

BOLU ABANT İZZET BAYSAL ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ



YAPAY ZEKA DESTEKLİ TİTREME ANALİZİ İLE
PARKİNSON HASTALIĞI TEŞHİSİ VE YAŞAM KALİTESİ
İYİLEŞTİRMESİ

Servet GÜNEŞ
Melis DEĞIRMENCI

LİSANS BİTİRME TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

BOLU, 2024

DANIŞMAN ONAY

**Bu tez çalışması Bolu Abant İzzet Baysal Üniversitesi Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü Bitirme Projesi I-II dersleri kapsamında
hazırlanmıştır. Lisans Bitirme Tezi olarak uygun bulunmuştur.**

**Dr. Öğr. Üyesi Ümit ŞENTÜRK
Tez Danışmanı**

Tarih:

İmza:

YAPAY ZEKA DESTEKLİ TİTREME ANALİZİ İLE PARKİNSON HASTALIĞI TEŞHİSİ VE YAŞAM KALİTESİ İYİLEŞTİRMESİ

Servet GÜNEŞ, Melis DEĞİRMENCI

Anahtar Kelimeler: Parkinson Teşhis, Titreme, Yapay Zeka, Derin Öğrenme, Medikal Mobil Uygulama

Özet: Bu çalışma, Parkinson hastalığının erken teşhisini ve yaşam kalitesini iyileştirmeyi amaçlayan yapay zeka destekli titreme analizi üzerine odaklanmaktadır. Mobil uygulama, kullanıcıların el titremesi verilerini kaydederek beslenme, egzersizler, hastane randevuları, Parkinson belirti testi ve ilaç hatırlatma gibi özellikler sunar. Titreme takibi bölümü, ivmeölülerden elde edilen koordinatlar ve frekans verilerini kullanarak bir veri seti oluşturur ve yapay zeka algoritmalarıyla eğitilir. Bu eğitim sonucunda, kullanıcının Parkinson hastalığı olma olasılığı hakkında bilgi verilir. Çalışmanın amacı, erken teşhis ve tedavi süreçlerine katkıda bulunarak Parkinson hastalarının yaşam kalitesini artırmaktır. Yapılan çalışmada, yapay zeka destekli titreme analizinin Parkinson hastalığının teşhisinde etkili bir araç olduğu ve mobil uygulamanın hastaların yaşam kalitesini artırmada önemli bir rol oynadığı bulgulanmıştır. Bu bulgular, Parkinson hastalığının yönetiminde yeni ve yenilikçi yaklaşımların geliştirilmesine katkı sağlayabilir.

**DIAGNOSIS OF PARKINSON'S DISEASE AND IMPROVEMENT OF
QUALITY OF LIFE WITH ARTIFICIAL INTELLIGENCE SUPPORTED
VIBRATION ANALYSIS**

Şerbet GÜNEŞ, Melis DEĞİRMENCI

Key Words: Parkinson's Diagnosis, Tremor, Artificial Intelligence, Deep Learning, Medical Mobile Application

Abstract: This study focuses on AI-assisted tremor analysis, which aims to improve the early diagnosis and quality of life of Parkinson's disease. The mobile app offers features such as nutrition, exercises, hospital appointments, Parkinson's symptom testing, and medication reminder by recording users' hand tremor data. The tremor tracking section creates a dataset using coordinates and frequency data obtained from the accelerometer and is trained with artificial intelligence algorithms. As a result of this training, information is given about the possibility of the user having Parkinson's disease. The aim of the study is to improve the quality of life of Parkinson's patients by contributing to early diagnosis and treatment processes. In the study, it was found that artificial intelligence-supported tremor analysis is an effective tool in the diagnosis of Parkinson's disease and that the mobile application plays an important role in improving the quality of life of patients. These findings may contribute to the development of new and innovative approaches to the management of Parkinson's disease.

ÖNSÖZ ve TEŞEKKÜR

Bu tez, Parkinson hastalığının erken teşhisini ve yönetimini geliştirmeyi amaçlayan bir araştırmmanın ürünüdür. Parkinson hastalığı, yaşam kalitesini önemli ölçüde etkileyen kronik bir nörolojik rahatsızlıktır. Bu çalışmada, yapay zeka destekli titreme analiziyle elde edilen verilerin kullanılarak mobil uygulama aracılığıyla hastaların yaşam kalitesini artırma potansiyeli incelenmiştir.

Bu süreçte, danışman hocamız Ümit Şentürk'ün değerli rehberliği ve desteği bizim için büyük bir motivasyon kaynağı olmuştur. Ayrıca, ailemizin ve arkadaşlarımızın sabır, anlayış ve moral desteği de çalışmanın başarılı bir şekilde tamamlanmasına katkı sağlamıştır.

Bu çalışmanın, Parkinson hastalığıyla mücadelede yeni yöntemlerin geliştirilmesine ve hastaların yaşam kalitesinin iyileştirilmesine katkı sağlamasını umuyoruz.

Son olarak, bu tezi inceleyen ve üzerinde çalışan herkese teşekkürlerimizi sunarız. Umarız ki bu çalışma, bu alanda ilerlemeye katkı sağlar ve daha fazla insanın yaşam kalitesini artırır.

İÇİNDEKİLER

ÖZET	ii
ABSTRACT	iii
ÖNSÖZ ve TEŞEKKÜR	iv
İÇİNDEKİLER	v
RESİMLER DİZİNİ	viii
BÖLÜM 1 GİRİŞ	1
1.1 LİTERATÜR ÖZETİ	1
1.2 PROJE AMACI	6
BÖLÜM 2 MOBİL UYGULAMA KISIMLARI	7
2.1 MOBİL UYGULAMADA KAYIT OL	7
2.1.1 KAYIT OL AKTIVITESI KOD AÇIKLAMASI (SIGNUPACTIVITY)	8
2.2 MOBİL UYGULAMADA PROFIL DÜZENLEME	9
2.2.1 PROFIL DÜZENLEME AKTIVITESI KOD AÇIKLAMASI	10
(EDITPROFILE)	
2.3 MOBİL UYGULAMADA ANASAYFA	12
2.3.1 ANASAYFA KOD AÇIKLAMASI	13
2.4 MOBİL UYGULAMADA HAREKET TAKIBI VE ANALIZI	15
2.4.1 VERİTABANI OLUŞTURMA VE İŞLEME	15
2.4.2 YAPAY ZEKA MODELİNİN EĞİTİMİ	15
2.4.3 UYGULAMANIN ENTEGRASYONU	15
2.4.4 HAREKET TAKIBI VE ANALİZ BÖLÜMÜ KOD AÇIKLAMASI	17
2.5 MOBİL UYGULAMADA İLAÇ HATIRLATICI	21
2.5.1 İLAÇ HATIRLATICI AKTIVITESI KOD AÇIKLAMASI	22
2.6 MOBİL UYGULAMADA EGZERSİZ ÖNERILERİ	27
2.6.1 UYGULAMA YAPISI VE İŞLEVSELLİĞİ	27
2.6.2 ANA AKTIVITE (EXERCISEACTIVITY)	27
2.6.3 ANA AKTIVITE KOD AÇIKLAMASI	28

2.6.4 EGZERSIZ DETAYLARI AKTIVITESI.....	30
(SHOULDEREXERCISESACTIVITY)	
2.6.5 EGZERSIZ DETAYLARI AKTIVITESI KOD AÇIKLAMASI.....	31
2.6.6 LISTVIEW VE ÖZEL ADAPTÖR (EXERCISEADAPTER).....	32
2.7 MOBİL UYGULAMADA BESLENME ÖNERILERİ.....	33
2.7.1 UYGULAMA YAPISI VE İŞLEVSELLİĞİ.....	33
2.7.2 ANA AKTIVITE (PARKINSONVEBESLENMEACTIVITY).....	33
2.7.3 ANA AKTIVITE KOD AÇIKLAMASI.....	34
2.7.4 ÖRNEK DIYET LISTESİ AKTIVITESI (SAMPLEDIETSACTIVITY).....	35
2.7.5 ÖRNEK DIYET LISTESİ AKTIVITESI KOD AÇIKLAMASI.....	37
2.8 MOBİL UYGULAMADA PARKINSON BELIRTİ TESTİ.....	38
2.8.1 QUESTIONACTIVITY SINIFI.....	38
2.8.2 QUESTIONACTIVITY SINIFI KOD AÇIKLAMASI.....	39
2.8.3 RESULTACTIVITY SINIFI.....	41
2.8.4 RESULT ACTIVITY SINIFI KOD AÇIKLAMASI.....	43
BÖLÜM 3 YAPAY ZEKA	44
3.1 GEREKLİ KÜTÜPHANELERİN YÜKLENMESİ.....	44
3.2 GEREKLİ PYTHON KÜTÜPHANELERİ VE İŞLEVLERİ.....	44
3.3 VERİ SETİNİN YÜKLENMESİ VE TEMEL İNCELEME.....	45
3.4 VERİ HAZIRLIĞI.....	46
3.4.1 VERİ SETİNİN YAPISI VE ÖZELLİKLERİ.....	46
3.4.2 VERİ SETİNDEKİ EKSİK DEĞERLER.....	47
3.4.3 VERİLERDEN ÖZELLİK SEÇİMİ.....	48
3.4.4 VERİLERİN ÖZET İSTATİSTİKLERİ.....	49
3.4.5 SÜTUN İSİMLERİ.....	50
3.4.6 SINIF DAĞILIMI.....	50
3.5 VERİ BÖLÜMLEME.....	51
3.6 VERİ ÖN İŞLEME VE ÖLÇEKLENDİRME.....	52
3.7 DERİN ÖĞRENME MODELİ OLUŞTURMA.....	52

3.8 MODEL DERLEME.....	53
3.9 MODEL EĞİTİMİ.....	54
3.10 EĞİTİM SÜRECİNİN GÖRSELLEŞTİRİLMESİ.....	55
3.11 MODELİN KAYDEDİLMESİ.....	57
3.12 MODELİN YÜKLENMESİ.....	57
3.13 TENSORFLOW LİTE FORMATINA DÖNÜŞTÜRME.....	57
3.14 TENSORFLOW LİTE MODELİNİN KAYDEDİLMESİ.....	58
3.15 EĞİTİLEN MODELİN ANDROID STUDIO ENTEGRASYONU.....	58
3.15.1 MODELLOADER.....	59
3.15.2 TENSORFLOW LİTE MODELİNİN EKLENMESİ.....	60
3.15.3 ANDROID STUDIO PROJESİNE ENTEGRASYON.....	60
3.15.4 MODELİN YÜKLENMESİ.....	60
3.15.5 GİRİŞ VE ÇIKIŞ VERİLERİNİN İŞLENMESİ.....	61
3.15.6 MODELİN KULLANILMASI.....	61
3.15.7 PERFORMANS VE UYUM TESTLERİ.....	61
KAYNAKLAR.....	62
ÖZGEÇMİŞ	63

RESİMLER DİZİNİ

Resim 2.1: Kayıt Ol Sayfası	7
Resim 2.1.1: Kayıt Ol Sayfasının Kod Bloğu	8
Resim 2.1.2: Kayıt Ol Sayfasının Kod Bloğu Devamı	9
Resim 2.2: Profil Düzenleme Sayfası	10
Resim 2.2.1: Profil Düzenle Sayfasının Kod Bloğu	11
Resim 2.2.2: Profil Düzenle Sayfasının Kod Bloğu Devamı	11
Resim 2.3: Anasayfa	12
Resim 2.3.1: Anasayfa Kod Bloğu	13
Resim 2.3.2: Anasayfa Kod Bloğu Devamı	14
Resim 2.3.3: Anasayfa Kod Bloğu Devamı	14
Resim 2.4: Hareket Takibi ve Analizi Sayfası	16
Resim 2.4.1: Hareket Takibi ve Analizi Kod Bloğu	17
Resim 2.4.2: Hareket Takibi ve Analizi Kod Bloğu Devamı	18
Resim 2.4.3: Hareket Takibi ve Analizi Kod Bloğu Devamı	18
Resim 2.4.4: Hareket Takibi ve Analizi Kod Bloğu Devamı	19
Resim 2.4.5: Hareket Takibi ve Analizi Kod Bloğu Devamı	19
Resim 2.4.6: Hareket Takibi ve Analizi Kod Bloğu Devamı	20
Resim 2.4.7: Hareket Takibi ve Analizi Kod Bloğu Devamı	20
Resim 2.5: İlaç Hatırlatıcı Sayfası	22
Resim 2.5.1: İlaç Hatırlatıcı Kod Bloğu	24
Resim 2.5.2: İlaç Hatırlatıcı Kod Bloğu Devamı	24
Resim 2.5.3: İlaç Hatırlatıcı Kod Bloğu Devamı	25
Resim 2.5.4: İlaç Hatırlatıcı Kod Bloğu Devamı	25
Resim 2.5.5: İlaç Hatırlatıcı Kod Bloğu Devamı	26
Resim 2.5.6: İlaç Hatırlatıcı Kod Bloğu Devamı	26
Resim 2.6: Egzersiz Önerileri Ana Aktivite Sayfası	28
Resim 2.6.1: ExerciseActivity Kod Bloğu	29

Resim 2.6.2: Egzersiz Detay Sayfası	30
Resim 2.6.3: Egzersiz Detay Aktivitesi Kod Bloğu	31
Resim 2.6.4: Egzersiz Detay Aktivitesi Kod Bloğu Devamı	32
Resim 2.6.5: Egzersiz Detay Aktivitesi Kod Bloğu Devamı	32
Resim 2.7: Beslenme Önerileri Sayfası	34
Resim 2.7.1: Beslenme Önerileri Sayfası Devamı	34
Resim 2.7.2: Beslenme Aktivitesinin Kod Açıklaması	35
Resim 2.7.3: Örnek Diyet Listesi Sayfası	36
Resim 2.7.4: Örnek Diyet Listesi Sayfası Devamı	36
Resim 2.7.5: Diyet Listeleri Aktivite Kodunun Bir Parçası	37
Resim 2.8: Parkinson Belirti Testinden Bir Soru	39
Resim 2.8.1: Test Aktivitesi Kod Bloğu	40
Resim 2.8.2: Test Aktivitesi Kod Bloğu Devamı	40
Resim 2.8.3: Test Aktivitesi Kod Bloğu Devamı	41
Resim 2.8.4: Test Sonucu	42
Resim 2.8.5: Farklı Bir Test Sonucu	42
Resim 2.8.6: Sonuç Aktivitesi Kod Bloğu	43
Resim 3.1: Veri Setinin Yapısı	46
Resim 3.2: Veri Setinin Genel Yapısı	47
Resim 3.3: Veri Setindeki Eksik Değerler	48
Resim 3.4: Sayısal Sütunların İstatiksel Özeti	49
Resim 3.5: Veri Setindeki Sütunların İsimleri	50
Resim 3.6: Status Değerlerinin Tekrarlanması Sayısı	51
Resim 3.7: Modelin Eğitim Süreci	56
Resim 3.8: Modelin Kayıp Değerleri	56
Resim 3.9: Modelin Doğruluk Değerleri	56
Resim 3.10: Modelin Yükleniği ModelLoader Kod Bloğu	59

BÖLÜM 1

GİRİŞ

1.1 LİTERATÜR ÖZETİ

Parkinson hastalığının erken teşhisi ve etkili yönetimi, akustik özelliklerin kullanıldığı derin öğrenme yöntemleri gibi yeni teknolojilerle desteklenen çalışmalarla giderek önem kazanmaktadır. Mehmet Bilal ER (2021), konuşma sinyallerinin akustik özelliklerini kullanarak derin öğrenme tabanlı bir yöntem önermektedir. Çalışmada, akustik özelliklerin belirlenmesi için genetik algoritma ve ReliefF özellik seçme algoritması kullanılmıştır. Bu özellikler, Evrişimsel Sinir Ağı (ESA) mimarisi içerisinde kullanılarak Parkinson hastalığının sınıflandırılması amaçlanmıştır. Yapılan deneylerde, özellik seçimi yapıldığında %94,23'lük bir doğruluk elde edilmiştir.

Bir başka araştırma ise Parkinson hastalığının erken teşhisinde akıllı saatlerin kullanımının potansiyelini araştırmaktadır. Cardiff Üniversitesi bünyesindeki İngiltere Demans Araştırma Enstitüsünden araştırmacılar tarafından yapılan bir çalışmada, akıllı saatlerin, Parkinson'un erken teşhisinde rol oynayabileceği tespit edilmiştir. 103 bin 712 kişinin verilerinin incelendiği çalışmada, Parkinson hastalarının hareketlerinin sağlıklı bireylere göre daha yavaş olduğu belirlenmiştir.

Tsviya Fay-Karmon ve ark. (2024), Parkinson hastalığının motor semptomlarının değerlendirilmesinde akıllı saatlerin kullanılmasını incelemiştir. Ev tabanlı izleme sistemi kullanılarak yapılan çalışmada, kişiselleştirilmiş motor dalgalanmaların profillerini tasvir etmek ve bireysel farklılıklarını yakalamak amaçlanmıştır. Yapılan değerlendirmelerde, hastaların veriye dayalı dört motor dalgalanma profili arasında bölündüğü gözlemlenmiştir.

Son olarak, Mine Boz ve Yeliz Dursun (2023), ivme sensörü kullanarak titreme tespiti için bir sistem geliştirmiştir. Bu sistem, Edge Impulse yazılımı ve Arduino Nano 33 BLE mikrodenetleyicisi ile entegre edilmiştir. Yapılan çalışmada, önerilen sistemin %85 tanıma doğruluğu sağladığı bulunmuştur.

Bu çalışmalar, Parkinson hastalığının teşhis ve yönetiminde yeni teknolojilerin ve akıllı cihazların önemli bir potansiyeli olduğunu göstermektedir.

Parkinson Hastalığının belirlenmesinde önemli bir odak noktası, fonasyon bozukluğu olan Dysphonia'dır (Ho, Iansek, Marigliani, Bradshaw, ve Gates, 1999). Dysphonic kişilerin sesleri genellikle boğuk, kısık, sert, hırıltılı ve zor çıkar. Bu belirtiler, temel frekans salınımları kullanılarak ölçülebilir ve Parkinson hastalarına ait seslerin akustik ölçümü, hastaya zarar vermeden kolayca yapılabilir. Seslerin akustik ölçümü için çeşitli yöntemler önerilmiştir, bunlar arasında; temel frekanstaki kısa süreli düzensizlikler, jitter, shimmer gibi akustik ölçümelerin yanı sıra ses kısıklığının ölçümü için harmonik gürültü oranı parametresi kullanılmaktadır. Ayrıca, Mel frekansı kepstrum katsayıları, doğrusal öngörü modelleme, işitsel modelleme ve kepstral tepe önem ölçümleri gibi yöntemler de sesin akustik ölçümelerinde kullanılmıştır.

Little ve ark. (2007), ses bozukluklarını belirlemeye geçerli bir metot olarak doğrusal olmayan dinamiklerin analizini önermektedir. Bu çalışmada, SVM sınıflandırıcı ve süzgeç temelli ilişkisiz ölçme (STİÖ) kullanılarak öznitelikler seçilmiş ve Parkinson hastalığı verisetindeki hastalar ve normal kişiler başarıyla sınıflandırılmıştır.

Diger bir arastirmada, Ene (2008), olasılıksal sinir ağını (PNN) Parkinson hastalığının teşhisinde sınıflandırıcı olarak kullanmıştır. Sakar ve Kursun (2010) çalışmasında, BBDB doğrulama yöntemi ve karşılıklı bilgi temelli bir öznitelik seçme yöntemi (MRMR) kullanılmıştır. Resul (2010) çalışmasında, bir sinir ağı temelli sınıflandırıcı (NN) Parkinson hastalarını ayırmak için kullanılmış ve %92,9 doğru sınıflandırma başarımı elde edilmiştir.

Çağlar ve ark. (2010) çalışmasında, dilsel kuvvetli adaptif sinir-bulanık sınıflayıcı (ANFC+LH) kullanılarak %94.72 test başarı oranı elde edilmiştir. Polat (2012) çalışmasında, bulanık c-ortalamalı (FCM) öznitelik ağırlıklandırma yöntemi önerilmiş ve değişik k değerleri için %97.93'e varan başarı oranları elde edilmiştir.

Bu çalışmalar, Parkinson Hastalığının teşhisini için çeşitli sınıflandırma ve tanıma tekniklerinin kullanılabilirliğini göstermektedir.

Şule Yücelbaş ve Cüneyt Yücelbaş (2019), İstanbul Üniversitesi Cerrahpaşa Tıp Fakültesi Nöroloji Bölümünde kaydedilmiş 188 Parkinson hasta ve 64 sağlıklı kişiden elde edilen ses sinyallerini temel alarak Parkinson Hastalığının otomatik teşhisini üzerine bir araştırma gerçekleştirmiştir. Ses sinyallerine Ayarlanabilir Q-faktör Dalgacık Dönüşümü (AQDD) metodu uygulanarak elde edilen özellikler boyut indirgeme yöntemleri olan temel bileşen analizi (TBA) ve bunun çeşitleri olan kernel TBA (KTBA) ile olasılıksal TBA (OTBA) uygulanmıştır. OTBA yöntemi %87.56 doğruluk orANIyla en başarılı boyut indirgeme yöntemi olarak belirlenmiştir ve ROC ve PRC alan değerleri yaklaşık 0.95 bandına ulaşarak hasta ve sağlıklı sınıf ayrışımının mükemmelle yaklaştığını göstermiştir.

Hasan Badem (2019) tarafından gerçekleştirilen araştırmada ise Parkinson Hastalığının ses sinyalleri üzerinden sınıflandırılması için KYK, ROS, DVM, NB ve KA gibi makine öğrenmesi teknikleri kullanılmıştır. Bu araştırmada PDC veri seti kullanılmıştır ki bu veri seti, Parkinson hastalığının sınıflandırılması için yüksek boyutlu öznitelik ve örneklemeye içermektedir. Yapılan deneysel çalışmalarında, oldukça yüksek doğruluk değerleri elde edilmiş ve kullanılan yöntemler istatistiksel olarak karşılaştırılmıştır. Ayrıca, boyut indirme tekniklerinin başarına etkileri de analiz edilmiştir.

Bu çalışmalar, Parkinson Hastalığının teşhisi için ses sinyallerinin kullanılmasının etkili bir yöntem olabileceğini ve bu yöntemlerin gerçek yaşam uygulamalarına uygun olduğunu göstermektedir.

Sadullah Esmer, Muhammed Kürşad Uçar, İbrahim Çil ve Mehmet Recep Bozkurt (2020) tarafından gerçekleştirilen “Parkinson Hastalığı Teşhisi İçin Makine Öğrenmesi Tabanlı Yeni Bir Yöntem” araştırmasında, doktorun kararına destek olabilmesi amacıyla 252 bireyden elde edilen ses verileriyle Parkinson Hastalığının teşhis edilmesi hedeflenmiştir. Verilerin daha iyi sonuçlar vermesi için çeşitli ön işlemler uygulanmıştır, bunlar arasında veri dengeleme işlemi ve sistematik örneklemeye metodu ile işleme alınacak verilerin belirlenmesi bulunmaktadır. Öznitelik seçme algoritması kullanılarak özniteliklerin etkisi hesaplanmış ve bazı veri grupları oluşturulmuştur. Karar ağacı, Destek Vektör Makineleri ve K En Yakın Komşu Algoritması gibi sınıflandırma algoritmaları kullanılarak performans değerlendirmesi yapılmış ve en yüksek performans değerine sahip veri grubu belirlenerek bir model oluşturulmuştur. Bu model, PH olma ihtimali olan bireylerin ses kayıtlarından oluşturulan veri seti ve uygulanan model yardımıyla doktora tıbbi karar destek sağlayabileceği sonucuna varılmıştır.

2018-2019 yıllarında, Parkinson Hastalığı teşhisi için makine öğrenmesi tabanlı birçok yeni sistem geliştirilmiştir. Bunlardan en güncel yaklaşım, derin öğrenme tabanlı Parkinson Hastalığı şiddetini tahmin eden bir modeldir. Tensorflow makine öğrenme kütüphanesi kullanılarak geliştirilen bu sistem, UCI Machine Learning Repository'den alınan verilerle %62-81 doğruluk oranıyla çalışmaktadır. Ayrıca, Sadek ve arkadaşlarının (2019) yapay sinir ağları tabanlı bir çalışmasında %93 başarı oranıyla Parkinson Hastalığının tespit edilebildiği görülmüştür, bu çalışmada 31 bireyden elde edilen 195 ölçüm kullanılmıştır.

B.Karan ve arkadaşları, 45 kişiden alınan 150 ölçümle destek vektör makineleri ve rastgele orman algoritması kullanarak bir model kurmuşlardır. Ancak, küçük boyuttaki bu veri setleriyle kurulan model, eğitim ve test verileri birlikte değerlendirildiğinde makine öğrenme yönünde ezberleme eğiliminde olmuş ve taraflı kararlar vermeye başlamıştır. Veri seti eğitim ve test setlerine uygun şekilde bölündüğünde, önerilen modelin doğruluğu etkili bir şekilde azalmıştır. Bu nedenle, daha büyük veri setlerinin kullanılması ve veri setinde dengeleme işlemi yapılması gibi yöntemlerle daha uygun sonuçlar elde edilmesi gerekmektedir.

Muhammed Erdem İsenkul (2011) tarafından gerçekleştirilen “Parkinson Hastalığı’nın Teşhisi İçin Veri Toplama Ve Örüntü Tanıma Sistemi” yüksek lisans tezinde, hastaların sosyal bilgileri ve ses kayıtları gibi bilgileri kayıt ve analiz edebilen bir sistem geliştirilmiştir. Tüm hastaların tıbbi muayeneleri sonrasında sosyal bilgileri de toplanmış ve her bir hasta için ayrı bir ses kayıt veritabanı oluşturulmuştur. Parkinson hastalarından alınan verilerin doğruluğunu anlayabilmek için, kontrol grubu olarak Parkinson hastaları olmayan veya tamamen sağlıklı bireylerden oluşan bir veri bankası da oluşturulmuştur. Toplanan veriler, Destek Vektör Makineleri (SVM) ve En Yakın Komşu (KNN) algoritmaları kullanılarak analiz edilmiştir. Ayrıca, öznitelik seçiminde Minimum Artıklık-Maksimum İlişki (MRMR) yöntemi kullanılmıştır.

Elde edilen sonuçlar, doğru ses bilgileri ve kayıtlarının seçilerek oluşturulan ses bilgileri sayesinde, hasta ve sağlıklı bireylerin sadece ses kayıtlarının incelenerek hastalık teşhisi ve hastalık gelişim takibinin yapılabileceğini göstermiştir. Bu çalışma, doktorlara karar destek aşamasında yardımcı olabilecek bir sistem geliştirmiştir.

1.2 PROJE AMACI

"Yapay Zeka Destekli Titreme Analizi ile Parkinson Hastalığı Teşhisi ve Yaşam Kalitesi İyileştirmesi" adlı bu çalışma, Parkinson hastalığının erken teşhisi ve etkili yönetimi için yenilikçi bir yaklaşım sunmaktadır. Projemiz, kullanıcıların günlük yaşamlarında telefonlarını kullanırken el titremelerini kaydetmelerine ve bu verileri bir mobil uygulama aracılığıyla analiz etmelerine olanak tanır. Mobil uygulama, titreme takibi, beslenme önerileri, egzersiz programları, hastane randevuları, Parkinson belirti testi ve ilaç hatırlatma gibi çeşitli özellikler sunar.

Titreme takibi bölümü, ivmeölçer tarafından elde edilen koordinat ve frekans verilerini kullanarak bir veri seti oluşturur. Bu veriler, yapay zeka algoritmalarıyla eğitilerek Parkinson hastalığı riskini belirlemeye yönelik bir modele dönüştürülür. Kullanıcılar, bu model aracılığıyla Parkinson hastalığı olma olasılıklarını öğrenebilir ve gerekiğinde tıbbi yardım alabilirler.

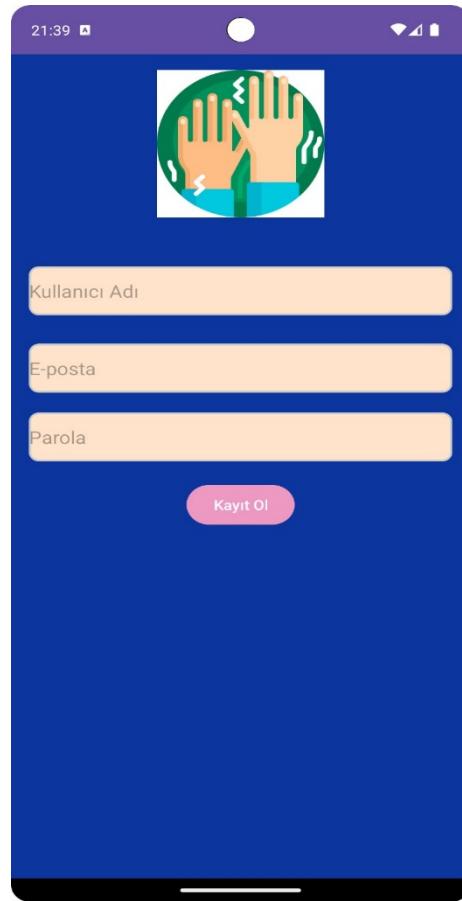
Bu çalışma, Parkinson hastalığının erken teşhisini sağlayarak hastaların yaşam kalitesini artırmayı hedeflemektedir. Yapay zeka destekli titreme analizi, kullanıcıların daha bilinçli sağlık kararları almasını ve hastalıkla daha etkin bir şekilde mücadele etmesini sağlayarak yaşam kalitesini iyileştirmeyi amaçlamaktadır. Ayrıca, mobil uygulama aracılığıyla sunulan çeşitli özellikler, hastaların yaşamlarını daha kolay ve daha yönetilebilir hale getirerek günlük yaşam kalitesini artırabilir."

BÖLÜM 2

MOBİL UYGULAMA KİSİMLARI

2.1 Mobil Uygulamada Kayıt Ol

İlk aşamada, kullanıcı anasayfaya erişmek için kayıt olma sürecini tamamlamalıdır. Kayıt sayfası, kullanıcı adı, e-posta adresi ve şifre gibi temel bilgilerin girilmesine imkan tanır. Kullanıcı, bu bilgileri sağladıkten sonra "Kayıt Ol" butonuna tıklar ve sistem tarafından sunulan doğrulama mekanizmaları aracılığıyla hesabını oluşturur.



2.1: Kayıt ol Sayfası

2.1.1 Kayıt Ol Aktivitesi Kod Açıklaması (SignupActivity)

SignupActivity sınıfı, kullanıcının uygulamaya kaydolmasını ve profil bilgilerini düzenlemesini sağlayan bir Android aktivitesidir. Aktivite, kullanıcının giriş bilgilerini (username, email, password) alır. Bu bilgiler, uygulamanın önceki oturumlarını hatırlamak için SharedPreferences üzerinde saklanır. Kullanıcı tüm gerekli alanları doldurduktan sonra, kayıt butonuna tıklayarak kayıt işlemini tamamlar ve anasayfaya yönlendirilir. Ansayfadaki çıkış yap butonuna tıklanmadığı sürece kayıt ol sayfası tekrar açılmaz. Bu işlevsellik, kullanıcı dostu arayüzüne katkıda bulunarak, kullanıcının uygulamayı daha kolay ve etkili bir şekilde kullanmasını sağlar.



```
© signup.java ×
1 package com.example.bitirme;
2
3 > import ...
16 <public class signup extends AppCompatActivity {
17     2 usages
18     private static final int PICK_IMAGE = 100;
19     3 usages
20     private Uri selectedImageUri;
21
22     2 usages
23     EditText usernameEditText, emailEditText, passwordEditText;
24     3 usages
25     ImageView profileImageView;
26     2 usages
27     Button registerButton;
28
29     @SuppressLint("MissingInflatedId")
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33
34         SharedPreferences sharedPreferences = getSharedPreferences( name: "MyPrefs", Context.MODE_PRIVATE );
35         String username = sharedPreferences.getString( key: "username", defValue: "" );
36         String email = sharedPreferences.getString( key: "email", defValue: "" );
37         String password = sharedPreferences.getString( key: "password", defValue: "" );
38
39         if (!username.isEmpty() && !email.isEmpty() && !password.isEmpty()) {
40             // Kullanıcı bilgileri mevcut, otomatik giriş yap
41             Intent intent = new Intent( packageContext.signup.this, anasayfa.class );
42             startActivity(intent);
43             finish(); // Kayıt ekranını kapat
44         }
45     }
46 }
```

2.1.1: Kayıt Ol Sayfasının Kod Bloğu

```

② signup.java ×

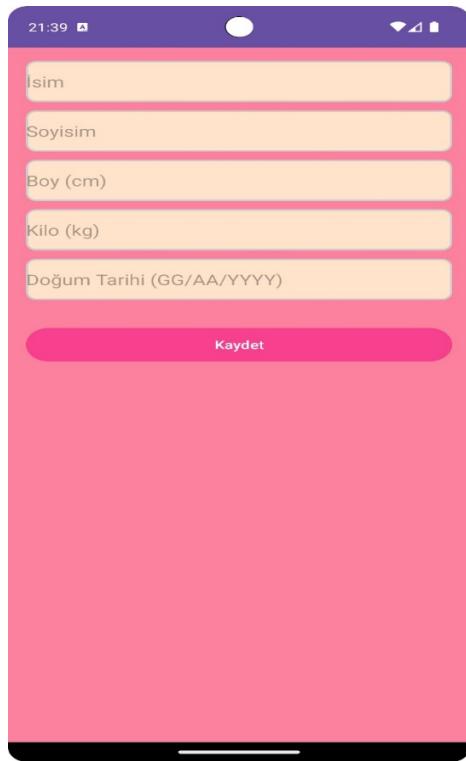
40    setContentView(R.layout.activity_signup);
41    usernameEditText = findViewById(R.id.editTextUsername);
42    emailEditText = findViewById(R.id.editTextEmail);
43    passwordEditText = findViewById(R.id.editTextPassword);
44    profileImageView = findViewById(R.id.profileImageView);
45    registerButton = findViewById(R.id.buttonRegister);
46    profileImageView.setOnClickListener(v -> openGallery());
47    registerButton.setOnClickListener(v -> {
48        String newUsername = usernameEditText.getText().toString(); // username değişkenini newUsername olarak değiştirdik
49        String newEmail = emailEditText.getText().toString();
50        String newPassword = passwordEditText.getText().toString();
51        if (newUsername.isEmpty() || newEmail.isEmpty() || newPassword.isEmpty()) {
52            Toast.makeText(context, R.string.signup_error_all_fields, Toast.LENGTH_SHORT).show();
53        } else {
54            SharedPreferences.Editor editor = sharedPreferences.edit();
55            editor.putString("username", newUsername); // Burada da newUsername kullanılacak
56            editor.putString("email", newEmail);
57            editor.putString("password", newPassword);
58            if (selectedImageUri != null) {
59                editor.putString("profileImageUri", selectedImageUri.toString());
60            }
61            editor.apply();
62            Toast.makeText(context, R.string.signup_success, Toast.LENGTH_SHORT).show();
63            // Kayıt işlemi başarılı olduğunda anasayfaya geçiş yap
64            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
65            startActivity(intent);
66            finish(); // Kayıt ekranınızı kapat
67        }
68    });
69 });
70

```

2.1.2: Kayıt Ol Sayfasının Kod Bloğu Devamı

2.2 Mobil Uygulamada Profil Düzenleme

Kayıt işlemi tamamlandıktan sonra, kullanıcı profil bilgilerini düzenleme işlevselligine erişebilir. Profil düzenleme ekranı, kullanıcının kişisel bilgilerini (isim, soyisim, boy, kilo, doğum tarihi vb.) güncellemesine olanak tanır. Kullanıcı, bu bilgileri doldurduktan sonra değişiklikleri kaydedebilir ve profil sayfasına dönerek güncellenmiş bilgilerini görebilir.



2.2: Profil Düzenleme Sayfası

2.2.1 Profil Düzenleme Aktivitesi Kod Açıklaması (EditProfile)

EditProfile sınıfı, kullanıcının ad, soyad, boy, kilo ve doğum tarihi gibi profil bilgilerini düzenlemesine olanak tanır. Aktivite, kullanıcidan alınan bu bilgileri SharedPreferences kullanarak kalıcı olarak depolar. Kullanıcı tüm bilgileri girdikten sonra "Kaydet" butonuna tıkladığında, girilen bilgiler kontrol edilir ve eksik bir alan varsa kullanıcıya uyarı verilir. Eğer tüm alanlar doldurulmuşsa, bilgiler depolanır ve kullanıcıya bilgi mesajı gösterilir. Ardından, kullanıcı anasayfaya yönlendirilir ve bu aktivite sonlandırılır. Bu işlevsellik, kullanıcı dostu arayüzüne katkıda bulunarak, kullanıcının kişisel bilgilerini güncellemesini ve uygulama içinde tutmasını sağlar.

```

@ editProfile.java ×
1 package com.example.bitirme; // editProfile.java
2
3 > import ...
13
14 public class editProfile extends AppCompatActivity {
15     2 usages
15     private EditText nameEditText, surnameEditText, heightEditText, weightEditText, birthDateEditText;
16     2 usages
16     private Button saveButton;
17
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_edit_profile);
21         nameEditText = findViewById(R.id.nameEditText);
22         surnameEditText = findViewById(R.id.surnameEditText);
23         heightEditText = findViewById(R.id.heightEditText);
24         weightEditText = findViewById(R.id.weightEditText);
25         birthDateEditText = findViewById(R.id.birthDateEditText);
26         saveButton = findViewById(R.id.saveButton);
27         saveButton.setOnClickListener(new View.OnClickListener() {
28             @Override
29             public void onClick(View v) {
30                 String name = nameEditText.getText().toString().trim();
31                 String surname = surnameEditText.getText().toString().trim();
32                 String heightStr = heightEditText.getText().toString().trim();
33                 String weightStr = weightEditText.getText().toString().trim();
34                 String birthDate = birthDateEditText.getText().toString().trim();
35                 if (name.isEmpty() || surname.isEmpty() || heightStr.isEmpty() || weightStr.isEmpty() || birthDate.isEmpty()) {
36                     // Eksik bilgi varsa kullaniciya uyarı ver
37                     Toast.makeText(context: editProfile.this, text: "Lütfen tüm bilgileri girin.", Toast.LENGTH_SHORT).show();
38                 }
39             }
}

```

2.2.1: Profil Düzenle Sayfasının Kod Bloğu

```

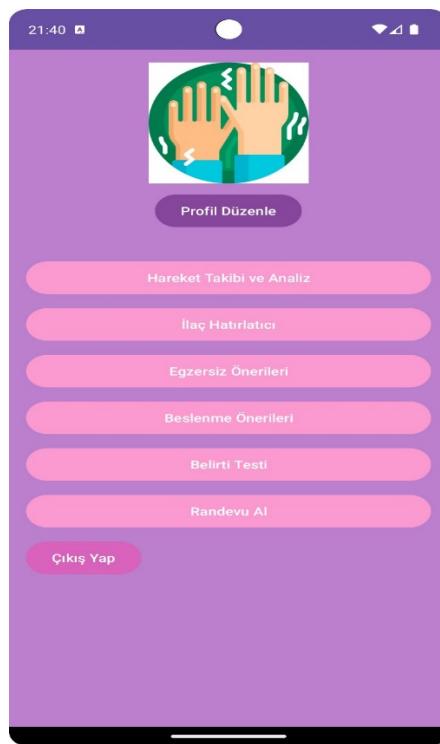
40         double height = Double.parseDouble(heightStr);
41         double weight = Double.parseDouble(weightStr);
42         SharedPreferences sharedpreferences = getSharedPreferences( name: "MyPrefs", Context.MODE_PRIVATE);
43         SharedPreferences.Editor editor = sharedpreferences.edit();
44         editor.putString("name", name);
45         editor.putString("surname", surname);
46         editor.putString("height", heightStr);
47         editor.putString("weight", weightStr);
48         editor.putString("birthDate", birthDate);
49         editor.apply();
50         Toast.makeText( context: editProfile.this, text: "Profil bilgileriniz kaydedildi.", Toast.LENGTH_SHORT).show();
51         Intent intent = new Intent( packageContext: editProfile.this, anasayfa.class);
52         startActivity(intent);
53         finish();
54     }
55 }
56 }
57 }

```

2.2.2: Profil Düzenle Sayfasının Kod Bloğu Devamı

2.3 Mobil Uygulamada Anasayfa

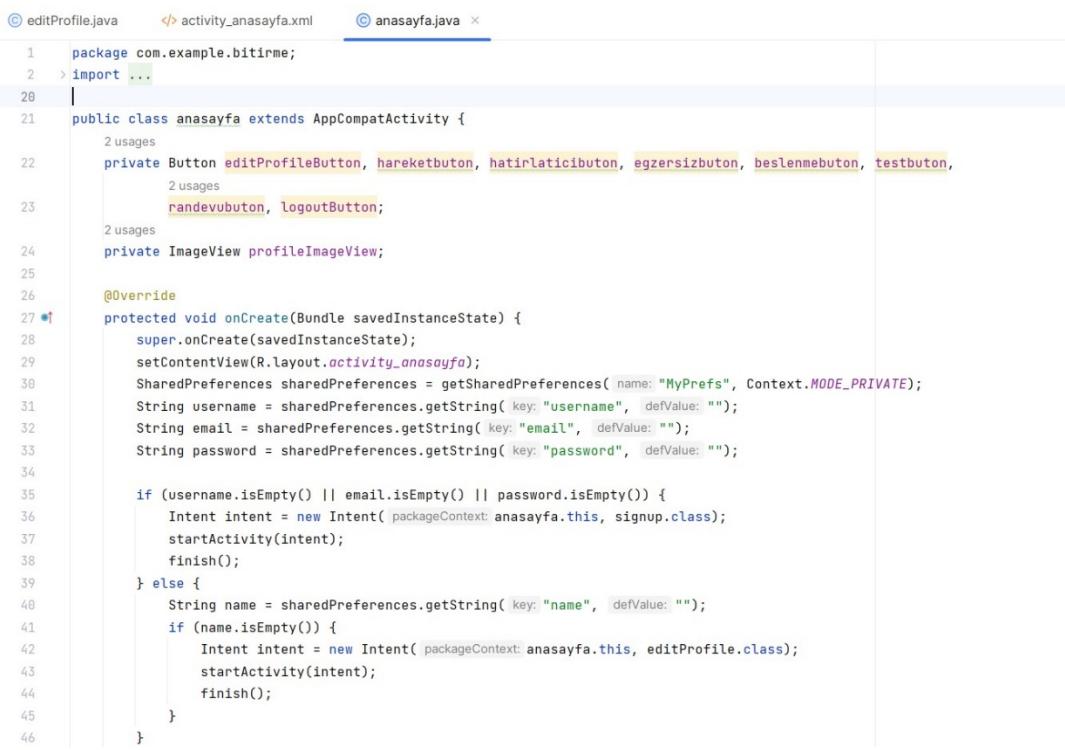
Uygulamanın ana ekranı, kullanıcının sağlık ve sağlıklı yaşam ile ilgili çeşitli işlevleri erişmesini sağlayan merkezi bir noktadır. Üst kısmında uygulamanın logosu bulunur ve hemen altında, kullanıcının kolay erişim sağlayabileceği dokuz buton yer alır. Profil düzenleme butonu, kullanıcının kişisel bilgilerini güncellemesine olanak tanır ve bu işlevsellik, uygulamanın kullanıcı odaklı yapısını vurgular. Hareket takibi ve analiz, ilaç hatırlatıcı, egzersiz önerileri, beslenme önerileri ve belirti testi butonları, kullanıcının sağlık durumunu yönetmesine yardımcı olur ve uygulamanın sağlık odaklı misyonunu destekler. Randevu al butonu, kullanıcıyı MHRŞ randevu sistemine yönlendirerek, tıbbi hizmetlere daha kolay erişim sağlar. Çıkış yap butonu, kullanıcının oturumu kapatmasını ve uygulamadan çıkışmasını sağlar, ancak oturum kapatılmadığı sürece uygulama her açıldığında doğrudan ana ekrana yönlendirilir. Bu tasarım, uygulamanın kullanıcı deneyimini optimize ederken, sağlık ve sağlıklı yaşam hizmetlerine kolay ve etkili erişim sağlamayı hedefler.



Resim 2.3: Anasayfa

2.3.1 Anasayfa Kod Açıklaması

Uygulamanın ana ekranını oluşturan anasayfa sınıfını içerir. anasayfa sınıfı, kullanıcının uygulamaya erişimini yönetir ve çeşitli işlevlere erişim sağlar. İlk olarak, kullanıcının oturum bilgileri kontrol edilir ve eksik bilgi varsa kullanıcı kayıt ol ekranına yönlendirilir. Oturum bilgileri mevcutsa, profil bilgileri kontrol edilir ve eksik bilgi varsa profil düzenleme ekranına yönlendirilir. Ana ekranda, profil düzenleme, hareket takibi, ilaç hatırlatıcı, egzersiz önerileri, beslenme önerileri, belirti testi, randevu alma ve çıkış yapma gibi işlevlere erişim sağlayan butonlar bulunur. Butonlardan herhangi birine tıklanması durumunda ilgili işlevsellik sağlanır ve kullanıcı ilgili ekrana yönlendirilir. Çıkış yap butonuna tıklanması durumunda kullanıcı oturumu kapatılır ve giriş ekranına yönlendirilir.



```
1 package com.example.bitirme;
2 > import ...
20 |
21 public class anasayfa extends AppCompatActivity {
22     2 usages
23     private Button editProfileButton, hareketbuton, hatirlaticibuton, egzersizbuton, beslenmebuton, testbuton,
24         2 usages
25         randevubuton, logoutButton;
26     2 usages
27     private ImageView profileImageView;
28
29     @Override
30     protected void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         setContentView(R.layout.activity_anasayfa);
33         SharedPreferences sharedPreferences = getSharedPreferences( name: "MyPrefs", Context.MODE_PRIVATE );
34         String username = sharedPreferences.getString( key: "username", defValue: "" );
35         String email = sharedPreferences.getString( key: "email", defValue: "" );
36         String password = sharedPreferences.getString( key: "password", defValue: "" );
37
38         if (username.isEmpty() || email.isEmpty() || password.isEmpty()) {
39             Intent intent = new Intent( packageContext: anasayfa.this, signup.class );
40             startActivityForResult(intent);
41             finish();
42         } else {
43             String name = sharedPreferences.getString( key: "name", defValue: "" );
44             if (name.isEmpty()) {
45                 Intent intent = new Intent( packageContext: anasayfa.this, editProfile.class );
46                 startActivityForResult(intent);
47             }
48         }
49     }
50 }
```

Resim 2.3.1: Anasayfa Kod Bloğu

```

    48     editProfileButton = findViewById(R.id.editProfileButton);
    49     hareketbutton = findViewById(R.id.hareketbutton);
    50     hatirlaticibutton = findViewById(R.id.hatirlaticibutton);
    51     egzersizbutton = findViewById(R.id.egzersizbutton);
    52     beslenmebutton = findViewById(R.id.beslenmebutton);
    53     testbutton = findViewById(R.id.testbutton);
    54     randevubutton = findViewById(R.id.randevubutton);
    55     logoutButton = findViewById(R.id.logoutButton);
    56     profileImageView = findViewById(R.id.profileImageView);
    57     loadProfileImage();
    58     editProfileButton.setOnClickListener(new View.OnClickListener() {
    59         @Override
    60         public void onClick(View v) {
    61             // Profil düzenleme sayfasına git
    62             Intent intent = new Intent(getApplicationContext(), editProfile.class);
    63             startActivity(intent);
    64         }
    65     });
    66     hareketbutton.setOnClickListener(new View.OnClickListener() {
    67         @Override
    68         public void onClick(View v) {
    69             Intent intent = new Intent(getApplicationContext(), accelerometer.class);
    70             startActivity(intent);
    71         }
    72     });
    73     hatirlaticibutton.setOnClickListener(new View.OnClickListener() {
    74         @Override
    75         public void onClick(View v) {
    76             Intent intent = new Intent(getApplicationContext(), hatirlaticici.class);
    77         }
    78     });

```

Resim 2.3.2: Anasayfa Kod Bloğu Devamı

```

    79     egzersizbutton.setOnClickListener(new View.OnClickListener() {
    80         @Override
    81         public void onClick(View v) {
    82             Intent intent = new Intent(getApplicationContext(), ExerciseActivity.class);
    83             startActivity(intent);
    84         }
    85     });
    86     beslenmebutton.setOnClickListener(new View.OnClickListener() {
    87         @Override
    88         public void onClick(View v) {
    89             Intent intent = new Intent(getApplicationContext(), beslenme.class);
    90             startActivity(intent);
    91         }
    92     });
    93     testbutton.setOnClickListener(new View.OnClickListener() {
    94         @Override
    95         public void onClick(View v) {
    96             Intent intent = new Intent(getApplicationContext(), test.class);
    97             startActivity(intent);
    98         }
    99     });
    100    randevubutton.setOnClickListener(new View.OnClickListener() {
    101        @Override
    102        public void onClick(View v) {
    103            String url = "https://www.mhrs.gov.tr";
    104            Intent intent = new Intent(Intent.ACTION_VIEW);
    105            intent.setData(Uri.parse(url));
    106            startActivity(intent);
    107        }
    108    });
    109    logoutButton.setOnClickListener(v -> {
    110        SharedPreferences.Editor editor = sharedPreferences.edit();
    111        editor.clear();
    112        editor.apply();
    113        Intent intent = new Intent(getApplicationContext(), signup.class);
    114        startActivity(intent);
    115        finish();
    116    });

```

Resim 2.3.3: Anasayfa Kod Bloğu Devamı

2.4 Mobil Uygulamada Hareket Takibi ve Analizi

Hareket takibi ve analiz bölümü, projenin temel taşlarından birini oluşturur. Bu aşama, el titreme frekansı ve verilerine bağlı olarak parkinson hasta olma olasılığını belirlemeye odaklanmıştır. Burada, kişinin kendi takip edebileceği ve bu verilerin analiz edilerek sağlık durumu hakkında bilgi edinebileceği bir süreç geliştirilmiştir.

2.4.1 Veritabanı Oluşturma Ve İşleme

İvmeölçer verilerinin toplanmasıyla başlayan bu aşamada, farklı kişilerden bu bölümde kullanılması istenip x, y ve z koordinat değişimleri kaydedilmiştir. Ayrıca, el titremesi frekansı da belirlenmiş ve kaydedilmiştir. Bu veriler ile bir veritabanı oluşturulmuştur.

2.4.2 Yapay Zeka Modelinin Eğitimi

TensorFlow gibi derin öğrenme kütüphanelerinden yararlanılarak geliştirilen yapay zeka modeli, el titremesi verilerini analiz etmek için eğitilmiştir. Bu süreçte, oluşturulan veritabanı kullanılarak model optimize edilmiş ve el titremesi belirtilerini doğru bir şekilde tanımlayabilmesi için eğitilmiştir.

2.4.3 Uygulamanın Entegrasyonu

Yapay zeka modelinin eğitimi tamamlandıktan sonra, bu model uygulamaya entegre edilmiştir. Entegrasyon sürecinde, kullanıcı dostu bir arayüz geliştirilmiş ve kullanıcının kolayca el titremesi verilerini analiz edebilmesini sağlamıştır.

Kullanıcı, uygulamayı açtığında, hareket takibi ve analizi bölümüne erişebilir. Bu bölümde, kullanıcı el titremesi verilerini kaydetmek için bir "Veri Kaydet"

düğmesine basar. Bu düğmeye basıldığında, telefonun ivmeölçeri x, y ve z koordinat değişimlerini kaydeder ve aynı zamanda el titremesi frekansını belirler. Zaman bilgisini de yazdırır. Kullanıcı, istedikleri kadar veri kaydedebilir. Ayrıca bu verileri txt dosyası olarak telefonlarına indirilir ve istedikleri zaman açıp bakabilir.

Veri kayıt işlemi tamamlandıktan sonra, kullanıcı "Sonuç" düğmesine basarak analiz sonucunu görebilir. Uygulama, entegre edilen yapay zeka modelini kullanarak kaydedilen verileri analiz eder ve el titremesi belirtilerini değerlendirir. Sonuç, kullanıcıya ekranda gösterilir ve "Parkinson hastası olma olasılığınız yüksek" veya "Parkinson hastalığı olma olasılığınız düşük" gibi bir mesaj ile sunulur.

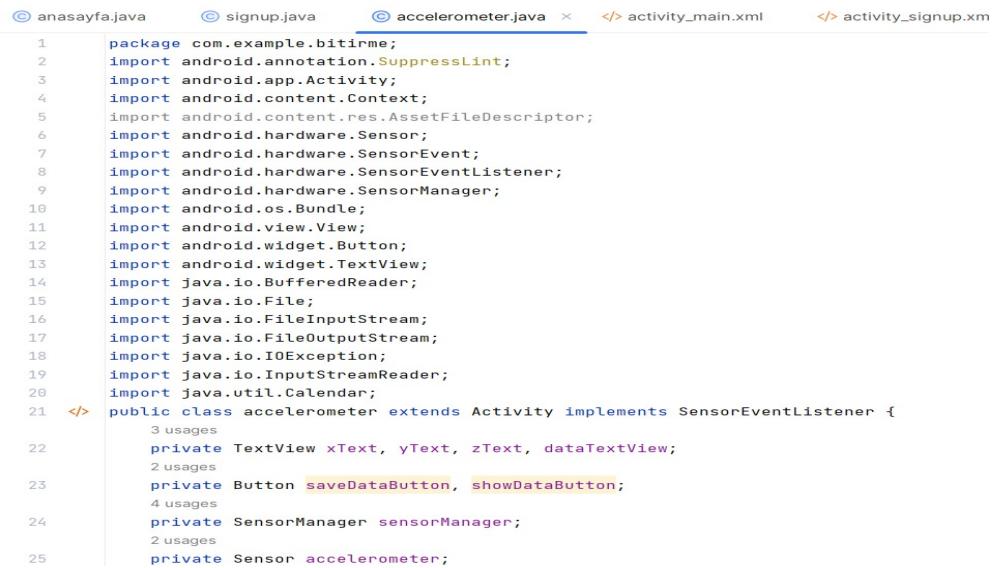
Bu entegrasyon süreci, kullanıcıların kolayca el titremesi verilerini analiz etmelerini sağlar ve sağlık durumları hakkında bilgi edinmelerine yardımcı olur.



Resim 2.4: Hareket Takibi ve Analizi Sayfası

2.4.4 Hareket Takibi Ve Analiz Bölümü Kod Açıklaması

Kullanıcının telefonundan gelen ivmeölçer verilerini kaydederek başlar. Bu veriler, x, y ve z koordinat değişimlerini içerir ve ayrıca zaman damgası ve frekans bilgisini hesaplar. Kaydedilen veriler, txt dosyasına yazılır ve daha sonra analiz için kullanılır. Projede kullanılan TensorFlow Lite modeli, uygulama içinde yüklenir ve el titremesi verilerini analiz ederek Parkinson hastalığı olasılığını tahmin eder. Model, eğitim verisiyle önceden eğitilmiş ve mobil cihazlarda hızlı ve etkili bir şekilde çalışacak şekilde optimize edilmiştir. Kullanıcı, veri kaydetme ve kaydedilen verileri görüntüleme işlevlerini kullanabilir. Veri kaydetme işlevi, kullanıcının ivmeölçer verilerini txt dosyasına kaydetmesini sağlar. Kaydedilen veriler, daha sonra kullanıcı tarafından görüntülenebilir ve analiz edilebilir. Uygulama, kaydedilen verileri analiz ederek Parkinson hastalığı olasılığını tahmin eder. Bu tahmin, TensorFlow Lite modeli tarafından yapılır ve sonuçlar kullanıcıya ekranda gösterilir. Kullanıcıya, el titremesi belirtilerine dayanarak Parkinson hastalığı olma olasılığı hakkında bilgi verilir.



```
1 package com.example.bitirme;
2 import android.annotation.SuppressLint;
3 import android.app.Activity;
4 import android.content.Context;
5 import android.content.res.AssetFileDescriptor;
6 import android.hardware.Sensor;
7 import android.hardware.SensorEvent;
8 import android.hardware.SensorEventListener;
9 import android.hardware.SensorManager;
10 import android.os.Bundle;
11 import android.view.View;
12 import android.widget.Button;
13 import android.widget.TextView;
14 import java.io.BufferedReader;
15 import java.io.File;
16 import java.io.FileInputStream;
17 import java.io.FileOutputStream;
18 import java.io.IOException;
19 import java.io.InputStreamReader;
20 import java.util.Calendar;
21 </> public class accelerometer extends Activity implements SensorEventListener {
    3 usages
22     private TextView xText, yText, zText, dataTextView;
    2 usages
23     private Button saveDataButton, showDataButton;
    4 usages
24     private SensorManager sensorManager;
    2 usages
25     private Sensor accelerometer;
```

Resim 2.4.1: Hareket Takibi ve Analizi Kod Bloğu

```

26     private File file;
27     3 usages
28     private long lastSaveTimeStamp = 0;
29     no usages
30     private final long INTERVAL = 30000; // 30 saniyede bir veri kaydet
31     // TensorFlow Lite modeli
32     2 usages
33     private ModelLoader modelLoader;
34     @SuppressLint("MissingInflatedId")
35     @Override
36     protected void onCreate(Bundle savedInstanceState) {
37         super.onCreate(savedInstanceState);
38         setContentView(R.layout.activity_main);
39         // Create sensor manager
40         sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
41         // Accelerometer sensor
42         accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
43         // Assign TextViews
44         xText = findViewById(R.id.xText);
45         yText = findViewById(R.id.yText);
46         zText = findViewById(R.id.zText);
47         dataTextView = findViewById(R.id.dataTextView);
48         saveDataButton = findViewById(R.id.saveDataButton);
49         showDataButton = findViewById(R.id.showDataButton);
        // Dosya yolu oluştur
        File externalDir = getExternalFilesDir(type: null);
        file = new File(externalDir, child: "ivme_verileri.txt");
    }

```

Resim 2.4.2: Hareket Takibi ve Analizi Kod Bloğu Devamı

```

50     // TensorFlow Lite modelini yükle
51     String modelPath = "model.tflite";
52     this.modelLoader = new ModelLoader(modelPath);
53     // Save data button onClick listener
54     saveDataButton.setOnClickListener(new View.OnClickListener() {
55         @Override
56         public void onClick(View v) {
57             // Verileri kaydet
58             saveDataToFile();
59         }
60     });
61     // Show data button onClick listener
62     showDataButton.setOnClickListener(new View.OnClickListener() {
63         @Override
64         public void onClick(View v) {
65             // Kaydedilen verileri göster
66             showSavedData();
67             predictParkinson();
68         }
69     });
70 }
71
72
73     @Override
74     protected void onResume() {
75         super.onResume();
76         // Register accelerometer sensor listener
77         sensorManager.registerListener(listener: this, accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
}

```

Resim 2.4.3: Hareket Takibi ve Analizi Kod Bloğu Devamı

```

 78     @Override
 79     protected void onPause() {
 80         super.onPause();
 81         // Unregister accelerometer sensor listener
 82         sensorManager.unregisterListener(this);
 83     }
 84     @Override
 85     public void onAccuracyChanged(Sensor sensor, int accuracy) {
 86     }
 87     @Override
 88     public void onSensorChanged(SensorEvent event) {
 89         // Get accelerometer values
 90         float xValue = event.values[0];
 91         float yValue = event.values[1];
 92         float zValue = event.values[2];
 93         // Update TextViews
 94         xText.setText("X: " + xValue);
 95         yText.setText("Y: " + yValue);
 96         zText.setText("Z: " + zValue);
 97     }

```

1 usage

Resim 2.4.4: Hareket Takibi ve Analizi Kod Bloğu Devamı

```

 98     private void saveToFile() {
 99         // Veriyi dosyaya kaydet
100         Calendar cal = Calendar.getInstance();
101         int year = cal.get(Calendar.YEAR);
102         int month = cal.get(Calendar.MONTH) + 1;
103         int day = cal.get(Calendar.DAY_OF_MONTH);
104         int hour = cal.get(Calendar.HOUR_OF_DAY);
105         int minute = cal.get(Calendar.MINUTE);
106         int second = cal.get(Calendar.SECOND);
107         int millis = cal.get(Calendar.MILLISECOND);
108         // Calculate frequency
109         long currentTimestamp = System.currentTimeMillis();
110         double frequency = 0.0;
111         if (lastSaveTimestamp != 0) {
112             long elapsedTime = currentTimestamp - lastSaveTimestamp;
113             if (elapsedTime > 0) {
114                 frequency = 1000.0 / elapsedTime;
115             }
116         }
117         // Veriyi dosyaya yaz
118         try {
119             FileOutputStream fos = new FileOutputStream(file, append: true);
120             String data = "Time: " + year + "/" + month + "/" + day + " " + hour + ":" + minute + ":" + second + "." + millis +
121                     ", X: " + xText.getText().toString().split( regex: ":" )[1] +
122                     ", Y: " + yText.getText().toString().split( regex: ":" )[1] +
123                     ", Z: " + zText.getText().toString().split( regex: ":" )[1] +
124                     ", Frequency: " + frequency + " Hz\n";
125             fos.write(data.getBytes());
126             fos.close();
127             lastSaveTimestamp = currentTimestamp; // Update last save time stamp
128         } catch (IOException e) {
129             e.printStackTrace();

```

Resim 2.4.5: Hareket Takibi ve Analizi Kod Bloğu Devamı

```

private void showSavedData() {
    // Kaydedilen verileri dosyadan oku ve TextView'e ekle
    try {
        FileInputStream fis = new FileInputStream(file);
        InputStreamReader isr = new InputStreamReader(fis);
        BufferedReader reader = new BufferedReader(isr);

        StringBuilder stringBuilder = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            stringBuilder.append(line).append("\n");
        }
        dataTextView.setText(stringBuilder.toString());
        fis.close();
        isr.close();
        reader.close();

    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

Resim 2.4.6: Hareket Takibi ve Analizi Kod Bloğu Devamı

```

private void predictParkinson() {
    // Frekansı al
    String lastLine = dataTextView.getText().toString().trim();
    double frequency = Double.parseDouble(lastLine.substring(lastLine.lastIndexOf("Frequency: ") + 11, lastLine.lastIndexOf(" Hz")));
    // TensorFlow Lite modelini belirle
    float[][] input = new float[1][1];
    input[0] = new float[]{(float) frequency};
    // TensorFlow Lite modelini çalıştır
    float[][] output = new float[1][1];
    modelLoader.runModel(input);
    // Tahmin sonucunu al
    float prediction = output[0][0];
    // Tahmin sonucuna göre mesaj oluştur
    String message;
    if (prediction < 0.5) {
        message = "Parkinson hastası olma olasılığı düşük.";
    } else {
        message = "Parkinson hastası olma olasılığı yüksek.";
    }
    dataTextView.setText(message);
}

```

Resim 2.4.7: Hareket Takibi ve Analizi Kod Bloğu Devamı

2.5 Mobil Uygulamada İlaç Hatırlatıcı

Bu kısım, kullanıcının ilaçları ve alınma zamanlarını kaydedebilecekleri, ardından hatırlatıcılarla ilaç alımlarını yönetmelerine olanak tanıyan bir özellik içeriyor. Bu özellik, kullanıcının ilaçlarını düzenli olarak alabilmelerini sağlayarak sağlık yönetimine de katkıda bulunur ve bu süreci kolaylaştırır.

·İlaç Ekleme:

İlacın adını girmek için bir metin kutusu ve ilaçın alınma saati bir saat seçme aracı bulunur. Kullanıcı, ilaçın alınma saati için uygun bir saat seçtikten sonra "Hatırlatıcı Oluştur" butonuna tıklayarak ilacı ekleyebilir.

·İlaç Listesi:

Eklenen her ilaç, ekranda bir liste şeklinde görüntülenir. Listede her ilaçın adı ve belirlenen alınma saati bulunur. Ayrıca, her ilaçın yanında bir "Sil" butonu bulunur. Bu sayede kullanıcı istediği zaman ilaçları listeden silebilir.

·Hatırlatıcı Oluşturma:

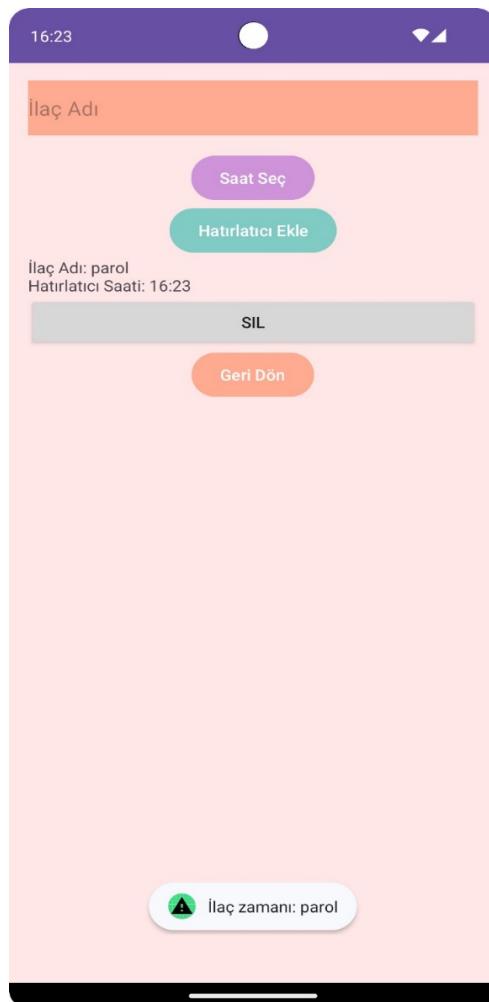
İlaç ekledikten sonra, belirlenen alınma saatinde kullanıcıya bir hatırlatıcı gönderilir. Hatırlatıcıda ilaçın adı ve alınma saati yer alır. Bu hatırlatıcılar, kullanıcının ilaç alımını hatırlamasına ve zamanında almasına yardımcı olur.

·Uyarılar:

İlaç alınma saati geldiğinde, kullanıcıya bir uyarı gönderilir. Uyarı, "İlaç Zamanı: [İlaç Adı]" şeklinde ekranda görüntülenir. Bu sayede kullanıcı ilacı almayı unutmaz.

·Geri Dön Butonu:

Kullanıcı, herhangi bir zamanda ana ekran'a geri dönmek istediginde "Geri Dön" butonunu kullanabilir. Bu buton, kullanıcının kolayca ana ekran'a dönmesini sağlar ve kullanımını daha da kolaylaştırır.



Resim 2.5: İlaç Hatırlatıcı Sayfası

2.5.1 İlaç Hatırlatıcı Aktivitesi Kod Açıklaması

ilacAdiEditText, saatSecButton, hatirlaticiEkleButton, geriDonButton ve ilaclarLayout gibi arayüz öğeleri, ilgili XML dosyasındaki bileşenlerle ilişkilendirilir. Bu, kullanıcı arayüzü üzerindeki etkileşimleri yönetmek için gereklidir.

Kullanıcı, ilaçlarının alınma saatini belirlemek için saatSecButton üzerine tıklayarak bir saat seçici aracılığıyla bir saat seçer. Seçilen saat ve dakika selectedHour ve selectedMinute değişkenlerinde saklanır.

İlaç adını ilacAdiEditText alanına girdikten sonra hatırlatıcıEkleButton düğmesine tıklayarak bir hatırlatıcı ekler. Eğer ilaç adı alanı boş bırakılırsa, kullanıcıya bir hata mesajı gösterilir. Aksi takdirde, setAlarm ve addMedicineEntry metodları çağrılarak ilaç hatırlatıcı ayarlanır ve eklenen ilaçlar listelenir.

geriDonButton düğmesine tıklayarak bu aktiviteden çıkabilir ve ana ekrana geri dönebilirler.

setAlarm metodu, AlarmManager kullanarak bir hatırlatıcı oluşturur ve ilgili alarmları ayarlar. Bu, belirtilen saatte kullanıcıya ilacını alması gerektiğini hatırlatmak için kullanılır.

showNotification metodu, kullanıcıya hatırlatma bildirimi göstermek için NotificationCompat.Builder kullanır. Bu bildirim, ilaç alınma saati geldiğinde kullanıcıya gönderilir.

addMedicineEntry metodu, eklenen ilaçı TextView olarak listeler ve yanına bir "Sil" düğmesi ekler. "Sil" düğmesine tıklandığında, ilgili ilaçın listeden kaldırılmasını sağlar.

```

1 package com.example.bitirme;
2
3 > import ...
4
5 < public class hatirlatici extends AppCompatActivity {
6
7     3 usages
8     private EditText ilacAdiEditText;
9     2 usages
10    private Button saatSecButton, hatirlaticiEkleButton, geriDonButton; // Geri dön butonu eklendi
11    5 usages
12    private LinearLayout ilaclarLayout;
13    5 usages
14    private int selectedHour, selectedMinute;
15    1 usage
16    private SharedPreferences sharedPreferences;
17    no usages
18    private ArrayList<String> ilaclarListesi = new ArrayList<>();
19
20    @Override
21    protected void onCreate(Bundle savedInstanceState) {
22        super.onCreate(savedInstanceState);
23        setContentView(R.layout.activity_hatirlatici);
24
25        ilacAdiEditText = findViewById(R.id.ilacAdiEditText);
26        saatSecButton = findViewById(R.id.saatSecButton);
27        hatirlaticiEkleButton = findViewById(R.id.hatirlaticiEkleButton);
28        geriDonButton = findViewById(R.id.geriDonButton); // Geri dön butonu tanımlandı
29        ilaclarLayout = findViewById(R.id.ilaclarLayout);
30        sharedPreferences = getSharedPreferences( name: "MyPrefs", Context.MODE_PRIVATE);
31
32    }
33
34    @Override
35    public void onClick(View v) {
36        switch (v.getId()) {
37            case R.id.saatSecButton:
38                Intent intent = new Intent("android.intent.action.TIME_PICKER");
39                intent.putExtra("ampm", false);
40                startActivityForResult(intent, 1);
41                break;
42            case R.id.hatirlaticiEkleButton:
43                String ilacAdi = ilacAdiEditText.getText().toString();
44                if (!ilacAdi.isEmpty()) {
45                    ilaclarListesi.add(ilacAdi);
46                    sharedPreferences.edit().putStringSet("ilaclarListesi", ilaclarListesi).apply();
47                }
48                break;
49            case R.id.geriDonButton:
50                finish();
51                break;
52        }
53    }
54
55    private void setAlarm() {
56        Intent intent = new Intent("com.android.alarm.intent.action.SEND_ALARM");
57        intent.putExtra("alarm_type", 1);
58        intent.putExtra("when", System.currentTimeMillis() + (selectedHour * 3600000) + (selectedMinute * 60000));
59        PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, intent, 0);
60        AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
61        alarmManager.setExact(AlarmManager.RTC_WAKEUP, pendingIntent.getWhen(), pendingIntent);
62    }
63
64    @Override
65    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
66        super.onActivityResult(requestCode, resultCode, data);
67        if (requestCode == 1) {
68            if (resultCode == RESULT_OK) {
69                String hourStr = data.getStringExtra("hour");
70                String minuteStr = data.getStringExtra("minute");
71                selectedHour = Integer.parseInt(hourStr);
72                selectedMinute = Integer.parseInt(minuteStr);
73                setAlarm();
74            }
75        }
76    }
77}

```

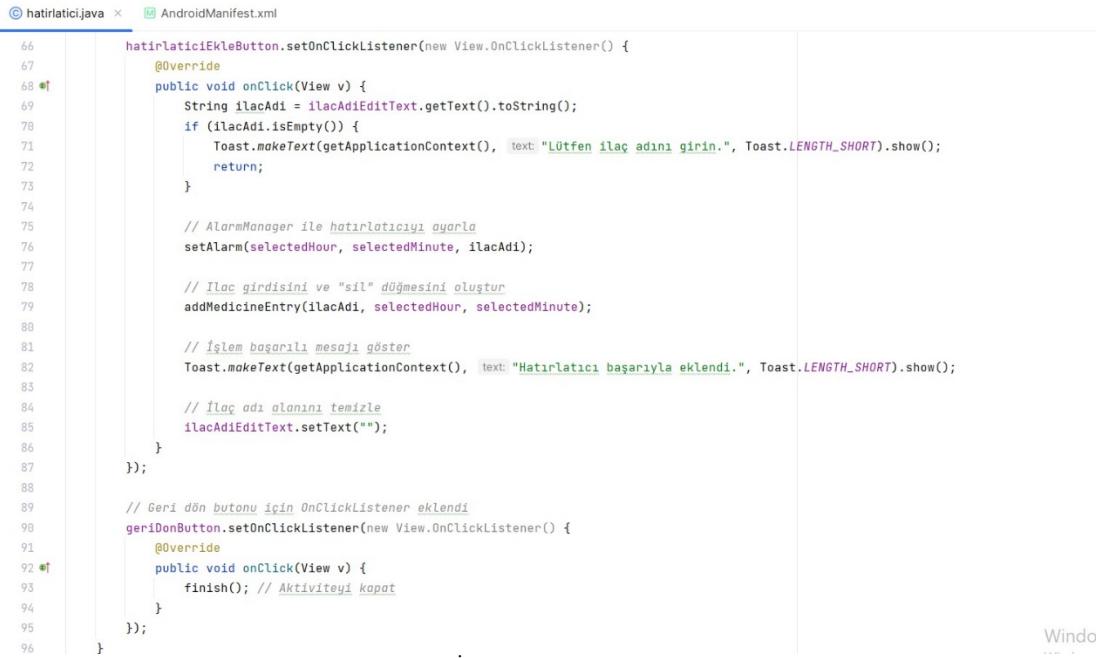
Resim 2.5.1: İlaç Hatırlatıcı Kod Bloğu

```

40
41
42    saatSecButton.setOnClickListener(new View.OnClickListener() {
43        @Override
44        public void onClick(View v) {
45            // Saat seçiminin başlat
46            final Calendar c = Calendar.getInstance();
47            selectedHour = c.get(Calendar.HOUR_OF_DAY);
48            selectedMinute = c.get(Calendar.MINUTE);
49
50            TimePickerDialog timePickerDialog = new TimePickerDialog( context: hatirlatici.this,
51                new TimePickerDialog.OnTimeSetListener() {
52                    1 usage
53                    @Override
54                    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
55                        selectedHour = hourOfDay;
56                        selectedMinute = minute;
57                    }
58                }, selectedHour, selectedMinute, is24HourView: true);
59            timePickerDialog.show();
60        }
61    }
62}

```

Resim 2.5.2: İlaç Hatırlatıcı Kod Bloğu Devamı



```

66     hatirlaticiEkleButton.setOnClickListener(new View.OnClickListener() {
67         @Override
68         public void onClick(View v) {
69             String ilacAdi = ilacAdiEditText.getText().toString();
70             if (ilacAdi.isEmpty()) {
71                 Toast.makeText(getApplicationContext(), text: "Lütfen ilaç adını girin.", Toast.LENGTH_SHORT).show();
72                 return;
73             }
74
75             // AlarmManager ile hatırlatıcıyı ayarla
76             setAlarm(selectedHour, selectedMinute, ilacAdi);
77
78             // İlac girdisini ve "sil" düğmesini oluştur
79             addMedicineEntry(ilacAdi, selectedHour, selectedMinute);
80
81             // İşlem başarılı mesajı göster
82             Toast.makeText(getApplicationContext(), text: "Hاتırlatıcı başarıyla eklendi.", Toast.LENGTH_SHORT).show();
83
84             // İlac adı alanını temizle
85             ilacAdiEditText.setText("");
86         }
87     });
88
89     // Geri döñ butonu için OnClickListener eklendi
90     geriDönButton.setOnClickListener(new View.OnClickListener() {
91         @Override
92         public void onClick(View v) {
93             finish(); // Aktiviteyi kapat
94         }
95     });
96 }

```

Resim 2.5.3: İlaç Hatırlatıcı Kod Bloğu Devamı



```

97     private void setAlarm(int hour, int minute, String ilacAdi) {
98         AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
99         Intent alarmIntent = new Intent(getApplicationContext(), AlarmReceiver.class);
100        alarmIntent.putExtra("ilacAdi", ilacAdi);
101        PendingIntent pendingIntent = PendingIntent.getBroadcast(getApplicationContext(), requestCode: 0, alarmIntent, flags: PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_MUTABLE);
102        Calendar alarmTime = Calendar.getInstance();
103        alarmTime.set(Calendar.HOUR_OF_DAY, hour);
104        alarmTime.set(Calendar.MINUTE, minute);
105        alarmTime.set(Calendar.SECOND, 0);
106
107        if (alarmTime.before(Calendar.getInstance())) {
108            alarmTime.add(Calendar.DATE, amount: 1); // Eğer belirlenen saat geçmişse, alarmı bir sonraki güne ertele
109        }
110
111        // Alarm kurma işlemi
112        alarmManager.setExact(AlarmManager.RTC_WAKEUP, alarmTime.getTimeInMillis(), pendingIntent);
113
114        // Bildirim gösterme işlemi
115        showNotification(alarmTime, ilacAdi);
116    }

```

Resim 2.5.4: İlaç Hatırlatıcı Kod Bloğu Devamı

```

118     @SuppressLint("MissingPermission")
119     private void showNotification(Calendar alarmTime, String ilacAdi) {
120         // Bildirim için PendingIntent oluştur
121         Intent intent = new Intent(getApplicationContext(), hatirlatici.class);
122         PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(), requestCode: 0, intent, flags: PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE);
123
124         // Bildirim mesajını oluştur
125         String notificationMessage = "Hatırlatıcı: " + ilacAdi + "\nSaat: " + String.format("%02d:%02d", alarmTime.get(Calendar.HOUR_OF_DAY), alarmTime.get(Calendar.MINUTE));
126         NotificationCompat.Builder builder = new NotificationCompat.Builder(context: this, channelId: "default")
127             .setSmallIcon(R.drawable.ic_notification_foreground)
128             .setContentTitle("İlaç Hatırlatıcı")
129             .setContentText(notificationMessage)
130             .setContentIntent(pendingIntent)
131             .setPriority(NotificationCompat.PRIORITY_DEFAULT)
132             .setAutoCancel(true);
133
134         // Bildirimi göster
135         NotificationManagerCompat notificationManager = NotificationManagerCompat.from(context: this);
136         notificationManager.notify(id: 1, builder.build());
137     }

```

Resim 2.5.5: İlaç Hatırlatıcı Kod Bloğu Devamı

```

139     i manager
140     private void addMedicineEntry(final String ilacAdi, int hour, int minute) {
141         // İlaç metnini oluştur
142         String ilacMetni = "İlaç Adı: " + ilacAdi + "\nHatırlatıcı Saati: " + String.format("%02d:%02d", hour, minute);
143
144         // TextView olarak ekle
145         TextView textView = new TextView(context: this);
146         textView.setText(ilacMetni);
147         ilaçlarLayout.addView(textView);
148
149         // "Sil" düğmesini oluştur
150         Button deleteButton = new Button(context: this);
151         deleteButton.setText("Sil");
152         deleteButton.setOnClickListener(new View.OnClickListener() {
153             @Override
154             public void onClick(View v) {
155                 // İlaç girişini kaldır
156                 ilaçlarLayout.removeView(textView);
157                 ilaçlarLayout.removeView((Button) v); // Sil düğmesini kaldır
158             }
159         });
160         ilaçlarLayout.addView(deleteButton);
161     }

```

Resim 2.5.6: İlaç Hatırlatıcı Kod Bloğu Devamı

2.7 Mobil Uygulamada Beslenme Önerileri

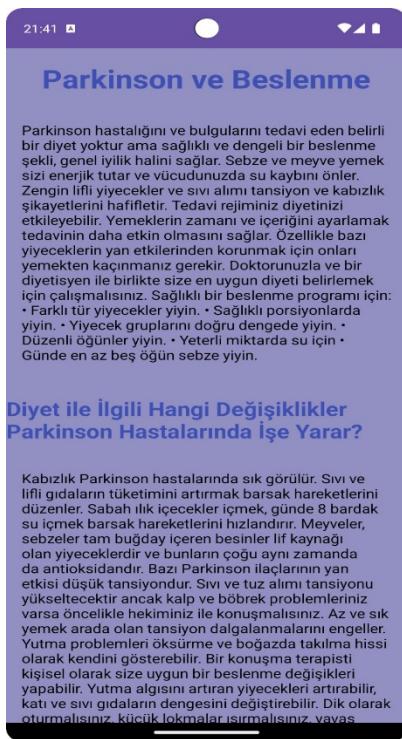
Bu bölümdeki çalışma, Android uygulamamızda beslenme ile ilgili açıklamalar ve örnek diyet listesi gösterilmesi sürecini inceler. Beslenme bölümünde parkinson ve beslenme ile ilgili bilgi verilir, diyet ile ilgili de açıklamalar yapılmaktadır. Bu şekilde kullanıcıya sunulması, kullanıcı deneyimini artırırken, 5 günlük diyet listesinin detaylı bir şekilde sunulması, kullanıcının beslenme planlarını daha etkili bir şekilde uygulamasına yardımcı olabilir.

2.7.1 Uygulama Yapısı ve İşlevselligi

Bu uygulama, kullanıcılarla sağlıklı beslenme ve örnek diyet listeleri sunar. Ana aktivite (`ParkinsonVeBeslenmeActivity`), kullanıcılarla Parkinson ve beslenme ile ilgili bilgiler verir diyet Parkinson hastalarında nasıl işe yarar anlatır. Örnek diyet listesi aktivitesi, kullanıcılarla önceden belirlenmiş diyet planlarına erişimini kolaylaştırır.

2.7.2 Ana Aktivite (`ParkinsonVeBeslenmeActivity`)

Ana aktivite, beslenme uygulamasının giriş noktasıdır. Kullanıcı, buradan beslenme ile ilgili bilgilere erişebilir. Nasıl bir beslenme alışkanlığı olması gerektiğini, diyetin parkinsonla ilişkisiyle ilgili bilgi edinebilir.



Resim 2.7: Beslenme Önerileri Sayfası



Resim 2.7.1: Beslenme Önerileri Sayfası Devamı

2.7.3 Ana Aktivite Kod Açıklaması

Bu sınıf, Parkinson hastalığı ve beslenme ile ilgili bilgilerin sunulduğu bir aktiviteyi tanımlar. Aktivite, örnek diyet listelerine erişim sağlayan bir buton içerir. Butona tıklandığında, kullanıcı örnek diyet listelerini görüntülemek için ilgili aktiviteye yönlendirilir. Bu kod, Parkinson hastalığına özgü beslenme önerilerine erişimi kolaylaştırır ve kullanıcının sağlıklı beslenme alışkanlıklarını geliştirmesine yardımcı olur.

```
1 package com.example.bitirme;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.widget.Button;
7
8 import androidx.appcompat.app.AppCompatActivity;
9
10 public class beslenme extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_beslenme);
16
17         // Örnek Diyet Listeleri Butonu'nun referansını al
18         Button buttonSampleDiets = findViewById(R.id.buttonSampleDiets);
19
20         // Örnek Diyet Listeleri Butonu'na tıklama dinleyicisi ekle
21         buttonSampleDiets.setOnClickListener(new View.OnClickListener() {
22             @Override
23             public void onClick(View v) {
24                 // Örnek Diyet Listeleri sayfasını başlat
25                 startActivity(new Intent(packageContext: beslenme.this, diyetlisteleri.class));
26             }
27         });
28     }
29 }
```

Resim 2.7.2: Beslenme Aktivitesinin Kod Açıklaması

2.7.4 Örnek Diyet Listesi Aktivitesi

Bu aktivite, kullanıcılarla önceden belirlenmiş örnek diyet listelerini sunar. Her bir diyet listesi, gün ve öğün bazında ayrıntılı olarak sunulur.



Resim 2.7.3: Örnek Diyet Listesi Sayfası



Resim 2.7.4: Örnek Diyet Listesi Sayfası Devamı

2.7.5 Örnek Diyet Listesi Aktivitesi Kod Açıklaması

Bu sınıf, örnek diyet listelerinin gösterildiği bir aktiviteyi tanımlar. Aktivite, beş günlük bir diyet programını içerir. Her bir gün için kahvaltı, ögle yemeği, akşam yemeği ve ara öğünlerin başlıklarları ve içerikleri metin görüntüleyiciler aracılığıyla kullanıcıya sunulur. Ayrıca, geri dönüş butonuna tıklanması durumunda bu aktiviteden önceki aktiviteye geri dönüş sağlanır. Bu kod, kullanıcıların sağlıklı beslenme alışkanlıklarını geliştirmelerine yardımcı olmak için örnek diyet listelerini kolayca erişilebilir hale getirir.

```
1 package com.example.bitirme;
2 import androidx.appcompat.app.AppCompatActivity;
3 import android.os.Bundle;
4 import android.widget.Button;
5 import android.widget.TextView;
6 public class diyetlisteleri extends AppCompatActivity {
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_diyetlisteleri);
11        // Geri döndür butonunu bul
12        Button backButton = findViewById(R.id.buttonBack);
13        // Geri döndür butonuna tıklama işlevi ekle
14        backButton.setOnClickListener(view -> onBackPressed());
15        // Gün 1
16        TextView textViewDay1Title = findViewById(R.id.textViewTitleSampleDiets1);
17        TextView textViewDay1BreakfastTitle = findViewById(R.id.textViewTitle1);
18        TextView textViewDay1BreakfastContent = findViewById(R.id.textViewContent1);
19        TextView textViewDay1MorningSnackTitle = findViewById(R.id.textViewTitle2);
20        TextView textViewDay1MorningSnackContent = findViewById(R.id.textViewContent2);
21        TextView textViewDay1LunchTitle = findViewById(R.id.textViewTitle3);
22        TextView textViewDay1LunchContent = findViewById(R.id.textViewContent3);
23        TextView textViewDay1AfternoonSnackTitle = findViewById(R.id.textViewTitle4);
24        TextView textViewDay1AfternoonSnackContent = findViewById(R.id.textViewContent4);
25        TextView textViewDay1DinnerTitle = findViewById(R.id.textViewTitle5);
26        TextView textViewDay1DinnerContent = findViewById(R.id.textViewContent5);
```

Resim 2.7.5: Diyet Listeleri Aktivite Kodunun Bir Parçası

2.6 Mobil Uygulamada Egzersiz Önerileri

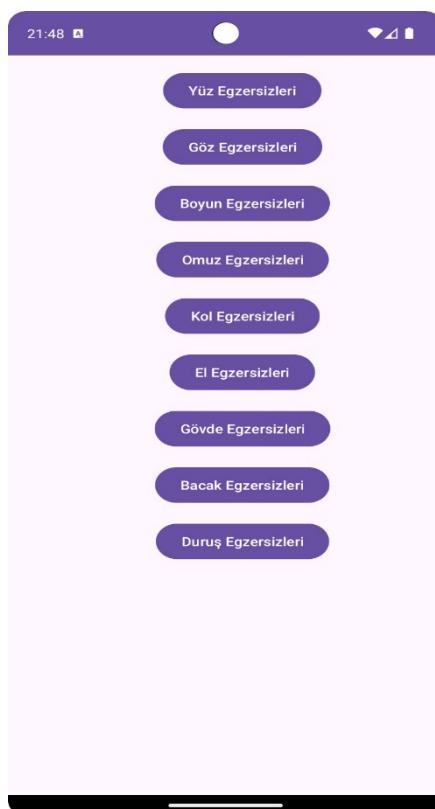
Bu bölümdeki çalışma, Android uygulamamızda egzersizlerin listelenmesi ve her bir egzersizin detaylarının gösterilmesi sürecini inceler. Egzersizlerin kategorilendirilmiş bir şekilde kullanıcıya sunulması, kullanıcı deneyimini artırırken, egzersiz detaylarının detaylı bir şekilde sunulması, kullanıcının egzersizleri daha etkili bir şekilde uygulamasına yardımcı olabilir.

2.6.1 Uygulama Yapısı ve İşlevselliği

Çalışmada incelenen Android uygulaması, farklı egzersiz kategorilerini kullanıcıya sunan bir ana aktivite (ExerciseActivity) ve her bir kategoriye ait egzersizleri detaylı olarak gösteren alt aktivitelerden oluşmaktadır. Ana aktivite, kullanıcının egzersiz kategorileri arasında geçiş yapmasına olanak tanırken, alt aktiviteler ise seçilen kategoriye ait egzersizleri listelemektedir. Egzersiz detaylarını içeren alt aktiviteler, kullanıcının seçtiği egzersizin başlık, açıklama ve görselini sunar.

2.6.2 Ana Aktivite (ExerciseActivity)

Ana aktivite, egzersiz uygulamasının giriş noktasıdır ve kullanıcıya farklı egzersiz kategorilerini sunar. Kullanıcı, bu aktiviteden istediği kategoriyi seçerek ilgili egzersizlerin listesini görüntüleyebilir.



Resim 2.6: Egzersiz Önerileri Ana Aktivite Sayfası

2.6.3 Ana Aktivite Kod Açıklaması

Bu bölümde, `ExerciseActivity` adlı ana aktivitenin kod yapısı ve işlevselligi açıklanacaktır. Bu aktivite, uygulamanın kullanıcı arayüzünde farklı egzersiz kategorilerine erişim sağlayan düğmeleri yönetir.

```
© ExerciseActivity.java ×

1 package com.example.bitirme;
2 import android.content.Intent;
3 import android.os.Bundle;
4 import android.view.View;
5 import android.widget.Button;
6 import androidx.appcompat.app.AppCompatActivity;
7 <?> public class ExerciseActivity extends AppCompatActivity {
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_exercise);
12         // Yüz Egzersizi Butonu
13         Button buttonFaceExercises = findViewById(R.id.buttonFaceExercises);
14         buttonFaceExercises.setOnClickListener(new View.OnClickListener() {
15             @Override
16             public void onClick(View v) {
17                 // Yüz egzersizleri sayfasına yönlendir
18                 Intent intent = new Intent(packageContext: ExerciseActivity.this, FaceExercisesActivity.class);
19                 startActivity(intent);
20             }
21         });
22         // Göz Egzersizi Butonu
23         Button buttonEyeExercises = findViewById(R.id.buttonEyeExercises);
24         buttonEyeExercises.setOnClickListener(new View.OnClickListener() {
25             @Override
26             public void onClick(View v) {
27                 // Göz egzersizleri sayfasına yönlendir
28                 Intent intent = new Intent(packageContext: ExerciseActivity.this, EyeExercisesActivity.class);
29                 startActivity(intent);
30             }
31         });
32     }
33 }
```

Resim 2.6.1: ExerciseActivity Kod Bloğu

Yukarıdaki kod, her bir egzersiz kategorisi için bir düğme tanımlar ve her bir düğmeye tıklanma durumunda ilgili egzersiz kategorisi sayfasına yönlendirme işlemini gerçekleştirir. Örneğin, buttonFaceExercises düğmesi yüz egzersizleri için tanımlanır ve tıklama dinleyicisi, bu düğmeye tıklandığında yüz egzersizleri sayfasına yönlendirilir. Her egzersiz için aynı işlem yapılır.

Bu kod, kullanıcının farklı egzersiz kategorilerine kolayca erişim sağlayabilmesini ve seçtiği kategorideki egzersizleri görüntüleyebilmesini sağlar. Bu, standart bir Android uygulama tasarımidır ve kullanıcı dostu bir deneyim sunmayı amaçlar.

2.6.4 Egzersiz Detayları Aktivitesi (ShoulderExercisesActivity)

Her egzersiz kategorisi için ayrı bir detay aktivitesi bulunmaktadır. Örneğin, omuz egzersizlerinin detaylarını içeren ShoulderExercisesActivity aktivitesi, kullanıcının seçtiği egzersizin başlık, açıklama ve görselini detaylı olarak sunar.

Tüm egzersizler için aynı kod bloğu kullanılmaktadır.



Resim 2.6.2: Egzersiz Detay Sayfası

2.6.5 Egzersiz Detayları Aktivitesi Kod Açıklaması

Bu kod parçası, omuz egzersizlerinin detaylarını içeren bir liste görünümünü yönetir. Her bir egzersiz için bir başlık, açıklama ve görsel içeren bir liste öğesi oluşturur. Kullanıcı bir öğeye tıkladığında, seçilen egzersizin başlığı bir Toast mesajı ile gösterilir. Bu aktivite, uygulamanın kullanıcı arayüzünde omuz egzersizlerinin detaylarını göstermek için kullanılır.

```
1  package com.example.bitirme;
2  import android.os.Bundle;
3  import androidx.appcompat.app.AppCompatActivity;
4  import androidx.annotation.Nullable;
5  import android.view.LayoutInflater;
6  import android.view.View;
7  import android.view.ViewGroup;
8  import android.widget.ArrayAdapter;
9  import android.widget.ImageView;
10 import android.widget.ListView;
11 import android.widget.TextView;
12 import android.widget.Toast;
13 <|-- public class ShoulderExercisesActivity extends AppCompatActivity {
14     // Yüz egzersizlerinin bilgilerini içeren dizi
15     2 usages
16     private ExerciseInfo[] shoulderExercises = {
17     };
18     @Override
19     protected void onCreate(@Nullable Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_shoulder_exercises);
22         // ListView'i bul
23         ListView listViewShoulderExercises = findViewById(R.id.listViewShoulderExercises);
24         // ArrayAdapter ile ListView'ye egzersizleri bağla
25         ExerciseAdapter adapter = new ExerciseAdapter(context: this, shoulderExercises);
26         listViewShoulderExercises.setAdapter(adapter);
27         // ListView'de bir öğeye tıklandığında o egzersizi göster
28         listViewShoulderExercises.setOnItemClickListener((parent, view, position, id) -> {
29             ExerciseInfo selectedExercise = shoulderExercises[position];
30             Toast.makeText(context: ShoulderExercisesActivity.this, selectedExercise.getTitle(), Toast.LENGTH_SHORT).show();
31         });
32     }
33 }
```

Resim 2.6.3: Egzersiz Detay Aktivitesi Kod Bloğu

```

32     // Egzersiz bilgilerini tutan iç içe sınıf
33     5 usages
34     private static class ExerciseInfo {
35         2 usages
36         private String title;
37         2 usages
38         private String description;
39         2 usages
40         private int imageResourceId;
41         no usages
42         ExerciseInfo(String title, String description, int imageResourceId) {
43             this.title = title;
44             this.description = description;
45             this.imageResourceId = imageResourceId;
46         }
47         1 usage
48         String getTitle() { return title; }
49         no usages
50         String getDescription() { return description; }
51         no usages
52         int getImageResourceId() { return imageResourceId; }
53     }
54     // ListView için özel bir ArrayAdapter
55     2 usages
56     private static class ExerciseAdapter extends ArrayAdapter<ExerciseInfo> {
57         1 usage
58         ExerciseAdapter(ShoulderExercisesActivity context, ExerciseInfo[] exercises) {
59             super(context, R.layout.activity_shoulder_exercises, exercises);
60         }

```

Resim 2.6.4: Egzersiz Detay Aktivitesi Kod Bloğu Devamı

```

57     @Override
58     public View getView(int position, View convertView, ViewGroup parent) {
59         // Öğrenin görünümünü al
60         View listItemView = convertView;
61         if (listItemView == null) {
62             LayoutInflator inflater = LayoutInflator.from(getContext());
63             listItemView = inflater.inflate(R.layout.activity_shoulder_exercises, parent, attachToRoot: false);
64         }
65         // Egzersiz bilgilerini al
66         ExerciseInfo currentExercise = getItem(position);
67         // Görünümdeki bilesenlere egzersiz bilgilerini yerleştir
68         ImageView imageViewExercise14 = listItemView.findViewById(R.id.ondortexercise);
69         TextView textViewExerciseTitle14 = listItemView.findViewById(R.id.ondortexercisead);
70         TextView textViewExerciseDescription14 = listItemView.findViewById(R.id.onbirexercisetext);
71         ImageView imageViewExercise15 = listItemView.findViewById(R.id.onbesexercise);
72         TextView textViewExerciseTitle15 = listItemView.findViewById(R.id.onbesexercisead);
73         TextView textViewExerciseDescription15 = listItemView.findViewById(R.id.onbesexercisetext);
74         return listItemView;
75     }
76 }

```

Resim 2.6.5: Egzersiz Detay Aktivitesi Kod Bloğu Devamı

2.6.6 ListView ve Özel Adaptör (ExerciseAdapter)

Egzersizlerin listelendiği aktivitelerde, ListView ve özel bir adaptör kullanılmaktadır. Adaptör, ListView'e egzersiz verilerini bağlar ve her bir liste öğesini özelleştirir.

Bu çalışma, kullanıcıların egzersiz yaparken bilinçli ve etkili bir şekilde hareket etmelerine yardımcı olabilir. Ayrıca, Android uygulama geliştiricileri için egzersizlerin listelenmesi ve detaylarının gösterilmesi konularında bir başlangıç noktası olabilir.

2.8 Mobil Uygulamada Parkinson Belirti Testi

Belirti testi, kullanıcıya 9 basit soru sorarak titreme, hareket yavaşlaması, kas sertliği, denge kaybı, hareket zorluğu, yazı değişiklikleri, yüz ifadesi değişiklikleri, ses değişiklikleri ve bilişsel sorunlar gibi tipik Parkinson belirtilerini değerlendirir. Kullanıcının cevaplarına dayanarak, uygulama belirtilerin şiddetini ve olası Parkinson hastalığı riskini değerlendirir. Bu belirti testi, kullanıcının evde veya günlük yaşamlarında kendi belirtilerini izlemelerine ve gerektiğinde bir sağlık uzmanına danışmalarına olanak tanır. Sonuç olarak, bu uygulama Parkinson hastalarının belirtilerini izlemelerine ve erken teşhis için önemli bir araç sağlar. Ayrıca, kullanıcı dostu arayüzü hastaların kendi sağlık durumlarını anlamalarına ve sağlık uzmanlarıyla daha etkili bir iletişim kurmalarına yardımcı olur.

2.8.1 QuestionActivity Sınıfı

QuestionActivity sınıfı, kullanıcıya Parkinson hastalığının belirtilerini değerlendirmek için bir dizi soru sunan ve kullanıcının cevaplarını işleyen bir etkinlik sınıfını temsil etmektedir. Bu sınıf, kullanıcıya her bir soruyu gösterir, cevaplarını işler ve bir sonraki soruya geçiş yapmasını sağlar. Ayrıca, tüm soruları cevaplardıktan sonra ResultActivity sınıfına geçiş yapar.



Resim 2.8: Parkinson Belirti Testinden Bir Soru

2.8.2 QuestionActivity Sınıfı Kod Açıklaması

Bu sınıf, uygulamanın belirti testi bölümünün temel işlevsellliğini sağlar. `onCreate()` metodu aktivite oluşturulduğunda çağrılır ve arayüz bileşenlerini başlatır; `showQuestion()` metodu mevcut soruyu göstermek için `questions` dizisinden soruyu alır ve `questionTextView` bileşenine ayarlar, ayrıca kullanıcının önceki seçimini temizler; `processAnswer()` metodu, kullanıcının seçtiği cevaba göre puan hesaplar ve bu puanı `answers` dizisine kaydeder; `showResult()` metodu tüm soruları cevaplardıktan sonra toplam puanı hesaplar ve `ResultActivity` sınıfını başlatarak kullanıcıya sonucu gösterir. Kullanıcı "Sonraki" butonuna tıkladığında, seçilen cevabı işler, bir sonraki soruyu gösterir ve tüm soruları cevaplandırdıysa sonucu gösterir.

```

@ anasayfa.java  @ test.java  x  @ testSonucu.java
33 // "Sonraki" butonuna tıklama işlemi ekle
34 nextButton.setOnClickListener(new View.OnClickListener() {
35     @Override
36     public void onClick(View v) {
37         // Seçilen cevabı al
38         int selectedRadioButtonId = answersRadioGroup.getCheckedRadioButtonId();
39         RadioButton selectedRadioButton = findViewById(selectedRadioButtonId);
40         String answerText = selectedRadioButton.getText().toString();
41         // Cevabı işle
42         processAnswer(answerText);
43         // Bir sonraki soruyu göster
44         questionIndex++;
45         if (questionIndex < questions.length) {
46             showQuestion();
47         } else {
48             // Tüm sorular cevaplandı, sonucu göster
49             showResult();
50         }
51     }
52 };
53 }
54 2 usages
55 private void showQuestion() {
56     // Soruya göster
57     questionTextView.setText(questions[questionIndex]);
58     // Önceki seçimi temizle
59     answersRadioGroup.clearCheck();
}
1 usage

```

Resim 2.8.1: Test Aktivitesi Kod Blogu

```

@ anasayfa.java  @ test.java  x  @ testSonucu.java
1 usage
private void processAnswer(String answerText) {
    // Seçilen cevabı log mesajıyla kontrol et
    Log.d("tag: QuestionActivity", "msg: " + "Seçilen cevap: " + answerText);
    // Seçilen cevabı göre puanı kaydet
    int answerScore;
    switch (answerText) {
        case "a) Evet, çok fazla":
            answerScore = 3; // Doğru puanı buraya tanımlayın
            break;
        case "b) Evet, orta seviyede":
            answerScore = 2;
            break;
        case "c) Evet, az seviyede":
            answerScore = 1;
            break;
        default:
            answerScore = 0;
            break;
    }
    // Cevabı kaydet
    answers[questionIndex] = answerScore;
    // Puanlama sonucunu log mesajıyla kontrol et
    Log.d("tag: QuestionActivity", "msg: " + "Cevap puanı: " + answerScore);
}
1 usage

```

Resim 2.8.2: Test Aktivitesi Kod Bloğu Devamı

```

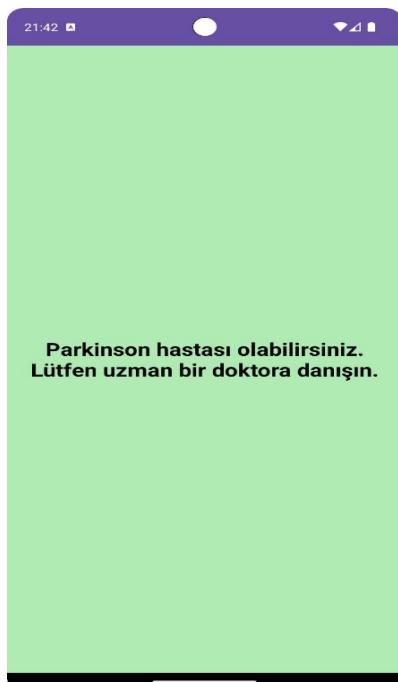
    anasayfa.java      test.java      testSonucu.java
81   // Puanlama sonucunu log mesajıyla kontrol et
82   Log.d( tag: "QuestionActivity", msg: "Cevap puanı: " + answerScore);
83 }
84 1usage
85 private void showResult() {
86     // Sonuçları hesapla
87     int totalScore = 0;
88     for (int score : answers) {
89         totalScore += score;
90     }
91     // Sonucu belirle
92     String resultMessage;
93     if (totalScore >= 15) {
94         resultMessage = "Parkinson hastası olabilirsiniz. Lütfen uzman bir doktora danışın.";
95     } else {
96         resultMessage = "Parkinson hastası olma olasılığınız düşük.";
97     }
98     // Log mesajı ekle
99     Log.d( tag: "QuestionActivity", msg: "Sonuç mesajı: " + resultMessage);
100    // Sonucu göster
101    Intent intent = new Intent( packageContext: this, testSonucu.class);
102    intent.putExtra( name: "RESULT_MESSAGE", resultMessage); // Doğru ekstra veriyi gönderiyoruz
103    startActivity(intent);
104 }
105 }
106

```

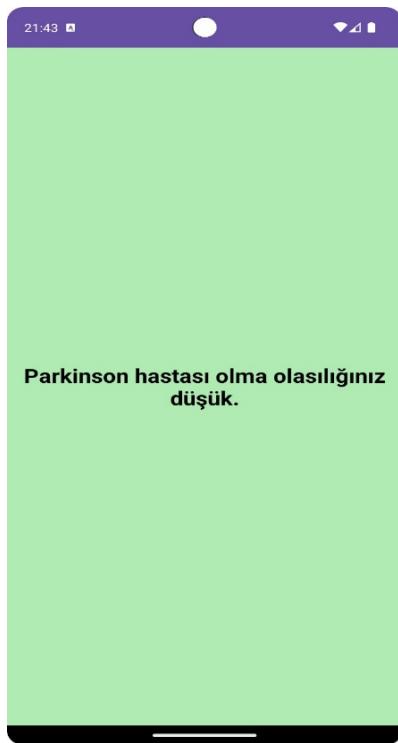
Resim 2.8.3: Test Aktivitesi Kod Bloğu Devamı

2.8.3 ResultActivity Sınıfı

ResultActivity sınıfı, eğer puan belirli bir eşik değerinden yüksekse, kullanıcıya Parkinson hastası olabileceği konusunda uyarı verir ve uzman bir kişiye başvurması gerektiğini belirtir. Aksi takdirde, Parkinson hastalığı olma olasılığının düşük olduğunu bildiren bir mesaj gösterir.



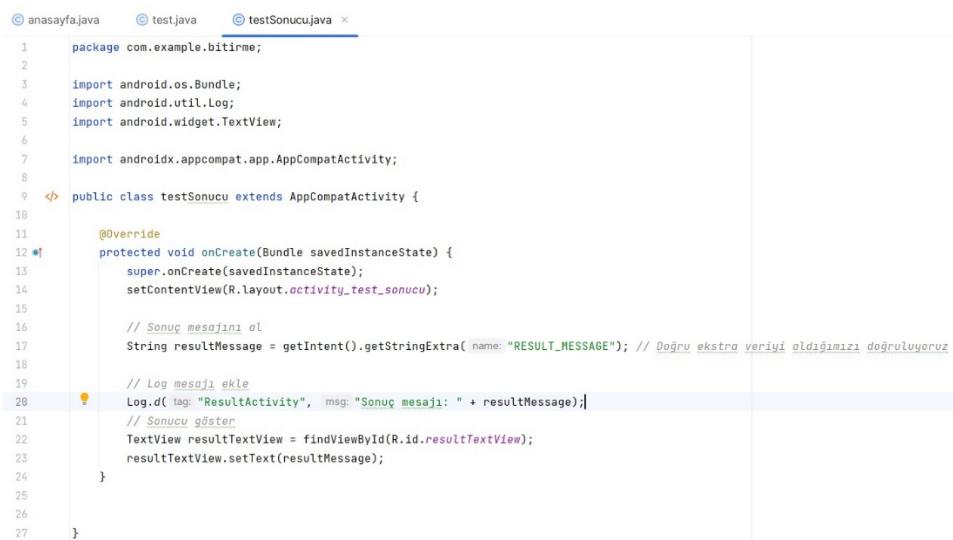
Resim 2.8.4: Test Sonucu



Resim 2.8.5: Farklı Bir Test Sonucu

2.8.4 Result Activity Sınıfı Kod Açıklaması

OnCreate() metodunda, aktivitenin başlatılmasıyla birlikte arayüz bileşenleri başlatılır ve resultTextView adlı bir TextView bileşeni belirlenir. Sonuç mesajı, önceki aktiviteden gelen ekstra veri olarak alınır ve resultTextView bileşenine yerleştirilerek kullanıcıya gösterilir. Ayrıca, alınan sonuç mesajı logcat üzerinde görüntülenir. Bu sınıf, uygulamanın belirtti testi sonuçlarını kullanıcılaraya göstermek için önemli bir işlevi yerine getirir.



```
anasayfa.java    test.java    testSonucu.java ×
1 package com.example.bitirme;
2
3 import android.os.Bundle;
4 import android.util.Log;
5 import android.widget.TextView;
6
7 import androidx.appcompat.app.AppCompatActivity;
8
9 public class testSonucu extends AppCompatActivity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_test_sonucu);
15
16         // Sonuç mesajını al
17         String resultMessage = getIntent().getStringExtra("RESULT_MESSAGE"); // Doğru ekstra veriyi aldığımızı doğruluyoruz
18
19         // Log mesajı ekle
20         Log.d("ResultActivity", "Sonuç mesajı: " + resultMessage);
21         // Sonucu göster
22         TextView resultTextView = findViewById(R.id.resultTextView);
23         resultTextView.setText(resultMessage);
24     }
25
26 }
27 }
```

Resim 2.8.6: Sonuç Aktivitesi Kod Bloğu

BÖLÜM 3

YAPAY ZEKA

3.1 Gerekli Kütüphanelerin Yüklenmesi

Bu çalışma kapsamında, yapay zeka modelini eğitmek ve değerlendirmek için Google Colab IDE'sini tercih ettim. Google Colab, bulut tabanlı bir Jupyter defteri hizmetidir ve derin öğrenme gibi yoğun hesaplama gerektiren görevleri yürütmek için ideal bir platform sağlar. Google Colab üzerinde model eğitimini gerçekleştirirken, çeşitli kütüphaneleri yükledim. Bu kütüphaneler aracılığıyla modelimizi oluşturduk ve eğittik. Öncelikle, derin öğrenme modeli için gereken Keras kütüphanesini yüklemek için pip install keras komutunu kullandık. Keras, derin öğrenme modellerini oluşturmak ve eğitmek için popüler bir kütüphanedir ve bu projede kullanılan ana araçlardan biridir. Ayrıca, veri manipülasyonu ve modelin eğitiminde kullanılan diğer kütüphaneleri de yükledim. Örneğin, veri önişleme için pandas ve numpy, veri görselleştirme için matplotlib ve seaborn, veri görselleştirme ve sonuçların sunumu için plotly, model eğitimi ve değerlendirmesi için scikit-learn, dengesiz veri kümelerini için de imbalanced-learn gibi kütüphaneleri kullandık.

3.2 Gerekli Python Kütüphaneleri ve İşlevleri

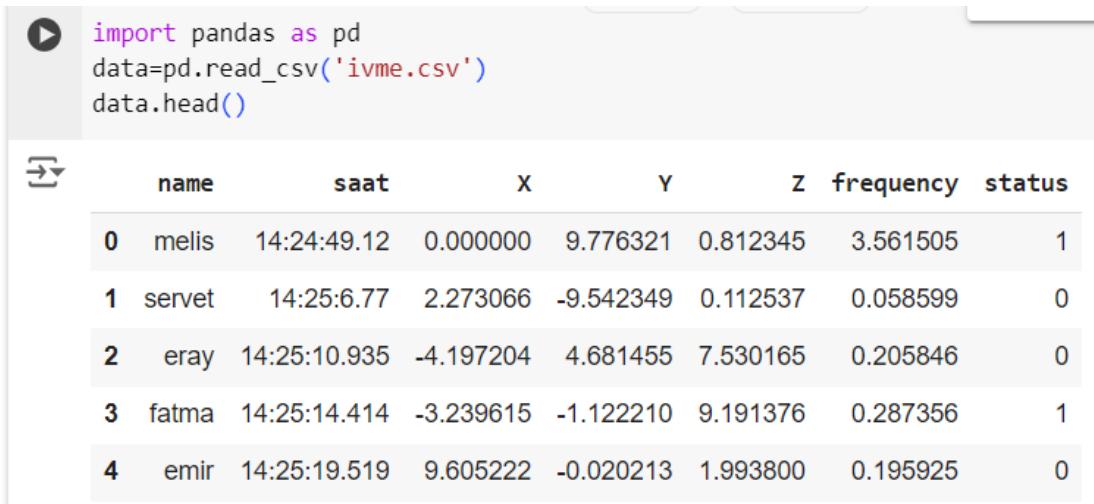
Bu bölümde, veri analizi ve makine öğrenimi modeli seçimi için gerekli olan Python kütüphaneleri ve işlevler kullanılmıştır. matplotlib.pyplot, seaborn ve plotly.express: Bu kütüphaneler, veri setinin yapısal özelliklerini görsel olarak analiz etmek ve model sonuçlarını sunmak için kullanılır. Matplotlib ve Seaborn, çeşitli grafik türlerini oluşturmak için kullanılırken, Plotly Express etkileşimli ve çekici grafikler oluşturmak için kullanılır. termcolor.colored: Bu işlev, konsola veya terminale yazdırılan metinleri renklendirmek için kullanılır. Metin tabanlı çıktıları daha

belirgin hale getirir ve sunumu geliştirir. `sklearn.model_selection.train_test_split`: Veri setini eğitim ve test alt kümelerine bölmek için kullanılır. Bu, modelin eğitilmesi ve performansının değerlendirilmesi için kullanılan standart bir işlemidir. `sklearn.model_selection.GridSearchCV`: Model hiperparametrelerinin en iyi değerlerini aramak için kullanılır. Bu, modelin optimize edilmesini ve en iyi performansın elde edilmesini sağlar. `sklearn.model_selection.cross_val_score`: Çapraz doğrulama kullanarak modelin performansını değerlendirmek için kullanılır. Bu, modelin genellemeye yeteneğini daha güvenilir bir şekilde değerlendirmeye olanak tanır. `sklearn.linear_model.LogisticRegression`, `sklearn.svm.SVC`: Lojistik regresyon ve destek vektör makineleri (SVM) gibi sınıflandırma algoritmalarını içeren modelleri oluşturmak için kullanılır. Bu modeller, sınıflandırma problemlerini çözmek için yaygın olarak kullanılır. `sklearn.metrics`: Doğruluk, F1 skoru, hatırlama, hassasiyet, karmaşıklık matrisi, ROC eğrisi ve AUC gibi çeşitli performans metriklerini içeren metrikler modülüdür. Bu metrikler, modelin sınıflandırma performansını ölçmek ve karşılaştırmak için kullanılır. Bu kütüphaneler ve işlevler, veri analizi yapmak, model seçimi yapmak ve modelin performansını değerlendirmek için kullanılan temel araçlardır. Bu araçlar, makine öğrenimi projelerinin farklı aşamalarında kullanılarak modelin geliştirilmesi ve doğrulanması için önemli bir rol oynar.

3.3 Veri Setinin Yüklenmesi ve Temel İnceleme

Bu bölümde, veri setinin yüklenmesi ve temel bir inceleme adımları gerçekleştirilmiştir. `import pandas as pd`: Pandas kütüphanesini içeri aktarır ve bu kütüphaneye `pd` kısaltmasıyla erişim sağlar. Pandas, veri manipülasyonu ve analizi için yaygın olarak kullanılan bir Python kütüphanesidir. `data = pd.read_csv('ivme.csv')`: `pd.read_csv()` işlevi, bir CSV dosyasını okumak ve bir `DataFrame`'e dönüştürmek için kullanılır. '`ivme.csv`', okunacak CSV dosyasının adını temsil eder. Dosya, mevcut çalışma dizininde bulunmalıdır. Dosya okunduktan

sonra, oluşturulan DataFrame data değişkenine atanır. `data.head()`: DataFrame'in ilk birkaç satırını (varsayılan olarak ilk beş satır) görüntüler. Bu, veri setinin genel yapısını ve içeriğini hızlı bir şekilde gözden geçirmenizi sağlar.



```
import pandas as pd  
data=pd.read_csv('ivme.csv')  
data.head()
```

	name	saat	x	y	z	frequency	status
0	melis	14:24:49.12	0.000000	9.776321	0.812345	3.561505	1
1	servet	14:25:6.77	2.273066	-9.542349	0.112537	0.058599	0
2	eray	14:25:10.935	-4.197204	4.681455	7.530165	0.205846	0
3	fatma	14:25:14.414	-3.239615	-1.122210	9.191376	0.287356	1
4	emir	14:25:19.519	9.605222	-0.020213	1.993800	0.195925	0

Resim 3.1: Veri Setinin Yapısı

Bu kod parçası, 'ivme.csv' adlı bir CSV dosyasını okur ve verileri DataFrame olarak data değişkenine yükler. `data.head()` işlevi, bu DataFrame'in ilk beş satırını görüntüler. Bu, veri setinin yapısal özelliklerini hızlıca incelememizi sağlar ve veri setinin içeriği hakkında bir fikir edinmemize yardımcı olur. Bu bilgiler, veri setinin daha detaylı bir analizine ve modelleme sürecine geçiş yapmadan önce önemli bir ön bilgi sağlar.

3.4 Veri Hazırlığı

3.4.1 Veri Setinin Yapısı ve Özellikleri

Bu bölümde, veri setinin yapısı ve özellikleri daha ayrıntılı olarak incelemektedir.

`data.info()`: Bu işlev, DataFrame'in genel yapısal bilgilerini görüntüler.

Özellikle, sütunların adları, veri türleri ve eksik değerler hakkında bilgi

sağlar. Bu işlev, veri setinin genel durumu hakkında bir özet sunar ve veri setinin daha ayrıntılı bir şekilde anlaşılmasına yardımcı olur.

```
✓ [13] data.info()  
sn.  
→ <class 'pandas.core.frame.DataFrame'>  
RangeIndex: 25 entries, 0 to 24  
Data columns (total 7 columns):  
 #   Column      Non-Null Count  Dtype     
---  --          --          --          --  
 0   name        25 non-null    object    
 1   saat        25 non-null    object    
 2   X            25 non-null    float64  
 3   Y            25 non-null    float64  
 4   Z            25 non-null    float64  
 5   frequency   25 non-null    float64  
 6   status       25 non-null    int64     
dtypes: float64(4), int64(1), object(2)  
memory usage: 1.5+ KB
```

Resim 3.2: Veri Setinin Genel Yapısı

Bu kod parçası, veri setinin yapısı ve özellikleri hakkında daha ayrıntılı bilgi sağlamak için kullanılır. `data.info()` işlevi, `DataFrame`'in genel yapısını özetler ve eksik veya boş değerlerin varlığı gibi önemli bilgileri sunar. Bu, veri setinin daha iyi anlaşmasına ve analizine yardımcı olur.

3.4.2 Veri Setindeki Eksik Değerler

Bu bölümde, veri setindeki eksik değerlerin incelenmesi ve ele alınması gerçekleştirilmektedir. `data.isna().sum()`: Bu işlev, `DataFrame`'in her bir sütununda eksik değerlerin sayısını hesaplar. `.isna()` işlevi, her hücrenin eksik olup olmadığını kontrol eder ve True veya False değerler döndürür. `sum()` işlevi, her sütundaki True değerlerin toplam sayısını hesaplar, yani her sütunda kaç tane eksik değer olduğunu verir. Bu, veri setindeki eksik değerlerin dağılımını anlamak ve gerekirse bunlarla başa çıkmak için kullanılır.

```
✓ 0
sn. data.isna().sum()
→ name      0
    saat     0
    X        0
    Y        0
    Z        0
    frequency 0
    status    0
    dtype: int64
```

Resim 3.3: Veri Setindeki Eksik Değerler

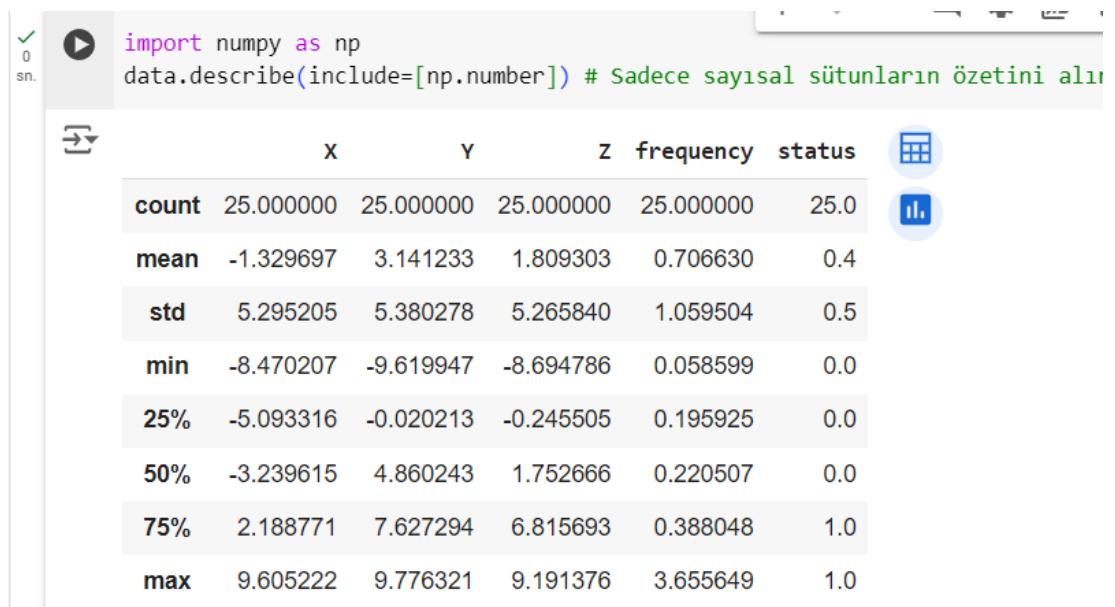
Bu kod parçası, veri setindeki eksik değerlerin sayısını hesaplayarak veri setinin eksikliklerini belirler. Eksik değerler, veri setinin doğru şekilde analiz edilmesini ve modele dahil edilmesini engelleyebilir, bu nedenle eksik değerlerin nasıl ele alınacağı veri hazırlığı sürecinde önemlidir.

3.4.3 Verilerden Özellik Seçimi

Bu bölümde, veri setinden özelliklerin ve etiketlerin ayrılması işlemi gerçekleştirilmektedir. `X = data.drop(["name", "status", "saat"], axis=1)`: Bu işlev, `DataFrame`'inden belirtilen sütunları ("name", "status" ve "saat") çıkararak özelliklerin oluşturulmasını sağlar. `drop()` işlevi, belirtilen sütunları (ve ekseni) kaldırır ve kalan sütunları yeni bir `DataFrame` olarak `X` değişkenine atar. Bu, modelin eğitiminde kullanılacak özelliklerin belirlenmesini sağlar. `y = data["status"]`: Bu işlev, `DataFrame`'indeki "status" sütununu etiketler olarak ayırır ve `y` değişkenine atar. Bu, modelin eğitiminde kullanılacak hedef değişkeni veya etiketlerin belirlenmesini sağlar. Bu kod parçası, veri setinden belirli sütunları çıkararak özelliklerin ve etiketlerin ayrılmasını gerçekleştirir. Özellikler, modelin eğitiminde bağımsız değişkenler olarak kullanılırken, etiketler, modelin eğitiminde hedef değişkeni olarak kullanılır. Bu ayrılmak, makine öğrenimi modelinin doğru bir şekilde eğitilmesini sağlar ve özelliklerin etiketlerden ayrılmasını kolaylaştırır.

3.4.4 Verilerin Özet İstatistikleri

Bu bölümde, veri setindeki sayısal sütunların istatistiksel özetinin alınması işlemi gerçekleştirilmektedir. `import numpy as np`: Numpy kütüphanesini içeri aktarır ve bu kütüphaneye `np` kısaltmasıyla erişim sağlar. Numpy, çok boyutlu diziler ve matrisler üzerinde işlem yapmak için kullanılan bir Python kütüphanesidir. `data.describe(include=[np.number])`: Bu işlev, DataFrame'deki sayısal sütunların istatistiksel özetini alır. `describe()` işlevi, bir DataFrame'in sayısal sütunları için çeşitli istatistiksel özelliklerin (count, mean, std, min, 25%, 50%, 75%, max) bir özetini sağlar. `include=[np.number]` parametresi, yalnızca sayısal sütunların özetini almayı sağlar, bu da kategorik sütunların özetini dışlar. Bu işlev, veri setindeki sayısal değerlerin dağılımı hakkında bir özet sunar.



The screenshot shows a Jupyter Notebook cell with the following code:

```
✓ 0 sn. ⏎ import numpy as np  
data.describe(include=[np.number]) # Sadece sayısal sütunların özetini alı
```

Below the code, the resulting output is displayed as a table:

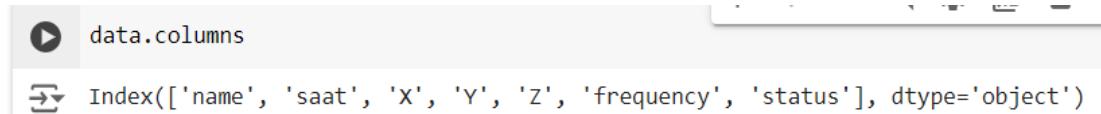
	X	Y	Z	frequency	status
count	25.000000	25.000000	25.000000	25.000000	25.0
mean	-1.329697	3.141233	1.809303	0.706630	0.4
std	5.295205	5.380278	5.265840	1.059504	0.5
min	-8.470207	-9.619947	-8.694786	0.058599	0.0
25%	-5.093316	-0.020213	-0.245505	0.195925	0.0
50%	-3.239615	4.860243	1.752666	0.220507	0.0
75%	2.188771	7.627294	6.815693	0.388048	1.0
max	9.605222	9.776321	9.191376	3.655649	1.0

Resim 3.4: Sayısal Sütunların İstatistiksel Özeti

Bu kod parçası, yalnızca sayısal sütunların istatistiksel özetini alarak veri setinin genel dağılımı hakkında bir fikir edinmemizi sağlar. Bu istatistiksel özet, veri setindeki sayısal değerlerin merkezi eğilimini, yayılımını ve dağılımını anlamamıza yardımcı olur. Bu bilgi, veri setinin analizi ve modelleme sürecinde önemli bir rol oynar.

3.4.5 Sütun İsimleri

Bu bölümde, veri setinde bulunan sütunların adlarının incelenmesi işlemi gerçekleştirilmektedir. `data.columns`: Bu işlev, DataFrame'deki tüm sütunların adlarını listeler. Bu, veri setindeki mevcut sütunların adlarını görüntülemek için kullanılır. Sütun adları, veri setindeki her bir özelliğin veya değişkenin adını temsil eder. Bu bilgi, veri setindeki özelliklerin ve değişkenlerin adlarını anlamamıza ve doğru şekilde kullanmamıza yardımcı olur.



```
data.columns
Index(['name', 'saat', 'X', 'Y', 'Z', 'frequency', 'status'], dtype='object')
```

Resim 3.5: Veri Setindeki Sütunların İsimleri

Bu kod parçası, veri setindeki tüm sütunların adlarını listeler ve bu sütun adları, veri setindeki özelliklerin ve değişkenlerin adlarını temsil eder. Bu bilgi, veri setinin analiz ve modelleme sürecinde hangi özelliklerin kullanılacağını belirlememize ve doğru sütunları seçmemize yardımcı olur.

3.4.6 Sınıf Dağılımı

Bu bölümde, veri setinde bulunan bir sütundaki benzersiz değerlerin sayısının hesaplanması işlemi gerçekleştirilmektedir. `data["status"].value_counts()`: Bu işlev, belirli bir sütundaki benzersiz değerlerin sayısını hesaplar ve bu değerleri frekans

tablosu olarak görüntüler. "status" sütununda bulunan her bir benzersiz değerin kaç kez tekrarlandığını gösterir. Bu, özellikle sınıflandırma problemlerinde sınıf dağılımını incelemek ve dengesizlikleri belirlemek için kullanılır.

```
data["status"].value_counts()
```

status	count
0	15
1	10

Name: count, dtype: int64

Resim 3.6: Status Değerlerinin Tekrarlanma Sayısı

Bu kod parçası, "status" sütunundaki benzersiz değerlerin sayısını hesaplar ve her bir değerin veri setinde kaç kez tekrarlandığını gösterir. Bu, veri setindeki sınıf dağılımını anlamamıza ve sınıf dengesizliklerini tespit etmemize yardımcı olur. Özellikle sınıflandırma problemlerinde, sınıf dağılımının dengeli olup olmadığını belirlemek için kullanılır.

3.5 Veri Bölümleme

Bu bölümde, veri setinin eğitim ve test alt kümelerine bölünmesi işlemi gerçekleştirilmektedir. `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)`: Bu işlev, veri setini özellikler ve etiketler olmak üzere eğitim ve test alt kümelerine ayırır. X, bağımsız değişkenlerin (özelliklerin) DataFrame'i ve y, bağımlı değişkenin (etiketlerin) DataFrame'i olarak temsil edilir. `test_size=0.2` parametresi, veri setinin %80'ini eğitim kümesine ve %20'sini test kümesine böler. `random_state=42` parametresi, veri setinin rastgele bölünmesini kontrol eden bir tohum değeridir. Bu işlev, veri setinin doğru şekilde eğitim ve test alt kümelerine ayrılmasını sağlar, böylece modelin eğitimi ve performansının değerlendirilmesi için kullanılır. Bu kod parçası, veri setini eğitim ve test alt

kümelerine böler ve bu alt kümeleri X_train, X_test, y_train ve y_test değişkenlerine atar. Bu, veri setinin model eğitimi ve performansının değerlendirilmesi için hazırlanmasını sağlar. Eğitim alt kümesi, modelin öğrenmesi için kullanılırken, test alt kümesi, modelin genellemeye yeteneğini değerlendirmek için kullanılır.

3.6 Veri Ön İşleme ve Ölçeklendirme

Bu bölümde, veri setindeki özelliklerin ölçeklendirilmesi işlemi gerçekleştirilmektedir. `from sklearn.preprocessing import StandardScaler: StandardScaler sınıfını sklearn.preprocessing modülünden içeri aktarır.` Bu sınıf, özelliklerin standart bir normal dağılıma sahip olacak şekilde ölçeklendirilmesini sağlar. `scaler = StandardScaler(): StandardScaler sınıfından bir örnek oluşturur.` Bu, veri setindeki özelliklerin ölçeklendirilmesi için kullanılacak bir standart ölçekleyiciyi temsil eder. `X_train_scaled = scaler.fit_transform(X_train):` Eğitim özelliklerini (`X_train`) standart ölçekleyiciye uyarlar ve dönüştürür. `fit_transform()` işlevi, ölçekleyiciyi eğitim verilerine uyum sağlamak için kullanılır ve ardından eğitim verilerini dönüştürür. `X_test_scaled = scaler.transform(X_test):` Test özelliklerini (`X_test`) standart ölçekleyiciye dönüştürür. Bu işlev, eğitilmiş ölçekleyiciyi kullanarak test verilerini ölçeklendirir, ancak ölçekleyiciyi yeniden eğitmez. Bu kod parçası, veri setindeki özelliklerin standart bir normal dağılıma sahip olacak şekilde ölçeklendirilmesini sağlar. Bu, özelliklerin model tarafından daha iyi anlaşılmasını ve daha iyi performans göstermesini sağlar. Ölçeklendirme işlemi, veri setinin ön işleme aşamasının önemli bir parçasıdır ve model performansını artırabilir.

3.7 Derin Öğrenme Modeli Oluşturma

Bu bölümde, derin öğrenme modelinin tanımlanması ve oluşturulması işlemi gerçekleştirilmektedir. `from tensorflow.keras.layers import Input: TensorFlow`

Keras'tan Input sınıfını içeri aktarır. Bu sınıf, modelin giriş katmanını tanımlamak için kullanılır. `import tensorflow as tf`: TensorFlow kütüphanesini içeri aktarır. TensorFlow, derin öğrenme modelleri oluşturmak ve eğitmek için popüler bir açık kaynaklı makine öğrenimi kütüphanesidir. `input_layer = Input(shape=(X_train.shape[1],))`: Input sınıfından bir örnek oluşturur ve giriş katmanını tanımlar. `shape=(X_train.shape[1],)` parametresi, giriş verisinin şeklini belirtir ve bu durumda özelliklerin sayısına eşittir. `model = tf.keras.Sequential([...])`: TensorFlow Keras'tan bir Sequential model oluşturur. Bu model, katmanları sıralı bir şekilde birbirine bağlar. `tf.keras.layers.Dense(64, activation='relu')`: İlk gizli katmanı tanımlar. Dense katmanı, tam bağlantılı bir katmandır, yani her bir birim önceki katmandaki tüm birimlerle bağlıdır. 64 birimden oluşur ve ReLU aktivasyon fonksiyonu kullanır. `tf.keras.layers.Dense(32, activation='relu')`: İkinci gizli katmanı tanımlar. Benzer şekilde, 32 birimden oluşur ve ReLU aktivasyon fonksiyonu kullanır. `tf.keras.layers.Dense(1)`: Çıkış katmanını tanımlar. Bu, tek bir çıkış birimiyle ve aktivasyon fonksiyonu olmadan (linear) bir Dense katmanıdır. Bu, regresyon problemleri için yaygın bir yapıdır. Bu kod parçası, derin öğrenme modeli için bir yapay sinir ağı (YSA) oluşturur. Model, giriş katmanı, iki gizli katman ve bir çıkış katmanından oluşur. Gizli katmanlar, ReLU aktivasyon fonksiyonunu kullanırken, çıkış katmanı doğrusal bir çıkış üretir. Bu yapı, regresyon problemleri için uygundur.

3.8 Model Derleme

Bu bölümde, derin öğrenme modelinin derlenmesi ve eğitimine hazırlanması işlemi gerçekleştirilmektedir. `model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])`: compile() işlevi, derin öğrenme modelini derler. Bu işlev, modelin optimize edici, kayıp fonksiyonu ve metriklerini belirlememizi sağlar. `optimizer='adam'`: Modelin optimize edici olarak Adam optimizer'ı kullanacağını belirtir. Adam, adaptif momentum tahmini ve köşeli düşük düzeyli ikinci momentin

adaptif tahmini (AdaGrad ve RMSProp'in bir kombinasyonu) ile öne çıkan bir optimizasyon algoritmasıdır. `loss='mean_squared_error'`: Modelin kayıp fonksiyonunu belirtir. Bu durumda, ortalama kare hatası (MSE) kullanılır. MSE, regresyon problemleri için yaygın bir kayıp fonksiyonudur ve tahmin edilen değerler ile gerçek değerler arasındaki farkın karesinin ortalamasıdır. `metrics=['accuracy']`: Modelin değerlendirmeye metriklerini belirtir. Bu durumda, sadece doğruluk (accuracy) metriği belirlenmiştir. Ancak, çoğunlukla regresyon problemleri için doğruluk değil, farklı bir değerlendirme metriği kullanılır. Bu genellikle R-kare veya MSE gibi regresyon performansını ölçen metriklerdir. Bu kod parçası, derin öğrenme modelini derlerken kullanılan optimize edici, kayıp fonksiyonu ve metrikleri belirtir. Bu, modelin eğitimini ve performans değerlendirmesini yapmak için gerekli ayarları yapar.

3.9 Model Eğitimi

Bu bölümde, derin öğrenme modelinin eğitimi işlemi gerçekleştirilmektedir. `model.fit(X_train_scaled, y_train, epochs=10, batch_size=32, validation_split=0.1)`: `fit()` işlevi, derin öğrenme modelini eğitmek için kullanılır. Bu işlev, eğitim verilerini modelde uyum sağlamak için kullanır ve modelin belirtilen sayıda epoch (iterasyon) boyunca eğitilmesini sağlar. `X_train_scaled`, `y_train`: Eğitim verilerini (özellikler ve etiketler) belirtir. Özellikler, ölçeklendirilmiş eğitim verilerini (`X_train_scaled`) temsil ederken, etiketler (`y_train`), eğitim verilerindeki hedef değişkenleri temsil eder. `epochs=10`: Modelin kaç epoch boyunca eğitileceğini belirtir. Bir epoch, modelin tüm eğitim verilerini bir kez geçmesi anlamına gelir. Bu durumda, model 10 epoch boyunca eğitilecektir. `batch_size=32`: Minibatch eğitimi sırasında kullanılacak batch boyutunu belirtir. Bir minibatch, eğitim verilerinin küçük bir alt kümesidir. Bu durumda, her bir minibatch'in 32 örneği vardır. `validation_split=0.1`: Eğitim sırasında doğrulama için kullanılacak veri oranını belirtir. Bu durumda, eğitim verilerinin %10'u doğrulama için ayrılacaktır. Model her epoch sonunda doğrulama verilerini

kullanarak performansını değerlendirecektir. Bu kod parçası, derin öğrenme modelini belirtilen eğitim verileriyle eğitmek için kullanılır. Model, eğitim verileri üzerinde belirtilen sayıda epoch boyunca eğitilir ve her epoch sonunda doğrulama verileriyle performansı değerlendirilir.

3.10 Eğitim Sürecinin Görselleştirilmesi

Bu bölümde, derin öğrenme modelinin eğitim sürecinin görselleştirilmesi işlemi gerçekleştirilmektedir. `history = model.fit(X_train_scaled, y_train, epochs=10, batch_size=32, validation_data=(X_test_scaled, y_test))`: Modelin eğitimi için `fit()` işlevi kullanılır. Bu işlev, eğitim verilerini modelde uyum sağlamak için kullanır ve eğitim süreci boyunca kayıp ve doğruluk metriklerini kaydeder. `validation_data` parametresi, doğrulama için kullanılacak veri setini belirtir. Bu durumda, test verileri (`X_test_scaled` ve `y_test`) doğrulama için kullanılacaktır. Eğitim süreci tamamlandıktan sonra, `history` değişkeninde eğitim sürecine ait bilgiler bulunur. `plt.plot(history.history['loss'], label='Training Loss')`: Eğitim kaybının (loss) epoch'a göre değişimini görselleştirmek için kullanılır. `history.history['loss']`, eğitim süreci boyunca kaydedilen eğitim kaybı değerlerini içerir. `plt.plot(history.history['val_loss'], label='Validation Loss')`: Doğrulama kaybının (loss) epoch'a göre değişimini görselleştirmek için kullanılır. `history.history['val_loss']`, eğitim süreci boyunca kaydedilen doğrulama kaybı değerlerini içerir. `plt.plot(history.history['accuracy'], label='Training Accuracy')`: Eğitim doğruluğunun (accuracy) epoch'a göre değişimini görselleştirmek için kullanılır. `history.history['accuracy']`, eğitim süreci boyunca kaydedilen eğitim doğruluk değerlerini içerir. `plt.plot(history.history['val_accuracy'], label='Validation Accuracy')`: Doğrulama doğruluğunun (accuracy) epoch'a göre değişimini görselleştirmek için kullanılır. `history.history['val_accuracy']`, eğitim süreci boyunca kaydedilen doğrulama doğruluk değerlerini içerir.

```

import matplotlib.pyplot as plt

history = model.fit(X_train_scaled, y_train, epochs=10, batch_size=32, validation_data=(X_test_scaled, y_test))

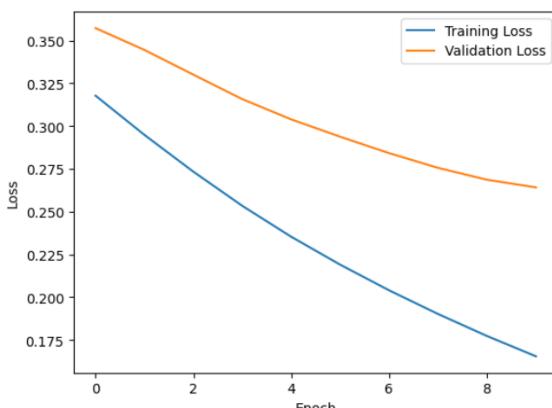
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

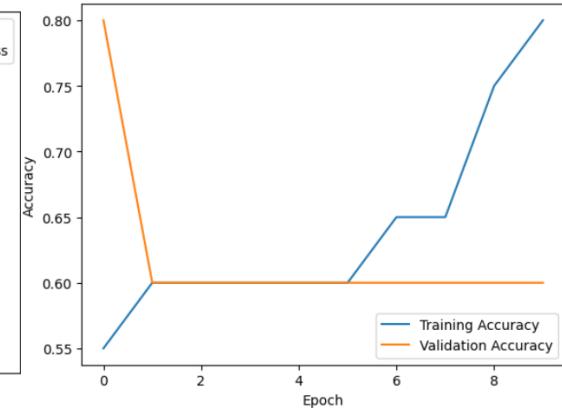
Epoch 1/10
1/1 [=====] - 3s 3s/step - loss: 0.3178 - accuracy: 0.5500 - val_loss: 0.3572 - val_accuracy: 0.8000
Epoch 2/10
1/1 [=====] - 0s 178ms/step - loss: 0.2948 - accuracy: 0.6000 - val_loss: 0.3445 - val_accuracy: 0.6000
Epoch 3/10
1/1 [=====] - 0s 145ms/step - loss: 0.2733 - accuracy: 0.6000 - val_loss: 0.3300 - val_accuracy: 0.6000
Epoch 4/10
1/1 [=====] - 0s 136ms/step - loss: 0.2534 - accuracy: 0.6000 - val_loss: 0.3157 - val_accuracy: 0.6000
Epoch 5/10
1/1 [=====] - 0s 151ms/step - loss: 0.2353 - accuracy: 0.6000 - val_loss: 0.3039 - val_accuracy: 0.6000
Epoch 6/10
1/1 [=====] - 0s 164ms/step - loss: 0.2190 - accuracy: 0.6000 - val_loss: 0.2938 - val_accuracy: 0.6000
Epoch 7/10
1/1 [=====] - 0s 133ms/step - loss: 0.2040 - accuracy: 0.6500 - val_loss: 0.2842 - val_accuracy: 0.6000
Epoch 8/10
1/1 [=====] - 0s 162ms/step - loss: 0.1902 - accuracy: 0.6500 - val_loss: 0.2756 - val_accuracy: 0.6000
Epoch 9/10
1/1 [=====] - 0s 161ms/step - loss: 0.1773 - accuracy: 0.7500 - val_loss: 0.2686 - val_accuracy: 0.6000
Epoch 10/10
1/1 [=====] - 0s 309ms/step - loss: 0.1655 - accuracy: 0.8000 - val_loss: 0.2642 - val_accuracy: 0.6000

```

Resim 3.7: Modelin Eğitim Süreci



Resim 3.8: Modelin Kayıp Değerleri



Resim 3.9: Modelin Doğruluk Değerleri

Bu kod parçasığı, derin öğrenme modelinin eğitim sürecindeki kayıp ve doğruluk değerlerini görselleştirmek modelin performansının izlenmesini sağlar. Bu grafikler, modelin eğitim sürecindeki iyileşmeyi ve aşırı uyumu (overfitting) izlemek için kullanılır.

3.11 Modelin Kaydedilmesi

Bu bölümde, eğitilmiş bir TensorFlow Keras modelinin diskte saklanması işlemi gerçekleştirilmektedir. `model.save('acce_model.keras')`: Eğitilmiş modelin diskte saklanması için `save()` işlevi kullanılır. Bu işlev, eğitilmiş modelin belirtilen dosya adı ve uzantısıyla kaydedilmesini sağlar. Bu durumda, eğitilmiş model `acce_model.keras` adıyla kaydedilecektir ve `.keras` uzantısıyla bir TensorFlow Keras modeli olarak kaydedilecektir. Bu kod parçası, eğitilmiş bir TensorFlow Keras modelini diskte saklamak için kullanılır. Bu, modeli daha sonra yükleyerek kullanma veya başka bir ortamda dağıtmaya gibi işlemler için kullanılmalıdır.

3.12 Modelin Yüklenmesi

Bu bölümde, önceden kaydedilmiş bir TensorFlow Keras modelinin diskten yüklenmesi işlemi gerçekleştirilmektedir. `model = tf.keras.models.load_model('acce_model.keras')`: Önceden kaydedilmiş bir TensorFlow Keras modelinin diskten yüklenmesi için `load_model()` işlevi kullanılır. Bu işlev, belirtilen dosya adı ve uzantısından modeli yükler. Bu durumda, `acce_model.keras` adlı dosyadan model yüklenir. Bu kod parçası, önceden eğitilmiş bir TensorFlow Keras modelini diskten yüklemek için kullanılır. Bu, modelin daha sonra kullanılması veya başka bir ortamda dağıtılması gibi işlemler için kullanılmalıdır.

3.13 TensorFlow Lite Formatına Dönüşüm

Bu bölümde, TensorFlow Lite Converter kullanılarak TensorFlow Keras modelinin TensorFlow Lite formatına dönüştürülmesi işlemi gerçekleştirilmektedir. `converter = tf.lite.TFLiteConverter.from_keras_model(model)`: TensorFlow Lite Converter'ı kullanarak TensorFlow Keras modelini TensorFlow Lite formatına dönüştürmek için `from_keras_model()` işlevi kullanılır. Bu işlev, önceden yüklenmiş bir TensorFlow

Keras modelini alır ve onu TensorFlow Lite formatına dönüştürmek için bir dönüştürücü oluşturur. `tflite_model = converter.convert()`: Oluşturulan dönüştürücüyü kullanarak modeli TensorFlow Lite formatına dönüştürmek için `convert()` işlevi kullanılır. Bu işlev, TensorFlow Lite formatına dönüştürülmüş bir model döndürür. Bu kod parçası, TensorFlow Keras modelini TensorFlow Lite formatına dönüştürmek için TensorFlow Lite Converter'ı kullanır. Bu, modelin daha hafif ve daha verimli bir biçimde kullanılmasını sağlar, özellikle kaynak sınırlı cihazlarda (örneğin, mobil cihazlar veya IoT cihazları) dağıtım için uygun olabilir.

3.14 TensorFlow Lite Modelinin Kaydedilmesi

Bu bölümde, TensorFlow Lite modelinin diskte bir dosyaya kaydedilmesi işlemi gerçekleştirilmektedir. `with open('model.tflite', 'wb') as f: f.write(tflite_model)`: TensorFlow Lite modelini diskte bir dosyaya kaydetmek için `open()` işlevi kullanılır. Bu işlev, belirtilen dosya adı (`model.tflite`) ve yazma modu ('wb' - binary yazma) ile bir dosya nesnesi oluşturur. Ardından, `write()` işlevi kullanılarak TensorFlow Lite modeli (`tflite_model`) bu dosyaya yazılır. Bu kod parçası, TensorFlow Lite modelini diskte bir dosyaya kaydetmek için kullanılır. Bu, TensorFlow Lite modelini daha sonra kullanmak veya başka bir ortamda dağıtmak için gereklidir.

3.15 Eğitilen Modelin Android Studio Entegrasyonu

Eğitilen bir modelin Android Studio'da kullanılması, mobil uygulamaların yapay zekâ teknolojilerinden yararlanmasında kritik bir adımdır. Bu bölümde, eğitilmiş bir TensorFlow Lite modelinin Android Studio projelerine nasıl entegre edileceği adım adım açıklanmaktadır.

3.15.1 ModelLoader

Bu sınıf, eğitilmiş bir TensorFlow Lite modelinin Android uygulamasına entegrasyonunu kolaylaştırır ve modelin doğru şekilde yüklenmesini ve çalıştırılmasını sağlar.

```
7  public class ModelLoader {
8      3 usages
9      private Interpreter interpreter;
10     1 usage
11     public ModelLoader(String modelPath) {
12         try {
13             interpreter = new Interpreter(loadModelFile(modelPath));
14         } catch (IOException e) {
15             e.printStackTrace();
16         }
17     }
18     1 usage
19     private MappedByteBuffer loadModelFile(String modelPath) throws IOException {
20         FileInputStream inputStream = new FileInputStream(modelPath);
21         FileChannel fileChannel = inputStream.getChannel();
22         long startOffset = 0;
23         long declaredLength = fileChannel.size();
24         return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength);
25     }
26     1 usage
27     public void runModel(float[][] input) {
28         if (interpreter != null) {
29             int inputSize = input.length;
30             float[] outputArray = new float[inputSize];
31             for (int i = 0; i < inputSize; i++) {
32                 float[] currentInput = input[i];
33                 float[] currentOutput = new float[1]; // Tek bir çıktı değeri olduğunu varsayıyoruz
34                 interpreter.run(new float[][]{currentInput}, currentOutput);
35                 float output = currentOutput[0]; // Tek bir çıktı olduğunu varsayıyoruz
36                 System.out.println("Model output for input " + i + ": " + output);
37             }
38         } else {
39             System.out.println("Interpreter is null. Model is not loaded or initialized properly.");
40         }
41     }
42 }
```

Resim 3.10: Modelin Yüklediği ModelLoader Kod Bloğu

Constructor (ModelLoader(String modelPath)): Bu constructor, modelPath parametresiyle belirtilen TensorFlow Lite modelinin yüklenmesini gerçekleştirir. Yüklenen model, Interpreter nesnesi olarak oluşturulur ve sınıfın örneği içinde saklanır. loadModelFile(String modelPath): Bu özel metot, belirtilen modelPath

üzerindeki TensorFlow Lite model dosyasını yükler. Model dosyası bir MappedByteBuffer olarak yüklenir ve dönüştürülür. runModel(float[][] input): Bu metot, yüklenen model üzerinde giriş verilerini çalıştırır ve çıkışları alır. Her bir giriş verisi, modelde çalıştırılmadan önce uygun formata dönüştürülür. Her bir giriş için model çalıştırılır ve çıkışlar alınır. Her bir çıkış, System.out.println() ile konsola yazdırılır. Yükleme işlemi sırasında herhangi bir hata oluşursa, IOException ile ele alınır ve hata mesajı konsola yazdırılır.

3.15.2 TensorFlow Lite Modelinin Eklenmesi

İlk adım olarak, eğitilmiş TensorFlow Lite modelin, Android projesine eklenmesi gerekmektedir. Bu genellikle model dosyasının assets veya res/raw klasörlerine kopyalanmasıyla gerçekleştirilir. Bu adım, modelin uygulama paketi içinde yer almاسını sağlar.

3.15.3 Android Studio Projesine Entegrasyon

TensorFlow Lite kütüphanesinin Android Studio projelerine entegre edilmesi gerekmektedir. Bu işlem, Gradle dosyalarında (build.gradle) uygun bağımlılıkların eklenmesiyle gerçekleştirilir. TensorFlow Lite kütüphanesi, projenin derleme sürecinde kullanılabilir hale gelir.

3.15.4 Modelin Yüklenmesi

Android uygulamasında, TensorFlow Lite modelinin yüklenmesi için gerekli kod parçacıkları yazılmalıdır. Bu adım, genellikle TensorFlow Lite Interpreter sınıfının kullanılmasıyla gerçekleştirilir. Modelin yüklenmesi, model dosyasının okunması ve belleğe yüklenmesini içerir.

3.15.5 Giriş ve Çıkış Verilerinin İşlenmesi

Modelin giriş ve çıkış formatına uygun şekilde giriş ve çıkış verilerinin işlenmesi gerekmektedir. Giriş verileri, modelin beklentilerine uygun formata dönüştürülmeli ve modelden gelen çıkış verileri doğru şekilde yorumlanmalıdır.

3.15.6 Modelin Kullanılması

Eğitilen model, Android uygulamasında kullanılmak üzere çağrılmalıdır. Modelin çağrılması, giriş verilerinin iletilmesi ve modelden gelen çıktıların işlenmesini içerir. Sonuçlar, kullanıcı arayüzünde gösterilebilir veya başka bir amaç için kullanılabilir.

3.15.7 Performans ve Uyum Testleri

Entegre edilen modelin doğru çalıştığından ve kullanıcı deneyimini olumsuz yönde etkilemediğinden emin olmak için performans ve uyumluluk testleri yapılmalıdır. Modelin doğru çalışması ve performansının beklenen düzeyde olması sağlanmalıdır.

Eğitilen bir modelin Android Studio'da başarıyla entegre edilmesi, mobil uygulamanın daha zengin ve etkili hale gelmesine olanak sağlar. Bu adımlar takip edilerek, eğitilmiş bir TensorFlow Lite modeli Android Studio projelerine başarıyla entegre edilebilir.

KAYNAKLAR

- <https://dergipark.org.tr/tr/pub/ejosat/issue/45333/568544>
- <https://dergipark.org.tr/en/pub/ngumuh/issue/47481/524658>
- <https://dergipark.org.tr/en/pub/dubited/issue/56311/688223>
- <https://acikbilim.yok.gov.tr/handle/20.500.12812/152988>
- <https://dergipark.org.tr/en/download/article-file/1366537>
- <https://www.medyahaber.com/umut-olan-arastirma-akilli-saatler-parkinsonda-care-buldu-parkinson-hastaligina-akilli-saatler-sayesinde-7-yil-onceden-teshis-hareket-takip-cihazi-ve-akilli-saatler-tibbi-gozlem-amaciyla-kullanilabilecek-haber-107826>
- <https://dergipark.org.tr/en/download/article-file/2542729>
- https://www.researchgate.net/profile/Ozal-Bizal-2/publication/266201055_Determination_of_Parkinson's_Disease_with_Machine_Learning_Techniques/links/542981170cf28c9e5f2f72a6/Determination-of-Parkinsons-Disease-with-Machine-Learning-Techniques.pdf
- Ene, Marius. "Neural network-based approach to discriminate healthy people from those with Parkinson's disease." Annals of the University of Craiova-Mathematics and Computer Science Series 35 (2008): 112-116.
- Little, Max A., et al. "Suitability of dysphonia measurements for telemonitoring of Parkinson's disease." Biomedical Engineering, IEEE Transactions on 56.4 (2009): 1015-1022.
- Sakar, C. Okan, and Olcay Kursun. "Telediagnosis of Parkinson's disease using measurements of dysphonia." Journal of medical systems 34.4 (2010): 591-599.
- Das, Resul. "A comparison of multiple classification methods for diagnosis of Parkinson disease." Expert Systems with Applications 37.2 (2010): 1568-1572.

Caglar, Mehmet Fatih, Bayram Cetisli, and Inayet Burcu Toprak. "Automatic Recognition of Parkinson's Disease from Sustained Phonation Tests Using ANN and Adaptive Neuro-Fuzzy Classifier." *Journal of Engineering Science and Design* 1.2 (2010): 59-64.

Polat, Kemal. "Classification of Parkinson's disease using feature weighting method on the basis of fuzzy C-means clustering." *International Journal of Systems Science* 43.4 (2012): 597-609.

Luukka, Pasi. "Feature selection using fuzzy entropy measures with similarity classifier." *Expert Systems with Applications* 38.4 (2011): 4600-4607.

<https://www.acarindex.com/bilisim-teknolojileri-dergisi/bulut-tabanli-mobil-diyabet-kontrol-uygulamasi-mobil-diyabetim-293498>

https://www.researchgate.net/publication/352773666_Yapay_Zeka_Tabani_Akilli_Telefon_Uygulamasi_ile_Kan_Sekeri_Tahmini

<https://www.sciencedirect.com/science/article/pii/S1877050918308883>

https://www.tensorflow.org/lite/inference_with_metadata/codegen?hl=tr#java

<https://www.tensorflow.org/lite/models/convert/metadata?hl=tr>

<https://colab.research.google.com/drive/1EV31xIMnldySkkHbiQ613Au-WYPhHjPy?authuser=1&pli=1#scrollTo=1wDlXFwJLncl>

<https://chatgpt.com>

<https://www.tensorflow.org/?hl=tr>

<https://www.tensorflow.org/lite?hl=tr>

ÖZGEÇMİŞ

Servet GÜNEŞ

2001 yılında İstanbul'da doğdu. İlk, orta öğrenimini İstanbul'da tamamladı. 2015 yılında İstanbul'da Beşir Balcioğlu Anadolu Lisesi'nde başladığı lise öğrenimini 2019 yılında bitirdi. 2019 yılında Bolu Abant İzzet Baysal Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü'ne girdi. Bilgisayar Mühendisliği Bölümü'nde son sınıf öğrencisi olarak mezun olma aşamasındadır.

İletişim Adresi:

e-posta: servetgunes662@gmail.com

Melis DEĞIRMENCI

2002 yılında Kocaeli'nde doğdu. İlk, orta öğrenimini Kocaeli'nde tamamladı. 2016 yılında Kocaeli'nde Gebze Anadolu Lisesi'nde başladığı lise öğrenimini 2020 yılında Anibal Anadolu Lisesi'nde bitirdi. 2020 yılında Bolu Abant İzzet Baysal Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü'ne girdi. Bilgisayar Mühendisliği Bölümü'nde son sınıf öğrencisi olarak mezun olma aşamasındadır.

İletişim Adresi:

e-posta: meliisdegirmencii@gmail.com