T.C.
DÜZCE ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

NORMALİZE EDİLMİŞ KALP HIZI DEĞİŞKENLİĞİ ANALİZİ İLE PAROKSİSMAL ATRİYAL FİBRİLASYON TESPİTİ

MURAT SÜRÜCÜ

DOKTORA TEZİ
ELEKTRİK-ELEKTRONİK VE BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

DANIŞMAN
PROF. DR. RESUL KARA

DÜZCE, 2021

T.C.
DÜZCE ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

NORMALİZE EDİLMİŞ KALP HIZI DEĞİŞKENLİĞİ ANALİZİ İLE PAROKSİSMAL ATRİYAL FİBRİLASYON TESPİTİ

Murat SÜRÜCÜ tarafından hazırlanan tez çalışması aşağıdaki jüri tarafından Düzce Üniversitesi Lisansüstü Eğitim Enstitüsü Elektrik-Elektronik ve Bilgisayar Mühendisliği Anabilim Dalı'nda **DOKTORA TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Prof. Dr. Resul KARA

Düzce Üniversitesi

Eş Danışman

Doç. Dr. Yalçın İŞLER

İzmir Kâtip Çelebi Üniversitesi

Jüri Üyeleri

Prof. Dr. Resul KARA
Düzce Üniversitesi _____

Doç. Dr. Mehmet ŞİMŞEK
Düzce Üniversitesi _____

Dr. Öğr. Üyesi İbrahim Halil DAMAR
Düzce Üniversitesi _____

Doç. Dr. Yakup KUTLU
İskenderun Teknik Üniversitesi _____

Dr. Öğr. Üyesi Yılmaz Kemal YÜCE
Alanya Alaaddin Keykubat Üniversitesi _____

Tez Savunma Tarihi: 25/10//2021

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

25 Ekim 2021

Murat SÜRÜCÜ

TEŞEKKÜR

Doktora öğrenimimde ve bu tezin hazırlanmasında gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Prof. Dr. Resul KARA'ya en içten dileklerimle teşekkür ederim.

Tez çalışmam boyunca değerli katkılarını, tecrübelerini ve enerjisini esirgemeyen eş danışmanım Doç. Dr. Yalçın İŞLER'e, desteğini esirgemeyen ve çalışkanlığı ile esin kaynağım olan Prof. Dr. Mahmut ÖZER'e ve eğitim hayatım boyunca yardımlarını ve desteklerini esirgemeyen sevgili anne babama, eşime ve çocuklarıma sonsuz teşekkürlerimi sunarım.

25 Ekim 2021 **Murat SÜRÜCÜ**

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

ÇİZELGE LİSTESİ

Sayfa No

KISALTMALAR

ACC	Genel başarım
ADD	Ayrık dalgacık dönüşümü
AF	Atriyal fibrilasyon
AFPDB	Atriyal fibrilasyon tahmini veritabanı
AV	Atriyoventriküler
CHD	Koroner kalp rahatsızlığı
CNN	Evrişimli sinir ağları
CVSD	KHD değişiminin varyasyonu
CVNN	KHD verisinin varyasyonu
DDD	Daubechies dalgacık dönüşümü
DFT	Ayrık Fourier dönüşümü
DL	Derin öğrenme
DT	Karar ağacı
DWT	Ayrık dalgacık dönüşümü
EKG	Elektrokardiyografi
FFT	Hızlı Fourier dönüşümü
FIR	Sonlu darbe cevabı
FL	Bulanık mantık
GA	Genetik algoritma
HDD	Haar dalgacık dönüşümü
HF	Yüksek frekans
KHD	Kalp hızı değişkenliği
KNN	K- en yakın komşu
LDA	Doğrusal ayraç analizi
LF	Düşük frekans
LS	Lomb-Scargle
MAXHR	En büyük kalp atış hızı
MEDNN	KHD ortanca değeri
MINHR	En küçük kalp atış hızı
MHR	Kalp atış hızı ortalaması
MLP	Çok katmanlı algılayıcı ağ
MNN	Ortalama KHD değeri
NKHD	Normalize kalp hızı değişkenliği
NN20	KHD değişimi 20 ms den fazla olanların sayısı
NN50	KHD değişimi 50 ms den fazla olanların sayısı
NSR	Normal sinüs ritmi
OSS	Otonom sinir sistemi
PAC	Prematüre atriyal kompleks (Erken kulakçık vurusu)
PAF	Paroksismal atriyal fibrilasyon
PNN20	KHD değişimi 20 ms den fazla olanların oranı
PNN50	KHD değişimi 50 ms den fazla olanların oranı
PSD	Güç spektral yoğunluk

PVC	Prematüre ventriküler kompleks (Erken karıncık vurusu)
QRS	EKG dalgasındaki en yüksek genlikli dalga formu
RANGENN	KHD değişim aralığı
RBF	Radyal tabanlı fonksiyon
RLS	Düzenlileştirilmiş en küçük kareler
RMSSD	KHD değişiminin RMS değeri
SA	Sinoatriyal
SDNN	KHD verisinin standart sapması
SDHR	Kalp atış hızının standart sapması
SDSD	KHD değişiminin standart sapması
SEN	Hassaslık
SGD	Olasılıksal dereceli azalma
SPE	Seçicilik
SVM	Destek vektör makinaları
SVT	Supraventriküler taşikardi
ULF	Ultra düşük frekans
VLF	Çok düşük frekans
VT	Ventriküler taşikardi

SİMGELER

σ	Standart sapma
μ	Ortalama
λ	Eğilim yoketme düzenlileştirme katsayısı
θ	Eğilim yoketme regresyon katsayısı
γ	RBF fonksiyonu eğrilik miktarı
τ	Zaman gecikmesi
ω	Açısal frekans
ψ	Dalgacık fonksiyonu
Ω	Direnç büyüklüğü (Ohm)
ϕ	Eşik değer

ÖZET

NORMALİZE EDİLMİŞ KALP HIZI DEĞİŞKENLİĞİ ANALİZİ İLE PAROKSİSMAL ATRİYAL FİBRİLASYON TESPİTİ

Murat SÜRÜCÜ
Düzce Üniversitesi
Lisansüstü Eğitim Enstitüsü, Elektrik-Elektronik ve Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi
Danışman: Prof. Dr. Resul KARA
Ekim 2021, 126 sayfa

Paroksismal Atriyal Fibrilasyon (PAF), en yaygın kalp ritmi bozukluğu türlerinden biri olan Atriyal Fibrilasyon'un başlangıç aşamasıdır. PAF doğrudan ölümcül olmasa bile, ölümcül diğer rahatsızlıkları tetiklemekte ve inme riskini artırmaktadır. Üstelik hasta PAF atağı geçirdiği sırada, kendisinin ve başkalarının hayatını etkileyen bir aktivite içinde olabilir. Bu nedenle, hem PAF'ın mümkün olduğunca erken teşhis edilerek tedaviye başlanması hem de PAF atağı geçirmeden önce hastanın güvenli bir konuma geçmesi önemlidir. Bu çalışmada, her iki görev için geleneksel makine öğrenmesi yöntemleri ve derin öğrenme yöntemleri çalışılmıştır. Bu tez çalışmasında, açık erişimli olup 49 sağlıklı bireye, yakın zamanda atak geçirmeyen 24 PAF hastasına ve yakın zamanda atak geçirecek olan 25 PAF hastasına ait 30 dakikalık EKG verileri kullanılmıştır. Bu EKG verilerinden elde edilen Kalp Hızı Değişkenliği (KHD) verilerinin yanısıra kalp hızı normalizasyonu uygulanan KHD (NKHD) verileri elde edilmiştir. KHD ve NKHD verileri için yaygın kullanılan zaman alanı, frekans alanı, dalgacık dönüşümü ve doğrusal olmayan öznitelikler hesaplanmıştır. Bu özniteliklere farklı öznitelik normalizasyonu yöntemleri uygulanmıştır. Bu şekilde oluşturulan altı farklı öznitelik setine genetik algoritma ile öznitelik seçim işlemi uygulanmıştır. Çıkarılan ve seçilen öznitelikler k yakın komşu, çok katmanlı algılayıcı, radyal tabanlı çekirdek fonksiyonuna sahip destek vektör makinesi ve evrişimli sinir ağı temelli derin öğrenme sınıflandırıcı algoritmalarının girişlerine uygulanmıştır. Sonuç olarak, z-skor normalizasyonu uygulanmış NKHD özniteliklerinin derin öğrenme algoritması ile hem PAF teşhisinde hem PAF atağı erken tespitinde %100 sınıflandırıcı başarımına ulaşılmıştır. Üstelik literatürde derin öğrenme ham veri üzerinde uygulanıyor olsa da, KHD analizi gibi kendine özgü öznitelik çıkarma yöntemi bulunan problemlerde, öncelikle öznitelik çıkarmanın faydalı olacağı görülmüştür.

Anahtar sözcükler: Derin öğrenme, Genetik algoritma, Kalp hızı değişkenliği, Normalize kalp hızı değişkenliği, Paroksismal Atriyal Fibrilasyon.

ABSTRACT

DETECTION OF PAROXISMAL ATRIAL FIBRILATION WITH NORMALIZED HEART RATE VARIABILITY ANALYSIS

Murat SÜRÜCÜ
Düzce University
Institute of Graduate Programs, Department of Electrical-Electronics and Computer Engineering
Doctoral Thesis
Supervisor: Prof. Dr. Resul KARA
October 2021, 126 pages

Paroxysmal Atrial Fibrillation (PAF) is the initial stage of Atrial Fibrillation, one of the most common arrhythmia types. Although PAF is not directly fatal, it triggers other deadly conditions and increases the risk of stroke. Moreover, during the PAF episode, the patient may be in an action that affects his and others' lives. Therefore, both the early diagnosis to start treatment immediately and the prediction of attacks to warn the patient to take a safe place are essential. In this study, traditional machine learning methods including deep learning were evaluated for both tasks. In this thesis, an open-access database that consists of 30-minute ECG data from 49 healthy individuals, 24 PAF patients with no recent attack, and 25 PAF patients with the recent attack was used. Heart Rate Variability (HRV) data and heart rate normalized HRV (NHRV) data were obtained from these ECGs. Commonly used time-domain, frequency-domain, wavelet transform, and nonlinear features were extracted for both data. Feature normalization methods were also utilized for these features. The genetic algorithm chose the more valuable features. Extracted and selected features are applied to inputs of k-nearest neighbors, multi-layer perceptron, support vector machines with radial basis kernel function, and convolutional neural-network-based deep learning classifiers. As a result, both tasks resulted in 100% classifier performances with the deep learning algorithm using NHRV features with z-score normalization. Moreover, although deep learning runs with raw data in the literature, feature extraction may be efficient in problems that require specific feature extraction methods such as HRV analysis.

Keywords: Deep learning, Genetic algorithm, Heart rate variability, Normalized heart rate variability, Paroxysmal Atrial Fibrillation.

EXTENDED ABSTRACT

DETECTION OF PAROXISMAL ATRIAL FIBRILATION WITH NORMALIZED HEART RATE VARIABILITY ANALYSIS

Murat SÜRÜCÜ
Düzce University
Institute of Graduate Programs, Department of Electrical-Electronics and Computer Engineering
Doctoral Thesis
Supervisor: Prof. Dr. Resul KARA
October 2021, 126 pages

1. INTRODUCTION

Paroxysmal Atrial Fibrillation (PAF) is the initial stage of Atrial Fibrillation that is one of the most common arrhythmia types. Although it does not threaten life directly, it can trigger other fatal disorders and increase the risk of stroke. Therefore, it is essential to diagnose PAF as early as possible. In addition, there may be dangers for the patient and others during the PAF episodes. This study aims to determine PAF diagnosis (Task 1) and early prediction of PAF attacks (Task 2). We focused on whether heart rate and feature normalization methods affect the classifier performances on both tasks in particular.

2. MATERIAL AND METHODS

The data, freely acquired from the Atrial Fibrillation Prediction Database, consists of 30-minute ECG recordings from 49 normal subjects, 24 PAF patients with no near attack, and 25 PAF patients with having near an attack. The heart rate variability (HRV) data were obtained from these ECG recordings. The optional preprocess of heart rate normalization to a fixed 75 beats is used to determine heart rate normalized HRV (NHRV) data. Then, conventional HRV features of time-domain, frequency-domain, wavelet transform, and nonlinear-domain were calculated from HRV and NHRV data. These extracted features directly or their MinMax and z-score normalized versions constructed six distinct feature sets. Optionally, the genetic algorithm selects the efficient features from these sets. These all features and selected features only were applied to classifiers of k-nearest neighbors (kNN), multi-layer perceptron (MLP), support vector machines (SVM), and convolutional neural network (CNN). As a result, twelve different feature

sets and four distinct classifier algorithms with different hyperparameters were utilized for both tasks.

3. RESULTS AND DISCUSSIONS

Task 1 resulted in the accuracies of 81.00% with MinMax normalization using kNN algorithm, 91.92% with Z-Score normalization using the MLP algorithm, 95.92% with Z-Score normalization using the CNN algorithm, and 97.96% with Z-Score normalization using the SVM algorithm in HRV data. This task achieved the accuracies of 86.00% with z-score normalization using the kNN algorithm, 95.96% with Z-Score normalization using the MLP algorithm, 96.94% with MinMax normalization using the SVM algorithm, and 100% with Z-Score normalization using the CNN algorithm in NHRV data. Similarly, task 2 resulted in the accuracies of 78.00% with Z-Score normalization using the kNN algorithm, 87,76% with Z-Score normalization using the CNN algorithm, 89.80% with MinMax normalization using the SVM algorithm, and 93.88% with Z-Score normalization using the MLP algorithm in HRV data. Finally, task 2 achieved the accuracies of 82.00% with MinMax normalization using the kNN algorithm, 97.96% with Z-Score normalization using the MLP algorithm, 97.96% with MinMax normalization using the SVM algorithm, and 100% with Z-Score normalization using the CNN algorithm in NHRV data. As a result, the use of the CNN algorithm with z-score normalized features from heart rate normalized HRV data gave the maximum accuracies for both tasks.

4. CONCLUSION AND OUTLOOK

CNN classifier outperforms the similar studies for both PAF diagnosis and PAF attack prediction using the heart rate normalization and z-score feature normalization. In addition, this thesis presented a new potential use of the heart rate normalization method in both the PAF diagnosis and screening of recent PAF episodes.

1. GİRİŞ

Kalp, vücudun hayatta kalması için gerekli olan kanın pompalandığı organdır [1]. Sağ ve sol yarısında, karıncık ve kulakçık olarak adlandırılan toplam dört kısımdan oluşur. Vücutta kirlenen kanı akciğerlere iletmek, akciğerde oksijen doygunluğuna ulaşmış kanı ise tekrar vücuda pompalama işlevini üstlenmektedir. Bu işlemleri kendi ürettiği elektriksel uyartılarla gerçekleştirmektedir [2].

Kalp rahatsızlıkları, ölüme sebep olan hastalıklar arasında başta gelmektedir. Avrupa kıtasındaki tüm ölümlerin yaklaşık %45'i kalp rahatsızlıkları temellidir [3]. Atriyal fibrilasyon (AF) ise yaşamı doğrudan tehdit etmeyen en yaygın kalp ritmi bozukluğudur. Ancak bazı diğer önemli rahatsızlıkları tetiklemektedir. Örneğin inme riskini 5 kat artırmaktadır [4]. Ayrıca AF temelli inmeler daha fazla kalıcı hasara ve ölüme sebebiyet vermektedir [5]. Sigara tüketimi, yüksek tansiyon, diyabet ve ileri yaşın AF gelişiminde etkisi olmaktadır. Üstelik koroner kalp hastalığı (CHD) ve kalp kapak hastalığı gibi hastalıkların eşliğinde de AF gözlenebilmektedir [6].

Hayatı tehdit eden pek çok hastalık genellikle klinik olarak kolay tespit edilemeyen bir evrede başlar. Hastalıklar tedavi edilemez noktaya evrilmeden önce erken teşhiste bulunarak tedaviye başlamak, hastalığa bağlı rahatsızlık ve ölüm risklerini azaltmaktadır [7]. Benzer şekilde AF hastalarında da yaşam kalitesi ve ölüm riskinin azaltılması için erken teşhis önemlidir [8]. Yakın tarihli bir çalışmada, binlerce hasta üzerinde yapılan taramalar sonucunda, klinik olarak tespit edilememiş erken teşhis AF vakalarında başlanılan erken tedavi sayesinde 1,5 yıllık süreçte inme riski %4'ten %1'e, ölüm oranı ise %7'den %4'e indiği gözlenmiştir [9]. Paroksismal Atriyal Fibrilasyon (Paroxysmal Atrial Fibrillation – PAF), AF vakalarının %25 ila %62'sini oluşturmaktadır [10]. PAF, 2 dakika ile 7 gün arasında kendiliğinden sonlanan AF türüdür, ancak zamanla daha kalıcı hale gelmektedir, bu açıdan erken tespiti oldukça önemlidir [11]. PAF rahatsızlığı çoğunlukla asemptomatik seyrettiği için erken tanı konulamayan bir AF türüdür [12]. Ayrıca PAF vakaları klinik olarak tespit edilebilir evreye geldiklerinde ise aniden rahatsızlık geliştirebilmektedir [13]. Bu sebeple PAF vakalarının klinik evre oluşmadan erken teşhisi, hem yaşam konforu hem de kalıcı AF oluşmaması açısından önemlidir. Çözüm bekleyen bir diğer önemli problem ise PAF şüphesi bulunan kişilerde gelişecek

olan atakların önceden kestirilmesidir. Önceden atak kestirimi sayesinde atak gelişmeden önlemlerin alınması ve kalp ritminin Normal Sinüs Ritmine (NSR) dönmesi için gerekli tedavi ve uygulamaların gerçekleştirilmesi hedeflenmektedir [14].

Tüm kalp rahatsızlıklarında olduğu gibi PAF vakalarının tespitinde de elektrokardiyografi (EKG) başta olmak üzere çeşitli fonksiyonel testler kullanılmaktadır [15], [16]. EKG çoğu klinikte bulunan elektriksel bir kayıt türüdür. EKG kaydı, sağlık personeli tarafından doğrudan incelenebileceği gibi yazılım aracılığıyla da işlemeye müsait sayısal veriler içermektedir. Literatürde EKG sayısal verisi üzerinden kalp rahatsızlıklarının tespiti ile ilgili pek çok çalışma bulunmaktadır [17]. EKG sinyalinde periyot, 5 farklı bölgeye ayrılmaktadır. Bu bölgeler P-QRS-T dalgaları olarak isimlendirilmektedir [18]. Son zamanlarda EKG işaretlerindeki kalp atımını gösteren QRS kompleksi adı verilen dalganın tepe noktaları arasındaki değişim üzerinden analiz şeklinde ortaya çıkan kalp hızı değişkenliği (KHD) analizi yaygın kullanıma sahip hale gelmiştir [19]. KHD analizi AF tespiti dışında, diyabetik otonom nöropati tanısı [20], enfeksiyon prognozu [21], ani kardiyak ölümü [22], damar sertliği [23], miyokard iskemisi [24], Parkinson hastalarında otonomik disfonksiyon [25], Konjestif kalp yetmezliği [26] tespiti gibi klinik çalışmalarda da yaygın olarak kullanılan bir veri analizi türüdür. KHD analizinde, ortalama, standart sapma ve benzeri zaman alanı istatistiksel hesaplamaları kullanılmaktadır [27], [28]. Ayrıca frekans bölgesi özellikleri için Fast Fourier Transform (FFT), Lomb-Scargle (LS) ve diğer periodogramlar kullanılmaktadır [29]. Bu ölçümlerde, belirli bir frekans bölgesindeki KHD sinyalinin güç spektral yoğunluğu (PSD) öznitelik olarak kullanılmaktadır [30]. Çalışmalarda, 0 – 0,4 Hz aralığı genellikle ultra düşük frekans (ULF), çok düşük frekans (VLF), düşük frekans (LF) ve yüksek frekans (HF) şeklinde 4 bölgeye ayrılmaktadır. Bu yöntemlere ek olarak, zaman-frekans bölgesi analizi için dalgacık dönüşümü ve dalgacık entropisi de kullanılmaktadır [31], [32]. Ayrıca KHD verisi Gauss dağılımlı ve doğrusal olmadığından bispektral analizle de başarılı sonuçlar elde edilmektedir [11]. Benzer şekilde, KHD verisine göre çizilen kalp atış hızı aralıklarının grafiği olan Poincare grafiği gibi doğrusal olmayan yöntemler de literatürde kullanılmaktadır [33], [34].

Bu çalışmada da kullanılan “The Computer in Cardiology Challenge 2001” kapsamında erişime açık sunulan AF tahmin veritabanı (Atrial Fibrillation Prediction Database - AFPDB) , PAF tespiti başarımını ölçmede sıklıkla kullanılmaktadır [35]. Literatürde AFPDB veri seti ile PAF teşhisinde ve atak tespitinde KHD analizi kullanılarak yapılan

çalışmalarda, veri uzunluğu 5 dakikadan 30 dakikaya kadar farklı uzunluklarda ve farklı sınıflandırıcılarla pek çok çalışma yapılmıştır. KHD çalışmalarındaki çeşitliliğin artması sonucunda oluşturulan bir çalışma grubu tarafından KHD çalışmalarına bir standart getirilmiştir. Genel kullanım için kısa süreli (5 dakika) ve uzun süreli (24 saat) KHD verileri üzerinden çalışılması tavsiye edilmiştir [19]. Bununla birlikte, yukarıdaki çalışmalarda da görüleceği üzere PAF teşhisi çalışmalarında farklı veri uzunlukları kullanılmaktadır. Şeker ve arkadaşları [36] tarafından yapılan bir çalışmada Poincare grafiği dışındaki doğrusal olmayan yöntemlerle elde edilen özniteliklerin geçerliliği test edilmiş ve 10000 örnekten az olan KHD verilerinden elde edilen ölçümlerin sağlıklı sonuç vermediği ortaya konulmuştur. Bu nedenle, bu çalışmada 30 dakikalık KHD verileri kullanılmıştır.

Dahası, 2004 yılında yapılan bir çalışmada, Hallstorm ve arkadaşları [37] tarafından kalp hızının ortalamasının özellikle zaman alanı KHD ölçümleri üzerindeki etkisini azaltmak için kalp hızını dakikada 75 vuru gibi sabit bir değere getirilmesini önermişlerdir. Bu işlem neticesinde elde edilen KHD verisine kalp hızı normalize edilmiş KHD (NKHD) verisi denilmektedir. 2009 yılında Isler ve Kuntalp [38] tarafından NKHD analizinin konjetif kalp yetmezliği teşhisi için sınıflandırıcı başarımını artırdığı gösterilmiştir. Bu nedenle, bu çalışmada kalp hızı normalizasyonu ve sınıflandırıcı başarımında etkili olduğu ispat edilmiş olan öznitelik normalizasyonu yöntemleri birlikte araştırılmıştır.

Tüm örüntü tanıma çalışmalarında öznitelik çıkarımından sonra probleme özgü olacak şekilde pek çok sınıflandırıcı kullanılmaktadır. Benzer şekilde KHD analizi kullanılan hastalık teşhisi çalışmalarında da k en yakın komşuluk (kNN), doğrusal ayraç analizi (LDA), karar ağacı (DT), bulanık mantık (FL), çok katmanlı algılayıcı (MLP), evrişimli sinir ağları (CNN) ve destek vektör makineleri (SVM) gibi sınıflandırıcılar sıklıkla kullanılmaktadır [13], [38]–[40]. Ayrıca PAF hastalığının teşhisi ve PAF hastalarında atak zamanının tespiti için literatürde 30 dakikalık veri ile çalışan pek çok sınıflandırıcı kullanılmış ve farklı performanslar elde edilmiştir.

Bu çalışmada da ele alınan iki görevden ilki olan PAF hastalığı teşhisi çalışmalarına bakacak olursak, Maier ve arkadaşları [41], LDA ve polinomal LDA kullanarak %80,00 başarıma ulaşmışlardır. Bir başka çalışmada Scherier ve arkadaşları, morfolojik EKG öznitelikleri ve bulanık mantık uzman sınıflandırıcı ile %82,00 başarıma ulaşmıştır [42]. Surucu ve arkadaşları [43] ise kNN sınıflandırıcı yanı sıra KHD normalizasyonu da kullanarak %86 doğruluk değerine ulaşmışlardır. Thong ve arkadaşları erken kulakçık

vurusu (PAC) analizinde karar ağacı kullanarak %90 başarıma ulaşmışlardır [44]. Ros ve arkadaşları ise kNN sınıflandırıcı ile %92 doğruluğa ulaşmışlardır [45]. Özcan ve Kuntalp aynı sınıflandırıcı ve genetik algoritmaya dayalı öznitelik seçimi sayesinde %92,20 doğruluk elde etmişlerdir [46]. Chesnekov ve arkadaşları [47] MLP sınıflandırıcı ile %95,5 doğrulukta yüksek bir başarıma ulaşırken; Martinez ve arkadaşları [48] ise olasılıksal dereceli azalma (Stochastic Gradient Descent – SGD) sınıflandırıcısı ile %96,05 doğruluk değerine ulaşmışlardır. Sürücü ve arkadaşları ise CNN temelli derin öğrenme sınıflandırıcı kullanarak %100 doğruluk değerine ulaşmışlardır [49].

Benzer şekilde çalışmamızdaki ikinci görev olan PAF atak zamanının tespiti amaçlı olarak literatürde pek çok farklı sınıflandırıcı ile çalışma yapılmıştır. Lynn ve Maier kNN sınıflandırıcı kullanarak %78 doğruluğa ulaşmışlardır [50]. Chesnokov [51], MLP sınıflandırıcı ile yaptığı çalışmada %82,05 doğruluk değerine ulaşırken, Surucu ve arkadaşları RBF (Radyal temelli fonksiyon – Radial based function) çekirdek fonksiyonlu SVM ile %83,67 doğruluğa ulaşmışlardır [52]. Boon ve arkadaşları yaptıkları iki çalışmada genetik algoritma ile birlikte SVM sınıflandırıcı kullanarak %83,90 ve %86,80 doğruluğa ulaşmışlardır [53], [54]. Diğer bir çalışmada Scherier ve arkadaşları FL kullanarak %84 doğruluk elde etmişlerdir [42]. DT sınıflandırıcı ile çalışan Zong ve arkadaşları [55] %88 doğruluğa, Thong ve arkadaşları [44] %89,29 doğruluğa ulaşırken, Costin ve arkadaşları [56] ise %89,40 doğruluk elde etmişlerdir. Maier ve arkadaşları [41] LDA sınıflandırıcı ile %92 doğruluğa ulaşırken son olarak çalışmamız öncesi en yüksek doğruluk değeri olan %94,50 ile Mohebbi ve Ghassemian [11] SVM sınıflandırıcıyla ulaşmışlardır. Sürücü ve arkadaşları ise yine CNN temelli derin öğrenme sınıflandırıcı kullanarak bu görev için de %100 doğruluk değerine ulaşmışlardır [57].

Bu çalışmada öncelikle AFPDB veritabanında bulunan 30 dakikalık EKG verileri işlenerek KHD verileri elde edilmiş, KHD verileri normalize edilerek NKHD veri seti de oluşturulmuştur. Oluşturulan bu iki farklı veri seti ile Shaffer ve Ginsberg [58] tarafından da önerilen yöntemlerle elde edilen özniteliklere normalizasyon yöntemleri de uygulanarak toplamda 6 farklı öznitelik seti ortaya çıkmıştır. Ortaya çıkan bu öznitelik setlerinin her biri ile çeşitli sınıflandırıcılar üzerinden sağlıklı / hasta ayrımı ile kısa sürede atak geçirmeyecek / kısa süre sonra atak geçirecek ayrımı yapılmıştır. Sınıflandırıcının türüne göre öznitelik seçimi, eğitim ve test verisi şeklinde veri bölümlemesi gibi başarım artışı ile performans artışı sağlayan ve sınıflandırıcı ezberlemesini önleyen önişlem

adımları uygulanmıştır. Ayrıca derin öğrenme sınıflandırıcısında ağın ezberlemesini (overfitting) önlemek için önerilen seyreltme katmanı da ağa eklenmiştir [59].

2. İNSAN FİZYOLOJİSİ VE KALP HIZI DEĞİŞKENLİĞİ

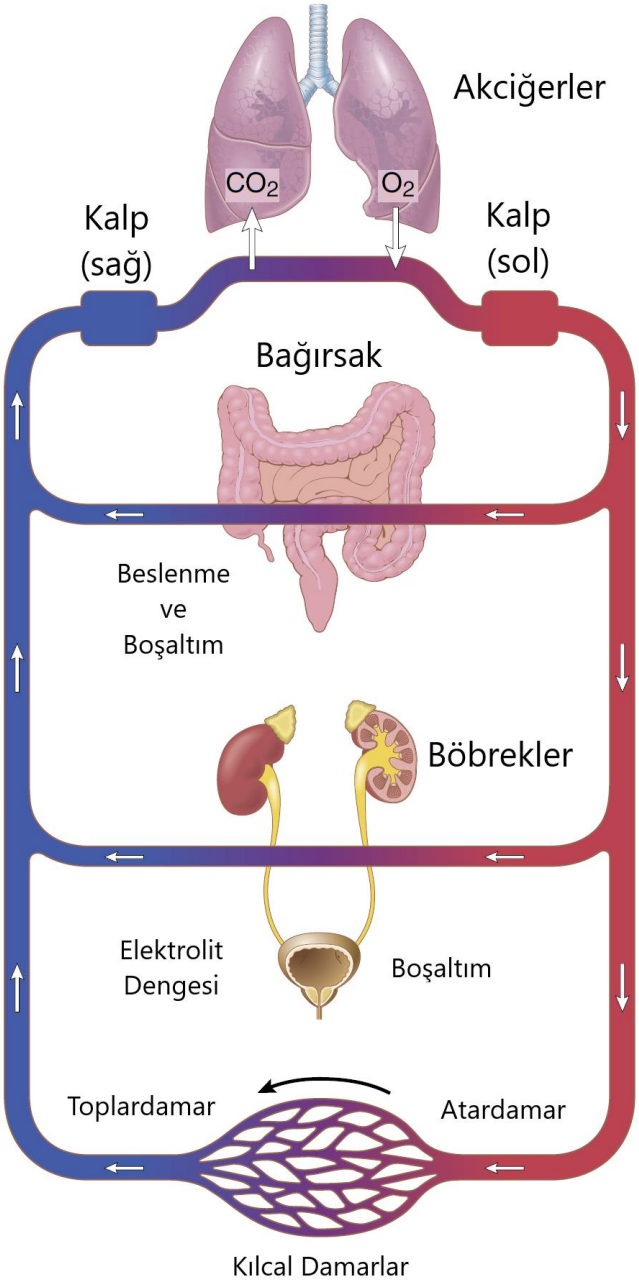
Tezin bu bölümünde kalbin çalışması ve elektriksel özelliklerinin yanısıra, KHD verisinin analizi için literatürde sıklıkla kullanılan işlemler hakkında bilgiler verilmiştir.

2.1. DOLAŞIM SİSTEMİ

Bütün canlı organizmalar hayatta kalabilmek için dışarıdan enerji kaynaklarına ve yapıtaşlarına ihtiyaç duyarlar. Vücudun ihtiyaç duyduğu enerji ve yapı kaynağı olan besin ve diğer maddeleri, organlara taşıyan sisteme kan dolaşım sistemi denmektedir. Şekil 2.1 kanın genel dolaşım sistemini gösterir. Hücre dışı sıvı vücutta iki aşamada taşınmaktadır. İlk aşama, kan damarlarındaki kanın vücuttaki hareketidir. İkincisi ise kan kılcal damarları ile doku hücreleri arasındaki hücreler arası boşluklardaki hareketidir [60]. Kan kılcal damarlardan geçerken, plazma sıvısı ile hücreler arası sıvı arasında transfer gerçekleşir ve gerekli besin ve ihtiyaç duyulan yapılar hücrelere bu şekilde ulaşır. Bu dolaşım sisteminin içinde kanın dolaşımını sağlayan organ kalptir. Kalpteki sorunlar pek çok rahatsızlığı tetiklemektedir.

2.2. KALBİN ÇALIŞMASI

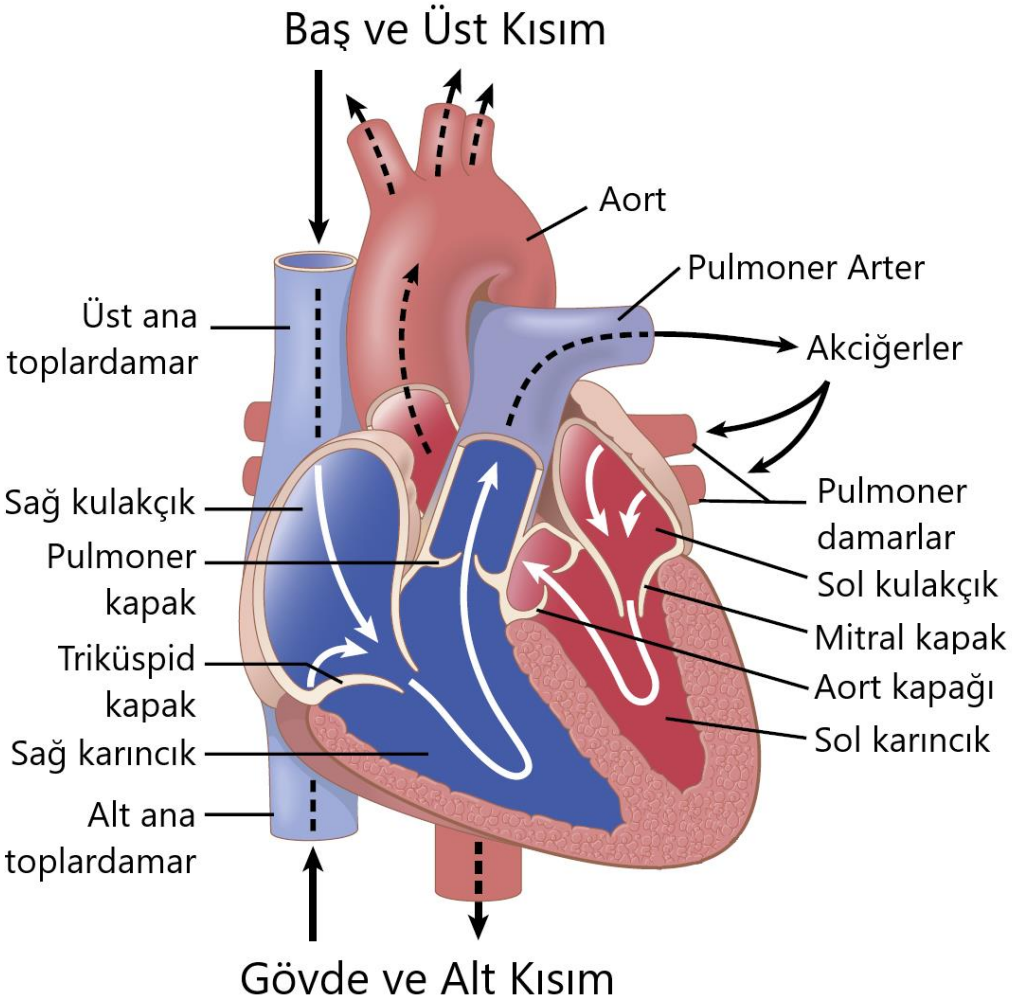
Kalp, vücudun hayatta kalması için gerekli olan kanın pompalandığı organdır. Şekil 2.2'de görüldüğü gibi dört kısımdan oluşmaktadır. Bunlar sağ ve sol kulakçıklar (atrium) ile sağ ve sol karıncıklardır (ventrikül). Vücutta kirlenmiş olan kan kalbin sağ kulakçığına gelir, kulakçıkların kasılması ile (P dalgası), kirli kan sağ karıncığa dolar. Kulakçıkların gevşeyip karıncıkların kasılmasıyla (QRS dalgası), kirli kan sağ karıncıktan akciğere pompalanır. Karıncıklar gevşerken de T dalgası oluşmaktadır. Şekil 2.3 söz konusu dalga formlarını EKG işareti üzerinde göstermektedir. Kirli kanın dolaşımına benzer şekilde temiz kan da akciğerlerden kalbin sol kulakçığına gelir, kulakçıkların kasılmasıyla temiz kan sol karıncığa dolar. Kulakçıkların gevşeyip karıncıkların kasılmasıyla temiz kan sol karıncıktan vücuda pompalanır.



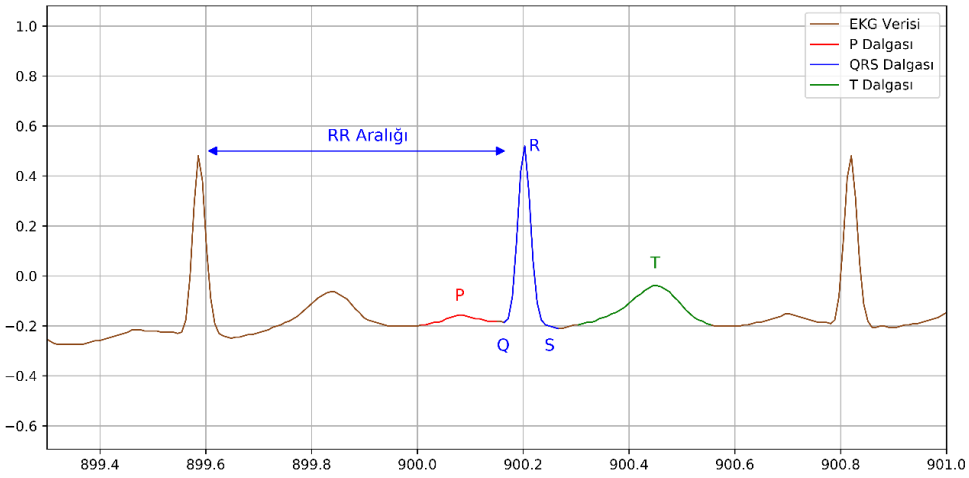
Şekil 2.1. Kan Dolaşım Sistemi [60].

Tüm bu kasılıp gevşemeler kalbin kendi elektriksel uyartılarıyla oluşmaktadır. Bu uyartımlar Sinoatriyal düğüm (SA Node) tarafından başlatılır ve sağlıklı bir kalpte önce internodal yollarla kulakçıkların kasılması sağlanır, ardından Atriyoventriküler düğüm (AV Node) tarafından ikinci uyartım belirli bir gecikme ile tetiklenir. Bu gecikme kalbe kanın tam dolmasını ve boşalmasını sağlar, sonrasında uyartım his demetlerinden ilerleyerek Purkinje lifleri aracılığıyla karıncıklar uyartılır. Sağlıklı bir kalpte örnek gecikme değerleri Şekil 2.4 üzerinde gösterilmiştir. Eğer bu iletim sisteminde aksamalar oluşursa kalp atımları düzensizleşir. Kulakçık temelli ve Karıncık temelli kalp atım

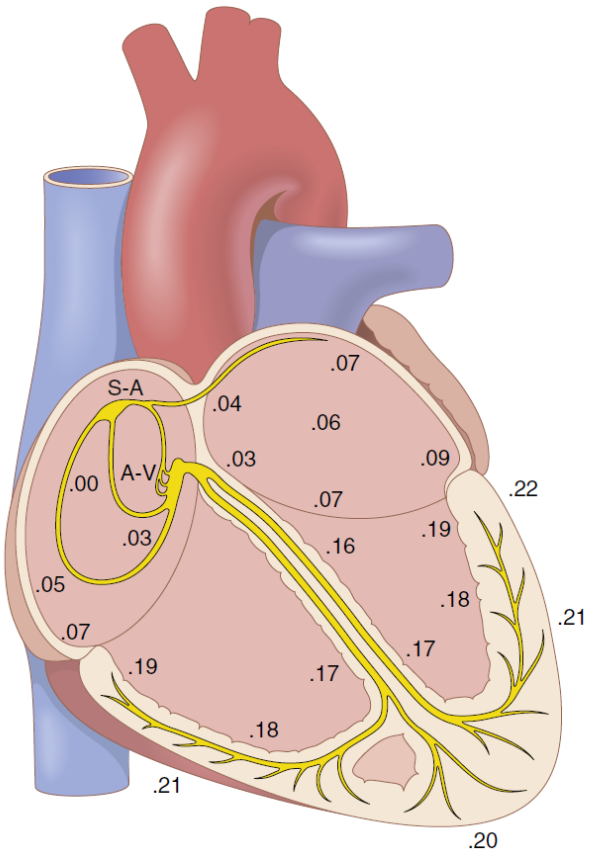
düzensizlikleri sonucu rahatsızlıklar gelişir. AF'ye SA düğüm dışında, kulakçıklarda düzensiz dağılan uyartımlar sebep olmaktadır.



Şekil 2.2. Kalbin yapısı [60].



Şekil 2.3. EKG işaretinde dalga formları.



Şekil 2.4. Kalpte uyarımın iletimi [60].

2.3. ARİTMİ

Aritmi, kalbin normal çalışma ritmi dışında çalışmasıdır ve pek çok türü bulunmaktadır. Bunların çoğu kalbin hızının değişmesi veya düzensizleşmesi ile alakalıdır. Kalp çarpma hızı dakikada 100 atım üzerine çıkarsa taşikardi, 60 atımın altına inerse bradikardi ve kalp atım düzeni bozulan taşikardi ise fibrilasyon olarak adlandırıldığı gibi; temel olarak kalbin üst bölümünü ilgilendiren aritmiler Supraventriküler Taşikardi (SVT), kalbin alt bölümünü ilgilendiren aritmiler ise Ventriküler Taşikardi (VT) olarak da isimlendirilir [61].

2.4. ATRİYAL FİBRİLASYON TÜRLERİ

AF, genel olarak dakikada 400'ün üzerinde kulakçık kaynaklı EKG aktivasyonunun olduğu aritmi türüdür [61]. Yaygın kabule göre dört farklı türü mevcuttur, Çizelge 2.1'de bu türler belirtilmiştir [62].

Çizelge 2.1. AF Türleri.

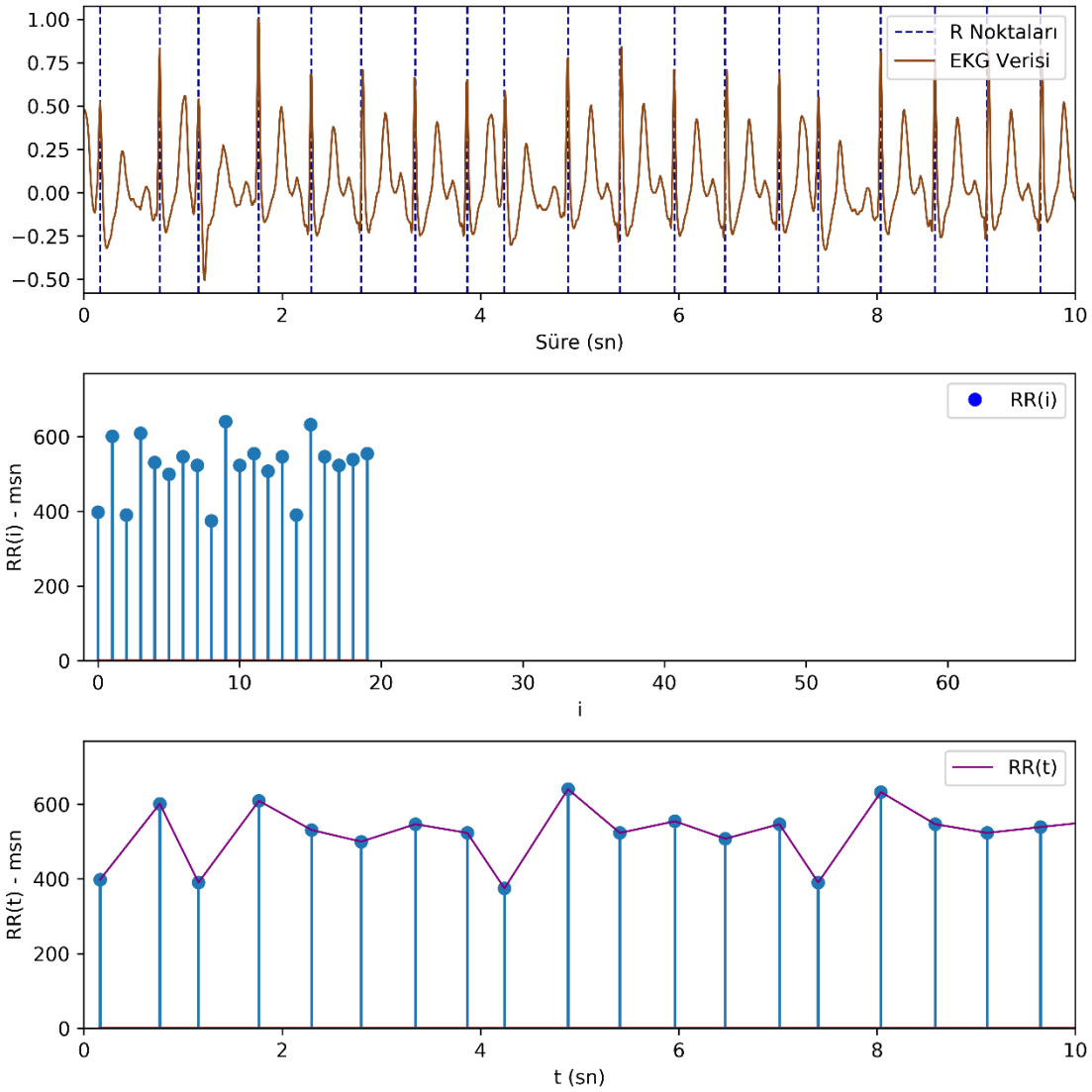
Terim	Açıklama
Paroksismal AF	Müdahale veya kendiliğinden 7 gün içinde sonlanan
Israrcı AF	7 günden uzun süren AF
Uzun Dönem Israrcı AF	12 aydan uzun süren AF
Kalıcı AF	NSR dönülemeyen veya tedavisi sonlandırılan AF

2.5. KALP HIZI DEĞİŞKENLİĞİ

KHD, ardışık kalp atışları arasında geçen sürenin zaman içindeki değişimini ifade etmektedir. KHD analizi ile kalp sağlığı ve OSS (Otonom Sinir Sistemi) durumu hakkında genel bir bilgi edinilebilmektedir [63]. İlk defa, anne karnındaki bebeklerin stres altında kalp atış hızının değiştiğinin anlaşılması sayesinde OSS ile KHD arasındaki bağlantı fark edilmiştir [64]. Sonrasında KHD analizi klinik çalışmalarda yaygın olarak kullanılan bir veri analizi türü olmuştur [20]–[26]. 1981 yılında Akselrod ve arkadaşları tarafından KHD'nin güç dağılımının belirli frekans bölgelerinde yoğunlaştığı gösterilmiştir [65]. Daha sonra KHD analizinde zaman alanı ölçümleri yanı sıra, frekans alanı ölçümleri, dalgacık dönüşümü ve doğrusal olmayan yöntemler gibi pek çok öznitelik çıkarım şekli de kullanılmıştır [11], [27]–[34]. Dahası 1996 yılında ESC ve NASPE Task Force isimli grup tarafından KHD analizi ile ilgili pek çok standart belirlenmiştir [19].

2.6. KHD VERİSİ

KHD verisinin elde edilmesi için kalp atışları arası geçen sürenin hesaplanması gerekmektedir. Bu sebeple EKG işaretinde en baskın dalga formu olan QRS dalgasının en yüksek genlikli noktası olan R tepe noktaları kullanılmaktadır. Tespit edilen R noktaları arasında geçen süreler KHD verisini oluşturmaktadır. ESC ve NASPE Task Force grubunun önerisi ve fazladan kalp atım anına ait bilgi içermesi sebebi ile KHD verisinin R tespit anları cinsinden fonksiyon olarak gösterilmesi tercih edilmektedir [19]. Şekil 2.5'de KHD verisi örneği gösterilmiştir. Burada RR(i) KHD veri katarı, RR(t) ise KHD fonksiyonudur. Genellikle RR(t) fonksiyonu işlenirken şekilde görüldüğü gibi ara değerler interpolasyonla belirlenmektedir [66].

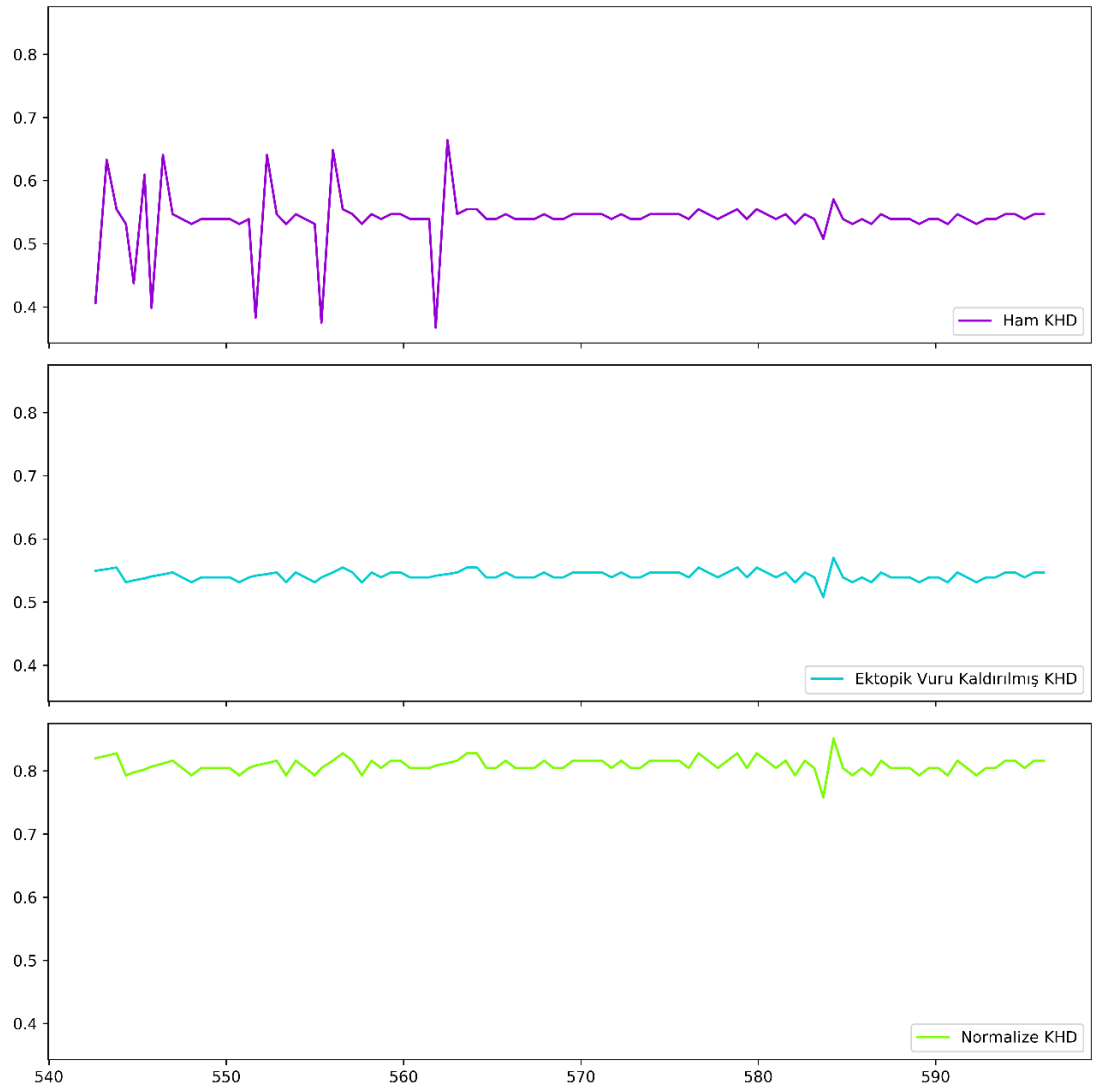


Şekil 2.5. KHD verisi elde edilmesi.

2.7. KHD NORMALİZASYONU

KHD verisinin OSS ile bağıntılı olduğu çalışmalarla gösterilmiştir [63]. Ancak kalp atış hızındaki değişim miktarları ve NSR durumundaki kalp hızı, kişiden kişiye değişmektedir. Ayrıca kişinin kalp hızını değiştiren aynı anda pek çok parametre de bulunabilmektedir. Bu sebeple teşhis edilmeye çalışılan hastalık veya atak anı tespiti işlemi için diğer tüm dış etkenlerin etkisini azaltma amaçlı ortalama kalp atım hızının 75 atım/dk olacak şekilde normalize edilmesi Hallstorm ve arkadaşları [37] tarafından önerilmiş ve bu çalışmada da gerçekleştirilmiştir. Bu sayede PAF hastalığının teşhisi ve PAF atak anının önceden kestirimi işlemlerinde, KHD verisinin normalize edilmesinin

sınıflandırıcı performansına katkısını incelemek hedeflenmiştir. Şekil 2.6'de örnek bir KHD verisi ile NKHD verisi gösterilmiştir.



Şekil 2.6. KHD verisi normalizasyonu.

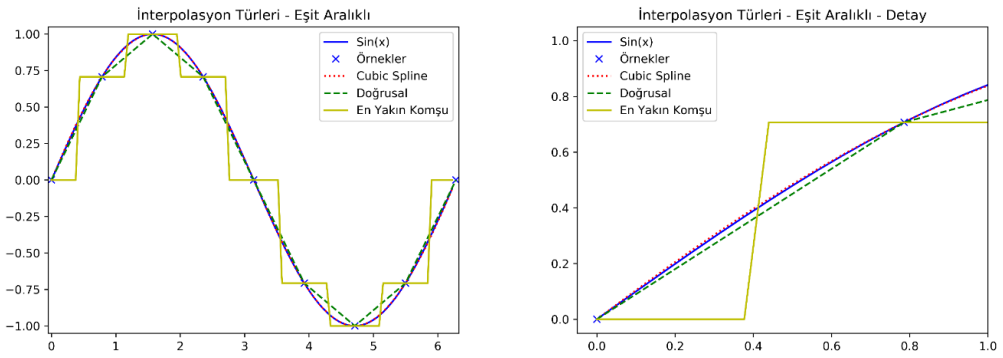
2.8. ÖNİŞLEMLER

İşaret işleme uygulamalarında gürültüyü işaretten ayırmak veya işaret işleme performansını artırmak amacıyla önişlemler sıklıkla uygulanır. KHD verilerine özgü olarak sıklıkla uygulanan önişlemler aşağıda anlatılmıştır.

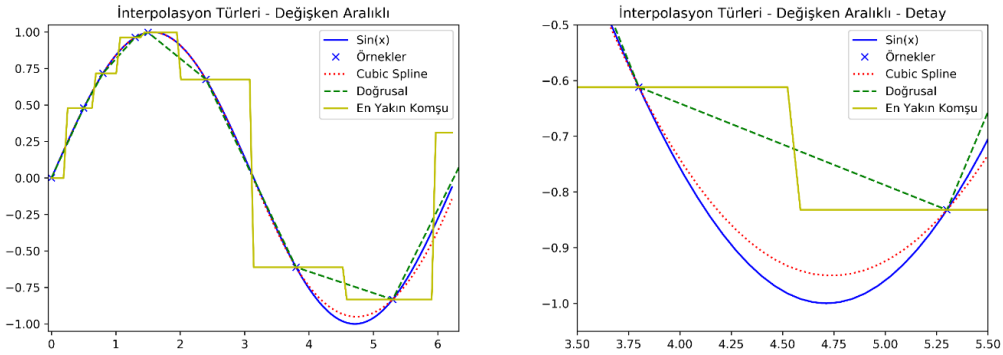
2.8.1. İnterpolasyon

KHD verileri doğası gereği sürekli olmayan ve eşit aralıklı örneklenmemiş işaretlerdir. Bu tarz işaretler üzerinde bazı matematiksel işlemleri gerçeklemek mümkün olmayabilir,

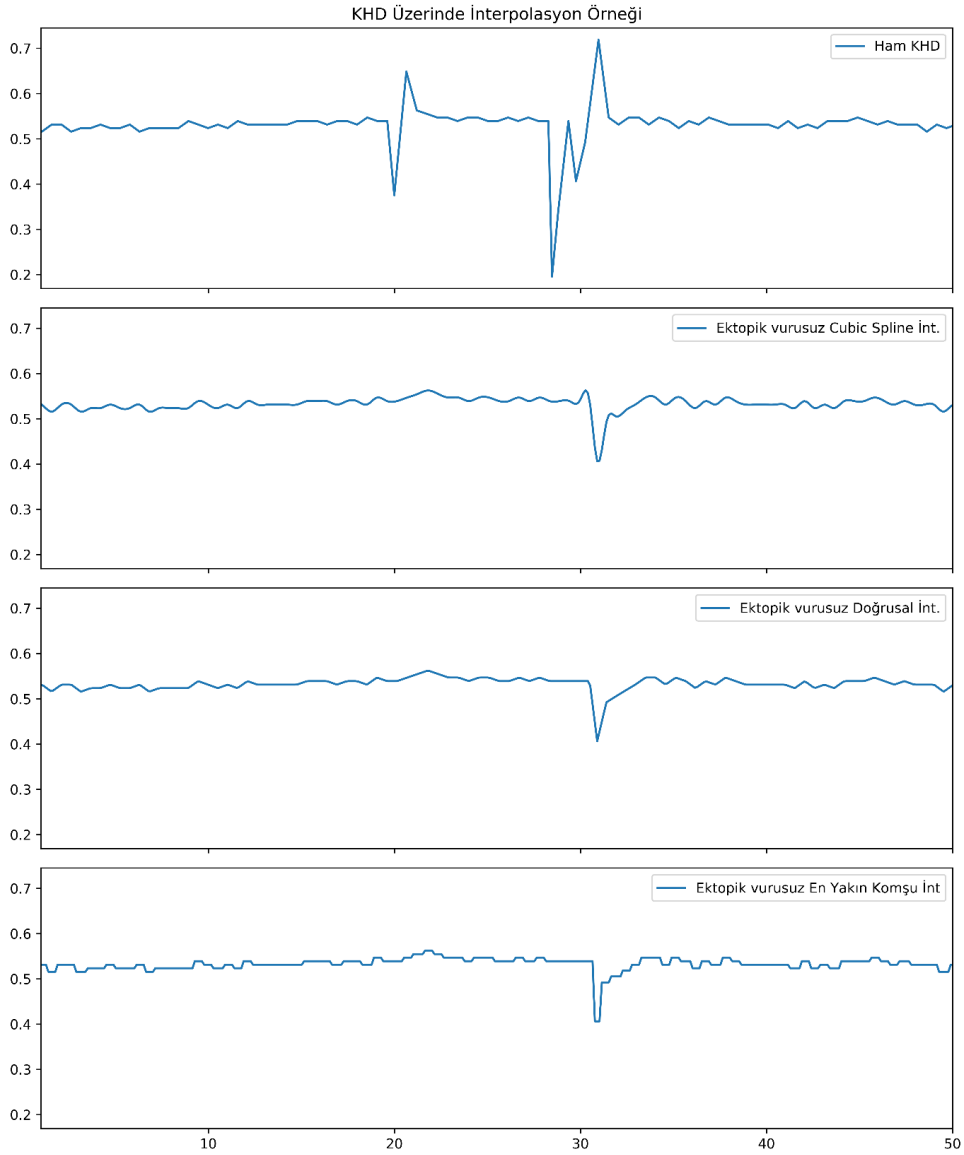
ya da işlem sonunda çok fazla gürültü oluşabilir. Bu sebeple yapılacak işleme göre KHD işaretlerini sürekli hale getirebilmek amaçlı interpolasyon işlemine tabi tutmak gerekmektedir. İnterpolasyon işleminde amaç, elde olan veri noktalarından yola çıkarak farklı noktalardaki verilerin tahmin edilmesidir. Eksik verilerin bulunduğu bölgeleri tahmin etmede kullanılan fonksiyon türüne veya metoda göre en yakın komşu, doğrusal, kübik gibi isimlendirilen interpolasyon çeşitleri bulunmaktadır. Örnek bir fonksiyonun (sin(x)) eşit aralıklı ayrık zamanlı örneklenmiş hali Şekil 2.7'de, değişken aralıklı örneklenmiş hali ise Şekil 2.8'de gösterilmiş olup, en yakın komşu, doğrusal ve kübik spline interpolasyonları şekillerde görülmektedir. Şekil 2.9'da ise ektopik vuru kaldırımı sonrası KHD işaretine interpolasyon uygulanması gösterilmiştir. KHD analizi işlemlerinde Cubic Spline metodu, daha düşük hata ürettiği için doğrusal interpolasyona göre tercih edilmektedir [67].



Şekil 2.7.Eşit örneklenmiş işarette interpolasyon örnekleri.



Şekil 2.8. Değişken aralıklı örneklenmiş işarette interpolasyon örnekleri.



Şekil 2.9. Ektopik vuru kaldırımında interpolasyon örnekleri.

2.8.2. Eğilim Yok Etme

KHD analizi uygulamalarında önerilen ve sıklıkla kullanılan bir diğer işlem eğilim yok etmedir. KHD analizi için zaman ve frekans alanında pek çok işlem gerçekleştirilmektedir. Bu işlemlerden biri de frekans bölgesi güç dağılımı (Power Spectral Density – PSD) bilgisinin çıkarılmasıdır. Güç dağılımı bilgisi çıkarılacak işaretin genelde zaman içinde durağan, eğilimsiz bir işaret olması beklenirken KHD verisi ise genellikle eğilimli bir işarettir [68]. Bu sebeple KHD işaretinin eğiliminin güç dağılımı işlemleri öncesinde yok edilmesi önerilmektedir. Tarvainen ve arkadaşlarının önerdiği Smoothness Prior yöntemi ile KHD işaretinin eğilimi yok edilebilmektedir [68]–[70]. KHD verisinde durağan olan periyodik işaret, eğimli işarete göre yüksek frekanslı

bileşenler içermektedir. Tavsiye edilen Smoothness Prior yöntemi zamanla değişen sonlu dürtü yanıtlı (FIR) yüksek geçiren filtre gibi işlemektedir. Bu eğilim yok etme işlemi sonrası KHD verisinin güç dağılımında 0-0.05 Hz aralığı için 90 dB'e kadar zayıflama gözlemlenebilmektedir [71].

$$Z = (R_2 - R_1, R_3 - R_2, \dots, R_N - R_{N-1},)^T \in \mathbb{R}^{N-1} \tag{2.1}$$

$$Z = Z_{durağan} + Z_{eğilim} \tag{2.2}$$

$$Z_{eğilim} = H\theta + \nu\,, \quad H \in \mathbb{R}^{(N-1)\times M} \vee \theta \in \mathbb{R}^M \tag{2.3}$$

$$\hat{\theta} = arg\min_{\theta} = \{\|H\theta - Z\|^2 + \lambda^2 \|D_d(H\theta)\|^2\} \tag{2.4}$$

$$\hat{\theta}_\lambda = (H^T H + \lambda^2 H^T D_d^T D_d H)^{-1} H^T Z \tag{2.5}$$

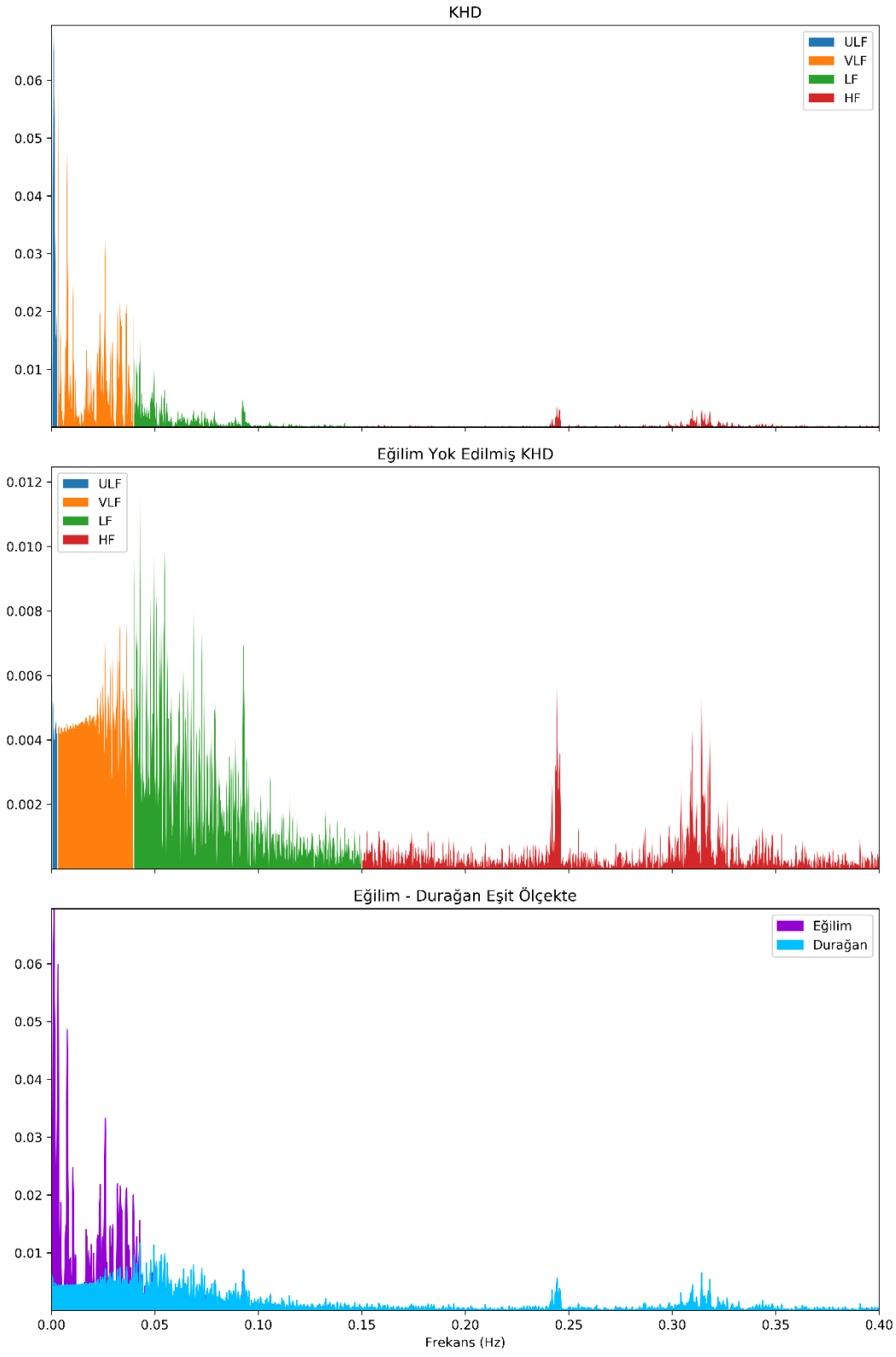
$$\hat{Z}_{eğilim} = H\hat{\theta}_\lambda \tag{2.6}$$

$$\hat{Z}_{durağan} = Z - H\hat{\theta}_\lambda = (I - (I + \lambda^2 D_2^T D_2)^{-1})Z \tag{2.7}$$

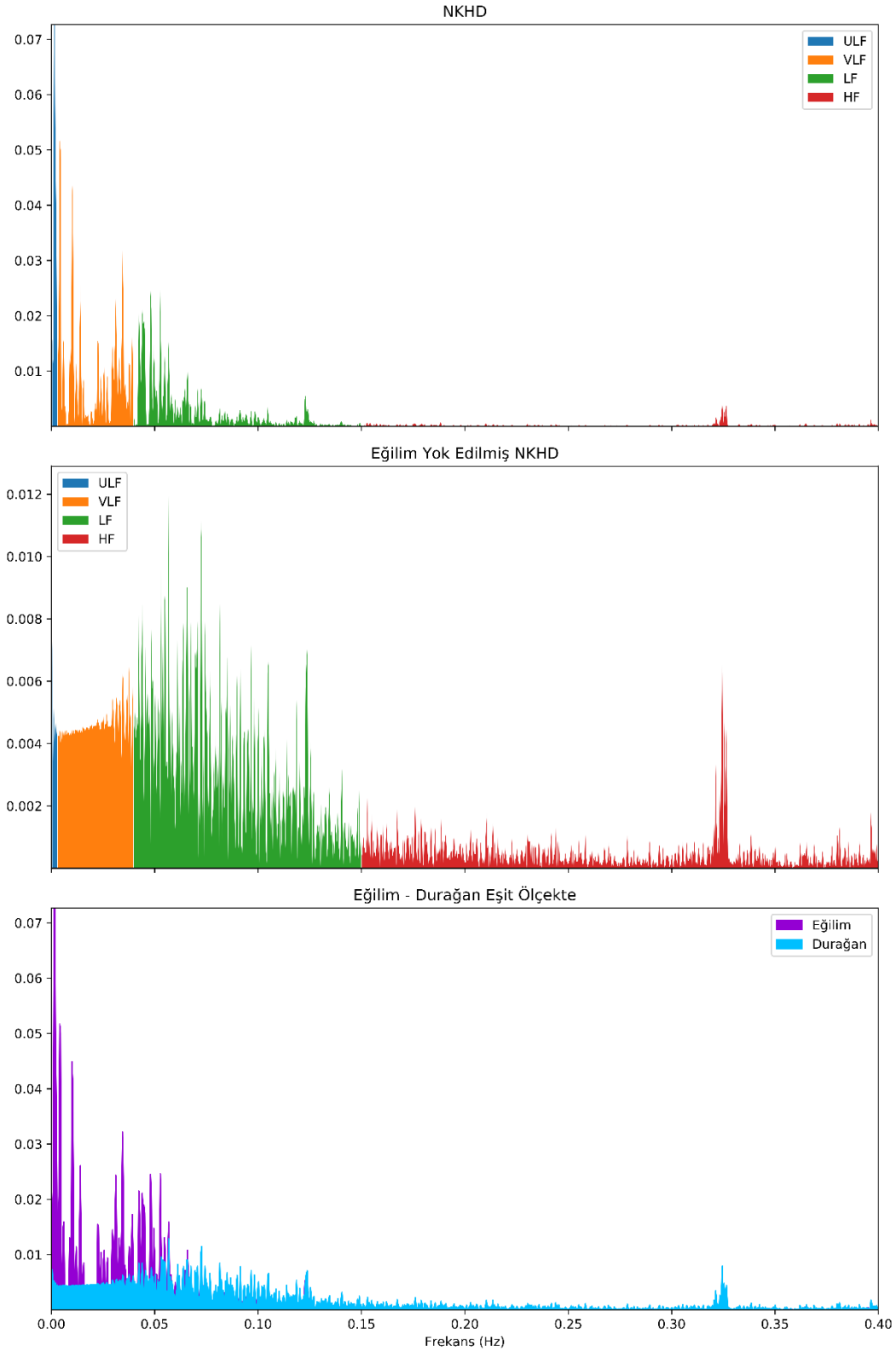
KHD verisini oluşturan R tepe noktaları arası geçen süreler, N adet tepe noktası için Denklem (2.1)'de olduğu gibi yazılabilmektedir. Z ifadesi temel olarak Denklem (2.2)'de olduğu gibi iki ayrı bileşenden oluşmaktadır. Burada yok etmek istediğimiz eğilimli işaret, düşük frekanslı periyodik olmayan işaret olarak ifade edilebilir (2.3). Burada H gözlem matrisini, θ regresyon parametresini ve ν ise gözlem hatasını göstermektedir. Regresyon parametresini düzenlileştirilmiş en küçük kareler (Regularized Least Squares - RLS) yöntemi ile tahmin etmeye çalışırsak Denklem (2.4) oluşur. Burada λ düzenlileştirme parametresi ve D_d ise d. mertebe türev operatörüdür. (2.4) denklemi çözümünden (2.5) ve (2.6) çıkarılabilir. Burada $\hat{Z}_{\text{eğilim}}$, yok edilecek olan tahmin edilen eğimdir. KHD işareti için gözlem matrisi yerine birim matris, D_d türev operatörü için ise $d = 2$ seçilirse (2.7) denklemi elde edilir.

Eğilim yok etme işlemine örnek LS periyodogramları Şekil 2.10'de görülmektedir. Şeklin ilk kısmında eğilim yok edilmemiş haliyle KHD verisine ait LS periyodogramı görülmektedir. İkinci kısımda ise eğilim yok edilmiş durağan KHD verisinin LS periyodogramı gösterilmiştir. İlk ve ikinci kısımdaki ölçek farkı yüzünden karşılaştırma yapılması zor olacağı için son kısımda Eğilim işareti ve Durağan işaret birlikte çizilmiştir.

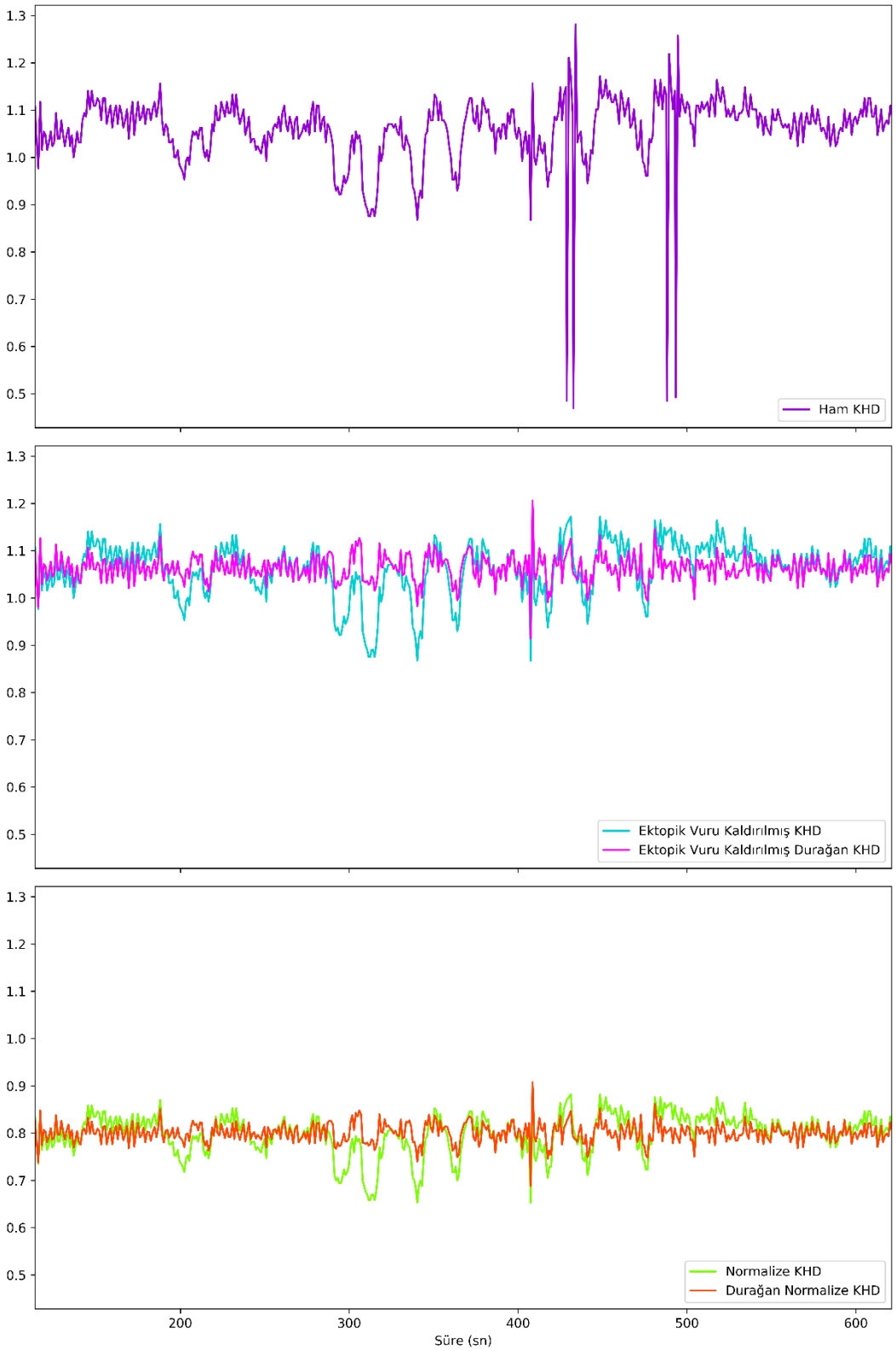
Benzer işlemler Normalize edilmiş KHD olan NKHD işareti için de Şekil 2.11'de gerçekleştirilmiştir. Şekil 2.12'da ise eğilim yok etme işlemi öncesi ve sonrası KHD ve NKHD işaretleri görülmektedir.



Şekil 2.10. KHD eğilim yok etme işlemi için LS periyodogramları.



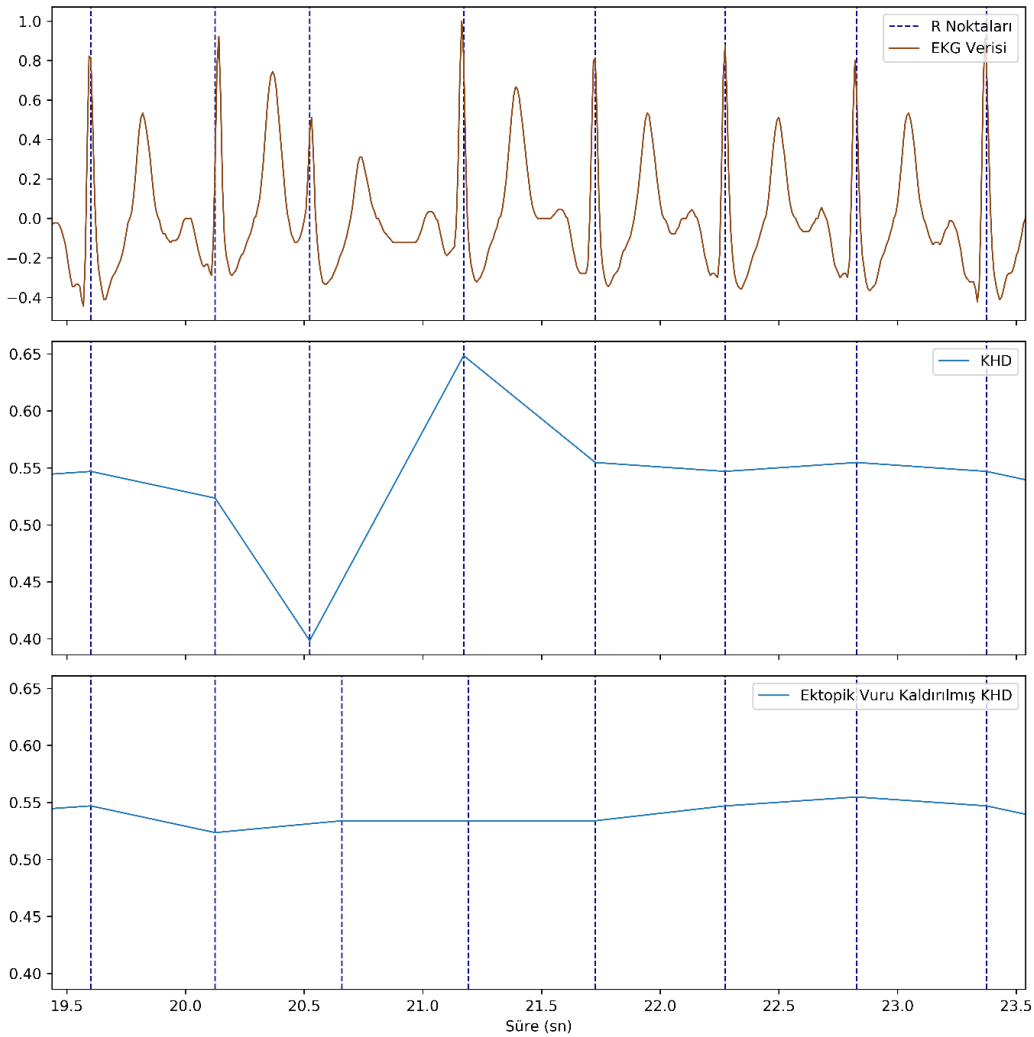
Şekil 2.11. NKHD eğilim yok etme işlemi için LS periyodogramları.



Şekil 2.12. Eğilim yok etme sonucu KHD ve NKHD işaretleri.

2.8.3. Ektopik Vuruların Kaldırılması

Sinoatriyal düğüm kaynaklı olmayan vurular ektopik vuru olarak adlandırılmaktadır. Kısa süreli bu vurular, sağlıklı bireylerde de oluşabildiği için KHD verisinden kaldırılmıştır. Bir vurunun ektopik vuru olup olmadığına karar verirken ilk bakılması gereken nokta, mevcut KHD değerinin belirli bir aralıktaki ortalama KHD değerine kıyasla en az %20 altına inmesi gerekmesidir. Ortalama KHD değerinin %20 altına inen şüpheli atımdan bir sonraki atım ortalama KHD değerinin ±%10 içinde kalırsa şüpheli atım kulakçık kaynaklı ektopik vuru olarak değerlendirilir. Eğer takip eden atım süresi ortalama KHD değerinden %30 fazlaysa şüpheli vuru, karıncık (ventricular) kaynaklı ektopik vuru olmaktadır [72]. Şekil 2.13'de ektopik vuruların tespit edilmesi ve kaldırılması görülmektedir. Kaldırılan vurular yerine doğrusal interpolasyon ile yeni vurular eklenmektedir.



Şekil 2.13. Ektopik vuruların kaldırılması.

2.9. ÖZNİTELİKLER

1996 yılında ESC ve NASPE Task Force isimli grup tarafından KHD analizi ile ilgili kullanılabilecek özniteliklerin bir çoğu tanımlanmıştır [19]. Sonrasında bu öznitelikler ve fazlası literatürdeki çalışmalarda farklı kombinasyonlarda kullanılmıştır. Örüntü tanıma işlemlerinde sınıflandırma performansının ana belirleyicilerinden birisi de öznitelik uzayının doğru seçilmesidir.

2.9.1. Zaman Bölgesi Ölçümleri

Zaman bölgesi ölçümleri, RR veya NN olarak isimlendirilen KHD verisi üzerinden temel istatistiki büyüklüklerin çıkarılmasıyla elde edilir [19]. Diğer özniteliklere göre hesaplanması görece daha basit olmasına rağmen sınıflandırıcı performansını artırmaktadır. Denklem (2.8) - (2.23) arasında KHD analizinde sıklıkla kullanılan zaman bölgesi ölçümleri gösterilmiştir. Burada RR veya NN, R tepe noktaları arası geçen süre veya atımlar arası geçen süre anlamında olup aynı bilgiyi işaret etmektedir. HR ise dakikadaki kalp vurum sayısıdır. Çizelge 2.2'de zaman bölgesi öznitelikleri açıklanmıştır.

$$MNN = \overline{RR} = \frac{1}{N}\sum_{i=1}^{N} RR_i \tag{2.8}$$

$$SDNN = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(RR_i - \overline{RR})} \tag{2.9}$$

$$RMSSD = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}(RR_{i+1} - RR_i)^2} \tag{2.10}$$

$$SDSD = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}(|RR_{i+1} - RR_i| - \overline{RR})^2} \tag{2.11}$$

$$NN50 = \sum_{i=1}^{N-1}\{|RR_{i+1} - RR_i| > 50ms\} \tag{2.12}$$

$$pNN50 = \frac{NN50}{N} x100 \tag{2.13}$$

$$NN20 = \sum_{i=1}^{N-1} \{|RR_{i+1} - RR_i| > 20ms\} \tag{2.14}$$

$$pNN20 = \frac{NN20}{N} x100 \tag{2.15}$$

$$MEDNN = NN_{N/2}, NN\ sıralı \tag{2.16}$$

$$RANGENN = max\, NN - min\, NN \tag{2.17}$$

$$CVSD = \frac{RMSSD}{MNN} \tag{2.18}$$

$$CVNN = \frac{SDNN}{MNN} \tag{2.19}$$

$$MHR = \overline{HR} = \frac{1}{N} \sum_{i=1}^{N} HR_i \tag{2.20}$$

$$MAXHR = max\, HR_i \tag{2.21}$$

$$MINHR = min\, HR_i \tag{2.22}$$

$$SDHR = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (HR_i - \overline{HR})} \tag{2.23}$$

Çizelge 2.2. Zaman bölgesi ölçümleri.

Öznitelik	Açıklama
MNN (ms)	Ortalama KHD değeri
SDNN (ms)	KHD verisinin standart sapması
RMSSD (ms)	KHD değişiminin RMS değeri
SDSD (ms)	KHD değişiminin standart sapması
NN50 (adet)	KHD değişimi 50 ms den fazla olanların sayısı
PNN50 (%)	KHD değişimi 50 ms den fazla olanların oranı
NN20 (adet)	KHD değişimi 20 ms den fazla olanların sayısı
PNN20 (%)	KHD değişimi 20 ms den fazla olanların oranı
MEDNN (ms)	KHD ortanca değeri
RANGENN (ms)	KHD değişim aralığı
CVSD (ms^2)	KHD değişiminin varyasyonu
CVNN (ms^2)	KHD verisinin varyasyonu
MHR (bpm)	Kalp atış hızı ortalaması
MAXHR (bpm)	En büyük kalp atış hızı
MINHR (bpm)	En küçük kalp atış hızı
SDHR (bpm)	Kalp atış hızının standart sapması

2.9.2. Frekans Bölgesi Ölçümleri

1981 yılında Akselrod ve arkadaşları tarafından yapılan çalışmada, OSS ile KHD frekans alanı ölçümleri arasındaki bağlantı fark edilmiştir [65]. 1996 yılında ESC ve NASPE Task Force isimli grup tarafından yapılan KHD ölçüm standartları belirleme çalışmasında da frekans bölgesi güç dağılımları bölgelere ayrılmıştır. Frekans bölgesinin 5dk gibi kısa süreli KHD verilerinde 3 bölgeye, 24 saat gibi uzun süreli verilerde ise 4 bölgeye ayrılması önerilmiştir. Ayrıca incelenmesi düşünülen en düşük frekans bölgesinin dalga boyunun en az 10 katı uzunlukta veri ile çalışılması tavsiye edilmiştir. Bu sebeple ULF bölgesi ölçümleri yapılacağı zaman en az 27dk KHD verisi gerekli olmaktadır [19]. Frekans bölgesi özellikleri için FFT (Fast Fourier Transform), LS (Lomb-Scargle) ve diğer periodogramlar kullanılmaktadır [29]. Bu ölçümlerde, belirli bir frekans bölgesindeki KHD sinyalinin güç spektral yoğunluğu (PSD), öznitelik olarak kullanılmaktadır [30].

2.9.2.1. *Lomb-Scargle Periyodogram*

Lomb – Scargle periyodogramı (LS), Lomb [73] tarafından önerilmiş ve Scargle [74] tarafından geliştirilmiş, değişken aralıklı örneklenmiş zaman serisi verilerinin eğilim yok etmeye veya tekrar örneklemeye gerek kalmadan güç spektral yoğunluğunu hesaplamaya yarayan yöntemdir. Denklem (2.24) ve (2.25)'te periyodogramın hesaplanması gösterilmiştir. Fourier analizi ve en küçük kareler yöntemine dayanması ve değişken aralıklı örneklenmiş işaretlerde de kullanılabilmesi, bu yöntemi oldukça popüler hale getirmiştir [75].

$$P_X(\omega) = \frac{1}{2}\left\{\frac{\left[\sum_{j=0}^{N-1} X_j \cos\omega(t_j-\tau)\right]^2}{\sum_{j=0}^{N-1}\cos^2\omega(t_j-\tau)} + \frac{\left[\sum_{j=0}^{N-1} X_j \sin\omega(t_j-\tau)\right]^2}{\sum_{j=0}^{N-1}\sin^2\omega(t_j-\tau)}\right\} \tag{2.24}$$

$$\tau \equiv \frac{1}{2\omega}\tan^{-1}\left(\frac{\sum_{j=1}^{N}\sin 2\omega t_j}{\sum_{j=1}^{N}\cos 2\omega t_j}\right) \tag{2.25}$$

2.9.2.2. *Hızlı Fourier Dönüşümü*

Hızlı fourier dönüşümü (FFT), ayrık fourier dönüşümünün (DFT) hesap karmaşıklığını azaltmak adına kullanılan algoritmalardır. Ayrık fourier dönüşümünün hızlı hesaplanması için yaklaşımlar 1805'li yıllara kadar dayanmasına rağmen anlamlı derecede işlem karmaşıklığının düşürülmesi 1965'li yılları bulmuştur [76]. Cooley ve Tukey'in FFT algoritması sayesinde DFT işlem karmaşıklığı $O(N^2)$'den $O(N\log N)$ değerine düşmüştür [77].

DFT kompleks fourier serisi Denklem (2.26)'da tanımlanmış olup Denklem (2.27)'de ki şekilde dönüşüm uygulanırsa seri Denklem (2.28)'de olduğu gibi ifade edilebilir.

$$X(j) = \sum_{k=0}^{N-1} A(k)\cdot W^{jk}, \qquad j = 0,1,\dots,N-1 \tag{2.26}$$

$$W = e^{-2\pi i/N}$$

$$N = r_1 \cdot r_2 \tag{2.27}$$

$$\begin{aligned} j &= j_1 r_1 + j_0, \qquad j_0 = 0,1,\dots,r_1-1, \qquad j_1 = 0,1,\dots,r_2-1 \\ k &= k_1 r_2 + k_0, \qquad k_0 = 0,1,\dots,r_2-1, \qquad k_1 = 0,1,\dots,r_1-1 \end{aligned} \tag{2.28}$$

$$X(j_1, j_0) = \sum_{k_0} \sum_{k_1} A(k_1, k_0) \cdot W^{jk_1 r_2} W^{jk_0}$$

Burada (2.27) denklemini tekrar düzenler ve daha fazla çarpan şeklinde ifade edebilirsek;

$$N = r_1 \cdot r_2 \cdots r_m \tag{2.29}$$

şeklinde yazabiliriz. Burada tüm r değerleri eşit olursa karmaşıklık $O(N \log_r N)$ değerine düşecektir. Bilgisayar aracılığıyla yapılan hesaplamalarda performans açısından da fayda sağladığı için $r = 2$ seçilmektedir. Büyük N değerleri içinse split-radix yöntemi ile daha iyi hesaplama performansları elde edilebilmektedir [78].

2.9.2.3. *Welch Periyodogram*

Parametrik olmayan, büyük sinyal uzunluklarında iyi performans sağlayan Welch periyodogramı, işareti üst üste binmiş alt gruplara bölerek gruplara ait periyodogramların ayrı ayrı alınması ile hesaplanır [79]. Bunun için öncelikle $X(j)$ işaretinin periyodogramı oluşturulurken, Denklem (2.30)'da yazıldığı gibi L uzunluklu K alt işaret cinsinden yazılması gerekmektedir. Denklem (2.31)'de ifade edilen D ise, K adet alt grubun kaç elemanının üst üste bindiğini göstermektedir. Her X_K işaretine ait Fourier dönüşümü (2.32)'de gösterildiği gibi hesaplanabilmektedir. Buradan yola çıkarak her işarete ait periyodogram (2.33)'de gösterildiği gibi f_n ve U cinsinden ifade edilerek yazılabilir. Sonuç olarak $X(j)$ işaretine ait periyodogram yaklaşık olarak (2.34)'de gösterildiği gibi hesaplanabilmektedir.

$$\begin{aligned} X_1(j) &= X(j), \qquad j = 0,1, \dots, L-1 \\ X_2(j) &= X(j+D), \qquad j = 0,1, \dots, L-1 \\ X_K(j) &= X(j+(K-1)D), \qquad j = 0,1, \dots, L-1 \end{aligned} \tag{2.30}$$

$$N = (K-1)D + L \tag{2.31}$$

$$A_k(n) = \frac{1}{L} \sum_{j=0}^{L-1} X_k(j) W(j) e^{-2kijn/L} \tag{2.32}$$

$$I_k(f_n) = \frac{L}{U} |A_k(n)|^2 , \qquad k = 1,2, \cdots, K \tag{2.33}$$

$$f_n = \frac{n}{L}, \qquad n = 0, \cdots, L/2$$

$$U = \frac{1}{L}\sum_{j=0}^{L-1} W^2(j)$$

$$P(f_n) = \frac{1}{K}\sum_{k=1}^{K} I_k(f_n) \tag{2.34}$$

2.9.2.4. *Sürekli Dalgacık Dönüşümü*

Dalgacık dönüşümü, bir işareti hem frekans hem de zaman bölgesinde aynı anda incelemeye yarayan bir dönüşüm metodudur. Sürekli zamanda ve ayrık zamanda işaretler için benzer sonuçlar içeren farklı yaklaşımları vardır. Bir ana dalgacık fonksiyonu ile işaretin ilişkisi olarak tanımlanmaktadır. Ana dalgacık fonksiyonu, zamanda sınırlı ve ortalaması sıfır olan bir dalga formudur. Dalgacık fonksiyonunun $\psi(t)$ integrali sıfır olmalı (2.35), enerjisi ise birim enerjiye eşit olmalıdır (2.36). Ayrıca tüm aralık içerisinde sınırlı tanımlı olmalıdır (2.37). $\hat{\psi}(\omega)$, dalgacık fonksiyonunun fourier dönüşümü olmak üzere (2.38) koşulunu da sağlaması gerekmektedir. Bu 4 koşulu sağlayan dalga formundaki işarete ana dalgacık fonksiyonu denmektedir [80]. Daubechies, Coiflet, Symlet gibi adlandırılan dalgacık fonksiyonları sıklıkla kullanılmaktadır. Şekil 2.14'te ana dalgacık fonksiyonları örnekleri gösterilmiştir. Burada dalgacık ailesinden sonra gelen rakam o fonksiyonun mertebesini göstermektedir. Fonksiyonun mertebesi aynı zamanda dalgacık fonksiyonuna ait moment sayısıdır [81]. Moment sayısı ile dalgacık fonksiyonu ilişkisi (2.39)'da verilmiştir.

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0 \tag{2.35}$$

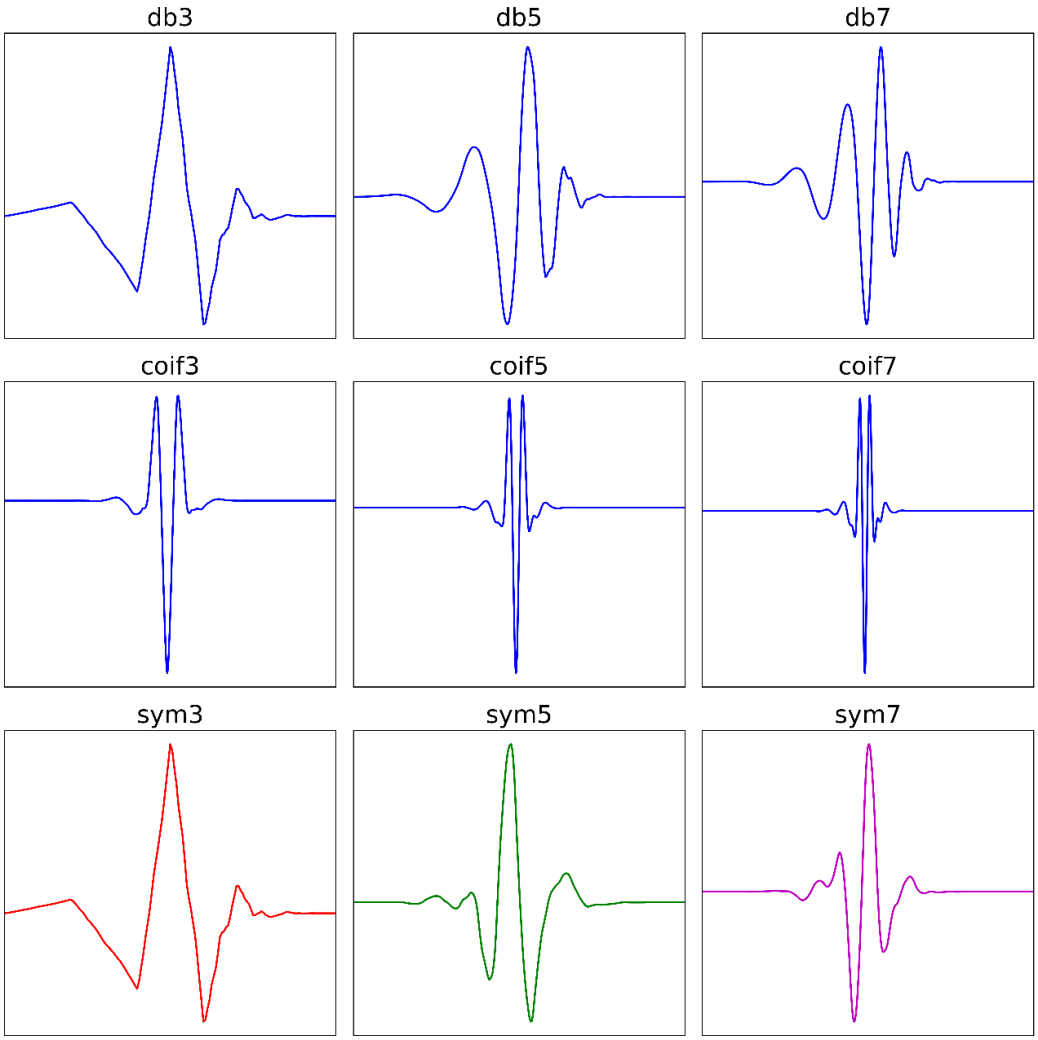
$$\int_{-\infty}^{+\infty} \psi^2(t)dt = 1 \tag{2.36}$$

$$\int_{-\infty}^{+\infty} |\psi(t)|dt < \infty \tag{2.37}$$

$$\hat{\psi}(\omega) = \int_{-\infty}^{+\infty} \psi(t)e^{-j\omega t}dt \tag{2.38}$$

$$\int_{-\infty}^{+\infty} \frac{|\psi(t)|^2}{|\omega|} d\omega < \infty$$

$$\int_{-\infty}^{+\infty} t^k \psi(t) dt = 0\,, \qquad 0 \le k < N \tag{2.39}$$



Şekil 2.14. Daubechies, Coiflet ve Symlet ana dalgacık fonksiyon örnekleri.

2.9.2.5. *Ayrık Dalgacık Dönüşümü*

Ayrık zamanlı işaretlerin zamansal çözünürlük incelemeleri için dalgacık dönüşümleri Ayrık Dalgacık Dönüşümü (ADD) yöntemleri ile yapılmaktadır. İlk ADD, Alfred Haar tarafından geliştirilmiş olup Haar Dalgacık Dönüşümü (HDD) olarak da adlandırılmaktadır [82]. HDD en basit ADD metodudur. Ayrık bir işaret için Denklem (2.40)'da verilen formülle hesaplanır. Burada H_n, Haar dönüşüm matrisi, gerçel ve ortogonaldir. Dolayısıyla Denklem (2.41)'deki gibi yazılabilir. Haar dönüşüm matrisi

Denklem (2.42)'de yazıldığı gibi hesaplanabilmektedir. Temel olarak ADD iki farklı filtre ile temsil edilerek Şekil 2.15'te gösterildiği gibi hesaplanabilmektedir.

$$y_n = H_n x_n \tag{2.40}$$

$$\begin{aligned} H &= H^* \\ H^{-1} &= H^T \\ HH^T &= I \end{aligned} \tag{2.41}$$

$$\begin{aligned} H_n &= \frac{1}{\sqrt{2}}\begin{bmatrix} I_{n-1} & 0 \\ 0 & H_{n-1} \end{bmatrix} \\ H_1 &= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \end{aligned} \tag{2.42}$$

Belçikalı Matematikçi Ingrid Daubechies, HDD geliştirerek Daubechies Dalgacık Dönüşümünü (DDD) sunmuştur [83]. DDD halen en çok kullanılan ADD yöntemidir. DDD çok sayıda ana dalgacık dönüşüm fonksiyonunu barındıran bir aileyi kapsamaktadır. Bu ailenin ilk üyesi ise Haar dalgacık fonksiyonudur. Dalgacık fonksiyonlarının derecesi arttıkça çözünürlüğü de katlanmaktadır. Daubechies dalgacık fonksiyonu seçilirken ya kaybolma momentleri üzerinden (dbA), ya da katsayı adedi üzerinden (DN) isimlendirme yapılmaktadır. Katsayı adedi kaybolma momentinin 2 katı olduğu için örneğin db2=D4 olmaktadır. db1 dalgacığı Haar dalgacığıdır ve 2 katsayıya sahiptir. Örneğin db2 dalgacık katsayılarını (2.43) hesaplamak istersek ortogonallik koşullarından (2.44)'ü elde ederiz.

$$l(n) = \{l(0), l(1), l(2), l(3)\} \tag{2.43}$$

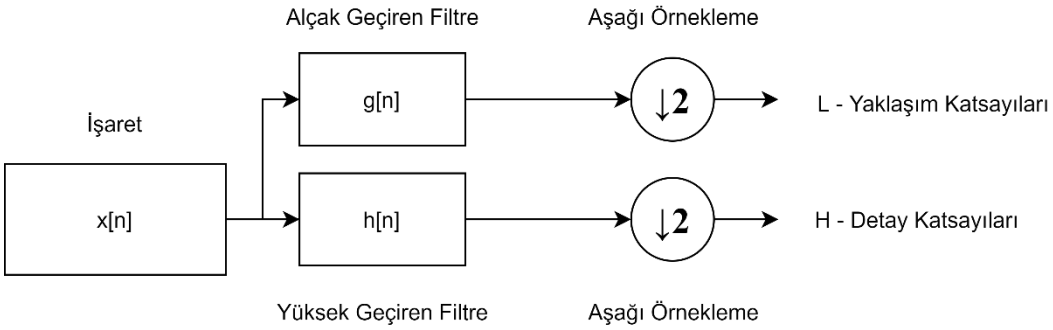
$$\begin{aligned} l^2(0) + l^2(1) + l^2(2) + l^2(3) &= 1 \\ l(0)l(2) + l(1)l(3) &= 0 \end{aligned} \tag{2.44}$$

Denklemleri çözersek alçak geçiren filtre $g(n)$ katsayıları Denklem (2.45)'teki gibi bulunur.

$$g(n) = \{l(0), l(1), l(2), l(3)\}$$
$$= \frac{1}{4\sqrt{2}}\{(1-\sqrt{3}), (3-\sqrt{3}), (3+\sqrt{3}), (1+\sqrt{3})\} \quad (2.45)$$

Benzer şekilde yüksek geçiren filtre $h(n)$ katsayıları da Denklem (2.46)daki gibi bulunacaktır.

$$h(n) = \{-l(3), l(2), -l(1), l(0)\}$$
$$= \frac{1}{4\sqrt{2}}\{-(1+\sqrt{3}), (3+\sqrt{3}), -(3-\sqrt{3}), (1-\sqrt{3})\} \quad (2.46)$$

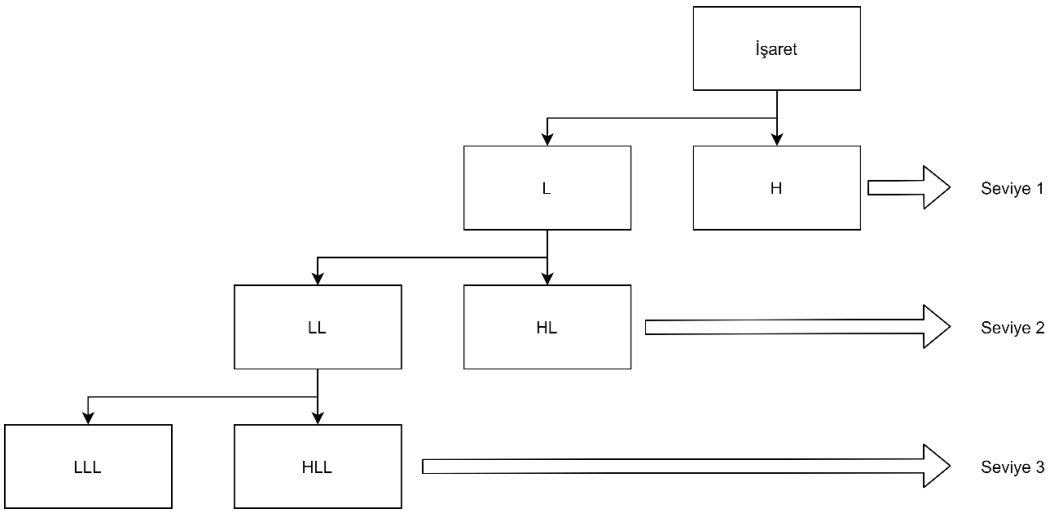


Şekil 2.15. Ayrık dalgacık dönüşümü gösterimi.

ADD işlemleri neticesinde işaretimiz alçak frekanslı ve yüksek frekanslı bileşenler içeren iki farklı işarete dönüşmektedir. Her bir dönüşümden yüksek frekanslı bileşenleri içeren kısma ait katsayılar Detay katsayıları, düşük frekanslı bileşenleri içeren kısma ait katsayılar ise Yaklaşım Katsayıları olarak isimlendirilmektedir. Mallat tarafından önerilen dönüşüm ağaç yapısı bu katsayıları hesaplamakta sıklıkla kullanılmaktadır [84]. Mallat ağacının blok gösterimi Şekil 2.16'da verilmiş olup örnek KHD işaretine ait dönüşümler Şekil 2.17'de sunulmuştur. Bu elde edilen katsayılar ile işaretin belirli frekans bandındaki güç değerleri Parseval teoremine göre hesaplanabilmektedir [85]. Parseval teoremine göre 1 Ω direnç üzerinden geçen $x[n]$ ayrık akımının gücü a_k Fourier katsayıları cinsinden Denklem (2.47) ile hesaplanabilmektedir. Benzer şekilde işaretin ADD hesaplanarak işarette yerine konursa Denklem (2.48) ile yaklaşık güç değerleri hesaplanabilmektedir.

$$\frac{1}{N}\sum_{N}|x[n]|^2 = \sum_{N}|a_k|^2 \tag{2.47}$$

$$P_j = \frac{1}{N_j}\sum_{k}\left|w_{j,k}\right|^2 = \frac{\left\|w_j\right\|^2}{N_j} \tag{2.48}$$



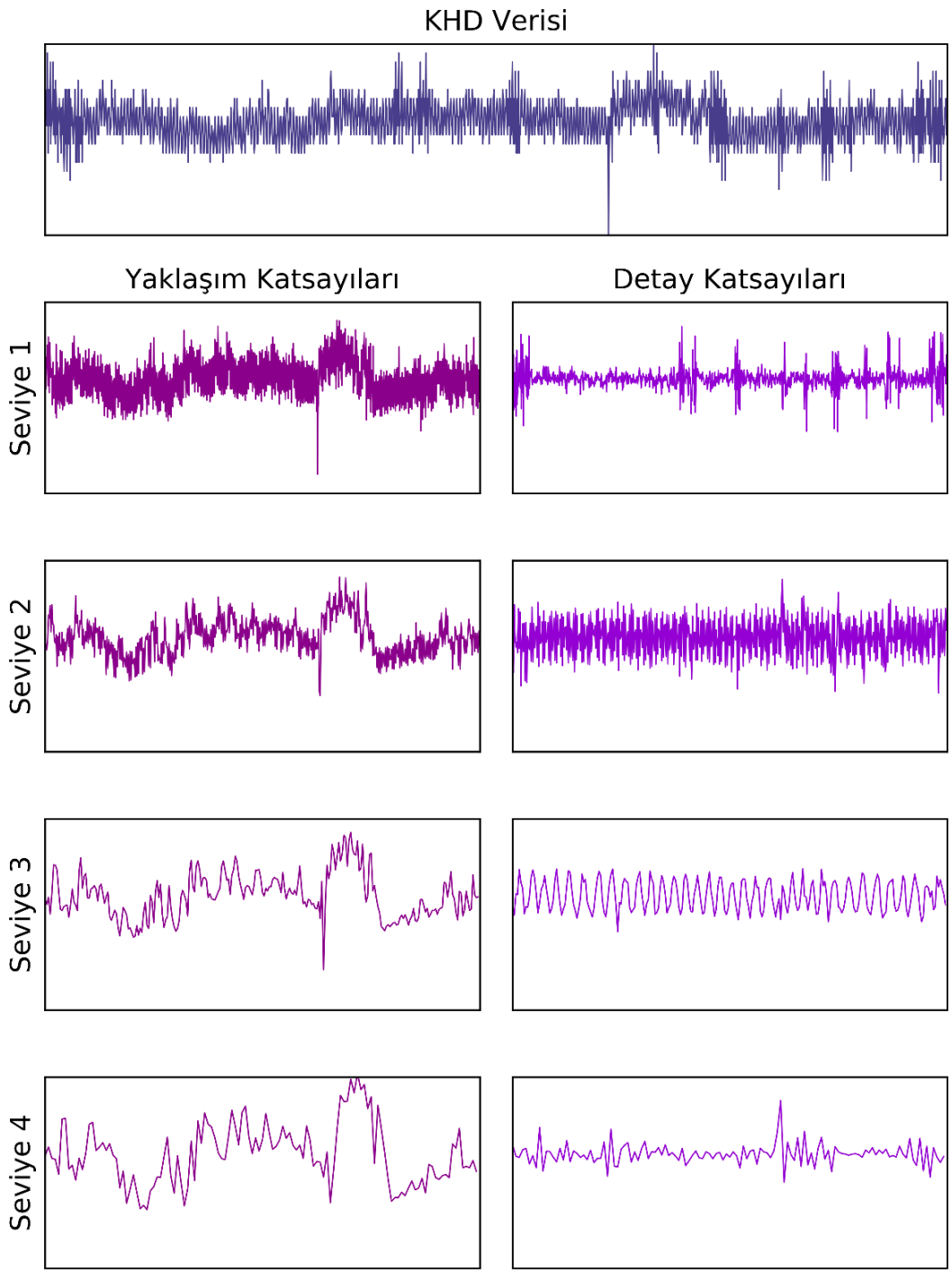
Şekil 2.16. Mallat ayrık dalgacık dönüşümü ağacı.

2.9.2.6. *Dalgacık Entropisi*

Bir işaretin entropisi, işaretin ne kadar düzensiz olduğu bilgisini vermektedir. Shannon tarafından düşünülen rasgele işaretlerin entropisini inceleme fikri bilgi teorisine önemli katkı sunmuştur [86]. Shannon entropisi herhangi bir işaretin olasılık dağılımlarını incelemeye ve işaret hakkında bilgi sahibi olmaya fırsat vermektedir [87]. Shannon entropisi Denklem (2.49)'da verilen formülle hesaplanmaktadır. Dalgacık entropisini hesaplamak için Shannon teorisindeki olasılık fonksiyonlarını dalgacık enerjisi şeklinde yazarsak Denklem (2.50) elde edilmektedir.

$$H(X) = -\sum_{i} p_i \, log_2 \, p_i \tag{2.49}$$

$$W_E = -\sum_{j}\left(\frac{E_j}{\sum_j E_j} log_2\left(\frac{E_j}{\sum_j E_j}\right)\right) \tag{2.50}$$



Şekil 2.17. Ayrık dalgacık dönüşümü yaklaşım ve detay katsayıları.

2.9.3. Doğrusal Olmayan Parametreler

Kalp atışlarındaki düzensizlikler KHD verisinde doğrusal olmayan bileşenler oluşturmaktadır. Doğrusal olmayan bileşenlerin analizinde kullanılabilen Poincare çizimi, KHD analizinde de kullanılmaktadır [88].

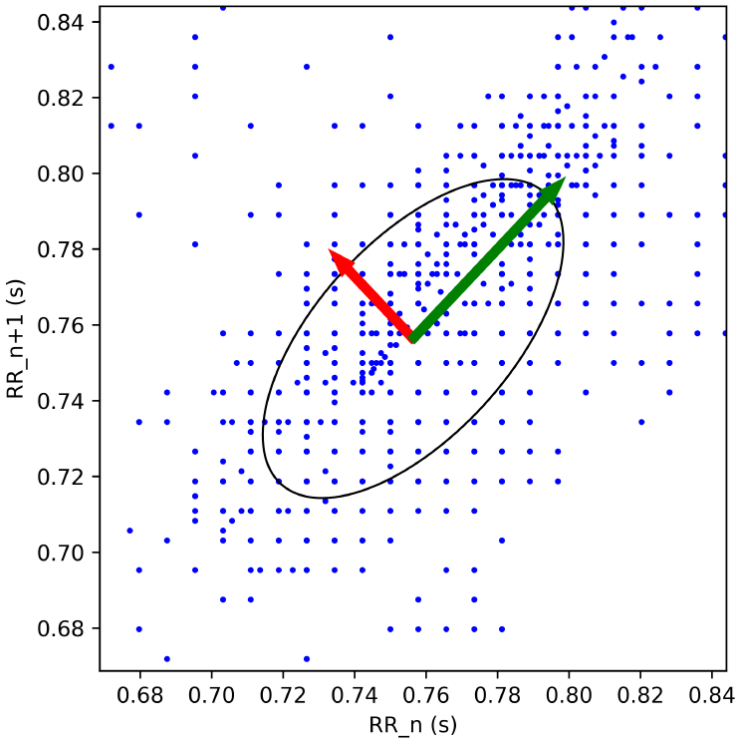
2.9.3.1. *Poincare Çizimi*

Her bir kalp atışı arası geçen süre olan RR aralıklarının, kendisinden sonra gelen RR aralığı ile olan farklarının gösterildiği Poincare çizimi Şekil 2.18'da görülmektedir. Çizimdeki noktalar RR aralıklarını göstermektedir. Noktalar arası mesafelerin standart sapmalarından oluşturulan elipse ait büyüklükler öznitelik olarak seçilmiştir. Standart sapmaların SD_1 ile SD_2 büyüklükleri ile çarpımı ve birbirine oranları, sempatik ve parasempatik aktivitelerle ilişkilendirilmiştir [89]. SD_1 ve SD_2'nin hesaplanması (2.51) ve (2.52) denklemleriyle yapılmıştır [90].

$$SD_1 = \frac{s_{R_i - R_{i-1}}}{\sqrt{2}} = \sqrt{\frac{1}{2}\frac{1}{n}\sum_{i=1}^{n}\left((R_i - R_{i-1}) - R_{ort}\right)^2}, \qquad s = \text{standart sapma} \tag{2.51}$$

$$SD_2 = \sqrt{2s_{R_i}^2 - \frac{1}{2}s_{R_i - R_{i-1}}^2} = \sqrt{\frac{1}{2}\frac{1}{n}\sum_{i=1}^{n}\left((R_i + R_{i-1}) - R_{ort}\right)^2}, \tag{2.52}$$

$s^2 = \text{varyans}$



Şekil 2.18. KHD verisine ait örnek Poincare çizimi.

2.9.4. Öznitelik Normalizasyonu

Hesaplanan özniteliklerin farklı ölçeklerde olması, sınıflandırma performansını olumsuz etkilemektedir. Bu sebeple özniteliklerin veri aralıklarını standartlaştırmak adına literatürde öznitelik normalizasyonu sıklıkla kullanılmaktadır.

2.9.4.1. *MinMax Normalizasyon*

MinMax normalizasyonu en çok kullanılan normalizasyon yöntemlerinden biridir. Özniteliklerin her birinin minimum değerini 0, maksimum değerini ise 1 değerine ölçekler. Tüm özniteliklerin aynı ölçekte olmasını garantilemesine karşın aykırı değerleri (outliers) temsilde başarısızdır. İstatistikte verinin geri kalanından oldukça farklı değerlere sahip olan verilere aykırı değerler denilmektedir. X_i özniteliğine ait verilerin yeni değerleri olan ${X_i}^+$, denklem (2.53) ile hesaplanmaktadır.

$$ {X_i}^+ = \frac{X_i - min\, X_i}{max\, X_i - min\, X_i}\,, \qquad i = \text{öznitelik indeksi} \tag{2.53} $$

2.9.4.2. *Z-Skor Normalizasyon*

MinMax normalizasyon her ne kadar öznitelik ölçeklerini eşitlese de aykırı değerleri kapsama sorununun önüne geçmek için Z-Skor yada diğer adıyla standart skor normalizasyonu kullanılmaktadır. μ özniteliğin ortalaması ve σ özniteliğe ait değerlerin standart sapması olmak üzere yeni öznitelik değerleri denklem (2.54) ile hesaplanmaktadır. Eğer vakaya ait öznitelik değeri diğer vakalara ait aynı öznitelik değerlerinin ortalamasına eşitse yeni öznitelik değeri 0, ortalamanın altında ise negatif ve ortalamanın üzerinde ise pozitif değere sahip olmaktadır. Alınan değerler ise ham öznitelik değerlerinin standart sapması tarafından belirlenmektedir. Yüksek standart sapmaya sahip öznitelik gruplarının normalize değerleri sıfıra yakın olmaktadır. Z-Skor normalizasyonunda MinMax normalizasyonla karşılaştırıldığında aykırı değer kapsaması daha iyi olsa da, tüm öznitelikler eşit ölçekte temsil edilememektedir.

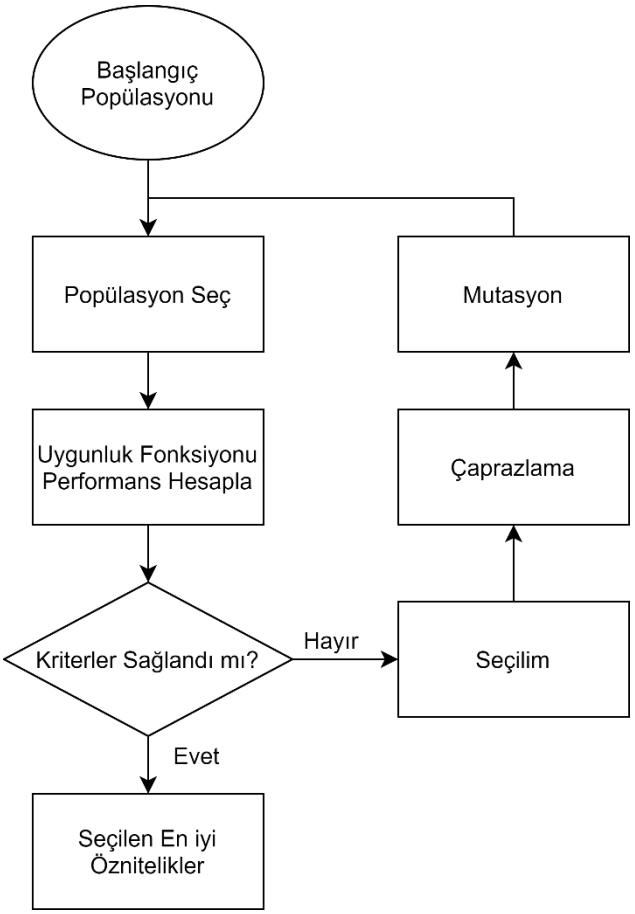
$$ {X_i}^+ = \frac{X_i - \mu_i}{\sigma_i}\,, \qquad i = \text{öznitelik indeksi} \tag{2.54} $$

2.9.5. Öznitelik Seçimi

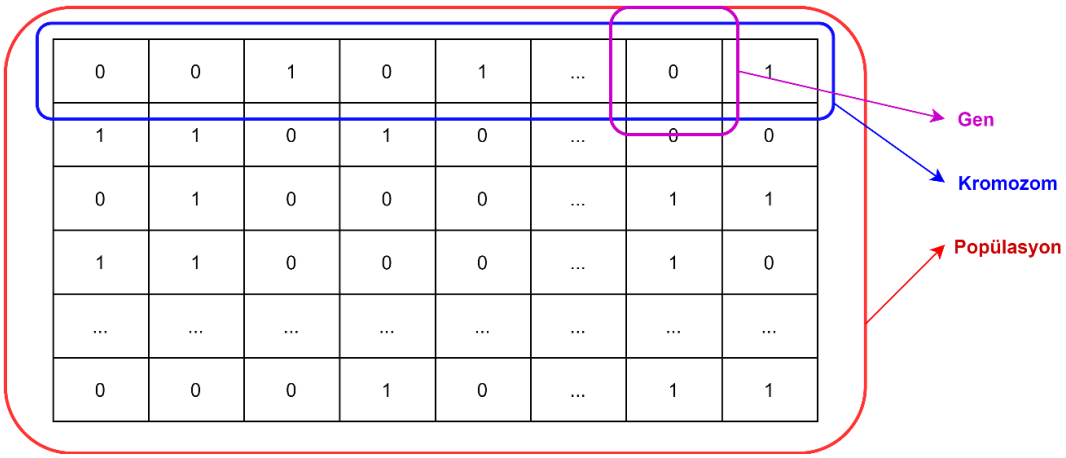
Vaka sınıflarını en iyi temsil eden öznitelik alt gruplarını bulmak çoğu sınıflandırma çalışması için önemli adımlardan biridir. Bazı sınıflandırıcıların karmaşıklığı sebebi ile öznitelik ön seçimi yapılması mümkün olmazken, bazı sınıflandırıcılar da doğal olarak öznitelik seçimi işlemini kendi içerisinde barındırmaktadır. Bu tezde çalışılan bazı sınıflandırıcılarda öznitelik seçme işlemi için Genetik Algoritma kullanılmıştır.

2.9.5.1. *Genetik Algoritma*

Genetik algoritma, öznitelik/veri seçilimlerinin bireyleri temsil ettiği, güçlü olan bireyin hayatta kaldığı, mutasyon ile kısmi değişimlerin yanı sıra rasgele bireylerin de yeni popülasyona katıldığı bir optimizasyon türüdür. Deterministik olmayan raslantısal çözüm üretmektedir. Bu sebeple her başlangıç koşulu için farklı çalıştırmalarda farklı bir sonuç üretebilmektedir. Çözümü iyileştirirken uygunluk fonksiyonu üzerinden hatayı küçültmeye çalışır. Özniteliklerin seçilimini temsil eden bireylerden rasgele oluşturulan başlangıç popülasyonu ile uygunluk fonksiyonu üzerinden en iyi bireylerin seçildiği, bireyler arası çaprazlamaların yapıldığı ve belirli sayıda birey üzerinde rasgele değişiminin yapıldığı temel algoritma Şekil 2.19 ile gösterilmiştir. Öznitelik seçimi için genellikle ikili genetik algoritma kullanılmaktadır. İkili gösterimde özniteliğin kullanılıp kullanılmayacağı bilgisi bireyler üzerinde bitlerle temsil edilir ve bireylere kromozom, bitle temsil edilen kısımlara da gen ismi verilmektedir. Şekil 2.20'de örnek bir popülasyon gösterilmiştir.



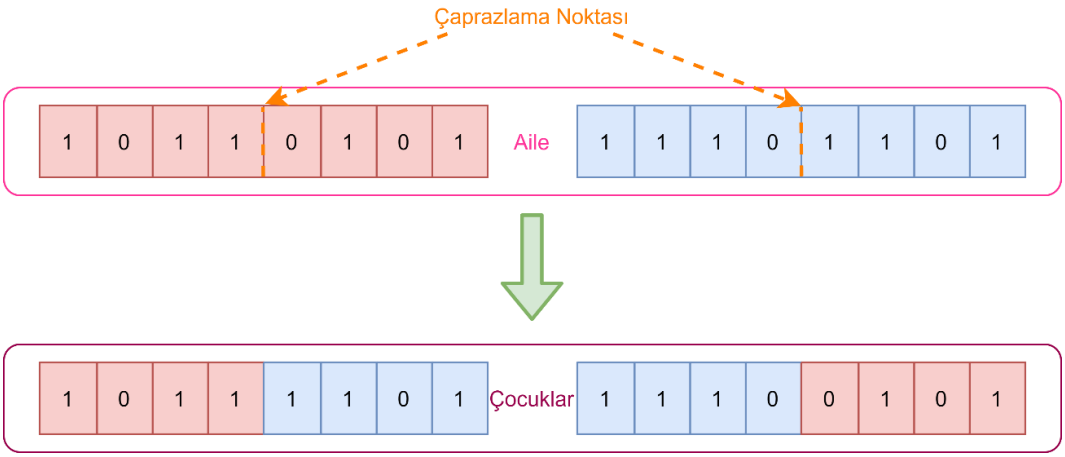
Şekil 2.19. GA ile öznitelik seçimi.



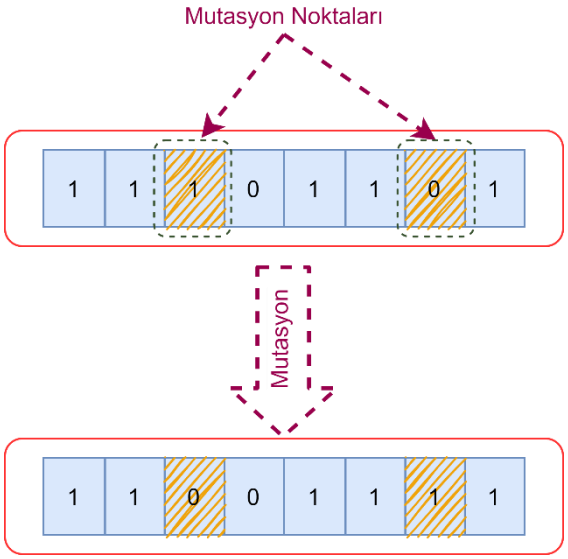
Şekil 2.20. İkili GA'da popülasyon temsili.

Hangi öznitelik grubunun daha başarılı olduğunu tespit etmek için ise uygunluk fonksiyonu (fitness function) kullanılmaktadır. Uygunluk fonksiyonunun çıktısını en küçük yapan değerler başarılı sayılarak yeni nesiller seçilmekte, böylelikle nesilden nesile iyileşmeler sağlanabilmektedir. Ayrıca Şekil 2.21'de gösterildiği gibi başarılı bireyler arası çaprazlamalar yapılarak yeni bireyler oluşturulmaktadır. Benzer şekilde Şekil

2.22'de gösterildiği gibi rasgele genlerde mutasyonlar yapılarak uygunluk fonksiyonunun yerel minimumlara takılması önlenmektedir.



Şekil 2.21. Çaprazlama örneği.



Şekil 2.22. Mutasyon örneği.

2.10. SINIFLANDIRICILAR

2.10.1. k En Yakın Komşuluk

Eğitime ihtiyaç duymadığı için görece performanslı ve başarılı sonuçlar üretebilen, parametrik olmayan, basitliği sebebiyle de sıkça kullanılan bir sınıflandırma algoritmasıdır. Çok boyutlu uzayda tüm veri setleri bir nokta ile gösterilmek üzere, incelenen noktanın k adet en yakın komşusuna ait sınıf değerlerinin çoğunluğuna bakılarak sınıf belirlenmektedir. Çoğunluğa dayalı sınıf belirlendiği için k değeri tek sayı

seçilmektedir. İncelenen noktaya ait komşuların k adet en yakınını tespit etmek için, ilgilenilen nokta ile tüm örneklerin uzaklıklarının hesaplanması gerekmektedir. Uzaklık hesaplaması içinse en çok kullanılan fonksiyonlar Öklid, Manhattan, Minkowski ve Hamming uzaklık fonksiyonlarıdır. Öklid, Manhattan ve Minkowski uzaklık fonksiyonları sürekli değişkenler için kullanılmakla birlikte, sınıfsal değişkenler için Hamming uzaklık fonksiyonu kullanılmaktadır. Bu uzaklık fonksiyonlarına ait formüller sırasıyla Denklem (2.55) ile (2.58) arasında verilmiştir. Burada x ve y, uzaklığı hesaplanacak iki noktanın konumunu belirtmekte olup, n değeri ise noktayı tanımlayan öznitelik sayısıdır.

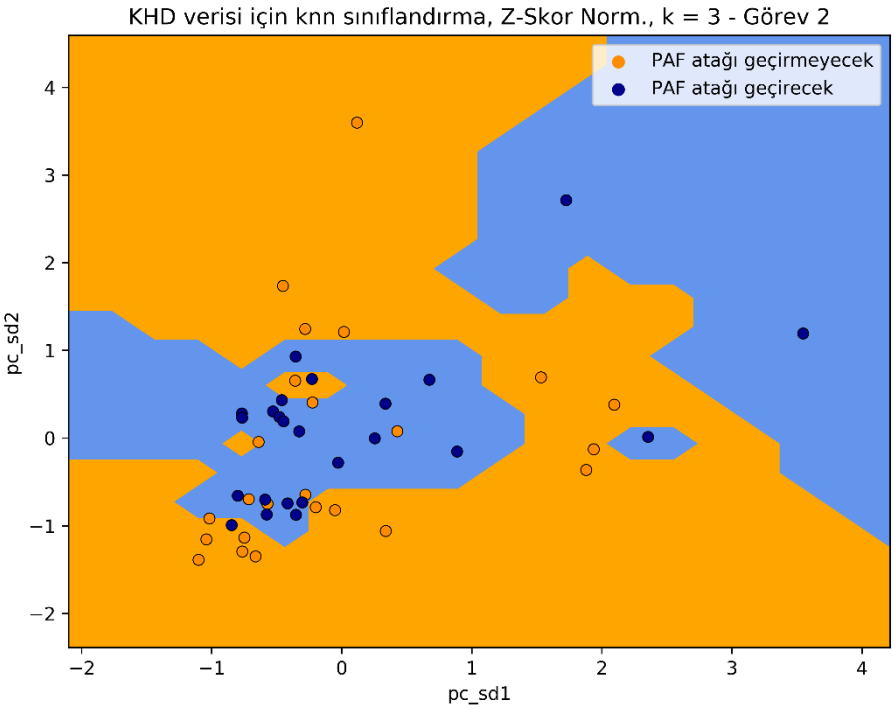
$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.55}$$

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i| \tag{2.56}$$

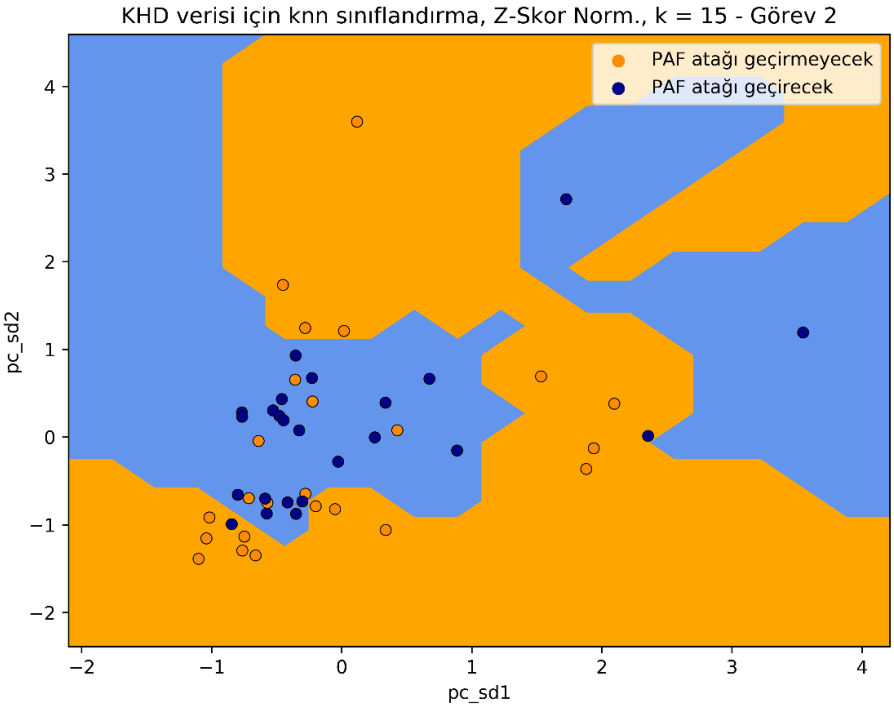
$$d(x,y) = \sqrt[q]{\sum_{i=1}^{n}(x_i - y_i)^q}\ , \qquad q = \text{ölçekleme parametresi} \tag{2.57}$$

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i|$$
$$x = y \Rightarrow d = 0 \tag{2.58}$$
$$x \neq y \Rightarrow d = 1$$

kNN sınıflandırıcıda k komşuluk değerinin genellikle küçük bir tamsayı seçilmesi beklenir. k değeri çok fazla büyürse birbirine yakın farklı sınıf değerine sahip noktalarda sınıflandırma performansı düşmektedir. Şekil 2.23'te k=3 değeri için ve Şekil 2.24'te ise k=15 değeri için, sadece Poincare çizimine ait SD1 ve SD2 özniteliklerinin kullanıldığı kNN sınıflandırıcıların karar sınır gösterimleri sunulmuştur.



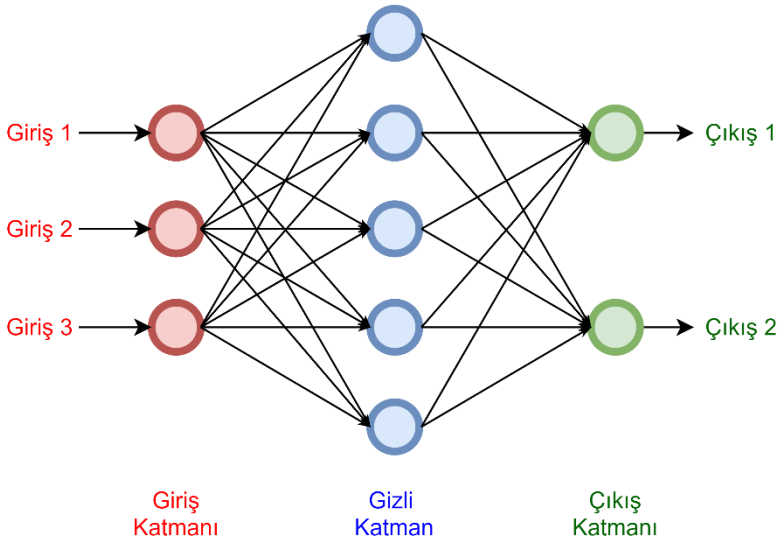
Şekil 2.23. k=3 değeri için kNN sınıflandırıcı karar sınır gösterimi.



Şekil 2.24. k=15 değeri için kNN sınıflandırıcı karar sınır gösterimi.

2.10.2. Çok Katmanlı Algılayıcı

Çok katmanlı algılayıcılar (MLP), ağın giriş ve çıkış katmanlarına ilave olarak gizli ara katmanlara sahip en az üç katmandan oluşan sinir ağlarıdır. En çok kullanılan yapay sinir ağı modellerindendir [91]. Şekil 2.25'de MLP temel yapısı gösterilmiştir. Her bir katman pek çok şekilde ve ağırlıkta birbirlerine bağlı olduğu için tam bağlantılı katman olarak isimlendirilmektedir. Ağın çıktısı, her bir sinir hücresinin çıktılarının belirli ağırlıklarda birbirine eklenmesi ile bulunmaktadır. Tek bir sinir hücresinin çıkışı Denklem (2.59)'da verilen formülle hesaplanmaktadır. Her sinir hücresi arası bağlantılar da w_{ij} ağırlık katsayıları ile hesaplanmaktadır. Burada istenen çıkış değerine yaklaşana kadar ağ ağırlık katsayıları iyileştirilir. Bu işlemler sonrası ağ eğitilmiş olmaktadır. Eğitilen ağın girişine gelen her yeni değer için ağın mevcut ağırlık katsayıları ile ürettiği çıkış ise tahmin edilen değer olmaktadır.

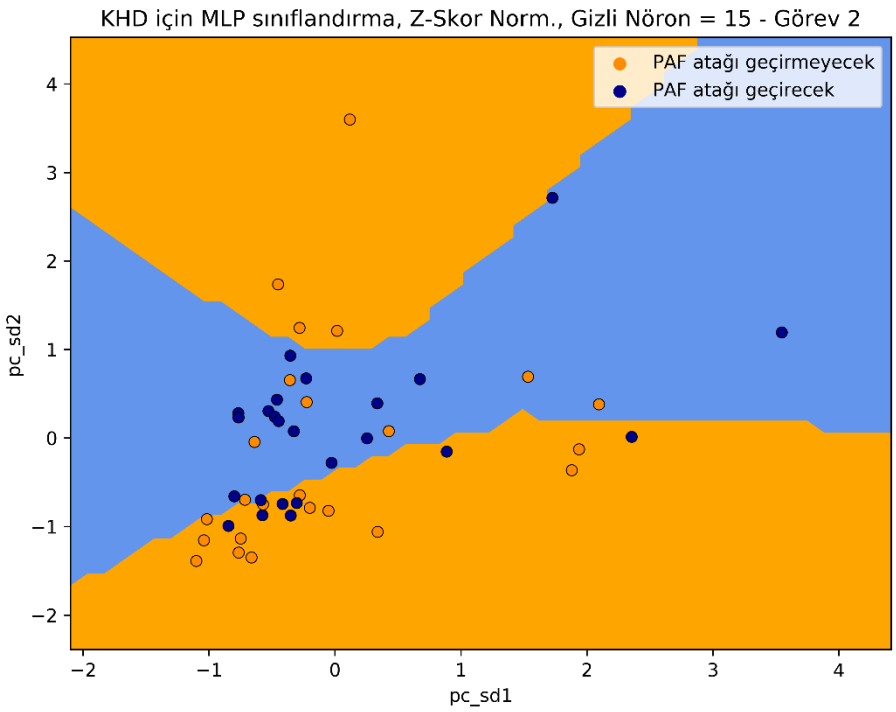


Şekil 2.25. Çok katmanlı algılayıcı (MLP) yapısı.

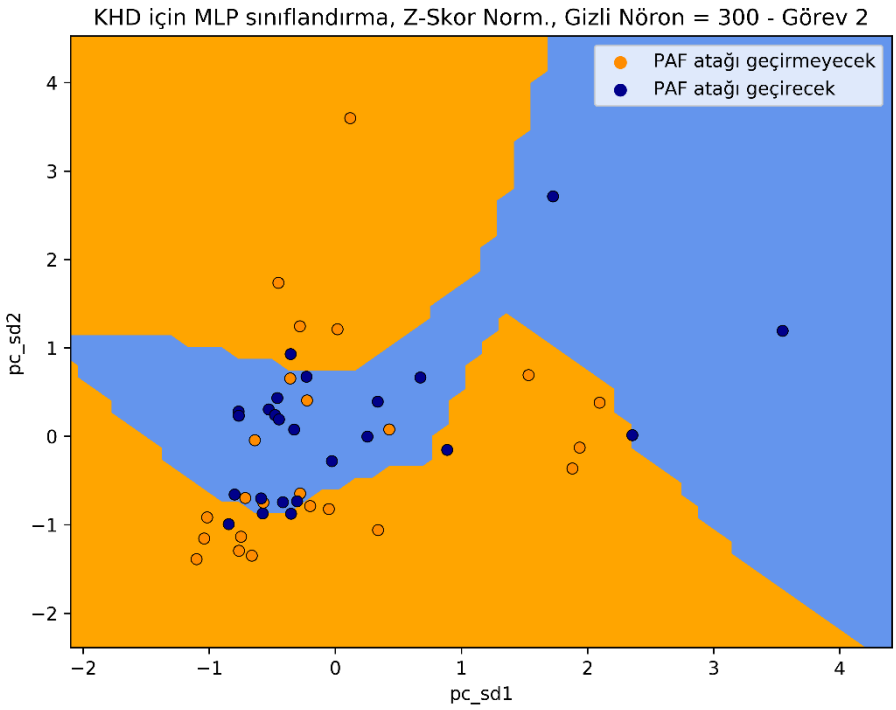
$$Ç_i = f\left(\sum_{i=1}^{n} w_i G_i + \emptyset\right), \qquad f = \text{aktivasyon fonksiyonu}, \emptyset = \text{ eşik değer} \tag{2.59}$$

Ağın sınıflandırma problemini doğru çözebilmesi için, problem karmaşıklığına göre nöron sayısını yeterli seçmek gerekmektedir. Poincare çizimi verileri olan SD1 ve SD2 öznitelikleri giriş olarak seçildiğinde, KHD verisi için sınıflandırma karar sınırları 15 nöron ve 300 nöron için sırasıyla Şekil 2.26 ve Şekil 2.27'de verilmiştir. Nöron sayısı azaldıkça başarım azalsa da yüksek nörona sahip ağlarda da ezberleme sorunu ortaya

çıkabilmektedir. Bu sebeple ağın ezberlemesini önleme adına işlem sayısını sınırlama veya eğitim verisini sınırlama gibi önlemler alınmaktadır.



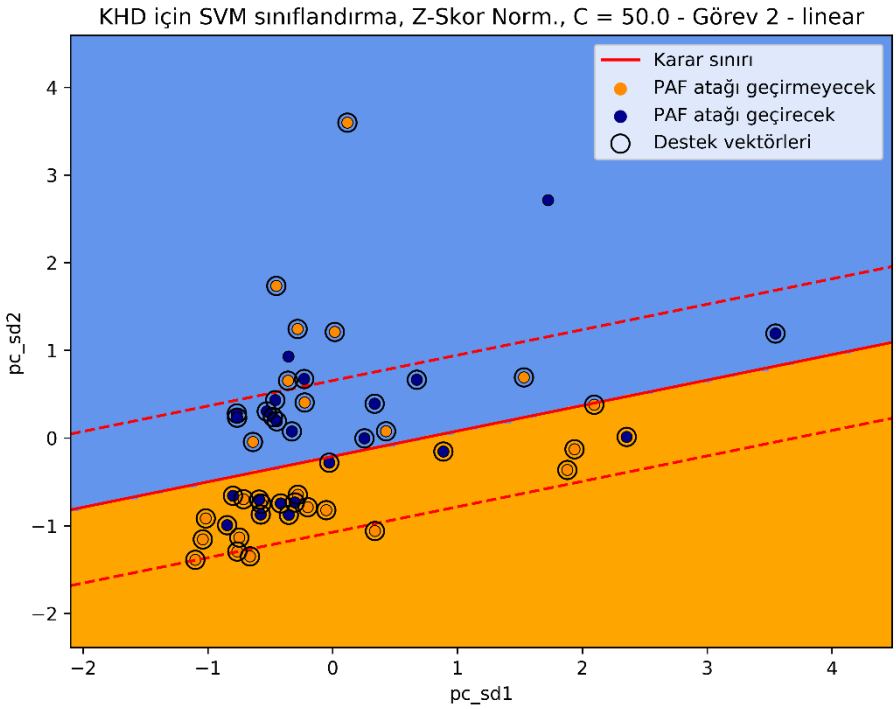
Şekil 2.26. 15 nörona sahip MLP sınıflandırıcı karar sınır gösterimi.



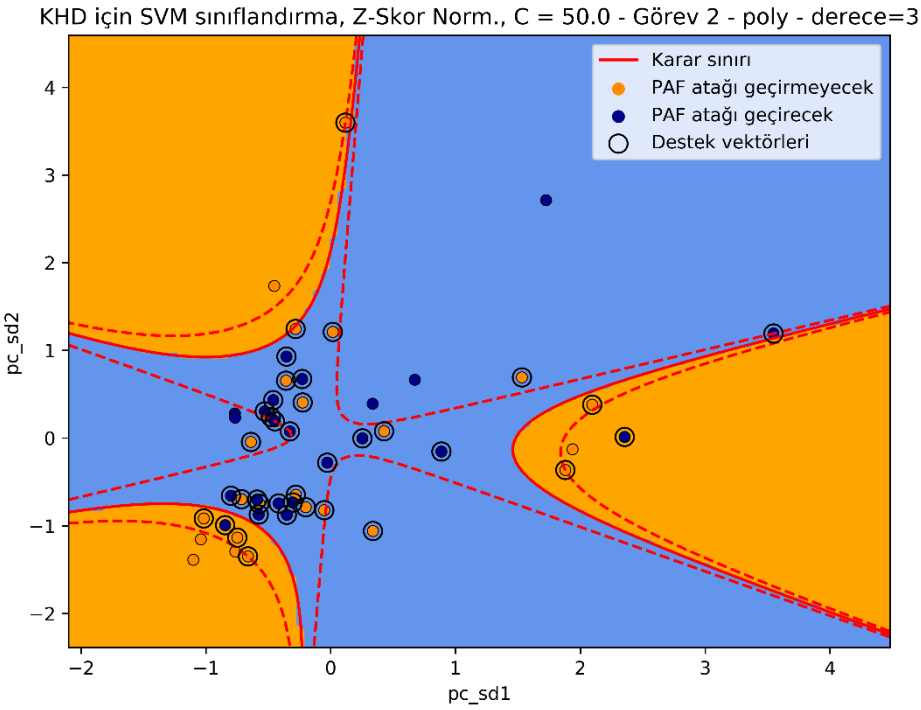
Şekil 2.27. 300 nörona sahip MLP sınıflandırıcı karar sınır gösterimi.

2.10.3. Destek Vektör Makinesi

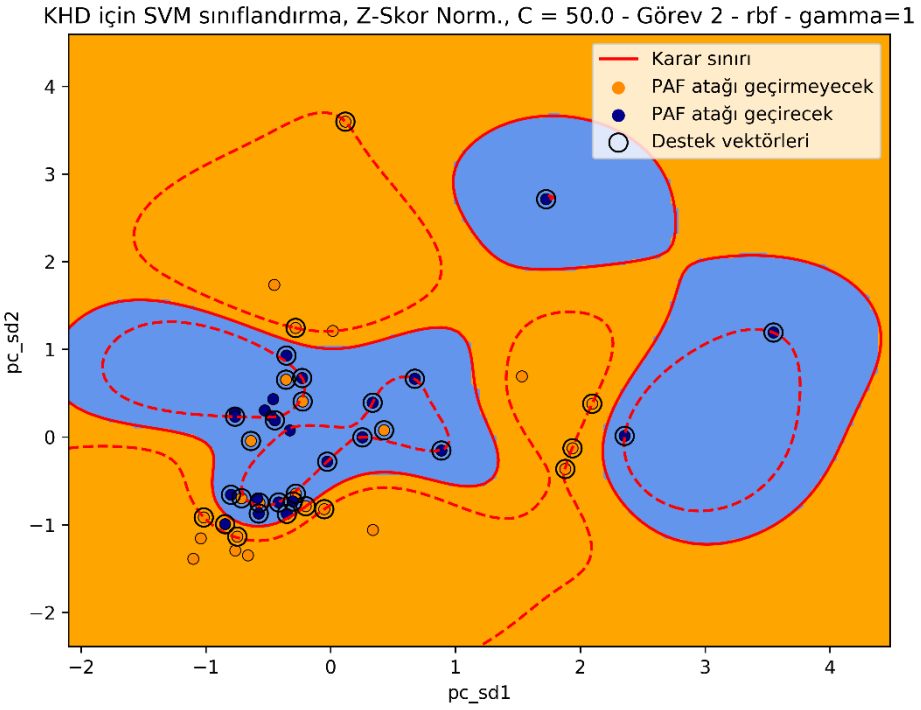
Destek vektör makinesi yada diğer adıyla destek vektör ağları, iki sınıf arasında en iyi karar sınırını bulan makine öğrenme yöntemidir. Bunu gerçekleştirmek adına, yüksek boyutlu öznitelik uzayına doğrusal olmayan bir haritalandırma yöntemi kullanan çekirdek fonksiyonlarıyla giriş vektörlerini aktarmak gerekmektedir. Bu öznitelik uzayında sınıflandırmayı gerçekleştirebilecek doğrusal bir karar yüzeyi oluşturulabilmektedir [92]. Doğrusal olarak sınıflandırılması mümkün olmayan veriler için farklı çekirdek fonksiyonları da kullanılabilmektedir. Polinom ve radyal temelli fonksiyonlar (RBF) bu işlemlerde sıklıkla kullanılmaktadır. Marjin sınırı, SVM tarafından belirlenen karar yüzeyinin en yakındaki özniteliklere olan uzaklığıdır. Karar yüzeyinin tüm bileşenleri doğru sınıflandıramadığı noktada bazı bileşenlerin karar yüzeyinin yanlış tarafında olmasına izin verilir. Böylelikle oluşan yeni marjin sınırına, yumuşak marjin sınırı (Soft Margin) denilmektedir. Şekil 2.28, Şekil 2.29 ve Şekil 2.30'da sırasıyla doğrusal, polinomal ve RBF çekirdek fonksiyonları için SVM sınıflandırıcıya ait karar sınırları, yumuşak marjin sınırları ve destek vektörleri sunulmuştur.



Şekil 2.28. Doğrusal çekirdek fonksiyonlu SVM karar sınır gösterimi.



Şekil 2.29. Polinomal çekirdek fonksiyonlu SVM karar sınır gösterimi.



Şekil 2.30. RBF çekirdek fonksiyonlu SVM karar sınır gösterimi.

SVM sınıflandırıcıların çalışmasını etkileyen önemli bazı parametreler mevcuttur. Bunlar veri setine göre optimize edilmesi gereken parametrelerdir. Çizelge 2.3'te önemli parametreler sunulmuştur. C değeri çok küçük seçilirse geniş marjinli bir karar sınır alanı, çok büyük seçilirse de dar marjinli bir karar sınır alanı seçilir. Düşük C değeri ile, sınıflandırma performansı düşmesine sebep olunmasına karşın, çok yüksek C değerleri ile de ezberlemeye sebep olunmaktadır. Gama değerinin çok yüksek seçilmesi de benzer bir etkiye sebep olmaktadır. Ezberlemiş ağlar mevcut veri seti için yüksek başarım gösterse de genel bir sınıflandırıcı oluşturulamadığı için ağa ilk defa gelen verilerde sınıflandırma performansı tahmin edilemez olmaktadır.

Çizelge 2.3. SVM sınıflandırmada hiper parametreler.

Parametre	Açıklama
C	Düzenlileştirme (Regularization) parametresidir, yüksek C değerleri düşük yumuşak marjin değeri üretir
γ	Gama değeri RBF çekirdek fonksiyonu kullanıldığında RBF fonksiyonunun eğrilik miktarını belirler
Derece	Polinom çekirdek fonksiyonu kullanıldığında polinomun derecesini belirler

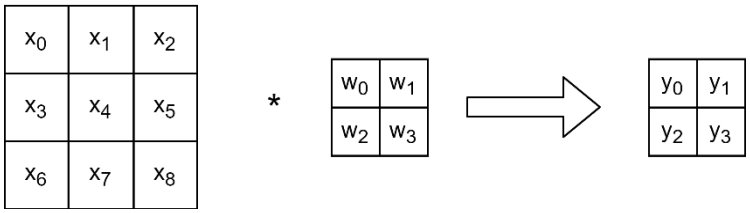
2.10.4. Evrişimli Sinir Ağları

Evrişimli sinir ağları (CNN), katmanlarından en az birinde evrişim (Konvolüsyon – Convolutional) işlemi barındıran çok katmanlı algılayıcılardır. MLP ağlarındaki gizli katmanlar CNN ağlarında evrişim işlemlerinin yapıldığı katmanları barındırmaktadır. Ayrıca evrişimli sinir ağlarında evrişim katmanının yanı sıra bazı özel katmanlar da sıklıkla kullanılmaktadır.

2.10.4.1. *Evrişim Katmanı*

CNN'in ana bileşeni olup, evrişim işleminin yapıldığı katmandır. Genellikle veriye dair özniteliklerin belirlendiği katmanlardır. Bu tezde de kullanıldığı gibi bazı çalışmalarda, mevcut öznitelik kümesi üzerinden öznitelik seçim işlemini de üstlenebilmektedir [93]. Evrişim katmanı sayesinde tüm veri ile çalışmak yerine önemli noktalar üzerine odaklanabilindiğinden, daha derin bir öğrenme algoritması oluşturulabilmektedir [94]. Evrişim ağına ait öğrenilen ağırlık katsayıları ile girişte bulunan veriler evrişim işlemine tabi tutularak çıkış hesaplanmaktadır. Örnek $Y = X * W$ evrişim işlemi Şekil 2.31'de gösterilmiş olup çıkışlardan y_0 değerinin hesabı Denklem (2.60)'da verilmiştir. Benzer

biçimde Y matrisine ait diğer değerler de, W matrisinin X matrisi üzerinde gezdirilmesi ile hesaplanır. Sonuç olarak evrişim işlemi için Denklem (2.61) elde edilir.



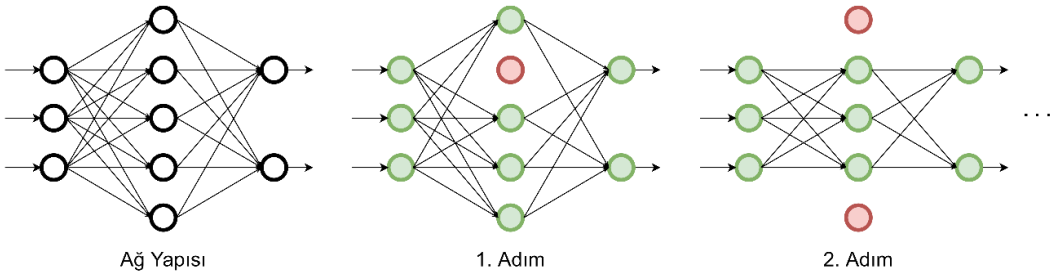
Şekil 2.31. Evrişim işlemi gösterimi.

$$y_0 = x_0 w_0 + x_1 w_1 + x_3 w_2 + x_4 w_3 \tag{2.60}$$

$$y(n) = x(n) * w(n) = \sum_{k=0}^{n} w(k) x(n-k) \tag{2.61}$$

2.10.4.2. *Seyreltme Katmanı*

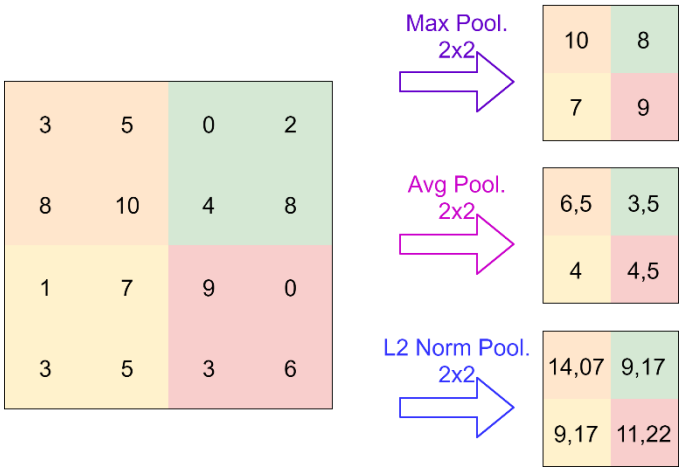
CNN ile sınıflandırmanın en olumsuz noktalarından birisi de ağın çok hızlı öğrenme potansiyelinin olmasıdır. Bu hızlı öğrenme önlenmez ise ezbere dönüşerek aşırı öğrenmiş ve eğitim sonrası dar bir veri kümesi için iyi çalışan bir ağ elde etmiş oluruz. Bunu önlemek adına seyreltme (Dropout) katmanı ağda düzenleme yaparak ezberlemeyi önlemesi için CNN ağında kullanılmaktadır [59]. Bu katmanda verilen parametrelere göre giriş ile çıkış arasındaki bağlantı yolları rasgele şekilde kesilmektedir. Dolayısıyla her adımda farklı bağlantılar üzerinden ağ parametreleri güncellendiğinden aşırı uyum önlenmiş olmaktadır. Şekil 2.32'de sunulan örnek seyreltme işleminde her adımda farklı sinir yollarının devre dışı bırakılması gösterilmiştir. Yeşil daireler aktif hücreleri, kırmızı daireler ise pasif hücreleri temsil etmektedir.



Şekil 2.32. Örnek bir ağ için Seyreltme işlemi.

2.10.4.3. *Havuzlama Katmanı*

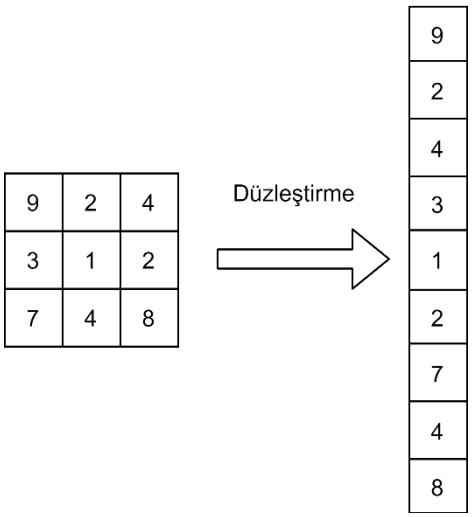
Havuzlama (Pooling) katmanı veri boyutunu küçültmek için kullanılmaktadır. Ayrıca bu işlem veride bir kısım kayıplara sebebiyet verir. Bu kayıp ise, ağın ezberlemesinin önlenmesinde faydalı olabilmektedir. Bu katmanın kullanılması zorunlu olmamakla birlikte genellikle kullanımda, ya evrişim katmanları arasına ya da bütün evrişim katmanlarının sonuna eklenmektedir. Evrişim katmanlarının arasına konulduğunda ağ karmaşıklığını önemli ölçüde düşürmektedir. Çok sayıda havuzlama işlemi vardır. Bunlardan en çok kullanılanları Max Pooling, Avg. Pooling ve L2-Norm Pooling işlemleridir. Havuzlama işlemlerinden Max Pooling, Avg. Pooling ve L2 Norm işlemleri Şekil 2.33'te gösterilmiş olup, 2x2 bir havuzlama filtresi için giriş 2x2 alanlara bölünerek Max Pooling için her bir alandaki en büyük değer çıkışa aktarılırken, Avg. Pooling için ise her alanın ortalaması çıkışa aktarılmaktadır. L2 Norm Pooling işleminde ise Öklid mesafeleri hesaplanmaktadır. Buradaki örnek için ise 4 büyüklüğün kareleri toplanıp, toplamın karekökü alınmaktadır.



Şekil 2.33. Havuzlama işlemi gösterimi.

2.10.4.4. *Düzleştirme Katmanı*

CNN katmanlarından geçen veriler matris formunda olmaktadır. Bir sonraki adıma geçiş için bu verilerin düzleştirilmesi gerekmektedir. Bu sebeple düzleştirme (Flattening) katmanı kullanılmaktadır. Şekil 2.34'de düzleştirme işlemi için örnek bir çıktı gösterilmiştir.



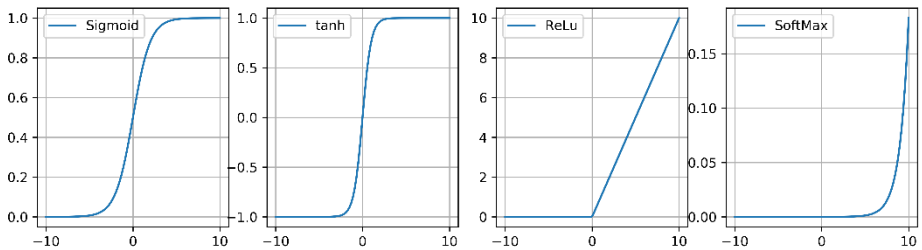
Şekil 2.34. Düzleştirme işlemi gösterimi.

2.10.4.5. *Tam Bağlantılı Katman*

Tam bağlantılı katman, düzleştirme katmanın tüm çıkışlarıyla bağlantılı olan katmandır. MLP ağ gibi çalışmaktadır. CNN ağında belirlenen veya seçilen öznitelikler üzerinden sınıflandırma işlemi burada yapılmaktadır.

2.10.4.6. *Aktivasyon Katmanı*

Aktivasyon katmanı ya da işlemi, her katmanda yapılan sayısal işlemler sonrası hem bir standartlaşma sağlamak için hem de ağın tepkisini şekillendirmek için kullanılabilmektedir. ReLu (Rectified Linear Unit), SoftMax, Sigmoid ve tanh gibi çeşitli aktivasyon işlemleri mevcuttur. Şekil 2.35'te aktivasyon katmanlarında kullanılan fonksiyonlar gösterilmiştir. Çoğunlukla ağın genelinde ReLu fonksiyonu kullanılmakta, çıkışta ise sınıfların durumuna göre ikili sınıflandırmada SofMax, çok sınıflı çıktılarda ise Sigmoid tercih edilmektedir.



Şekil 2.35. Aktivasyon fonksiyonları.

2.11. PERFORMANS ÖLÇÜMÜ

Bu tezde sınıflandırıcıların başarımlarını değerlendirirken karışıklık matrisi (Confusion Matrix) üzerinden elde edilen büyüklükler seçilmiştir. Ana büyüklük olarak ise doğruluk değeri kullanılmıştır. Karışıklık matrisi gerçek değerleri de bilinen sınıflandırma işlemlerinde sıklıkla kullanılmaktadır. Şekil 2.36'da tezde çalışılan görevler için kullanılan karışıklık matrisleri gösterilmiştir. Denklem (2.62) ile (2.65) arasında ise tezde kullanılan başarım kriterleri listelenmiştir.

Görev 1		
	Tahminen Hasta	Tahminen Sağlıklı
Gerçekte Hasta	Gerçek Pozitif TP	Yanlış Negatif FN
Gerçekte Sağlıklı	Yanlış Pozitif FP	Gerçek Negatif TN

Görev 2		
	Tahminen Atak Geçiren	Tahminen Atak Geçirmeyen
Gerçekte Atak Geçiren	Gerçek Pozitif TP	Yanlış Negatif FN
Gerçekte Atak Geçirmeyen	Yanlış Pozitif FP	Gerçek Negatif TN

Şekil 2.36. Karışıklık matrisleri.

$$Doğruluk = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.62}$$

$$Kesinlik = \frac{TP}{TP + FP} \tag{2.63}$$

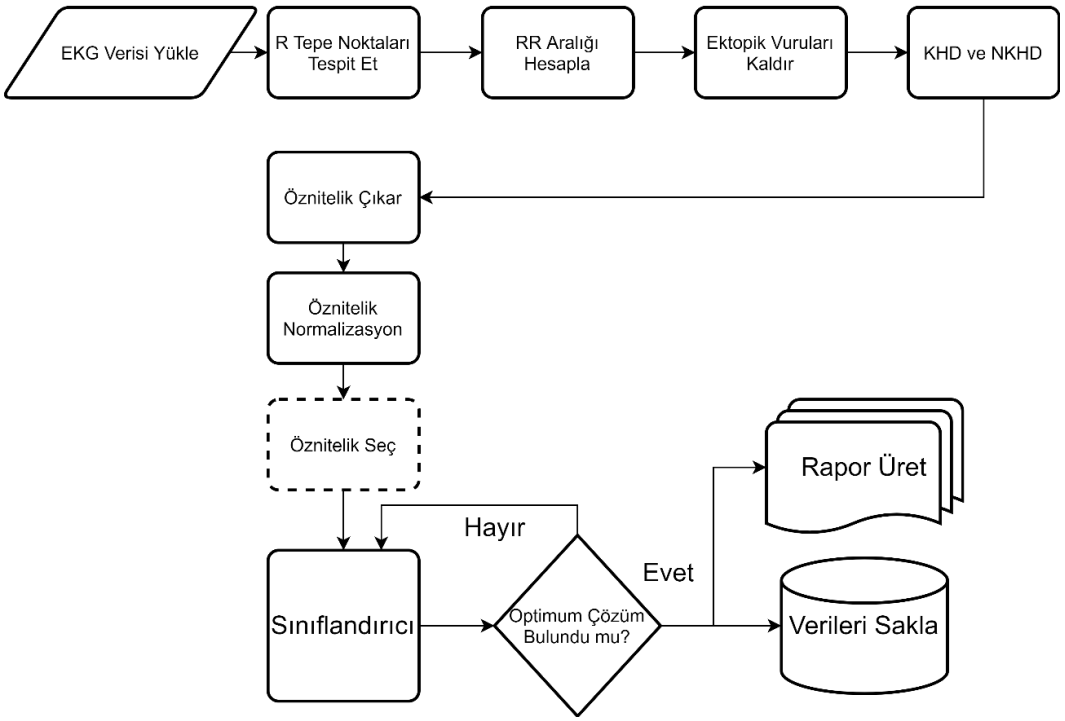
$$Duyarlılık = \frac{TP}{TP + FN} \tag{2.64}$$

$$F1_{skor} = 2 * \frac{Kesinlik * Duyarlılık}{Kesinlik + Duyarlılık} \tag{2.65}$$

3. NORMALİZE EDİLMİŞ KALP HIZI DEĞİŞKENLİĞİ ANALİZİ

Bu tezde PAF hastalığı tespiti (Görev 1) ve PAF atağı tahmini (Görev 2) amacıyla KHD ve NKHD verileri üzerinden çalışılarak KHD normalizasyonunun başarıma etkisi incelenmiştir. Bu amaçla öncelikle AFPDB veritabanından hastalık tespiti için 49 sağlıklı ve 49 hasta teşhisi olan kişilere ait veri (veritabanı 1) ile atak tahmini için 24 yakın zamanda atak geçirmeyecek hastaya ve 25 kısa sürede atak geçirecek hastaya ait olan veri (veritabanı 2) seçilerek iki farklı veritabanı oluşturulmuştur. Oluşturulan veritabanlarındaki EKG işareti üzerinden R tepe noktaları tespit edilmiş, KHD verisi oluşturulmuş ve ektopik vurular kaldırılmıştır. Oluşan KHD verileri normalize edilerek NKHD veri seti üretilmiştir. Her veritabanı için oluşan KHD ve NKHD veri setleri için, doğrusal ve doğrusal olmayan 6 farklı ölçümleme metoduyla 48 adet öznitelikten oluşan temel öznitelik seti elde edilmiştir. Oluşturulan temel öznitelik setleri de MinMax ve Z-Skor normalizasyon aşamalarından geçirilmiş, her veri seti için 3 farklı öznitelik seti üretilmiştir.

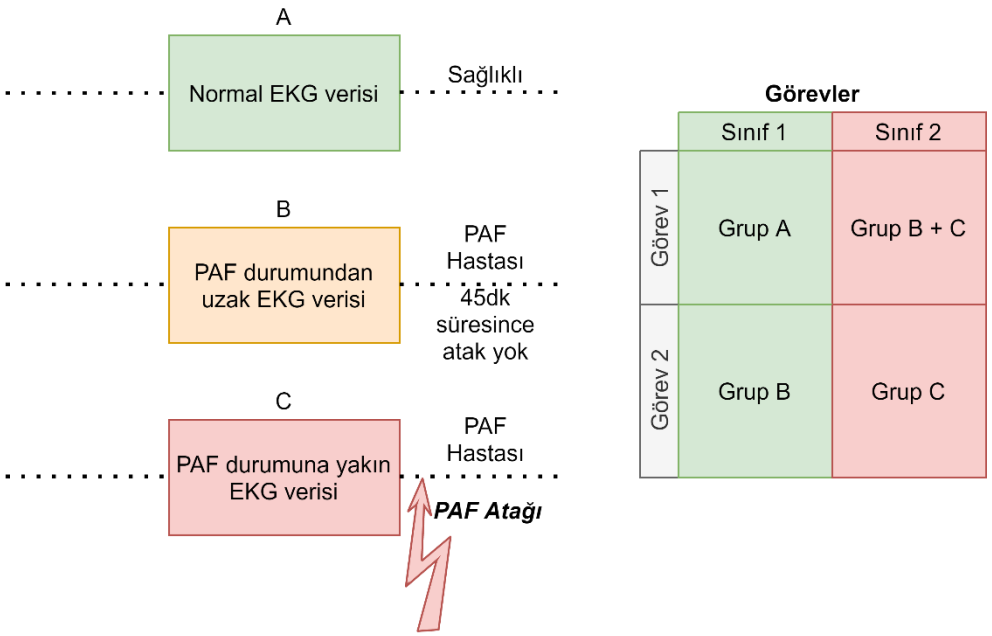
Bir sonraki adımda ise, çalışmadaki iki görev için üretilen öznitelik setleri sınıflandırma işlemine tabi tutulmuştur. Kalp hızı değişkenliği normalizasyonunun başarıma olan etkisi farklı koşullarda incelenmiştir. Sınıflandırıcı olarak ise k en yakın komşuluk (kNN), destek vektör makinesi (SVM), çok katmanlı algılayıcı (MLP) ve evrişimli sinir ağları (CNN) kullanılmıştır. Çalışmadaki her iki görev için de Şekil 3.1'deki temel örüntü tanıma algoritması kullanılmıştır. Yapılan çalışmalarla ilgili detaylar ve kullanılan kodlar bu kısımda verilmiştir.



Şekil 3.1. Genel algoritma.

3.1. VERİNİN ELDE EDİLMESİ

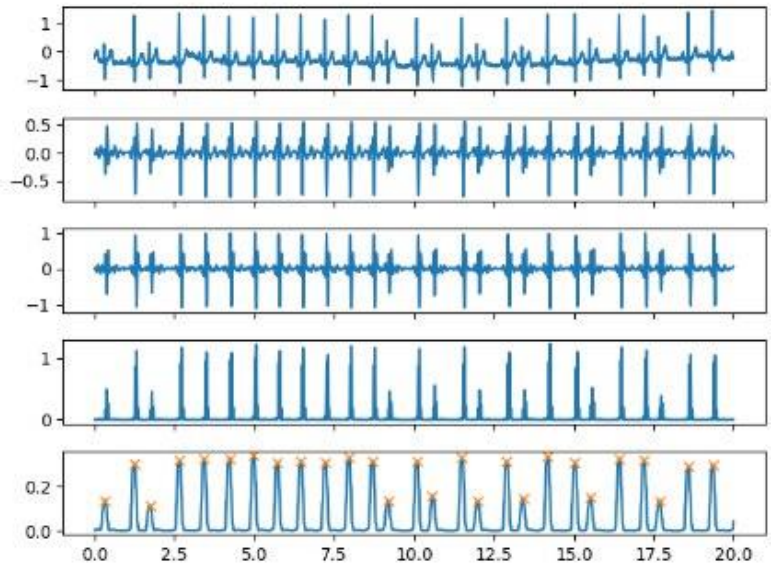
Pyshio.net sitesinde erişime açık sunulan AFPDB veritabanı, 128 Hz örnekleme frekansında 16 bit çözünürlüğe sahip iki kanal EKG verilerinden oluşmaktadır [35]. Veriler, her biri 30 dakikalık olmak üzere, n ile başlayan 50 adet sağlıklı kişilerden, p ile başlayıp tek sayı ile biten 25 adet kısa süre öncesi veya sonrasında atak geçirmemiş PAF hastalarından ve p ile başlayıp çift sayı ile biten 25 adet atak geçirmek üzere olan PAF hastalarından alınmıştır. Bu EKG örneklerinden n27 ve p37 isimli veriler sorunlu olduğu için kullanılmamıştır. Bu sebeple çalışmada 49 adet sağlıklı (Grup A), 24 adet yakın zamanda atak geçirmeyen PAF hastası (Grup B) ve 25 adet yakın zamanda atak geçirecek olan PAF hastasına (Grup C) ait EKG verileri kullanılmış olup, Şekil 3.2'de veritabanlarının gruplanması gösterilmiştir. Yapılacak PAF teşhisi sınıflandırması için (Görev 1) Grup A verisine karşı Grup B ve Grup C toplamı kıyaslanmış, PAF atak tahmini için ise (Görev 2) Grup B verisi ile Grup C verisi kıyaslanmıştır.



Şekil 3.2. Veri grupları ve görevler.

Seçilen gruplardan oluşturulan veritabanlarındaki EKG verilerinin öncelikle türevi alınmış, daha sonra karesi alınarak tepe noktaları bulunmuştur (Şekil 3.3). Tespit edilen tepe noktaları arasında geçen süre (3.1) RR aralığı verisini oluşturmaktadır.

$$RR_n = R_n - R_{n-1} \tag{3.1}$$

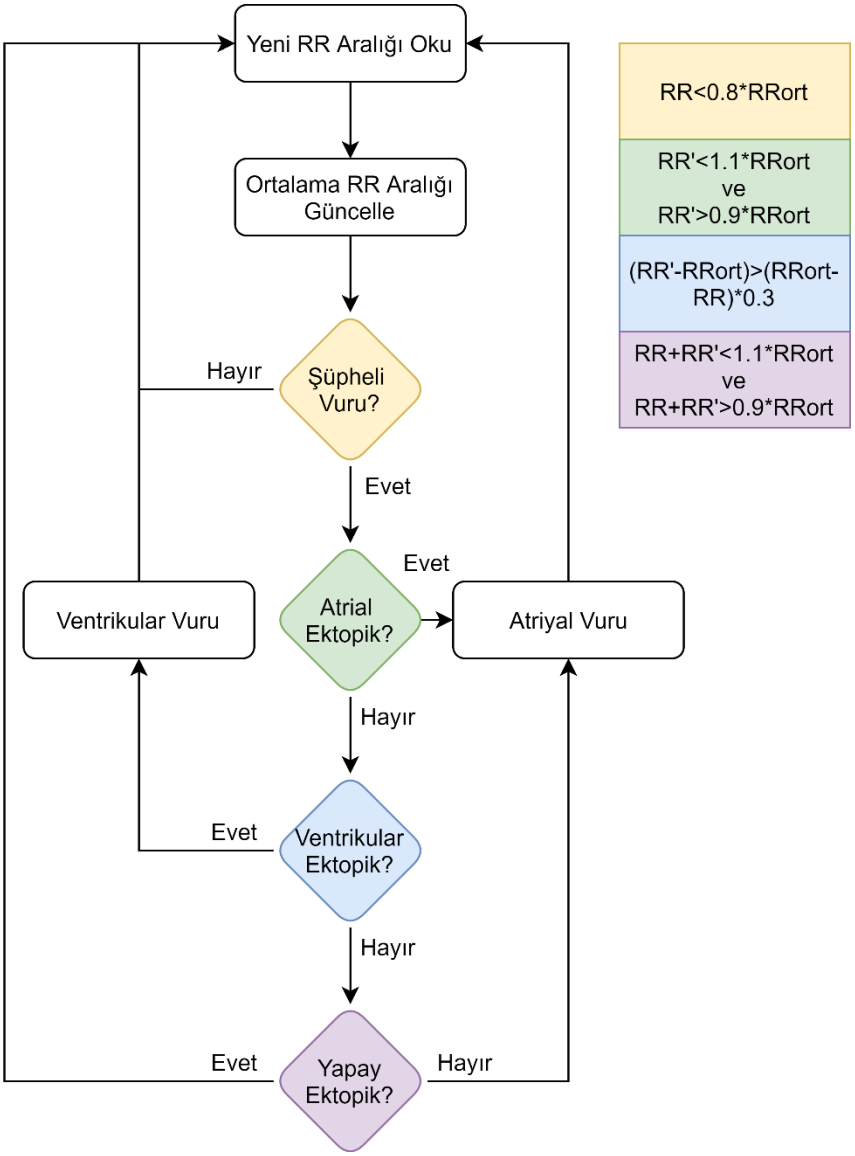


Şekil 3.3. R noktaları tespiti.

3.2. ÖNİŞLEMLER

KHD ve NKHD verilerinden veritabanları ve öznitelik veri setleri oluşturulurken veriler bazı önişlemlerden geçirilmişlerdir. Bunlar ektopik vuruların kaldırılması, eğilimin yok edilmesi ve verilerin yeniden örneklenmesi işlemleridir.

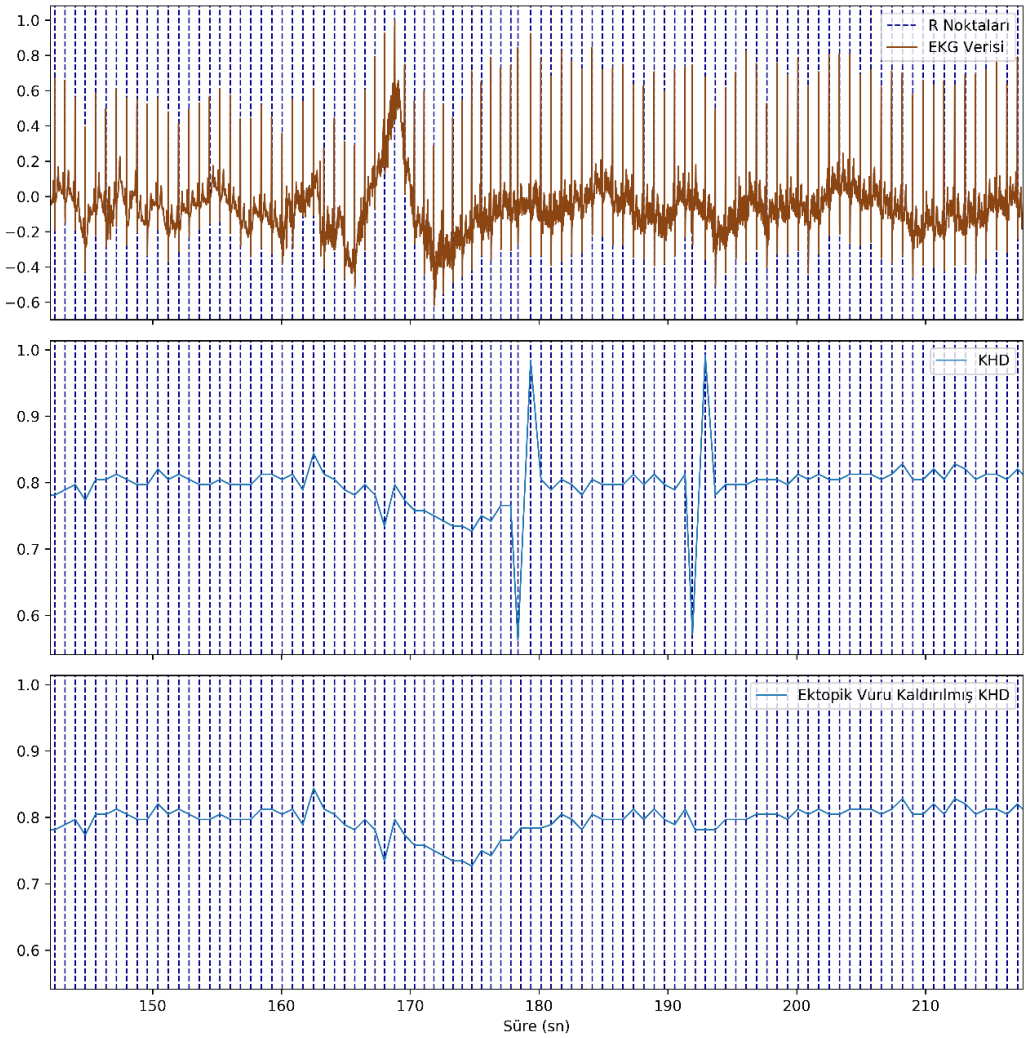
3.2.1. Kullanılan Ektopik Vuru Kaldırma Önişlemi



Şekil 3.4. Ektopik vuru kaldırma algoritması.

Ektopik vurular, sağlıklı bireylerde de oluşabildiği için bu tezde KHD işaretinden ayrılmıştır. Ortalama KHD değerinin %20 altına inen şüpheli atımdan bir sonraki atım ortalama KHD değerinin ±%10 içinde kalırsa şüpheli atım kulakçık kaynaklı ektopik (normalden farklı) vuru olarak değerlendirilir. Eğer takip eden atım süresi ortalama KHD

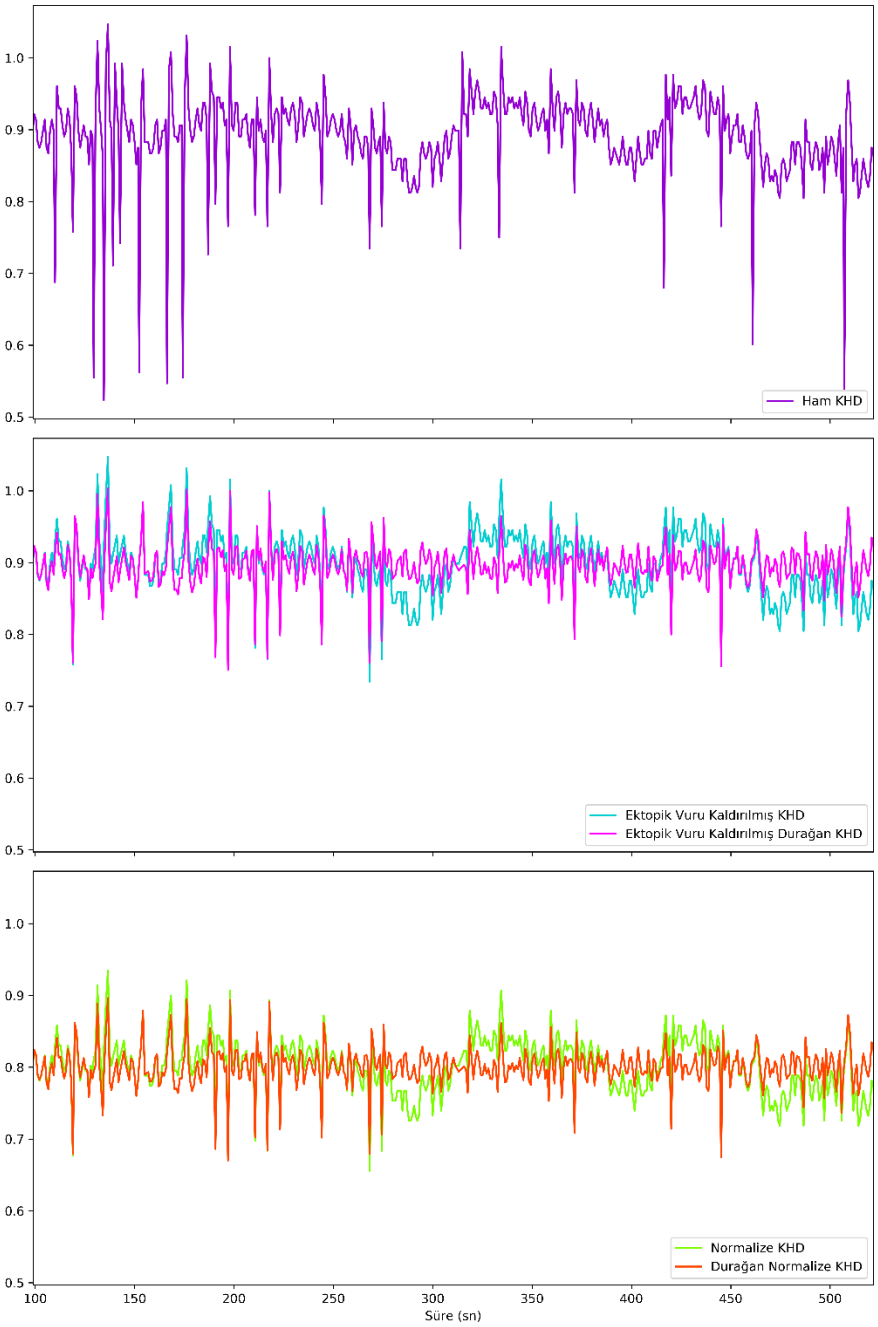
değerinden farkı, mevcut atım süresinin ortalama KHD değerinden farkının %30'unun üzerindeyse şüpheli vuru, karıncık (ventricular) kaynaklı ektopik vuru olmaktadır [72]. Kullanılan algoritma Şekil 3.4'de kısaca gösterilmiştir. Şekil 3.5'de ise p22 kodlu hastaya ait EKG verisinde ektopik vuru kaldırma işlemi gösterilmiştir. Tezde kullanılmış olan kod parçacığı Ek 1 içerisinde sunulmuştur. Burada ortalama KHD değeri hesabı için pencere uzunluğu on birim seçilmiş, kaldırılan ektopik vurular yerine ise doğrusal interpolasyon metodu uygulanarak yeni KHD değerleri hesaplanmıştır.



Şekil 3.5. Ektopik vuru kaldırılmış KHD verisi.

3.2.2. Kullanılan Eğilim Yok Etme Önişlemi

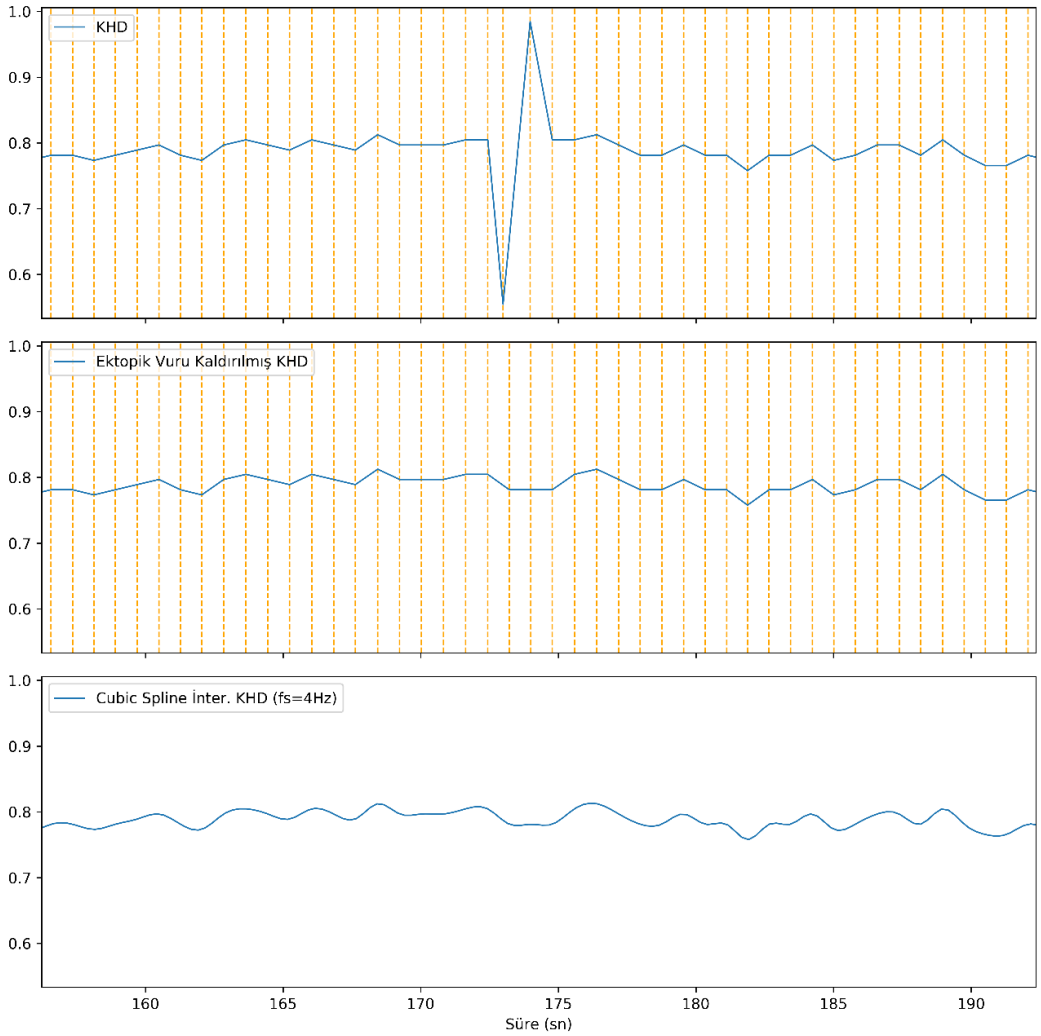
Güç dağılımı bilgisi çıkarılacak işaretlerin zaman içinde durağan ve eğilimsiz olması istenmektedir. KHD verisi eğilimli bir işarettir [68]. Bu sebeple KHD verisinden güç dağılımı çıkarılırken eğilim yok edilmiştir. Tezde, Tarvainen ve arkadaşlarının önerdiği Smoothness Prior yöntemi kullanılmıştır [70]. Kullanılan kod parçacığı Ek 1 içerisinde sunulmuştur. Şekil 3.6'da ise p11 kodlu hastaya ait KHD verisinin ektopik vuru kaldırma işlemi sonrasında eğilim yok edilmiş hali gösterilmektedir.



Şekil 3.6. Eğilimi yok edilmiş KHD verisi.

3.2.3. Kullanılan İnterpolasyon Önişlemi

KHD işaretlerine ait frekans bölgesi özniteliklerini oluşturmak amacıyla güç dağılım verilerini çıkarmak için kullanılan bazı dönüşüm yöntemleri, eşit aralıklı örneklenmiş işarete ihtiyaç duymaktadır. KHD işaretleri eşit zaman aralıklı olmadığı için tekrar örneklenmesi gerekmektedir. Bu tezde ilgilendiğimiz frekans aralığı 0-0.4 Hz değerindedir. Literatürde örnekleme işlemi için en az 1 Hz, en çok 10 Hz örnekleme frekansı kullanılması önerilmiştir. Bu tezde örnekleme frekansı 4 Hz olarak seçilmiştir. Frekans dönüşümünde keskin örnekleme değerleri sebebiyle doğrusal interpolasyon yerine Cubic Spline interpolasyon tercih edilmiştir. Şekil 3.7'de p09 kodlu hastaya ait Cubic Spline interpolasyon yöntemi, fs=4Hz örnekleme frekansı için gösterilmiştir. Bu tezde interpolasyon işlemi için Ek1 içerisinde sunulan kod parçacığı kullanılmıştır.



Şekil 3.7. Güç dağılımı işlemi öncesi Cubic Spline interpolasyon örneği.

3.3. ÖZNİTELİKLERİN ÇIKARILMASI

Zaman ve frekans bölgesi ölçümleri ile doğrusal olmayan ölçümlerden öznitelikler çıkarılırken ESC ve NASPE Task Force isimli grup tarafından sunulan önerilerden faydalanılmıştır [19]. Öznitelik çıkarımında Robin Champseix tarafından geliştirilen "HRV Analysis" kütüphanesi ile birlikte farklı python kütüphanelerinden de faydalanılarak kodlar geliştirilmiştir [95]. Kullanılan kod kısımları, kullanılan kütüphaneler ve örnek program çıktıları bu bölümde sunulmuştur.

3.3.1. Zaman Bölgesi Öznitelikleri

Zaman bölgesinde kullanılan öznitelikler Çizelge 2.2'de gösterilmiş, hesaplama yöntemleri ise Denklem (2.8) - (2.23) aralığında incelenmiştir. Örnek olarak p15 kodlu hastaya ait KHD ve NKHD verisi için hesaplanan değerler Çizelge 3.1'te gösterilmiş olup, kullanılan kod parçacığı Ek 1 içerisinde sunulmuştur.

Çizelge 3.1. Zaman bölgesi öznitelikleri.

Öznitelik	KHD	NKHD
mnn (ms)	739,097	800,000
sdnn (ms)	45,474	49,221
rmssd (ms)	14,054	15,212
sdsd (ms)	10,077	10,907
nn50 (adet)	9	13
pnn50 (%)	0,371	0,537
nn20 (adet)	307	307
pnn20 (%)	12,670	12,670
mednn (ms)	750,000	811,801
rangenn (ms)	429,688	465,095
cvsd (ms^2)	0,019	0,019
cvnn (ms^2)	0,062	0,062
mhr (bpm)	81,586	75,375
maxhr (bpm)	128,000	118,256
minhr (bpm)	66,783	61,699
stdhr (bpm)	6,657	6,151

3.3.2. Frekans Bölgesi Öznitelikleri

Frekans bölgesi öznitelikleri için bu çalışmada, güç spektral yoğunluğunun dört farklı frekans bandındaki güç miktarları ve oranları kullanılmıştır [19]. Bu tezde kullanılan frekans bölgesi öznitelikleri Çizelge 3.2'da gösterilmiştir. Güç spektrumu tahminini iyileştirmek ve işaretin frekans bölgesindeki yansıtmasını artırmak adına, ilk 6 öznitelik olan güç değerleri için Lomb-Scargle (LS), Hızlı Fourier (FFT), Welch metodu ve Ayrık Dalgacık dönüşümü (DWT) olmak üzere dört farklı yöntem ile hesaplamalar yapılmıştır. Ayrıca dalgacık dönüşümüne özel olarak bu dört bölgeye ait entropi değeri de öznitelik olarak kullanılmıştır.

Çizelge 3.2. Frekans bölgesi öznitelikleri.

Öznitelik	Açıklama
P_ULF (ms^2/Hz)	ULF bölgesindeki (0-0.003Hz) güç miktarı
P_VLF (ms^2/Hz)	VLF bölgesindeki (0.003-0.04Hz) güç miktarı
P_LF (ms^2/Hz)	LF bölgesindeki (0.04-0.15Hz) güç miktarı
P_HF (ms^2/Hz)	HF bölgesindeki (0.15-0.4Hz) güç miktarı
P_TOT (ms^2/Hz)	Tüm frekans bandındaki (0-0.4Hz) güç miktarı
P_LF/P_HF	LF bölgesindeki güç miktarının HF bölgesindeki güç miktarına oranı
E_ULF (bit)	ULF bölgesi için entropi
E_VLF (bit)	VLF bölgesi için entropi
E_LF (bit)	LF bölgesi için entropi
E_HF (bit)	HF bölgesi için entropi

Şekil 3.8'de p02 numaralı hastaya ait KHD verisinin Welch, FFT ve LS periyodogramları gösterilmiştir. Şekil 3.9'da ise aynı hastaya ait NKHD verisinden elde edilmiş periyodogramlar gösterilmiştir. Aynı işarete ait frekans bölgesi temsilleri benzer yapıda olsa da farklı dönüşüm yöntemleri için farklı detaylar ortaya çıkmaktadır. LS ve DWT dönüşümünün diğer dönüşümlerden farklı olarak düşük frekans bölgesinde daha fazla detay barındırdığı Çizelge 3.3 ve Çizelge 3.5'da görülmektedir. Bunun sebebi ise Welch metodu ve FFT dönüşümü öncesi kullanılması gereken eğilim yok etme işleminin yüksek geçiren filtre gibi davranması sonucu düşük frekanslı bileşenleri sönümlemesidir [71]. LS ve DWT dönüşümünde ise eğilim yok etme önişlemi gerekmediği için KHD işaretinin frekans bölgesinde ULF ve VLF kısımlarını daha iyi yansıttığı düşünülmektedir. Parseval

teoremine göre DWT dönüşümünün farklı ayrışma değerlerinden faydalanarak güç değerleri hesaplanmıştır [85].

Çizelge 3.3'de p02 kodlu hastaya ait KHD verilerinden elde edilen, dört farklı dönüşüme ait güç dağılımından hesaplanan güç değerleri gösterilmiştir. Çizelge 3.4'de ise yine aynı hastaya ait KHD verisinden elde edilen DWT dönüşümünden faydalanarak Shannon entropi değerleri hesaplanmıştır. Benzer şekilde aynı hastaya ait NKHD değerleri için güç değerleri Çizelge 3.5'da, entropi değerleri ise Çizelge 3.6'da gösterilmiştir. Frekans bölgesi özniteliklerini çıkarmada kullanılan Python kodları ise Ek 1 içerisinde sunulmuştur.

Çizelge 3.3. KHD güç dağılımı değerleri.

Bölge	Welch (10^{-7})	FFT (10^{-7})	LS (10^{-7})	DWT (10^{-7})
ULF Güç (ms^2/Hz)	0,006	1,072	2569,582	2770,781
VLF Güç (ms^2/Hz)	0,383	5,963	4375,867	3965,784
LF Güç (ms^2/Hz)	186,168	241,706	156,117	298,694
HF Güç (ms^2/Hz)	236,693	374,548	141,978	249,904
Toplam Güç (ms^2/Hz)	423,250	623,289	7243,545	7285,163
L/H Güç Oranı	0,787	0,645	1,100	1,195

Çizelge 3.4. KHD entropi değerleri.

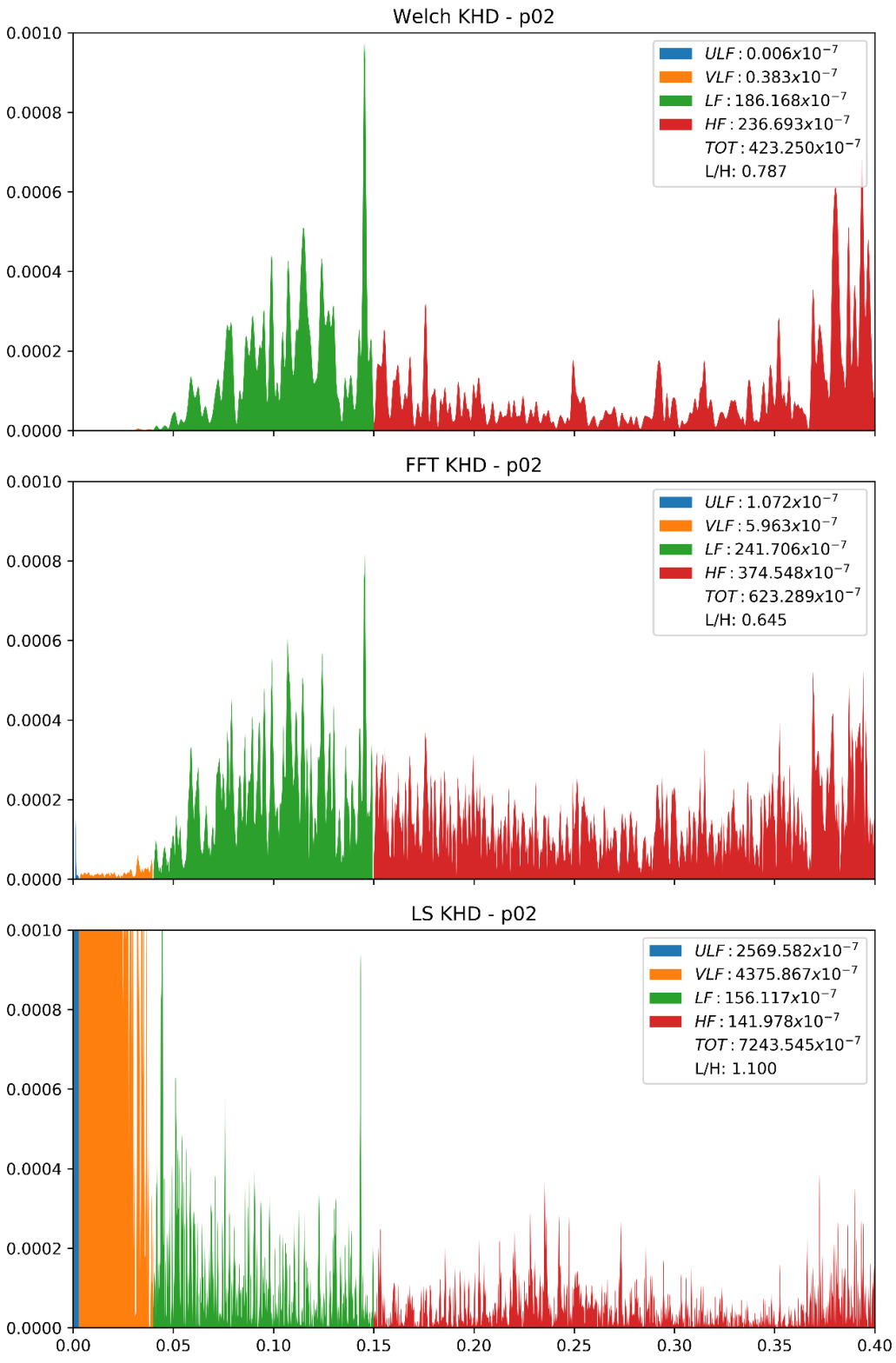
Bölge	Değer
ULF Entropi (bit)	2,557
VLF Entropi (bit)	17,191
LF Entropi (bit)	28,687
HF Entropi (bit)	36,744

Çizelge 3.5. NKHD güç dağılımı değerleri.

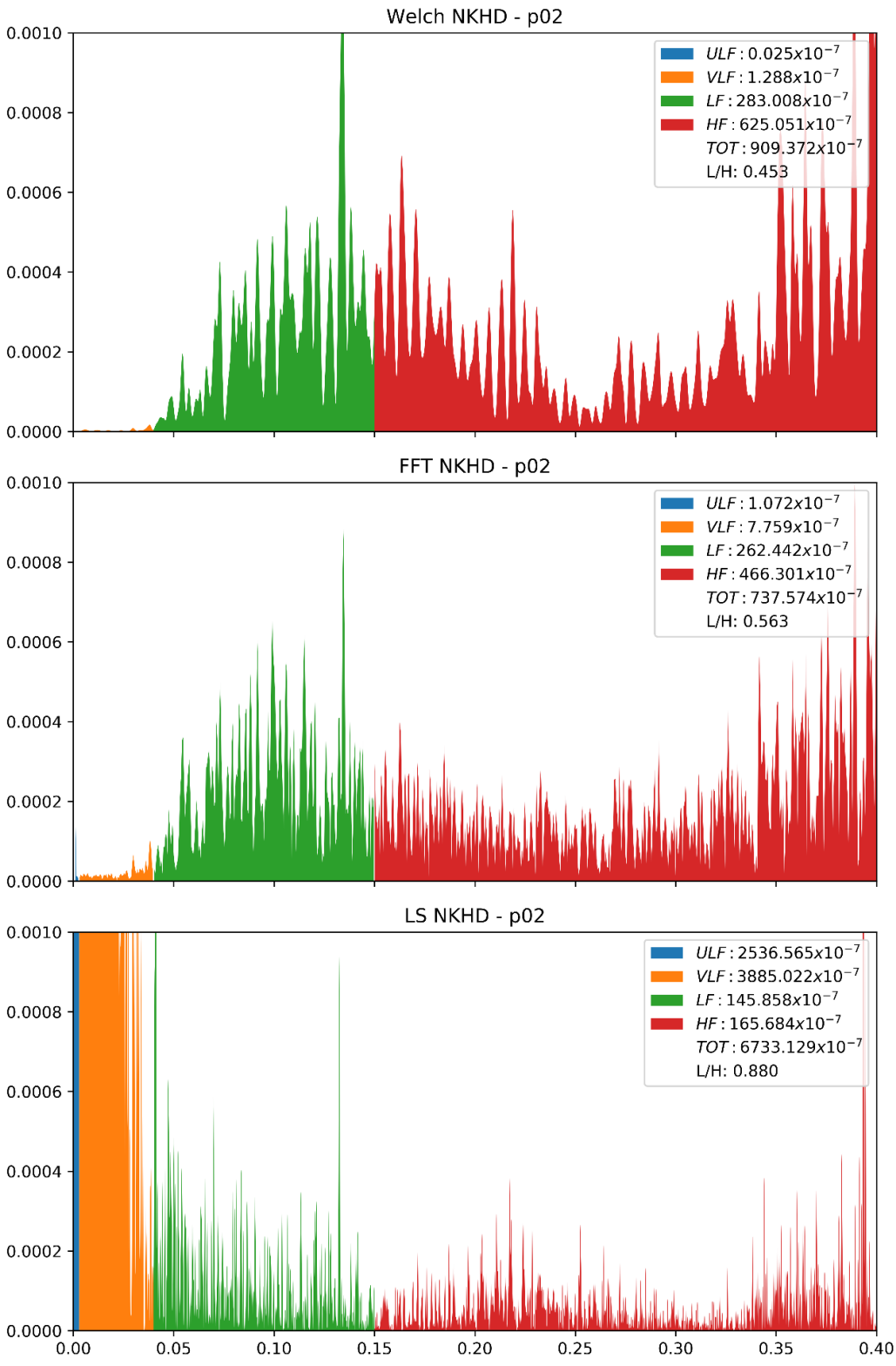
Bölge	Welch (10-7)	FFT (10-7)	LS (10-7)	DWT (10-7)
ULF Güç (ms2/Hz)	0,025	1,072	2536,565	3588,306
VLF Güç (ms2/Hz)	1,288	7,759	3885,022	4707,359
LF Güç (ms2/Hz)	283,008	262,442	145,858	337,781
HF Güç (ms2/Hz)	625,051	466,301	165,684	267,197
Toplam Güç (ms2/Hz)	909,372	737,574	6733,129	8900,644
L/H Güç Oranı	0,453	0,563	0,880	1,264

Çizelge 3.6. NKHD entropi değerleri.

Bölge	Değer
ULF Entropi (bit)	2,998
VLF Entropi (bit)	17,741
LF Entropi (bit)	29,479
HF Entropi (bit)	38,034



Şekil 3.8. KHD güç dağılımı örnekleri.



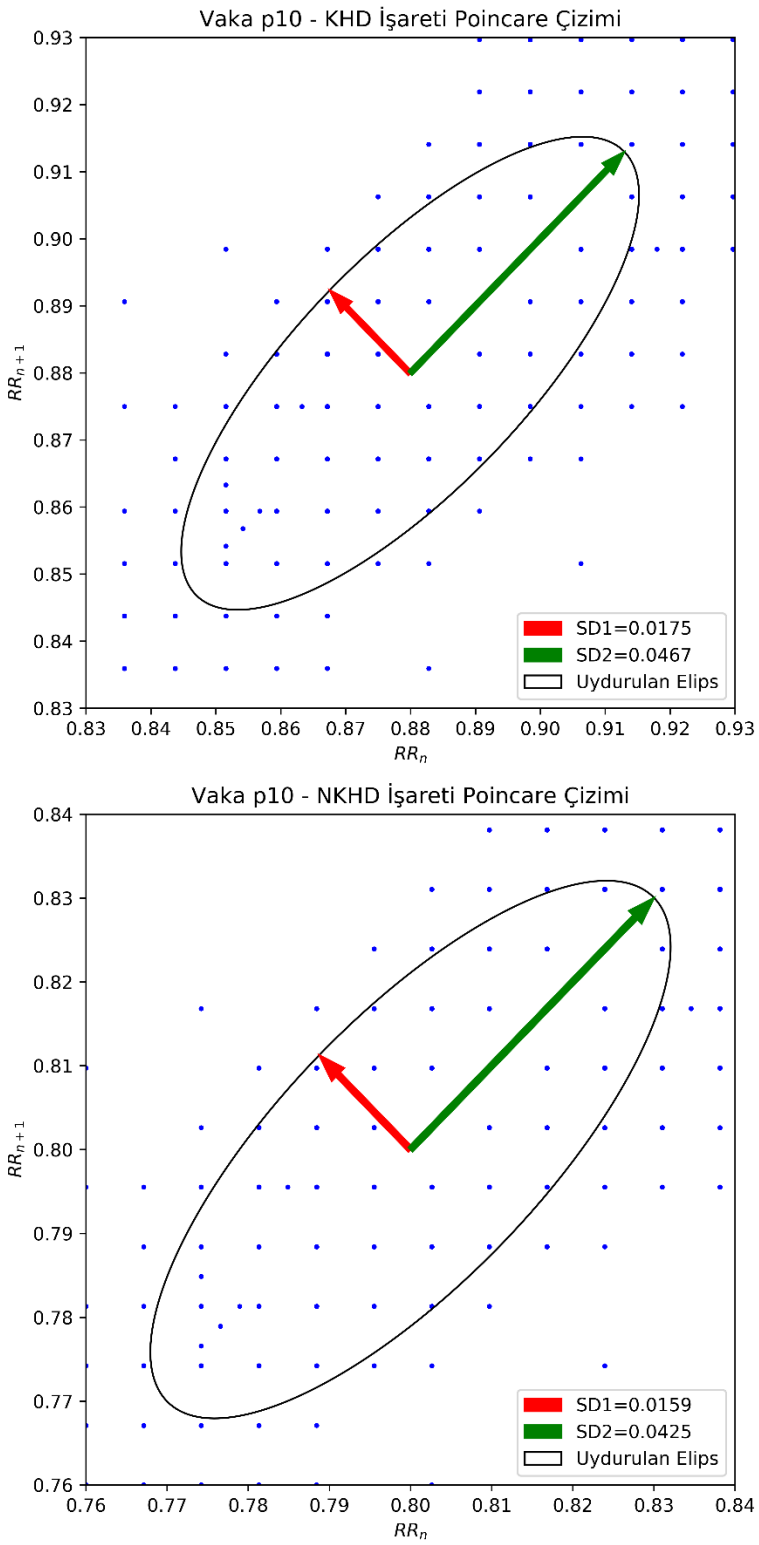
Şekil 3.9. NKHD güç dağılımı örnekleri.

3.3.3. Poincare Çizimi Öznitelikleri

Bu çalışmada KHD verisindeki sempatik ve parasempatik bileşenlerden oluşan doğrusal olmayan büyüklüklerin tespiti amacıyla Poincare Çizimi üzerinden elde edilen öznitelikler kullanılmıştır. RR aralıkları arasındaki farklardan oluşan noktalara ait mesafelerin standart sapmaları sayesinde Poincare çizimine elips uydurma işlemi gerçekleştirilmiştir. Uydurulan elipsin genişliği ve uzunluğu ile bu büyüklüklerin oranı ve elipsin alanı öznitelik olarak seçilmiştir. Çizelge 3.7'de p10 kodlu vakanın KHD ve NKHD verilerine ait Poincare çizimi öznitelikleri sunulmuş olup, Şekil 3.10'da uydurulan elipslerle birlikte Poincare çizimleri gösterilmiştir. Ek 1 içerisinde ise Poincare çizimi özniteliklerinin hesaplanmasında kullanılan Python kodu verilmiştir.

Çizelge 3.7. p10 vakası için Poincare çizimi öznitelikleri.

Öznitelik/İşaret	KHD	NKHD
SD1 (sn)	0,0175	0,0159
SD2 (sn)	0,0467	0,0425
SD2/SD1	2,6668	2,6668
Alan (sn^2)	0,0026	0,0021



Şekil 3.10. KHD ve NKHD için Poincare çizimi örneği.

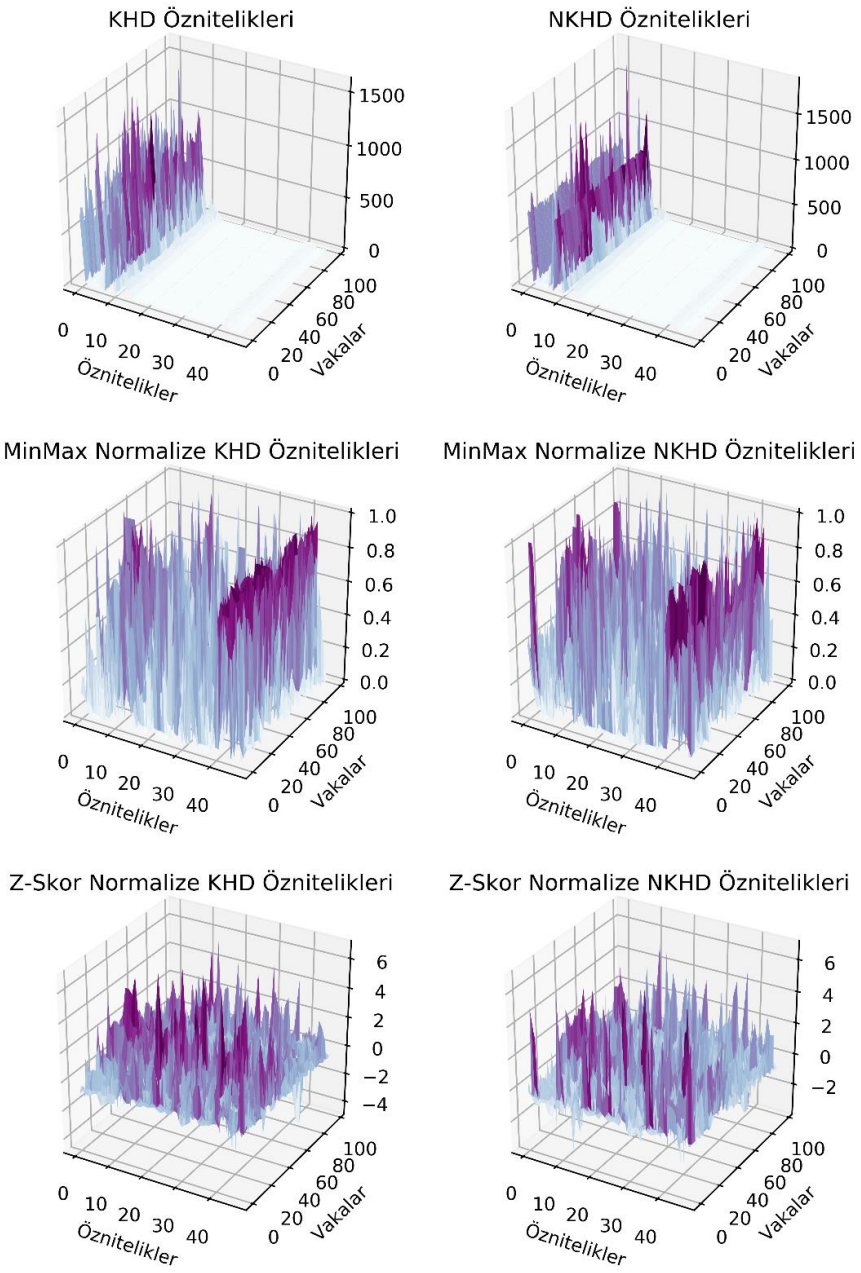
3.4. ÖZNİTELİK NORMALİZASYONU

Çalışmada kullanılan özniteliklere ait ölçek farklarının, sınıflandırma performansını olumsuz etkilemesi sebebiyle öznitelik setlerinin ham hallerinin yanı sıra MinMax ve Z-Skor normalizasyonuna tabi tutulmuş halleriyle de ayrıca çalışılmıştır. Şekil 3.11'de öznitelikler arasındaki farklı ölçeklerin MinMax ve Z-skor normalizasyon işlemleri sonrası eşitlenmesi görselleştirilmiştir. Kullanılan özniteliklere dair ortalamaları, potansiyel aykırı değerleri, orta değerleri, veri aralıklarını detaylı gösteren kutu grafikleri (Boxplot) ise Ek1 içerisinde Şekil 7.1, Şekil 7.2 ve Şekil 7.3'te sunulmuştur. Özniteliklerin sınıf verilerine göre dağılımlarını gösteren kutu grafikleri ise Ek2 içerisinde sunulmuştur. Görev 1 için KHD verilerinden elde edilen özniteliklerin dağılımı Şekil 7.4, Şekil 7.5 ve Şekil 7.6'da; NKHD verilerinden elde edilen özniteliklerin dağılımı ise Şekil 7.7, Şekil 7.8 ve Şekil 7.9 ile sunulmuştur. Benzer şekilde Görev 2 için KHD verilerinden elde edilen özniteliklerin dağılımı Şekil 7.10, Şekil 7.11 ve Şekil 7.12'de; son olarak NKHD verilerinden elde edilen öznitelik dağılımları ise Şekil 7.13, Şekil 7.14 ve Şekil 7.15 ile sunulmuştur.

Öznitelik normalizasyon işlemleri sonrası sınıflandırma performanslarının arttığı gözlemlenmiştir. MinMax normalizasyonunda öznitelik değerleri [0,1] aralığına çekilirken, Z-skor yada diğer adıyla standart skor normalizasyonunda ise değerler özniteliklerin aritmetik ortalaması ve standart sapması ile ilişkilendirilerek ölçeklenmektedir. Çizelge 3.8'da çalışmada kullanılan öznitelik setleri için normalizasyon işlemleri öncesi ve sonrası verilerin aralıkları gösterilmiştir. Ek 1 içerisinde ise öznitelik normalleştirme için kullanılan kod parçacığı sunulmuştur.

Çizelge 3.8. Öznitelik setleri için veri değer aralıkları.

Normalizasyon	KHD		NKHD	
	En küçük aralık	En büyük aralık	En küçük aralık	En büyük aralık
Yok	0,00029	1589	0,00028	1846
MinMax	1	1	1	1
Z-Skor	3,605	7,405	3,888	9,765



Şekil 3.11. KHD ve NKHD verisine ait tüm özniteliklerin normalizasyonu.

3.5. GENETİK ALGORİTMA

Öznitelik seçimi için Kopczyk tarafından yazılmış olan basit GA sınıflandırıcı, özelleştirilerek kullanılmıştır [96]. Ek 1 içerisinde kNN sınıflandırıcı için özelleştirilmiş kod parçacığı sunulmuştur. Örnek olarak kNN sınıflandırıcı için seçilen öznitelikler "1" ile gösterilecek şekilde Çizelge 3.9'de GA öznitelik seçimi sonuçları aktarılmıştır.

Çizelge 3.9. GA ile kNN sınıflandırıcısı kullanılarak seçilen öznitelikler. 1 yazan hücrenin bulunduğu satırdaki öznitelik, sütundaki görev, normalizasyon tekniği ve veri için kullanıldığını göstermektedir. 0 yazan hücre ise o özniteliğin GA tarafından seçilmediğini göstermektedir.

Veri Türü	KHD						NKHD					
Görev	Görev 1			Görev 2			Görev 1			Görev 2		
Öznitelik Norm. - Öznitelikler	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
k	7	1	1	5	17	3	1	1	1	3	1	5
Öznitelik Sayısı	23	27	23	24	18	15	24	19	20	27	18	17
mnn	1	0	1	0	0	0	0	1	1	1	0	0
sdnn	1	1	1	0	0	0	1	0	0	0	1	1
rmssd	0	1	1	0	0	0	0	0	1	0	0	0
sdsd	1	1	1	1	0	1	1	0	0	0	0	1
nn50	0	0	0	0	0	1	0	0	0	0	0	1
pnn50	0	0	0	0	0	0	1	0	1	1	0	0
nn20	0	1	1	0	0	1	0	1	0	1	1	0
pnn20	1	0	1	0	0	1	1	1	0	0	1	0
medn n	1	1	0	0	1	0	0	0	1	1	0	0
rangenn	0	0	1	0	0	1	0	0	1	0	1	0
cvsd	1	1	0	0	0	1	1	0	0	0	1	0
cvnn	1	0	1	1	1	0	0	1	0	0	0	1
mhr	0	0	1	1	0	0	1	0	0	1	0	0
maxhr	0	1	1	1	1	0	1	0	1	1	1	1
minhr	0	1	1	1	0	0	1	1	1	0	0	0
stdhr	1	1	1	0	1	0	0	0	0	1	1	1

Çizelge 3.9. (devam) GA ile kNN sınıflandırıcısı kullanılarak seçilen öznitelikler. 1 yazan hücrenin bulunduğu satırdaki öznitelik, sütundaki görev, normalizasyon tekniği ve veri için kullanıldığını göstermektedir. 0 yazan hücre ise o özniteliğin GA tarafından seçilmediğini göstermektedir.

Veri Türü	KHD						NKHD					
Görev	Görev 1			Görev 2			Görev 1			Görev 2		
Öznitelik Norm. - Öznitelikler	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
fft_ulf	0	1	0	1	1	0	1	1	1	1	0	0
fft_vlf	0	1	1	0	0	0	0	0	1	0	1	0
fft_lf	0	0	0	0	0	1	1	1	1	0	0	0
fft_hf	1	1	1	1	0	0	1	1	1	1	0	0
fft_total	1	1	1	0	1	0	1	0	1	0	1	0
fft_lftohf	0	0	0	1	1	0	1	0	0	1	1	0
welch_ulf	1	0	1	1	0	0	0	1	1	0	0	0
welch_vlf	0	1	0	0	0	0	1	1	0	1	0	0
welch_lf	0	1	1	1	0	0	0	0	0	0	0	1
welch_hf	1	1	0	1	0	0	1	0	0	1	1	0
welch_total	1	1	0	1	0	0	1	1	1	1	0	1
welch_lftohf	1	0	0	0	0	0	0	0	0	1	0	1
ls_ulf	0	0	0	1	0	0	0	0	0	1	0	0
ls_vlf	1	0	0	0	1	0	0	0	1	0	0	1
ls_lf	0	0	0	0	1	0	0	1	0	1	0	1
ls_hf	0	0	1	1	0	0	1	0	1	0	0	0
ls_total	1	1	0	1	0	0	0	0	0	0	1	0
ls_lftohf	0	1	0	1	0	0	0	0	0	1	1	1
wt_ulf	0	1	0	1	1	0	1	1	1	0	0	0
wt_vlf	1	0	0	1	1	0	0	1	0	0	0	0
wt_lf	1	1	1	0	1	1	1	0	0	1	0	0
wt_hf	0	0	0	0	1	0	0	0	0	1	1	1
wt_total	0	1	0	1	1	1	1	1	0	1	0	1
wt_lftohf	0	0	1	1	1	0	1	0	1	0	0	0
wt_ent_ulf	1	0	0	0	1	1	1	0	0	0	1	1

Çizelge 3.9. (devam) GA ile kNN sınıflandırıcısı kullanılarak seçilen öznitelikler. 1 yazan hücrenin bulunduğu satırdaki öznitelik, sütundaki görev, normalizasyon tekniği ve veri için kullanıldığını göstermektedir. 0 yazan hücre ise o özniteliğin GA tarafından seçilmediğini göstermektedir.

Veri Türü	KHD						NKHD					
Görev	Görev 1			Görev 2			Görev 1			Görev 2		
Öznitelik Norm. - Öznitelikler	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
wt_ent_vlf	1	0	0	1	1	0	1	0	0	1	1	1
wt_ent_lf	0	1	0	0	0	1	0	0	0	1	1	0
wt_ent_hf	1	1	1	1	0	1	0	0	0	1	0	0
pc_sd1	0	1	0	1	0	1	0	1	0	1	0	0
pc_sd2	1	0	1	1	1	0	0	1	1	1	0	1
pc_sd2tosd1	0	1	0	0	0	1	1	1	1	1	0	0
pc_area	1	1	1	0	0	1	0	1	0	1	1	0

3.6. KULLANILAN SINIFLANDIRICILARIN ÇIKTILARI

Örüntü tanıma problemlerinde önemli adımlardan birisi de probleme özgün yaklaşımdır. Sınıflandırma temelli problemlerde sınıflandırıcı seçimi başarımı etkileyen faktörlerin başında gelmektedir. Bu çalışmadaki ana amacımız KHD verilerinin normalize edilmesinin Görev 1 ve Görev 2 için başarımı artırıp artırmadığını tespit etmektir. Bu amaçla öncelikle diğer sınıflandırıcılara göre karmaşıklığı düşük, geliştirmesi basit ve parametrik olmayan yapısısı sebebiyle kNN sınıflandırıcı ile çalışma yapılmış, NKHD analizinin başarıma olumlu etkisi kanıtlandıktan sonra toplam başarımı artırma adına MLP, SVM ve son olarak da DL sınıflandırma için CNN üzerinde çalışılmıştır.

3.6.1. k En Yakın Komşuluk Sonuçları

kNN sınıflandırmada GA ile öznitelik seçimi yapılmıştır. Doğrulama için birini hariç bırakma (Leave-One-Out) yöntemi kullanılmıştır. Tüm özniteliklerin kullanıldığı ve GA ile seçilen özniteliklerin kullanıldığı durumlarda k=1,3,5,…,23 değerleri için kNN sınıflandırıcı çıkışları hesaplanmış ve sınıflandırma raporları kaydedilmiştir. Ek 1 içerisinde kullanılan kNN sınıflandırıcı kod parçacığı verilmiştir. Çizelge 3.9'de verilen,

GA ile seçilmiş özniteliklerin yanısıra tüm öznitelikler için ayrı ayrı sınıflandırıcı çalıştırılmış ve Çizelge 3.10'te sunulan başarımlar elde edilmiştir. GA ile öznitelik seçimi başarımı önemli derecede artırmış, öznitelik normalizasyonu da beklendiği gibi başarımda artışa sebep olmuştur. Böylelikle KHD verilerinin normalize edilmiş hali olan NKHD verilerinin sınıflandırmada kullanılmasının, başarımı artırdığı ispatlanmıştır.

Çizelge 3.10. kNN sınıflandırıcı başarımları.

	Veri Türü	KHD						NKHD					
	Görev	Görev 1			Görev 2			Görev 1			Görev 2		
	Normalizasyon	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
Tüm Öznitelikler	k	1	1	1	1	9	9	13	1	1	21	17	13
	Kesinlik (%)	54,17	59,65	65,38	62,50	55,00	60,00	68,52	67,24	67,27	56,52	64,71	65,00
	Duyarlılık (%)	52,00	68,00	68,00	60,00	44,00	48,00	74,00	78,00	74,00	52,00	44,00	52,00
	F1 Skor (%)	53,06	63,55	66,67	61,22	48,89	53,33	71,15	72,22	70,48	54,17	52,38	57,78
	Doğruluk (%)	54,00	61,00	66,00	62,00	54,00	58,00	70,00	70,00	69,00	56,00	60,00	62,00
GA ile Öznitelik Seçimi	k	7	1	1	5	17	3	1	1	1	3	1	5
	Kesinlik (%)	82,00	84,00	78,00	68,00	60,00	80,00	82,00	86,00	88,00	84,00	80,00	60,00
	Duyarlılık (%)	71,93	79,25	81,25	77,27	83,33	76,92	78,85	84,31	84,62	75,00	83,33	78,95
	F1 Skor (%)	76,64	81,55	79,59	72,34	69,77	78,43	80,39	85,15	86,27	79,25	81,63	68,18
	Doğruluk (%)	75,00	81,00	80,00	74,00	74,00	78,00	80,00	85,00	86,00	78,00	82,00	72,00

3.6.2. Çok Katmanlı Algılayıcı Sonuçları

MLP sınıflandırıcı için Gad tarafından geliştirilen PyGad kütüphanesi özelliştirilerek kullanılmıştır [97]. Ek 1 içerisinde örnek kod sunulmuştur. MLP sınıflandırıcı için iki farklı çalışma yapılmıştır. İlk olarak sonuçları analiz etme adına sadece Görev 1 için çalışılmış olup, GA ile öznitelik seçimi yapılmıştır. Algoritmanın kapsayıcılığından emin olunamadığı için kullanılan algoritma iyileştirilmeye çalışılmıştır. Bu değişikliklerden ilki GA ile ağ katsayıları hesaplanmış, öznitelik seçimi yapılmamıştır. Burada da ağ seçilirken ilk çalışmanın aksine 30 nöron yerine 80 nöronluk tek gizli katmana sahip basit bir ağ ile sınıflandırma ölçümleri alınmış, veri setleri %90 eğitim, %10 test için kullanılmıştır. Ayrıca ezberi önleme adına, ikinci çalışmada iyileştirme yapılamayan iterasyon sınırı 100 olarak belirlenmiştir. Sınıflandırıcı başarımları Çizelge 3.11 ve Çizelge 3.12'de gösterilmiştir. NKHD verisi kullanımı çok katmanlı algılayıcı için de

başarımı artırmıştır. İterasyon sınırı konulması ve öznitelik seçimi yapılmaması başarımı bir nebze düşürse de, daha kapsayıcı bir sınıflandırıcı elde edilmiştir.

Çizelge 3.11. MLP sınıflandırıcı iterasyon sınırı olmadan başarımlar.

Veri Türü	KHD			NKHD		
Görev	Görev 1			Görev 1		
Normalizasyon	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
Kesinlik (%)	62,00	88,00	90,00	78,00	74,00	96,00
Duyarlılık (%)	88,57	86,27	93,75	75,00	92,50	96,00
F1 Skor (%)	72,94	87,13	91,84	76,47	82,22	96,00
Doğruluk (%)	76,77	86,87	91,92	75,76	83,84	95,96

Çizelge 3.12. MLP sınıflandırıcı iterasyon sınırı olan başarımlar.

Veri Türü	KHD						NKHD					
Görev	Görev 1			Görev 2			Görev 1			Görev 2		
Normalizasyon	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
Kesinlik (%)	68,25	86,96	91,49	81,82	87,50	95,83	72,00	88,37	88,46	80,95	85,00	100,0
Duyarlılık (%)	87,76	81,63	87,76	72,00	84,00	92,00	73,47	77,55	93,88	68,00	68,00	96,00
F1 Skor (%)	76,79	84,21	89,58	76,60	85,71	93,88	72,73	82,61	91,09	73,91	75,56	97,96
Doğruluk (%)	73,47	84,69	89,80	77,55	85,71	93,88	72,45	83,67	90,82	75,51	77,55	97,96

3.6.3. Destek Vektör Makinesi Sonuçları

SVM sınıflandırmada doğrusal ve RBF çekirdek fonksiyonları kullanılmıştır. Farklı C değerleri ve RBF için özel olarak farklı γ değerleri için sınıflandırıcı çalıştırılmış ve doğruluk değerine göre parametreler optimize edilmiştir. Bu sınıflandırıcı için iki farklı çalışma yapılmıştır. İlk çalışmada çapraz doğrulama yapılmamış, GA ile öznitelik seçimi yapılmış ve veri seti %80 eğitim, %20 test amaçlı ayrılmıştır. İkinci çalışmada, ilk çalışmanın çapraz doğrulama eksiği ve tüm verilerin kullanımı durumundaki başarımın ölçülmemesi sebebiyle algoritma iyileştirilerek daha kapsayıcı bir sınıflandırıcı tasarlanmıştır. İkinci çalışma için veri seti %85 eğitim, %15 test için ayrılmış ve 10 katlamalı çapraz doğrulama (10-fold cross validation) ile sonuçlar bulunmuştur. İlk

algoritmada doğrusal ve RBF çekirdek fonksiyonlarının her ikisi için ve her iki görev için NKHD analizi daha başarılı sonuçlar üretmiştir. İyileştirilen algoritmada ise farklı olarak iki çekirdek fonksiyonu için de Görev 1'de KHD verisi analizi, Görev 2'de ise NKHD verisi analizi daha başarılı olmuştur. İkinci çalışmada görülmüştür ki, öznitelik seçiminin başarıma etkisi olsa da sınırlıdır. Ek 1 içerisinde SVM sınıflandırmada kullanılan iyileştirilmiş ana kod parçacığı verilmiştir. Çizelge 3.13'da doğrusal çekirdek fonksiyonuna, Çizelge 3.14'da ise RBF çekirdek fonksiyonuna sahip birinci algoritmaya ait SVM sınıflandırıcı başarımları özetlenmiştir. Benzer şekilde genişletilmiş ikinci algoritma için ise Çizelge 3.15'de doğrusal, Çizelge 3.16'de ise RBF çekirdek fonksiyonlu SVM başarımları verilmiştir.

Çizelge 3.13. SVM doğrusal sınıflandırıcı birinci algoritma başarımları.

	Veri Türü	KHD						NKHD					
	Görev	Görev 1			Görev 2			Görev 1			Görev 2		
	Normalizasyon	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
GA ile Öznitelik Seçimi	C	1	0,01	0,1	0,1	5	10	1	1	0,1	5	0,001	5
	Kesinlik (%)	72,00	67,19	61,04	77,78	76,19	86,36	69,81	71,43	61,04	85,19	0,00	88,00
	Duyarlılık (%)	73,47	87,76	95,92	56,00	64,00	76,00	75,51	81,63	95,92	92,00	0,00	88,00
	F1 Skor (%)	72,73	76,11	74,60	65,12	69,57	80,85	72,55	76,19	74,60	88,46	0,00	88,00
	Doğruluk (%)	72,45	72,45	67,35	69,39	71,43	81,63	71,43	74,49	67,35	87,76	48,98	87,76

Çizelge 3.14. SVM RBF sınıflandırıcı birinci algoritma başarımları.

	Veri Türü	KHD						NKHD					
	Görev	Görev 1			Görev 2			Görev 1			Görev 2		
	Normalizasyon	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
GA ile Öznitelik Seçimi	C	1000	100	10	10	1	1e+09	1e+06	1000	1e+06	100	1e+07	100
	γ	0,0001	0,01	0,01	0,0001	1	1e-08	1e-08	0,01	1e-06	1e-05	1e-07	0,01
	Kesinlik (%)	70,97	84,62	67,14	0,00	94,74	73,08	59,72	97,73	79,17	0,00	90,00	70,69
	Duyarlılık (%)	89,80	89,80	95,92	0,00	72,00	76,00	87,76	87,76	77,55	0,00	72,00	82,00
	F1 Skor (%)	79,28	87,13	78,99	0,00	81,82	74,51	71,07	92,47	78,35	0,00	80,00	75,93
	Doğruluk (%)	76,53	86,73	74,49	48,98	83,67	73,47	64,29	92,86	78,57	48,98	81,63	73,74

Çizelge 3.15. SVM doğrusal sınıflandırıcı ikinci algoritma başarımları.

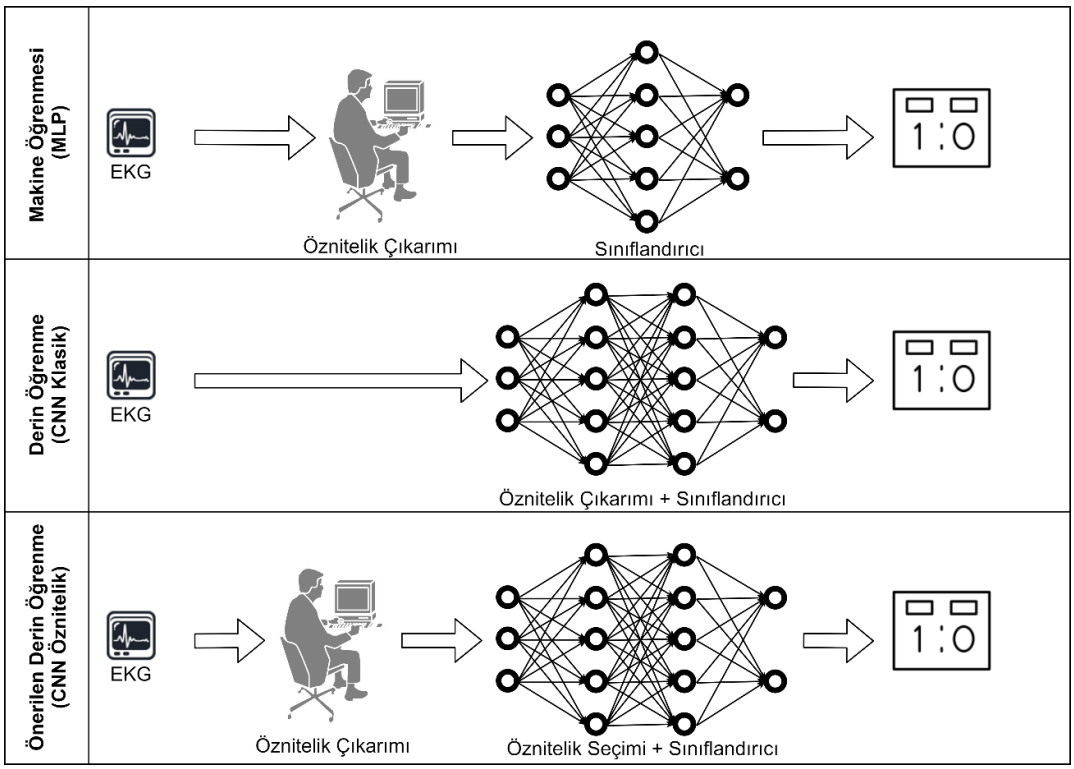
	Veri Türü	KHD						NKHD					
	Görev	Görev 1			Görev 2			Görev 1			Görev 2		
	Normalizasyon	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
Tüm Öznitelikler	C	10	10	0,1	0,001	1	0,1	0,001	10	0,1	0,001	50	0,1
	Kesinlik (%)	89,36	89,36	73,68	0,00	85,71	86,36	52,69	81,25	71,19	0,00	100,0	86,96
	Duyarlılık (%)	85,71	85,71	85,71	0,00	72,00	76,00	100,0	79,59	85,71	0,00	88,00	80,00
	F1 Skor (%)	87,50	87,50	79,25	0,00	78,26	80,85	69,01	80,41	77,78	0,00	93,62	83,33
	Doğruluk (%)	87,76	87,76	77,55	48,98	79,59	81,63	55,10	80,61	75,51	48,98	93,88	83,67
GA ile Öznitelik Seçimi	C	50	1	0,1	0,001	1	0,1	1	50	1	50	50	10
	Kesinlik (%)	81,63	71,19	61,25	0,00	82,35	76,19	67,24	84,44	75,47	95,65	87,50	88,00
	Duyarlılık (%)	81,63	85,71	100,0	0,00	56,00	64,00	79,59	77,55	81,63	88,00	84,00	88,00
	F1 Skor (%)	81,63	77,78	75,97	0,00	66,67	69,57	72,90	80,85	78,43	91,67	85,71	88,00
	Doğruluk (%)	81,63	75,51	68,37	48,98	71,43	71,43	70,41	81,63	77,55	91,84	85,71	87,76

Çizelge 3.16. SVM RBF sınıflandırıcı ikinci algoritma başarımları.

	Veri Türü	KHD						NKHD					
	Görev	Görev 1			Görev 2			Görev 1			Görev 2		
	Normalizasyon	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
Tüm Öznitelikler	C	0,001	10	50	1	10	50	0,001	50	50	10	50	50
	γ	0,001	0,1	0,001	0,001	0,1	0,001	0,001	0,1	0,001	0,001	0,1	0,001
	Kesinlik (%)	50,00	97,96	73,68	0,00	91,67	81,82	50,00	94,23	68,85	81,82	96,00	86,36
	Duyarlılık (%)	100,0	97,96	85,71	0,00	88,00	72,00	100,0	100,0	85,71	72,00	96,00	76,00
	F1 Skor (%)	66,67	97,96	79,25	0,00	89,80	76,60	66,67	97,03	76,36	76,60	96,00	80,85
	Doğruluk (%)	50,00	97,96	77,55	48,98	89,80	77,55	50,00	96,94	73,47	77,55	95,92	81,63
GA ile Öznitelik Seçimi	C	10	10	10	1	10	50	50	10	50	1	50	1
	γ	0,001	10	1	0,001	0,1	0,001	0,001	10	0,001	0,001	0,1	1
	Kesinlik (%)	60,76	84,48	100,0	0,00	90,91	85,00	62,86	84,48	70,69	0,00	96,15	95,83
	Duyarlılık (%)	97,96	100,0	95,92	0,00	80,00	68,00	89,80	100,0	83,67	0,00	100,0	92,00
	F1 Skor (%)	75,00	91,59	97,92	0,00	85,11	75,56	73,95	91,59	76,64	0,00	98,04	93,88
	Doğruluk (%)	67,35	90,82	97,96	48,98	85,71	77,55	68,37	90,82	74,49	48,98	97,96	93,88

3.6.4. Derin Öğrenme Sonuçları

Derin öğrenme sınıflandırma işleminde CNN sınıflandırıcı kullanılmıştır. İlk olarak doğrudan KHD ve NKHD verileri üzerinden klasik CNN ile çalışma yapılmıştır. Başarımın mevcut ağ parametre uzayı için düşük çıkması sebebiyle daha önce farklı sınıflandırıcılarda kullanılıp başarılı olunan öznitelik seti üzerinden CNN ile öznitelik seçimi işlemi yapılarak derin öğrenme sınıflandırıcısı tasarlanmıştır. Önerilen modelin yapısı Şekil 3.12'te gösterilmiştir. Ek 1 içerisinde CNN sınıflandırıcı için kullanılan kod örneği verilmiştir. Çizelge 3.17'de KHD ve NKHD verileri üzerinden CNN sınıflandırma başarımları, Çizelge 3.18'de ise öznitelik veri setleri üzerinden CNN sınıflandırma başarımları verilmiştir. Sınıflandırıcı hiper parametreleri hyperopt kütüphanesi sayesinde optimize edilmiş ve sunulmuştur. Burada "Filtre adet" parametresi CNN Evrişim katmanında aynı anda kaç farklı evrişim yapıldığını gösterir. "Filtre boyut" parametresi, evrişim pencere boyutunun büyüklüğünü belirtir, öznitelikler giriş olarak seçildiğinde öznitelik boyutuna eşit seçilmiştir. "Batch Size" parametresi, ağın verileri hangi büyüklükte işleyeceğini belirtir. "Dropout" parametresi seyreltme oranını, "Pooling" parametresi havuzlama matrisinin boyutunu ve son olarak "Dense" parametresi ise CNN ağı çıkışında bulunan tam bağlantılı katmanı temsil etmektedir.



Şekil 3.12. Önerilen derin öğrenme sınıflandırıcı modeli.

Ham KHD verileri ile çalışılırken başarım görece düşük çıkmıştır. Her iki görev için de KHD analizi daha başarılı olsa da Görev 2 için sonuçlar anlamlı olmaktan uzaktır. Öznitelik üzerinden yapılan çalışmada ise benzer büyüklükte ağ tasarımları için başarım değerleri yükselmiştir. Her iki görev için de NKHD öznitelikleri ile sınıflandırma daha başarılı sonuçlar üretmiştir.

Çizelge 3.17. KHD ve NKHD ham verileri ile CNN sınıflandırıcı başarımları.

Veri	KHD		NKHD	
Görev	Görev 1	Görev 2	Görev 1	Görev 2
Kesinlik (%)	89,70	53,50	53,01	51,07
Duyarlılık (%)	89,38	94,76	56,46	100,0
F1 Skor (%)	89,54	68,39	54,68	67,61
Doğruluk (%)	89,38	55,44	52,21	51,07
Batch Size	120	32	112	16
Filtre Adet 1	6	6	2	5
Filtre Boyut 1	144	184	16	40
Filtre Adet 2	6	6	---	---
Filtre Boyut 2	72	32	---	---
Filtre Adet 3	3	---	---	---
Filtre Boyut 3	16	---	---	---
Dropout	0,1	0,1	0,2	0,1
Pooling	3	3	3	1
Dense	192	192	200	136

Çizelge 3.18. CNN ile öznitelik seçimi temelli sınıflandırıcı başarımları.

Veri Türü	KHD						NKHD					
Görev	Görev 1			Görev 2			Görev 1			Görev 2		
Normalizasyon	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor	Yok	MinMax	Z-Skor
Kesinlik (%)	86,36	97,87	100,0	52,00	86,96	91,30	75,00	100,0	100,0	53,57	95,83	100,0
Duyarlılık (%)	77,55	93,88	91,84	52,00	80,00	84,00	85,71	97,96	100,0	60,00	92,00	100,0
F1 Skor (%)	81,72	95,83	95,74	52,00	83,33	87,50	80,00	98,97	100,0	56,60	93,88	100,0
Doğruluk (%)	82,65	95,92	95,92	51,02	83,67	87,76	78,57	98,98	100,0	53,06	93,88	100,0
Batch Size	48	8	16	48	48	24	48	24	32	48	48	8
Filtre Adet 1	8	6	2	4	2	2	6	2	7	6	5	2
Filtre Adet 2	---	4	8	8	---	---	3	3	1	8	7	---
Filtre Adet 3	---	5	2	---	---	---	---	---	5	4	3	---
Dropout	0,1	0,4	0,1	0,1	0,2	0,2	0,1	0,1	0,1	0,1	0,3	0,5
Pooling	3	2	1	2	3	2	2	2	2	2	2	1
Dense	248	64	216	96	256	128	216	232	248	168	224	200

4. BULGULAR VE TARTIŞMA

KHD verilerindeki değişimlerin sempatik ve parasempatik sinir sistemi ile ilişkili olduğu önceki çalışmalarda gösterilmiştir [19], [89]. Ancak kişilerin ortalama KHD değeri farklı ve zamanla da değişebilir olmaktadır. Dolayısıyla veriye ait değişimler ve oranlar da kişiden kişiye farklılık göstermektedir. Bu yüzden Hallstorm ve arkadaşları tarafından da önerilen KHD normalizasyonunun, PAF hastaları için teşhis ve atak tahmini amaçlı çalışmalardaki başarımı da artıracağı düşünülmüştür [37]. Bu sebeple iki farklı görev için Pyhsio.net adresinde sunulan AFPDB veritabanı verileri çekilmiştir [35]. Yapılacak çalışmalar için ultra düşük frekans bölgesindeki değişimlerin etkisini de görebilmek adına 30 dakikalık verilerin tamamı ile çalışılmıştır. İlk görev için, sağlıklı olduğu bilinen 49 kişiye ait EKG verileri ile PAF hastası olduğu bilinen 49 hastaya ait EKG verilerinden ilk veritabanı oluşturulmuştur. İkinci görev için ise tamamı PAF hastalığı teşhisi alan kişilerden yakın zamanda PAF atağı geçirmeyeceği bilinen 24 kişiye ait EKG verileri ile kısa süre sonra PAF atağı geçireceği bilinen 25 kişiye ait EKG verilerinin birleşiminden ikinci veritabanı oluşturulmuştur. Veritabanları üzerinden KHD ve NKHD veri setleri çıkarılmıştır. Literatürde önerilen özniteliklerin 48 tanesi ile çalışılmıştır [58]. Bu özniteliklerin MinMax ve Z-Skor normalleştirilmesi yapılmıştır. NKHD verilerinin başarımı artıracağı hipotezini test etmek üzere ilk olarak kNN sınıflandırıcı ile KHD ve NKHD verileri ile iki farklı görev için başarım ölçümleri yapılmış ve NKHD verisi kullanılmasının her iki görev için de başarıma olumlu etkisi olabileceği ispat edilmiştir. Sonrasında farklı sınıflandırıcılar üzerinde NKHD verisini kullanmanın etkilerini incelemek üzere sırasıyla MLP, SVM ve CNN sınıflandırıcılarla çalışılmıştır. Görev 1 için Çizelge 4.1'de görüleceği üzere kNN, MLP ve öznitelik verisini işleyen CNN sınıflandırıcı daha başarılı sonuçlar elde etmişlerdir. SVM doğrusal, SVM-RBF ve ham veri üzerinde çalışan klasik CNN sınıflandırıcı için ise başarım ya değişmemiş ya da düşmüştür. Benzer şekilde Görev 2 için yapılan çalışmaların sonucu Çizelge 4.2'de görülmektedir. Ham veri ile çalışan klasik CNN sınıflandırıcı hariç diğer tüm sınıflandırıcılarda başarım olumlu bir şekilde yükselmiştir.

Çizelge 4.1. Görev 1 için en iyi başarımlar.

Veri Türü	Sınıflandırıcı	Kesinlik (%)	Duyarlılık (%)	F1 Skor (%)	Doğruluk (%)
KHD	kNN	84,00	79,25	81,55	81,00
	MLP1	90,00	93,75	91,84	91,92
	MLP2	91,49	87,76	89,58	89,80
	SVM1-Doğ.	72,00	73,47	72,73	72,45
	SVM1-RBF	84,62	89,80	87,13	86,73
	SVM2-Doğ.	89,36	85,71	87,50	87,76
	SVM2-RBF	100,0	95,92	97,92	97,96
	CNN Ham	89,70	89,38	89,54	89,38
	CNN Özn.	100,0	91,84	95,74	95,92
NKHD	kNN	88,00	84,62	86,27	86,00
	MLP1	96,00	96,00	96,00	95,96
	MLP2	88,46	93,88	91,09	90,82
	SVM1-Doğ.	71,43	81,63	76,19	74,49
	SVM1-RBF	97,73	87,76	92,47	92,86
	SVM2-Doğ.	84,44	77,55	80,85	81,63
	SVM2-RBF	94,23	100	97,03	96,94
	CNN Ham	53,01	56,46	54,68	52,21
	CNN Özn.	100,0	100,0	100,0	100,0

Çizelge 4.2. Görev 2 için en iyi başarımlar.

Veri Türü	Sınıflandırıcı	Kesinlik (%)	Duyarlılık (%)	F1 Skor (%)	Doğruluk (%)
KHD	kNN	80,00	76,92	78,43	78,00
	MLP2	95,83	92,00	93,88	93,88
	SVM1-Doğ.	86,36	76,00	80,85	81,63
	SVM1-RBF	94,74	72,00	81,82	83,67
	SVM2-Doğ.	86,36	76,00	80,85	81,63
	SVM2-RBF	91,67	88,00	89,80	89,80
	CNN Ham	53,50	94,76	68,39	55,44
	CNN Özn.	91,30	84,00	87,50	87,76
NKHD	kNN	80,00	83,33	81,63	82,00
	MLP2	100,0	96,00	97,96	97,96
	SVM1-Doğ.	88,00	88,00	88,00	87,76
	SVM1-RBF	90,00	72,00	80,00	81,63
	SVM2-Doğ.	100,0	88,00	93,62	93,88
	SVM2-RBF	96,15	100,0	98,04	97,96
	CNN Ham	51,07	100,0	67,61	51,07
	CNN Özn.	100,0	100,0	100,0	100,0

Literatürde AFPDB veritabanını kullanan, farklı sınıflandırıcı ve veri işlem basamaklarına sahip pekçok çalışma mevcuttur. Bu çalışmalardan 30 dakikalık veri uzunluğunu kullananlar ile sonuçlarımızı karşılaştırmak adına çalışmalarımıza ait başarımların, KHD ve NKHD verileri için ayrı ayrı olmak üzere kNN, MLP, SVM ve CNN sınıflandırıcılar için en iyileri seçilmiştir. Bu seçimler Görev 1 için Çizelge 4.3'te ve Görev 2 için ise Çizelge 4.4'te gösterilmiştir. Literatür özeti karşılaştırması ise Görev 1 için Çizelge 4.5 ve Görev 2 için ise Çizelge 4.6'da sunulmuştur.

Literatüre Görev 1 için bakacak olursak, Maier ve arkadaşları KHD verileri ve LDA sınıflandırıcı ile %80 başarım elde etmişlerdir [41]. Schreier ve arkadaşları ise Morfolojik EKG öznitelikleri ve FL ile %82 başarıma ulaşmışlardır [42]. Boon ve arkadaşları GA ile öznitelik seçimi yaptıkları SVM sınıflandırıcıda %83,9 başarım sağlamışlardır [53]. Thong ve arkadaşları ise PACs ve DT ile %90 başarım elde etmişlerdir [44]. Pourbabaee ve arkadaşları [98] DL kullanarak %91 başarıma ulaşırken Ros ve arkadaşları ile Özcan ve Kuntalp, çalışmalarında kNN sınıflandırıcı ile %92 ve %92,2 başarım sağlamışlardır [45], [46]. Bu çalışmamızdan önce literatürdeki en başarılı sonuçlara ise MLP sınıflandırıcı ve %95,5 başarımla Chesnokov ve arkadaşları ile SGD sınıflandırıcı ve %96,05 başarımla Martinez ve arkadaşları ulaşmışlardır [47], [48].

Görev 2 için literatür çalışmalarına baktığımızda ise Lynn ve Chiang kNN sınıflandırıcı ile %78 başarım elde etmiştir [50]. Chesnokov bu görevde ise MLP sınıflandırıcı ile %82,05 başarımı yakalamıştır [51]. Boon ve arkadaşları SVM sınıflandırıcı ile farklı çalışmalarda sırasıyla %83,09 ve %86,8 şeklinde başarımlar elde etmiştir [53], [54]. Schreier ve arkadaşları bu görevde de FL kullanarak %84 başarıma ulaşabilmiştir [42]. DT kullanan çalışmalardan Zong ve arkadaşları [55] %88, Thong ve arkadaşları [44] %89,29 ve son olarak Costin ve arkadaşları [56] ise %89,4 başarım yakalamışlardır. Maier ve arkadaşları yine LDA sınıflandırma kullanarak önceki görevin aksine %92 gibi yüksek bir başarıma ulaşmışlardır [41]. Görev 2 için yaptığımız çalışmalardan önceki en yüksek başarıma ise SVM sınıflandırıcı ile Mohebbi ve arkadaşları %94,5 değerle ulaşmışlardır [11].

Bu özetlerden de görüleceği üzere, iki görev için de literatürdeki çalışmalardan daha yüksek başarımlar elde edilmiş; ayrıca çoğu durumda kalp hızı değişkenliğinin normalize edilmesinin PAF hastaları için sınıflandırıcı performansını artırdığı gösterilmiştir.

Çizelge 4.3. Görev 1 için özet ve her sınıflandırıcı için en iyi başarımlar.

Çalışma	Öznitelik	Sınıflandırıcı Türü	Açıklama	Doğruluk (%)
Bu çalışma	NKHD	CNN Ham	-	52,21
Bu çalışma	KHD	SVM Doğrusal	GA	72,45
Bu çalışma	NKHD	SVM Doğrusal	GA, MinMax	74,49
[43] – Bu çalışma	KHD	kNN	GA, MinMax	81,00
Bu çalışma	NKHD	SVM Doğrusal	GA, MinMax, 10-fold	81,63
[43] – Bu çalışma	NKHD	kNN	GA, Z-Skor	86,00
Bu çalışma	KHD	SVM RBF	GA, MinMax	86,73
Bu çalışma	KHD	SVM Doğrusal	10-fold	87,76
Bu çalışma	KHD	CNN Ham	-	89,38
Bu çalışma	KHD	MLP	Z-Skor	89,80
Bu çalışma	NKHD	MLP	Z-Skor	90,82
[99] – Bu çalışma	KHD	MLP	GA, Z-Skor	91,92
Bu çalışma	NKHD	SVM RBF	GA, MinMax	92,86
[57] – Bu çalışma	KHD	CNN	Z-Skor	95,92
[99] – Bu çalışma	NKHD	MLP	GA, Z-Skor	95,96
Bu çalışma	NKHD	SVM RBF	MinMax, 10-fold	96,94
Bu çalışma	KHD	SVM RBF	GA, Z-Skor, 10-fold	97,96
[57] – Bu çalışma	NKHD	CNN	Z-Skor	100,0

Çizelge 4.4. Görev 2 için özet ve her sınıflandırıcı için en iyi başarımlar.

Çalışma	Öznitelik	Sınıflandırıcı Türü	Açıklama	Doğruluk (%)
Bu çalışma	NKHD	CNN Ham	-	51,07
Bu çalışma	KHD	CNN Ham	-	55,44
Bu çalışma	KHD	kNN	GA, Z-Skor	78,00
Bu çalışma	KHD	SVM Doğrusal	Z-Skor, 10-fold	81,63
Bu çalışma	KHD	SVM Doğrusal	GA, Z-Skor	81,63
[52] – Bu çalışma	NKHD	SVM RBF	GA, MinMax	81,63
Bu çalışma	NKHD	kNN	GA, MinMax	82,00
[52] – Bu çalışma	KHD	SVM RBF	GA, MinMax	83,67
[49] – Bu çalışma	KHD	CNN	Z-Skor	87,76
Bu çalışma	NKHD	SVM Doğrusal	GA, Z-Skor	87,76
Bu çalışma	KHD	SVM RBF	MinMax, 10-fold	89,80
Bu çalışma	KHD	MLP	Z-Skor	93,88
Bu çalışma	NKHD	SVM Doğrusal	MinMax, 10-fold	93,88
Bu çalışma	NKHD	MLP	Z-Skor	97,96
Bu çalışma	NKHD	SVM RBF	GA, MinMax, 10-fold	97,96
[49] – Bu çalışma	NKHD	CNN	Z-Skor	100,0

Çizelge 4.5. Görev 1 için bu çalışma kapsamında elde edilmiş en yüksek sınıflandırıcı başarımları ile 30 dk veri kullanan literatürdeki diğer çalışmaların başarımlarını karşılaştırma tablosu.

Çalışma	Öznitelik	Sınıflandırıcı	Açıklama	Doğruluk (%)
[41]	KHD	LDA, polinom LDA	-	80,00
[43] – Bu çalışma	KHD	kNN	GA, MinMax	81,00
[42]	Morfolojik EKG	FL	-	82,00
[53]	KHD	SVM	GA	83,90
[43] – Bu çalışma	NKHD	kNN	GA, Z-Skor	86,00
[44]	PAC	DT	-	90,00
[98]	EKG	CNN ve kNN	-	91,00
[99] – Bu çalışma	KHD	MLP	GA, Z-Skor	91,92
[45]	Morfolojik EKG	kNN	-	92,00
[46]	KHD	kNN	GA	92,20
[47]	KHD	MLP	-	95,50
[57] – Bu çalışma	KHD	CNN	Z-Skor	95,92
[99] – Bu çalışma	NKHD	MLP	GA, Z-Skor	95,96
[48]	Morfolojik EKG	SGD	-	96,05
Bu çalışma	NKHD	SVM RBF	MinMax, 10-fold	96,94
Bu çalışma	KHD	SVM RBF	GA, Z-Skor, 10-fold	97,96
[57] – Bu çalışma	NKHD	CNN	Z-Skor	100,0

Çizelge 4.6. Görev 2 için bu çalışma kapsamında elde edilmiş en yüksek sınıflandırıcı başarımları ile 30 dk veri kullanan literatürdeki diğer çalışmaların başarımlarını karşılaştırma tablosu.

Çalışma	Öznitelik	Sınıflandırıcı	Açıklama	Doğruluk (%)
[50]	KHD	kNN	-	78,00
Bu çalışma	KHD	kNN	GA, Z-Skor	78,00
Bu çalışma	NKHD	kNN	GA, MinMax	82,00
[51]	KHD	MLP	-	82,05
[53]	KHD	SVM	GA	83,90
[42]	EKG	FL	-	84,00
[54]	KHD	SVM	GA	86,80
[49] – Bu çalışma	KHD	CNN	Z-Skor	87,76
[55]	EKG	DT	-	88,00
[44]	EKG	DT	-	89,29
[56]	EKG + KHD	DT	-	89,40
Bu çalışma	KHD	SVM RBF	MinMax, 10-fold	89,80
[41]	KHD	LDA	-	92,00
Bu çalışma	KHD	MLP	Z-Skor	93,88
Bu çalışma	NKHD	SVM Doğrusal	MinMax, 10-fold	93,88
[11]	KHD	SVM	-	94,50
Bu çalışma	NKHD	MLP	Z-Skor	97,96
Bu çalışma	NKHD	SVM RBF	GA, MinMax, 10-fold	97,96
[49] – Bu çalışma	NKHD	CNN	Z-Skor	100,0

5. SONUÇLAR VE ÖNERİLER

Bu tezde, KHD verilerinin normalize edilmesinin PAF hastalığı teşhisi ve atak tahmini süreçlerinde sınıflandırıcı performansı üzerinde etkisi incelenmiştir. Bu amaçla Phsionet.org internet sitesindeki "The Computer in Cardiology Challenge 2001" kapsamında kullanılan "Atrial Fibrillation Prediction Database (AFPDB)" veri seti kullanılmıştır. Veri setinde bulunan EKG verilerinden KHD verileri elde edilmiş, bu verilerden ve bu verilerin normalize edilmesinden (NKHD) öznitelikler çıkartılmıştır. Oluşan öznitelik veri gruplarının da normalize edilmiş ve edilmemiş halleri ile toplam altı farklı öznitelik veri seti oluşturulmuş ve kaydedilmiştir. Oluşan öznitelikler üzerinden Genetik Algoritma ile öznitelik seçimi yapılarak boyutu düşürülmüş öznitelik veri seti elde edilerek kaydedilmiştir. Kaydedilen öznitelik grupları ile kNN, MLP, SVM ve CNN sınıflandırıcılar kullanılarak farklı koşullarda başarımlar test edilmiştir.

kNN sınıflandırıcı basitliğine rağmen ortalama bir başarım elde etmiştir. Veri setinin normalize edilmesi kNN sınıflandırıcıda her koşulda başarımı artırmıştır. Öznitelik normalizasyonu ise genel olarak her şartta başarımda iyileşme sağlamıştır. Seçilen basit MLP ağı, basitliğine karşın kNN'den daha başarılı sonuçlar elde etmeyi sağlamıştır. Burada da NKHD verisinin kullanılması başarımı artırmıştır. SVM sınıflandırıcıda ise beklendiği gibi RBF çekirdek fonksiyon kullanılması sınıflandırmada daha iyi sonuçlar elde edilmesini sağlamıştır. Literatürdeki tüm çalışmalardan daha iyi sonuçlar alınmasına karşın, Görev 1 için KHD verisinin kullanılması daha iyi sonuçlar alınmasına sebep olmuştur. Ama Görev 2 için yine NKHD verisinin kullanımı başarımı artırmıştır. Derin öğrenmede ise öznitelik girişi yapılan CNN sınıflandırıcı NKHD verileri için eğitim verisi bölümlemesi ve Seyreltme (Dropout) katmanı kullanılmasına rağmen %100 başarıma ulaşmıştır. Burada %100 başarıma ulaşmanın önemli bir etkeni de CNN gibi çabuk öğrenen bir ağ için çok küçük sayılabilecek veri seti girişine sahip olunmasıdır. Daha geniş bir veri örneği için başarımın az da olsa azalacağı düşünülmektedir.

Yapılan çalışma sonucunda kalp hızı değişkenliği verilerinin normalizasyon işlemine tutulmasının, bazı istisnalar hariç başarımı artırdığı ispatlanmıştır. NKHD verisi kullanmak her koşulda başarım artışı sağlamadığı için, benzer veri gruplarında yapılacak farklı çalışmalarda NKHD verilerinin yanında KHD verilerinin de incelenmesinin daha doğru olacağı düşünülmektedir.

Yapılan çalışmalar ile literatürde aynı veri uzunluğunu inceleyen ve aynı görev için sınıflandırma yapan tüm çalışmalardan daha iyi sonuçlar elde edilmiş ve sonuçlardan bazıları uluslararası dergilerde yayınlanarak literatüre katkı sağlanmıştır.

Bundan sonra yapılacak çalışmalarda daha fazla sayıda bireyden alınmış örnek veri seti üzerinde çalışılmasının, daha anlamlı sonuçlar elde edilmesini ve sınıflandırıcıların daha kapsayıcı olmasını sağlayacağı düşünülmektedir.

6. KAYNAKLAR

[1] A. Pappano ve W. Wier, *Cardiovascular Physiology*, 11. baskı, USA: Elsevier, 2019, böl. 1, ss. 1-9.

[2] J. G. Webster, *Medical Instrumentation: Application and Design*. 4. baskı, USA: Wiley, 2020, böl. 1, ss. 1-44.

[3] E. Wilkins, L. Wilson, K. Wickramasinghe ve P. Bhatnagar, "European Cardiovascular Disease Statistics 2017", Brussels, Belgium:European Heart Network, 2017, ss. 1-186.

[4] K. M. Ryder ve E. J. Benjamin, "Epidemiology and significance of atrial fibrillation", *The American Journal of Cardiology*, c. 84, sayı 9, ss. 131–138, 1999.

[5] C. Steger A. Pratter, M. Martinekbregel, M. Avanzini, A. Valentin, J. Slany ve C. Stollberger, "Stroke patients with atrial fibrillation have a worse prognosis than patients without: data from the Austrian Stroke registry", *European Heart Journal*, c. 25, sayı 19, ss. 1734–1740, 2004.

[6] E. J. Benjamin, "Independent Risk Factors for Atrial Fibrillation in a Population-Based Cohort", *JAMA*, c. 271, sayı 11, ss. 840, 1994.

[7] S. F. Parsa, A. Vafajoo, A. Rostami, R. Salarian, M. Rabiee, N. Rabiee, G. Rabiee, M. Tahriri, A. Yadegari, D. Vashaee, L. Tayebi ve M. R. Hamblin, "Early diagnosis of disease using microbead array technology: A review", *Analytica Chimica Acta*, c. 1032, ss. 1–17, 2018.

[8] P. S. Moran , C. Teljeur, P. Harrington, S. M. Smith, B. Smyth, J. Harbison, C. Normand ve M. Ryan, "Cost-Effectiveness of a National Opportunistic Screening Program for Atrial Fibrillation in Ireland", *Value in Health*, c. 19, sayı 8, ss. 985–995, 2016.

[9] B. Freedman, "Screening for atrial fibrillation", *Circulation*, c. 135, sayı 19, ss. 1851–1867, 2017.

[10] E. Ebrahimzadeh, M. Kalantari, M. Joulani, R. S. Shahraki, F. Fayaz ve F. Ahmadi, "Prediction of paroxysmal Atrial Fibrillation: A machine learning based approach using combined feature vector and mixture of expert classification on HRV signal", *Computer Methods and Programs in Biomedicine*, c. 165, ss. 53–67, 2018.

[11] M. Mohebbi ve H. Ghassemian, "Prediction of paroxysmal atrial fibrillation based on non-linear analysis and spectrum and bispectrum features of the heart rate variability signal", *Computer Methods and Programs in Biomedicine*, c. 105, sayı 1, ss. 40–49, 2012.

[12] D. J. Gladstone, M. Spring, P. Dorian, V. Panzov, K. E. Thorpe, J. Hall, H. Vaid, M. O'Donnell, A. Laupacis, R. Côté, M. Sharma, J. A. Blakely, A. Shuaib, V. Hachinski, S. B. Coutts, D. J. Sahlas, P. Teal, S. Yip, J. D. Spence, B. Buck, S. Verreault, L. K. Casaubon, A. Penn, D. Selchen, A. Jin, D. Howse, M. Mehdiratta, K. Boyle, R. Aviv, M. K. Kapral ve M. Mamdani, "Atrial Fibrillation in Patients with Cryptogenic Stroke", *New England Journal of Medicine*, c. 370, sayı 26, ss.

2467–2477, 2014.

[13] A. Narin, Y. Isler, M. Ozer ve M. Perc, "Early prediction of paroxysmal atrial fibrillation based on short-term heart rate variability", *Physica A: Statistical Mechanics and its Applications*, c. 509, ss. 56–65, 2018.

[14] A. Prakash, S. Saksena, M. Hill, PhD, R. B. Krol, A. N. Munsif, I. Giorgberidze, P. Mathew ve R. Mehra, "Acute Effects of Dual-Site Right Atrial Pacing in Patients With Spontaneous and Inducible Atrial Flutter and Fibrillation", *Journal of the American College of Cardiology*, c. 29, sayı 5, ss. 1007–1014, 1997.

[15] A. Soudani ve M. Almusallam, "Atrial Fibrillation detection based on ECG-Features Extraction in WBSN", *Procedia Computer Science*, c. 130, ss. 472–479, 2018.

[16] J. L. Salinet, J. P. V. Madeiro, P. C. Cortez, P. J. Stafford, G. André Ng ve F. S. Schlindwein, "Analysis of QRS-T subtraction in unipolar atrial fibrillation electrograms", *Medical & Biological Engineering & Computing*, c. 51, sayı 12, ss. 1381–1391, 2013.

[17] U. R. Acharya, H. Fujita, O. S. Lih, M. Adam, J. H. Tan ve C. K. Chua, "Automated detection of coronary artery disease using different durations of ECG segments with convolutional neural network", *Knowledge-Based Systems*, c. 132, ss. 62–71, 2017.

[18] H. Guruler, M. Sahin ve A. Ferikoglu, "Feature selection on single-lead ECG for obstructive sleep apnea diagnosis", *Turkish Journal of Electrical Engineering & Computer Sciences*, c. 22, sayı 2, ss. 465–478, 2014.

[19] Task Force of the European Society of Cardiology the North American Society of Pacing Electrophysiology, "Heart Rate Variability", *Circulation*, c. 93, sayı 5, ss. 1043–1065, 1996.

[20] M. Chessa, G. Butera, G. A. Lanza, E. Bossone, A. Delogu, G. De Rosa, G. Marietti, L. Rosti ve M. Carminati, "Role of Heart Rate Variability in the Early Diagnosis of Diabetic Autonomic Neuropathy in Children", *Herz*, c. 27, sayı 8, ss. 785–790, 2002.

[21] S. Ahmad, A. Tejuja, K. D. Newman, R. Zarychanski ve A. J. Seely, "Clinical review: A review and analysis of heart rate variability and the diagnosis and prognosis of infection", *Critical Care*, c. 13, sayı 6, s. 232, 2009.

[22] F. Sessa, V. Anna, G. Messina, G. Cibelli, V. Monda, G. Marsala, M. Ruberto, A. Biondi, O. Cascio, G. Bertozzi, D. Pisanelli, F. Maglietta, A. Messina, M. P. Mollica ve M. Salerno, "Heart rate variability as predictive factor for sudden cardiac death", *Aging*, c. 10, sayı 2, ss. 166–177, 2018.

[23] L. Verde ve G. De Pietro, "A neural network approach to classify carotid disorders from Heart Rate Variability analysis", *Computers in Biology and Medicine*, c. 109, ss. 226–234, 2019.

[24] I. Goldenberg, R. Goldkorn, N. Shlomo, M. Einhorn, J. Levitan, R. Kuperstein, R. Klempfner ve B. Johnson, "Heart Rate Variability for Risk Assessment of Myocardial Ischemia in Patients Without Known Coronary Artery Disease: The HRV-DETECT (Heart Rate Variability for the Detection of Myocardial Ischemia) Study", *Journal of the American Heart Association*, c. 8, sayı 24, 2019.

[25] J.-Q. Ke, S.-M. Shao, Y.-Y. Zheng, F.-W. Fu, G.-Q. Zheng ve C.-F. Liu,

"Sympathetic skin response and heart rate variability in predicting autonomic disorders in patients with Parkinson disease", *Medicine*, c. 96, sayı 18, ss. 1-5, 2017.

[26] Y. Isler, A. Narin, M. Ozer ve M. Perc, "Multi-stage classification of congestive heart failure based on short-term heart rate variability", *Chaos, Solitons & Fractals*, c. 118, ss. 145–151, 2019.

[27] B. M. Asl, S. K. Setarehdan ve M. Mohebbi, "Support vector machine-based arrhythmia classification using reduced features of heart rate variability signal", *Artificial Intelligence in Medicine*, c. 44, sayı 1, ss. 51–64, 2008.

[28] M. G. Tsipouras ve D. I. Fotiadis, "Automatic arrhythmia detection based on time and time–frequency analysis of heart rate variability", *Computer Methods and Programs in Biomedicine*, c. 74, sayı 2, ss. 95–108, 2004.

[29] M. Rouhani ve R. Soleymani, "Neural Networks Based Diagnosis of Heart Arrhythmias Using Chaotic and Nonlinear Features of HRV Signals", *2009 International Association of Computer Science and Information Technology*, Singapur, 2009, ss. 545–549.

[30] G. Y. H. Lip, "Paroxysmal atrial fibrillation", *QJM: An International Journal of Medicine*, c. 94, sayı 12, ss. 665–678, 2001.

[31] Ming-Yuan Lee ve Sung-Nien Yu, "Multiscale sample entropy based on discrete wavelet transform for clinical heart rate variability recognition", *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, San Diego, USA, 2012, ss. 4299–4302.

[32] D. Duverney, J.-M. Gaspoz, V. Pichot, F. Roche, R. Brion, A. Antoniadis ve J.-C. Barthelemy, "High Accuracy of Automatic Detection of Atrial Fibrillation Using Wavelet Transform of Heart Rate Intervals", *Pacing and Clinical Electrophysiology*, c. 25, sayı 4, ss. 457–462, 2002.

[33] J. Park, S. Lee ve M. Jeon, "Atrial fibrillation detection by heart rate variability in Poincare plot", *BioMedical Engineering OnLine*, c. 8, sayı 1, s. 38, 2009.

[34] J. P. Sepulveda-Suescun, J. Murillo-Escobar, R. D. Urda-Benitez, D. A. Orrego-Metaute ve A. Orozco-Duque, "Atrial fibrillation detection through heart rate variability using a machine learning approach and Poincare plot features", *IFMBE Proceedings*, Singapur, 2017, ss. 565–568.

[35] G. Moody, A. Goldberger, S. McClennen ve S. Swiryn, "Predicting the onset of paroxysmal atrial fibrillation: the Computers in Cardiology Challenge 2001", *Computers in Cardiology 2001*, c. 28, Rotterdam, Hollanda, 2001, ss. 113–116.

[36] R. Seker, S. Saliu, A. Birand ve G. Kudaiberdieva, "Validity test for a set of nonlinear measures for short data length with reference to short-term heart rate variability signal", *Journal of Systems Integration*, c. 10, sayı 1, ss. 41–53, 2000.

[37] A. P. Hallstrom, P. K. Stein, R. Schneider, M. Hodges, G. Schmidt ve K. Ulm, "Structural Relationships Between Measures Based on Heart Beat Intervals: Potential for Improved Risk Assessment", *IEEE Transactions on Biomedical Engineering*, c. 51, sayı 8, ss. 1414–1420, 2004.

[38] Y. Isler ve M. Kuntalp, "Heart rate normalization in the analysis of heart rate variability in congestive heart failure", *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, c. 224, sayı

3, ss. 453–463, 2010.

[39] Y. Isler ve M. Kuntalp, "Combining classical HRV indices with wavelet entropy measures improves to performance in diagnosing congestive heart failure", *Computers in Biology and Medicine*, c. 37, sayı 10, ss. 1502–1510, 2007.

[40] Y. Isler, "Discrimination of systolic and diastolic dysfunctions using multi-layer perceptron in heart rate variability analysis", *Computers in Biology and Medicine*, c. 76, ss. 113–119, 2016.

[41] C. Maier, M. Bauch ve H. Dickhaus, "Screening and prediction of paroxysmal atrial fibrillation by analysis of heart rate variability parameters", *Computers in Cardiology 2001*, c. 28, Rotterdam, Hollanda, 2001, ss. 129–132.

[42] G. Schreier, P. Kastner ve W. Marko, "An automatic ECG processing algorithm to identify patients prone to paroxysmal atrial fibrillation", *Computers in Cardiology 2001*, c. 28, Rotterdam, Hollanda, 2001, ss. 133–135.

[43] M. Surucu, Y. Isler ve R. Kara, "Investigation of the Effect of Normalization Techniques on Discriminating Patients with Paroxysmal Atrial Fibrillation", *2nd International Conference of Applied Sciences, Engineering and Mathematics*, Skopje, Makedonya, 2020, s. 12.

[44] T. Thong, J. McNames, M. Aboy ve B. Goldstein, "Prediction of Paroxysmal Atrial Fibrillation by Analysis of Atrial Premature Complexes", *IEEE Transactions on Biomedical Engineering*, c. 51, sayı 4, ss. 561–569, 2004.

[45] E. Ros, S. Mota, F. J. Fernández, F. J. Toro ve J. L. Bernier, "ECG Characterization of paroxysmal atrial fibrillation: parameter extraction and automatic diagnosis algorithm", *Computers in Biology and Medicine*, c. 34, sayı 8, ss. 679–696, 2004.

[46] N. Ozcan ve M. Kuntalp, "Determining best HRV indices for PAF screening using genetic algorithm", *10th International Conference on Electrical and Electronics Engineering (ELECO)*, Bursa, Türkiye, 2018, ss. 1385-1388..

[47] Y. V. Chesnokov, A. V. Holden ve H. Zhang, "Screening patients with paroxysmal atrial fibrillation (PAF) from non-PAF heart rhythm using HRV data analysis", *Computers in Cardiology*, Kuzey Carolina, USA, 2007, c. 34, ss. 459–462.

[48] A. Martínez, R. Alcaraz ve J. J. Rieta, "Study on the P-wave feature time course as early predictors of paroxysmal atrial fibrillation", *Physiological Measurement*, c. 33, sayı 12, ss. 1959–1974, 2012.

[49] M. Surucu, Y. Isler ve R. Kara, "Diagnosis of Paroxysmal Atrial Fibrillation from Thirty-Minute Heart Rate Variability Data using Convolutional Neural Networks", *Turkish Journal of Electrical Engineering & Computer Sciences*, 2021.

[50] K. S. Lynn ve H. D. Chiang, "A two-stage solution algorithm for paroxysmal atrial fibrillation prediction", *Computers in Cardiology 2001*, c. 28, Rotterdam, Hollanda, 2001, ss. 405–407.

[51] Y. V. Chesnokov, "Complexity and spectral analysis of the heart rate variability dynamics for distant prediction of paroxysmal atrial fibrillation with artificial intelligence methods", *Artificial Intelligence in Medicine*, c. 43, sayı 2, ss. 151–165, 2008.

[52] M. Surucu, Y. Isler ve R. Kara, "Heart Rate Normalization on Determining a Paroxysmal Atrial Fibrillation Attack", *3rd International Conference of Applied*

Sciences, Engineering and Mathematics, Skopje, Makedonya, 2021, s. 19.

[53] K. H. Boon, M. Khalil-Hani, M. B. Malarvili ve C. W. Sia, "Paroxysmal atrial fibrillation prediction method with shorter HRV sequences", *Computer Methods and Programs in Biomedicine*, c. 134, ss. 187–196, 2016.

[54] K. H. Boon, M. Khalil-Hani ve C. W. Sia, "Paroxysmal Atrial Fibrillation Onset Prediction Using Heart Rate Variability Analysis and Genetic Algorithm for Optimization", *Lecture Notes in Electrical Engineering*, Singapur: Springer, 2019, ss. 609–617.

[55] W. Zong, R. Mukkamala ve R. G. Mark, "A methodology for predicting paroxysmal atrial fibrillation based on ECG arrhythmia feature analysis", *Computers in Cardiology 2001*, c. 28, Rotterdam, Hollanda, 2001, ss. 125–128.

[56] H. Costin, C. Rotariu ve A. Pasarica, "Atrial fibrillation onset prediction using variability of ECG signals", *2013 8th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, Bukreş, Romanya, 2013, ss. 1–4.

[57] M. Surucu, Y. Isler, M. Perc ve R. Kara, "Convolutional Neural Networks Predict the Onset of Paroxysmal Atrial Fibrillation: Theory and Applications", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, Basımda.

[58] F. Shaffer ve J. P. Ginsberg, "An Overview of Heart Rate Variability Metrics and Norms", *Frontiers in Public Health*, c. 5, s. 258, 2017.

[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever ve R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, c. 15, sayı 56, ss. 1929–1958, 2014.

[60] J. E. Hall ve M. E. Hall, *Guyton and Hall: Textbook of Medical Physiology*, 14. baskı, Kanada: Elsevier, 2020, ss. 1-140.

[61] E. N. Prystowsky, G. J. Klein ve J. P. Daubert, *Cardiac arrhythmias : interpretation, diagnosis, and treatment*, 2. baskı,USA: McGraw-Hill Education, 2020.

[62] C. T. January, L. S. Wann, J. S. Alpert, H. Calkins, J. E. Cigarroa, J. C. Cleveland, J. B. Conti, P. T. Ellinor, M. D. Ezekowitz, M. E. Field, K. T. Murray, R. L. Sacco, W. G. Stevenson, P. J. Tchou, C. M. Tracy ve C. W. Yancy, "2014 AHA/ACC/HRS Guideline for the Management of Patients With Atrial Fibrillation: Executive Summary", *Circulation*, c. 130, sayı 23, ss. 2071–2104, 2014.

[63] J. Saul, "Beat-To-Beat Variations of Heart Rate Reflect Modulation of Cardiac Autonomic Outflow", *Physiology*, c. 5, sayı 1, ss. 32–37, 1990.

[64] E. H. Hon ve S. T. Lee, "Electronic Evaluation of The Fetal Heart Rate. VIII. Patterns Preceding Fetal Death, Further Observations", *American journal of obstetrics and gynecology*, c. 87, 1963.

[65] S. Akselrod, D. Gordon, F. Ubel, D. Shannon, A. Berger ve R. Cohen, "Power spectrum analysis of heart rate fluctuation: a quantitative probe of beat-to-beat cardiovascular control", *Science*, c. 213, sayı 4504, ss. 220–222, 1981.

[66] M. A. Peltola, "Role of editing of R–R intervals in the analysis of heart rate variability", *Frontiers in Physiology*, c. 3, 2012.

[67] D. G. Jang, M. Hahn, J. K. Jang, U. Farooq ve S. H. Park, "A comparison of

interpolation techniques for RR interval fitting in AR spectrum estimation", *2012 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Hsinchu, Tayvan, 2012, ss. 352-355.

[68] G. G. Berntson, J. Thomas Bigger, D. L. Eckberg, P. Grossman, P. G. Kaufmann, M. Malik, H. N. Nagaraja, S. W. Porges, J. P. Saul, P. H. Stone ve M. W. Van Der Molen , "Heart rate variability: Origins, methods, and interpretive caveats", *Psychophysiology*, c. 34, sayı 6, ss. 623–648, 1997.

[69] W. Gersch ve G. Kitagawa, *Smoothness Priors in Time Series*, New York, USA:Springer, 1987.

[70] M. P. Tarvainen, P. O. Ranta-aho ve P. A. Karjalainen, "An advanced detrending method with application to HRV analysis", *IEEE Transactions on Biomedical Engineering*, c. 49, sayı 2, ss. 172–175, 2002.

[71] S. Akdemir Akar, S. Kara, F. Latifoğlu ve V. Bilgiç, "Spectral analysis of photoplethysmographic signals: The importance of preprocessing", *Biomedical Signal Processing and Control*, c. 8, sayı 1, ss. 16–22, 2013.

[72] P. Langley, D. di Bernardo, J. Allen, E. Bowers, F. E. Smith, S. Vecchietti ve A. Murray, "Can paroxysmal atrial fibrillation be predicted?", *Computers in Cardiology 2001*, c. 28, Rotterdam, Hollanda, 2001, ss. 121–124.

[73] N. R. Lomb, "Least-squares frequency analysis of unequally spaced data", *Astrophysics and Space Science*, c. 39, sayı 2, ss. 447–462, 1976.

[74] J. D. Scargle, "Studies in astronomical time series analysis. II - Statistical aspects of spectral analysis of unevenly spaced data", *The Astrophysical Journal*, c. 263, s. 835, 1982.

[75] J. T. VanderPlas, "Understanding the Lomb–Scargle Periodogram", *The Astrophysical Journal Supplement Series*, c. 236, sayı 1, s. 16, 2018.

[76] M. Heideman, D. Johnson ve C. Burrus, "Gauss and the history of the fast fourier transform", *IEEE ASSP Magazine*, c. 1, sayı 4, ss. 14–21, 1984.

[77] J. W. Cooley ve J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", *Mathematics of Computation*, c. 19, sayı 90, ss. 297–297, 1965.

[78] P. Duhamel ve H. Hollmann, "'Split radix' FFT algorithm", *Electronics Letters*, c. 20, sayı 1, s. 14, 1984.

[79] P. Welch, "The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms", *IEEE Transactions on Audio and Electroacoustics*, c. 15, sayı 2, ss. 70–73, 1967.

[80] C. Chiann ve P. A. Morettin, "A wavelet analysis for time series", *Journal of Nonparametric Statistics*, c. 10, sayı 1, ss. 1–46, 1998.

[81] G. Lindfield ve J. Penny, *Numerical Methods Using MATLAB*, 4. baskı, USA:Elsevier, 2019, ss. 383-431.

[82] A. Haar, "Zur Theorie der orthogonalen Funktionensysteme", *Mathematische Annalen*, c. 69, sayı 3, ss. 331–371, 1910.

[83] I. Daubechies, "Orthonormal bases of compactly supported wavelets", *Communications on Pure and Applied Mathematics*, c. 41, sayı 7, ss. 909–996,

1988.

[84] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, c. 11, sayı 7, ss. 674–693, 1989.

[85] Z.-L. Gaing, "Wavelet-Based Neural Network for Power Disturbance Recognition and Classification", *IEEE Transactions on Power Delivery*, c. 19, sayı 4, ss. 1560–1568, 2004.

[86] R. M. Gray, *Entropy and Information Theory*, 2. baskı, USA:Springer, 2011, ss. 61-96.

[87] C. E. Shannon, "A Mathematical Theory of Communication", *Bell System Technical Journal*, c. 27, sayı 3, ss. 379–423, 1948.

[88] P. W. Kamen ve A. M. Tonkin, "Application of the Poincaré plot to heart rate variability: a new measure of functional status in heart failure", *Australian and New Zealand Journal of Medicine*, c. 25, sayı 1, ss. 18–26, 1995.

[89] M. Brennan, M. Palaniswami ve P. Kamen, "Do existing measures of Poincare plot geometry reflect nonlinear features of heart rate variability?", *IEEE Transactions on Biomedical Engineering*, c. 48, sayı 11, ss. 1342–1347, 2001.

[90] L. García Maset, L. B. González, G. L. Furquet, F. M. Suay ve R. H. Marco, "Study of Glycemic Variability Through Time Series Analyses (Detrended Fluctuation Analysis and Poincaré Plot) in Children and Adolescents with Type 1 Diabetes", *Diabetes Technology & Therapeutics*, c. 18, sayı 11, ss. 719–724, 2016.

[91] S. Marsland, *Machine Learning: An Algorithmic Perspective*, 2. baskı, Londra, İngiltere: Chapman and Hall/CRC, 2014, ss. 39-108.

[92] C. Cortes ve V. Vapnik, "Support-vector networks", *Machine Learning*, c. 20, sayı 3, ss. 273–297, 1995.

[93] C. Yang, B. Hou, B. Ren, Y. Hu ve L. Jiao, "CNN-Based Polarimetric Decomposition Feature Selection for PolSAR Image Classification", *IEEE Transactions on Geoscience and Remote Sensing*, c. 57, sayı 11, ss. 8796–8812, 2019.

[94] H. Habibi Aghdam ve E. Jahani Heravi, *Guide to Convolutional Neural Networks*, Switzerland: Springer, 2017, ss. 85-127

[95] R. Champseix. (2021, 4 Temmuz). *Aura-healthcare/hrv-analysis: Package for Heart Rate Variability analysis in Python* [Online] Erişim: https://github.com/Aura-healthcare/hrv-analysis.

[96] D. Kopczyk. (2021, 29 Ağustos). *GitHub - dawidkopczyk/genetic: Genetic algorithm used in feature selection* [Online]. Erişim: https://github.com/dawidkopczyk/genetic.

[97] A. F. Gad. (2021, 1 Haziran). *PyGAD: An Intuitive Genetic Algorithm Python Library* [Online]. Erişim: https://pygad.readthedocs.io/en/latest/.

[98] B. Pourbabaee, M. J. Roshtkhari ve K. Khorasani, "Deep Convolutional Neural Networks and Learning ECG Features for Screening Paroxysmal Atrial Fibrillation Patients", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, c. 48, sayı 12, ss. 2095–2104, 2018.

[99] M. Surucu, Y. Isler ve R. Kara, “Effect of heart rate normalization on 30-minute heart rate variability data used in the diagnosis of paroxysmal atrial fibrillation”, *Journal of the Faculty of Engineering and Architecture of Gazi University*, yayımlanmak üzere gönderildi, 2021.

7. EKLER

7.1. EK 1: TEZDE KULLANILAN KOD PARÇACIKLARI

7.1.1. Ektopik Vuru Kaldirmada Kullanilan Python Kod Parçacığı

```
def ektopik_vuru_kaldir(rr_aralik):
    rr_t = np.cumsum(rr_aralik)
    ektopikler = []
    atrial_vuru = 0
    ventricular_vuru = 0
    supheli_vuru = False
    rr_t_new = [rr_t[0]]
    avPencere = min(10, len(rr_aralik))
    avPencereHalf = int(avPencere / 2)
    for i, rr in enumerate(rr_aralik[1:]):
        if i < avPencereHalf:
            rrav = sum(rr_aralik[:avPencere]) / avPencere
        elif i > len(rr_aralik) - (avPencereHalf):
            rrav = sum(rr_aralik[-avPencere:]) / avPencere
        else:
            rrav = sum(rr_aralik[i - avPencereHalf:i + avPencereHalf]) / (avPencere)
        if supheli_vuru:
            supheli_vuru = False
            if abs(rr - rrav) > 0.1 * rrav:
                if (rr - rrav) / (rrav - rr_aralik[i]) > 0.3:
                    rr_t_new[-1] = np.nan
```

```
                rr_t_new.append(np.nan)
                ventricular_vuru += 1
                ektopikler.append(i + 1)
            elif (rr + rr_aralik[i] < rrav * 1.1) & (rr + rr_aralik[i] > rrav * 0.9):
                rr_t_new.append(rr_t[i + 1])
            else:
                atrial_vuru += 1
                rr_t_new[-1] = np.nan
                rr_t_new.append(np.nan)
                ektopikler.append(i + 1)
        else:
            rr_t_new[-1] = np.nan
            rr_t_new.append(np.nan)
            atrial_vuru += 1
            ektopikler.append(i + 1)
        continue
    if rr < 0.8 * rrav:
        supheli_vuru = True
    rr_t_new.append(rr_t[i + 1])
rr_t_new = np.array((pd.Series(rr_t_new).interpolate("linear")).values.tolist())
rr_t_shifted = np.insert(np.delete(rr_t_new, -1), 0, 0)
rr_t_new = rr_t_new - rr_t_shifted
return np.array(rr_t_new).flatten(), np.array(ektopikler).flatten(), atrial_vuru,
ventricular_vuru
```

7.1.2. Eğilim Yok Etmede Kullanilan Python Kod Parçacığı

```
def egilim_yoket(z):
    zmean=z.mean()
    l=10
    T=len(z)
    I=np.identity(T)
    ex = np.ones(T-2)
    data = np.array([ex, -2 * ex, ex])
    diag=np.array([0,1,2])
    D2=sparse.spdiags(data,diag,T-2,T).toarray()
    D2_t=np.transpose(D2)
    z_stat=(I-np.linalg.inv(I + l*l * D2_t.dot(D2)))
    z_stat=np.matmul(z_stat,z)
    return z_stat+zmean
```

7.1.3. İnterpolasyon İşleminde Kullanilan Python Kod Parçacığı

```
def inter(rr, fs=4):
    t = np.cumsum(rr)
    f_interpol = interp1d(t, rr, 'cubic')
    t_interpol = np.arange(t[0], t[-1], 1. / fs)
    y = f_interpol(t_interpol)
    return y,t_interpol
```

7.1.4. Zaman bölgesi öznitelikleri çıkarmada kullanılan Python Kod Parçacığı

```
def time_features(rr):
    nn = rr*1000
    diff_nn = abs(np.diff(nn))
    len_nn = len(nn)
    mnn = np.mean(nn)
    mednn = np.median(nn)
```

```
rangenn = max(nn) - min(nn)
sdsd = np.std(diff_nn)
rmssd = np.sqrt(np.mean(diff_nn ** 2))
nn50 = int(sum(np.abs(diff_nn) > 50))
pnn50 = 100 * nn50 / len_nn
nn20 = int(sum(np.abs(diff_nn) > 20))
pnn20 = 100 * nn20 / len_nn
cvsd = rmssd / mnn
sdnn = np.std(nn, ddof=1)
cvnn = sdnn / mnn
hrs = np.divide(60000, nn)
mhr = np.mean(hrs)
minhr = min(hrs)
maxhr = max(hrs)
stdhr = np.std(hrs)
tdf = {
    'mnn': mnn,
    'sdnn': sdnn,
    'rmssd': rmssd,
    'sdsd': sdsd,
    'nn50': nn50,
    'pnn50': pnn50,
    'nn20': nn20,
    'pnn20': pnn20,
    'mednn': mednn,
    'rangenn': rangenn,
```

```
        'cvsd': cvsd,
        'cvnn': cvnn,
        'mhr': mhr,
        "maxhr": maxhr,
        "minhr": minhr,
        "stdhr": stdhr,
    }
    return tdf
```

7.1.5. Welch metodu için kullanılan Python Kod Parçacığı

```
def rr_welch(rr, fs=4):
    t = np.cumsum(rr)
    t -= t[0]
    f_interpol = interp1d(t, rr, 'cubic')
    t_interpol = np.arange(t[0], t[-1], 1. / fs)
    y = f_interpol(t_interpol)
    y = y - np.mean(y)
    frq, psd = welch(y, fs=fs, window='hanning', nperseg=3000, nfft=2 ** 16,
scaling='density')
    return frq, psd
```

7.1.6. FFT dönüşümü için kullanılan Python Kod Parçacığı

```
def rr_fft(rr, fs=4):
    t = np.cumsum(rr)
    t -= t[0]
    f_interpol = interp1d(t, rr, 'cubic')
    t_interpol = np.arange(t[0], t[-1], 1. / fs)
    y = f_interpol(t_interpol)
    y = y - np.mean(y)
    N = len(t_interpol)
    T = 1. / fs
```

```
    w = windows.blackman(N)
    ywf = fft.fft(y * w)
    xf = fft.fftfreq(N, T)[:N // 2]
    return xf[1:N // 2], 2.0 / N * np.abs(ywf[1:N // 2])
```

7.1.7. LS dönüşümü için kullanılan Python Kod Parçacığı

```
def rr_sig_ls(rr):
    fmax=0.4
    frq = np.linspace(0.00000001, fmax, 10000)
    t = np.cumsum(rr)
    t -= t[0]
    y = rr
    y = y - np.mean(y)
    a_frq = 2 * np.pi * frq
    psd = np.asarray(signal.lombscargle(t, y, a_frq))
    psd *= 2 / (len(t) * (y).std() ** 2)
    return frq, psd
```

7.1.8. Güç dağılımından güç hesaplamak için kullanılan Python Kod Parçacığı

```
def powers(frq, psd):
    lo = [0, 0.003, 0.04, 0.15]
    hi = [0.003, 0.04, 0.15, 0.4]
    pr = np.zeros(6)
    nbands = 4
    df = (frq[1] - frq[0])
    for index in range(0, len(frq)):
        pwr = np.power(psd[index], 1) * df
        for n in range(0, nbands):
            if (frq[index] >= lo[n]) and frq[index] <= hi[n]:
                pr[n] += np.abs(pwr)
                break
```

```
    pr[4] = pr[0] + pr[1] + pr[2] + pr[3]
    pr[5] = pr[2] / pr[3]
    return pr
```

7.1.9. Shannon entropi değeri için kullanılan Python Kod Parçacığı

```
def Shannon (data, lendata):
    S = 0
    if lendata == 1:
        S = data
    else:
        E = data ** 2 / lendata
        P = E / sum(E)
        for p in P:
            if p > 0: S += -(p * log(p, 2))
    return S
```

7.1.10. DWT güç ve entropi hesabı için kullanılan Python Kod Parçacığı

```
def rr_dwt(rr, fs=4):
    t = np.cumsum(rr)
    t -= t[0]
    f_interpol = interp1d(t, rr, 'cubic')
    t_interpol = np.arange(t[0], t[-1], 1. / fs)
    y = f_interpol(t_interpol)
    y = y - np.mean(y)
    ULF = []
    VLF = []
    LF = []
    HF = []
    wt_type = 'db8'
    wt_mode = 'periodization'
    next_power_of_two = int(np.floor(np.log2(len(y))) + 1)
```

```
    y2 = pywt.pad(y, 2 ** next_power_of_two, wt_mode)
    ULF.append(np.asarray(pywt.downcoef('a', y2, wt_type, level=9, mode=wt_mode)))
    VLF.append(np.asarray(pywt.downcoef('d', y2, wt_type, level=9, mode=wt_mode)))
    VLF.append(np.asarray(pywt.downcoef('d', y2, wt_type, level=8, mode=wt_mode)))
    VLF.append(np.asarray(pywt.downcoef('d', y2, wt_type, level=7, mode=wt_mode)))
    tmpa = np.asarray(pywt.downcoef('d', y2, wt_type, level=6, mode=wt_mode))
    VLF.append(np.asarray(pywt.downcoef('a', tmpa, wt_type, level=2,
mode=wt_mode)))
    LF.append(np.asarray(pywt.downcoef('d', tmpa, wt_type, level=2, mode=wt_mode)))
    LF.append(np.asarray(pywt.downcoef('d', tmpa, wt_type, level=1, mode=wt_mode)))
    LF.append(np.asarray(pywt.downcoef('d', y2, wt_type, level=5, mode=wt_mode)))
    tmpa = np.asarray(pywt.downcoef('d', y2, wt_type, level=4, mode=wt_mode))
    LF.append(np.asarray(pywt.downcoef('a', tmpa, wt_type, level=3, mode=wt_mode)))
    tmpb = np.asarray(pywt.downcoef('d', tmpa, wt_type, level=3, mode=wt_mode))
    LF.append(np.asarray(pywt.downcoef('a', tmpb, wt_type, level=1, mode=wt_mode)))
    HF.append(np.asarray(pywt.downcoef('d', tmpb, wt_type, level=1, mode=wt_mode)))
    HF.append(np.asarray(pywt.downcoef('d', tmpa, wt_type, level=2, mode=wt_mode)))
    HF.append(np.asarray(pywt.downcoef('d', tmpa, wt_type, level=1, mode=wt_mode)))
    tmpa = np.asarray(pywt.downcoef('d', y2, wt_type, level=3, mode=wt_mode))
    HF.append(np.asarray(pywt.downcoef('a', tmpa, wt_type, level=1, mode=wt_mode)))
    tmpb = np.asarray(pywt.downcoef('d', tmpa, wt_type, level=1, mode=wt_mode))
    HF.append(np.asarray(pywt.downcoef('a', tmpb, wt_type, level=3, mode=wt_mode)))
    tmpc = np.asarray(pywt.downcoef('d', tmpb, wt_type, level=3, mode=wt_mode))
    HF.append(np.asarray(pywt.downcoef('a', tmpc, wt_type, level=1, mode=wt_mode)))
    pr = np.zeros(6)
    ent = np.zeros(4)
    for index in range(0, len(ULF)):
        pr[0] += np.sum(np.abs(ULF[index]) ** 2) / (2 * len(y2))
        ent[0] += Shannon(ULF[index], len(y2))
    for index in range(0, len(VLF)):
        pr[1] += np.sum(np.abs(VLF[index]) ** 2) / (2 * len(y2))
        ent[1] += Shannon(VLF[index], len(y2))
    for index in range(0, len(LF)):
        pr[2] += np.sum(np.abs(LF[index]) ** 2) / (2 * len(y2))
```

```
        ent[2] += Shannon(LF[index], len(y2))
    for index in range(0, len(HF)):
        pr[3] += np.sum(np.abs(HF[index]) ** 2) / (2 * len(y2))
        ent[3] += Shannon(HF[index], len(y2))
    pr[4] = pr[0] + pr[1] + pr[2] + pr[3]
    pr[5] = pr[2] / pr[3]
    return pr, ent
```

7.1.11. Poincare çizimi öznitelikleri için kullanılan Python Kod Parçacığı

```
def poincare_features(rr):
    x1 = np.asarray(rr[:-1])
    x2 = np.asarray(rr[1:])
    sd1 = np.std(np.subtract(x1, x2) / np.sqrt(2))
    sd2 = np.std(np.add(x1, x2) / np.sqrt(2))
    area = np.pi * sd1 * sd2
    pcare_features = {
        "sd1": sd1,
        "sd2": sd2,
        "sd2/sd1": sd2 / sd1,
        "sd1xsd2": area,
    }
    return pcare_features
```

7.1.12. Öznitelik normalizasyonu için kullanılan Python Kod Parçacığı

```
def feat_norm(feats):
    feats_minmax = np.nan_to_num((feats - feats.min(axis=0)) / feats.ptp(axis=0))
    feats_z = np.nan_to_num((feats - feats.mean(axis=0)) / feats.std(axis=0))
    return feats_minmax,feats_z
```

7.1.13. Öznitelik seçimi için kullanılan GA temelli Python Kod Parçacığı

```
class GeneticSelector():
    def __init__(self, n_gen, size, n_best, n_rand,
                 n_children, mutation_rate,k):
        self.n_gen = n_gen
        self.size = size
        self.n_best = n_best
        self.n_rand = n_rand
        self.n_children = n_children
        self.mutation_rate = mutation_rate
        self.k=k
        if int((self.n_best + self.n_rand) / 2) * self.n_children != self.size:
            raise ValueError("The population size is not stable.")

    def initilize(self):
        population = []
        for i in range(self.size):
            chromosome = np.ones(self.n_features, dtype=np.bool)
            mask = np.random.rand(len(chromosome)) < 0.3
            chromosome[mask] = False
            population.append(chromosome)
        return population

    def fitness(self, population):
        X, y = self.dataset
        scores = []
        for chromosome in population:
            score = 1-knn_hesapla(self.k,X,y,chromosome)
            scores.append(score)
        scores, population = np.array(scores), np.array(population)
        inds = np.argsort(scores)
        return list(scores[inds]), list(population[inds, :])
```

```
    def select(self, population_sorted):
        population_next = []
        for i in range(self.n_best):
            population_next.append(population_sorted[i])
        for i in range(self.n_rand):
            population_next.append(random.choice(population_sorted))
        random.shuffle(population_next)
        return population_next

    def crossover(self, population):
        population_next = []
        for i in range(int(len(population) / 2)):
            for j in range(self.n_children):
                chromosome1, chromosome2 = population[i], population[len(population) - 1 -
i]
                child = chromosome1
                mask = np.random.rand(len(child)) > 0.5
                child[mask] = chromosome2[mask]
                population_next.append(child)
        return population_next

    def mutate(self, population):
        population_next = []
        for i in range(len(population)):
            chromosome = population[i]
            if random.random() < self.mutation_rate:
                mask = np.random.rand(len(chromosome)) < 0.05
                chromosome[mask] = False
            population_next.append(chromosome)
        return population_next

    def generate(self, population):
        # Selection, crossover and mutation
        scores_sorted, population_sorted = self.fitness(population)
```

```
            population = self.select(population_sorted)
            population = self.crossover(population)
            population = self.mutate(population)
            self.chromosomes_best.append(population_sorted[0])
            self.scores_best.append(scores_sorted[0])
            self.scores_avg.append(np.mean(scores_sorted))
            return population

        def fit(self, X, y):
            self.chromosomes_best = []
            self.scores_best, self.scores_avg = [], []
            self.dataset = X, y
            self.n_features = X.shape[1]
            population = self.initilize()
            for i in range(self.n_gen):
                population = self.generate(population)
            return self

        @property
        def support_(self):
            return self.chromosomes_best[-1]

    def knn_hesapla(_k,_dataX,_dataY, _feat):
        len_data=len(_dataX)
        _pred_all=np.zeros(len_data)
        _pred_sel=np.zeros(len_data)
        for i in range(len_data):
            mask=np.arange(len_data)!=i
            knn_alldata=KNeighborsClassifier(n_neighbors=int(_k),metric='minkowski')
            knn_alldata.fit(_dataX[mask,:],_dataY[mask])
            _pred_all[i]=(knn_alldata.predict(_dataX[~mask,:]))
            knn_seldata=KNeighborsClassifier(n_neighbors=int(_k),metric='minkowski')
            knn_seldata.fit(_dataX[mask,:][:, _feat>0.5],_dataY[mask])
            _pred_sel[i]=int(knn_seldata.predict(_dataX[~mask,:][:, _feat>0.5]))
```

```
        del knn_alldata,knn_seldata
    _cm = confusion_matrix(_dataY, _pred_sel)
    acc=(np.sum(np.diagonal(_cm)))/(np.sum(_cm))
    return acc

def hesapla(X,y)
    secimler=[]
    for k in range(1,23,2):
        sel = GeneticSelector(n_gen=2, size=40, n_best=4, n_rand=4,
                              n_children=10, mutation_rate=0.05,k=k)
        sel.fit(X, y)
        score = 1-knn_hesapla(k,X,y,sel.support_)
        res = [i for i, val in enumerate(sel.support_) if val]
        secimler.append((score,res))
    return secimler
```

7.1.14. kNN sınıflandırıcı Python Kod Parçacığı

```
def knn_hesapla(_k, _dataX, _dataY, _feat):
    len_data = len(_dataX)
    _pred_all = np.zeros(len_data)
    _pred_sel = np.zeros(len_data)
    for i in range(len_data):
        mask = np.arange(len_data) != i
        knn_alldata = KNeighborsClassifier(n_neighbors=int(_k), metric='minkowski')
        knn_alldata.fit(_dataX[mask, :], _dataY[mask])
        _pred_all[i] = (knn_alldata.predict(_dataX[~mask, :]))
        knn_seldata = KNeighborsClassifier(n_neighbors=int(_k), metric='minkowski')
        knn_seldata.fit(_dataX[mask, :][:, _feat > 0.5], _dataY[mask])
         pred_sel[i] = int(knn_seldata.predict(_dataX[~mask, :][:, _feat > 0.5]))
        _cr = classification_report(_dataY, _pred_sel, target_names=CL_NAMES_,
output_dict=True)
```

```
    _cr_all = classification_report(_dataY, _pred_all, target_names=CL_NAMES_,
output_dict=True)
    return _cr_all,_cr
```

7.1.15. MLP sınıflandırıcı Python Kod Parçacığı

```
def fitness_func(solution, sol_idx):
    global GANN_instance, data_inputs, data_outputs
    predictions = pygad.nn.predict(last_layer=
GANN_instance.population_networks[sol_idx],
                                   data_inputs=data_inputs)
    correct_predictions = np.where(predictions == data_outputs)[0].size
    solution_fitness = (correct_predictions / data_outputs.size) * 100
    return solution_fitness

def callback_generation(ga_instance):
    global GANN_instance, last_fitness , duragan
    population_matrices = pygad.gann.population_as_matrices(population_networks=
GANN_instance.population_networks,
                                                population_vectors=ga_instance.population)
    GANN_instance.update_population_trained_weights(population_trained_weights=
population_matrices)
    if ga_instance.best_solution()[1] - last_fitness ==0:
        duragan=duragan+1
    else:
        duragan=0
    if (duragan>100 or ga_instance.best_solution()[1]>99): return "stop"
    last_fitness = ga_instance.best_solution()[1].copy()

def mlp_siniflandir(X, y):
    global GANN_instance, data_inputs, data_outputs, last_fitness
    _dataXtrain, _dataXtest, _dataYtrain, _dataYtest =  train_test_split(X,y, test_size =
0.1, random_state = 42)
```

```
    data_outputs=_dataYtrain
    data_inputs=_dataXtrain
    num_inputs = data_inputs.shape[1]
    num_classes = 2
    num_solutions = 100
    GANN_instance = pygad.gann.GANN(num_solutions=num_solutions,
                                    num_neurons_input=num_inputs,
                                    num_neurons_hidden_layers=[30],
                                    num_neurons_output=num_classes,
                                    hidden_activations=["relu"],
                                    output_activation="softmax")
    population_vectors =
pygad.gann.population_as_vectors(population_networks=GANN_instance.population_n
etworks)
    initial_population = population_vectors.copy()
    num_parents_mating = 8
    num_generations = 500
    mutation_percent_genes = 30
    ga_instance = pygad.GA(num_generations=num_generations,
                           num_parents_mating=num_parents_mating,
                           initial_population=initial_population,
                           fitness_func=fitness_func,
                           mutation_percent_genes=mutation_percent_genes,
                           on_generation=callback_generation)
    ga_instance.run()
    predictions = pygad.nn.predict(last_layer=
GANN_instance.population_networks[solution_idx],
                                   data_inputs=data_inputs)
    return predictions
```

7.1.16. SVM sınıflandırıcı Python Kod Parçacığı

```
def siniflandir_svm_rbf(X, y, test_size=0.15, n_splits=10):
```

```
    C_range=np.array([0.001,0.1,1,10,50])
    gamma_range=np.array([0.001,0.1,1,10])
    param_grid = dict(gamma=gamma_range, C=C_range)
    cv = StratifiedShuffleSplit(n_splits=n_splits, test_size=test_size, random_state=42)
    grid = GridSearchCV(SVC(), param_grid=param_grid, cv=cv)
    grid.fit(X, y)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = test_size, random_state = 42)
    SV_classifier = make_pipeline(StandardScaler(), SVC(gamma=grid.best_params_['gamma'], C=grid.best_params_['C']))
    SV_classifier.fit(X_train, y_train)
    pred_all = SV_classifier.predict(X)
    pred_train = SV_classifier.predict(X_train)
    pred_test = SV_classifier.predict(X_test)
    acc_all = classification_accuracy(y, pred_all)
    acc_train = classification_accuracy(y_train, pred_train)
    acc_test = classification_accuracy(y_test, pred_test)
    return acc_all, pred_all,acc_train,acc_test, SV_classifier.get_params()
def siniflandir_svm_lin(X, y, test_size=0.15, n_splits=10):
    C_range=np.array([0.001,0.1,1,10,50])
    param_grid = dict(kernel= ['linear'],C=C_range)
    cv = StratifiedShuffleSplit(n_splits=n_splits, test_size=test_size, random_state=42)
    grid = GridSearchCV(SVC(), param_grid=param_grid, cv=cv)
    grid.fit(X, y)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = test_size, random_state = 42)
```

```
    SV_classifier = make_pipeline(StandardScaler(),
SVC(C=grid.best_params_['C'],kernel='linear'))
    SV_classifier.fit(X_train, y_train)
    pred_all = SV_classifier.predict(X)
    pred_train = SV_classifier.predict(X_train)
    pred_test = SV_classifier.predict(X_test)
    acc_all = classification_accuracy(y, pred_all)
    acc_train = classification_accuracy(y_train, pred_train)
    acc_test = classification_accuracy(y_test, pred_test)
    return acc_all, pred_all,acc_train,acc_test, grid.get_params()
```

7.1.17. CNN sınıflandırıcı Python Kod Parçacığı

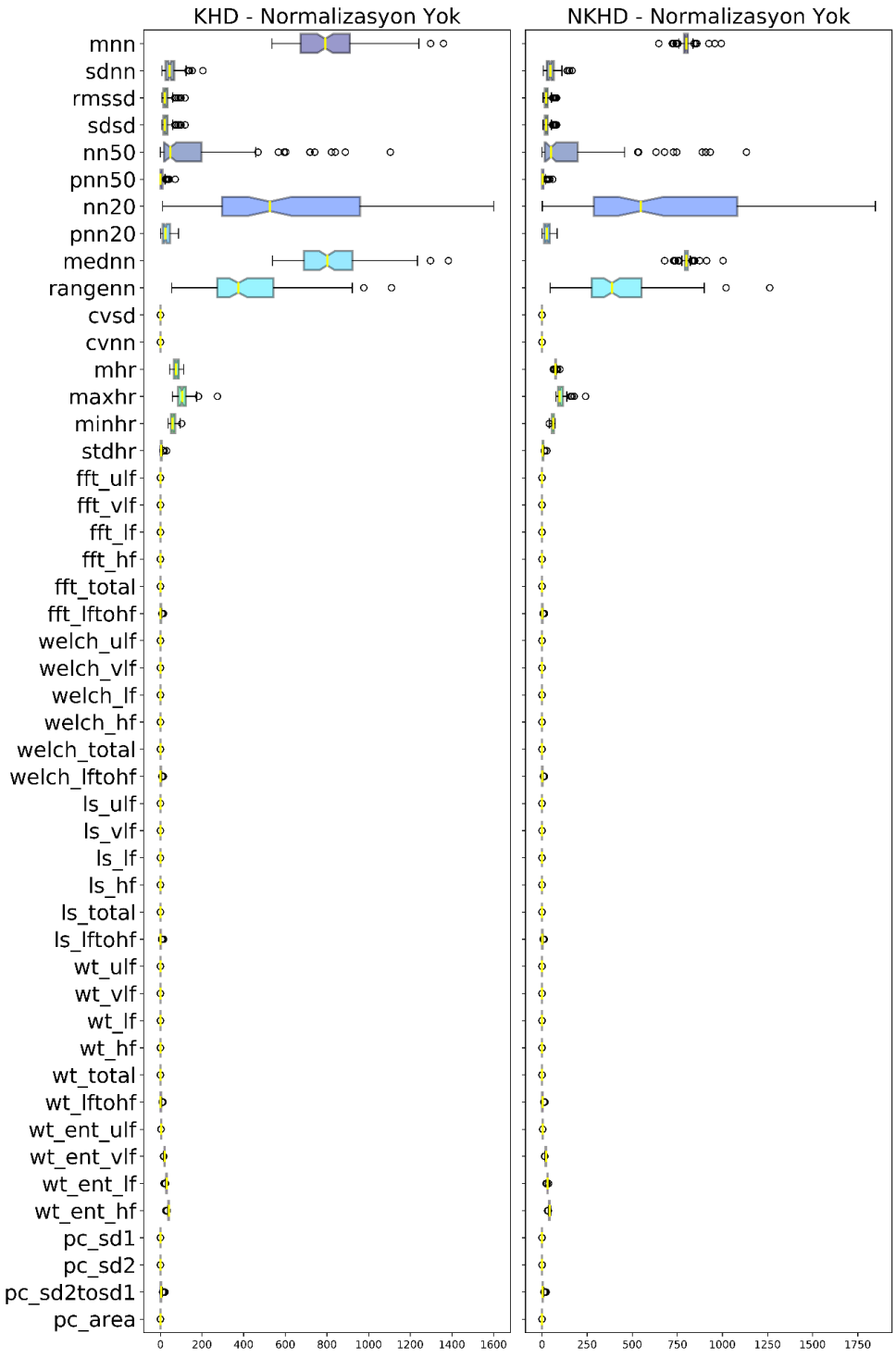
```
from hyperopt import fmin, tpe, hp, STATUS_OK, Trials, space_eval

DROPOUT_CHOICES = np.arange(0.1,0.8, 0.1)
UNIT_CHOICES = np.arange(8, 257, 8, dtype=int)
FILTER_CHOICES = list(range(1, 9, 1))
POOL_CHOICES = list(range(1, 4, 1))
EMBED_UNITS = np.arange(32, 513, 32, dtype=int)
space = {
    'conv1_units': hp.choice('conv1_units', UNIT_CHOICES),
    'conv1_filters': hp.choice('conv1_filters', FILTER_CHOICES),
    'conv2': hp.choice('conv2', [False, {
        'conv2_units': hp.choice('conv2_units', UNIT_CHOICES),
        'conv2_filters': hp.choice('conv2_filters', FILTER_CHOICES),
        'conv3': hp.choice('conv3', [False, {
            'conv3_units': hp.choice('conv3_units', UNIT_CHOICES),
            'conv3_filters': hp.choice('conv3_filters', FILTER_CHOICES),
        }]),
    }]),
```

```
    'pool_size': hp.choice('pool_size',POOL_CHOICES),
    'dense_units': hp.choice('dense_units', UNIT_CHOICES),
    'batch_size': hp.choice('batch_size', UNIT_CHOICES),
    'dropout1': hp.choice('dropout1', DROPOUT_CHOICES),
}

def parcala(X,y,boyut):
    i=0
    b = (np.array(list(chunks(X[0],boyut))+list([0]))[:-2])
    yy = np.ones((len(b), 1), dtype=int)*y[0]
    for row in X[1:]:
        i+=1
        c = (np.array(list(chunks(row, boyut)) + list([0]))[:-2])
        b=np.concatenate([b,c])
        yy = np.concatenate((yy, np.ones((len(c), 1), dtype=int)*y[i]))
    return np.stack(b),yy.flatten()

def objective(params, verbose=0):
    global X, y, n_splits
    X = X.reshape(X.shape[0], X.shape[1], 1)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/n_splits,
random_state=42)
    history = History()
    model = Sequential()
    model.add(Conv1D(params['conv1_units'], params['conv1_filters'],
input_shape=(X.shape[1], X.shape[2]), padding='same'))
    if params['conv2']:
        model.add(Conv1D(params['conv2']['conv2_units'],
         params['conv2']['conv2_filters'],padding='same'))
    if params['conv2'] and params['conv2']['conv3']:
        model.add(Conv1D(params['conv2']['conv3']['conv3_units'],
         params['conv2']['conv3']['conv3_filters'],padding='same'))
    model.add(Dropout(params['dropout1']))
    model.add(MaxPooling1D(pool_size=params['pool_size']))
```

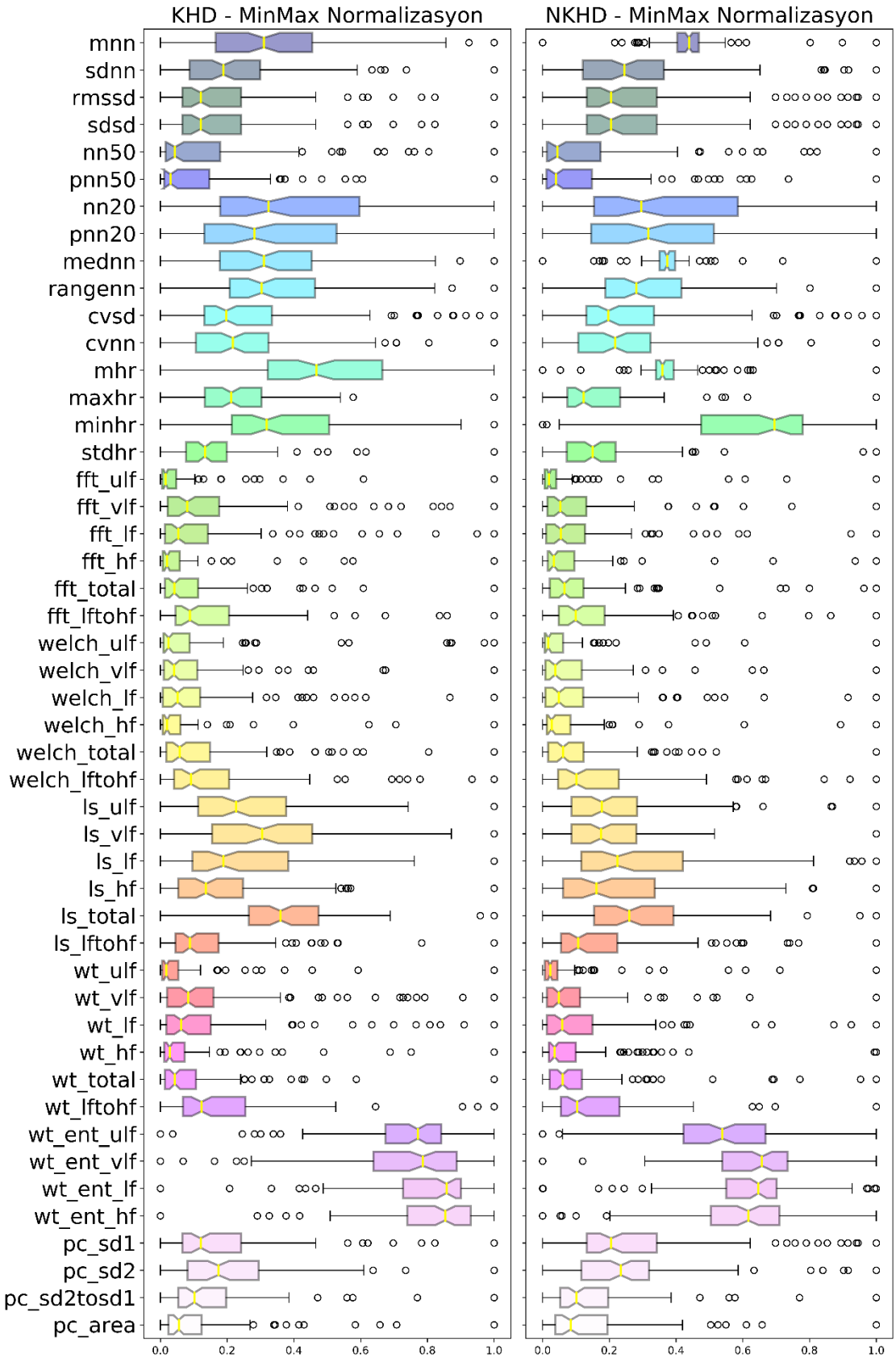
```
    model.add(Flatten())
    model.add(Dense(params['dense_units'], activation='relu'))
    model.add(Dense(np.unique(y).size, activation='softmax'))
    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    model.fit(
        X_train,
        y_train,
        validation_data=(X_test, y_test),
        epochs=1000,
        batch_size=params['batch_size'],
        callbacks=[
            history,
            EarlyStopping(patience=40, verbose=verbose, mode='max', min_delta=0,
             monitor='val_accuracy',restore_best_weights=True)
        ],
        verbose=verbose
    )
    ypred = model.predict(X)
    ypred = np.argmax(ypred, axis=-1)
    pred = model.predict(X_train)
    pred_train = pred.argmax(axis=-1)
    pred = model.predict(X_test)
    pred_test = pred.argmax(axis=-1)
    best_epoch = np.argmin(history.history['val_loss']) + 1
    acc_all = metrics.accuracy_score(y, ypred)
    acc_train = classification_accuracy(y_train, pred_train)
    acc_test =classification_accuracy(y_test, pred_test)
    return {'loss': 1-acc_all,'acc_all':acc_all,'acc_train':acc_train,'acc_test':acc_test,
            'pred_all':ypred,'best_epoch':int(best_epoch),
            'status': STATUS_OK}


def cnn_siniflandir():
```

```
global X, y, n_splits
X = np.array(X)
X, y = parcala(X, y, 250)
X = X.reshape(X.shape[0], X.shape[1], 1)
objective({
    'conv1_units': 17,
    'conv1_filters': 1,
    'conv2': {
        'conv2_units': 11,
        'conv2_filters': 4,
        'conv3': {
            'conv3_units': 8,
            'conv3_filters': 7,
        },
    },
    'pool_size': 1, 'dense_units': 31, 'batch_size': 25, 'dropout1': 0,
}, verbose=2)
trials = Trials()
best = fmin(objective, space, algo=tpe.suggest, trials=trials, max_evals=100,
            rstate=np.random.RandomState(99))
filt_values = space_eval(space, best)
filt_results = objective(filt_values)
pred_all = filt_results['pred_all']
return pred_all,filt_values
```

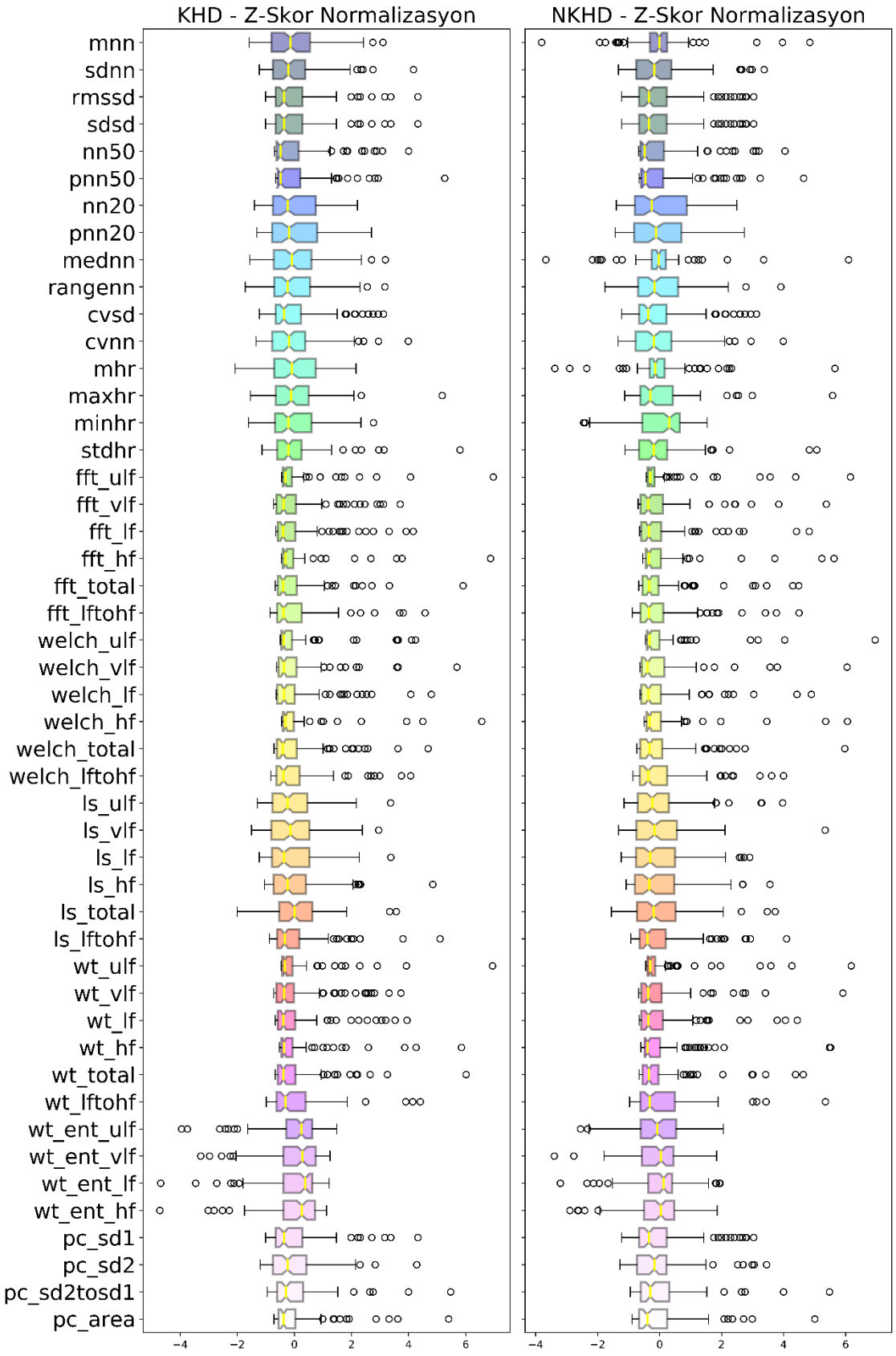
7.2. EK 2: ÖZNİTELİK VERİ DAĞILIMI



Şekil 7.1. Öznitelik normalizasyonu olmayan verilerin kutu grafiği çizimi.

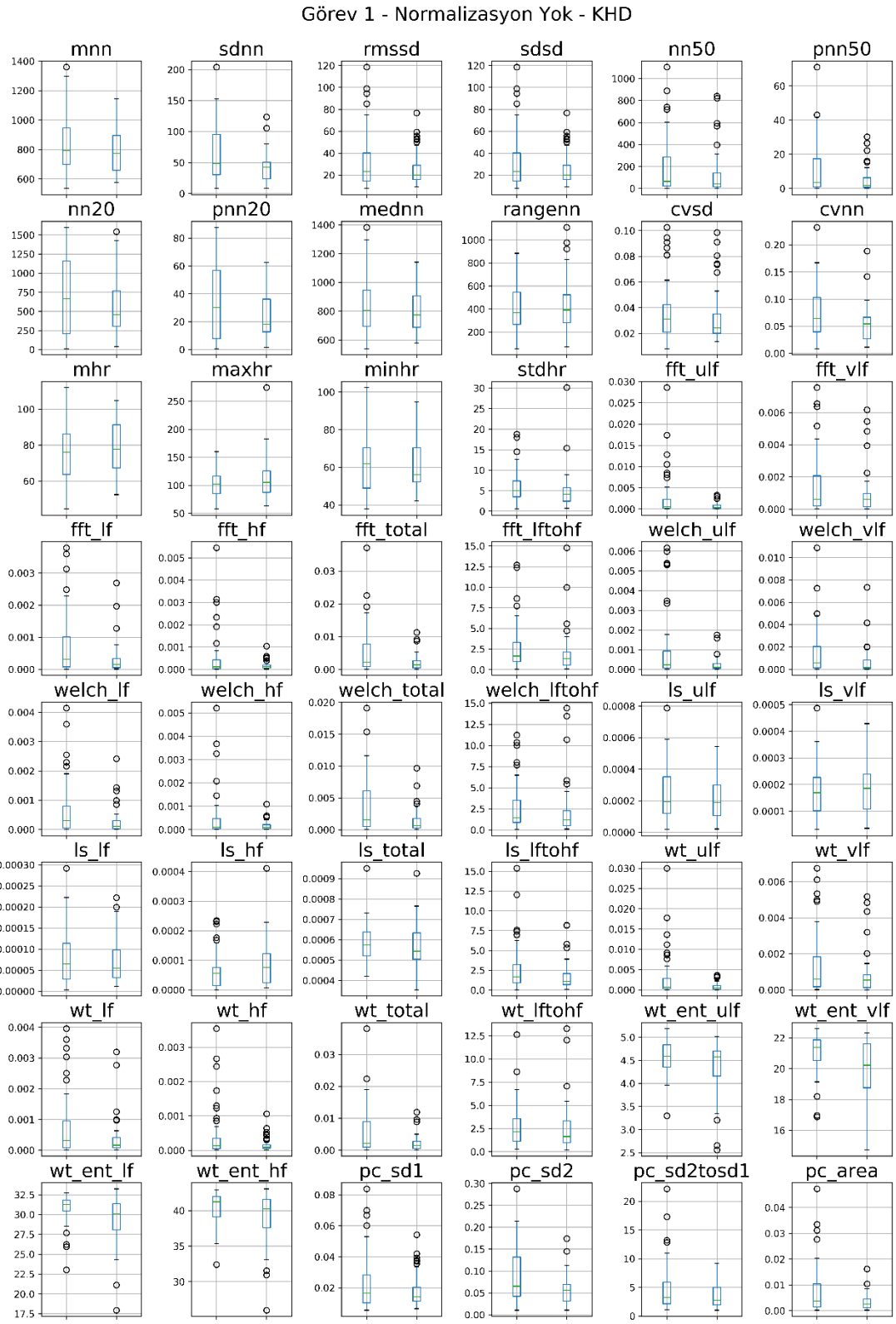


Şekil 7.2. MinMax öznitelik normalizasyonu olan verilerin kutu grafiği çizimi.

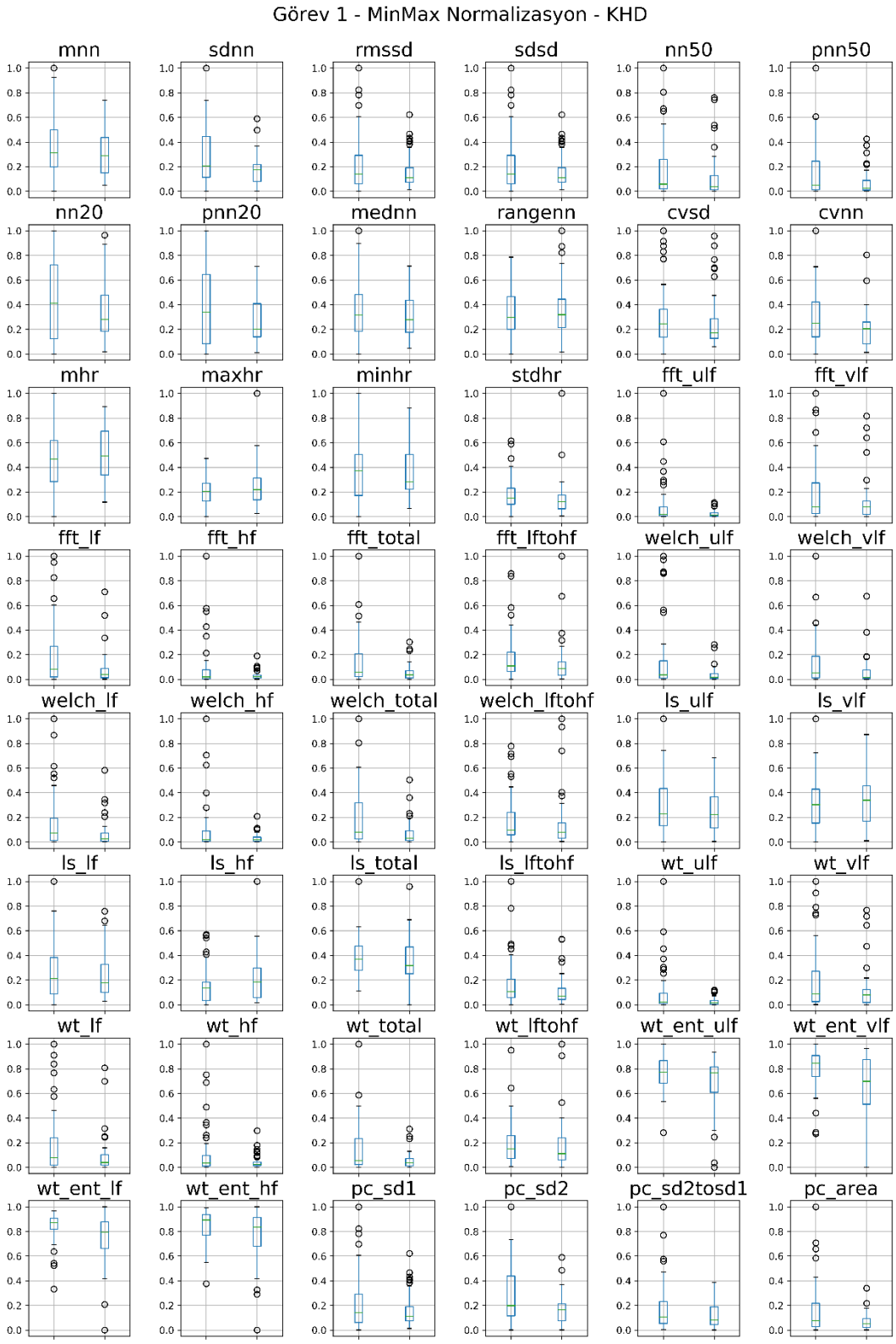


Şekil 7.3. Z-Skor öznitelik normalizasyonu olan verilerin kutu grafiği çizimi.

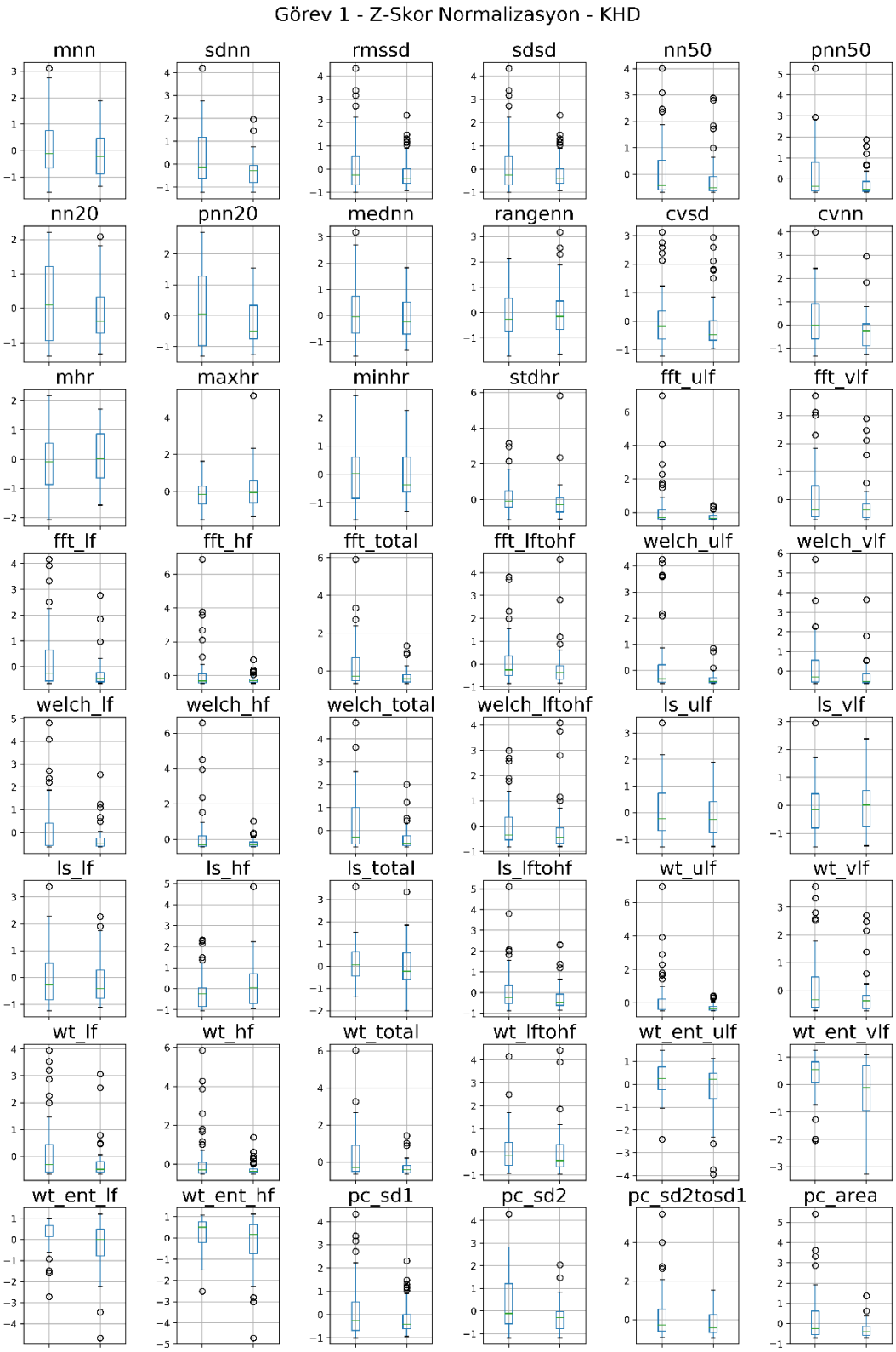
7.3. EK 3: ÖZNİTELİKLERİN SINIFLARA GÖRE DAĞILIMI



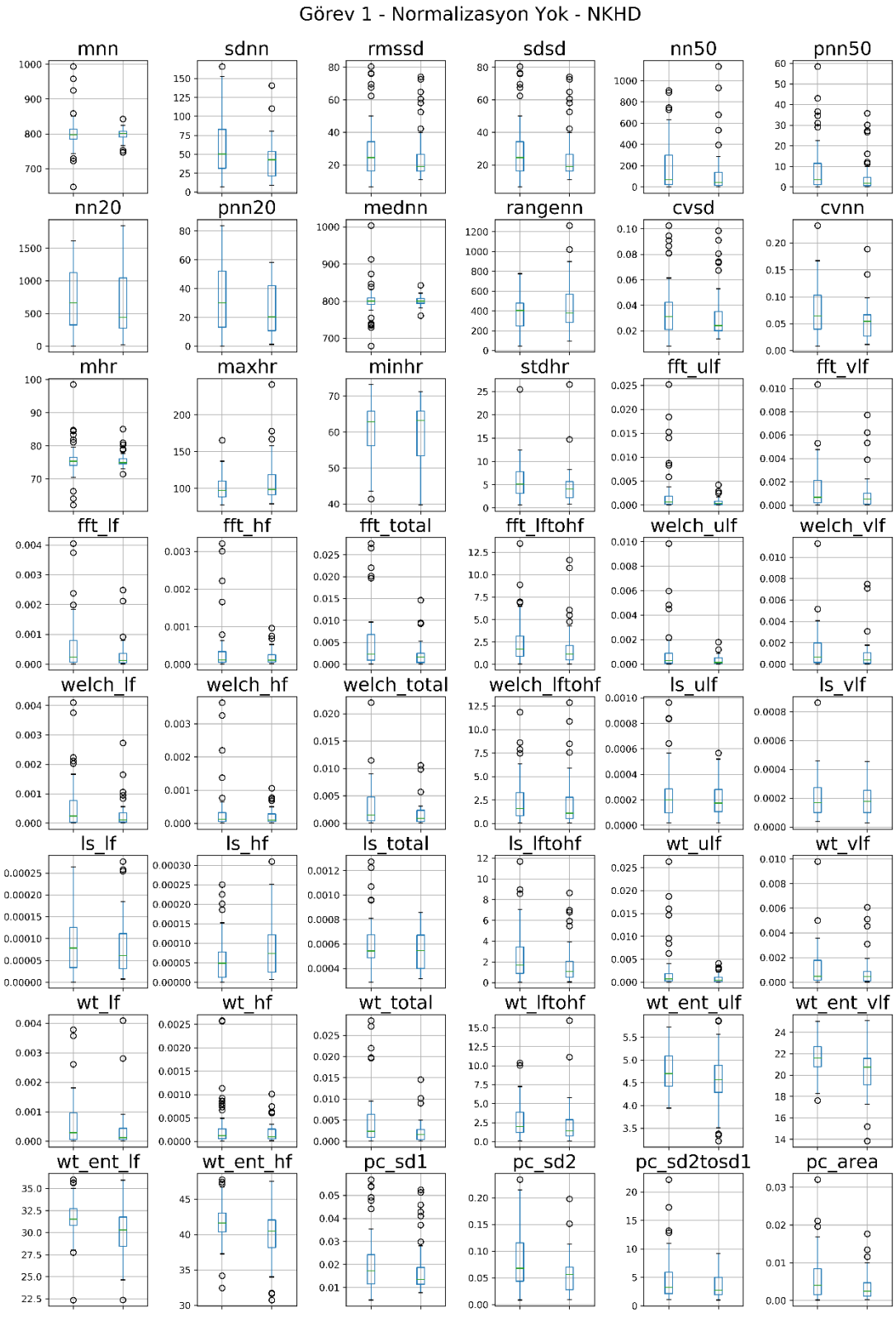
Şekil 7.4. Görev 1 için öznitelik normalizasyonu olmayan KHD verilerinin sınıflara göre kutu grafiği çizimi.



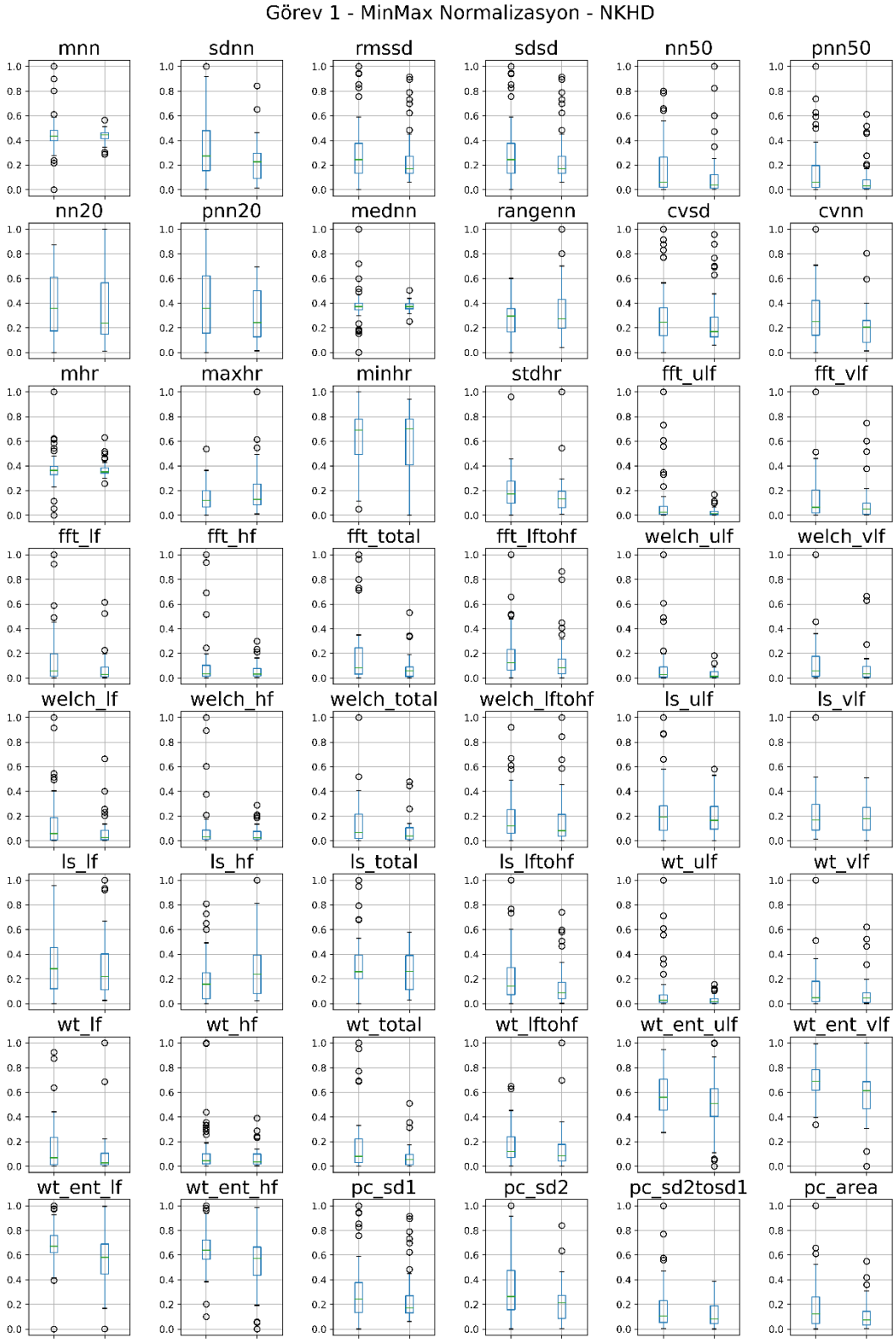
Şekil 7.5. Görev 1 için MinMax öznitelik normalizasyonu olan KHD verilerinin sınıflara göre kutu grafiği çizimi.



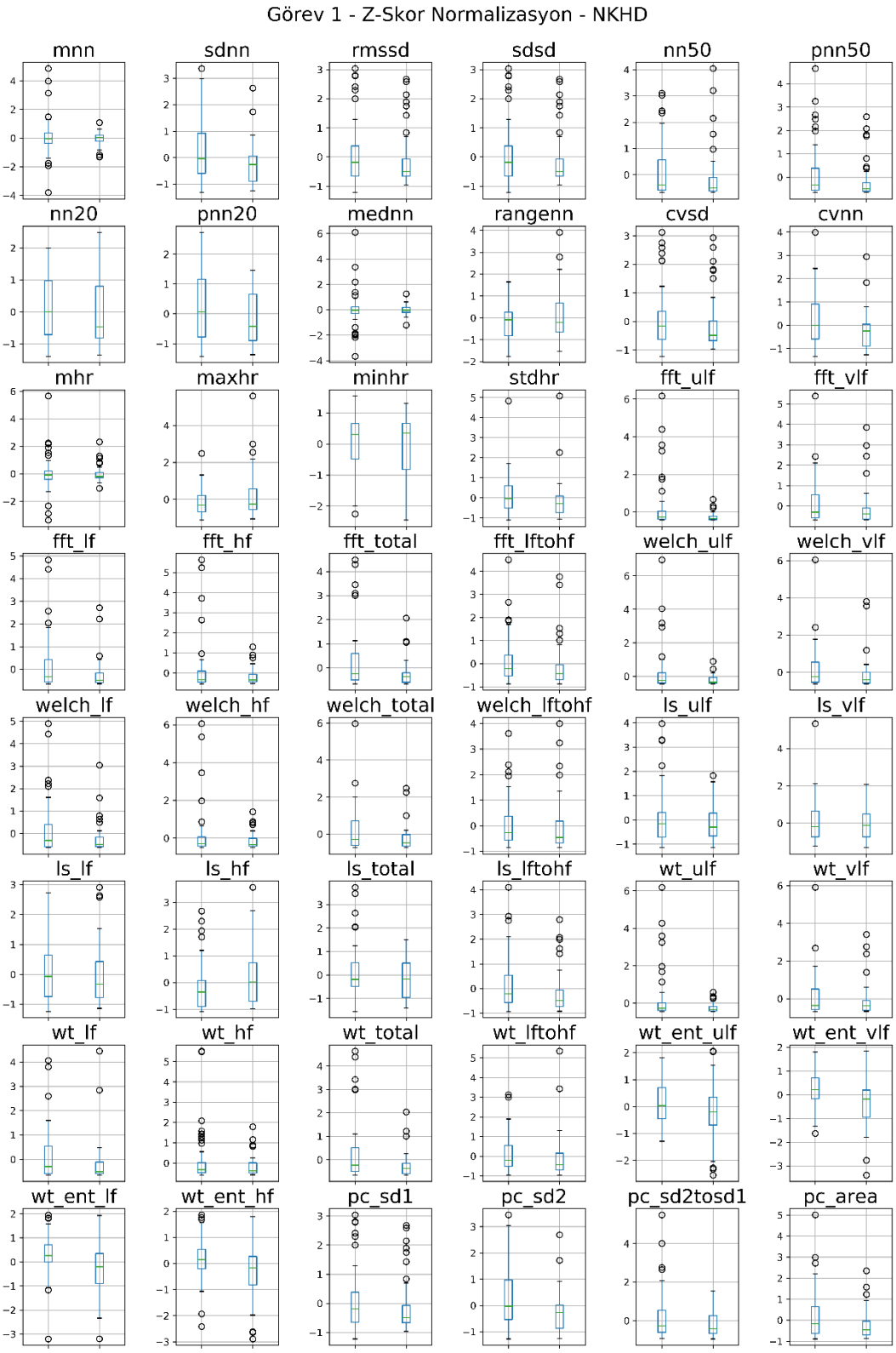
Şekil 7.6. Görev 1 için Z-Skor öznitelik normalizasyonu olan KHD verilerinin sınıflara göre kutu grafiği çizimi.



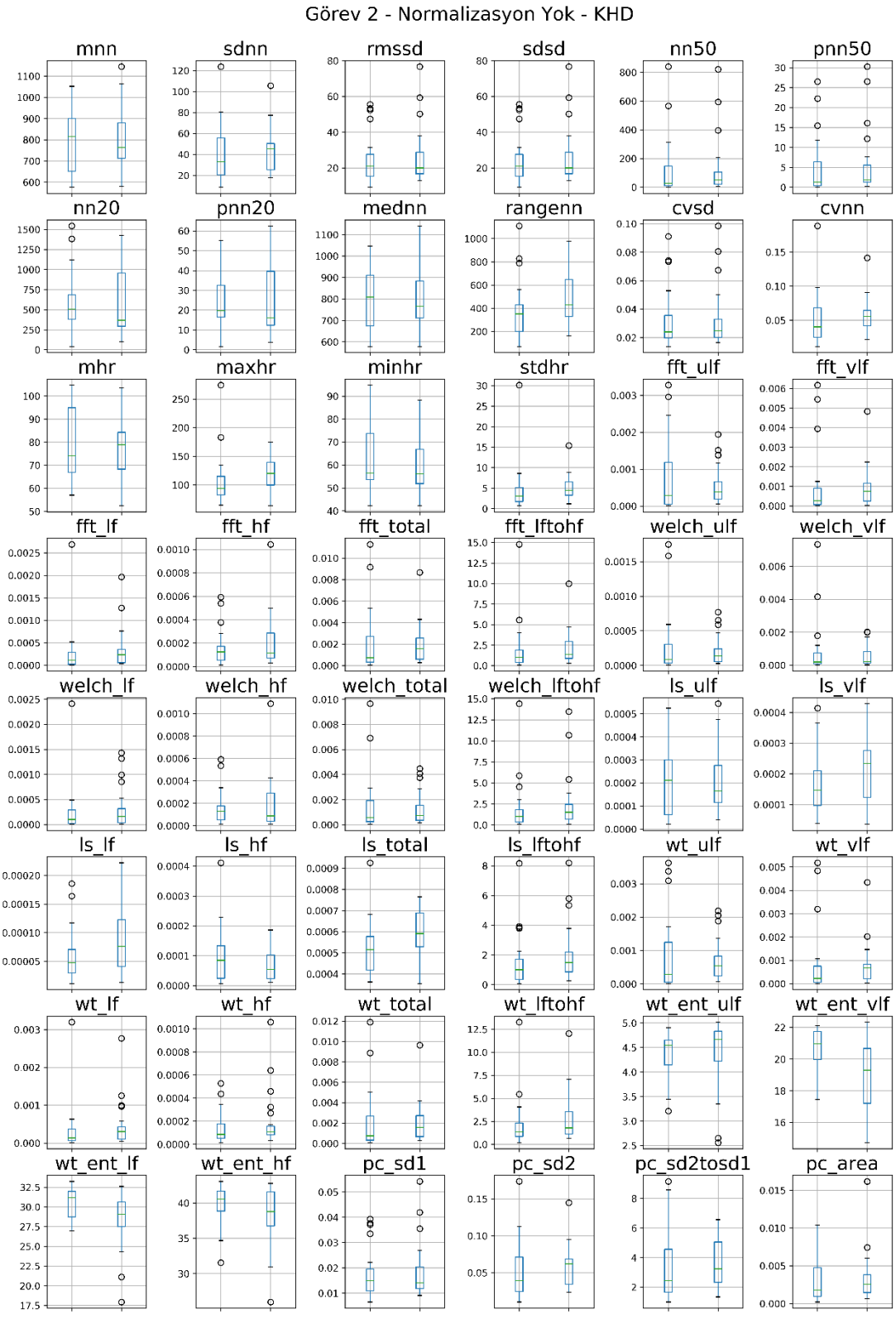
Şekil 7.7. Görev 1 için öznitelik normalizasyonu olmayan NKHD verilerinin sınıflara göre kutu grafiği çizimi.



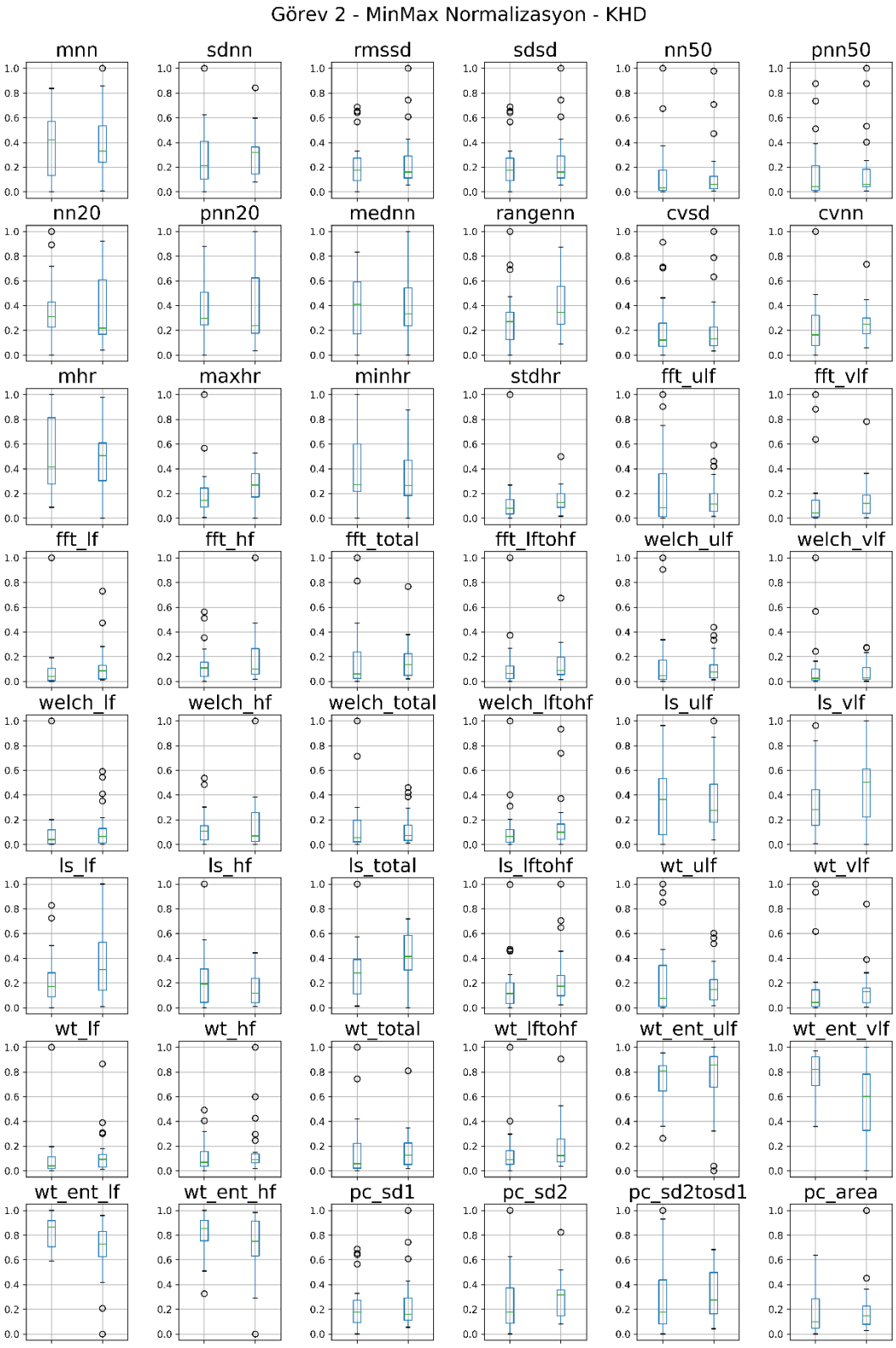
Şekil 7.8. Görev 1 için MinMax öznitelik normalizasyonu olan NKHD verilerinin sınıflara göre kutu grafiği çizimi.



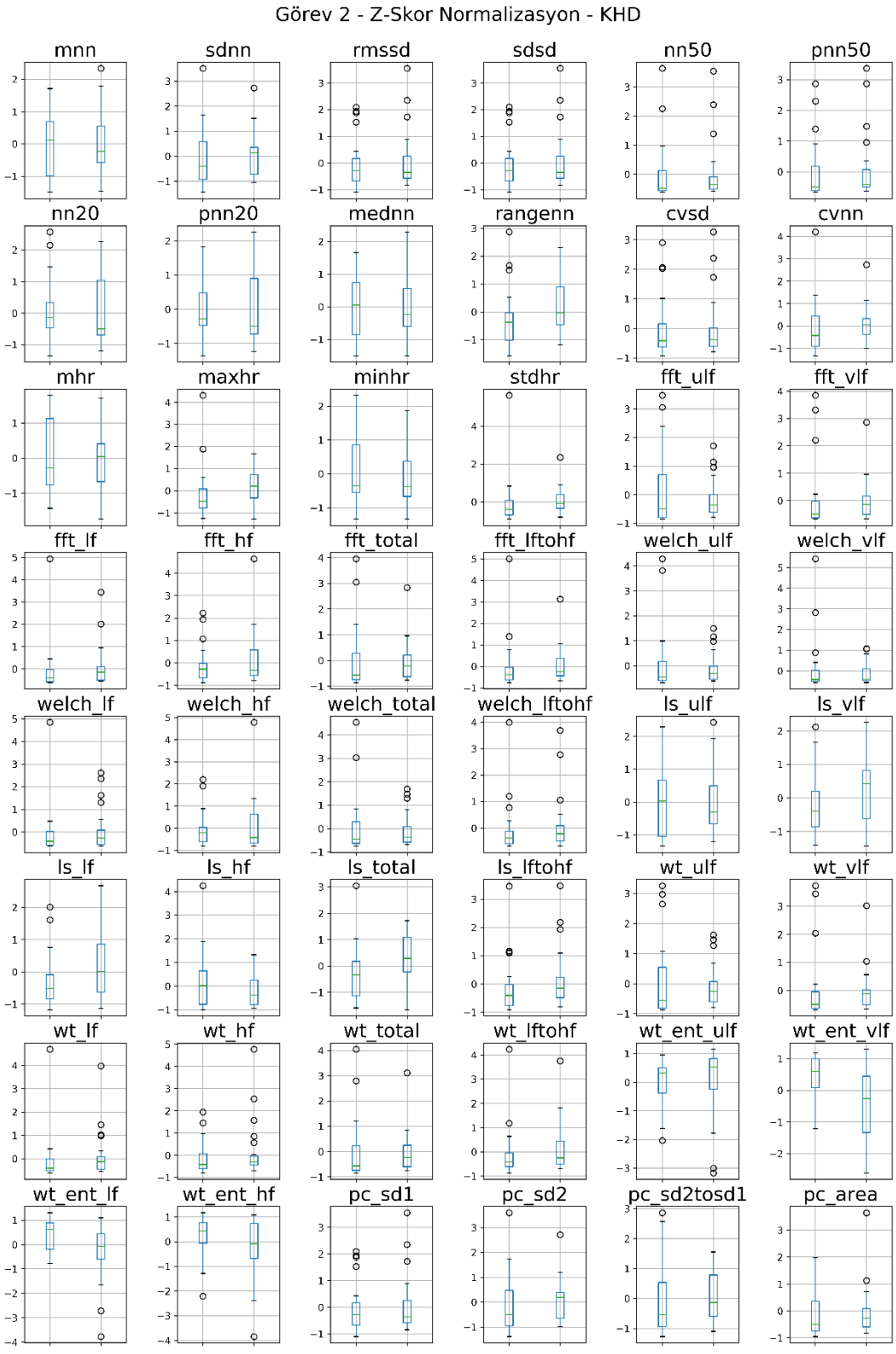
Şekil 7.9. Görev 1 için Z-Skor öznitelik normalizasyonu olan NKHD verilerinin sınıflara göre kutu grafiği çizimi.



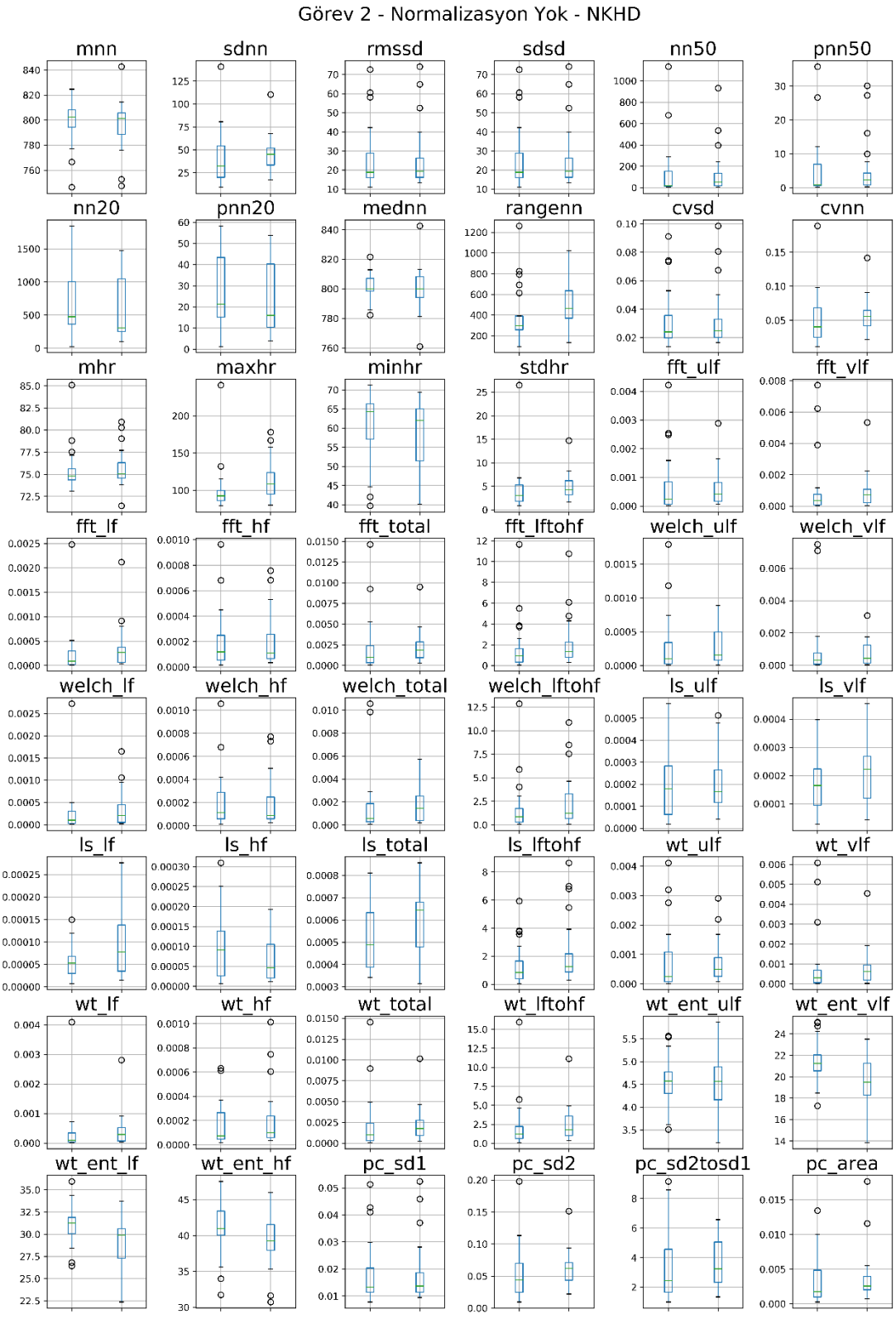
Şekil 7.10. Görev 2 için öznitelik normalizasyonu olmayan KHD verilerinin sınıflara göre kutu grafiği çizimi.



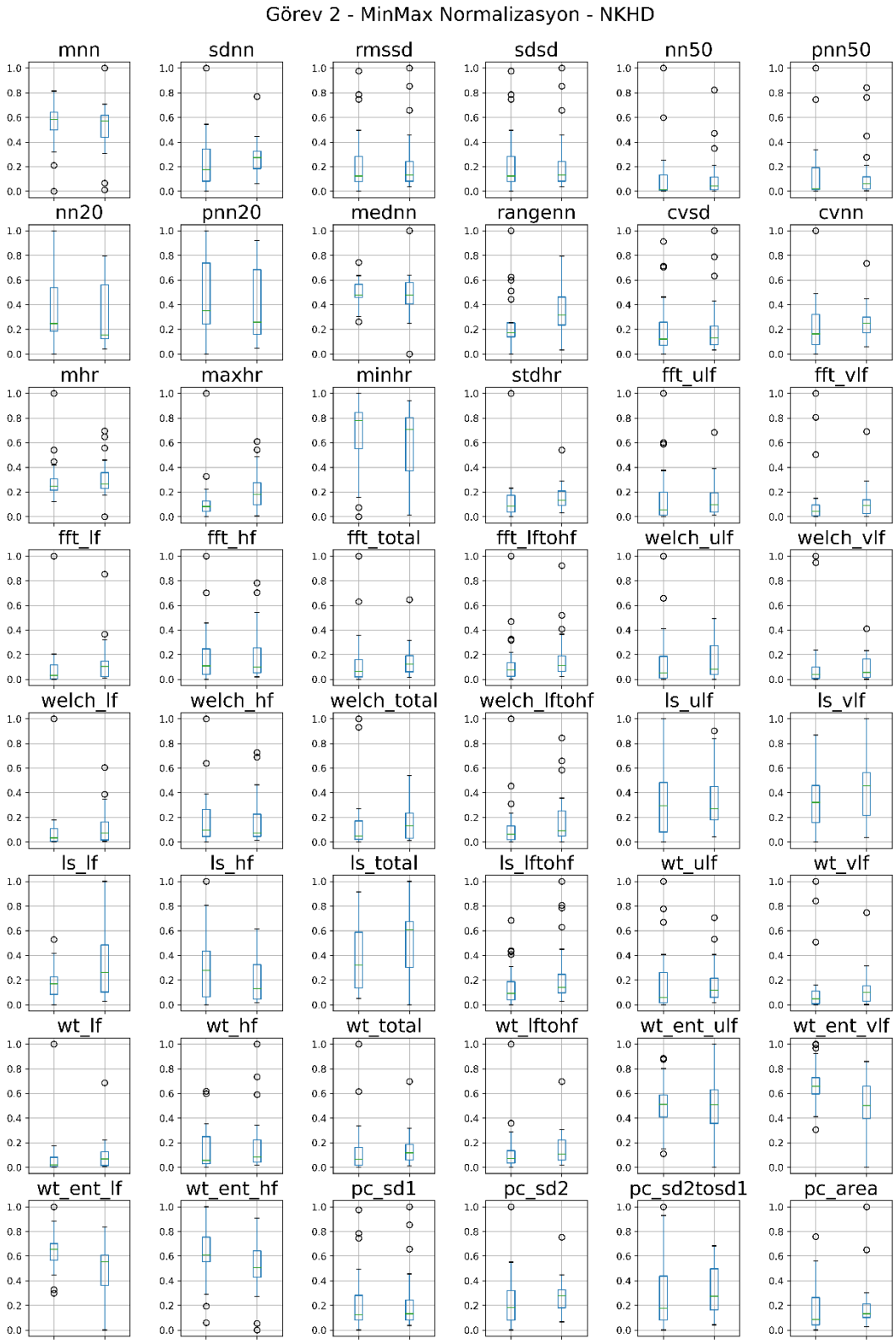
Şekil 7.11. Görev 2 için MinMax öznitelik normalizasyonu olan KHD verilerinin sınıflara göre kutu grafiği çizimi.



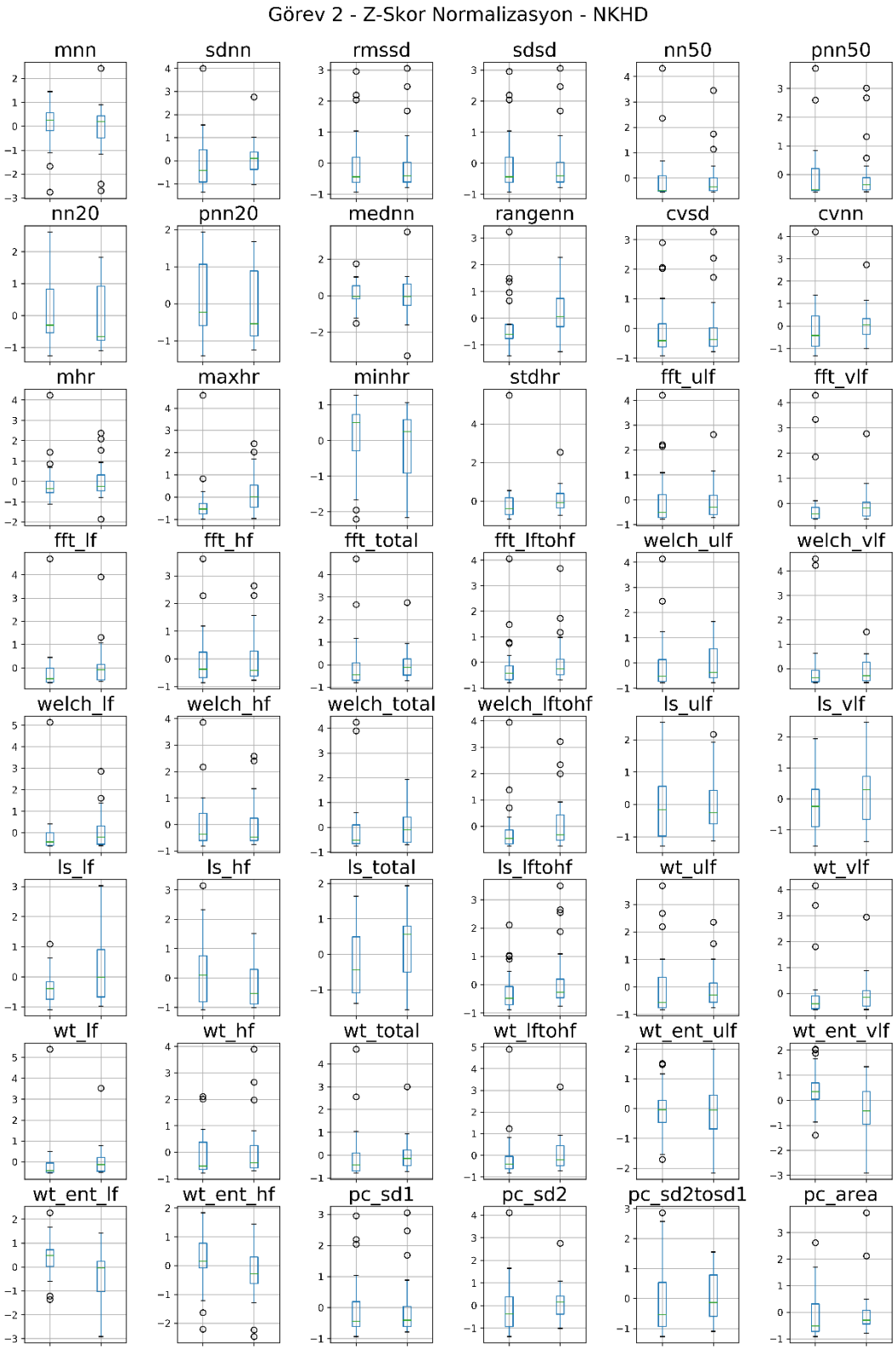
Şekil 7.12. Görev 2 için Z-Skor öznitelik normalizasyonu olan KHD verilerinin sınıflara göre kutu grafiği çizimi.



Şekil 7.13. Görev 2 için öznitelik normalizasyonu olmayan NKHD verilerinin sınıflara göre kutu grafiği çizimi.



Şekil 7.14. Görev 2 için MinMax öznitelik normalizasyonu olan NKHD verilerinin sınıflara göre kutu grafiği çizimi.



Şekil 7.15. Görev 2 için Z-Skor öznitelik normalizasyonu olan NKHD verilerinin sınıflara göre kutu grafiği çizimi.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı :Murat SÜRÜCÜ

Yabancı Dili :İngilizce

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Doktora	Bilgisayar Müh.	Düzce Üniversitesi	2021
Y. Lisans	Elektrik Elektronik Müh.	Bülent Ecevit Üniversitesi	2010
Lisans	Elektronik ve Haberleşme Müh.	Süleyman Demirel Üniversitesi	2002
Lise	Bilgisayar Donanım	Çınarlı Teknik Lisesi	1997

[1] M. Surucu, D. Surucu, ve R. Hacioglu, "Detecting and tracking moving objects in real time images via active camera", içinde *2010 IEEE 18th Signal Processing and Communications Applications Conference*, Nis. 2010, ss. 764–767.

[2] K. Koksal, D. Surucu, M. Surucu, ve R. Hacioglu, "Visual line tracking with vector field guidance for UAV", içinde *2014 22nd Signal Processing and Communications Applications Conference (SIU)*, Nis. 2014, ss. 646–649.

[3] D. Surucu, M. Surucu, K. Koksal, ve R. Hacioglu, "Visual tracking and control of Unmanned Aerial Vehicle", içinde *2015 23nd Signal Processing and Communications Applications Conference (SIU)*, May. 2015, ss. 1849–1852.

[4] M. Surucu, Y. Isler, ve R. Kara, "Investigation of the Effect of Normalization Techniques on Discriminating Patients with Paroxysmal Atrial Fibrillation", içinde *2nd International Conference of Applied Sciences, Engineering and Mathematics*, 2020, s. 12.

[5] M. Surucu, Y. Isler, ve R. Kara, "Heart Rate Normalization on Determining a Paroxysmal Atrial Fibrillation Attack", içinde *3rd International Conference of Applied Sciences, Engineering and Mathematics*, 2021, s. 19.

[6] M. Surucu, Y. Isler, ve R. Kara, "Diagnosis of Paroxysmal Atrial Fibrillation from Thirty-Minute Heart Rate Variability Data using Convolutional Neural Networks", *Turkish J. Electr. Eng. Comput. Sci.*, 2021.

[7] M. Surucu, Y. Isler, M. Perc, ve R. Kara, "Convolutional Neural Networks Predict the Onset of Paroxysmal Atrial Fibrillation: Theory and Applications", *Chaos An Interdiscip. J. Nonlinear Sci. Accept.*, 2021.

[8] M. Surucu, Y. Isler, ve R. Kara, "Effect of heart rate normalization on 30-minute heart rate variability data used in the diagnosis of paroxysmal atrial fibrillation", *J. Fac. Eng. Archit. Gazi Univ. Under Rev.*, 2021.