KTO KARATAY ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
ELEKTRİK VE BİLGİSAYAR MÜHENDİSLİĞİ TEZLİ YÜKSEK
LİSANS PROGRAMI

CEP TELEFONU İLE ALINAN İVME VERİLERİNDEN SÜRÜCÜLERİN KASİS GEÇİŞLERİNİN SINIFLANDIRILMASI

Bahadırhan KELEŞ

Yüksek Lisans Tezi

KONYA

Aralık 2022

CEP TELEFONU İLE ALINAN İVME VERİLERİNDEN SÜRÜCÜLERİN KASİS GEÇİŞLERİNİN SINIFLANDIRILMASI

Bahadırhan KELEŞ

KTO Karatay Üniversitesi
Lisansüstü Eğitim Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı
Elektrik ve Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı

Yüksek Lisans Tezi

Tez Danışmanı: Dr. Öğr. Üyesi H. Oktay ALTUN

Konya
Aralık 2022

BİLDİRİM

Enstitü tarafından onaylanan Yüksek Lisans tezimin tamamını veya herhangi bir kısmını basılı veya dijital biçimde arşivleme ve aşağıda belirtilen koşullar dahilinde erişime açma iznini KTO Karatay Üniversitesine verdiğimi bildiririm. Bu izinle, Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak ve gelecekteki çalışmalar (makale, kitap, lisans, patent vb.) için tezimin tamamının veya bir bölümünün kullanım hakları yalnızca bana ait olacaktır.

Tezimin bütünüyle kendi çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Telif hakkı bulunan ve sahiplerinden yazılı izinle kullanılması zorunlu olan kaynakları, yazılı izin alarak kullandığımı ve istenildiğinde izinlerin suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayımlanan "Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge" kapsamında, tezim, aşağıda belirtilen koşullar haricince, YÖK Ulusal Tez Merkezi ve KTO Karatay Üniversitesi Açık Erişim Sisteminde erişime açılır.

☐ Enstitü / Fakülte Yönetim Kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir.[1]

☐ Enstitü / Fakülte Yönetim Kurulunun gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren ... ay ertelenmiştir.[2]

☐ Tezimle ilgili gizlilik kararı verilmiştir.[3][4]

14 Aralık 2022

Bahadırhan KELEŞ

[1] MADDE 6(1) Lisansüstü tezle ilgili patent başvurusu yapılması veya patent alma sürecinin devam etmesi durumunda, tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulu iki yıl süre ile tezin erişime açılmasının ertelenmesine karar verebilir.

[2] MADDE 6(2) Yeni teknik, materyal ve metotların kullanıldığı, henüz makaleye dönüşmemiş veya patent gibi yöntemlerle korunmamış ve internetten paylaşılması durumunda 3. şahıslara veya kurumlara haksız kazanç imkanı oluşturabilecek bilgi ve bulguları içeren tezler hakkında tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulunun gerekçeli kararı ile altı ayı aşmamak üzere tezin erişime açılması engellenebilir.

[3] MADDE 7(1) Ulusal çıkarları veya güvenliği ilgilendiren, emniyet, istihbarat, savunma ve güvenlik, sağlık vb. konulara ilişkin lisansüstü tezlerle ilgili gizlilik kararı, tezin yapıldığı kurum tarafından verilir. Kurum ve kuruluşlarla yapılan işbirliği protokolü çerçevesinde hazırlanan lisansüstü tezlere ilişkin gizlilik kararı ise, ilgili kurum ve kuruluşun önerisi ile enstitü veya fakültenin uygun görüşü üzerine üniversite yönetim kurulu tarafından verilir. Gizlilik kararı verilen tezler Yükseköğretim Kuruluna bildirilir.

[4] MADDE 7(2) Gizlilik kararı verilen tezler gizlilik süresince enstitü veya fakülte tarafından gizlilik kuralları çerçevesinde muhafaza edilir, gizlilik kararının kaldırılması halinde Tez Otomasyon Sistemine yüklenir.

ETİK BEYAN

KTO Karatay Üniversitesi Lisansüstü Eğitim Enstitüsü Tez Hazırlama ve Yazım Kurallarına uygun olarak Dr. Öğr. Üyesi H. Oktay ALTUN danışmanlığında tarafımdan üretilen bu tez çalışmasında; sunduğum tüm veri, enformasyon, bilgi ve belgeleri bilimsel etik kuralları çerçevesinde elde ettiğimi, tüm değerlendirme, analiz, bulgu ve sonuçları bilimsel usullere uygun olarak sunduğumu, tez çalışmasında yararlandığım kaynakların tümüne bilimsel normlara uygun biçimde atıfta bulunarak kaynak gösterdiğimi, tezimin kaynak gösterilen durumlar dışında özgün olduğunu bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

14 Aralık 2022

Bahadırhan KELEŞ

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, değerli bilgilerini benimle paylaşan, kullandığı her kelimenin hayatıma kattığı önemini asla unutmayacağım saygıdeğer danışman hocam; Dr. Öğr. Üyesi H. Oktay ALTUN'a, çalışmam boyunca benden bir an olsun yardımlarını esirgemeyen çalışma arkadaşlarım Önder DEMİRTAŞ, Hüseyin YAVUZ ve Mücahit PALA'ya sonsuz teşekkürlerimi sunarım.

Ayrıca bana çalışma alanı sunan KTO Karatay Üniversitesi'ne bunun yanında tezi tamamlamamda beni destekleyen şirketim YCP Bilgi Teknolojileri'ne teşekkürlerimi sunarım.

Bahadırhan KELEŞ
Aralık-2022

ÖZET

Bahadırhan KELEŞ

Cep Telefonu ile Alınan İvme Verilerinden
Sürücülerin Kasis Geçişlerinin Sınıflandırılması

Yüksek Lisans Tezi

Konya, 2022

Kasislerden hızlı ve dikkatsiz geçmek motorlu araçlara zarar verebilir. Kasisin araca vereceği zararı en aza indirgemek için doğru hızla kasis üzerinden geçilmesi gerekir. Sürüş esnasında kasis görüldüğünde frene basarak aracın hızı azaltılmalıdır. Kasise mümkün olduğunca yavaş girilmeli ve kasis üzerindeyken fren bırakılmalıdır. Özellikle kiralık araçlarda araç kiralayan kişilerin özensizliği araç kiralama şirketlerine ciddi maliyetler çıkarmakta, bu maliyetler de yine kiralama fiyatlarına yansıtılmaktadır. Bu çalışma, sürücülerin kasislerden nasıl geçtiği ile ilgili fikir edinilebilmesi amacıyla yapılmıştır. Geliştirilen mobil uygulama sayesinde sürücüler araba kullanırken ivmelerinin ve kasise hızlı girip girmediğinin bilgisi alınmıştır. Sürücülere ait yaklaşık olarak 85.000 satırlık veri, k-merkez (İng., k-means) ve ağaç çizit (İng., dendogram) algoritmaları ile değerlendirilmiş ve böylelikle sürücülerin sınıf sayıları doğrulanmıştır. Etiketlenen verilerin öğenilmesi ise XGBoost, yapay sinir ağları ve evrişimli sinir ağları algoritmalarıyla gerçekleştirilmiştir. Bu algoritmaların sınıflandırma performansı değerlendirilmiştir. Sürücülerin araba kullanırken kasislerden nasıl geçtiği belirlenmiştir.

Anahtar Kelimeler

İvme verilerinden sürüş sınıflandırma, derin öğrenme, k-merkezli kümeleme, evrişimli sinir ağları

ABSTRACT

Bahadırhan KELEŞ

Classification Of Drivers' Transition Over Speed Bumps Based On Acceleration Data Via Celluar Phones

Master's Thesis

Konya 2022

Driving quickly and carelessly through bumps can damage motor vehicles. To minimize damage to the vehicle, it is necessary to drive over the bump at the right speed. During the journey, the speed of the vehicle must be reduced in the sight of the collision. The collision should be entered as slowly as possible and the brake should be released while on the bump. In particular, the carelessness of users who rent a car in vehicles eliminates serious costs to car rental companies, and this is reflected in the rental prices again. In this study, it was aimed to give an idea about how drivers pass through bumps. Via the developed mobile application, data was obtained about the acceleration of driving and entering the accident quickly. Approximately 85,000 lines of data of the drivers were determined by k-means algorithm and the number of classes of the drivers were determined. The classification of the tagged data was carried out with the implementation of XGBoost, artificial neural networks and convolutional neural networks. Their performance performance was evaluated. As a result, it was determined how the drivers pass through the bumps when driving.

Keywords:

Classification of drivers' transition over speed bumps, deep learning, k-means clustering, convolutional neural networks

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

1. GİRİŞ

Araç sigorta endüstrisi, geleneksel olarak primlerini kümelenmiş gruplara göre hesaplamaktadır. Makne öğrenmesi algoritmaları ile müşterilerin bireysel olarak değerlendirildiği bir sisteme geçerek, müşteri memnuniyetini ön plana çıkaran yaklaşımları tercih etmeye yönelik eğilimler bulunmaktadır. Son zamanlarda yapılan araç sigortalarındaki maliyet artışları ile sigorta şirketleri, iyi sürücü olan ve karlı müşterileri kaybetmemek için bireysel araç sigortası pirim hesaplamasına geçiş yapmanın kar oranını arttıracağını fark etmişlerdir. Müşterilerin yalnızca kullandıkları kilometre bazında değerlendirildiği sisteme baktığımızda, kullandığın kadar öde prensibine dayalı geleneksel sigorta yaklaşımları popülerlik kazanmaktadır. Bu fiyatlandırma yöntemi sürücülerin çoğu için geçerli olamayacak kadar genel istatistiklere dayanmaktadır. Örneğin, erkekler hayatları boyunca kadınlara kıyasla ortalama 15.000 dolar daha fazla para ödemektedirler ve genç sürücüler dünya çapında uygun fiyatlı sigortalar bulmakta zorlanmaktadır. (Bosari, 2013).

Sigorta ve araç kiralama firmaları arabaya monte edilen karakutular ve bunların içerisindeki bir grup yüksek seviye sensörleri kullanarak sürüş davranışlarını izleyip değerlendirebilmektedirler. Bu karakutuların maliyetinin yüksek olması nedeniyle bazı projelerin başarısız bir şekilde sonlandığı bilinmektedir. Bu maliyeti azaltmak adına, bu çalışmada mobil uygulama ile veriler alınıp maliyet minimuma indirgenmeye çalışılmıştır. Artık sürüş agresifliği, refleks ve hızlanma davranışlarına bağlı olarak daha doğru bir bireysel fiyatlandırma yapılabilir. Yapılan bu bireysel fiyatlandırma ile sigorta ve araç kiralama firmalarının kar marjının artacağı öngörülmektedir. Sonuç olarak, müşterilerin çoğu dikkatli bir şekilde aracı sürdüklerinde hem aracın aksamlarının

eskimesinin önüne geçilecek, hem de müşterilerin alacakları hizmet karşılığında ödeyecekleri ücretler düşecektir. Bunun aksi durumlarda agresif ve dikkatsiz sürüş yapan sürücülerin ise ücretleri artacaktır.

Dikkatli ve düzgün sürüşe göre fiyatlandırma sistemini için geliştirilecek karakutu sistemlerinin birçok avantajları olmasına rağmen, bu sistemlerin birtakım dezavantajları da olabilir: Yüksek kurulum ve bakım maliyeti, araçların ortak kullanımı durumları vb. Bu dezavantajların üstesinden gelmek için akıllı telefon sensörlerinden yararlanmamızı sağlayan özel bir yazılım geliştirilmiştir. Akıllı telefonların büyük kısmı, pahalı karakutuların içerisinde bulunan ivme ölçer, jiroskop, manyetometre ve GPS gibi sensörleri halihazırda içermekte olup, çok daha hesaplıdır (Sentiance, 2018).

Günümüzde, hemen herkesin mobil cihazının olduğu düşünülürse, geliştirilen bu ve benzeri yaklaşımlar daha da çekici hale gelmektedir. Mevzubahis sensörler, telefonun yönü değiştiğinde ekranı otomatik olarak çevirmek, navigasyon uygulamalarını kullanabilmek ve bazı oyunlar için akıllı telefonlara halihazırda entegre edilmiştir. Bu çalışmada ivmeölçer vasıtasıyla sürücünün kasislerden geçişleri kayıt altına alınmıştır. Sonrasında, üzerinde kasisler bulunan bir yolda farklı sürüş tarzlarıyla toplanan verilere makine öğrenmesi teknikleri uygulanarak sürücülerin kasis geçişlerinin sınıflandırılması sağlanmıştır .

2. İLGİLİ ÇALIŞMALAR

Bir sigorta şirketi için, sürücülerin gruplandırılması ve sınıflandırılması, daha fazla kişiselleştirilmiş bir sigorta poliçesi oluşturulması açısından önem arz etmektedir. Seyir halindeki araç miktarının tüm dünyada artmaya devam etmesi, riskleri öngörebilme gerekliliği bir sigorta şirketinin rezervdeki parasını kaybetmesinin önlenmesine kesinlikle yardımcı olabilir. Risk analizinin çözümünde mevcut metodolojilere alternatif olarak sinir ağlarının kullanılması ve araç sürücüsü davranışının risk tahmin problemi ele alınmıştır. Kişiselleştirilmiş sigorta poliçelerinin geliştirilmesinde sınıflandırma analizi kullanılarak yapılan çalışmalarda araç sürücüsü davranışına ilişkin risk analizi ve risk tahmini probleminin çözümünde mevcut metodolojilere alternatif olarak sinir ağlarının kullanımına yer verilmiştir.

Yapılan çalışmalarda karakteristik analiz için üç farklı faktör ele alınmıştır. Bunlardan sürücü faktörleri; sürücü yaşı, sürücü cinsiyeti, sürücü deneyimi ve sürücü kaza geçmişi olarak dört alt kategoriye ayrılmıştır. Araç faktörleri; araç tipi, santimetreküp, beygir gücü ve araç yaşı olmak üzere dört farklı kategoriye ayrılmıştır. Bununla birlikte her faktörün birkaç ölçeği vardır. Son olarak genel faktör; hem sürücüyü hem de aracı etkileyen ve sınıflandırma için kullanılan, karayolu ortamı, yani sürücünün aracını sürdüğü alandır. Yapay sinir ağı modeline girdi modeli olarak kullanılacak faktörleri belirledikten sonra, çıktı modeli yani sürücülerin hangi kategoride sınıflandırılabileceği belirlenmiştir. Risk analizi ve risk tahmininde daha fazla kullanıma izin vererek sinir ağı modeli için gerekli çıktı modelini sağlanmıştır (Ekonomou, 2012).

Sürücü davranışı riskini değerlendirmek için ivme ölçer verilerini ve bağlamsal

değişkenleri birleştirmek makalesinde, araç içi veri kaydediciler tarafından üretilen ivmeölçer ve hız verilerini sürücü davranışını belirlemek için referans olarak ele alınmıştır. Araştırmada büyük hacimli verilerle başa çıkmak için zorlu olayları ayıklamak yerine, verileri izlenebilir ve sınırlı bir risk alanına ayrılmıştır, böylelikle hem kabul edilebilir hem de kabul edilemez sürüş davranışını izlenmesinde ve veri zarfını kullanarak öncelikli eşikleri değil, daha kapsamlı bir risk modeli hesaplanmasına olanak sağlanmıştır (Johan W. Joubert, 2016).

Sürücüleri sınıflandırmak için pencere tabanlı bir destek vektör makinesi modeli geliştirmiştir (İng., window based support vector machine). Model hem kontrollü deney ortamında hem de doğal koşullar altında toplanan iki veri seti ile test edilmiştir. Her bir sensör kaynağı (örneğin araba veya telefon) bağımsız olarak kullanılarak ve ayriyeten iki sensör birleştirilerek de model değerlendirilmiştir. Doğal bir ortamda evli çiftlerarasında paylaşılan üç farklı arabadan toplanan verilerle elde edilen ortalama sınıflandırma doğrulukları, yalnızca telefon sensörleri, yalnızca arabasensörleri ve kombine araba ve telefon sensörleri kullanılarak sırasıyla %75,83, %85,83 ve %86,67 olmuştur (Cheng Zhang, 2016).

Sürüş tarzı karakteristik olarak normal ve agresif olarak ikiye ayrılabilir. İlgili araştırmalar, GPS yardımıyla aracın atalet ölçüm sinyalleri kullanılarak sürüş tarzı hakkında faydalı bilgilerin elde edilebileceğini göstermektedir. Ancak, toplu taşıma araçları için genellikle rotanın tekrar etmesi nedeniyle GPS sensörü gerekli değildir. Bu varsayım, agresif ve normal sürücüyü sınıflandırma yeteneğine sahip düşük maliyetli akıllı toplu taşıma izleme sistemi oluşturmaya yardımcı olur. Aynı rotayı farklı sürüş tarzlarında sürerken ivmeölçer verilerini kullanarak uzman değerlendirmesi ve bilgisi olmadan sürüş tarzını otomatik olarak agresif veya normal olarak sınıflandırmak için örüntü tanıma yaklaşımı önerilmiştir. Sınıflandırıcı girdileri olarak 3 eksenli ivmeölçer sinyali istatistiksel özellikleri kullanılmıştır. Sonuçlar agresif ve normal sürüş stili

sınıflandırmasının %100 olduğunu göstermektedir (Vygandas Vaitkus, 2014).

Akıllı telefon tabanlı sürüş davranış değerlendirme sistemi çalışmasında, sürücülerin sürüş davranışlarının ne kadar agresif olduğunu fark etmelerine ve yolcuların sürüş konforu seviyesinin farkında olmalarına yardımcı olan, Join Driving adlı akıllı telefon tabanlı bir sürüş davranışı değerlendirme sistemi geliştirilmiştir. Önerilen değerlendirme sistemi iki bölümden oluşmaktadır: Sürüş olaylarını tespit ve değerlendirme bölümü ve sürüş konfor seviyesi değerlendirme bölümü. Sürüş olaylarının tespiti ve değerlendirilmesi bölümünde, önerilen sistem olan Join Driving, öncelikle akıllı telefonlardaki hızlanma, yönlendirme ve GPS sensörlerinden toplanan verilere dayanarak sürücülerin sürüş olaylarını tespit etmek için bir model sunmaktadır. Ardından, tespit edilen sürücülerin sürüş olaylarına dayalı olarak, Join Driving, bu sürüş olaylarının ne kadar agresif olduğunu nicel olarak değerlendirmek için yeni bir puanlama mekanizması uygulamıştır. Sürüş konfor seviyesi değerlendirme bölümünde, önerilen sistem yolcuların sürüş konfor seviyesini ISO 2631'e göre derecelendirmek için belirli puanlar vermektedir. Son olarak, önerilen puanlama sisteminin etkinliğini değerlendirmek için birkaç pratik deney yapılmıştır (Hongyang Zhao, 2013).

3. KURAMSAL ÇERÇEVE

Bu çalışmada sınıflandırma metodolojisi olarak yapay zekanın alt dallarından biri olan makine öğrenmesi algoritmaları kullanılmıştır. Bu algoritmalarda denetimsiz öğrenme metodlarıyla sürücülerin kaç sınıfa ayrılacağının sayısı doğrulanmış olup, denetimli öğrenme yöntemlerinden sınıflandırma işlemlerinde de kullanılan derin öğrenme yöntemlerinden evrişimli sinir ağları (CNN) kullanılmıştır.

3.1. Makine Öğrenmesi

Makine öğrenmesi, bilgisayarların kural tabanlı olarak programlanmadan, verilerden dolayımlı olarak öğrenerek aksiyon/karar almayı sağlayan bir disiplin olarak tanımlanabilir. Gerçek dünyadan toplanan verilerden anlam çıkarma, veriler arasında örüntü bulma, girdi verilerinin hedef değerlerle olan ilişkisinin yorumlanmasında kullanılmaktadır. Veri madenciliği, doğal dil işleme, görüntü/ses tanıma ve uzman sistemler gibi çok çeşitli kritik uygulamalarda kilit bir rol oynamaktadır. Makine öğrenimi konusundaki algoritmalara her geçen gün yeni algoritmalar eklenerek tahminleme süreci daha da kolaylaşmaktadır. Projelerde en çok zaman alan bölümler verilerin toplanması ve işlenmesi süreçlerinde yaşanmaktadır. Makine öğrenimi denetimli, denetimsiz ve takviyeli öğrenme olarak temelde 3 ana bölümde incelenmektedir. Takviyeli öğrenme, bir aracının eylemleri gerçekleştirerek ve sonuçları görerek bir ortamda nasıl davranacağını öğrendiği önemli bir makine öğrenmesi türü olmasıyla birlikte çalışmaya konu olacak bir durum içermediğinden kullanılmamıştır. Makine öğreniminde çalışmada etiketlenen verilerin bulunması nedeniye denetimli öğrenme, sınıflandırmanın işleminin çarpraz kontrolünün yapılabilmesi için denetimsiz öğrenme ve denetimli öğrenme altında eti-

ketli verilerin tahminlemesi aşamasında derin öğrenme algoritmaları kullanılmıştır.

3.2. Denetimli Öğrenme

Denetimli öğrenmenin amacı, bir veri örneği ve istenen çıktılar verildiğinde, veride gözlenen girdi ve çıktı arasındaki ilişkiyi en iyi şekilde formüle eden bir işlevi öğrenmektir. Denetimli öğrenme sınıflandırma veya regresyon olarak ikiye ayrılmaktadır. Bizim çalışmamızda ikili sınıflandırma yapılacağı için, bu çalışma sınıflandırma alanına girmektedir.

Denetimli öğrenmedeki yaygın algoritmalar arasında lojistik regresyon, naif Bayes'ler, destek vektör makineleri, yapay sinir ağları ve rastgele ormanlar bulunur. Hem regresyon hem de sınıflandırmada amaç, girdi verilerinde etkili bir şekilde doğru çıktı verisi üretmemizi sağlayan belirli ilişkileri veya yapıları bulmaktır. "Doğru" çıktı tamamen eğitim verilerinden belirlenir, bu nedenle modelimizin doğru varsaydığı kesin bir gerçeğe sahip olsak da, veri etiketlerinin her zaman gerçek dünyada doğru olduğunu söylemek mümkün değildir. Gürültülü veya yanlış veri etiketleri, modellerin doğruluğunu açıkça azaltır.

Denetimli öğrenmeyi yürütürken, temel hususlar model karmaşıklığı ve yanlılık değişiminin giderilmesidir. Her ikisinin de birbiriyle ilişkili olduğuna dikkat edilmelidir. Bir polinomun derecesine benzer şekilde model karmaşıklığı, öğrenmeye çalışılan işlevin karmaşıklığını ifade eder. Uygun bir model karmaşıklığı seviyesi genellikle eğitim verilerinin niteliği ile belirlenir. Az miktarda veri varsa veya veriler farklı olası senaryolar boyunca eşit bir şekilde dağılmamışsa, düşük karmaşıklıkta bir model seçmek gerekir. Bunun nedeni, az sayıda veri noktasında kullanıldığında yüksek karmaşıklık modelinin yerine geçmesidir. Aşırı yükleme, eğitim verilerinize çok uyan bir işlevi öğrenmeyi ifade eder, ancak diğer veri noktalarına genelleştirmez. İki nokta arasında bir eğri sığdırmaya çalıştığımızı düşünelim. Teoride, herhangi bir derece-

deki bir fonksiyonu kullanabilirsiniz, ancak pratikte, temel olarak karmaşıklığı ekler ve doğrusal bir fonksiyonla ilerleriz.

Yanlılık (İng., bias) ve varyans ilişkisi model genellemesi ile de yakından ilgilidir. Herhangi bir modelde, sabit hata terimi olan yanlılık ile hatanın farklı eğitim setleri arasında değişebileceği miktar olan varyans arasında bir denge vardır. Yanlılık ve varyansın tipik olarak birbirine zıt yönlerde hareket ettiği unutulmamalıdır. Yanlılığın artması genellikle daha düşük varyansa neden olur ve bunun tersi de geçerlidir. Modelimizioluştururken, spesifik problem ve verilerin niteliği, yanlılık-varyans spektrumunun neresine düşüleceğine dair bilinçli bir karar vermeyi sağlamalıdır. Genel olarak, artan yanlılık (ve azalan varyans), belirli görevlerde kritik olabilen, nispeten garantili temel performans seviyelerine sahip modellerle sonuçlanır. Ek olarak, iyi genelleyen modeller üretmek için modelimizin varyansını, eğitim verilerinin boyutuna ve karmaşıklığına göre ölçeklendirmeliyiz. Küçük ve basit veri kümeleri genellikle düşük varyanslı modellerle öğrenilmelidir. Öte yandan büyük ve karmaşık verilerin olduğu durumlarda ise genellikle verilerin yapısını tam olarak öğrenmek için daha yüksek varyans modelleri gerekecektir.

3.3. Denetimsiz Öğrenme

Denetimsiz öğrenme kümeleme ve boyut azaltımı başlıkları altında iki ana bölümde incelenmektedir. Kümeleme algoritmalarıyla benzer özellikler taşıyan verilerin aynı gruba dahil edilmesi işlemidir. Aynı grupta benzer özelliklerin çok fazla olması beklenirken, farklı gruplarda benzerliğin hemen hemen hiç yok denecek seviyede olması beklenmektedir. Denetimsiz öğrenme için iki yaygın kullanım örneği keşif analizi (İng., exploratory analysis) ve boyutsallığın azaltılmasıdır (İng., dimensionality reduction). Denetimsiz öğrenme, keşif analizinde (İng., exploratory analysis) çok kullanışlıdır, çünkü verilerdeki yapıyı otomatik olarak tanımlayabilir. Daha az sütun veya özellik kullanarak

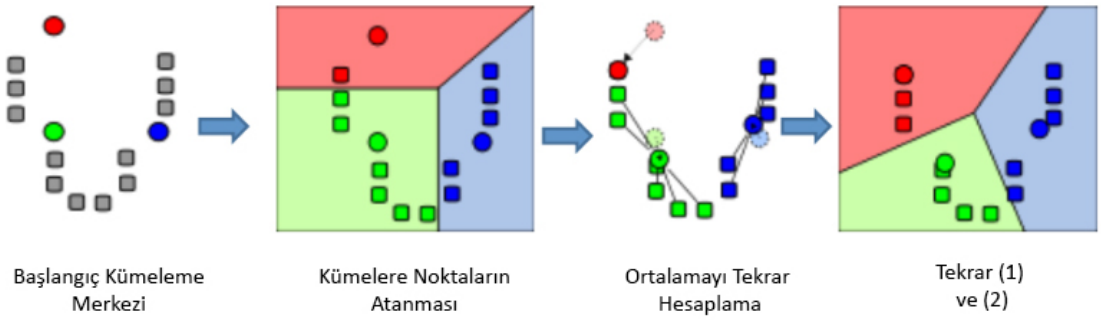
verileri temsil etmek için kullanılan yöntemleri ifade eden boyutsallık azaltma, denetimsiz yöntemlerle gerçekleştirilebilir. Temsil öğrenmede, bireysel özellikler arasındaki ilişkileri öğrenmek istiyoruz ve verilerimizi başlangıçtaki özelliklerimizle ilişkilendiren gizli özellikleri kullanarak göstermemize izin veriyoruz. Bu seyrek gizli yapı genellikle başladığımızdan çok daha az özellik kullanılarak temsil edilir, böylece daha fazla veri işlemeyi daha az yoğun hale getirebilir ve gereksiz özellikleri ortadan kaldırabilir. Boyut azaltımı ile veri setindeki çok sayıda özelliğin aslında mümkün olan en fazla oranda varyansının açıklanması işlemidir ki bu oran genelde %90'dan az olmayacak şekilde seçilir ve boyutların azaltımı işlemi gerçekleştirilir. Çalışmada çok fazla sayıda öznitelik bulunmadığı için bu metot kullanılmamıştır. Kümeleme algoritmalarından k-Merkezli öbekleme ve hiyerarşik bölütleme yaygın bir şekilde kullanıldığı için sınıf sayısının belirlenmesinde kullanılmıştır.

3.4. k-Merkezli Öbekleme

Kümelenme problemlerini çözmek için en basit ve yaygın kullanılan denetimsiz öğrenme algoritmalarından biridir. Prosedür, verilen bir veri setini algoritmayı çalıştırmadan önce konfigüre ettiğiniz belirli sayıda küme ile sınıflandırmanın basit ve kolay bir yolunu izler. Bu küme sayısına k adı verilir ve bu sayıyı eğitimli tahminle seçeriz. k-Merkezli öbekleme akış diagramı Şekil 3.1'da verilmiştir.

k-Merkezli öbekleme, kütle merkezi tabanlı kümeleme kategorisine girer. Bir kütle merkezi, kümenin ortasındaki bir veri noktasıdır. Kütle merkezi tabanlı kümelemede kümeler, merkezi bir vektör veya bir kütle merkezi ile temsil edilir. Bu kütle merkezi mutlaka veri kümesinin bir üyesi olmayabilir. Kütle merkezi tabanlı kümeleme, benzerlik kavramının bir veri noktasının kümenin kütle merkezine ne kadar yakın olduğu ile türetildiği yinelemeli bir algoritmadır. Örneğin; resim segmentasyonu (İng., image segmentation), gen segmentasyon verilerini kümeleme (İng., clustering gene segemen-

tation data), haber makaleleri kümeleme (İng., news article clustering), dil kümeleme (İng., clustering languages), tür kümeleme (İng., species clustering), anamoli tespiti (İng., anomaly detection).



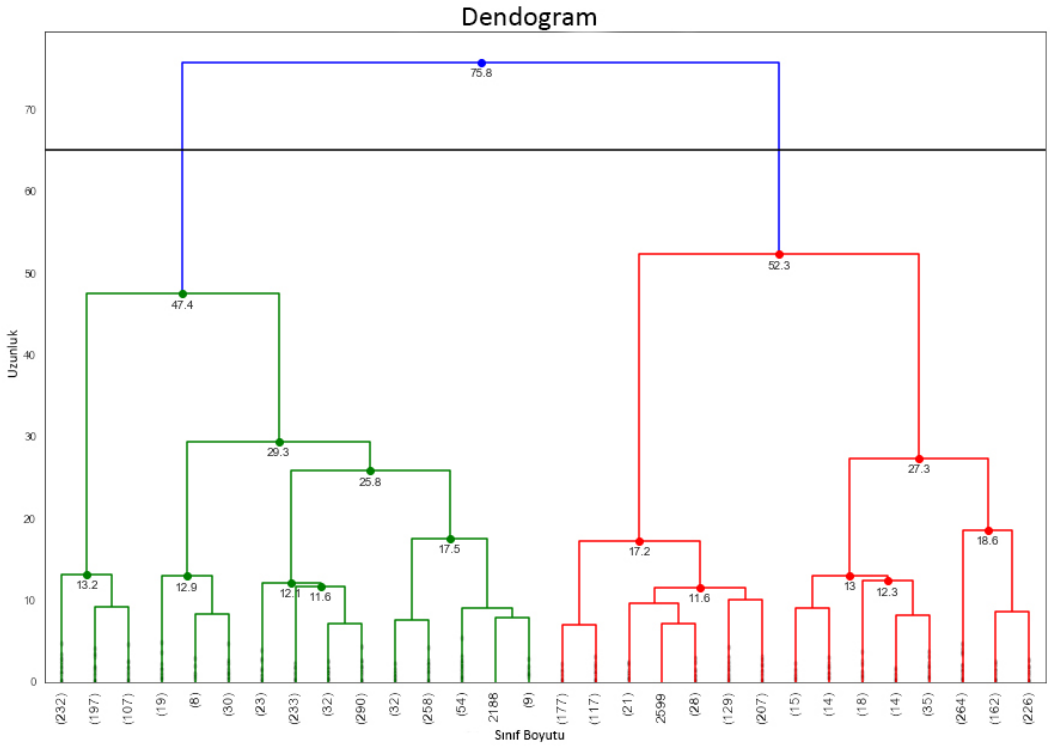
Şekil 3.1: k-Merkezli öbekleme akış diagramı (*k-means clustering, Demonstration of the standard algorithm*, 2018)

3.5. Hiyerarşik Bölütleme

Hiyerarşik kümelemede yukarıdan aşağıya doğru önceden belirlenmiş sıraya sahip kümeler oluşturulur. Toplayıcı kümeleme (İng., agglomerative clustering) 'de bunun bir türüdür. k-merkezli öbekleme'den farkı, kaç küme oluşturulacağı belirtilmez. Başlangıçta bütün örnekler (İng., samples) birer küme olarak kabul edilir. Daha sonra her bir küme kendisine en yakın diğer bir küme ile birleşir. En sonunda ortaya dendrogram şeması ve tek bir küme çıkar. Dendrogram, kümeler arasındaki hiyerarşik yapıyı göstermek için kullanılan ağaç şemasıdır. Dendrogram şemesı yorumlanarak, kaç tane küme oluşturulacağı belirlenir.

İlk adımda her bir öge kendi kümesine atanır (N örnek, N küme) ikinci adımda birbirine en yakın kümeler bulunur ve bunlar tek bir kümede birleştirilir. Üçüncü adımda yeni küme ile eski kümelerin her biri arasındaki uzaklıkları hesaplanır. Tüm öğeler tek bir küme halinde kümeleninceye kadar 2. ve 3. adımları tekrarlanmalıdır. Örnek bir den-

dogram yapısı Şekil 3.2'de görülmektedir.



Şekil 3.2: Dendrogram şeması

3.6. Derin Öğrenme

Derin öğrenme, insan beyninin veri işleme ve karar vermede kullandığı elektro-kimyasal ve biyolojik mekanizmaları taklit eden yapay bir zeka işlevidir. Derin sinir öğrenme veya derin sinir ağları olarak da bilinir. Klasik programlamada, geliştirici kuralları girer ve veriler program kurallarına göre girilen verilere bağlıdır.

Yapay sinir ağları için ilk fikirler 1940'lara dayanır. Temel kavram, birbirine bağlı eşik anahtarlarından oluşturulmuş bir yapay nöron ağının, bir hayvan beyninin ve sinir sisteminin (retina dahil) yaptığı gibi kalıpları tanımayı öğrenebilmesidir. Derin öğrenmede en önemli başlıklar aşağıdaki gibidir:

Geri Yayılım (İng., Backpropagation): Derin sinir ağlarında öğrenme, her ikisi de eğitim sırasında aynı anda aktif olduğunda iki nöron arasındaki bağlantıyı güçlendirerek gerçekleşir. Modern sinir ağı yazılımında bu genellikle hata, geri yayılım (İng., backpropation) işlemi adı verilen bir kural kullanarak nöronlar arasındaki bağlantılar için ağırlık değerlerini belirleme konusudur.

par Nöronlar: Nöronlar nasıl modellenir? Her biri, genellikle ağırlıklı bir toplamla, bağlı nöronların çıktılarını dönüştüren bir yayılma fonksiyonuna sahiptir. Yayılma fonksiyonunun çıktısı, girişi bir eşik değerini aştığında başlayan bir aktivasyon fonksiyonuna geçer.

Aktivasyon Fonksiyonları: 1940'larda ve 1950'lerde yapay nöronlar bir adım aktivasyon işlevi kullandı ve algılayıcı olarak adlandırıldı. Modern sinir ağları, algılayıcılar kullandıklarını söyleyebilirler, ancak lojistik veya sigmoid fonksiyonu, hiperbolik tanjant ve ReLU gibi aslında aktive edici fonksiyonlara sahiptirler. ReLU genellikle hızlı yakınsama için en iyi seçimdir, ancak öğrenme hızı çok yüksekse, eğitim sırasında "ölen" nöronları sorunu vardır. Aktivasyon fonksiyonunun çıkışı ek şekillendirme için bir çıkış fonksiyonuna geçebilir. Bununla birlikte, çoğu zaman, çıkış fonksiyonu, kimlik fonksiyonudur, yani aktivasyon fonksiyonunun çıktısı ile diğer bağlantılı nöronlara geçirilir.

Yapay Sinir Ağları Topolojileri: İleri beslemeli bir ağda, nöronlar farklı katmanlar halinde düzenlenir. Bir giriş katmanı, herhangi bir sayıda gizli işlem katmanı ve bir çıkış katmanı vardır, ve her katmandan gelen çıktılar yalnızca bir sonraki katmana gider. Kısayol bağlantıları olan ileri beslemeli bir ağda, bazı bağlantılar bir veya daha fazla ara katmandan atlayabilir. Tekrarlayan sinir ağlarında, nöronlar kendilerini doğrudan veya dolaylı olarak bir sonraki katmandan etkileyebilirler.

Eğitim: Bir sinir ağının denetimli öğrenimi, diğer herhangi bir makine öğrenmesi gibi yapılır. Ağa eğitim verisi grupları sunulur, ağ çıktısı istenen çıktıyla karşılaştırılır, bir

hata vektörü oluşturulur ve hata vektörünü temel alarak ağa düzeltmeler uygulanır. Düzeltmeler uygulanmadan önce bir araya getirilen eğitim verisi kümelerine batch denir.

Geri yayılma, hatayı en aza indirecek doğru yönü bulmak için modelin ağırlıklarına ve yanlılıklarına (İng., bias) göre hata işlevinin gradyanını kullanır. Düzeltmelerin uygulanmasını iki şey kontrol eder: Optimizasyon algoritması ve yakınsama garantisi vermek ve ölü ReLU nöronlarına neden olmaktan kaçınmak için genellikle küçük olması gereken öğrenme hızı değişkeni.

Eniyileştirme (Optimizers): Yapay sinir ağları için optimize ediciler genellikle geri yayılmayı yönlendirmek için genellikle bir çeşit gradyan iniş algoritması kullanırlar; genellikle rastgele seçilen mini *batch*'ler, (buna *stochastic gradient descent* algoritması denmektedir) optimize etmek ve gradyana momentum düzeltmeleri uygulamak gibi yerel minimumda sıkışıp kalmamak için bir mekanizma kullanırlar. Bazı optimizasyon algoritmaları, gradyan geçmişine (örn., AdaGrad, RMSProp ve Adam) bakarak model parametrelerinin öğrenme hızlarını da değiştirirler.

Tüm makine öğreniminde olduğu gibi, sinir ağının tahminlerini ayrı bir doğrulama verisine göre kontrol etmek gerekir. Bunu yapmadan, genelleştirilmiş belirleyici olmayı öğrenmek yerine yalnızca girdilerini ezberleyen sinir ağları oluşturma riski alınmış olur. Gerçek bir problem için derin bir sinir ağı, 10 gizli katmandan daha fazlasına sahip olabilir. Topolojisi basit veya oldukça karmaşık olabilir. Ağdaki katmanlar ne kadar fazlaysa, o kadar çok özelliği tanıyabilir. Ne yazık ki, ağdaki katmanlar ne kadar fazlaysa hesaplaması o kadar uzun sürer ve çalışması o kadar zorlaşır.

Derin öğrenme algoritmaları: Yukarıda da bahsedildiği gibi, birçok derin öğrenme derin sinir ağları ile yapılır. Evrişimsel sinir ağları (CNN) genellikle görüntü işlemede, tekrarlayan sinir ağları (RNN) genellikle, uzun kısa süreli bellek (LSTM) ağları ve dikkat temelli sinir ağları gibi doğal dil işleme ve diğer dizi işlemlerinde kullanılır. Çalışmada

ivme değişimden gelen sinyaller bazı kısımlarda özellikte genleşmede açık bir şekilde fark olduğu görselleştirme sonucunda görülmüş olup bu görüntüdeki farklılığı yakalayabilmek için evrişimsel sinir ağları kullanılmıştır.

Evrişimsel Sinir Ağları: Evrişimsel sinir ağları tipik olarak görsel bir korteksi simüle etmek için konvolüsyon, havuzlama, ReLU, tamamen bağlı ve kayıp tabakaları kullanır. Evrişimli tabaka temel olarak birçok küçük örtüşen bölgenin integralini alır. Havuzlama katmanı, doğrusal olmayan bir aşağı alt örnekleme biçimini gerçekleştirir. ReLU katmanları, aktivasyon fonksiyonunu $f(x) = max(0,x)$ olarak uygular. Tamamen bağlı bir katmanda, nöronların önceki kattaki tüm aktivasyonlara bağlantıları vardır. Bir kayıp katmanı, ağ eğitiminin, sınıflandırma için bir Softmax veya çapraz entropi kaybı fonksiyonunu veya regresyon için bir Euclid kaybı fonksiyonunu kullanarak öngörülen ve doğru etiketler arasındaki sapmayı nasıl cezalandırdığını hesaplar (Heller, 2019).

3.7. Derin Öğrenmenin Tarihçesi

Makine öğrenimi'nin bir dalı olarak derin öğrenme, verileri işlemek, düşünme sürecini taklit etmek veya soyutlamalar geliştirmek için algoritmalar kullanır. Verileri işlemek için algoritma katmanlarını (layer) kullanır. Bilgi, her bir katmanın içinden geçer, bir önceki katmanın çıktısı, sonraki katman için girdi olarak nitelendirilir. Bir ağdaki ilk katman giriş katmanı, son katman ise çıkış katmanı olarak adlandırılır. Bu ikisi arasındaki tüm katmanlara gizli katmanlar (hidden layer) denir. Her katman tipik olarak bir tür aktivasyon fonksiyonu içeren basit, tek tip bir algoritmadır.

Özellik çıkarma (İng., feature extraction), derin öğrenmenin başka bir kullanım alanıdır. Özellik çıkarma, eğitim, öğrenme ve anlama amacıyla verilerin anlamlı "özelliklerini" otomatik olarak oluşturmak için bir algoritma kullanır. Normal olarak veri bilimcisi veya programcısı, bu özelliklerin çıkarılmasından sorumludur.

Derin öğrenmenin tarihi, Walter Pitts ve Warren McCulloch'un insan beyninin si-

nir ağlarına dayanan bir bilgisayar modeli yarattığı 1943 yılına kadar takip edilebilir. Düşünce sürecini taklit etmek için "eşik mantığı" (threshold logic) olarak adlandırdıkları bir algoritma ve matematik kombinasyonu kullandılar. O zamandan beri, derin öğrenme gelişiminde iki önemli kırılma ile, sürekli gelişti.

Henry J. Kelley'ye 1960'da sürekli bir geri yayılım (Back Propagation) modelinin temellerini geliştirmek için kredi verildi. 1962'de, Stuart Dreyfus tarafından yalnızca zincir kuralına dayanan daha basit bir versiyon geliştirildi. 1960'ların başında geri yayılma kavramı (hataların antrenman amacıyla geri yayılması) ortaya çıkmış olsa da, sakar ve verimsizdi ve 1985'e kadar işe yaramazdı. 1960'ların başında geri yayılma kavramı (hataların eğitim amacıyla geri yayılması) ortaya çıkmış olsa da verimsizdi ve 1985'e kadar işe yaramazdı.

Derin öğrenme algoritmaları geliştirmedeki en erken çabalar, 1965'te Alexey Grigoryevich Ivakhnenko (Grup Veri İşleme Yöntemini geliştirdi) ve Valentin Grigorevich Lapa'dan (Sibernetik ve Tahmini Teknikleri'nin yazarı) geldi. Polinom (karmaşık denklemler) aktivasyon fonksiyonlarına sahip modeller kullanıldı, ardından istatistiksel olarak analiz edildiler. Her katmandan, istatistiksel olarak en iyi seçilen özellikler bir sonraki katmana iletildi (yavaş, manuel bir işlem) böylelikle basit nöral ağ yapısının temeli atılmış oldu.

İlk evrimsel sinir ağları (convolutional neural networks) Kunihiko Fukushima tarafından kullanıldı. Fukushima çoklu havuzlama ve evrişimli katmanlara sahip sinir ağları tasarladı. 1979'da, hiyerarşik, çok katmanlı bir tasarım kullanan Neocognitron adıyla yapay bir sinir ağı geliştirdi. Bu tasarım, bilgisayarın görsel desenleri tanımasını "öğrenmesini" sağladı. Ağlar modern versiyonlara benziyordu, ancak zamanla güçlenen çoklu katmanlarda tekrarlanan aktivasyon takviye stratejisi ile eğitildi. Fukushima'nın tasarımı, belirli bağlantıların "ağırlığını" artırarak önemli özelliklerin manuel olarak ayarlanmasına izin vermiştir (Foote, 2017).

3.8. Derin Öğrenmenin Mühendislik Alanlarında Kullanımı

Makine öğrenmesinin, otonom araçlardaki sensör verilerinin işlenmesinde kullanıldığını görüyoruz. Derin öğrenme ise buna örnek olarak hareket halindeki araçların yangın musluğu ile insan arasındaki farkı anlayabilen görsel bir algı oluşturuyor. Kısacası araçlara insan gibi düşünme yetisi kazandırmamıza olanak sağlıyor.

Yüksek enerji fiziği alanında yapılan keşifler, anlaşılması zor parçacıkların gerçekten keşif niteliği taşıyıp taşımadığını yüksek bir kesinlik düzeyi ile belirlemek için derin öğrenme modellerine ihtiyaç duyuyor.

Yeni Nikel-Titanyum alaşımları üretimi konusunda yapılan araştırmalarda, hangi deneylerin aranan en iyi özellikler ile sonuçlanacağını derin öğrenme teknikleri ile modelleyerek deney masrafları dörtte birine kadar inmesi sağlanıyor(Murphy, 2018).

3.9. Önceki Çalışmalardan Örnekler

Çeşitli biçimlerde görüntü tanıma ve sınıflandırma evrişimli sinir ağları için birincil kullanım alanıdır. CNN görüntü sınıflandırmasının amacı; bir görüntüyü dezenfekte edip, belirgin özelliklerini çıkartmaktır (denetimli makine öğrenmesi sınıflandırma algoritması). Betimlemeyi anahtar kimlik bilgilerine göre azaltır (bu boyut azaltma, denetimsiz makine öğrenme algoritmasının işidir). Bu işlemleri kullanan alanlar; görüntü etiketleme, görsel arama, tavsiye motorlarıdır.

Optik karakter tanıma, diğer bir deyişle OCR, özellikle yazılı sembolleri ve grafikleri ve çizelgeleri işlemek için tasarlanmış bir görüntü tanıma çeşididir. Tıpkı yüz tanıma gibi, hareketli parçalarla daha karmaşık bir işlem içerir. Özünde, OCR bilgisayar görselleştirmesinin doğal dil işleme ile bir kombinasyonudur. İlk olarak, görüntü tanınmakta ve karakterlere ayrılmaktadır; daha sonra karakterler uyumlu bir bütün halinde bir araya getirilerek yapılır. Görüntü etiketleme ve daha iyi indeks-

leme için görüntü içeriğinin diğer açıklamaları CNN kullanır. Optik karakter tanıma el yazısını tanıma, bankacılık ve sigorta gibi hassas hukuk alanlarda yaygın olarak uygulanır.

Sağlık sektörü en ileri teknolojilerin denendiği yaygın endüstrlerden biridir. Tıbbi görüntü hesaplama (İng. medical image computing), muhtemelen görevin karmaşıklığı ve çeşitliliği nedeniyle görüntü tanıma evrişimli sinir ağlarının en heyecan verici uygulamasıdır. Görüntünün içeriğini düz ve dar olarak tanımlamak için görevi az veya çok olan tüketici düzeyindeki görüntü tanımlamasının aksine, tıbbi görüntü, ilk görüntü tanıma işleminden çıkan çok fazla ek veri analizini içerir. Örneğin, tıbbi görüntü sınıflandırma CNN, x-ışını veya MRI görüntülerinde anomalileri, insan gözünden daha yüksek hassasiyetle saptamak için kullanılabilir. Bu tür sistemler, görüntülerin sırasının ve aralarındaki farkların nasıl olduğunu gösterebilir. Tıbbi görüntü sınıflandırma, kamu sağlığı kayıtları ve hastalara özel veri ve test sonuçlarını içeren geniş veritabanlarına dayanır (Bilyk, 2018).

4. VERİ TEMİNİ VE METOT

4.1. Veri Temini

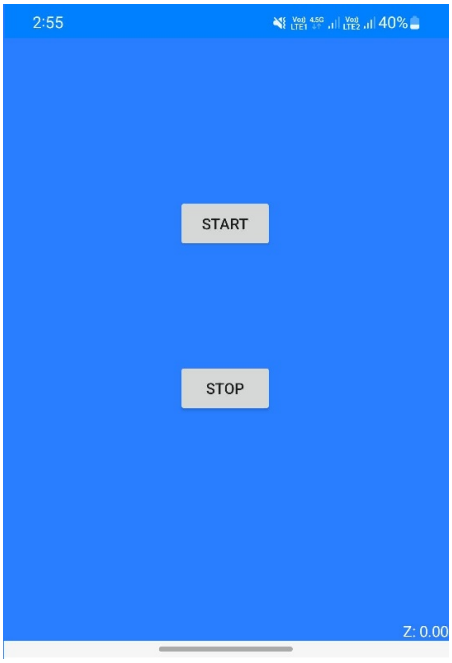
Veriler, iki kasisin bulunduğu düz bir yolda yapılan test sürüşleri sırasında toplanmıştır. Kasislerden geçişler hızlı ve yavaş olarak etiketlenmiştir. Veri toplanması sırasında toplanan deneysel veri tek bir sürücü ile yapılmış olup, araç hızı sabit arttırılarak hızlı ve yavaş olacak şekilde etiketlenmiştir. Aracın hızı saatte 25km'den büyük değerler için hızlı, bu değerden daha düşük hızda geçişler için yavaş olarak kategorize edilmiştir ve hızlarına göre uygun klasörler altında toplanmıştır. Veri, sürülen araç içerisine yerleştirilen bir mobil cihazın sensörleri aracılığı ile toplanmıştır. Şekil 4.2'de sensörlerinden ölçüm alınan cep telefonunun araç içerisinde nasıl konumlandırıldığı gösterilmektedir.

Şekil 4.1: Araç içerisi telefon konumu

Bu çalışmada cep telefonundan veriyi almak için kullanılan yazılım Xamarin mobil uygulama geliştirme platformunda yazılmıştır. Bu platformun tercih edilme sebebi Xamamrin'de mobil uygulamaların hem iOS hem de Android'e yüklenebilmesi ve ayrıca Microsoft teknolojilerini kullanmaya olanak sağlayan bir ortam sunmasıdır.

Proje için geliştirilen yazılımı kullanmak için kullanıcının herhangi bir hesap tanımı veya üyelik işlemi yapması gerekmemektedir. Yani bu çalışmalar sırasında kişisel veriler toplanmamaktadır. Kullanıcı uygulamayı kurduktan sonra kullanım için gerekli izinler istenir ve izinler verildikten sonra kullanıma başlanılabilir. Uygulama iki aşamadan oluşmaktadır. Birincisi; aracın hızını tespit etmek, ikinci işlem ise yeterli hıza ulaştığında aracın ivmelenme haraketlerini toplayarak merkeze iletmektir.

Uygulamanın yazılımsal mimarisinin oluşturulmasında model-view-viewmodel (MVC) teknolojisi kullanılmıştır. Uygulamanın amacı kapsamında uygulamada görsel olarak ayrıntılı bir tasarım kullanılmamıştır. Şekil 4.2'de okunan ivme sensörünün z değeri sağ alt köşede görülmektedir. Start butonuna bastıktan sonra ivme sensöründen alınan veriler, uygulama ekranında gösterilmeye başlanmaktadır. Saatte10 km hıza ulaştıktan sonra, bu veriler kaydedilmektedir. Saatte 10 km hızın altına düşüldüğünde veri kaydı otomatik olarak duraklatılmaktadır.



Şekil 4.2: Mobil uygulamanın kullanıcı arayüzü

Uygulamada işlevsel olarak ilk aşamada cihazın hızını alabilmek için 20 saniyede bir cihazın `geolocation` bilgisi alınarak `location.speed` metodu ile cihazın anlık hızı tespit edilmiştir. Konum sensörüne `System.Threading` kütüphanesinden `CancellationTokenSource` metodu ile token alarak cihazın anlık konum bilgisi alınmaktadır. Böylelikle cihazın hızı tespit edilerek ilk aşama tamamlanmıştır. İkinci ve asıl aşamaya geçmek için aracın anlık hızının 10 km/sn hıza ulasıp ulaşmadığı kontrol edilir. Eğer belirtilen hıza ulaşılmış ise cihazın ivme sensörü okumaya başlanır.

İvme sensöründen gelen verileri okumak için, ivme sensoründe tüm değişiklikeri farketmemize yarayan `OnReadingChanged` adında bir `event` oluşturulmuştur. Bu sensörde x ekseni yataydaki değişimi, y ekseni düşeydeki değişimi verirken; z ekseni ise ekranın ön yüzünden dik çıkan eksene göre derinliği vermektedir. Sensör aktif hale getirildikten sonra x, y, z ve zaman tutan bir modelde veriler topanır. Toplanan veriler 10.000 kayıda ulaştığında, belirlenen sunuculara aktarılması sağlanmıştır. Ayrıca kilit ekranında da çalışabilmesi için uygulama arkaplan servis olarak ayarlanmıştır.

4.2. Veri Setinin Hazırlanması

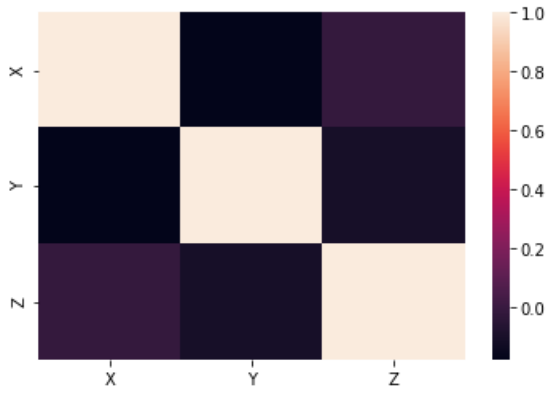
Bölüm 4.1.'de bahsedildiği şekilde toplanan verileri kayıt altında tutabilmek için mobil uygulama tarafında belirlenen sunuculara aktarılan verilerle veri seti hazırlandı. Buradaki amaç verileri depolamak ve sonrasında internet erişimi olan herhangi bir yerden istediğimiz şekilde ulaşıp hazır bir kategorileme sistemi üzerinde verileri izleyebilmekti. Sunucularda depolanan verilerin makine öğrenmesi ile öğrenilebilmesi aşamasına hazır olabilmesi için günlere göre gruplandı. Günlere göre gruplanan bu verileri kullanabilmek ve bir ara katman oluşturmak için her biri ayrı ayrı toplandığı günlerin başlığı altında metin belgelerine kaydedildi. Bu aşama sonucunda araç sürüş esnasındayken mobil uygulama ile almış olduğumuz veriler e-postalar aracılığı ile metin belgelerine dönüştürülerek kullanıma hazır hale getirilmiş oldu.

4.3. Keşfedici Veri Analizi

4.4. Korelasyon Matrisi

Bir korelasyon matrisi, değişken kümeleri arasındaki korelasyon katsayılarını gösteren bir tablodur. Bu matris, değişkenleri gösteren satır ve sütunlardan oluşur. Tablodaki her hücre, korelasyon katsayısını içerir. En sık kullanılan korelasyon katsayılarından biri Pearson korelasyon katsayısıdır.

`Dataframe` kolonları, ivme ölçerden gelen verilerden, yani x, y, z yönlerindeki ivme değişimleri ve bu değişimlerin gözlemlendiği zamanlardan oluşmaktadır. x, y ve z kolonlarının birbirleri ile ilişkisini görebilmek için ilişki matrisini incelediğimizde, aralarında anlamlı bir korelasyon olmadığını gözlemledik. Şekil 4.3'de x, y ve z doğrultularında ivmelenmelerin korelasyon matrisi görülmektedir. Korelasyon değerlerinin çok düşük veya ihmal edilebilir olduğu görülmektedir.



Şekil 4.3: x, y ve z doğrultularında ivmelenmelerin korelasyon matrisi

4.5. Aykırı Değer Analizi

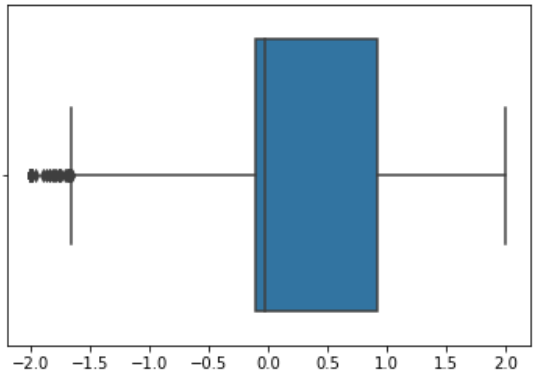
Bölümler arası aralık kuralı (İng., interquartile range rule), aykırı değerlerin varlığını tespit etmede oldukça iyi bir yöntemdir. Aykırı değerler genel düzenin dışında kalan bi-

reysel değerlerdir. Bir veri noktasının gerçekten aykırı olup olmadığını tespit etmede yardımcı olacak bir kuralın olması bizim için oldukça yararlıdır. Herhangi bir veri seti, beş bilgi özeti ile tanımlanabilir. Bu beş bilgiyi şöyle sıralayabiliriz: Minimum, veri kümesinin en düşük değeri. İlk çeyrek Q1: Bu tüm veri listesinin dörtte birini temsil eder. Veri setinin medyanı: Bu tüm veri listesinin orta noktasını temsil eder. Üçüncü çeyrek Q3: Bu tüm veri listesinin dörtte üçünü temsil eder. Maksimum: veri kümesinin en yüksek değeridir.

Bu beş öenmli istatiksel değer verilerimizi özetlemek için kullanılabilir. Örneğin, maksimumdan çıkartılan minimum olan aralık, veri setinin nasıl dağıtılacağının bir göstergesidir. Çeyrekler arası aralık, yayılma aralığına benzer, ancak aykırı değerlere karşı daha az hassastır. Çeyrekler arası aralık, yayılma aralığı ile aynı şekilde hesaplanır. Tek yapmamız gereken, ilk çeyreği üçüncü çeyrekten çıkartmaktır:

$$\text{IQR} = \text{Q3} - \text{Q1}$$

IQR metodu ile yaklaşık olarak 88.528 satırlık veri 75.543'e düşmüştür. Şekil 4.4'da ivme değerlerinin kutu grafiği verilmiştir.



Şekil 4.4: İvme değerlerinin kutu grafiği

4.6. Verilerin Standartlaştırılması

Verileri standartlaştırma işlemini kolaylaştırmak için verileri önce sözlük tipine dönüştürdük. Sonrasında veriler içerisinden sadece kullanacağımız tahmin algoritması için gerekli olan veri bloklarını veri havuzundan geliş sırasını bozmadan çektik.

Bu aşamalardan sonra artık verilerimize kolon ve sıra değerleriile kolay bir şekilde ulaşabilmekteyiz. Seçtiğimiz ve biçimlendirdiğimiz veriler arasından tarih-zaman kolunu seçilerek tipi kontrol edildi. Veri aktarımı mobil uygulamadan mail aracılığı ile olduğu için gelen bütün veriler metin tipindeydi. Verileri olması gerektiği tipe dönüştürmek için tarih-zaman kolonunu zaman formatına ve gelen diğer ivme verilerini sayı formatına dönüştürdük. Böylelikle mobil uygulamanın gönderdiği ivme ölçerden alınan veriler kod ortamına aktarılmış oldu.

Kod ortamına aktarılmış ve tipleri düzenlenmiş olan bütün veriler Python programlama dilinin Pandas kütüphanesinin yardımı ile kütüphane içerisinde bulunan bir matris yapısına aktarıldı. Elimizde bulunan bu dataframe üzerinde tahminleme işlemi yapabilmek için verilerin etiketlenmiş bir özelliğe ihtiyacı vardı. Bu amaç doğrultusunda Pandas kütüphanesinden oluşturulmuş olan dataframe üzerine yeni bir kolon eklenmesi yapılmıştır. Bu yeni kolon veri toplama aşamasında kayıt altına alınmış olan etiketlerle oluşturuldu. Hangi zamanda arabanın nasıl kullanıldığı bilindiği için hangi değerlerin hangi etikete sahip olacağı yeni kolona eklendi. Bu etiketler sücünün hızlı veya yavaş gittiğini belirten etiketlerdir.

4.7. Tahminleme Modelinin Belirlenmesi ve Uygulanması

XGBoost, hız ve performans için tasarlanmış gradyanlı yükseltilmiş karar ağaçlarının bir uygulamasıdır. Açılımı *eXtreme gradient boosting*'dir. Birçok geliştiricinin katkısıyla Tianqi Chen tarafından geliştirilmiştir. Ayrıca, popüler *mxnet* derin öğrenme

kütüphanesinin yaratıcısı olan Dağıtılmış Makine Öğrenme Topluluğu (DMLC) şemsiyesi altında daha geniş bir araç koleksiyonuna sahipttir. Kütüphane hesaplama hızına ve model performansına odaklanır. Modelin uygulanması, düzenlileştirme gibi yeni ilavelerle, bilim, kurgu, öğrenme ve R uygulamalarının özelliklerini de destekler. Degrade güçlendirmenin üç ana formu olan gradient boosting, stochastic gradient boosting ve regularized gradient boosting'leri destekler.

Elimizdeki veri setinin modellenmesi ve bu model üzerinden tahminleme algoritmaları geliştirebilmek için XGBoost'u kullanmayı tercih ettik. Eğitim ve test verilerini kullanarak XGBoost modelinin performansını değerlendirdik. XGBoost modelinin perfromansını k-katlamalı çapraz doğrulama (İng., k-fold cross validation) kullanarak değerlendirdik.

Oluşturduğumuz veri kümesini iki kısıma ayırabiliriz. İlk kısımda algoritmayı eğitir, ikinci kısımda tahminleri elde ederek beklenen sonuçlara karşı tahminleri değerlendirebiliriz. Bölünmenin boyutu, veri kümesinin boyutuna ve özelliklerine bağlı olabilir. Biz kullandığımız veri seti için verilerin %75'ini eğitim, %25'ini ise test verileri olacak şekilde böldük. Bu şekilde algoritma değerlendirme tekniği oldukça hızlıdır. Varsayılan konfigürasyona sahip bir XGBoost modeli eğitim veri seti üzerinden eğitilir ve test veri setinde değerlendirilir. Bizim veri setimizde oluşturduğumuz modelimiz XGBoost defult parametreleri ile çalıştığında test verileri üzerindeki performansını %59,63 olarak gözlemledik.

k-Katlamalı çapraz doğrulama ile XGBoost modellerini değerlendirerek, kullanacağımız modeli daha iyi nasıl eğitebileceğimizi inceledik. Çapraz doğrulama, bir makine öğrenme algoritmasının performansını, eğitim-test seti bölünmesine kıyasla daha az değişkenlikle tahmin etmek için kullanılabilecek bir yaklaşımdır. Veri setini k adet parçaya bölerek çalışır. Verilerin her bölünmesine katlama adı verilir. Algoritma, biri geride tutularak, k-1 tane kat üzerinde eğitilir ve geri tutulan tek katında test edilir. Veri

kümesinin her bir katına testlerde kullanılabilmek için bir şans verilecek şekilde tekrarlanır. Çapraz doğrulama yapıldıktan sonra, ortalama ve standart sapma kullanarak özetleyebileceğimiz k farklı performans puanları elde edersiniz. Bu sonuç, yeni verilerde algoritmanın performansının güvenilir bir tahminini verir. Algoritma farklı veriler üzerinde defalarca eğitilmiş ve değerlendirilmiş olduğundan daha doğrudur. Scikit-learn kütüphanesini kullanarak k-kat çapraz doğrulama işlemini gerçekleştirdik. Ortamalama başarıyı %55,66 ve standart sapmayı %14,20 olarak tespit ettik.

Bir sınıflandırma tipi tahmin modelleme problemi için birçok sınıfınız varsa veya sınıflar dengesizse, çapraz doğrulama gerçekleştirirken tabakalı katlamalar oluşturmak iyi bir fikir olabilir. Burada, çapraz doğrulama değerlendirmesini gerçekleştirirken, tüm eğitim veri kümesinde olduğu gibi her kat için aynı sınıf dağılımını uygulama etkisi vardır. Scikit-learn kütüphanesi, bu yeteneği `StratifiedKFold` sınıfında sağlar. Tabakalı katmanlar oluşturduğumuz zaman ortamalama başarıyı %56,41 ve standart sapmayı %3,33 olarak inceledik.

Tahmin/Gerçek	Pozitif	Negatif
Negatif	0,322	0,257
Pozitif	0,179	0,242

Tablo 4.1: XGBoost modelinden elde edilen karmaşıklık matrisi

XGboost algorimasından elde edilen başarı oranı yetersizliği nedeni ile derin öğrenmenin en temel yöntemlerinden olan yapay sinir ağları (ANN) algoritması kullanılarak başarı oranının arttırılması hedeflenmiştir. Bu çalışmada, çok katmanlı algılayıcı modeli (MLP) olarak bilinen tipik bir sinir ağı modeli kullanılmıştır. MLP, kaynak nöronlardan oluşan bir giriş katmanına, en az bir gizli sinir ağı katmanına ve bir çıkış katmanına sahip, ileri beslemeli bir sinir ağıdır. Giriş katmanı, dış dünyadan gelen giriş sinyallerini kabul eder ve bu sinyalleri gizli katmandaki tüm nöronlara yeniden dağıtır. Gizli katmanlar özellikleri algılar. Gizli katmanlardaki nöronların

ağırlıkları girdi örüntülerindeki özellikleri temsil eder. Çıktı katmanı, tüm ağın çıktı modelini oluşturur. Ağı eğitmek için, ağın problemin temel özelliklerini öğrenebilmesi için ilgili olgunun uygun sayıda temsili örneği seçilmelidir.

Çok fazla farklı öğrenme algoritması mevcuttur, ancak en popüler olanı geri yayılımdır. Bir geri yayılım algoritmasının iki aşaması vardır. İlk olarak, ağ giriş katmanına bir eğitim giriş modeli sunulur. Ağ daha sonra girdi verilerini, çıktı verileri çıktı katmanı tarafından küçük hatalarla oluşturuncayakadar katmandan katmana yayar. Bu model istenen çıktıdan farklıysa, bir hata hesabı yapılır ve ardından ağ üzerinden çıkış katmanından giriş katmanına doğru geriye doğru yayılır. Ağırlıklar, hata yayıldıkça değişir.

ANN modelleri, mimarilerine göre, yani ağ yapısına, transfer fonksiyonuna ve öğrenme kurallarına göre sınıflandırılır. Geri yayılım ağlarında MLP mimarisine genellikle a) gizli katman sayısı ve gizli katmandaki düğüm sayısının çeşitli kombinasyonları, b) farklı transfer fonksiyonları ve c) en uygun ANN modelini seçmek için öğrenme kuralları denenerek karar verilir. Denenen tüm kombinasyonlar arasında en iyi genelleme yeteneğine sahip mimari, yani toplam hatayı minimize eden mimaridir. Eğitim süreci tamamlandıktan ve sinir ağındaki her bir nöronun ağırlıkları ve sapmaları ayarlandıktan sonraki adım, ağın eğitilirken karşılaştığı ve daha önce karşılaşmadığı durumlarda nasıl performans gösterdiğini görerek, eğitim sonuçlarını kontrol etmektir.

Bu çalışmada aktivasyon fonksiyonu olarak relu seçilmiş olup 2, 3 ve 5'li katman sayıları ile 32, 64 ve 128'li nöron sayıları denenerek en iyi sonuç 3 gizli katmanlı mimari kullanılarak alınmıştır. Bu parametreler doğrultusunda %60,28 başarı oranı elde edilmiştir.

Evrişimsel katmanlar sinyalin özelliklerini algılamaktan sorumludur. Bu katman, sinyaldeki düşük ve yüksek seviyeli özellikleri çıkarmak için bazı fitreler uygular. Sinyallerin bazı kısımlarında özellikle genlikte bariz bir değişiklik vardır. Bu değişikliği yakalayabilmek için CNN kullanıldı. Bu sayede her vektörü daha küçük bir özellik vektörüyle

Tahmin/Gerçek	Negatif	Pozitif
Negatif	0,498	0,079
Pozitif	0,318	0,105

Tablo 4.2: ANN karmaşıklık matrisi

ifade etmiş olduk.

Tahmin/Gerçek	Negatif	Pozitif
Negatif	0,465	0,042
Pozitif	0,047	0,446

Tablo 4.3: CNN karmaşıklık matrisi

Dönem (İng., epoch) sayısı, eğitim sırasında tüm eğitim verilerinin ağa gösterilmesi doğruluğun azaldığı yere kadar optimize edildi. Doğrusal olmayan kestirim sınırlarını öğrenmek için öğrenme modeline izin veren modele doğrusal olmayanlığı tanıtmak için relu aktivasyon fonksiyonu kullanıldı. İkili tahminler yaptığımız için çıkış katmanında sigmoid kullanıldı. En iyi skoru veren CNN algoritması modelinde kullanılan parametreler Tablo 4.4'de verilmiştir.

bach size	epochs	optimizer
1024	500	adam

Tablo 4.4: CNN parametre tablosu

CNN ile frekans değişikliklerini yakalayıp tam bağlantılı katmanla özellikleri ileri beslemeyle öğretip sınıflandırdık. Konvolusyonu vektörler yeterince büyük olmadığından çok fazla tekrarlamadık. Tam bağlantılı katmanlarda en az 2 gizli katmanın daha iyi bir sınıflandırma yaptığını farkedip modeli o doğrultuda genişlettik. Eğitim verisinde vektörleri herhangi bir vektörden küçük bir pencereyi vektör içinde birer birer kaydırarak topladık. Bu şekilde modelimiz değişen örüntüyü daha iyi yakalamış oldu.

4.8. Sonuçlar ve Değerlendirme Kriterleri

Bu çalışma üç farklı algoritma kıyaslanarak yapılmıştır. Ağaç tabanlı algoritmalardan biri olan XGBoost, yapay sinir ağları (ANN) ve evrişimli sinir ağları (CNN) algoritmaları uygulanmıştır. Tablo 4.5'de de görüleceği üzere en iyi sınıflandırma test sonuçlarını CNN algoritması ile elde olup doğruluk skoru olarak 91,26 oranı elde edilmiştir. F1-skor, duyarlılık, hassasiyet ve doğruluk skoru metrikleri göz önüne alındığında her dört dönüşüm kriterinde de CNN'in açık ara daha iyi sonuç verdiği gözlemlenmiştir.

Algoritma	Katman	F1-skoru	duyarlılık	hassasiyet	d. skoru
XGBoost	-	57	56	58	56
ANN	2	33,48	24	56,75	60,28
CNN	2	91,18	91,53	91,15	91,26

Tablo 4.5: XGBoost, ANN ve CNN kullanılarak yapılan sınıflandırma çalışmalarının performans metrikleri

5. SONUÇ VE DEĞERLENDİRME

Bu çalışmada, araç sürücüsü davranışının risk analizi ve tahmin problemlerine katkıda bulunmak için araç seyir halinde iken kasislerden geçişlerin sınıflandırılması XGBoost, yapay sinir ağı (ANN) ve evrişimli sinir ağı (CNN) yöntemleri kullanılarak ele alınmaktadır. Bu araçlar arasından ANN modelinin sonuçları, sürücünün kasislerden temkinli veya hızlı geçmesiyle alınan verilerle kıyaslanarak performans metrikleri oluşturulduğunda, gerçek sınıflandırma kayıtlarına çok yakın bir şekilde doğruluğunu kanıtlamıştır, ve diğer yöntemlere göre daha başarılı sonuçlar vermiştir. Önerilen yöntem, kişiselleştirilmiş sigorta poliçelerinin geliştirilmesinde veri sigorta şirketlerinden kullanılabilir.

Bu araştırma ağırlıklı olarak sigorta ve lojistik firmaları için sürücünün aracı nasıl kullandığı ile ilgili bilgi vermeyi amaçlamıştır. Bireysel olarak fiyatlandırma, şirketler için kar marjını artıran bir yol oluşturmaktadır. Bu tezde önerilen yöntemle sakin ve dikkatli sürücüler ile aceleci ve agresif sürücülere ayrı fiyatlandırma yapılmasını, yani ödenmesi gereken gerçek değerlerin müşterilere yansıtılmasınıda kullanılacak yaklaşımlardan biri olan kasisler üzerinde geçişleri gözlemlenerek ele alınmıştır. Derin öğrenme metodlarından olan Convolutional Neural Network (CNN) kullanılarak geliştirilen modelde veri toplama işlemi, araçlara konulan akıllı telefonlardaki mobil uygulama vasıtasıyla yapılarak bakım maliyetlerinin azaltılabileceği veya bu maliyetlerin doğru kişilere yansıtılabileceği gösterilmiştir.

Keşfedici veri analizi ile veriler işlemeye hazır hale getirilirken özellikleri daha verimli kullanmak amaçlanmıştır. Modeldeki başarımı artırmak için batchsize, epoch sayısı, ac-

tivation function ve optimizer kullanımı en iyi değere ulaşana kadar optimize edilmiştir. Tam bağlantılı katmanlarla beraber en az iki gizli katmanın daha iyi bir sınıflandırma yaptığı ortaya konulmuş ve sayede fiyatlandırma hataları en aza indirgenmiştir.

6. EKLER

```
import pandas as pd
import numpy as np
import re
import datetime
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
from sklearn.utils import shuffle

with open('IVME.txt') as f:
    mylist = [line.rstrip('\n') for line in f]

# string ifadeyi sozluk(json) yapisna cevirmek icin kullanilan fonksiyon
  def funk(x):
      string = x

```

```
    #Now removing { and }

    s = string.replace("{" ,"")
    finalstring = s.replace("}" , "")
    lastfinalstring = finalstring.replace("\":", "?")
    lastfinalstring = lastfinalstring.replace("\"","")

    #Splitting the string based on , we get key value pairs
    lista = lastfinalstring.split(",")
    listvalue =[]
    listallvalue = []
    counter = 0
    for i in lista:
        #Get Key Value pairs separately to store in dictionary
        keyvalue = i.split("?")
        if (keyvalue[0]!=''):
            listvalue.append(keyvalue[1])
            counter = counter + 1
            if(counter%4 == 0):
                listallvalue.append(listvalue)
                listvalue =[]

    return listallvalue

# txt donsylarinda baslangic ve bitis [] ile oldugu icin [] leri kaldiran fonksiyon
def kose(a):
    ilk=a.replace("[","")
    ikinci=ilk.replace("]","")
    listallvalue = funk(ikinci)
```

```
53      return listallvalue
54
55  list_main_ivme=kose(mylist[0])
56
57  #Bu fonksiyon Ivme ve/veya Konum listelerinin tarih item ini str den datetime tipine
        ↪ donusturmeye yarar
58  def datetimeConvertor(data):
59      index=0
60      loop=len(data)
61      while index<loop:
62          if isinstance(data[index][3], str):
63              data[index][3]=data[index][3][:10]+" "+data[index][3][11:19]
64              data[index][3]=datetime.strptime(data[index][3],"%Y-%m-%d %H:%M:%S")
65          index=index+1
66
67  datetimeConvertor(list_main_ivme)
68  print("Ivme tipi: ",type(list_main_ivme[0][3]))
69
70  # datetime kisminda yapilicak saat bazli degisiklikler icin(ekleme cikarma)
71  def timeManupulateHr(data,number):
72      index=0
73      loop=len(data)
74      while index<loop:
75          if isinstance(data[index][3], datetime):
76              # hours=9
77              data[index][3]=data[index][3] + timedelta(hours=number)
78          index=index+1
79
80  timeManupulateHr(list_main_ivme,3)
```

```
81
82 df = pd.DataFrame(list_main_ivme, columns = ['X', 'Y','Z','DateTime'])
83
84 df['DateTime'] = pd.to_datetime(df['DateTime'])
85 df['DateTime'] = df['DateTime'].values.astype('<M8[s]')
86
87 df['X'] = pd.to_numeric(df['X'])
88 df['Y'] = pd.to_numeric(df['Y'])
89 df['Z'] = pd.to_numeric(df['Z'])
90
91 df.to_csv(r'ivmeDF.csv',index = None, header=True)
92
93 df_csv = pd.read_csv (r'ivmeDF.csv')
94 df_csv.head()
95
96 """LABEL EKLEME"""
97
98 df_label=df.copy()
99
100 df_label['Label']=''
101
102 df_label = df_label.set_index('DateTime')
103
104 # t1 ve t2 araliginda bulunan verileri dataframe de bulunan Label kolonuna bu aralikdaki butun
       ↪ satirlara 1 olarak atar
105 def AssignLabel(t1,t2,tag):
106     label=df_label.between_time(t1,t2)
107     for i in label.index:
108         #df_label.xs(i)['Label']=0
```

```
109         df_label.at[i, 'Label'] = tag
110 # Belirtien zaman araliklarini parametre olarak AssignLabel fonksiyonuna atar
111 def TimePar(times,tag):
112     for x,y in times:
113         AssignLabel(x,y,tag)
114
115 # Belirtilen zaman araliklari parametre olarak kullanilabilmesi icin tuple tipinde yazildi
116 slowLabel_time_interval=[
117                          ('09:31:39','09:32:00'),
118                          ('09:34:44','09:35:03'),
119                          ('09:38:11','09:38:30'),
120                          ('09:39:27','09:40:26'),
121                          ('09:42:50','09:43:11'),
122                          ('09:43:49','09:44:26'),
123                          ('09:51:18','09:51:41'),
124                          ('09:52:43','09:53:02'),
125                          ('09:58:09','09:58:59'),
126                          ('11:23:38','11:24:34'),
127                          ('11:30:15','11:31:32'),
128                          ('11:35:01','11:36:00'),
129                          ('11:39:29','11:40:49'),
130                          ('11:47:14','11:47:46'),
131                          ('11:48:12','11:49:00'),
132                          ('11:54:30','11:55:49'),
133                          ('12:00:07','12:01:14')
134                         ]
135 fastLabel_time_interval=[('09:32:00','09:33:09'),
136                         ('09:36:27','09:37:09'),
137                         ('09:45:22','09:46:01'),
```

```
                    ('09:47:07','09:47:51'),
                    ('09:48:46','09:49:01'),
                    ('09:49:51','06:50:14'),
                    ('09:54:03','06:54:53'),
                    ('09:55:52','06:56:32'),
                    ('09:59:52','07:00:36'),
                    ('11:17:07','11:18:33'),
                    ('11:19:20','11:20:38'),
                    ('11:21:25','11:22:40'),
                    ('11:25:33','11:26:17'),
                    ('11:27:02','11:27:48'),
                    ('11:28:49','11:29:35'),
                    ('11:32:55','11:33:56'),
                    ('11:37:22','11:38:05'),
                    ('11:41:39','11:42:22'),
                    ('11:45:06','11:45:54'),
                    ('11:50:10','11:51:03'),
                    ('11:52:24','11:53:49'),
                    ('11:56:44','11:57:55'),
                    ('11:58:48','11:59:38')
                   ]

df_label.head()

TimePar(fastLabel_time_interval,0)

TimePar(slowLabel_time_interval,1)

df_label.reset_index(level=0, inplace=True)
```

```
167
168 df_label.head()
169
170 df_del=df_label.copy()
171
172 del df_del['DateTime']
173
174 # # IQR metot outlier temizleme
175 Q1 = df_label.quantile(0.25)
176 Q3 = df_label.quantile(0.75)
177 IQR = Q3 − Q1
178
179 df_hyer= df_del[~((df_del < (Q1 − 1.5 ∗ IQR)) |(df_del > (Q3 + 1.5 ∗ IQR))).any(axis=1)]
180
181 df_hyer.reset_index(inplace=True)
182
183 print("ilk uzunluk ve ikinci uzunluk: ",len(df_label),len(df_hyer))
184
185 counter=0
186 counter2=0
187 counter3=0
188 list_one=[]
189 list_zero=[]
190 #for i in df_label["Label"]:
191 for i in df_hyer["Label"]:
192     if i==1:
193         list_one.append(df_hyer["X"][counter3])
194         list_one.append(df_hyer["Y"][counter3])
195         list_one.append(df_hyer["Z"][counter3])
```

```
196         counter=counter+1
197
198     elif i==0:
199         list_zero.append(df_hyer["X"][counter3])
200         list_zero.append(df_hyer["Y"][counter3])
201         list_zero.append(df_hyer["Z"][counter3])
202         counter2=counter2+1
203     counter3=counter3+1
204 # print('Toplam satir sayisi: ', len(df_label))
205 # print('1 olarak etiketlenen toplam satir sayisi: ',counter)
206 # print('0 olarak etiketlenen toplam satir sayisi: ',counter2)
207 # print('etiketlenmeyen toplam satir sayisi: ',len(df_label)-(counter+counter2))
208 print('Toplam satir sayisi: ', len(df_hyer))
209 print('1 olarak etiketlenen toplam satir sayisi: ',counter)
210 print('0 olarak etiketlenen toplam satir sayisi: ',counter2)
211 print('etiketlenmeyen toplam satir sayisi: ',len(df_hyer)-(counter+counter2))
212
213 list_zero=np.asarray(list_zero)
214 # list_zero=list_zero.reshape(50389,3)
215 list_zero=list_zero.reshape(43770,3)
216
217 list_one=np.asarray(list_one)
218 # list_one=list_one.reshape(35439,3)
219 list_one=list_one.reshape(31773,3)
220
221 print(list_zero.shape)
222 print(list_one.shape)
223
224 X_train=[]
```

```
y_train=[]
list_one/=np.max(list_one)
list_zero/=np.max(list_zero)
for x in range(9980):
    X_train.append(list_one[x:x+20])
    y_train.append(1)
for i in range(9980):
    X_train.append(list_zero[i:i+20])
    y_train.append(0)
X_test=[]
y_test=[]
for x in range(100):
    X_test.append(list_one[x+10000:x+10020])
    y_test.append(1)
for i in range(100):
    X_test.append(list_zero[i+10000:i+10020])
    y_test.append(0)

pd.set_option('display.float_format', lambda x: '%.3f' % x)

X_train=np.asarray(X_train)
y_train=np.asarray(y_train)
X_test=np.asarray(X_test)
y_test=np.asarray(y_test)

y_train.shape

X_train.shape

```

```
X_train=X_train.reshape(-1,20,3,1)
X_test=X_test.reshape(-1,20,3,1)

"""CNN"""

from keras import metrics

batch_size = 1024
num_classes = 1
epochs = 500
model = Sequential()
model.add(Conv2D(32, kernel_size=(2, 2),activation='relu',input_shape=(20,3,1))) # 32 x 2,2
    ↪ matris
model.add(Conv2D(32, kernel_size=(2, 2),activation='relu'))                # 32 x 2,2 matris
model.add(Flatten())
model.add(Dense(64, activation='relu')) #64 noron var
model.add(Dense(77, activation='relu')) #256 noron var
# model.add(Dense(256, activation='relu')) #256 noron var
# model.add(Dense(128, activation='sigmoid')) #128 noron var
# model.add(Dense(64, activation='sigmoid')) #64 noron var
model.add(Dense(num_classes, activation='sigmoid')) # binary classifiation oldugu icin
    ↪ activation fonksiyonu sigmoid secilir
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['acc'])
model_deneme=model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['acc'])
model.fit(X_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1)
model.evaluate(X_test, y_test)

"""XGBoost"""

```

```
X_train, X_test, y_train, y_test = train_test_split(df_hyer[['X','Y','Z']], df_hyer['Label'], stratify
    ↪ =df_hyer['Label'], random_state=1121218)

import xgboost as xgb

xgb_cl = xgb.XGBClassifier(scale_pos_weight=1.25,learning_rate=0.001,n_estimators=500)

from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
import numpy as np
# Fit
xgb_cl.fit(X_train, y_train)

# Predict
preds = xgb_cl.predict(X_test)
preds=preds.astype(int)
y_test=y_test.astype(int)
# Score
confusion_matrix(y_test, preds)

accuracy_score(y_test, preds)

print(classification_report(y_test, preds))

print(confusion_matrix(y_test, preds))

"""ANN"""

X_train, X_test, y_train, y_test = train_test_split(df_hyer[['X','Y','Z']], df_hyer['Label'], stratify
    ↪ =df_hyer['Label'], random_state=1121218)
```

```
308
309 import keras
310 from keras.models import Sequential  # importing Sequential model
311 from keras.layers import Dense       # importing Dense layers
312 from tensorflow import keras
313 from tensorflow.keras import layers
314
315
316 # declaring model
317 basic_model = Sequential()
318
319
320
321
322 model = keras.Sequential([
323     layers.Dense(32,input_shape=(3,), activation="relu"),
324     layers.Dense(64, activation="relu"),
325     layers.Dense(128, activation="relu"),
326     # layers.Dense(256, activation="relu"),
327     # layers.Dense(512, activation="relu"),
328     layers.Dense(1, activation="sigmoid")
329 ])
330
331 model.compile(optimizer="rmsprop",
332               loss="binary_crossentropy",
333               metrics=['acc'])
334
335 # training the model
336 X_train= np.asarray(X_train).astype('float32')
```

```
y_train=np.asarray(y_train).astype('float32')
X_test= np.asarray(X_test).astype('float32')
y_test= np.asarray(y_test).astype('float32')

model.fit(X_train, y_train, epochs=500, batch_size=1024)




# Test, Loss and accuracy
loss_and_metrics = model.evaluate(X_test, y_test)
print('Loss = ',loss_and_metrics[0])
print('Accuracy = ',loss_and_metrics[1])

ynew = model.predict(X_test[:1])

keras.metrics.confusion_matrix(y_test,X_test)
```

Listing 6..1: Python kodları

KAYNAKLAR

Bilyk, V. (2018). *Business applications of convolutional neural networks.* https://theappsolutions.com/blog/development/convolutional-neural-networks/.

Bosari, J. (2013). *What really goes into determining your insurance rates?* https://www.forbes.com/sites/moneywisewomen/2013/01/08/what-really-goes-into-determining-your-insurance-rates/#7cb697513f85.

Cheng Zhang, S. B. K. L. B. H. G. D. A., Mitesh Patel. (2016). Driver classification based on driving behaviors. In *Proceedings of the 21st international conference on intelligent user interfaces.*

Ekonomou, L. (2012). Driver classification identification using artificial intelligence. *Conference: 6th European Computing Conference (ECC '12).*

Foote, K. D. (2017). *A brief history of deep learning.* https://www.dataversity.net/brief-history-deep-learning/.

Heller, M. (2019). *Deep learning explained.* https://www.infoworld.com/article/3397142/deep-learning-explained.html.

Hongyang Zhao, C. C. J. C., Huan Zhou. (2013). Join driving: A smart phone-based driving behavior evaluation system. In *2013 ieee global communications conference (globecom).*

Johan W. Joubert, N. d. K., Dirk de Beer. (2016). Combining accelerometer data and contextual variables to evaluate the risk of driver behaviour. *Science Direct*, *Volume 41*, 80–96.

k-means clustering, demonstration of the standard algorithm. (2018). https://en.wikipedia.org/wiki/K-means_clustering.

Murphy, J. (2018). *Deep learning applications.* https://www.microway.com/hpc-tech-tips/deep-learning-applications/.

Sentiance. (2018). *Driving behavior modeling using smart phone sensor data.* https://sentiance.com/driving-behavior-modeling-using-smart-phone-sensor-data.

Vygandas Vaitkus, G. Z., Paulius Lengvenis. (2014). Driving style classification using long-term accelerometer information. In *2014 19th international conference on methods and models in automation and robotics (mmar).*

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı : Bahadırhan KELEŞ

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Lisans	: KTO Karatay Üniversitesi	Temmuz-2015
Lisans	: Touro Collage	Haziran-2012
Lise	: Meram Konya Lisesi	Haziran-2007

İş Deneyimi

Yıl	Yer	Görev
2013-2013	Türk Telekom	Stajyer Mühendis
2017-2018	ELPO Bilişim Otomasyon	Bilgisayar Mühendisi
2018-2020	YCP Bilgi Teknolojileri	Yazılım Geliştirme Uzmanı
2020-Devam	Setur Marinaları	Yazılım Geliştirme Uzmanı

Yabancı Dil

İngilizce