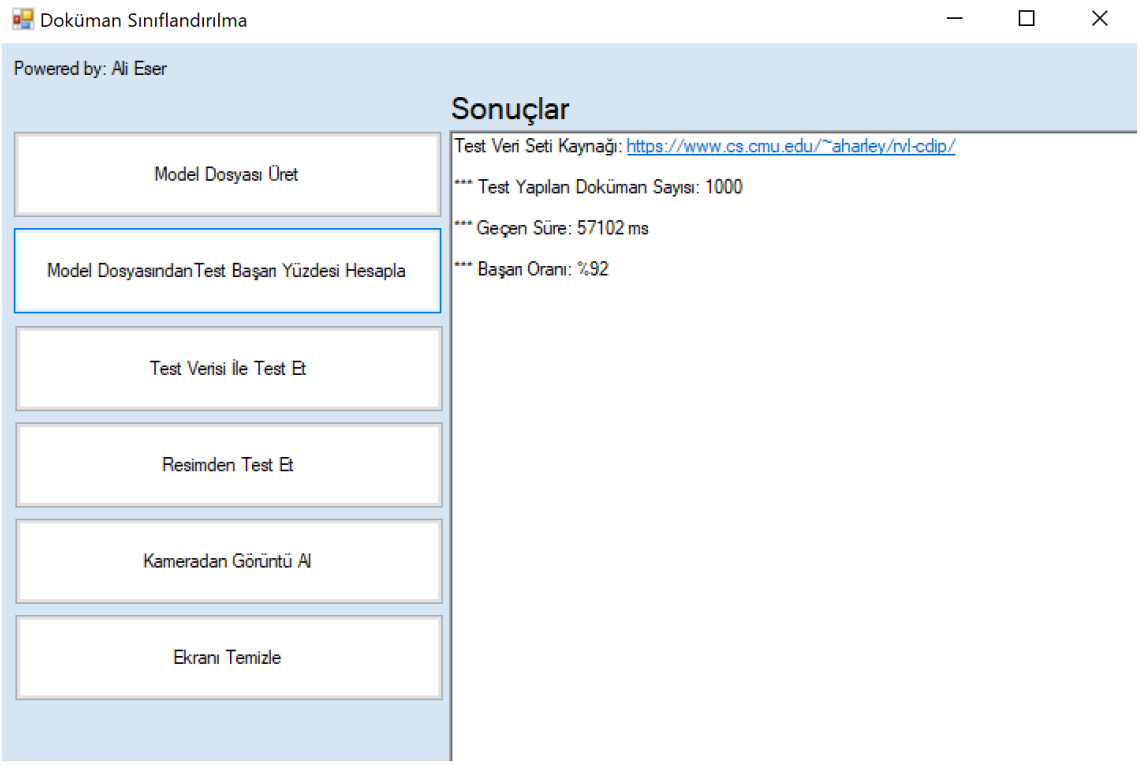
Dosya Adı	Tahmin	Oran	Hedef	Sonuç
87103403.tif	letter	0.6257103	letter	Doğru
50437856-7857.tif	resume	0.9927745	resume	Doğru
0000457436.tif	presentation	0.223622	presentation	Doğru
10121367.tif	scientific publication	0.4978631	scientific publication	Doğru
93503327.tif	handwritten	0.3098549	budget	Yanlış
502596897.tif	advertisement	0.7848506	advertisement	Doğru
10240676.tif	memo	0.1919242	letter	Doğru
2021511482.tif	memo	0.7744765	memo	Doğru
2076954791_4796.tif	scientific report	0.4508843	memo	Doğru
tcal0249104.tif	budget	0.6951491	budget	Doğru

2505944247.tif	letter	0.2438149	email	Doğru
2085270716.tif	email	0.9457675	email	Doğru
50521129-1130.tif	resume	0.9967162	resume	Doğru
1001761126.tif	handwritten	0.2949046	letter	Yanlış
2071863639a.tif	email	0.9975682	email	Doğru
2078784438.tif	email	0.9988124	email	Doğru
2071239110.tif	resume	0.9091407	resume	Doğru
98353876_3889.tif	presentation	0.2998922	presentation	Doğru
10165056_10165059.tif	scientific publication	0.7090687	scientific publication	Doğru
50529098-9099.tif	resume	0.8316645	resume	Doğru
527795574+-5574.tif	email	0.9919729	email	Doğru
2045745523_2045745524.tif	presentation	0.5815169	presentation	Doğru
40002791-2791.tif	memo	0.2582545	scientific report	Doğru
50295375-5378.tif	scientific report	0.4857952	scientific report	Doğru
2081500134.tif	email	0.9991192	email	Doğru
2501091720.tif	scientific report	0.7764832	scientific report	Doğru

2030559674.tif	file folder	0.6200318	file folder	Doğru
99214437_4439.tif	scientific publication	0.3086073	scientific report	Yanlış
50382789-2790.tif	resume	0.9859302	resume	Doğru
86328049_8050.tif	form	0.8707781	form	Doğru
2080180993.tif	email	0.9797198	email	Doğru
50408559-8561.tif	resume	0.997242	resume	Doğru
2084194384b.tif	memo	0.6931952	email	Doğru
00868598_00868646.tif	scientific report	0.9441859	scientific report	Doğru
505866581_505866584.tif	letter	0.8032759	letter	Doğru
91615674.tif	letter	0.7468702	letter	Doğru
tnwl0038839.tif	budget	0.665028	budget	Doğru
530002537+-2539.tif	letter	0.4570217	letter	Doğru
50447511-7517.tif	scientific publication	0.9227195	scientific publication	Doğru
1000349384.tif	form	0.5599716	form	Doğru
2085763670d_3671.tif	email	0.9276171	email	Doğru
50501852-1853.tif	resume	0.9916948	resume	Doğru
2069735582.tif	specification	0.9572318	specification	Doğru

1000089770_1000089773.tif	scientific publication	0.7279805	scientific publication	Doğru
96018094.tif	presentation	0.309122	presentation	Doğru
2078861322.tif	presentation	0.5598723	presentation	Doğru
88113649_3650.tif	presentation	0.4280991	presentation	Doğru
50737116-7119.tif	resume	0.9596622	resume	Doğru
10319873_10319886.tif	scientific publication	0.8868918	scientific publication	Doğru

Bu verilerin yorumlanmasında bilimsel rapor dokümanları ve mektup yazı türlerinde oranlar değişmektedir. Bunun sebebi değişken kelimelerin çok olması ve el yazısının bulunmasından dolayıdır. Özgeçmiş doküman türlerinde başarı oranı yüksektir bunun sebebi dokümanlarda benzer kelimeler geçtiğinden eşleştirilme oranı daha yüksek çıkmıştır.



Şekil 37 1000 adet test verisinden ortalama başarı oranı

Şekil 18'de 1000 adet test dokümanı üzerinden test edilmiştir. Başarı oranı yüzde 92 olarak sonuçlandırılmıştır. 57102ms sürede test tamamlanmıştır. Bu süre uygulama başlatılıp model dosyasının uygulamaya yüklenmesinden sonra dokümanların tek tek teste tabi tutulmasında geçen süredir. Model dosyası uygulamaya yüklendikten sonra bir dokümanın test edilmesi 1 saniyeden daha kısa sürmektedir.

Özellikle kurumsal şirketlerde uygulamanın tahmin süresi hızlı olması gerekmektedir. Bu sebeple uygulamanın performansı konusunda başarı elde edilmesi sağlanmıştır. Performans için OCR kütüphanelerinde araştırma yapılıp seçim yapılmıştır. Aynı sebeple performans için makine öğrenimi kütüphanelerinde ML.NET seçilmiştir. Her iki kütüphanede alanlarında popüler olan diğer kütüphanelerden daha performanslı olduğu araştırılmıştır.

5. SONUÇLAR VE ÖNERİLER

5.1 Sonuçlar

Özellikle kurumsal şirketlerde farklı alanlardaki sorunlara çözüm bulmak, süreçleri hızlandırmak, kaliteyi artırmak adına sınıflandırma işlemini yapmak ve yapım aşamasındaki sınıflandırmaların başarısını artırmak için birçok yeni yöntem, yeni algoritmalar geliştirilmektedir. Bu yüksek lisans tezinde dokümanların sınıflandırılması konusunda dokümanların sınıflandırılması konusu dokümanların metin haline getirilip kendi alanında başarılı kütüphaneler ile performans ve başarı odak noktası alınarak çalışılmıştır.

Çalışma sonunda taranmış dokümanların sınıflandırılabilmesi için Windows platformu için C# kodu ile uygulama yazılmıştır. Bu uygulama ile kullanıcı tarayıcıdan alınan dokümanın tipini bir saniyeden daha kısa bir süre de tespit edebilecektir. Şirketlerin hali hazırda olan süreçlerine uyum içinde uygulama kullanılabilir. Şirketlerin kullandığı alt yapı uygulamalarında müşteri dokümanlarını işleyen, kayıt eden ve yöneten sistemler bulunmaktadır. Bu çalışmada amaç şirketlerde bulunan doküman yönetim sistemleri süreçlerine uyum sağlayıp katkı sağlamasıdır.

Yapılan uygulama test sonuçlarında dokümanların başarılı bir şekilde sınıflandırıldığı görülmüştür. Başarı oranı makine yazısı el yazısına oranla daha başarılı olmaktadır. El yazısı ne kadar net olursa başarı o oranda artmaktadır. Birbirine yakın metinlerin olduğu doküman tiplerinde de başarı oranın arttığı görülmüştür. Örneğin kimlik belgesi, ehliyet belgesi gibi doküman türlerinde başarı oranı çok yüksektir. Çünkü her taranmış dokümanda adı, soyadı gibi alanlar olduğu için yazılan otomasyonun eşleştirmesi daha kolay ve tahmin oranı daha yüksek olmuştur.

Tez çalışmasına sonucuna göre taranmış şirket dokümanları Microsoft OCR kütüphanesi ile metin haline getirilip, metin haline getirilen dokümanlar ML.NET kütüphanesi ile doküman sınıflandırılması yapılabileceği değerlendirilebilir.

5.2 Öneriler

Dokümanların sınıflandırılması en önemli kısım sistemin öğrenim adımı ve model oluşturulmasıdır. Eğitim aşamasında ne kadar çok veri olursa başarı oranın o kadar arttığı, sistemin daha iyi öğrendiği görülmüştür. Eğitim verisinin düzenli olması, taranan dokümanların net olması ve makine yazısı olması başarı oranını arttırmıştır. Bu veriler ışığında doğru veri setinin kullanılması gerekmektedir.

Doğru veri seti adımından sonra performansı artırmak için doğru kütüphaneler ile doğru sınıflandırma algoritmaları seçilmelidir. Bu tez çalışması kapsamında kullanılan makine öğrenimi kütüphanesi olan ML.NET kütüphanesi, Scikit-learn ve H2O kütüphaneleri ile karşılaştırılması araştırılmıştır. ML.NET performans ve kalite bakımından daha başarılı olduğu sonucuna varılmıştır.

Şirketler uygulamalarının stabil olması, kaliteli olması ve performanslı olmasını bekler. Bu sebeple seçilen kütüphaneler bu minvalde belirlenmiştir. Şirket dokümanlarının sınıflandırma başarısını artırmak için birbirine benzer alanlar içermesi başarı oranını artırır. Çünkü öğrenilen model üzerinden kelime eşleşmesi ne kadar fazla olursa o kadar tahmin oranı ve başarı oranı artmaktadır. Dokümanların tarandığı tarayıcı yüksek kalitede olması gerekmektedir. Çünkü yüksek kaliteli tarayıcıdan üretilen dokümanları OCR uygulamaları daha net kelime ayrımı yapabilmektedir. Şirketlerin yönetim sistemlerinde bulunan doküman yönetim sistemlerinde tüm işlemlerin otomatik yapılması yanında elle işlem yapabilecek şekilde de ayarlanması gerekmektedir. Tarayıcı sorunlarının önüne bu şekilde geçilebilir. Eğitim verisinde aynı tür dokümanlarda ne kadar çok veri olursa başarı oranında o kadar artacaktır.

6. KAYNAKLAR

A, S. (2020). Creating Console Application In C# - https://www.c-sharpcorner.com/article/creating-console-application-in-c-sharp/.

Adam W. Harley , A. U. v. K. G. D. (2020). The RVL-CDIP Dataset - https://www.cs.cmu.edu/~aharley/rvl-cdip/.

Ahmed, Z., Amizadeh, S., Bilenko, M., Carr, R., Chin, W.-S., Dekel, Y., . . . Finley, T. (2019). *Machine Learning at Microsoft with ML. NET.* Paper presented at the Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.

Amasyalı, M. F., & Diri, B. (2006). *Automatic Turkish text categorization in terms of author, genre and gender.* Paper presented at the International Conference on Application of Natural Language to Information Systems.

Byun, Y., & Lee, Y. (2000). *Form classification using DP matching.* Paper presented at the Proceedings of the 2000 ACM symposium on Applied computing-Volume 1.

Csurka, G., Dance, C., Fan, L., Willamowski, J., & Bray, C. (2004). *Visual categorization with bags of keypoints.* Paper presented at the Workshop on statistical learning in computer vision, ECCV.

Çobanoğlu, Ö. E. (2015). *Comparison of document classification approaches for Turkish texts.* Izmir Institute of Technology.

Doğan, S., & Diri, B. (2010). Türkçe dokümanlar için N-gram tabanlı yeni bir sınıflandırma (Ng-ind): yazar, tür ve cinsiyet. *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 3*(1), 11-19.

Ergezer, H. (2003). Face Recognition: Eigenfaces. *Neural Networks, Gabor Wavelet Transform Methods, MS Thesis, Başkent University, Turkey*.

Fidan, Ü. (2013). *Destek Vektör Makineleri İle Doküman Sınıflandırma.*

Github, W. (2020). GitHub.

Harley, A. W., Ufkes, A., & Derpanis, K. G. (2015). *Evaluation of deep convolutional nets for document image classification and retrieval.* Paper presented at the 2015 13th International Conference on Document Analysis and Recognition (ICDAR).

Hu, J., Kashi, R., & Wilfong, G. (2000). Comparison and classification of documents based on layout similarity. *Information Retrieval, 2*(2-3), 227-243.

Josipovic, P. (2020). Optical Character Recognition (OCR) for Windows 10 - https://blogs.windows.com/windowsdeveloper/2016/02/08/optical-character-recognition-ocr-for-windows-10/.

Kang, L., Kumar, J., Ye, P., Li, Y., & Doermann, D. (2014). *Convolutional neural networks for document image classification.* Paper presented at the 2014 22nd International Conference on Pattern Recognition.

Knight, W. C. B. I., WA, US). (2017). United States Patent No. 20170277998.

Kucukgultekin, C. (2020). Açık Kaynak Yazılım- https://medium.com/@can.kucukgultekin/a%C3%A7%C4%B1k-kaynak-yaz%C4%B1l%C4%B1m-319bad28ca95.

Kumar, J., Ye, P., & Doermann, D. (2014). Structural similarity for document image classification and retrieval. *Pattern Recognition Letters, 43*, 119-126.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). *Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories.* Paper presented at the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06).

McCaffrey, J. D. (2020). Machine Learning Primals and Duals, and Ascent and Descent.

Mchenry, C. A. L., CO, US), Burt, Scott W. (Parker, CO, US). (2016). United States Patent No. 9378265.

Microsoft. (2020). ML.NET - https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet.

Öztemel, E. (2003). Yapay sinir ağlari. *PapatyaYayincilik, Istanbul*.

Perens, B. (1999). The open source definition. *Open sources: voices from the open source revolution, 1*, 171-188.

Qi, Y. P., NJ, US), Bai, Bing (Princeton Junction, NJ, US). (2014). United States Patent No. 8892488.

Sarı, M. (2018). *Derin öğrenme yöntemleri kullanılarak Türkçe doküman sınıflandırma.* TOBB ETÜ Fen Bilimleri Enstitüsü.

Taylor, S. L., Lipshutz, M., & Nilson, R. W. (1995). *Classification and functional decomposition of business documents.* Paper presented at the Proceedings of 3rd International Conference on Document Analysis and Recognition.

Torre, C. D. l. (2020). Announcing ML.NET 0.6 (Machine Learning .NET) - https://devblogs.microsoft.com/dotnet/announcing-ml-net-0-6-machine-learning-net/.

Wikipedia. (2021). Github - https://tr.wikipedia.org/wiki/GitHub.

Amasyalı, M. F., & Diri, B. (2006). *Automatic Turkish text categorization in terms of author, genre and gender.* Paper presented at the International Conference on Application of Natural Language to Information Systems.

Byun, Y., & Lee, Y. (2000). *Form classification using DP matching.* Paper presented at the Proceedings of the 2000 ACM symposium on Applied computing-Volume 1.

Csurka, G., Dance, C., Fan, L., Willamowski, J., & Bray, C. (2004). *Visual categorization with bags of keypoints.* Paper presented at the Workshop on statistical learning in computer vision, ECCV.

Çobanoğlu, Ö. E. (2015). *Comparison of document classification approaches for Turkish texts.* Izmir Institute of Technology.

Doğan, S., & Diri, B. (2010). Türkçe dokümanlar için N-gram tabanlı yeni bir sınıflandırma (Ng-ind): yazar, tür ve cinsiyet. *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 3*(1), 11-19.

Fidan, Ü. (2013). *Destek Vektör Makineleri İle Doküman Sınıflandırma.*

Harley, A. W., Ufkes, A., & Derpanis, K. G. (2015). *Evaluation of deep convolutional nets for document image classification and retrieval.* Paper presented at the 2015 13th International Conference on Document Analysis and Recognition (ICDAR).

Hu, J., Kashi, R., & Wilfong, G. (2000). Comparison and classification of documents based on layout similarity. *Information Retrieval, 2*(2-3), 227-243.

Kang, L., Kumar, J., Ye, P., Li, Y., & Doermann, D. (2014). *Convolutional neural networks for document image classification.* Paper presented at the 2014 22nd International Conference on Pattern Recognition.

Knight, W. C. B. I., WA, US). (2017). United States Patent No. 20170277998.

Kumar, J., Ye, P., & Doermann, D. (2014). Structural similarity for document image classification and retrieval. *Pattern Recognition Letters, 43*, 119-126.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). *Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories.* Paper presented at the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06).

Mchenry, C. A. L., CO, US), Burt, Scott W. (Parker, CO, US). (2016). United States Patent No. 9378265.

Qi, Y. P., NJ, US), Bai, Bing (Princeton Junction, NJ, US). (2014). United States Patent No. 8892488.

Sarı, M. (2018). *Derin öğrenme yöntemleri kullanılarak Türkçe doküman sınıflandırma.* TOBB ETÜ Fen Bilimleri Enstitüsü.

Taylor, S. L., Lipshutz, M., & Nilson, R. W. (1995). *Classification and functional decomposition of business documents.* Paper presented at the Proceedings of 3rd International Conference on Document Analysis and Recognition.

EKLER

EK-1 ML.NET Doküman Sınıflandırma Algoritmasının C# Kodu

```
1.   internal class DocumentIssue
2.       {
3.           [LoadColumn(0)]
4.           public string DocumentContent;
5.
6.           [LoadColumn(1)]
7.           public string DocumentType;
8.       }
9.
10. internal class DocumentIssuePrediction
11.      {
12.          [ColumnName("PredictedLabel")]
13.          public string DocumentType;
14.
15.          public float[] Score;
16.      }
17.
18. internal class DocumentIssueTransformed
19.      {
20.          public string DocumentType;
21.
22.          public string DocumentDescription;
23.      }
24.
25. public class FullPrediction
26.      {
27.          public string PredictedLabel;
28.          public float Score;
29.          public int OriginalSchemaIndex;
30.
31.          public FullPrediction(string predictedLabel, float score, int original
    SchemaIndex)
32.          {
33.              PredictedLabel = predictedLabel;
34.              Score = score;
35.              OriginalSchemaIndex = originalSchemaIndex;
36.          }
37.      }
38.
```

```
39. using DocumentClassification.DataStructures;
40. using Microsoft.ML;
41. using Microsoft.ML.Data;
42. using Octokit;
43. using System;
44. using System.Collections.Generic;
45. using System.IO;
46. using System.Linq;
47. using System.Threading.Tasks;
48. 
49. namespace DocumentClassification
50. {
51.     internal class Labeler
52.     {
53.         private readonly GitHubClient _client;
54.         private readonly string _repoOwner;
55.         private readonly string _repoName;
56.         private readonly string _modelPath;
57.         private readonly MLContext _mlContext;
58. 
59.         private readonly PredictionEngine<DocumentIssue, DocumentIssuePredicti
    on> _predEngine;
60.         private readonly ITransformer _trainedModel;
61. 
62.         private FullPrediction[] _fullPredictions;
63. 
64.         public Labeler(string modelPath, string repoOwner = "", string repoNam
    e = "", string accessToken = "")
65.         {
66.             _modelPath = modelPath;
67.             _repoOwner = repoOwner;
68.             _repoName = repoName;
69. 
70.             _mlContext = new MLContext(seed: 1);
71. 
72.             // Model yüklenmesi
73.             using (var stream = new FileStream(_modelPath, System.IO.FileMode.
    Open, FileAccess.Read, FileShare.Read))
74.             {
75.                 _trainedModel = _mlContext.Model.Load(stream);
76.             }
77. 
78. 
```

```
79.             _predEngine = _trainedModel.CreatePredictionEngine<DocumentIssue,
    DocumentIssuePrediction>(_mlContext);
80.
81.             if (accessToken != string.Empty)
82.             {
83.                 var productInformation = new ProductHeaderValue("MLDocumentCla
    ssificationLabeler");
84.                 _client = new GitHubClient(productInformation)
85.                 {
86.                     Credentials = new Credentials(accessToken)
87.                 };
88.             }
89.         }
90.
91.         public void TestPredictionForSingleIssue()
92.         {
93.             var lines = File.ReadAllLines(@"D:\Tez Çalışması\Data\TestData.txt
    ");
94.             var line = lines[Convert.ToInt32(new Random().Next(lines.Count() -
     1))];
95.
96.             DocumentIssue singleIssue = new DocumentIssue()
97.             {
98.                 DocumentContent = line.Split('\t').First()
99.             };
100.
101.                 var prediction = _predEngine.Predict(singleIssue);
102.
103.                 _fullPredictions = GetBestThreePredictions(prediction);
104.
105.                 Console.WriteLine("==== Doküman Tipi = {0} and Doküman İçer
    iği = {1} ====", singleIssue.DocumentType, singleIssue.DocumentContent);
106.
107.                 Console.WriteLine("1. Tahmin: " + _fullPredictions[0].Predi
    ctedLabel + " with score: " + _fullPredictions[0].Score);
108.                 Console.WriteLine("2. Tahmin: " + _fullPredictions[1].Predi
    ctedLabel + " with score: " + _fullPredictions[1].Score);
109.                 Console.WriteLine("3. Tahmin: " + _fullPredictions[2].Predi
    ctedLabel + " with score: " + _fullPredictions[2].Score);
110.
111.                 Console.WriteLine($"=============== Sonuç: {prediction.Docu
    mentType} ===============");
112.             }
113.
```

```
114.             private FullPrediction[] GetBestThreePredictions(DocumentIssueP
    rediction prediction)
115.             {
116.                 float[] scores = prediction.Score;
117.                 int size = scores.Length;
118.                 int index0, index1, index2 = 0;
119.
120.                 VBuffer<ReadOnlyMemory<char>> slotNames = default;
121.                 _predEngine.OutputSchema[nameof(DocumentIssuePrediction.Sco
    re)].GetSlotNames(ref slotNames);
122.
123.                 GetIndexesOfTopThreeScores(scores, size, out index0, out in
    dex1, out index2);
124.
125.                 _fullPredictions = new FullPrediction[]
126.                     {
127.                         new FullPrediction(slotNames.GetItemOrDefault(index
    0).ToString(),scores[index0],index0),
128.                         new FullPrediction(slotNames.GetItemOrDefault(index
    1).ToString(),scores[index1],index1),
129.                         new FullPrediction(slotNames.GetItemOrDefault(index
    2).ToString(),scores[index2],index2)
130.                     };
131.
132.                 return _fullPredictions;
133.             }
134.
135.             private void GetIndexesOfTopThreeScores(float[] scores, int n,
    out int index0, out int index1, out int index2)
136.             {
137.                 int i;
138.                 float first, second, third;
139.                 index0 = index1 = index2 = 0;
140.                 if (n < 3)
141.                 {
142.                     Console.WriteLine("Invalid Input");
143.                     return;
144.                 }
145.                 third = first = second = 000;
146.                 for (i = 0; i < n; i++)
147.                 {
148.                     if (scores[i] > first)
149.                     {
150.                         third = second;
```

```
151.                         second = first;
152.                         first = scores[i];
153.                     }
154.                     else if (scores[i] > second)
155.                     {
156.                         third = second;
157.                         second = scores[i];
158.                     }
159.                     else if (scores[i] > third)
160.                         third = scores[i];
161.                 }
162.                 var scoresList = scores.ToList();
163.                 index0 = scoresList.IndexOf(first);
164.                 index1 = scoresList.IndexOf(second);
165.                 index2 = scoresList.IndexOf(third);
166.             }
167.
168.             public async Task LabelAllNewIssuesInGitHubRepo()
169.             {
170.                 var newIssues = await GetNewIssues();
171.                 foreach (var issue in newIssues.Where(issue => !issue.Label
    s.Any()))
172.                 {
173.                     var label = PredictLabels(issue);
174.                     ApplyLabels(issue, label);
175.                 }
176.             }
177.
178.             private async Task<IReadOnlyList<Issue>> GetNewIssues()
179.             {
180.                 var issueRequest = new RepositoryIssueRequest
181.                 {
182.                     State = ItemStateFilter.Open,
183.                     Filter = IssueFilter.All,
184.                     Since = DateTime.Now.AddMinutes(-10)
185.                 };
186.
187.                 var allIssues = await _client.Issue.GetAllForRepository(_re
    poOwner, _repoName, issueRequest);
188.
189.                 return allIssues.Where(i => !i.HtmlUrl.Contains("/pull/"))

190.                                     .ToList();
191.             }
```

```
192.
193.                 private FullPrediction[] PredictLabels(Octokit.Issue issue)
194.                 {
195.                     var corefxIssue = new DocumentIssue
196.                     {
197.                         DocumentContent = issue.Body
198.                     };
199.
200.                     _fullPredictions = Predict(corefxIssue);
201.
202.                     return _fullPredictions;
203.                 }
204.
205.                 public FullPrediction[] Predict(DocumentIssue issue)
206.                 {
207.                     var prediction = _predEngine.Predict(issue);
208.
209.                     var fullPredictions = GetBestThreePredictions(prediction);

210.
211.                     return fullPredictions;
212.                 }
213.
214.                 private void ApplyLabels(Issue issue, FullPrediction[] fullPred
    ictions)
215.                 {
216.                     var issueUpdate = new IssueUpdate();
217.
218.                     foreach (var fullPrediction in fullPredictions)
219.                     {
220.                         if (fullPrediction.Score >= 0.3)
221.                         {
222.                             issueUpdate.AddLabel(fullPrediction.PredictedLabel)
    ;
223.                             _client.Issue.Update(_repoOwner, _repoName, issue.N
    umber, issueUpdate);
224.                         }
225.                     }
226.                 }
227.             }
228.         }
229.
```

EK-2 OCR Engine C# Kodu

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Drawing;
4.  using System.Drawing.Imaging;
5.  using System.IO;
6.  using System.Linq;
7.  using System.Threading.Tasks;
8.  using Windows.Globalization;
9.  using Windows.Graphics.Imaging;
10. using Windows.Media.Ocr;
11. using Windows.Storage.Streams;
12.
13. namespace ConAppOcr
14. {
15.     internal class UwpOcrEngine : IOcrEngine
16.     {
17.         public string RunOcr(byte[] imageBytes)
18.         {
19.             return RunOcrAsync(imageBytes).GetAwaiter().GetResult();
20.         }
21.
22.         /// <summary>
23.         /// Mains the asynchronous.
24.         /// </summary>
25.         /// <returns></returns>
26.         private static async Task<string> RunOcrAsync(byte[] imageBytes)
27.         {
28.             OcrEngine ocrEngine = OcrEngine.TryCreateFromLanguage(new Language
    ("tr"));
29.             if (ocrEngine == null) return null;
30.
31.             List<string> pageTexts = new List<string>();
32.
33.             using (var imgStream = new MemoryStream(imageBytes))
34.             {
35.                 using (var image = Image.FromStream(imgStream))
36.                 {
37.                     int pageCount = image.GetFrameCount(FrameDimension.Page);

38.                     for (int i = 0; i < pageCount; i++)
39.                     {
```

```
40.                         image.SelectActiveFrame(FrameDimension.Page, i);
41.                         using (var imgPageStream = new MemoryStream())
42.                         {
43.                             image.Save(imgPageStream, ImageFormat.Tiff);
44.                             using (var randomAccessStream = await ConvertToRan
    domAccessStream(imgPageStream))
45.                             {
46.                                 BitmapDecoder decoder = await BitmapDecoder.Cr
    eateAsync(randomAccessStream);
47.                                 var ocrResult = await ocrEngine.RecognizeAsync
    (await decoder.GetSoftwareBitmapAsync());
48.                                 pageTexts.Add(string.Join(" ", ocrResult.Lines
    .Select(l => l.Text)));
49.                             }
50.                         }
51.                     }
52.                 }
53.             }
54.
55.             return string.Join(" ", pageTexts);
56.         }
57.
58.         /// <summary>
59.         /// Converts to random access stream.
60.         /// </summary>
61.         /// <param name="memoryStream">The memory stream.</param>
62.         /// <returns></returns>
63.         private static async Task<IRandomAccessStream> ConvertToRandomAccessSt
    ream(MemoryStream memoryStream)
64.         {
65.             var randomAccessStream = new InMemoryRandomAccessStream();
66.             var outputStream = randomAccessStream.GetOutputStreamAt(0);
67.             var dw = new DataWriter(outputStream);
68.             await Task.Factory.StartNew(() => dw.WriteBytes(memoryStream.ToArr
    ay()));
69.             await dw.StoreAsync();
70.             await outputStream.FlushAsync();
71.             return randomAccessStream;
72.         }
73.     }
74. }
75.
76. namespace ConAppOcr
77. {
```

```
78.     internal interface IOcrEngine
79.     {
80.         /// <summary>
81.         /// Runs the ocr.
82.         /// </summary>
83.         /// <param name="imageBytes">The image bytes.</param>
84.         /// <returns></returns>
85.         string RunOcr(byte[] imageBytes);
86.     }
87. }
88.
89. using System;
90. using System.IO;
91.
92. namespace ConAppOcr
93. {
94.     class Program
95.     {
96.         static void Main(string[] args)
97.         {
98.             IOcrEngine ocrEngine = new UwpOcrEngine();
99.             string fileContent =
100.                         ocrEngine.RunOcr(File.ReadAllBytes(args[0]));
101.                 Console.WriteLine(fileContent);
102.             }
103.         }
104.     }
```