



T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ÖZELLİK ÇIKARMA VE DVM TABANLI ADABOOST ALGORİTMASI İLE BİYOMEDİKAL VERİ SINIFLANDIRMA

Mücahid BARSTUĞAN

YÜKSEK LİSANS

Elektrik Elektronik Mühendisliği Anabilim Dalı

Aralık 2014
KONYA
Her Hakkı Saklıdır

ÖZET

YÜKSEK LİSANS

ÖZELLİK ÇIKARMA VE DVM TABANLI ADABOOST ALGORİTMASI İLE BİYOMEDİKAL VERİ SINIFLANDIRMA

Mücahid BARSTUĞAN

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Rahime CEYLAN

2014, 91 Sayfa

Jüri
Danışman: Yrd. Doç. Dr. Rahime CEYLAN
Doç. Dr. Seral ÖZŞEN
Yrd. Doç. Dr. Nurdan BAYKAN

Günümüzde makine öğrenme algoritmaları hastalığın tanı ve teşhisinde doktorlara yardımcı olmaktadır. Bu amaçla kullanılan öğrenme algoritmalarından bazıları Destek Vektör Makineleri, AdaBoost ve Yapay Sinir Ağları' dır. AdaBoost birden fazla zayıf sınıflandırıcının bir araya getirilmesi ile oluşturulan bir topluluk sınıflandırıcıdır.

Tez çalışmasında AdaBoost topluluk sınıflandırıcısının biyomedikal veriler üzerindeki performansını artırmak için özellik seçme ve çıkarma algoritmaları ile uygulamalar gerçekleştirilmiştir. Bu amaçla Sıralı Özellik Seçimi ve Temel Bileşen Analizi algoritmaları olarak iki farklı özellik çıkarma algoritması kullanılmıştır. Bu algoritmalar ile biyomedikal verilerin en etkin özellikleri elde edilmiş; elde edilen özellikler Destek Vektör Makineleri tabanlı AdaBoost algoritması ile sınıflandırılarak sonuçlar karşılaştırılmıştır. Karşılaştırma için UCI veri tabanından alınan göğüs kanseri, Pima diyabet, karaciğer düzensizliği ve EKG verileri kullanılmıştır.

Anahtar Kelimeler: AdaBoost, Biyomedikal Veri Sınıflandırma, Makine Öğrenmesi, Özellik Çıkarma, Topluluk Sınıflandırıcı

ABSTRACT

MS THESIS

BIOMEDICAL DATA CLASSIFICATION WITH FEATURE EXTRACTION AND SVM BASED ADABOOST ALGORITHM

Mücahid BARSTUĞAN

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN ELECTRICAL AND ELECTRONICS ENGINEERING

Advisor: Asst.Prof.Dr. Rahime CEYLAN

2014, 91 Pages

Jury
Advisor: Asist. Prof. Dr. Rahime CEYLAN
Assoc. Prof. Dr. Seral ÖZŞEN
Asist. Prof. Dr. Nurdan BAYKAN

Machine learning algorithms helps doctors on diagnosis and diagnostics. Support Vector Machines, Neural Networks and AdaBoost algorithms are some machine learning algorithms used for this purpose. AdaBoost is an ensemble classifier which is obtained by combining a number of weak classifiers.

In this study, some experiments were done with feature selection and extraction algorithms to increase the performance of AdaBoost ensemble classifier on biomedical dataset. Thus, Forward Feature Selection and Principal Component Analysis algorithms were used as two different feature extraction algorithms. By these algorithms, the most effective features of biomedical data were obtained, and they were classified with SVM based AdaBoost algorithm. The breast cancer, pima diabetes, liverdisorders and ECG dataset which were taken from UCI were classified and obtained results were compared.

Anahtar Kelimeler: AdaBoost, Biomedical Data Classification, Ensemble Classifier, Feature Extraction, Machine Learning

ÖNSÖZ

Çalışmalarımda yardımlarını esirgemeyen danışmanım Selçuk Üniversitesi Elektrik-Elektronik Mühendisliği öğretim üyesi Yrd. Doç. Dr. Rahime CEYLAN' a ve bölümümüzün tüm değerli öğretim elemanlarına teşekkür ederim.

Mücahid BARSTUĞAN
KONYA-2014

İÇİNDEKİLER

ÖZET **3**

ABSTRACT **5**

ÖNSÖZ **6**

İÇİNDEKİLER **7**

SİMGELER VE KISALTMALAR **9**

1. GİRİŞ **10**

2. KAYNAK ARAŞTIRMASI **11**

3. ÖZELLİK TEMELLİ ANALİZ YÖNTEMLERİ **15**

3.1 Sıralı Özellik Seçimi 15
3.2 Temel Bileşen Analizi 16

4. ÇALIŞMADA KULLLANILAN SINIFLANDIRMA METOTLARI **18**

4.1 Topluluk Sınıflandırıcı: AdaBoost 18
4.1.1 AdaBoost Algoritması 18
4.1.2 İkili AdaBoost Algoritması 20
4.1.3 Temel Analiz 21
4.2 Destek Vektör Makineleri 23

5. GERÇEKLENEN UYGULAMALAR **29**

5.1 Kullanılan Verilerin Özellikleri ve Performans Değerlendirme Kriterleri 29
5.1.1 Kullanılan Verilerin Özellikleri 29
5.1.2 Performans Değerlendirme Kriteri 31
5.2 Geliştirilen Test Yöntemleri 31
5.2.1 Temel Öğrenici ve Zayıf Sınıflandırıcı Parametreleri 31
5.2.2 Her özelliğin en iyi özellik parametreleri kullanılarak test edilmesi 32
5.2.3 Her özelliğin kendi parametreleri kullanılarak test edilmesi 36
5.2.4 En iyi özelliklerin en iyi özellik parametreleri kullanılarak test edilmesi 40
5.3 Ayrık Zamanlı Veriler Üzerinde Gerçeklenen Uygulamalar 43
5.4 Sürekli Zamanlı Veriler Üzerinde Gerçeklenen Uygulamalar 50
5.4.1 EKG Verisini Oluşturan Kısımlar 52
5.4.2 EKG sinyalinin sınıflandırma sonuçları 54

6. SONUÇLAR VE ÖNERİLER **63**

6.1 Sonuçlar 63
6.2 Öneriler 65

KAYNAKLAR **66**

İNTERNET KAYNAKLARI 70

EKLER 71

ÖZGEÇMİŞ 91

SİMGELER VE KISALTMALAR

Simgeler

x	: Giriş verisi
C	: Sınıf etiketi
M	: Alt küme sayısı
D_n	: AdaBoost eğitim seti
T	: Algoritmanın iterasyon süresi
H	: AdaBoost algoritmasında sınıflandırıcı seti
h_t	: AdaBoost algoritmasında zayıf sınıflandırıcı
w	: AdaBoost algoritmasında ağırlıklar
a_t	: AdaBoost algoritmasında zayıf sınıflandırıcı ağırlığı
ε	: AdaBoost algoritmasında sınıflandırma hatası
d_+	: Destek Vektör Makineleri algoritmasında ayırıcı yüzey
d_-	: Destek Vektör Makineleri (DVM) algoritmasında ayırıcı yüzey
Φ	: DVM algoritmasında dönüştürülmüş eğitim örnekler
$\overline{w}$	: DVM algoritmasında hiper düzlemin normali
b	: DVM algoritmasında hiper düzlem skaler değeri
$\|b\|/\|\|\overline{w}\|\|$	: DVM algoritmasında hiper düzlemin orijinden uzaklığı
ξ_i	: DVM algoritmasında slack değişkeni
$K(\overline{x_\iota}, \bar{x})$	: Kernel fonksiyonu

Kısaltmalar

SÖS	: Sıralı Özellik Seçimi
BDBÇG	: Birini-Dışarıda-Bırak Çapraz Geçerliliği
DVM	: Destek Vektör Makineleri
TBA	: Temel Bileşen Analizi
EKG	: Elektrokardiyogram
BKH	: Birine-karşı-hepsi
BBA	: Bağımsız Bileşen Analizi

1. GİRİŞ

Biyomedikal mühendisliği alanında son 50 yıldır yapılan çalışmaların pek çoğu doktora yardımcı, tanı ve teşhisi güçlendirici bir sistem oluşturmayı amaçlar. Bilgisayar destekli tanı (Computer Aided Design - CAD) başlığı altında önerilen sistem ya da yapılar genel olarak yapay zekâ teknikleri kullanır. Biyomedikal problem, bazı durumlarda tetkik sonuçlarından hastalığın tanımlanması olabileceği gibi, bir sinyalin anlamlandırılması da olabilir. Literatürde her iki problemin çözümü için de pek çok sınıflama tekniği önerilmiştir. Bunlardan bazıları Karar Ağacı (Decision Trees), Boosting, Yapay Sinir Ağları (Artificial Neural Networks), Destek Vektör Makineleri (Support Vector Machines) vb. algoritmalardır.

Ayrıca herhangi bir sınıflandırıcının performansını artırmak için önerilen topluluk sınıflandırıcı yapıları ile ilgili uygulamalar da mevcuttur. Örneğin, Naive-Bayes sınıflandırıcıları, AdaBoost, Bagging, Rotational Forest vb. algoritmalar da topluluk sınıflandırıcı algoritmalardır. Bu yapılar birçok zayıf sınıflandırıcının bir araya getirilmesiyle oluşturulmuştur. Temel mantık olarak, birçok zayıf sınıflandırıcı içinden en yüksek performansı gösteren zayıf sınıflandırıcılar bulunur ve bu zayıf sınıflandırıcıların topluluk içerisindeki ağırlığı güncellenerek artırılır. Bu şekilde daha yüksek bir sınıflandırma başarısı elde edilir.

Tez çalışmasında, Destek Vektör Makinesi (DVM) tabanlı AdaBoost topluluk sınıflandırıcı kullanılarak biyomedikal verilerin sınıflandırma başarıları incelenmiştir. Çalışmada başarıyı artırmak için Sıralı Özellik Seçimi (Forward Feature Selection) ve Temel Bileşen Analizi (Principal Component Analysis) algoritmaları kullanılmıştır. Sıralı Özellik Seçimi (SÖS) ve Temel Bileşen Analizi (TBA) algoritmaları ile biyomedikal verilerin boyutu düşürülerek, etkin özellikleri çıkarılmıştır. Bu etkin özellikler kullanılarak biyomedikal verilerin sınıflandırma işlemi yapılmıştır.

Tez çalışmasında, yüksek boyutlu verilerden özellik çıkartılarak boyut düşümü sağlanmış ve sınıflandırma süresi kısaltılmıştır. Boyutu düşürülen veriler yüksek performans elde edilmesi amaçlanan hibrit AdaBoost-Destek Vektör Makineleri yapısı ile sınıflandırılmıştır. İki farklı özellik çıkarma algoritması kullanılarak, aynı hibrit yapı ile elde edilen performans sonuçları birbiri ile karşılaştırılmıştır.

2. KAYNAK ARAŞTIRMASI

Yun ve Reeves, 1998 yılında görüntü geri elde edimi üzerine çalışmıştır. Çalışmada, Sıralı Özellik Seçimi algoritmasına yeni kriterler eklenmiştir. Eklenen bu kriterler sayesinde MRI ve MRSI edinim stratejilerini optimize etmede umut vaat edici sonuçlar elde edilmiştir.

2001 yılında Zhao ve Principe tarafından yapılan çalışmada otomatik hedef tanıma sistemi gerçeklenmiştir. DVM' ler bu çalışmada gerçek bir uygulamada denenmiştir. Eğitilen DVM' lerin, hedef tanımada, geleneksel sınıflandırıcılardan üstün bir başarı sağladığı görülmüştür.

2003 yılında Garg ve ark. temel sınıflandırıcı temelli bir ses tanıma sistemi geliştirmiştir. Bu sistemde danışmansız Bayes ağı, sınıflandırmada zayıf sonuçlar verebileceğinden, AdaBoost algoritması danışmanlı Bayes ağı ile geliştirilmiştir. Önerilen yöntem konuşma tabanlı komut verme ve kontrol ara yüzünde test edilmiştir.

Guo ve Dyer yüz ifadelerini tanıma üzerine çalışmış (2005) ve bir sistem oluşturmuştur. Çalışmada, yüz ifadeleri görüntülerinden özellik çıkartılmış ve bu özellikler ile sınıflandırıcı eğitilmiştir. Basitleştirilmiş Bayes sınıflandırıcı, DVM ve AdaBoost algoritmalarından elde edilen sonuçlar karşılaştırılmıştır.

Lu ve ark. (2007) mikro-dizi gen ifadeleri öğrenmesi sistemi gerçeklemiştir. Önerilen hibrit boyut indirgemesi yönteminde, TBA ile LDA yöntemleri birleştirilerek gen ifadeleri verisi üzerinde uygulama yapılmıştır. Bu esnek yöntem ile karmaşıklık giderilmiştir. Bu yöntem 80 adet mikro-dizi seti ile test edilmiştir.

2007 yılında Chen ve ark. yüz tanıma üzerine bir sistem geliştirmiştir. Çalışmada, MIT ve CMU veri tabanlarından alınan yüz görüntüleri tekrar örneklenerek yüz ve yüz olmayan görüntü olmak üzere ayrılmıştır. Bu görüntüler ile AdaBoost algoritması eğitilmiş ve daha sonra yanlış sınıflandırılan görüntüler ile DVM algoritması eğitilmiştir. Önerilen yöntemin son zamanlardaki yöntemlerden başarılı sonuç verdiği görülmüştür.

Zhou ve ark. tarafından 2007 yılında yapılan çalışmada trafikte hareket eden bir araç tespiti sistemi oluşturulmuştur. Araçlardan alınan görüntülerin TBA ile özellikleri çıkartılmış ve bu özelliklerin araca ait olup olmadığının tespit edilmesi için DVM ile sınıflandırılma yapılmıştır. Önerilen yöntem aydınlık, karanlık, gölge vb. farklı ortamlarda denenmiş ve tatmin edici bir sonuç vermiştir.

Beltrân ve ark. (2008), yaptıkları çalışmada Şili şaraplarını sınıflandırma üzerine bir sistem gerçeklemiştir. Tat verisinin özellikleri TBA ve Wavelet dönüşümü ile çıkarılmıştır. Daha sonra, LDA, Radial Basis Function Neural Network ve DVM' ler gibi örüntü tanıma teknikleri ile sınıflandırma yapılmıştır. Elde edilen sonuçlar karşılaştırıldığında en yüksek sınıflandırma doğruluklarının DVM ile elde edildiği görülmüştür.

Subaşı ve Gürsoy' un çalışmasında (2008) EEG işaretleri üzerinden hastalık tespiti sistemi oluşturulmuştur. Çalışmada, EEG işaretinden öncelikle güç spektrumu elde edilmiş, daha sonra elde edilen özellik vektörlerinin, TBA ve BBA algoritmaları kullanılarak boyutları indirgenmiştir. Elde edilen değerler, DVM ile EEG sinyali epileptik ya da epileptik değil diye sınıflandırılmıştır.

2009 yılında Âlvarez ve ark. Alzheimer hastalığının erken teşhisi üzerine bir çalışma yapmıştır. Çalışmada, SPECT görüntülerinden TBA algoritması ile özellik uzayında boyut indirgemesi yapılmıştır. SPECT görüntülerinden Alzheimer hastalığı tespitinin doğruluğunun artırılması için DVM ile sınıflandırma yapılmıştır.

Jiangtao ve Zhenpu (2010), yaptıkları çalışmada farklı türlerdeki madeni yağları birbirlerinden ayıran sistem tasarlamıştır. BBA ile özellik çıkarımı yapılmış ve bu özellikler DVM ile sınıflandırılmıştır. Dizel yağ, kaba yağ, benzin ve gaz yağı %98,56 gibi bir başarı ile sınıflandırılmıştır.

2010 yılında A. Marcano-Cedeño ve ark. seçim kriterinde tahmini hatayı elde etmek üzerine Sıralı Özellik Seçimi tabanlı bir özellik seçimi yöntemi sunmuşlardır. Çalışmada, bu yöntemi test etmek için şarap, göğüs kanseri ve iris çiçeği verileri kullanılmıştır.

Tao ve ark. 2010 yılında yaptıkları bir çalışmada otomatik yüz tanıma sistemi geliştirmiştir. AdaBoost ile yüz tespiti, Wavelet dönüşümü ve önerilen K-TBA ile özellik çıkarımı yapılmıştır. Bu özellikler daha sonra DVM algoritması ile sınıflandırma işlemine tabi tutulmuştur.

Han (2010), yapmış olduğu çalışmada mikro dizi verilerde kanser moleküler örüntülerini tanıma sistemi gerçeklemiştir. Çalışmada, TBA' dan üstün olan N-TBA algoritması önerilmiştir. Önerilen algoritma ile DVM birleştirilerek mikro dizi örüntü tanıma uygulaması yapılmıştır. NTBA-DVM yapısı ile güçlü sınıflandırma sonuçları elde edilmiştir.

2010 yılında Ye ve ark. tarafından yapılan çalışmada morfolojik ve dinamik tabanlı aritmi sınıflandırmasına yeni bir yaklaşım getirilmiştir. Çalışmada, Wavelet

dönüşümü ve BBA algoritması ile kalp atışlarından özellik çıkartılarak, kalp atışlarının morfolojik özellikleri elde edilmiştir. Daha sonra bu özellikler DVM ile sınıflandırılmıştır. %99,66 gibi yüksek bir başarı elde edilmiştir.

Bui ve ark. (2011) yeni bir bir yüz tanıma sistemi gerçeklemiştir. AT&T ve Feret veri tabanı sitelerinden alınan yüz verilerinin, Gabor filtresi ve TBA algoritması ile özellik vektörleri çıkarılmış, bu özellikler DVM ve AdaBoost algoritmaları ile sınıflandırılmıştır.

Zhang ve ark. (2011) tarafından yapılan çalışmada bilgisayarlı tanımada aktif bir araştırma konusu olan insan hareketi tespiti ve tanımlaması üzerine sistem tasarlanmıştır. Üç aşamadan oluşan BEL (boosted exemplar learning) yöntemi kullanılmıştır. İlk aşamada her bir hareket, farklılıkları ve oluşumları değerlendirilmiş ve optimizasyon formülü ile algoritma eğitilmiştir. İkinci aşamada hareketler, aralarındaki benzerliklere göre tanıtılmıştır. Üçüncü aşamada ise en farklı hareketler özellik seçimi ile seçilmek üzere formülize edilmiştir. K TH ve Welzmann veri tabanından alınan veriler kullanılmıştır.

Kanaan ve ark. (2011) yapmış oldukları çalışmada günümüzde ölüme sebep olan kardiyak problemleri üzerine bir yöntem önermiştir. Bu yöntemde, elektrokardiyograf verilerinden TBA ile özellik çıkarılmış ve bu özellikler DVM ile sınıflandırılmıştır. Çalışmada MIT-BIH Arrhythmia Database' den alınan veriler kullanılmıştır.

Erkılınç ve Şahin' in çalışmasında (2011) parmaklarını kullanamayan engellilerin, bilekleri aracılığı ile kamera yönünü değiştiren joystickleri kullanabilmesi üzerine bir sistem kurulmuştur. Çalışmada, engellilerin beyin sinyallerine Hızlı Fourier Dönüşümü analizi uygulanmış ve TBA ve S-TBA ile sinyallerde özellik çıkarma yapılmıştır. Sinyaller daha sonra DVM algoritması ile teste tabi tutulmuş ve S-TBA ile %81 başarı sağlanmıştır.

Isa ve ark. (2011) EKG sinyali kullanarak uyku apneası tespit eden sistem tasarlamıştır. Uygulamada TBA kullanılmıştır. kNN, Naive Bayes ve DVM ile elde edilen sonuçlar karşılaştırılmıştır.

2011 yılında Tofighi ve Monadjemi tarafından yapılan çalışmada yüz tespiti ve tanıma sistemlerinin performansını artırmak üzere bir yöntem geliştirilmiştir. Çalışma iki aşamadan oluşturulmuştur. İlk olarak, yüzler tespit edilmiş ve daha sonra tanıma yapılmıştır. Tespit aşamasında AdaBoost algoritması ile Gaussian ten rengi modeli birleştirilerek ten rengi ayrımı yöntemi kullanılmıştır. Tanıma aşamasında ise

Gabor filtresi ile özellik çıkarılmış, TBA ile boyut azaltılmış, LDA ile özellik seçilmiş ve DVM ile de sınıflandırma yapılmıştır.

Roychowdhury ve ark. 2012 yılında yaptıkları çalışmada bilgisayar yardımlı görüntüleme sistemi oluşturmuştur. GMM, kNN, DVM ve AdaBoost algoritmaları ile sınıflandırmalar yapılmış ve elde edilen sonuçlar karşılaştırılmıştır. AdaBoost algoritması ile sınıflandırma yapılırken 78 özellikten 30 adedi seçilmiştir. Sistem %100 duyarlılık, %53,16 belirlilik ve %90,4 doğruluk değerlerini elde etmiştir.

Sun ve ark. (2012) yapmış oldukları çalışmada Eigen tabanlı sınıflandırıcı kullanılarak tümör sınıflandırma sistemi kurulmuştur. Önerilen yöntemde BBA algoritması ile özellikler çıkartılmıştır. Bu özellikler zayıf DVM sınıflandırıcıların eğitim verileri olarak kullanılmıştır. Bayes toplam kuralı ile zayıf DVM' lerin sınıflandırma sonuçları toplanarak bir karar verici algoritma ile tümörün sınıfı tespit edilmiştir.

Florea ve ark. (2012) ağız bölgesine bağlı olarak duygu ifadelerini sınıflandıran bir sistem kurmak üzere çalışmışlardır. Kurdukları bu sistemde sınıflandırma algoritmaları olarak AdaBoost ve TBA kullanmışlardır.

Kallas ve ark. 2012 yılında elektrokardiyogram sinyallerinin kullanılması ile kalbin anormal ve normal atışlarını belirleyen bir sistem tasarlamıştır. Özellik çıkarma yöntemi olarak Kernel TBA, sınıflandırma aşamasında iki adet çoklu-DVM yöntemi olan "birine karşı biri" ve "birine karşı hepsi" yöntemleri kullanılmıştır.

2012 yılında Gao ve ark. yaptıkları bir çalışmada EEG sınıflama tabanlı yalan tespit sistemi kurmuştur. EEG sinyallerinden BBA algoritması ile özellik çıkarılmıştır. Özellikler, DVM ile sınıflandırılmış ve %82,45 oranında bir başarı elde edilmiştir.

De Marsico ve ark. (2013) yapmış oldukları çalışmada yüz tanıma sistemi kurmuştur. FACE (Face Analysis for Commercial Entities) adı verilen yeni bir yüz tanıma sistemi önerilmiştir. Örüntüler sınıflandırıcıya sunulmadan önce yüzün aldığı şekil ve ışık değişimleri çıkartılmıştır. Düşük kaliteli görüntüler manuel olarak sınıflandırılmış veya yeni bir görüntü çekilmiştir. Önerilen sistemden elde edilen sonuçlar ile DVM, TBA ve BBA algoritmaları ile elde edilen sonuçlar kıyaslanmıştır. FACE algoritmasının elde ettiği sonuçların diğer algoritmalardan üstün olduğu görülmüştür.

3. ÖZELLİK TEMELLİ ANALİZ YÖNTEMLERİ

Makine öğrenmesi algoritmalarında, sınıflandırma problemini çözmek "bir *x* giriş verisini *C* sınıf etiketine sahip bir sonlu uzay etiketine haritalamak" demektir. Sınıflandırma algoritmaları I={(x^1,c^1), …. (x^n,c^n)} eğitim örnekleri üzerinden eğitilirken, algoritmanın başarısı ayrı bir test seti üzerinde elde edilen genelleştirme hatasına dayanılarak belirlenir. Özellikler; giriş verisi *x*'in ayrı bölümleri (skaler veya vektörel) olarak, f_i özelliği için özellik etiketleri $f_i \in F$ ve özellik değerleri $f_i(x)$ olarak ele alınır. İyi bir sınıflandırma sonucu için önemli adımlardan birisi gürültülü, ihtiyaç fazlası ve yanlış sınıflandırmaya neden olabilecek verileri filtrelemek, yani özellik seçimi yapmaktır. Özellik seçimi, genelleştirme hatasını en aza indiren özellikleri tanımlamaya çalışan bir optimizasyon problemidir. Özellik seçiminin temel amacı, işlem sırasında kullanışsız ve gereksiz olan verilerin sayısını azaltmaktır (Rückstieβ, Osendorfer ve Smagt, 2011).

Tez çalışmasında Sıralı Özellik Seçimi ve Temel Bileşen Analizi algoritmaları etkin özellikleri seçmek ve boyut düşümünü sağlamak için kullanılmıştır.

3.1 Sıralı Özellik Seçimi

Sıralı Özellik Seçimi (SÖS) işlemi sadece bir adet öznitelik içeren tüm özellik alt kümelerinin elde edilmesi ile başlar. Tek bileşenli alt kümeler ($\{X_1\}$, $\{X_2\}$, … ,$\{X_M\}$) Birini-Dışarıda-Bırak Çapraz Geçerliliği (BDBÇG) ile belirlenir. Alt küme sayısı *M*, giriş verisinin boyutudur. Yapılan özellik seçimi işlemi ile en iyi tek özellik $X_{(1)}$ bulunur (Deng, 1999).

3.1.1 Sıralı Özellik Seçimi Algoritması

Sıralı Özellik Seçimi algoritması şu şekildedir (Deng, 1999):

1. Giriş verisi alınır.
2. Veri seti L adet parçaya ayırılır.
3. Her bir parça için *(i = 0, 1, … , L-1)*

a. *Dış Eğitim Seti(i) = i* hariç tüm parçalar olsun.

b. *Dış Test Seti(i) = i*. parça olsun.

c. *İç Eğitim(i) = Dış Eğitim Seti*' nin rastgele seçilen %70' i olsun.

d. *İç Test(i) = Dış Eğitim Seti*' nin seçilmeyen %30' u olsun.

e. j = 0, 1, …, m için

Birini-dışarıda-bırak yöntemini kullanarak *İç Eğitim(i)*' de j bileşen ile en iyi özellik seti $ös_{ij}$ aranır.

$İç\ Test\ Sonucu_{ij}$ = İç Test(i)' de $ös_{ij}$' nin test sonucu elde edilen hata değeridir.

j döngüsünü bitir.

f. En iyi İç Test Sonucu' na sahip $ös_{ij}$ seçilir.

g. $Sistem\ Hatası_i$ = *Dış Test Seti(i)*' den seçilen özellik setinin test sonucu elde edilen hata değeridir.

(i) döngüsünü bitir.

5. Ortalama *Sistem Hatası*' nı elde et.

SÖS algoritması ile *m* özelliğe kadar özellik seçimi işlemi verilmiştir. Bu işlemden sonra ileri seçim, iki bileşen içeren en iyi alt kümeyi bulur. Bu bileşenlerden bir tanesi $X_{(1)}$, diğeri ise geriye kalan *M-1* giriş özniteliğinden seçilen bir özelliktir. Bundan dolayı *M-1* adet çift vardır. En iyi çiftteki $X_{(1)}$' in yanındaki diğer en iyi özellik $X_{(2)}$ olsun. Daha sonra üç, dört veya daha fazla özelliğe sahip alt kümeler elde edilir. İleri seçime göre *m* özellikli en iyi alt küme, *M* adım içinde tüm özellik setlerinin en iyisi olan *m*' li $X_{(1)}, X_{(2)}, \ldots, X_{(m)}$ alt kümesidir (Deng, 1999).

3.2 Temel Bileşen Analizi

Temel Bileşen Analizi (TBA) verideki şablonları tanımlamak, verinin benzerliklerini ve farklılıklarını ifade etmek için kullanılan yöntemlerden birisidir. Yüksek boyutlu verilerin şablonları bulunması zor olduğundan dolayı, şekilsel gösterimin de mümkün olmadığı durumlarda TBA veriyi analiz etmek için kullanılabilecek güçlü bir yöntemdir (Smith, 2002).

TBA' nın bir diğer ana avantajı verideki şablonlar bulunup, fazla bilgi kaybı olmadan, boyut düşürülerek verinin sıkıştırılabilmesidir. Bu sayede TBA görüntü işlemede de kullanılmaktadır (Smith, 2002).

Amaç, bir $x \in R^l$ vektörünün elemanlarını oluşturan, l örnekten (özellik) oluşan bir orijinal set kullanarak, yeni bir özellik seti elde etmek için doğrusal dönüşüm uygulamaktır (Theodoridis ve ark., 2010):

$$y = A^T x$$

Bu işlem sonucunda y'nin elemanlarının birbiriyle ilintisiz olması sağlanır. İkinci adımda ise bu elemanların en etkin olanı seçilir. İşlem aşaması şu şekilde özetlenebilir (Theodoridis ve ark., 2010):

1- Kovaryans matrisi S hesaplanır. Genelde ortalama değer $E[x]=0$ sıfır kabul edilir. Bu durumda, kovaryans ve otokorelasyon matrisleri birbirine denktir:

$$R \equiv E[xx^T] = S.$$

Bu durum haricinde ise ortalama çıkarılır. N özellik vektörü olduğunda $x_i \in R^l$, $i=1,2, \dots ,N$ durumunda otokorelasyon matrisi aşağıdaki gibi elde edilir:

$$R \approx \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T \tag{3.1}$$

2- S' nin özbileşen işlemi gerçeklenir ve l özdeğerleri/özvektörleri hesaplanır. λ_i, $a_i \in R^l$ $i=0,2, \dots ,l$-1.

3- Özdeğerler büyükten küçüğe doğru sıralanır: $\lambda_0 \geq \lambda_1 \geq \dots \geq \lambda_{l-1}$.

4- m adet en büyük özdeğerler seçilir. m genelde λ_{m-1} ve λ_m arasındaki boşluk büyük olacak şekilde seçilir. λ_0, λ_1, … , λ_{m-1} özdeğerleri m adet temel bileşen olarak bilinir.

5- Dönüşüm matrisini oluşturmak için sütun özvektörleri a_i kullanılır.

$$A= [a_0\ a_1\ a_2\ \dots\ a_{m-1}]$$

6- Orijinal uzaydaki her l boyutlu x vektörü $y = A^T x$ dönüşümü kullanılarak m boyutlu y vektörüne dönüştürülür. Başka bir deyişle, $y(i)$' nin i. elemanı x'in a_i üzerindeki yansımasıdır ($y(i) = a_i{}^T x$) (Theodoridis ve ark., 2010).

4. ÇALIŞMADA KULLLANILAN SINIFLANDIRMA METOTLARI

4.1 Topluluk Sınıflandırıcı: AdaBoost

AdaBoost, zayıf sınıflandırıcıları bir araya getirerek güçlü bir sınıflandırıcı oluşturan bir topluluk sınıflandırıcı yöntemidir. Her bir iterasyonda temel öğrenici adı verilen basit bir öğrenme algoritması çağırır ve sınıflandırıcıyı oluşturur. Daha sonra bu bu sınıflandırıcıya ağırlık katsayısını atar. Son sınıflandırma kararı ise zayıf sınıflandırıcıların ağırlık katsayılarına bağlı ağırlıklı oylaması sonucu elde edilir. Zayıf sınıflandırıcı hatası ne kadar düşük olursa, son oylamada ağırlığı o kadar yüksek olur. Zayıf sınıflandırıcıların rastgele tahminden biraz daha iyi tahmin yapmasından dolayı, zayıf sınıflandırıcı setinin tasarımında büyük bir esneklik sağlanır (Kégl, 2009).

AdaBoost tanımlanırken kullanılan bazı parametreler vardır: Eğitim seti parametresi $D_n=\{(x_1,y_1),\ldots,(x_n,y_n)\}$ iken algoritmanın iterasyon süresini belirten parametre *T*' dir. *T* sadece önceden tanımlanmış, elle ayarlanabilir bir AdaBoost parametredir. Her bir $t=1,\ldots,T$ iterasyonunda, sınıflandırıcı seti *H*' den bir $h^{(t)}$ zayıf sınıflandırıcı seçilir ve katsayısı $a^{(t)}$ ayarlanır. Algoritmanın en basit versiyonunda *H*, $h:R^d\rightarrow\{-1,+1\}$ formuna sahip sonlu ikili sınıflandırıcılar setidir ve temel öğrenici her iterasyonda *H* setinde ayrıntılı bir arama yapar. AdaBoost çıkışı zayıf sınıflandırıcıların ağırlıklı oylaması ile oluşturulan bir *diskriminant fonksiyon*dur (Kégl, 2009).

$$f^{(T)} \triangleq \sum_{t=1}^{T} a^t h^t(\cdot) \tag{4.1}$$

$f^{(T)}(x)$ işareti daha sonra x' in son sınıflandırıcısı olarak kullanılmıştır.

4.1.1 AdaBoost Algoritması

Algoritmada Dn eğitim seti, TÖ temel öğrenici, T iterasyon sayısı, n ise örüntü sayısıdır (Kégl, 2009).

1: $w^{(1)} \leftarrow (1/n,\ldots,1/n)$ Ağırlıklara ilk değer ataması yapıldı

2: **for *t*** =1' den ***T***' ye kadar

3: $h^{(t)} \leftarrow$ Temel(D_n, $w^{(t)}$) Zayıf sınıflandırıcı

4: $\varepsilon^{(t)} \leftarrow \sum_{i=1}^{n} w_i^{(t)} ||\{h^{(t)}(x_i) \neq y_i\}$ $h^{(t)}$' ye ait hata

5: $a^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}}\right)$ Zayıf sınıflandırıcının katsayısı

6: **for i** =1' den **n**'ye kadar Eğitim noktaları yeniden ağırlıklandırılıyor

7: **if** $h^{(t)}(x_i) \neq y_i$ **ise** Hata

8: $w_i^{(t+1)} = \frac{w_i^{(t)}}{2\varepsilon^{(t)}}$ Ağırlık artıyor

9: **değilse** Doğru sınıflandırma

10: $w_i^{(t+1)} = \frac{w_i^{(t)}}{2(1-\varepsilon^{(t)})}$ Ağırlık azalıyor

11: döngüyü bitir 6. adımdaki döngü

12: $f^{(T)}(\cdot) = \sum_{t=1}^{T} a^t h^t(\cdot)$ Zayıf sınıflandırıcılar için ağırlıklı oylama

Algoritmadaki ağırlık ile her bir örneğin ağırlığı ifade edilirken, katsayı ile her bir sınıflandırıcının katsayısı ifade edilmektedir. Algoritma, ağırlık noktaları üzerinde bir ağırlık dağılımı oluşturmaktadır. Ağırlıklara 1. adımda ilk değer ataması yapılmış ve 7-10 adımları arasında ağırlıklar her iterasyonda güncellenmiştir. Temel öğrenicinin amacı ağırlıklandırılmış hatayı en aza indirmektedir (Kégl, 2009).

$$\varepsilon(t) \triangleq \sum_{i=1}^{n} w_i^{(t)} ||\{h^{(t)}(x_i) \neq y_i\} \tag{4.2}$$

$h^{(t)}$ zayıf sınıflandırıcının $a^{(t)}$ katsayıları 5. adımda belirlenir.

$$a^{(t)} \triangleq \frac{1}{2} \ln\left(\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}}\right) \tag{4.3}$$

Algoritma boyunca *t* anlarında ağırlıklar normalize edilmiş durumdadır.

$$\sum_{i=1}^{n} w_i^{(t)} = 1 \tag{4.4}$$

Bu ifadenin ispatı için öncelikle $\sum_{i=1}^{n} w_i^{(1)} = 1$ ilk değer ataması yapılır. Daha sonra, $\sum_{i=1}^{n} w_i^{(t)} = 1$ olduğu varsayılarak

$$\sum_{i=1}^{n} w_i^{(t+1)} = \sum_{i=1}^{n} \frac{w_i^{(t)}}{2\varepsilon^{(t)}} ||\{h^{(t)}(x_i) \neq y_i\} + \sum_{i=1}^{n} \frac{w_i^{(t)}}{2(1-\varepsilon^{(t)})} ||\{h^{(t)}(x_i) = y_i\}$$

(4.5)

$$= \frac{1}{2\varepsilon^{(t)}} \sum_{i=1}^{n} w_i^{(t)} ||\{h^{(t)}(x_i) \neq y_i\} + \frac{1}{2(1-\varepsilon^{(t)})} \sum_{i=1}^{n} w_i^{(t)} (1 - ||\{h^{(t)}(x_i) \neq y_i\})$$

(4.6)

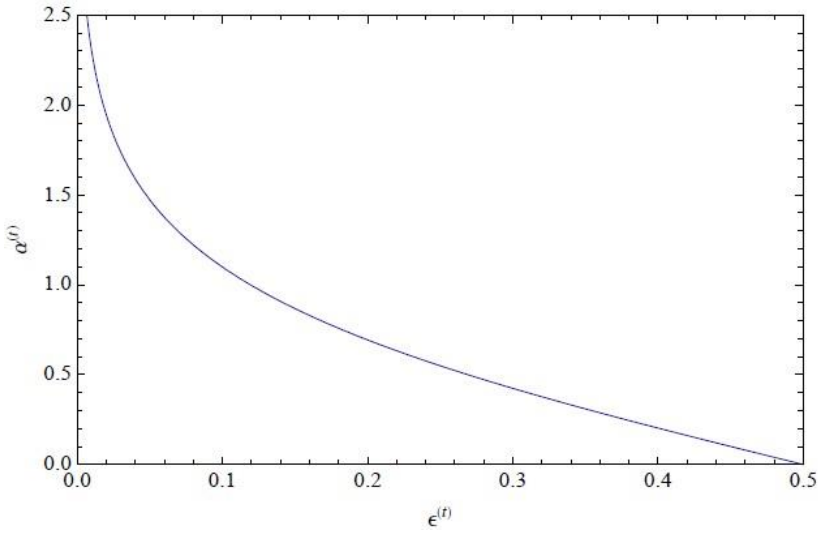
$$= \frac{1}{2\varepsilon^{(t)}} \varepsilon^{(t)} + \frac{1}{2(1-\varepsilon^{(t)})} (1 - \varepsilon^{(t)}) = \frac{1}{2} + \frac{1}{2} = 1. \quad (4.7)$$

bulunur.

(4.5) formülünde 8. ve 10. adımda bulunan ağırlık güncelleme kuralı uygulanmıştır. (4.6)' da ise ||{A} = 1 -||{¬A} ifadesi kullanılmıştır. "¬A" ifadesi A' nın sadece yanlış olduğu durumlarda "doğru" anlamına gelmektedir. (4.7)' de ise (4.2)' deki $\varepsilon^{(t)}$ tanımı ve $\sum_{i=1}^{n} w_i^{(t)} = 1$ tümevarım varsayımı uygulanmıştır. Ağırlıklar normalize edildiği için $\varepsilon^{(t)}$, 0 ve 1 arasında gerçek bir sayıdır. Üstelik belirli bir şart altında ($h \in H$ ise $-h \in H$' dir) H elde edildiğinde $\varepsilon^{(t)} \leq \frac{1}{2}$ olması garanti edilir. Diğer türlü işaret değiştirilir ve $h^{(t)}$ yerine $-h^{(t)}$ kullanılabilir (Kégl, 2009).

4.1.2 İkili AdaBoost Algoritması

$\varepsilon^{(t)} \leq \frac{1}{2}$ ifadesi (4.3) eşitliğinde yerine yazılırsa zayıf sınıflandırıcı katsayısı $a^{(t)}$' nin hiçbir zaman negatif olmayacağı görülebilir. $\varepsilon^{(t)}$ azalırken katsayı monoton olarak artar (Şekil 4.1), böylece zayıf sınıflandırıcı $h^{(t)}$ iyileşir ve son sınıflandırmada daha ağırlıklı bir oya sahip olur. Temel hata $\varepsilon^{(t)}$, 0' a yaklaşırken $a^{(t)}$ katsayısı sonsuza yakınsamaya çalışır. Normalde, $\varepsilon^{(t)} = 0$ olan güçlü bir temel öğrenici otomatik olarak sonsuz bir katsayı alır ve lineer kombinasyonda (4.1) büyük bir etkiye sahip olur. Prensip olarak, AdaBoost'un hedefi eğitim hatasını en aza indirmek olduğundan dolayı, $\varepsilon^{(t)} = 0$ ile hedefe ulaşıldığı anlamına geleceğinden algoritmayı sonlandırmak normal bir durumdur (Kégl, 2009).



Şekil 4.1 Temel hata $\varepsilon^{(t)}$'nin bir fonksiyonu olarak temel katsayı $a^{(t)}$. $\varepsilon^{(t)}= \frac{1}{2}$ iken $a^{(t)} =0$, $a^{(t)} \rightarrow\infty$ iken, $\varepsilon^{(t)} \rightarrow 0$.

$\varepsilon^{(t)} \leq \frac{1}{2}$' nin bir diğer şartı eğitim noktalarındaki w_i ağırlıklarının 7. ve 10. adımlar arasında güncellenmesi, yanlış sınıflandırılan noktaların ağırlıklarının artırılması (8. adım) ve doğru sınıflandırılan noktaların ağırlıklarının azaltılmasıdır (10. adım). Algoritma işleme devam ederken sıklıkla yanlış sınıflandırılan noktaların ağırlıklarının çok büyük olması beklenir, bu yüzden zayıf sınıflandırıcılar zor veri noktalarına daha çok odaklanır (Kégl, 2009).

Son olarak, (4.5) eşitliğinin açılımına bakılırsa, t. zayıf sınıflandırıcı $h^{(t)}$' nin $\varepsilon^{(t+1)}$ temel hatasının $\frac{1}{2}$ olduğu görülebilir. Bundan dolayı, ardışık iki iterasyonda aynı zayıf sınıflandırıcı asla seçilmez (Kégl, 2009).

4.1.3 Temel Analiz

İlk olarak, ağırlık formüllerindeki (7. ve 10. adımlar arası) iki durum birleştirilir:

$$w_i^{(t+1)} = w_i^{(t)} \times \begin{cases} \frac{1}{2\varepsilon^{(t)}} & h^{(t)}(x_i) \neq y_i \; durumunda \\ \frac{1}{2(1-\varepsilon^{(t)})} & h^{(t)}(x_i) = y_i \; durumunda \end{cases} \tag{4.8}$$

$$= w_i^{(t)} \text{ x } \begin{cases} \frac{\sqrt{\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}}}}{2\sqrt{\varepsilon^{(t)}(1-\varepsilon^{(t)})}} & h^{(t)}(x_i) \neq y_i \; durumunda \\ \frac{\sqrt{\frac{\varepsilon^{(t)}}{1-\varepsilon^{(t)}}}}{2\sqrt{\varepsilon^{(t)}(1-\varepsilon^{(t)})}} & h^{(t)}(x_i) = y_i \; durumunda \end{cases}$$

$$= w_i^{(t)} \text{ x } \begin{cases} \frac{\varepsilon^{a^{(t)}}}{2\sqrt{\varepsilon^{(t)}(1-\varepsilon^{(t)})}} & h^{(t)}(x_i) y_i = -1 \; durumunda \\ \frac{\varepsilon^{-a^{(t)}}}{2\sqrt{\varepsilon^{(t)}(1-\varepsilon^{(t)})}} & h^{(t)}(x_i) y_i = 1 \; durumunda \end{cases} \quad (4.9)$$

$$= w_i^{(t)} \; \frac{\exp(-a^{(t)} h^{(t)}(x_i) y_i)}{2\sqrt{\varepsilon^{(t)}(1-\varepsilon^{(t)})}}, \quad (4.10)$$

(4.9) denkleminde a$^{(t)}$ katsayısının (4.3)' teki tanımı ve hem y_i' nin hem de $h^{(t)}(x_i)$' nin {-1,1}-değerli olduğu gerçeği kullanılmıştır. Birleştirilmiş (4.10) denklemi, son diskriminant fonksiyonu $f^{(T)}$' nin (4.1) tanımı, 1. adımdaki $w_i^{(1)}$ ağırlıklarının değişmeyen ilk değer ataması kullanılarak, ağırlıklar özyinelemeli olmayan yöntemle ifade edilebilir. Bu yöntemde kullanılan yapı kendi kendine yinelenmemektedir (Kégl, 2009).

Bu durumda ağırlıklar:

$$w_i^{(t+1)} = w_i^{(t)} \; \frac{\exp(-a^{(t)} h^{(t)}(x_i) y_i)}{2\sqrt{\varepsilon^{(t)}(1-\varepsilon^{(t)})}} = \ldots = \frac{1}{n} \text{ x } \frac{\exp(-y_i \sum_{\tau=1}^{t} a^{(\tau)} h^{(\tau)}(x_i))}{\prod_{\tau=1}^{t} 2\sqrt{\varepsilon^{(t)}(1-\varepsilon^{(t)})}}$$

$$= \frac{\exp(-f^{(t)}(x_i) y_i)}{n \prod_{\tau=1}^{t} 2\sqrt{\varepsilon^{(t)}(1-\varepsilon^{(t)})}}. \quad (4.11)$$

İlk olarak, bu özyinelenmeyen formül, ağırlıkların exp(-$f^{(t)}(x_i)y_i$) ile kesirli olduğunu göstermektedir. Negatif eleman

$$\rho_i^{(t)} \triangleq f^{(t)}(x_i) y_i$$

*normalize edilmemiş marjin*dir, genel bir f diskriminant fonksiyonu ve veri noktası (x, y)
içinde

$$\rho_{x,y}(f) \triangleq f(x)y \tag{4.12}$$

olarak tanımlanmıştır.

Bu değer genelleştirme hatasının marjin-tabanlı analizlerinde önemli bir yer tutmaktadır. Marjin $\rho_i^{(t)}$ işareti (x_i, y_i)' nin $f^{(t)}$ tarafından doğru sınıflandırılıp sınıflandırılmadığını ifade eder. $\rho_i^{(t)}$ ne kadar büyük olursa (x_i, y_i)' nin sınıflandırma sonucundan o kadar emin olunur ve bu yüzden $w_i^{(t+1)}$ ağırlıklarının $\rho_i^{(t)}$ ile monoton olarak azalacaktır (Kégl, 2009).

İkinci olarak, (4.7) denklemi ile $w_i^{(t+1)}$ ağırlıklarının toplamının 1 olduğu bilinmektedir. Buradan, (4.11) formülü kullanılırsa:

$$w_i^{(t+1)} = \frac{\exp(-f^{(t)}(x_i)y_i)}{\sum_{\tau=1}^{n}\exp(-f^{(t)}(x_\tau)y_\tau)}. \tag{4.13}$$

(4.11) ve (4.13) kıyaslanarak, yakınsama teoreminin anahtar denklemi otomatik olarak elde edilir.

$$\frac{1}{n}\sum_{i=1}^{n}\exp(-f^{(t)}(x_i)y_i) = \prod_{\tau=1}^{t} 2\sqrt{\varepsilon^{(t)}(1-\varepsilon^{(t)})} \tag{4.14}$$

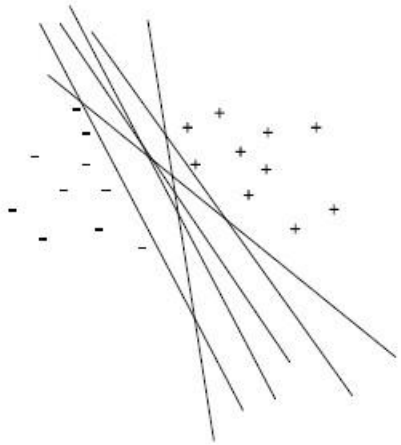
4.2 Destek Vektör Makineleri

Destek Vektör Makineleri (DVM) birçok uygulamada yüksek başarı sağlayan bir öğrenme yöntemidir. DVM' nin dayandığı iki anahtar düşünce vardır. İlk düşünce, özellik vektörlerini doğrusal olmayan bir yöntemle yüksek boyutlu uzaya haritalamak ve bu yeni uzayda doğrusal sınıflandırıcılardan yararlanmaktır. Örneğin, N yeni özellik uzayını, Φ ise haritalamayı ifade etsin. Bu durumda Φ: $R^d \rightarrow N$ olur. $\bar{x} \in R^d$ orijinal bir özellik vektörü için, dönüştürülmüş özellik vektörü $\Phi(\bar{x})$ ile gösterilir. Çıkış etiketi y aynı kalır. Böylece, eğitim örneği $(\bar{x_i}, y_i)$, $(\Phi(\bar{x_i}), y_i)$ olur (Kulkarni ve Harman, 2011).

Daha sonra, dönüştürülmüş N uzayında, dönüştürülmüş eğitim örnekleri ($\Phi(\overline{x_1}), y_1$), … , ($\Phi(\overline{x_n}), y_n$)' i ayıran hiper-düzlem aranır. Bu hiper-düzlemde, dönüştürülmüş $\Phi(\overline{x_i})$ özellik vektörü hiper-düzlemin bir tarafında ise y_i = -1, diğer tarafında ise y_i = +1 olarak etiketlenir.

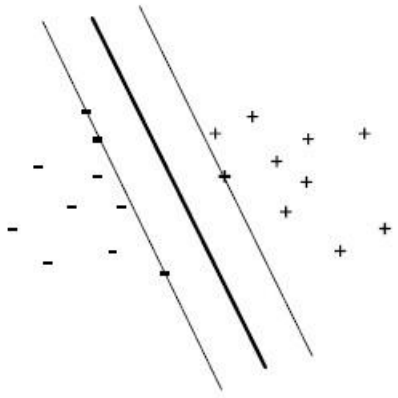
Bu ise, doğrusal sınıflandırıcıların sınırlamalarını aşan, orijinal uzayda doğrusal olmayan sınıflandırıcıları doğurur. Ancak doğrusal sınıflandırıcıların kullanımı, eğitim verisinde iyi bir performans veren sınıflandırıcı bulmada kullanılan hesaplamalara katkıda bulunur. İkinci düşünce, veriyi büyük bir marjinle ayıran hiper-düzlem bulmaya çalışmaktır. Bu düzlem sonsuz sayıdaki düzlemler arasında veriyi olabildiğince iyi ayıran düzlemdir. Birçok düzlem, eğitim verisi üzerinde eşdeğer bir şekilde iyi bir performans gösterirken, yeni veri üzerindeki genelleştirme performansı önemli bir şekilde farklılık gösterebilir (Kulkarni ve Harman, 2011).

Şekil 4.2' de iki boyutun birden fazla çizgi ile ayrılması gösterilmektedir. Tüm bu ayırıcı hiper-düzlemler eğitim verisi üzerinde eşdeğer bir performans göstermektedir, ama birbirinden farklı genelleştirme performansına sahip olabilirler. Yeni örnekler üzerinde ayırıcı düzlemlerin hata oranının birbirinden farklı olması normal bir durumdur.



Şekil 4.2 İki Boyutta Ayırıcı Düzlemler

Ayırıcı hiper-düzlem ile düzleme en kısa uzaklıktaki +1 etiketli veriler arasındaki mesafe d_+ ile ifade edilirken, -1 etiketli veriler arasındaki veriler d_- ile ifade ediliyor olsun. Bu durumda, hiper-düzlemin marjini $d_+ + d_-$ olarak tanımlanabilir. Ayırıcı yüzeyin doğru yönlendirilmesi ile $d_+ + d_-$ olabildiğince büyük yapılabilir. Buna paralel herhangi bir düzlem aynı $d_+ + d_-$ değerine sahip olacaktır (Kulkarni ve Harman, 2011).



Şekil 4.3 Büyük Marjin Ayrımı

Şekil 4.3' te iki boyut gösterilmektedir. $d_+ = d_-$ hiper-düzlemi ve $d_+ = 0$ ile $d_- = 0$ paralel hiper-düzlemler de görülmektedir. Son iki hiper-düzlem eğitim verisinden bir ya da daha fazla örneklerin üzerinden geçmektedir. Bu örneklere *destek vektörleri* adı verilir ve bu örnekler maksimum marjin hiper-düzlemini tanımlamaktadır. Destek vektörleri kaldırılır veya hareket ettirilirse maksimum marjin hiper-düzlemi değişebilir, ancak diğer eğitim örnekleri kaldırılır veya hareket ettirilirse maksimum marjin hiper-düzlemi değişmeksizin kalır (Kulkarni ve Harman, 2011).

Dönüştürülmüş eğitim örnekleri $(\Phi(\overline{x_1}), y_1), \ldots, (\Phi(\overline{x_n}), y_n)$ iken, $\Phi(\overline{x_\iota}) \in N$ ve $y_i \in \{-1,+1\}$ idi. N' deki bir hiper-düzlemin denklemi vektör $\overline{w}$ ve skaler b değerleri ile ifade edilebilir:

$$\overline{w} \cdot \Phi(\bar{x}) + b = 0 \tag{4.15}$$

$\overline{w}$, hiper-düzlemin normali iken; $|b|/||\overline{w}||$, hiper-düzlemin orijinden uzaklığıdır.

Sonlu sayıda eğitim örneği olduğu için, bir hiper-düzlem eğitim örneğini ayırıyor ise her bir eğitim örneği, bazı $\beta > 0$ durumlarında hiper-düzlemden en az β kadar uzak olmalıdır. Daha sonra, ayırıcı hiper-düzlemin 4.15 denklemindeki şartı sağlamasını elde edecek şekilde yeniden normalize işlemi yapılabilir (Kulkarni ve Harman, 2011):

$$\begin{aligned} &\Phi(\overline{x_\iota}) \cdot \overline{w} + b \geq 1, \quad && y_i = +1 \text{ durumunda} \\ &\Phi(\overline{x_\iota}) \cdot \overline{w} + b \leq -1, \quad && y_i = -1 \text{ durumunda} \end{aligned} \tag{4.16}$$

Eşdeğer olacak şekilde, bu şartlar

$$i = 1, \ldots, n \text{ için} \quad y_i \, (\Phi \, (\overline{x_i}) \cdot \overline{w} + b) - 1 \geq 0 \tag{4.17}$$

olarak yazılabilir (Kulkarni ve Harman, 2011).

Eğitim verisinin bir hiper-düzlemle ayrılamayacağı durumlar oluşabilir. Ancak veriyi bu tanımlanan şartlar altında, marjini olabildiğince maksimuma çıkaran hiper-düzlemler aranabilir. Bu amaçla eşitsizlikleri eşitliğe dönüştüren değişken olarak tanımlanan "slack değişkeni" ξ_i kullanılmıştır:

$$i = 1, \ldots, n \text{ için } \xi_i \geq 0$$

$$\begin{aligned} &\Phi \, (\overline{x_i}) \cdot \overline{w} + b \geq +1 - \xi_i, && y_i = +1 \text{ durumunda} \\ &\Phi \, (\overline{x_i}) \cdot \overline{w} + b \leq -1 + \xi_i, && y_i = -1 \text{ durumunda} \end{aligned} \tag{4.18}$$

ξ_i üzerinde sınırlama olmaksızın yukarıdaki şartlar yerine getirilebilir. Uygun bir C sabiti kullanarak $C \sum_i \xi_i$ formunda bir ceza terimi ekleyerek kullanışlı bir formül elde edilebilir. Ayrıca, 4.15 denkleminde yazıldığı gibi, $d_+ = d_- = 1 / \|\overline{w}\|$ durumunda marjin = $d_+ + d_- = 2 / \|\overline{w}\|$ olur (Kulkarni ve Harman, 2011).

Marjinin artırılması için $\|\overline{w}\|$ veya eşdeğer şekilde $\|\overline{w}\|^2$ azaltılabilir. Böylece, aşağıdaki optimizasyon problemini çözmek için bir hiper-düzlem aranır.

$$\begin{aligned} &\text{en aza indir} && \|\overline{w}\|^2 + C \sum_i \xi_i && \\ &\text{bağlı olarak} && \Phi \, (\overline{x_i}) \cdot \overline{w} + b - 1 + \xi_i \geq 0 && i = 1, \ldots, n \text{ için} \\ & && \xi_i \geq 0 && i = 1, \ldots, n \text{ için} \end{aligned} \tag{4.19}$$

Yukarıdaki problemin çözümünü sağlayan hiper-düzlemi bulmak için marjinin artırılması gerekmektedir. Marjin artırılınca hiper-düzlemin normali azaltılır ve her $i = 1, \ldots, n$ değerine bağlı olarak optimizasyon problemine çözüm aranır (Kulkarni ve Harman, 2011).

Optimizasyon teknikleri (Lagrange çarpanları ve dual problem) kullanılarak maksimum marjin ayırıcı hiper-düzlem aşağıdaki optimizasyon problemi çözülerek bulunabilir (Kulkarni ve Harman, 2011).

$$\text{en yükseğe çıkar} \qquad \sum_i a_i - \frac{1}{2} \sum_{i,j} a_i a_j y_i \, y_j \, (\Phi \, (\overline{x_i}) \cdot \Phi \, (\overline{x_j}))$$

bağlı olarak $\sum_i a_i y_i = 0$

$a_i \geq 0$ $i = 1, \ldots, n$ için (4.20)

Bu, iyi bilinen ve verimli algoritmalar için kullanılan dışbükey karesel programlama problemi adı verilen standart bir optimizasyon problemi türüdür. Bu optimizasyonu çözerek hiper-düzlemin denklemlerini veren aşağıdaki eşitlikler sağlanır(Kulkarni ve Harman, 2011).

$$\bar{w} = \sum_{i=1}^{n} a_i y_i \Phi(\bar{x}_i) \quad (4.21)$$

$$a_i(y_i(\bar{w} \cdot \Phi(\bar{x}_i) + b) - 1) = 0 \qquad i = 1, \ldots, n \text{ için} \quad (4.22)$$

$$y_i(\bar{w} \cdot \Phi(\bar{x}_i) + b) - 1) \geq 0 \qquad i = 1, \ldots, n \text{ için} \quad (4.23)$$

Optimizasyon problemi a_i ve ξ_i değerlerinin sonucunu verir. Bu değerler kullanılarak, 4.16 denklemindeki $\bar{w}$ doğrudan elde edilebilir. *b* skaler değeri ise herhangi bir *i* değeri için 4.17 denkleminden elde edilebilir. Buna rağmen, bu denklemi tüm *i* değerleri için kullanmak ve *b* değerini seçmek için, elde edilen değerlerin ortalamasını almak daha iyi bir yaklaşımdır (Kulkarni ve Harman, 2011).

Optimizasyon problemini çözmek ve hiper-düzlemi elde etmek eğitim aşamasıdır. Sınıflandırma için özellik vektörü $\bar{x}$' in hiper-düzlemin hangi tarafına düştüğünün kontrol edilmesi yeterlidir. Buradan

$$\bar{w} \cdot \Phi(\bar{x}) + b = \sum_{i=1}^{n} a_i y_i\big(\Phi(\bar{x}_i).\Phi(\bar{x})\big) + b > 0 \quad (4.24)$$

olursa $\bar{x}$, 1 olarak, diğer türlü -1 olarak sınıflandırılmaktadır (Kulkarni ve Harman, 2011).

Bazı *i* değerleri için a_i=0 olmaktadır. Bu *i* değerleri için, ilgili örnek $(\Phi(\bar{x}_i), y_i)$ maksimum marjin hiper-düzlemini etkilememektedir. Diğer *i* değerlerinde $a_i > 0$ olduğu için bu *i* değerlerine bağlı örnekler maksimum marjin hiper-düzlemini etkilemektedir. Bu örnekler (a_i ' nin pozitif değerlerine bağlı olanlar) destek vektörlerdir (Kulkarni ve Harman, 2011).

$\Phi(\cdot)$ dönüşümünü uygulamak önemli bir uygulama noktasıdır. Orijinal özellik vektörü $\bar{x}$ sıklıkla kendiliğinden yüksek boyutludur ve dönüştürülmüş uzay daha yüksek boyuta, muhtemelen sonsuz sayıda boyuta sahiptir. Bundan dolayı $\Phi(\bar{x}_i)$ ve $\Phi(\bar{x})$'in hesabı çok zor olabilmektedir. Bu sonuç, DVM' ler ile Kernel yöntemi arasında bir

bağlantı sağlamaktadır. Belli şartlar altında $\Phi(\bar{x_\iota}) \cdot \Phi(\bar{x})$ iç çarpımı hesaplaması kolay olan $K(\bar{x_\iota}, \bar{x})$ kernel fonksiyonu ile yer değiştirilebilir. Özellik vektörü $\bar{x}$ ise

$$\begin{aligned} \bar{w} \cdot \Phi(\bar{x}) + b &= \sum_{i=1}^{n} a_i\, y_i\big(\Phi(\bar{x_\iota}).\Phi(\bar{x})\big) + b \\ &= \sum_{i=1}^{n} a_i\, y_i K(\bar{x_\iota}, \bar{x}) + b \end{aligned} \tag{4.25}$$

ifadesinin sıfırdan büyük ya da küçük olmasına göre sınıflandırılır. a_i' yi bulmak için kullanılan optimizasyon problemindeki ifadeler, ayrıca, $K(\bar{x_\iota}, \bar{x})$ ile yer değiştirebilen $\Phi(\bar{x_\iota}) \cdot \Phi(\bar{x_J})$ iç çarpımı da içermektedir. Pratikte $\Phi(\cdot)$'nin haritalanması sırasında genellikle Kernel fonksiyonu *K* doğrudan seçilir ve dönüştürülmüş *N* uzayı *K* seçimi ile elde edilir. Aslında, bir Kernel belirlendikten sonra, *Φ* ve *M* ile ilgili olan durumların bilinmesine gerek kalmadan eğitim ve sınıflandırma kuralları uygulanabilir (Kulkarni ve Harman, 2011).

5. GERÇEKLENEN UYGULAMALAR

Bu çalışmada, biyomedikal verilerin sınıflandırılması için DVM temel öğrenicili AdaBoost sınıflandırıcısı önerilmiştir. Özellik Seçme algoritmaları ile en baskın özellikleri seçilerek boyutu azaltılan veriler, DVM tabanlı AdaBoost sınıflandırıcı yapı ile sınıflandırılmıştır. Çalışmada, biyomedikal verilere Sıralı Özellik Seçimi ve Temel Bileşen Analizi olmak üzere iki özellik seçme algoritması uygulanmıştır. Bütün veriler ve bütün uygulamalarda çapraz geçerlilik yöntemi kullanılmıştır. Doğruluk, belirlilik ve duyarlılık sonuçları elde edilerek algoritmanın performansı incelenmiştir.

Kullanılan sınıflandırıcı AdaBoost algoritmasında temel sınıflandırıcı ağırlığı, katsayılar, zayıf sınıflandırıcı hatası vb. parametreler vardır. Bu parametreler her bir özelliğe bağlı olarak değişmektedir. Bundan dolayı, sınıflandırıcının eğitim aşaması tamamlandıktan sonra test aşamasında üç farklı yöntem izlenmiştir:

1- Her özellik, en iyi özelliğin parametreleri kullanılarak test edildi.
2- Her özellik kendi parametreleri kullanılarak test edildi.
3- Seçilen en iyi özellikler kullanılarak en iyi özellikler test edildi.

Birinci ve üçüncü yöntemde en iyi özellik parametreleri kullanılarak test işlemi yapılmıştır. Ancak aralarındaki fark, birinci yöntemin test aşamasında, test verisindeki bütün özellikler test edilmiş iken, üçüncü yöntemin test aşamasında, test verisinde sadece en iyi özellikler test edilmiştir.

Bu üç farklı yöntem kullanılarak veriler sınıflandırılmış ve en yüksek performansı gösteren yöntem elde edilmiştir.

5.1 Kullanılan Verilerin Özellikleri ve Performans Değerlendirme Kriterleri

5.1.1 Kullanılan Verilerin Özellikleri

Wisconsin Wisconsin göğüs kanseri verisi 683 örüntü, 9 özellikten oluşmaktadır. İki sınıflıdır. 444 örüntü sağlıklı, 239 örüntü hasta sınıfına aittir. Çizelge 5.1’ de verinin özellikleri verilmiştir. Bu 9 özellik kanser hücresine ait bilgilerden oluşmaktadır.

Çizelge 5.1 Wisconsin Göğüs Kanseri Verisinin Özellikleri

1	Küme kalınlığı
2	Hücre büyüklüğü
3	Hücre şekli
4	Sınır yapışması
5	Tekli epitel hücre büyüklüğü
6	Saf çekirdek
7	Yumuşak kromatin
8	Normal çekirdek
9	Mitoz

Karaciğer düzensizliği verisi 345 örüntü, 6 özellikten oluşmaktadır. İki sınıflıdır. Çizelge 5.2' de verinin özellikleri verilmiştir.

Çizelge 5.2 Karaciğer Düzensizliği Verisinin Özellikleri

1	MCV ortalama eritrosit hacmi
2	Alkphos alkalin fosfataz
3	Sgpt alamin aminotransferaz
4	Aspartate aminotransferaz
5	Gammagt gama-glutamil transpeptidaz
6	Alkollü içeceklerin yarım litrelik eşdeğerlerinin günlük içim sayısı

Pima diyabet verisi 768 örüntü, 8 özellikten oluşmaktadır. İki sınıflıdır. İlk 7 özellik hastalığa, 8. özellik yaşa aittir. Çizelge 5.3' de verinin özellikleri görülmektedir.

Çizelge 5.3 Pima Diyabet Verisinin Özellikleri

1	Hamile kalma sayısı
2	İki saatlik oral glikoz tolerans testindeki plazma glükoz konsantrasyonu
3	Diastolik kan basıncı (mm Hg)
4	Üç başlı kas deri kıvrım kalınlığı (mm)
5	İki saatlik serum insülin verimi (mu U/ml)
6	Vücut kitle endeksi (kg/ (boy)m^2)
7	Diyabet soyağacı fonksiyonu
8	Yaş

5.1.2 Performans Değerlendirme Kriteri

Karmaşıklık matrisi performans değerlendirmek için kullanılan yöntemlerden birisidir. Karmaşıklık matrisi TP, TN, FP ve FN değerlerini içeren bir matristir. Bu değerler kullanılarak bir biyomedikal verinin sınıflandırılma işleminin duyarlılık, belirlilik ve doğruluk değerleri elde edilebilir. Burada duyarlılık, sağlıklı sınıfında olan bir verinin ne kadar doğrulukla sağlıklı olarak sınıflandırıldığını, belirlilik ise hasta sınıfında olan bir verinin ne kadar doğrulukla hasta olarak sınıflandırıldığını belirtmektedir. Doğruluk ise genel sınıflandırma başarısını ifade etmektedir (Xie & Minn, 2012).

TP sağlıklı sınıfında olan örüntülerin kaç tanesinin sağlıklı olarak sınıflandırıldığını, TN ise hasta sınıfında olan örüntülerin kaç tanesinin hasta olarak sınıflandırıldığını ifade etmektedir. FN sağlıklı sınıfında olan örüntülerin kaç tanesinin hasta olarak sınıflandırıldığını ifade ederken, FP hasta sınıfında olan örüntülerin kaç tanesinin sağlıklı olarak sınıflandırıldığını ifade etmektedir (Xie & Minn, 2012). Çizelge 5.4' te karmaşıklık matrisinin yapısı verilmiştir.

$$\text{Duyarlılık} = TP / (TP + FN) \tag{5.1}$$

$$\text{Belirlilik} = TN / (TN + FP) \tag{5.2}$$

$$\text{Doğruluk} = (TP + TN) / (TP + TN + FP + FN) \tag{5.3}$$

Çizelge 5.4 Karmaşıklık Matrisi

Veri		Asıl Sınıf	
		Sağlıklı	Hasta
Tahmin Edilen Sınıf	Sağlıklı	TP	FP
	Hasta	FN	TN

5.2 Geliştirilen Test Yöntemleri

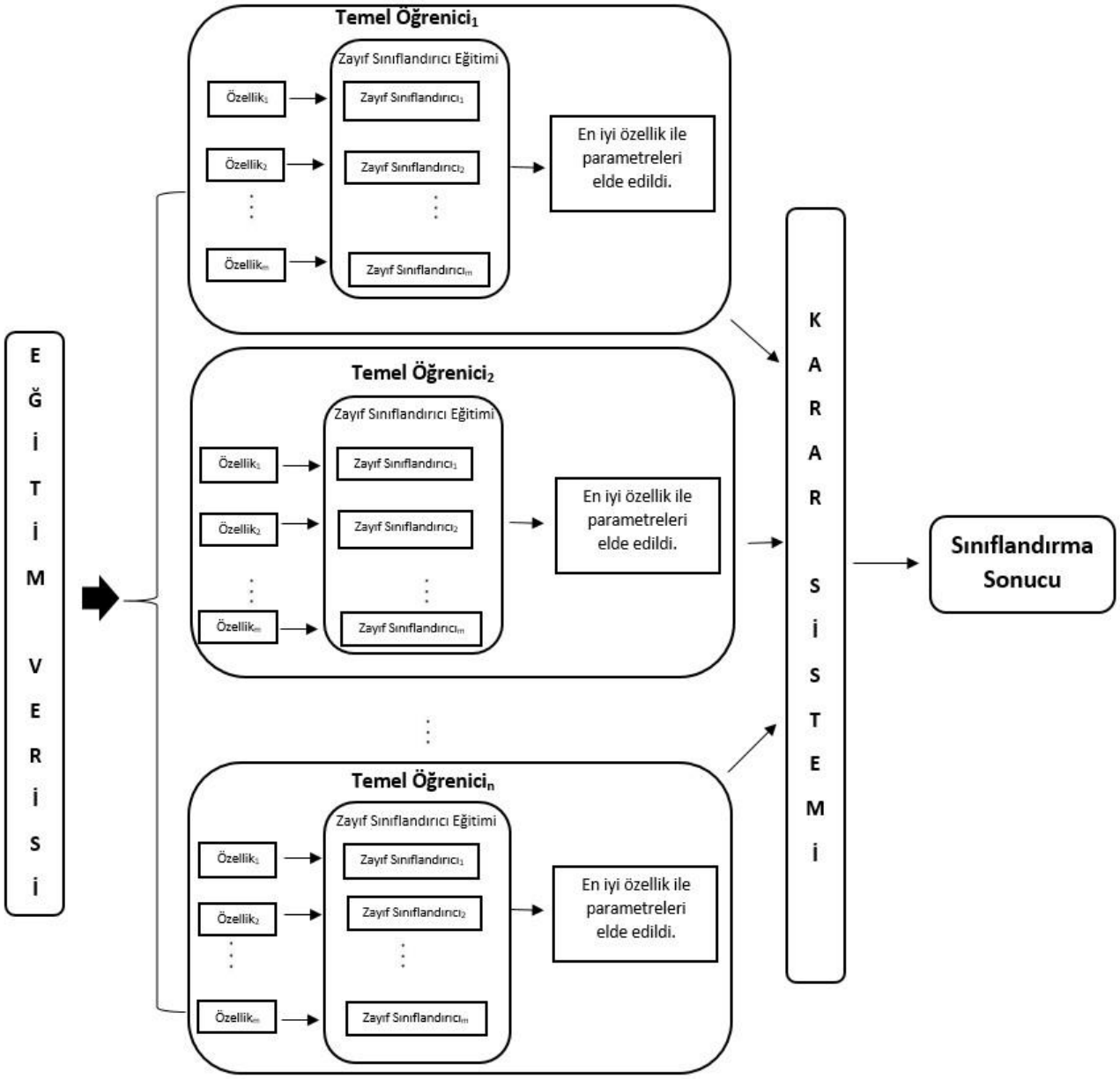
5.2.1 Temel Öğrenici ve Zayıf Sınıflandırıcı Parametreleri

AdaBoost, temel öğrenici ve bu temel öğrenici içerisinde bulunan zayıf sınıflandırıcılar bazı parametrelere sahiptir. AdaBoost için kullanılan en önemli

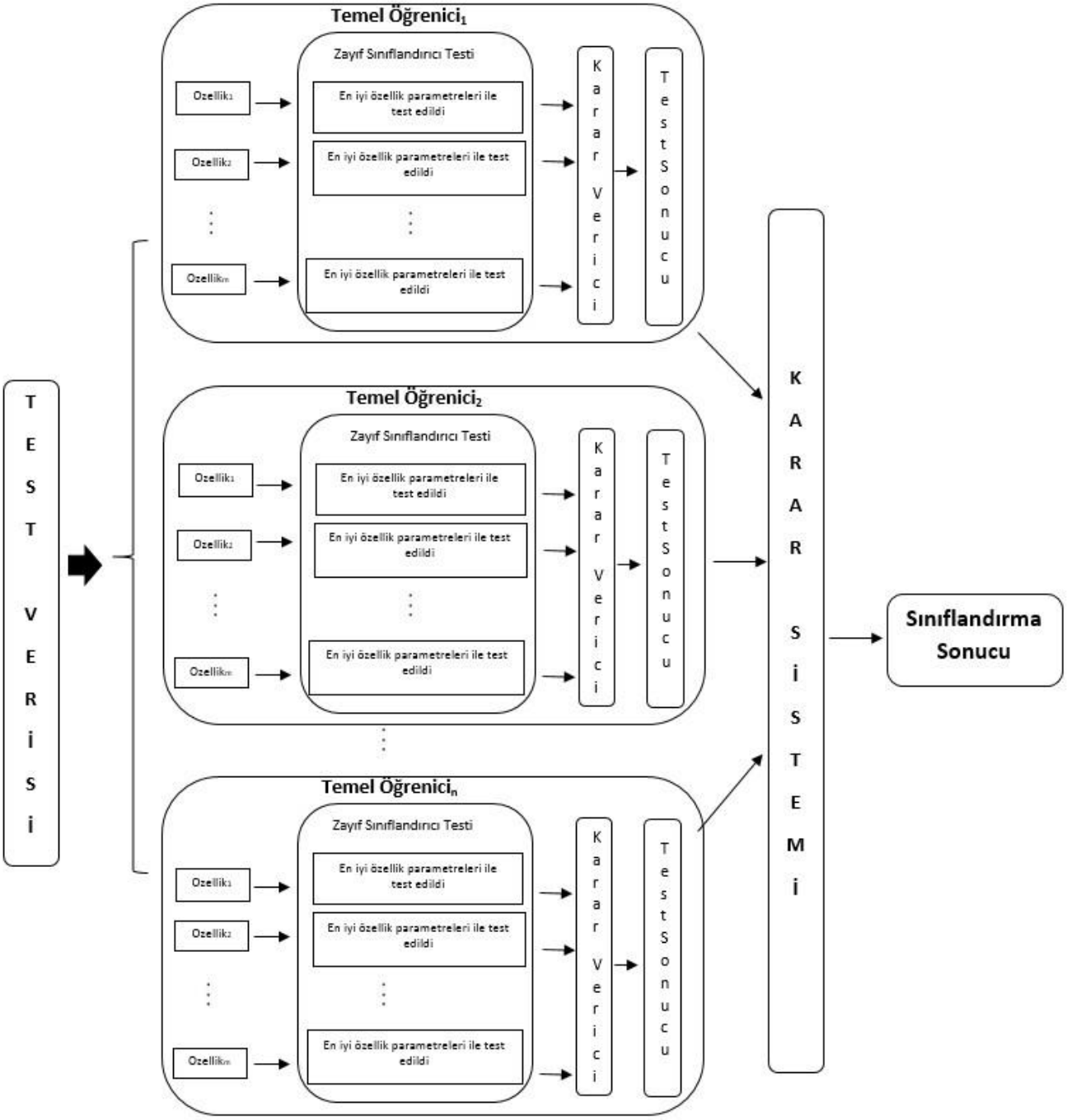
parametre temel öğrenici sayısı iken; temel öğrenici parametresi temel öğrenici ağırlığı, zayıf sınıflandırıcıların parametreleri ise her bir zayıf sınıflandırıcının hatasıdır. Bu parametrelerden temel öğrenici sayısı algoritmada elle değiştirilebilirken temel öğrenici ağırlığı ve zayıf sınıflandırıcı hatası algoritma tarafından elde edilmektedir. Adaboost sınıflayıcı yapı içindeki temel öğrenici sayısı ve her bir temel öğrenici içerisinde bulunan zayıf sınıflandırıcı sayısı elle değiştirilebilen bir parametredir. Sınıflama yapılırken kullanılan üç yöntemde de bu parametreler esas alınmıştır. Bu yöntemlerde en iyi özellik zayıf sınıflandırıcı hatası ve temel öğrenici ağırlığı en iyi özelliği ifade eden parametreler olarak kullanılmıştır. Zayıf sınıflandırıcı olarak kullanılan DVM' nin parametreleri düzlem normali olan ağırlık vektörü $\overline{w}$ ve biası b' dir. DVM' nin C değeri ise "1" alınmıştır.

5.2.2 Her özelliğin en iyi özellik parametreleri kullanılarak test edilmesi

Bu yöntemde, DVM tabanlı AdaBoost sınıflandırıcı, eğitim verisi ile eğitilir. Eğitme işleminde ilk olarak birinci örüntünün her özelliği birinci temel öğreniciye ait zayıf sınıflandırıcılar ile sınıflandırılır. Her temel öğrenici içinde özellik sayısı kadar zayıf sınıflandırıcı bulunur. 1. özellik 1. zayıf sınıflandırıcı ile, 2. özellik 2. zayıf sınıflandırıcı ile sınıflandırılacak şekilde bu işlem bütün özellikler için yapılır. Bu işlem bütün örüntüler kullanılarak tekrarlanır. Daha sonra diğer temel öğreniciler için aynı işlem tekrarlanır. Eğitme işlemi sonucunda en az hatayı veren zayıf sınıflandırıcı, başka bir deyişle en etkin özellik elde edilir. Şekil 5.1' de eğitim akış şeması verilmiştir. Bu özellik ve zayıf sınıflandırıcının parametreleri yani zayıf sınıflandırıcı hatası ve temel öğrenici ağırlığı bir değişkende tutulur. Karar sisteminde eğitme işlemi sonucunda elde edilen parametreler kullanılarak eğitim verileri için sistem cevabı üretilir. Burada sınıflandırma yapılırken en iyi özelliğin parametreleri kullanıldığına dikkat edilmelidir. İlk örüntü için her bir özelliğin sınıflandırma sonucu alınır. m adet özelliğin -1 ve +1 sınıflandırma sonuçlarına bakılarak ilk örüntünün hangi sınıfa ait olduğuna karar verilir. Örüntü, özellik sayısının yarısından fazla +1 sınıfı içeriyorsa +1, -1 sınıfı içeriyorsa -1 olarak sınıflandırılır. Daha sonra temel öğrenicinin ağırlığı ile sınıflandırma sonucu çarpılarak toplanır ve her bir örüntü için sınıflama sonucu elde edilir. Bu işlem bütün örüntüler için tekrarlanır. Test aşamasında ise aynı şekilde bütün özellikler, bu özelliği en etkin özellik bulan zayıf sınıflandırıcının parametreleri ile test edilerek sınıflandırılır. Şekil 5.2' de ise test akış şeması verilmiştir.



Şekil 5.1 Birinci yöntemin eğitim aşaması



Şekil 5.2 Birinci yöntemin test aşaması

Çizelge 5.5’ de göğüs kanseri verisi için birinci yöntem ile elde edilen sonuçlar verilmiştir. Bütün çizelgelerde sunulan sonuçlar 10 kat çapraz geçerlilik yöntemi ile AdaBoost-DVM yapısından elde edilmiştir.

Çizelge 5.5 Birinci yöntem ile elde edilen göğüs kanseri verisi sınıflandırma sonuçları

TÖ Sayısı	**4**	8	12	16	20	24
Duyarlılık	**98,70**	99,12	99,73	99,72	98,90	99,40
Belirlilik	**73,78**	63	67,56	54,83	69,03	60,34
Doğruluk	**90,14**	86,58	88,55	84,30	88,60	85,90
Süre (sn)	**12,66**	25	56,36	121,35	268,28	653,87

Temel öğrenici sayısı 4 ile 24 arasında değiştirilmiş ve en yüksek performans 4 temel öğrenici kullanıldığında elde edilmiştir. Temel öğrenici sayısı 4 iken duyarlılık; sağlıklı veriyi sağlıklı olarak sınıflandırma oranı %98,70' dir. Belirlilik; hasta veriyi hasta olarak sınıflandırma oranı %73,78' dir. Doğruluk; genel sınıflandırma oranı ise %90,14' tür.

Çizelge 5.6' da Pima diyabet verisi için bu yöntem ile elde edilen sonuçlar verilmiştir.

Çizelge 5.6 Birinci yöntem ile elde edilen pima diyabet verisi sınıflandırma sonuçları

TÖ Sayısı	**4**	8	12	16	20	24
Duyarlılık	**0**	0	0	0	0	0
Belirlilik	**100**	100	100	100	100	100
Doğruluk	**65,74**	65,91	65,69	65,91	65,69	65,7439
Süre (sn)	**33,84**	66,7	142,34	306,72	635,2	1320,14

Pima diyabet verisi sınıflandırılırken en yüksek performans 4 temel öğrenici kullanıldığında elde edilmiştir. 4 temel öğrenici ile sınıflama işlemi yapıldığında duyarlılık %0 iken belirlilik %100 çıkmıştır. Doğruluk oranı ise %65,74 bulunmuştur. Burada sınıflandırıcı yapı sağlıklı veriyi hiçbir şekilde sınıflayamamıştır.

Karaciğer düzensizliği verisi için bu yöntem ile elde edilen sonuçlar Çizelge 5.7' de verilmiştir.

Çizelge 5.7 Birinci yöntem ile elde edilen karaciğer düzensizliği verisi sınıflandırma sonuçları

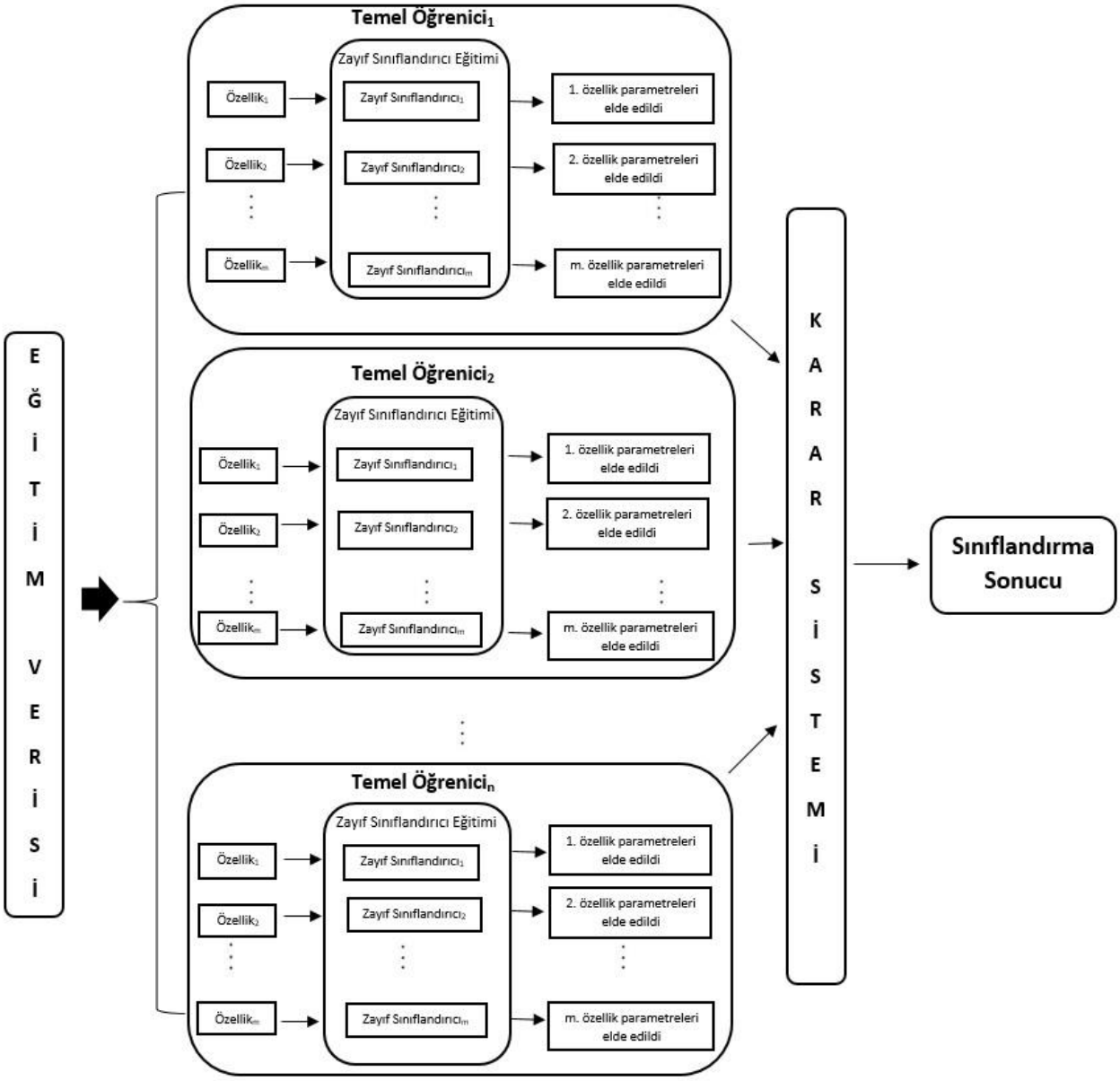
TÖ Sayısı	4	8	**12**	16	20	24
Duyarlılık	7,73	1,53	**2,71**	2,91	5,63	1,90
Belirlilik	95,59	94,96	**97,97**	96,65	95,53	91,79
Doğruluk	58,30	54,65	**58,70**	56,03	56,72	55,1
Süre (sn)	9,87	17,26	**25,79**	48,23	98,8	192,36

Karaciğer düzensizliği verisi için en yüksek performans 12 temel öğrenici kullanıldığında elde edilmiştir. Göğüs kanseri ve pima diyabet verilerinde ise en yüksek başarı 4 temel öğrenici kullanıldığında elde edilmişti. Karaciğer düzensizliği verisinde 4

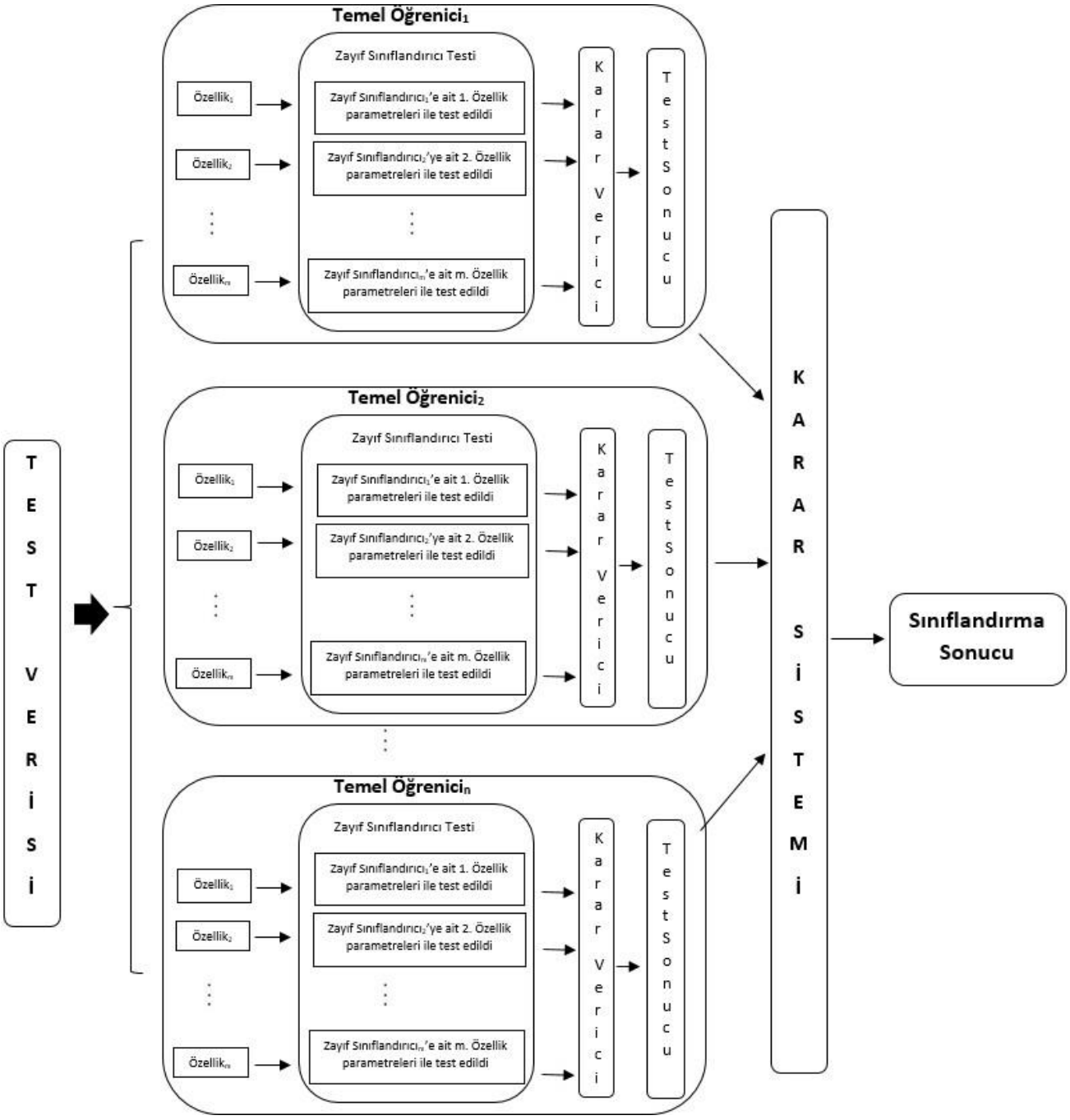
temel öğrenici kullanıldığında sınıflayıcı yapı, 12 temel öğrenici kullanıldığında elde edilen başarıdan daha düşük bir başarı vermiştir. Temel öğrenici sayısı 12 iken sağlıklı veriyi sağlıklı olarak sınıflandırma oranı %2,71 iken hasta veriyi hasta olarak sınıflandırma oranı %97,97' dir. Genel sınıflandırma oranı ise %58,70' tir. Burada sınıflandırıcı yapı sağlıklı veriyi çok düşük bir oranda sınıflandırmıştır.

5.2.3 Her özelliğin kendi parametreleri kullanılarak test edilmesi

Bu yöntemde, DVM tabanlı AdaBoost sınıflandırıcı, eğitim verisi kullanılarak eğitme işlemine alınır. Eğitme işleminde ilk olarak birinci örüntünün her özelliği birinci temel öğreniciye ait zayıf sınıflandırıcılar ile sınıflandırılır. Her temel öğrenici içinde özellik sayısı kadar zayıf sınıflandırıcı bulunur. 1. özellik 1. zayıf sınıflandırıcı ile, 2. özellik 2. zayıf sınıflandırıcı ile sınıflandırılacak şekilde bu işlem bütün özellikler için yapılır. Daha sonra diğer temel öğreniciler için aynı işlem tekrarlanır. Eğitme işlemi sonucunda zayıf sınıflandırıcı hatası ve temel öğrenici ağırlığı parametreleri elde edilir. Şekil 5.3' te bu yöntemin eğitim akış şeması verilmiştir. Karar sistemi uygulanırken, eğitme işlemi sonucunda elde edilen parametrelere eğitim verileri giriş olarak verilir. Burada sınıflandırma yapılırken her özelliğin kendi parametresi kullanılır. İlk örüntü için her bir özelliğin sınıflandırma sonucu alınır. m adet özelliğin -1 ve +1 sınıflandırma sonuçlarına bakılarak ilk örüntünün hangi sınıfa ait olduğuna karar verilir. Örüntü, özellik sayısının yarısından fazla +1 sınıfı içeriyorsa +1, -1 sınıfı içeriyorsa -1 olarak sınıflandırılır. Daha sonra temel öğrenicinin ağırlığı ile sınıflandırma sonucu çarpılarak toplanır ve tek bir sınıflandırma sonucu elde edilir. Bu işlem bütün örüntüler için tekrarlanır. Test aşamasında ise aynı şekilde bütün özellikler, kendi parametreleri ile test edilerek sınıflandırılmıştır. Şekil 5.4' te ise test akış şeması görülmektedir.



Şekil 5.3 İkinci yöntemin eğitim aşaması



Şekil 5.4 İkinci yöntemin test aşaması

Çizelge 5.8' de göğüs kanseri verisi için bu yöntem ile elde edilen sonuçlar verilmiştir. Bütün çizelgelerde 10 kat çapraz geçerlilik ile sonuçlar elde edilmiştir.

Çizelge 5.8 İkinci yöntem ile elde edilen göğüs kanseri verisi sınıflandırma sonuçları

TÖ Sayısı	4	8	**12**	16	20	24
Duyarlılık	97,75	97,52	**98,64**	97,97	97,96	98,86
Belirlilik	97,46	97,44	**96,55**	94,92	96,21	96,25
Doğruluk	97,64	97,5	**97,94**	96,91	97,35	97,94
Süre (sn)	12,54	22,79	**45,21**	82,14	178,25	366,41

Temel öğrenici sayısı 4 ile 24 arasında değiştirilmiş ve en yüksek performans 12 temel öğrenici kullanıldığında elde edilmiştir. 12 temel öğrenici ile duyarlılık %98,64 bulunmuşken, belirlilik %96,55 bulunmuştur. Doğruluk oranı ise %97,94 olarak bulunmuştur.

Çizelge 5.9' da Pima diyabet verisi için bu yöntem ile elde edilen sonuçlar verilmiştir.

Çizelge 5.9 İkinci yöntem ile elde edilen pima diyabet verisi sınıflandırma sonuçları

TÖ Sayısı	4	8	**12**	16	20	24
Duyarlılık	69,2	66,8	**66,8**	66,6	65,6	68,8
Belirlilik	71,53	66,53	**76,92**	69,61	68,84	70,76
Doğruluk	70	66,71	**70,26**	67,63	66,71	69,47
Süre (sn)	32,95	68,39	**142,32**	302,58	652,14	1451,69

Pima diabet verisinde de aynı şekilde en yüksek performans 12 temel öğrenici ile elde edilmiştir. Bu durumda sağlıklı veriyi sağlıklı olarak sınıflandırma oranı %66,8' dir. Hasta veriyi hasta olarak sınıflandırma oranı %76,92 iken genel sınıflandırma oranı ise %70,26'dır.

Çizelge 5.10' da karaciğer düzensizliği verisi için bu yöntem ile elde edilen sonuçlar verilmiştir.

Çizelge 5.10 İkinci yöntem ile elde edilen karaciğer düzensizliği verisi sınıflandırma sonuçları

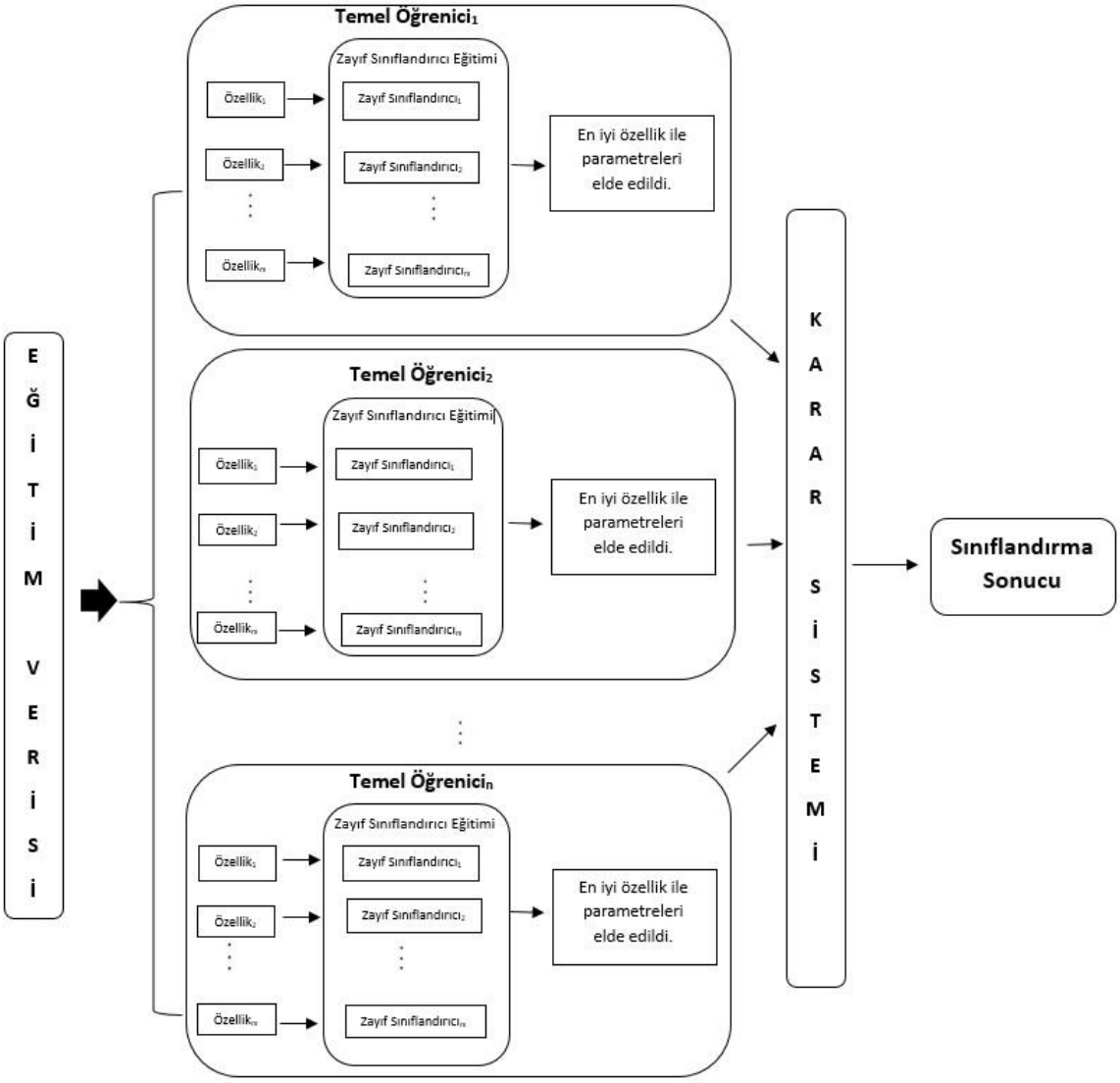
TÖ Sayısı	4	8	12	**16**	20	24
Duyarlılık	47,98	49,50	40,02	**46,42**	36,75	39,45
Belirlilik	71,40	73,54	82,20	**84,21**	71,89	69,74
Doğruluk	62,84	62,15	64,05	**69,71**	59,91	58,03
Süre (sn)	9,16	17,72	26,02	**41,15**	78,32	141,49

Karaciğer düzensizliği verisinde en yüksek başarı 16 temel öğrenici kullanıldığında elde edilmiştir. Göğüs kanseri ve pima diyabet verilerinde olduğu gibi 12 temel öğrenici kullanıldığında elde edilen başarı en yüksek ikinci sonucu vermektedir ancak performans yaklaşık %5 oranında düşüktür. Temel öğrenici sayısı 16 iken duyarlılık %46,42 iken belirlilik %84,21' dir. Doğruluk oranı ise %69,71 olarak bulunmuştur.

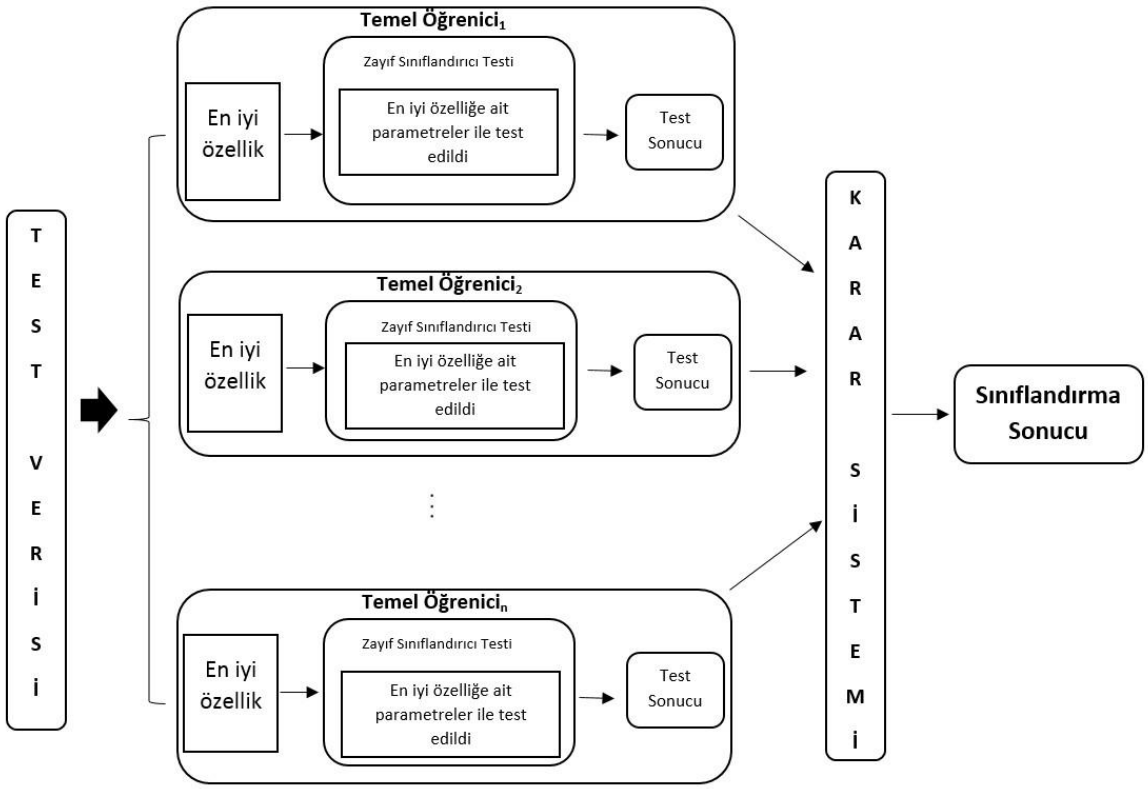
Elde edilen sonuçlara bakıldığında ikinci yöntem birinci yöntemden üstün çıkmıştır.

5.2.4 En iyi özelliklerin en iyi özellik parametreleri kullanılarak test edilmesi

Bu yöntemde, DVM tabanlı AdaBoost sınıflandırıcı, eğitim verisi kullanılarak eğitme işlemine alınır. Eğitme işleminde ilk olarak birinci örüntünün her özelliği birinci temel öğreniciye ait zayıf sınıflandırıcılar ile sınıflandırılır. Her temel öğrenici içinde özellik sayısı kadar zayıf sınıflandırıcı bulunur. 1. özellik 1. zayıf sınıflandırıcı ile, 2. özellik 2. zayıf sınıflandırıcı ile sınıflandırılacak şekilde bu işlem bütün özellikler için yapılır. Bu işlem bütün örüntüler kullanılarak yapılır. Daha sonra diğer temel öğreniciler için aynı işlem tekrarlanır ve zayıf sınıflandırıcı hatası ve temel öğrenici parametreleri elde edilir. Şekil 5.5' te bu yöntemin eğitim akış şeması verilmiştir. Karar sistemi uygulanırken, eğitme işlemi sonucunda elde edilen parametrelere eğitim verileri giriş olarak verilir. Burada sınıflandırma yapılırken en iyi özelliğin parametreleri kullanılır. İlk örüntü için her bir özelliğin sınıflandırma sonucu alınır. m adet özelliğin -1 ve +1 sınıflandırma sonuçlarına bakılarak ilk örüntünün hangi sınıfa ait olduğuna karar verilir. Örüntü, özellik sayısının yarısından fazla +1 sınıfı içeriyorsa +1, -1 sınıfı içeriyorsa -1 olarak sınıflandırılır. Daha sonra temel öğrenicinin ağırlığı ile sınıflandırma sonucu çarpılarak toplanır ve tek bir sınıflandırma sonucu elde edilir. Bu işlem bütün örüntüler için tekrarlanır. Test aşamasında ise sadece en iyi özellikler, en iyi özellik parametreleri ile test edilerek sınıflandırılmıştır. Şekil 5.6' da ise test akış şeması verilmiştir.



Şekil 5.5 Üçüncü yöntemin eğitim aşaması



Şekil 5.6 Üçüncü yöntemin test aşaması

Çizelge 5.11' de göğüs kanseri verisi için bu yöntem ile elde edilen sonuçlar verilmiştir. Bütün çizelgelerde 10 kat çapraz geçerlilik ile sonuçlar elde edilmiştir.

Çizelge 5.11 Üçüncü yöntem ile elde edilen göğüs kanseri verisi sınıflandırma sonuçları

TÖ Sayısı	4	8	12	16	20	**24**
Duyarlılık	91,45	90,59	92,81	93,26	92,10	**93,52**
Belirlilik	93,26	94,45	90,99	92,26	91,50	**92,30**
Doğruluk	92,05	91,91	92,20	92,94	91,91	**93,08**
Süre (sn)	11,56	22,6	33,29	45,25	57,14	**66,31**

Bu yöntemde göğüs kanseri verisi için en yüksek performans 24 temel öğrenici kullanıldığında elde edilmiştir. Önceki iki yöntemde temel öğrenici sayısı 4 ve 12 idi. Temel öğrenici sayısı 24 iken sınıflandırıcı yapının sağlıklı veriyi sağlıklı olarak sınıflandırma başarısı %93,52' dir. Hasta veriyi hasta olarak sınıflandırma başarısı ise %92,30' dur. Genel sınıflandırma başarısı ise %93,08' dir.

Pima diyabet verisi için bu yöntem ile elde edilen sonuçlar Çizelge 5.12' de verilmiştir.

Çizelge 5.12 Üçüncü yöntem ile elde edilen pima diyabet verisi sınıflandırma sonuçları

TÖ Sayısı	4	8	12	16	**20**	24
Duyarlılık	65,38	59,23	60	62,69	**65,76**	62,69
Belirlilik	79,4	76,8	79,8	75,2	**80,8**	80,2
Doğruluk	74,60	70,78	73,02	70,92	**75,65**	74,21
Süre (sn)	30,9	63,78	131,25	278,46	**551,41**	1124,18

Pima diyabet verisi sınıflandırıldığında en yüksek başarı 20 temel öğrenici kullanıldığında elde edilmiştir. Temel öğrenici sayısı 20 iken duyarlılık %65,76' dır. Belirlilik %80,8' dir. Doğruluk oranı ise %75,65' tir.

Çizelge 5.13' te karaciğer düzensizliği verisi için bu yöntem ile elde edilen sonuçlar verilmiştir.

Çizelge 5.13 Üçüncü yöntem ile elde edilen karaciğer düzensizliği verisi sınıflandırma sonuçları

TÖ Sayısı	**4**	8	12	16	20	24
Duyarlılık	**16,42**	17,85	22,14	21,42	26,42	11,42
Belirlilik	**87**	84,5	76	82,5	76,5	84,5
Doğruluk	**57,94**	57,05	53,82	57,35	55,88	54,41
Süre (sn)	**9,24**	17,08	25,24	38,14	47,21	58,24

Temel öğrenici sayısı 4 ile 24 arasında değiştirilmiş ve en yüksek performans 4 temel öğrenici kullanıldığında elde edilmiştir. Temel öğrenici sayısı 4 iken duyarlılık; sağlıklı veriyi sağlıklı olarak sınıflandırma oranı %16,42' dir. Belirlilik; hasta veriyi hasta olarak sınıflandırma oranı %87' dir. Doğruluk; genel sınıflandırma oranı ise %57,94' tür.

Sonuçlara bakıldığında ikinci yöntemin hem birinci hem de üçüncü yöntemden yüksek performansa sahip olduğu görülmüştür. Bundan dolayı sınıflandırma işleminde ikinci yöntem kullanılmıştır.

5.3 Ayrık Zamanlı Veriler Üzerinde Gerçeklenen Uygulamalar

Tez çalışmasında ayrık zamanlı ve sürekli zamanlı olmak üzere iki farklı veri türü kullanılmıştır. Ayrık zamanlı veriler olarak UCI veri tabanında bulunan Wisconsin göğüs kanseri, Pima diyabet ve karaciğer düzensizliği verileri ele alınmıştır. En yüksek performansı gösteren ikinci yöntem kullanılarak oluşturulan sınıflandırıcı yapı ile bu veriler sınıflandırılmıştır.

Göğüs kanseri, Pima diyabet, karaciğer düzensizliği verileri SÖS-AdaBoost-DVM ve TBA-AdaBoost-SVM yapıları ile sınıflandırılarak, bu yapıların performansı elde edilmiştir.

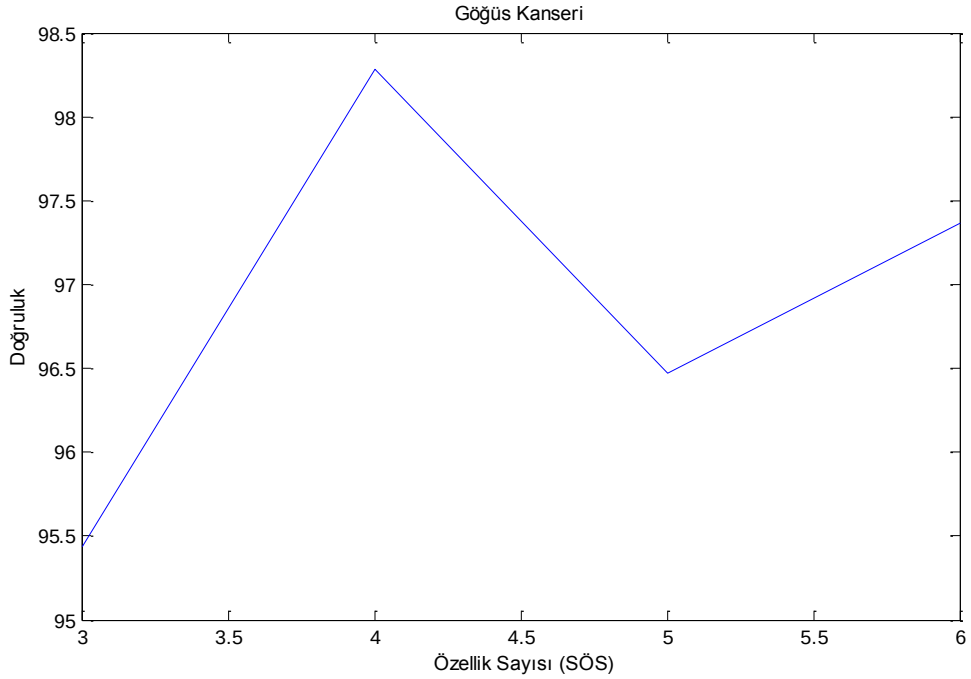
SÖS-AdaBoost-DVM yapısında göğüs kanseri verisi için 4 temel öğrenici kullanılmıştır. Her bir temel öğrenici içinde 4 adet zayıf sınıflandırıcı bulunmaktadır. Pima diyabet verisi için 5 temel öğrenici kullanılırken her bir temel öğrenici 5 adet zayıf sınıflandırıcıya sahiptir. Karaciğer düzensizliği verisinde ise 4 temel öğrenici kullanılmıştır, her bir temel öğrenici içinde 4 adet zayıf sınıflandırıcıya sahiptir.

Çizelge 5.14' te SÖS-AdaBoost-DVM yapısının Wisconsin göğüs kanseri, pima diyabet ve karaciğer düzensizliği için elde ettiği sınıflandırma sonuçları görülmektedir.

Çizelge 5.14 SÖS-AdaBoost-DVM yapısının sınıflandırma sonuçları

Veri	Duyarlılık	Belirlilik	Doğruluk	Seçilen Özellik Sayısı	Geçen Süre(sn)
Göğüs kanseri	97,56	100	98,36	4	8,505
Pima diyabet	81,63	28	63,51	5	13,181
Karaciğer düzensizliği	80	37,5	53,84	4	7,586

Şekil 5.7' de göğüs kanseri verisi için SÖS-AdaBoost-DVM yapısı ile farklı sayılarda özellik seçimi yapıldığında elde edilen doğruluk oranları verilmiştir.



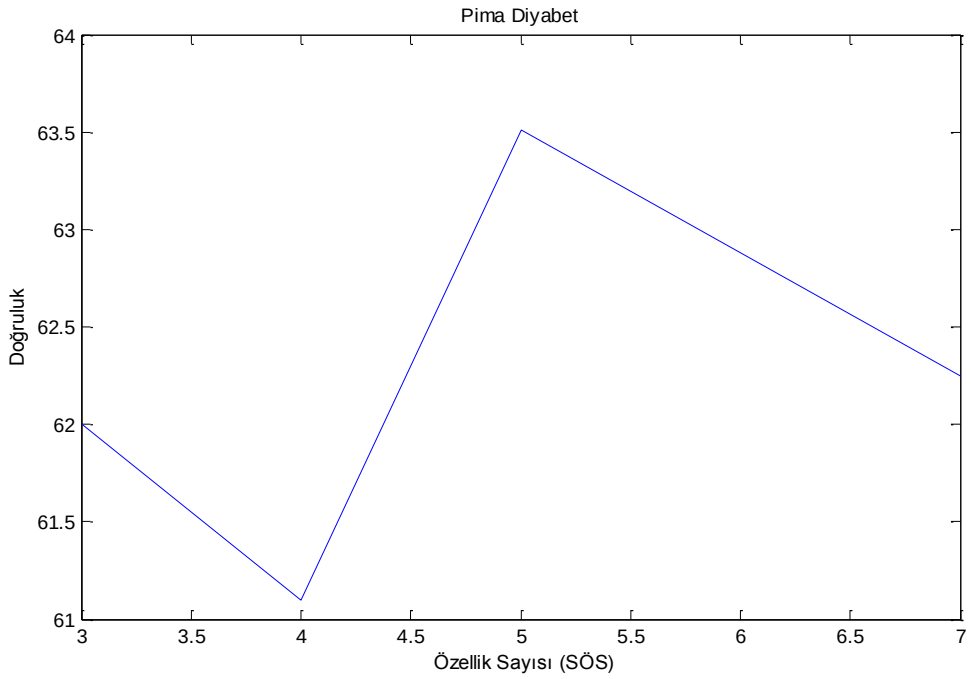
Şekil 5.7 Göğüs kanseri verisi için SÖS ile yapılan özellik seçme işlemi

En yüksek başarı dört özellik ile elde edildiğinden dolayı göğüs kanseri verisinde en etkin dört özellik Sıralı Özellik Seçimi algoritması ile seçilmiş, AdaBoost-DVM yapısı bu dört özelliği sınıflandırmıştır. Duyarlılık; sağlıklı veriyi sağlıklı olarak sınıflandırma oranı %97,56' dır. Belirlilik; hasta veriyi hasta olarak sınıflandırma oranı %100' dür. Doğruluk; genel sınıflandırma başarısı ise %98,36' dır. Çizelge 5.15' te göğüs kanseri verisi için karmaşıklık matrisi verilmiştir.

Çizelge 5.15 Göğüs kanseri verisi karmaşıklık matrisi

Göğüs Kanseri	Sağlıklı	Hasta
Sağlıklı	40	0
Hasta	1	20

Şekil 5.8' de pima diyabet verisi için SÖS-AdaBoost-DVM yapısı ile farklı sayılarda özellik seçimi yapıldığında elde edilen doğruluk oranları verilmiştir.



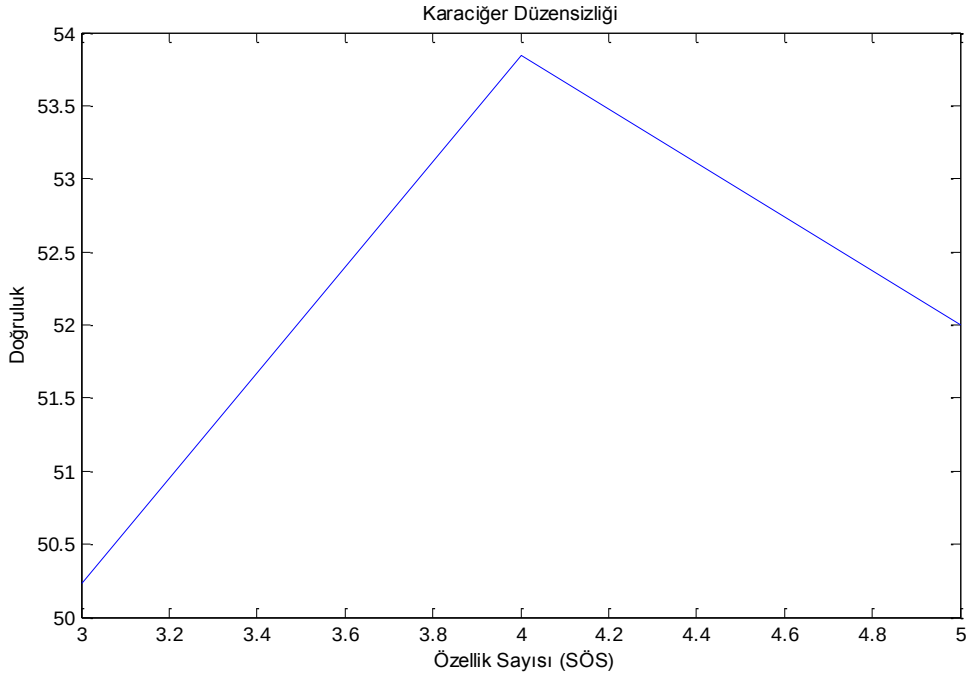
Şekil 5.8 Pima diyabet verisi için SÖS ile yapılan özellik seçme işlemi

Sıralı Özellik Seçimi algoritması ile en etkin beş özelliği seçilen pima diyabet verisi AdaBoost-DVM sınıflandırıcı yapı ile sınıflandırılmıştır. Duyarlılık %81,63, belirlilik %28 ve doğruluk %63,51 olarak bulunmuştur. Çizelge 5.16' da pima diyabet verisi için karmaşıklık matrisi verilmiştir.

Çizelge 5.16 Pima diyabet verisi karmaşıklık matrisi

Pima diyabet	Sağlıklı	Hasta
Sağlıklı	40	18
Hasta	9	7

Şekil 5.9' da karaciğer düzensizliği verisi için SÖS-AdaBoost-DVM yapısı ile farklı sayılarda özellik seçimi yapıldığında elde edilen doğruluk oranları verilmiştir.



Şekil 5.9 Karaciğer düzensizliği verisi için SÖS ile yapılan özellik seçme işlemi

Karaciğer düzensizliği verisinde Sıralı Özellik Seçimi algoritması ile en etkin dört özellik seçilmiş, AdaBoost-DVM yapısı ile bu özellikler sınıflandırılmıştır. Sınıflandırıcı yağının sağlıklı veriyi sağlıklı olarak sınıflandırma başarısı %80, hasta veriyi hasta olarak sınıflandırma başarısı %37,5 iken genel sınıflandırma başarısı ise %53,84' tür. Çizelge 5.17' de karaciğer düzensizliği verisi için karmaşıklık matrisi verilmiştir.

Çizelge 5.17 Karaciğer düzensizliği verisi karmaşıklık matrisi

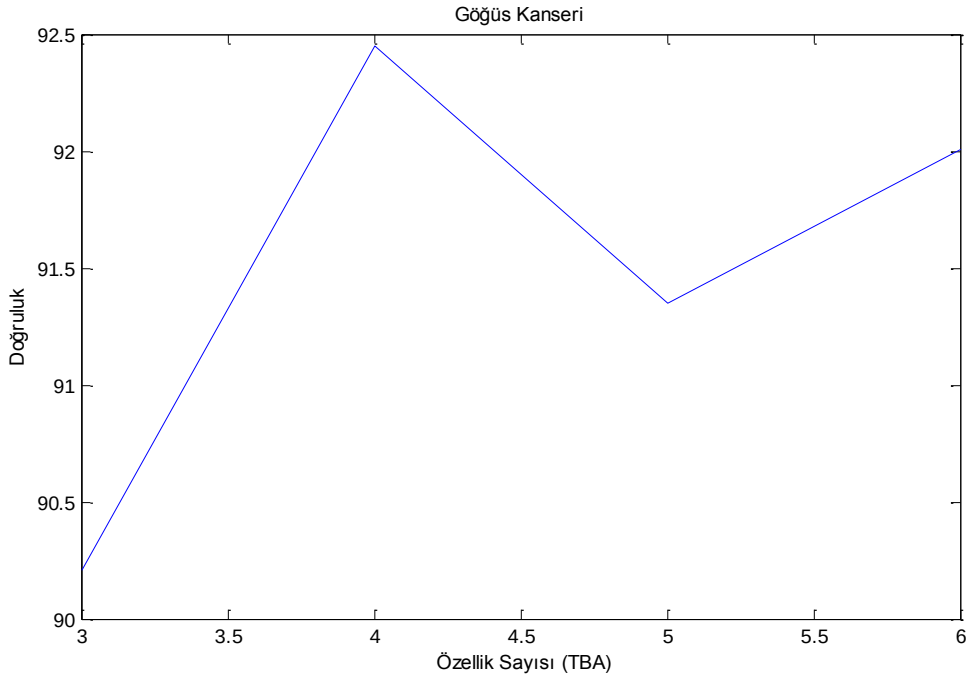
Karaciğer düzensizliği	Sağlıklı	Hasta
Sağlıklı	4	5
Hasta	1	3

TBA-AdaBoost-DVM yapısında ise SÖS-AdaBoost-DVM yapısında kullanılan parametrelerin aynıları kullanılmıştır. Her bir veri için temel öğrenici ve zayıf sınıflandırıcı sayıları aynı tutulmuştur. TBA-AdaBoost-DVM yapısının Wisconsin göğüs kanseri ve pima diyabet verileri için elde ettiği sınıflandırma sonuçları Çizelge 5.18' de görülmektedir.

Çizelge 5.18 TBA-AdaBoost-DVM yapısı ile elde edilen sınıflandırma sonuçları

Veri	Duyarlılık	Belirlilik	Doğruluk	Seçilen Özellik Sayısı	Geçen Süre(sn)
Göğüs kanseri	97,43	78,57	92,45	4	10,553
Pima diyabet	80	62,5	69,23	5	22,071

Şekil 5.10' da göğüs kanseri verisi için TBA-AdaBoost-DVM yapısı ile farklı sayılarda özellik seçimi yapıldığında elde edilen doğruluk oranları verilmiştir.



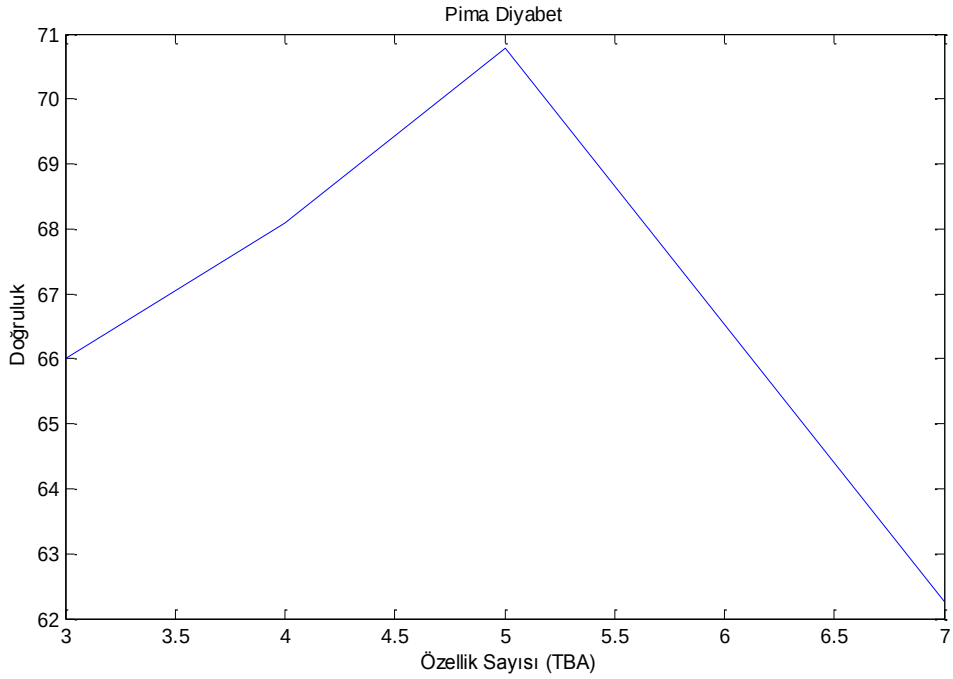
Şekil 5.10 Göğüs kanseri verisi için TBA ile yapılan özellik seçme işlemi

Temel Bileşen Analizi algoritması ile göğüs kanserinin en etkin dört özelliği seçilmiş, AdaBoost-DVM yapısı bu dört özelliği sınıflandırmıştır. Sınıflandırıcı yapının duyarlılığı %97,43 iken belirliliği %78,57' dir. Genel sınıflandırma başarısı ise %92,45' tir. Çizelge 5.19' da göğüs kanseri verisi için karmaşıklık matrisi verilmiştir.

Çizelge 5.19 Göğüs kanseri verisi karmaşıklık matrisi

Göğüs kanseri	Sağlıklı	Hasta
Sağlıklı	38	3
Hasta	1	11

Şekil 5.11’ de Pima diyabet verisi için TBA-AdaBoost-DVM yapısı ile farklı sayılarda özellik seçimi yapıldığında elde edilen doğruluk oranları verilmiştir.



Şekil 5.11 Göğüs kanseri verisi için TBA ile yapılan özellik seçme işlemi

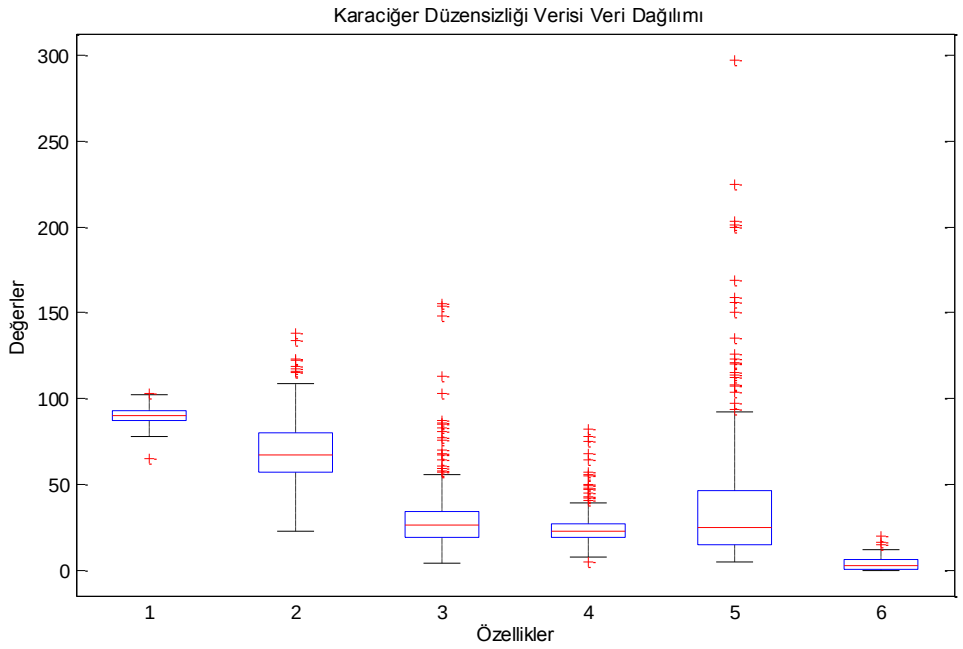
Pima diyabet verisinin en etkin beş özelliği TBA algoritması ile seçilmiş, AdaBoost-DVM yapısı kullanılarak bu beş özellik sınıflandırılmıştır. Duyarlılık; sağlıklı veriyi sağlıklı olarak sınıflandırma oranı %80’ dir. Belirlilik; hasta veriyi hasta olarak sınıflandırma oranı %62,5’ dir. Genel sınıflandırma başarısı ise %69,23’ dir. Çizelge 5.20’ de pima diyabet verisi için karmaşıklık matrisi verilmiştir.

Çizelge 5.20 Pima diyabet verisi karmaşıklık matrisi

Pima diyabet	Sağlıklı	Hasta
Sağlıklı	4	3
Hasta	1	5

TBA algoritması ile özellikleri çıkartılan karaciğer düzensizliği verisi eğitim aşamasında sınıflandırıcı yapının hata şartını sağlayamadığı için sınıflandırılamamıştır. Bundan dolayı karmaşıklık matrisi elde edilememiştir. Bu hata şartı sınıflandırma işlemi

sonucunda zayıf sınıflandırıcı hatasının 0.5' ten büyük çıkmasıdır. Şekil 5.12'de verinin dağılımı incelendiğinde 3, 4 ve 5. özelliklerin birbiri ile yakın ilişki içinde olduğu; 1,2 ve 6. özelliklerin ise yakın ilişkili olmadığı görülmektedir. TBA metodunda özellik çıkarma aşamasında veri ile çarpılacak olan özvektör bütün özellikler hesaba katılarak üretilmektedir. Bu neden TBA kullanımında özellik çıkartma uygulanacak olan verinin içindeki özelliklerin yakın ilişki içinde olması sistem çıkışını pozitif yönde etkiler. Ancak, buradaki gibi kullanılan verinin özelliklerinin sadece üç tanesi yakın ilişki içinde olduğundan dolayı TBA metodu karaciğer düzensizliği için pozitif çıkış etkisi yapamamaktadır. Elde edilen sonuçlar ve gerçeklenen grafik analizi söz konusu durumu ve birbirini destekler niteliktedir.



Şekil 5.12 Karaciğer düzensizliği verisi veri dağılımı

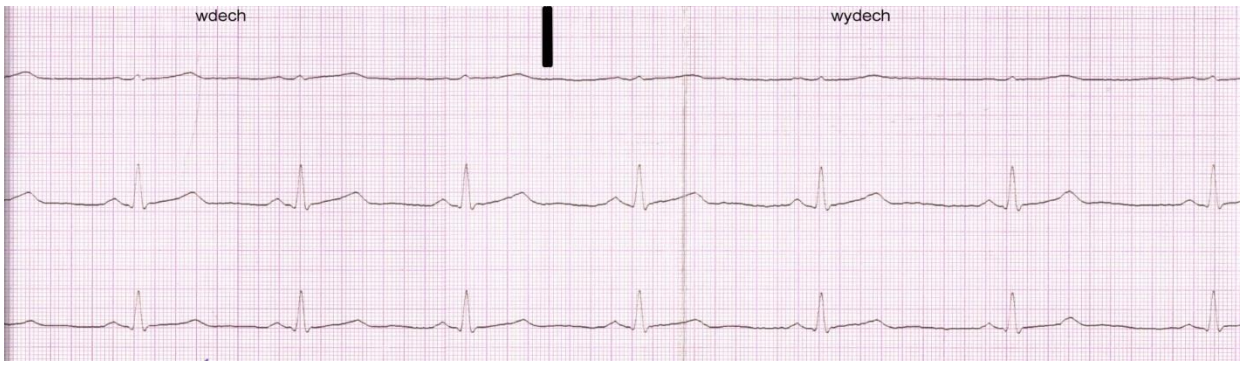
SÖS ve TBA algoritmalarından oluşan iki yapıda da temel öğrenici sayısı ikinci yöntemde en yüksek başarıyı veren temel öğrenici sayısı ile aynı tutulmuştur.

SÖS algoritması ile özellik çıkartma işleminde göğüs kanserinde TBA' ya göre %6 oranında daha yüksek bir başarı elde edilirken, algoritma daha kısa sürede sonuç vermiştir. Pima diyabet verisinde ise TBA algoritması SÖS' e göre %6 oranında yüksek performans gösterirken, SÖS algoritması daha hızlı sonuç vermiştir. Karaciğer düzensizliği verisinde ise SÖS algoritması ile seçilen özellikler kullanılarak eğitim

işlemi yapılan AdaBoost-DVM yapısında herhangi bir problem oluşmazken, TBA algoritması kullanılarak seçilen özellikler eğitim işlemine girdiğinde AdaBoost-DVM yapısı eğitilememektedir. Bunun sebebi ise algoritmanın hata şartlarının sağlanamamasıdır.

5.4 Sürekli Zamanlı Veriler Üzerinde Gerçeklenen Uygulamalar

Bu çalışmada, sürekli zaman verisi olarak Elektrokardiyogram (EKG) verisi kullanılmıştır. EKG, kalp akım grafiğidir. Kalpteki elektriksel aktivitenin, kalp kası hücreleri ve kalp odacıklarının depolarizasyonu ve repolarizasyonunun kaydıdır. Kalpteki elektriksel sinyaller yüzey elektrotları ve uygun elektronik cihazlar kullanılarak ölçülmektedir. Elde edilen veri, karakteristik bir şablona sahip olan EKG dalgasına çevrilir. Normal EKG şablonunda oluşan sapmalar kardiyak hastalıklarının tanısında kullanılabilmektedir (Kutscher, 2013). Şekil 5.13' de örnek bir EKG grafiği verilmiştir.



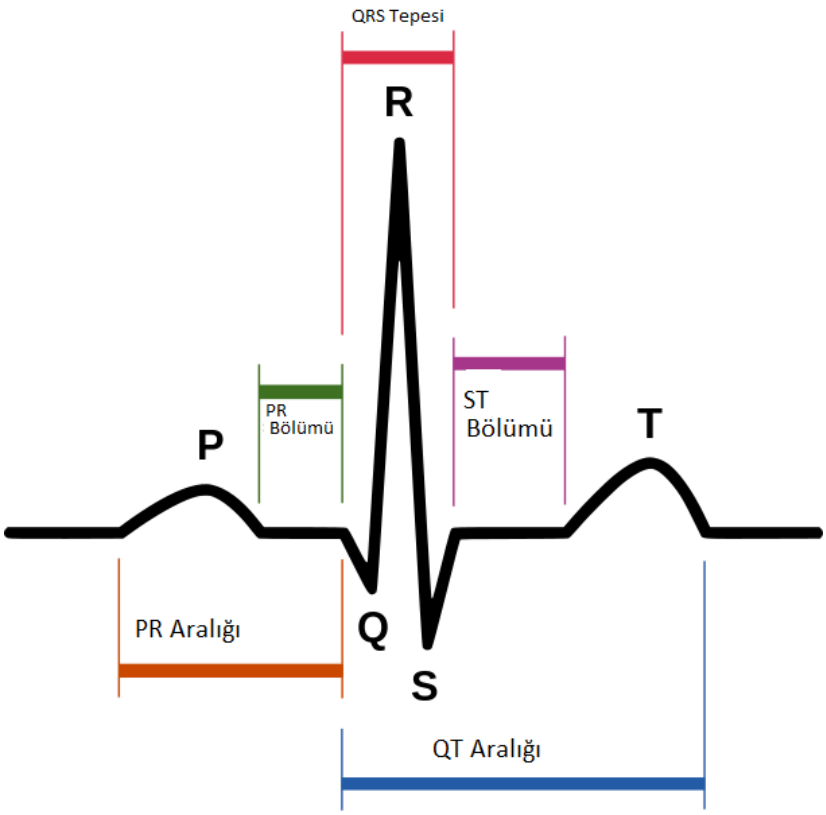
Şekil 5.13 EKG grafiği

Anatomik olarak kalp dört odacığa (sağ ve sol kulakçık, sağ ve sol karıncık), kan damarlarına ve kalp içinde ters yöne kan dolaşımını engelleyen dört kapakçığa sahiptir. Kalbin diğer bir özelliği elektrik iletim sistemidir (Kutscher, 2013).

Kalp ritmi ilk olarak sinüs düğümünden çıkan elektrik uyarıları ile oluşur. Bundan dolayı kalp kası kendi başına kasılma özelliğine sahiptir. Kalbin sinüs düğümü adı verilen noktasından çıkan düzenli uyarılar özel bir iletim yoluyla kalp kası hücrelerine ulaşır. Normalde dakikada 60-100 defa oluşan bu uyarılar kalp odacıklarını çevreleyen kalp kasını tamamen kasarak içinde bulunan kanı büyük ve küçük dolaşıma gönderir. Bu olaya kalp kasılması denir. Kalp kası hücreleri çok kısa süren bu kasılma

durumundan hemen sonra eski durumuna geçerek kulakçıklarda bulunan kanı karıncıklara boşaltır. Bu işlem nabız sayısı kadar tekrarlanır Kalpteki elektrik değişimleri ile meydana gelen akım değişiklikleri kan ve dokular yardımıyla bütün vücuda yayılır. Bu akım değişikliklerinin oluşturduğu elektriksel değişimler, vücudun çeşitli noktalarına konan elektrotlar yardımıyla yükseltilerek kaydedilir. Kalpte oluşan bu elektrik akımlarının kağıda dökülmesi işlemine EKG denir [1].

Normal bir EKG' de (5.14) P, QRS ve T diye adlandırılan 3 dalga ve bu dalgalar arasında düz çizgiler vardır. Bu dalga ve çizgilerdeki değişiklikler kalbin normal atışındaki sapmaları gösterir. P dalgası kulakçıkların uyarılması ile oluşan kasılmayı, QRS dalgası karıncıklara geçen uyarılar ile kasılmayı, T dalgası karıncıkların sakin hâle gelmesini gösterir. Dalgalarda düzenli olarak birbirlerini takip etmelerde oluşan değişiklikler ile de kalpte olabilecek yapı değişikliği veya hastalık kendini gösterebilir [1].



Şekil 5.14 Normal bir EKG dalgası

5.4.1 EKG Verisini Oluşturan Kısımlar

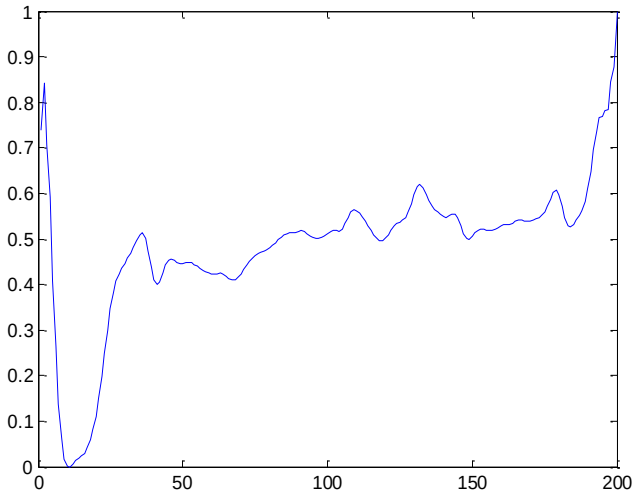
EKG verisi Normal Sinüs Ritmi, Sağ Dal Bloku ve Sol Dal Bloku olmak üzere üç kısımdan oluşmaktadır.

5.4.1.1 Normal sinüs ritmi

Kalbin normal atışıdır. Kalp bir dakikada 60-100 defa atar. Her QRS dalgası öncesinde bir P dalgası gelmektedir. PR aralığı sabit kalmaktadır (Surawicz ve Knilans, 2008).

5.4.1.2 Sağ dal bloku

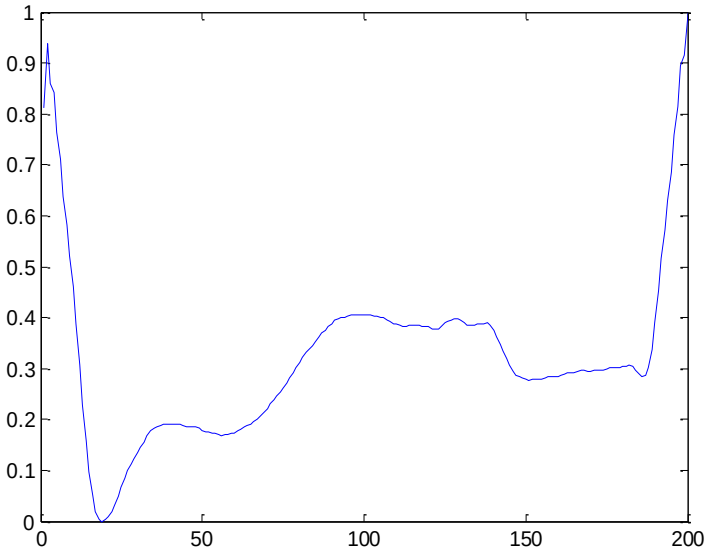
Şekil 5.15’ te sağ dal bloğunun grafiği görülmektedir. Sağ dal bloğunda QRS tepesi yetişkinlerde 120ms veya daha fazla (en az 3 küçük kare); 4-16 yaş arası çocuklarda 100ms veya daha fazla; 0-4 yaş arası çocuklarda 90ms veya daha fazladır. I ve aVL, V5 ve V6’ da geniş çentikli R dalgası vardır. Bazen V5’ te veya V6’ da RS şablonu gözlemlenebilir. ST ve T dalgaları genellikle QRS’ in zıt yönünde olur. Sağ dal bloğunun ortaya çıkması ile QRS aksı da değişebilir. Aks değişikliği bazı durumlarda kalp hızına da bağlı olabilir [2].



Şekil 5.15 Sağ dal bloğu grafiği

5.4.1.3 Sol dal bloku

Şekil 5.16' da sol dal bloğunun grafiği görülmektedir. Sol dal bloğunda QRS tepesi yetişkinlerde 120ms veya daha fazla (en az 3 küçük kare); 4-16 yaş arası çocuklarda 100ms veya daha fazla; 0-4 yaş arası çocuklarda 90ms veya daha fazladır. I ve V6' da geniş S dalgası vardır: S dalgası ya R dalgasından daha geniş ya da 40ms' den fazla olmalıdır. Sağlıklı bireylerde görülebilir (izole sağ dal bloğu). Sağlıklı bireylerde izole sağ dal bloğu, izole sol dal bloğuna göre daha sık görülür. Sağ kalp kateterizasyonundan sonra kateter travmasına bağlı olarak görülebilir [3].

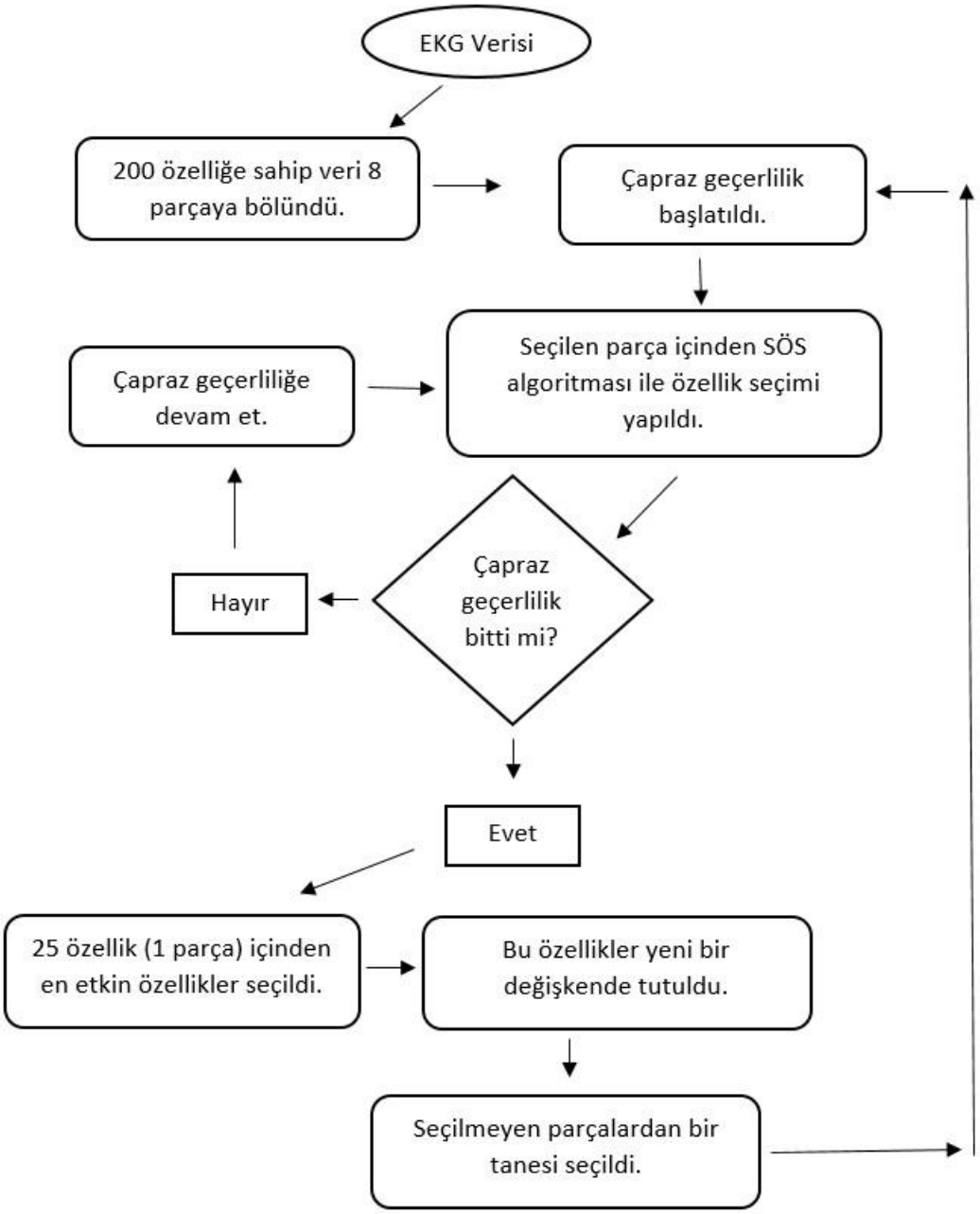


Şekil 5.16 Sol dal bloğu grafiği

5.4.2 EKG sinyalinin sınıflandırma sonuçları

Çalışmada üç farklı EKG sinyal tipi kullanılmıştır. Ayrıca, sunulan sınıflayıcı algoritmalar ikili (binary) bir yapıya sahip olduğundan, çoklu sınıflandırma teorilerinin kullanılması gerekir. Bundan dolayı, sınıflandırma işleminde birine-karşı-hepsi (BKH) yaklaşımı tercih edilmiştir (Barstuğan ve Ceylan, 2014). BKH yaklaşımının algoritması şu şekildedir: Veri $\Psi = \{\phi_1, \phi_2, \dots, \phi_M\}$ şeklinde ise, bu gösterimde M, veri içerisindeki farklı sınıf sayısıdır. BKH yaklaşımında her sınıfı verideki diğer sınıflardan ayıran bir ikili sınıflayıcı çalıştırılır. Bu ikili sınıflayıcının amacı söz konusu sınıfı ($\phi_i, i = 1,2, \dots, M$), diğer sınıflardan ($\Psi - \{\phi_i\}$) ayırmaktır. Bu şekilde M adet sınıflayıcı her sınıf için eğitilir (Beygelzimer ve ark., 2005).

EKG verisine SÖS ve TBA özellik seçme yöntemleri uygulanarak boyut azaltılma işlemi yapılmıştır. Şekil 5.17' de EKG verisinde SÖS ile özellik seçme işleminin nasıl yapıldığı gösterilmiştir.



Şekil 5.17 EKG verisinde Sıralı Özellik Seçimi ile yapılan özellik seçimi aşaması

Şekil 5.16' da SÖS' ün özellik seçerken 200 özelliği 25'er özellikten oluşan 8 parçaya böldüğü görülmektedir. EKG verisi burada 10 ya da 15'e de bölünebilirdi. 25' e bölünmesinin sebebi işlem süresini kısaltmak ve daha yüksek başarı elde etmektir. EKG verisi 28 ya da daha fazla parçaya bölündüğünde ise SÖS tarafından seçilemediği yapılan denemelerde görülmüştür. Bu durumlarda sadece bir adet özellik seçilebilmektedir. O özellik 200' üncü yani sonuncu özelliktir. Bundan dolayı optimum bölme sayısı 25 adet olarak kabul edildi. Her 25 özellikte en etkin özellikler seçilmekte ve yeni bir değişkene atanmaktadır. Daha sonra sıradaki 25 özellikten en etkin özellikler

seçilerek bir önceki yeni değişkene eklenir. Burada sıralı işlem yapıldığı için verilerin yeri değişmemekte, sadece seçilmeyen özellikler çıkarılmaktadır. Seçilen özellik sayısı ise değiştirilebilir bir kurala bağlıdır. Bu kural, 10-kat çapraz geçerlilik sırasında *m* defadan fazla seçilen özelliği ele almaktır. *m* sayısı, seçilen özelliğin sayısı artırılmak istendiği zaman azaltılır, azaltılmak istendiği zaman artırılır. Bu şekilde istenilen sayıda özellik seçilebilir.

5.4.2.1 Özellik sayısı ile AdaBoost temel öğrenici sayısının eşit olduğu durumda

Sıralı Özellik Seçimi ve TBA algoritmaları ile özellik seçimi yapıldıktan sonra elde edilen yeni veri sınıflandırıcı yapının girişi olmaktadır. Sınıflandırıcı yapıdaki Temel Öğrenici (TÖ) sayısı özellik sayısı ile eşit tutulabilir veya değiştirilebilir. Ayrıca, her bir temel öğrenici içinde özellik sayısı adedince zayıf sınıflandırıcı kullanılmıştır. Çizelge 5.21 ve Çizelge 5.22' de özellik sayısı ile temel öğrenici sayısının eşit olduğu durumda elde edilen sonuçlar verilmiştir.

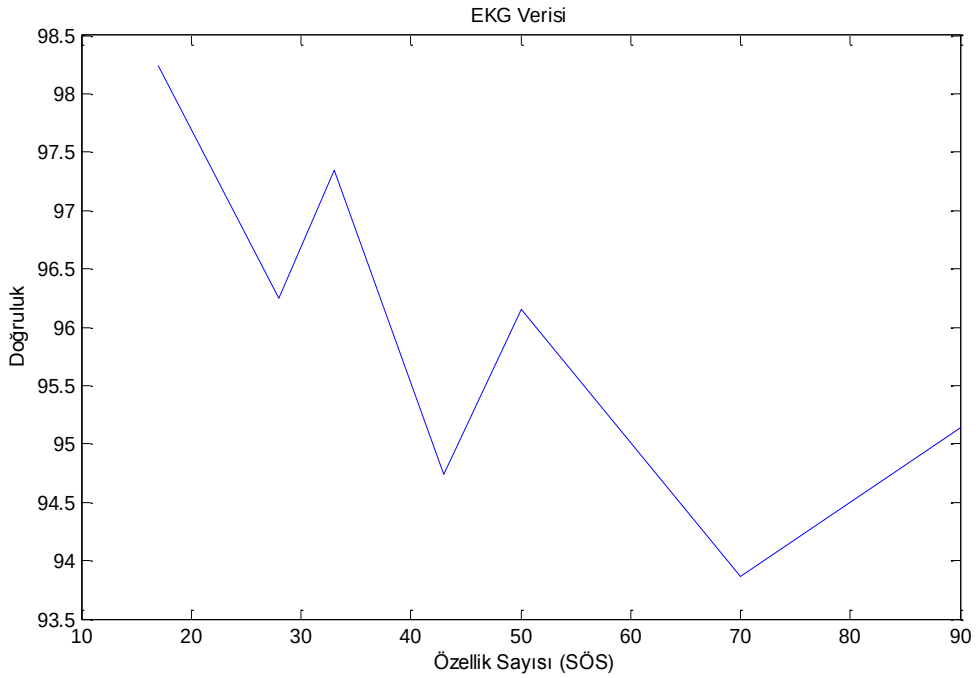
Çizelge 5.21 Özellik sayısı ile TÖ sayısı aynı olduğu durumda EKG verisinin sınıflandırma sonuçları

	TBA-AdaBoost-DVM				**SÖS-AdaBoost-DVM**			
Özellik Sayısı & TÖ Sayısı	17		33		17		33	
Duyarlılık	91,17		74,63		98,68		98,67	
Belirlilik	69,56		44,68		97,37		94,74	
Doğruluk	82,46		62,28		98,24		97,35	
Süre (sn)	120,79		390,3		222,25		290,49	
Karmaşıklık Matrisi	62	14	50	26	75	1	74	2
	6	32	17	21	1	37	1	36

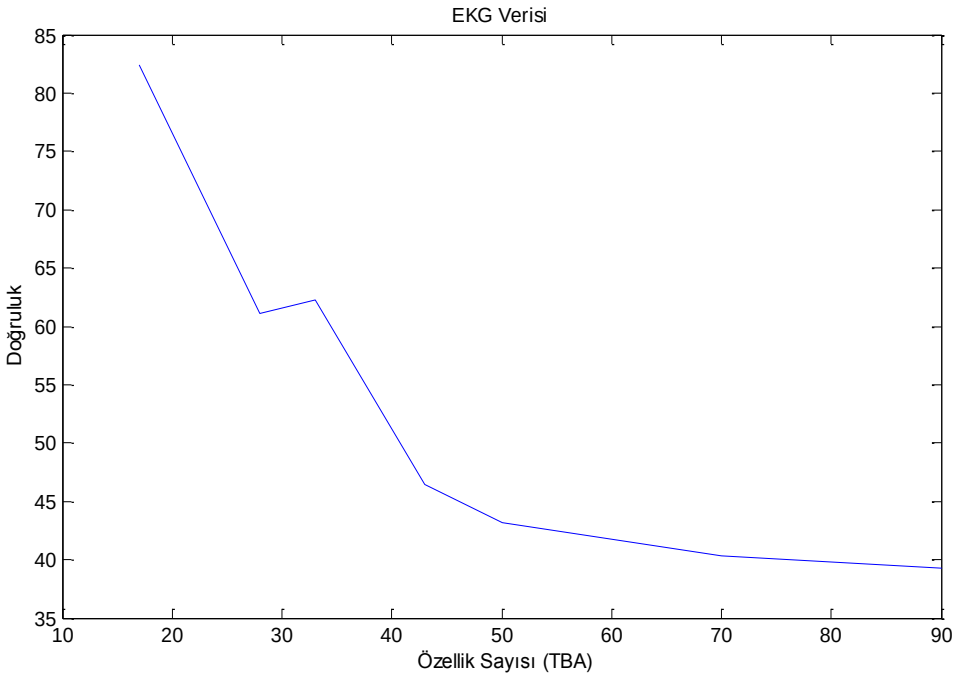
Çizelge 5.22 Özellik sayısı ile TÖ sayısı aynı olduğu durumda EKG verisinin sınıflandırma sonuçları

	TBA-AdaBoost-DVM				**SÖS-AdaBoost-DVM**			
Özellik Sayısı & TÖ Sayısı	43		70		43		70	
Duyarlılık	61,54		56,45		96,05		96	
Belirlilik	26,53		21,15		92,1		89,74	
Doğruluk	46,49		40,35		94,74		93,86	
Süre (sn)	540,79		1431,5		339,59		679,09	
Karmaşıklık Matrisi	40	36	35	41	73	3	72	4
	25	13	27	11	3	35	3	35

17, 33, 43 ve 70 özellik için elde edilen sonuçlara göre SÖS kullanılan sınıflandırıcı yapı, TBA kullanılan yapıdan üstün çıkmıştır. 17 özellik seçildiğinde, SÖS kullanılan yapının duyarlılık değeri %98,68 iken, TBA yapısında bu değer %91,17' dir. SÖS yapısının belirlilik değeri %97,37 iken, TBA yapısında bu değer %69,56' dır. Genel başarı oranı ise SÖS yapısında %98,24 iken, TBA yapısında %82,46' dır. TBA' nın SÖS' e tek üstünlüğü algoritmanın eğitim ve test süresinde harcadığı toplam zamandır. TBA yapısı 120,79 saniyede sonuç verirken, SÖS yapısı 222,25 saniyede sonuç vermektedir. Çizelgelerde farklı özellik sayıları için elde edilen sonuçlara bakılırsa SÖS algoritmasının kullanıldığı AdaBoost-DVM sınıflandırıcısının, TBA algoritmasının kullanıldığı AdaBoost-DVM sınıflandırıcısından daha iyi sonuçlar ürettiği görülmektedir. Şekil 5.18' de SÖS ile, Şekil 5.19' da ise TBA ile çıkarılan özellik sayılarına göre doğruluk oranının grafiği verilmiştir.



Şekil 5.18 EKG verisi için SÖS ile yapılan özellik seçme işlemi



Şekil 5.19 EKG verisi için TBA ile yapılan özellik seçme işlemi

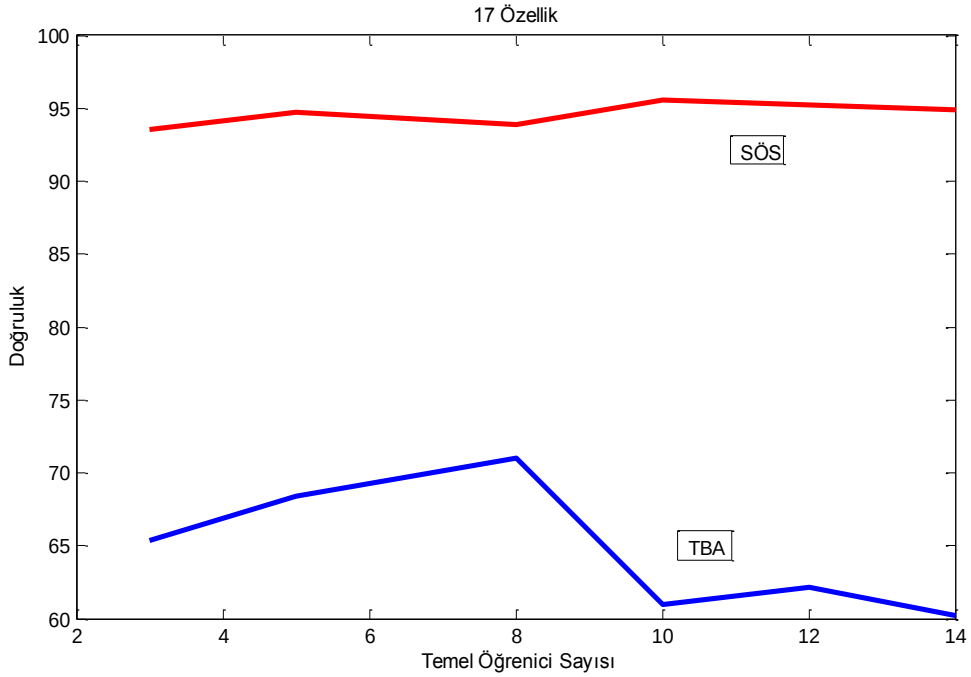
5.4.2.2 Özellik sayısı ile AdaBoost temel öğrenici sayısının farklı olduğu durumda

Temel öğrenici sayısı özellik sayısından bağımsız tutulmuş, ancak her bir temel öğrenici içinde özellik sayısı kadar zayıf sınıflandırıcı kullanılmıştır. Bu durumda elde edilen sonuçlar Çizelge 5.23, Çizelge 5.24 Çizelge 5.25 ve Çizelge 5.26' da sunulmuştur.

Çizelge 5.23 EKG verisinin 17 özellik için elde edilen sınıflandırma sonuçları

	TBA-AdaBoost-DVM						**SÖS-AdaBoost-DVM**					
Özellik Sayısı	17						17					
TÖ Sayısı	5		8		10		5		8		10	
Duyarlılık	76,32		80,28		71,42		96,05		94,81		96,1	
Belirlilik	52,63		55,81		41,03		92,11		91,89		94,59	
Doğruluk	68,42		71,05		61,21		94,74		93,86		95,61	
Süre (sn)	110,09		165,12		201,44		183,76		201,25		213,87	
Karmaşıklık Matrisi	58	18	57	19	55	23	73	3	73	3	74	2
	18	20	14	24	22	16	3	35	4	34	3	35

Çizelge 5.22’ de görüldüğü üzere, 17 özellik için elde edilen sonuçlara göre, SÖS kullanılan yapı en yüksek başarıyı 10 temel öğrenici kullanıldığında vermiştir. 10 temel öğrenici kullanıldığında yapının sağlıklı veriyi sağlıklı olarak sınıflama başarısı %96,1 iken, hasta veriyi hasta olarak sınıflandırma başarısı %94,59’ dur. Genel başarı oranı ise %95,61’dir. TBA yapısında ise en yüksek başarı ise 8 temel öğrenici kullanıldığında elde edilmiştir. 8 temel öğrenici kullanıldığında yapının sağlıklı veriyi sağlıklı olarak sınıflama başarısı %80,28 iken, hasta veriyi hasta olarak sınıflandırma başarısı %55,81’ dir. Genel başarı oranı ise %71,05’ dir. TBA’ nın SÖS’ e tek üstünlüğü algoritmanın eğitim ve test süresinde harcadığı toplam zamandır. TBA yapısı 165,12 saniyede sonuç verirken, SÖS yapısı 201,25 saniyede sonuç vermektedir. Şekil 5.20’ de SÖS ve TBA ile 17 özellik çıkarıldığında elde edilen doğruluk oranının grafiği verilmiştir.

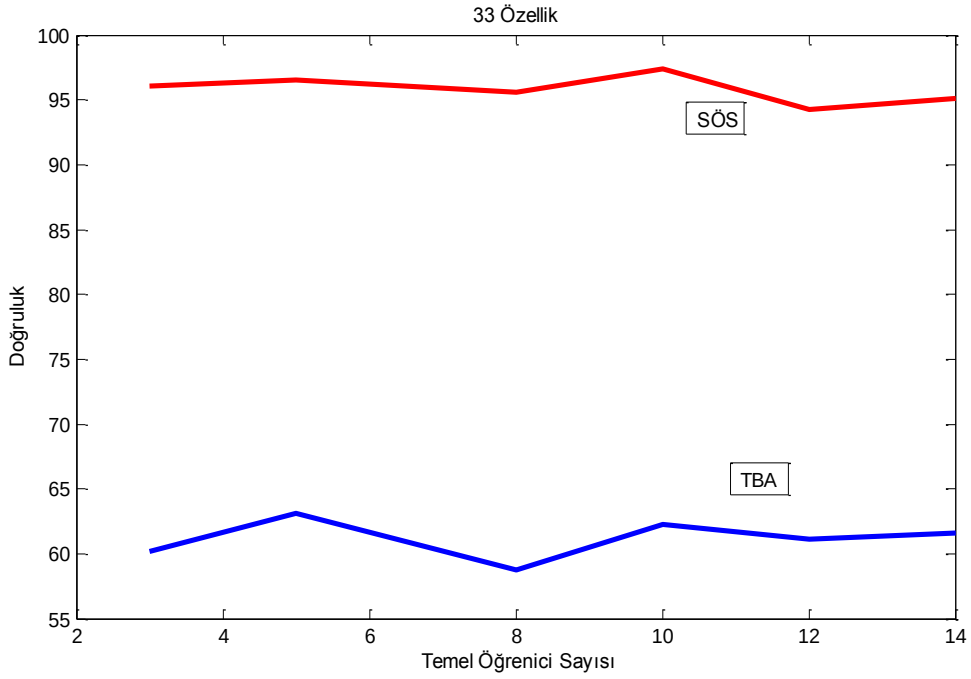


Şekil 5.20 Temel öğrenici sayısına doğruluk oranı grafiği

Çizelge 5.24 EKG verisinin 33 özellik için elde edilen sınıflandırma sonuçları

	TBA-AdaBoost-DVM						**SÖS-AdaBoost-DVM**					
Özellik Sayısı	33						33					
TÖ Sayısı	5		8		10		5		8		10	
Duyarlılık	75,76		71,64		74,63		96,15		94,93		94,4	
Belirlilik	45,83		40,43		44,68		97,22		97,14		97,29	
Doğruluk	63,15		58,77		62,28		96,49		95,61		97,37	
Süre (sn)	173,46		270,23		359,3		225,23		234,48		275,85	
Karmaşıklık Matrisi	50	26	48	28	50	26	75	1	75	1	75	1
	16	22	19	19	17	21	3	35	4	34	2	36

33 özellik için elde edilen sonuçlara bakıldığında SÖS yapısının daha başarılı olduğu görülmektedir. En yüksek doğruluk oranı elde edilen SÖS yapısında duyarlılık değeri %94,4 iken, en yüksek doğruluk oranı elde edilen TBA yapısında bu değer %63,15' dir. SÖS yapısının belirlilik değeri %97,29 iken, TBA yapısında bu değer %45,83' tür. Genel başarı oranı ise SÖS yapısında %97,37 iken, TBA yapısında %63,15' tir. TBA, SÖS' e 5 temel öğrenici kullanıldığında harcanan eğitim ve test süresi harici herhangi bir üstünlük sağlayamamıştır. En yüksek performans SÖS yapısında 10 adet temel öğrenici, TBA yapısında 5 adet temel öğrenici kullanıldığında elde edilmiştir. Şekil 5.21' de SÖS ve TBA ile 33 özellik çıkarıldığında elde edilen doğruluk oranının grafiği verilmiştir.

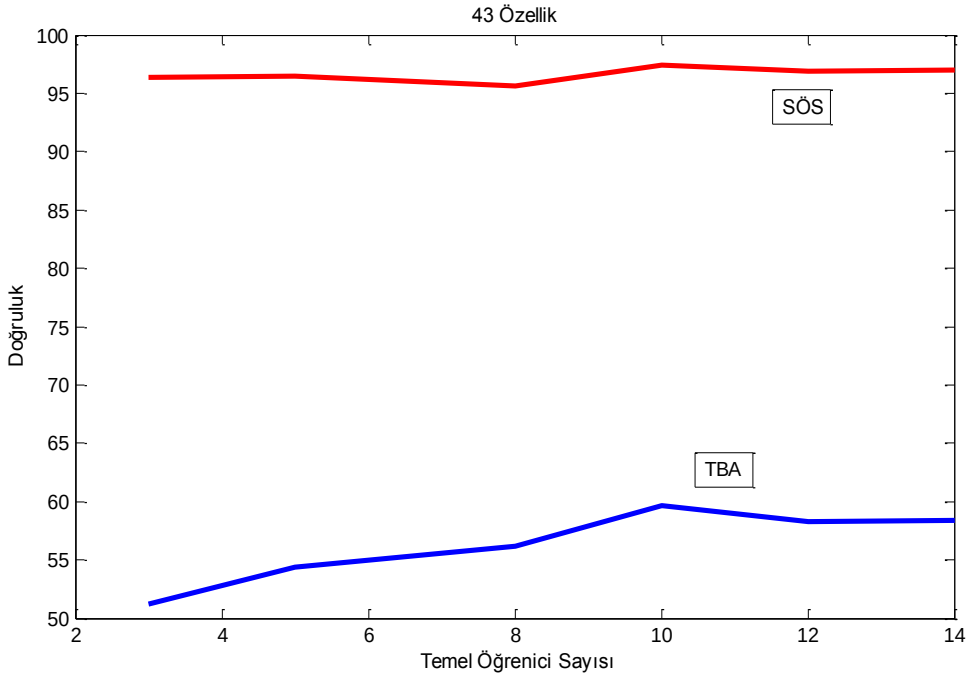


Şekil 5.21 Temel öğrenici sayısına doğruluk oranı grafiği

Çizelge 5.25 EKG verisinin 43 özellik için elde edilen sınıflandırma sonuçları

	TBA-AdaBoost-DVM						**SÖS-AdaBoost-DVM**					
Özellik Sayısı	43						43					
TÖ Sayısı	5		8		10		5		8		10	
Duyarlılık	66,66		67,57		70,27		97,37		96,1		98,66	
Belirlilik	33,33		35		40		94,73		94,59		94,87	
Doğruluk	54,39		56,14		59,65		96,49		95,61		97,37	
Süre (sn)	271,55		340,27		375,97		238,78		279,6		325,15	
Karmaşıklık Matrisi	48	28	50	26	52	24	74	2	74	2	74	2
	24	14	24	14	22	16	2	36	3	35	1	37

Çizelge 5.24' te görüldüğü üzere, 43 özellik için elde edilen sonuçlara göre, en yüksek doğruluk oranı elde edilen SÖS yapısının duyarlılık değeri; sağlıklı veriyi sağlıklı olarak sınıflama başarısı %98,66 iken, en yüksek doğruluk oranı elde edilen TBA yapısında bu değer %70,27' dir. SÖS yapısının belirlilik değeri; hasta veriyi hasta olarak sınıflandırma başarısı %94,87 iken, TBA yapısında bu değer %40' tır. Doğruluk; genel başarı oranı ise SÖS yapısında %97,37 iken, TBA yapısında %59,65' dir. TBA, SÖS' e hiçbir şekilde üstünlük sağlayamamıştır. En başarılı performans 10 adet temel öğrenicinin kullanıldığı yapı ile elde edilmiştir. Şekil 5.22' de SÖS ve TBA ile 43 özellik çıkarıldığında elde edilen doğruluk oranının grafiği verilmiştir.

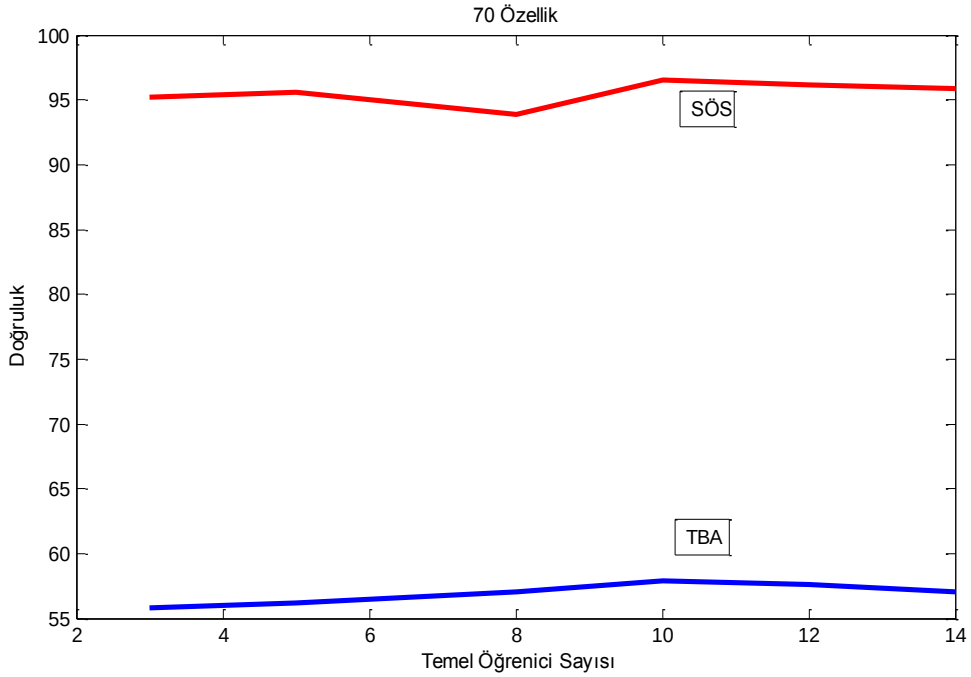


Şekil 5.22 Temel öğrenici sayısına doğruluk oranı grafiği

Çizelge 5.26 EKG verisinin 70 özellik için elde edilen sınıflandırma sonuçları

	TBA-AdaBoost-DVM						**SÖS-AdaBoost-DVM**					
Özellik Sayısı	70						70					
TÖ Sayısı	5		8		10		5		8		10	
Duyarlılık	67,1		67,53		68,42		97,33		94,81		97,37	
Belirlilik	34,21		35,14		36,84		92,31		91,89		94,74	
Doğruluk	56,14		57,02		57,89		95,63		93,86		96,49	
Süre (sn)	411,21		580,97		598,27		316,73		370,28		433,34	
Karmaşıklık Matrisi	51	25	52	24	52	24	73	3	73	3	74	2
	25	13	25	13	24	14	2	36	4	34	2	36

70 özellik için elde edilen sonuçlara bakıldığında, en yüksek performans her iki yapıda da 10 temel öğrenici ile elde edilmiştir. SÖS kullanılan yapının duyarlılık değeri %97,37 iken, TBA yapısında bu değer %68,42' dir. SÖS yapısının belirlilik değeri %94,74 iken, TBA yapısında bu değer %36,84' tür. Genel başarı oranı ise SÖS yapısında %96,49 iken, TBA yapısında %57,89' dur. Şekil 5.23' de SÖS ve TBA ile 70 özellik çıkarıldığında elde edilen doğruluk oranının grafiği verilmiştir.



Şekil 5.23 Temel öğrenici sayısına doğruluk oranı grafiği

70 özellik kullanıldığında da TBA, SÖS' e üstünlük sağlayamamıştır. En yüksek başarıya çoğunlukla 10 temel öğrenici kullanıldığında ulaşılırken, sadece 17 özellikte TBA yapısında 8 temel öğrenici ile ulaşılmıştır.

EKG verisi için en yüksek sınıflama başarısı 33 ve 43 özellik için 10 adet temel öğrenicinin kullanıldığı yapı ile elde edilmiştir.

6. SONUÇLAR VE ÖNERİLER

6.1 Sonuçlar

Yapılan tez çalışmasında DVM tabanlı AdaBoost topluluk sınıflandırıcı sistem tasarlanmıştır. Bu sisteme özellik çıkartma algoritmaları eklenmiştir. Oluşturulan sistem ayrık ve sürekli zaman verileri ile test edilmiştir.

Çalışmada, SÖS ve TBA özellik çıkarma algoritmaları ile biyomedikal verilerde boyut düşümü gerçekleştirilmiş ve eğitim-test süreleri azaltılmıştır. Oluşturulan özellik çıkarma-sınıflama yapısı 10-kat çapraz geçerlilik yöntemi ile eğitim-test işlemine tabi tutulmuştur. Bu şekilde, bütün örüntüler hem eğitim hem de test işleminde kullanılmıştır. Böylece sistem performans değerlendirmesi yapılmıştır.

Sıralı Özellik Seçimi algoritmasının uygulanması sırasında işlenen veri parçalara ayrılmıştır. 28 ve daha fazla özellik içeren verilerde etkin özellik olarak sadece bir adet özellik çıkarabilirken, daha az sayıda özellik içeren verilerde sorunsuz bir şekilde işlem yapmaktadır. Bunun sebebi SÖS algoritmasının belli bir özellik sayısından sonra uygun sonuçlar vermemesidir. Bundan dolayı, optimum parça sayısı her parçada 25 özellik olacak şekilde 8 olarak bulunmuştur.

AdaBoost topluluk sınıflandırıcısında temel öğrenici olarak Destek Vektör Makineleri kullanılmıştır. Karar Ağacı' na göre daha yüksek performans elde edilmiştir. Yüksek lisans seminer çalışmasında Karar Ağacı tabanlı AdaBoost algoritması ile göğüs kanseri verisinde %95,279 sınıflandırma doğruluğu elde edilmişken, Karar Ağacı yerine Destek Vektör Makineleri kullanılan başka bir çalışmada %96,08 oranında bir sınıflandırma doğruluğu elde edilmiştir (Barstugan & Ceylan, 2014).

AdaBoost algoritmasının ağırlık güncelleme formülünde logaritmik ifade, üzeri 1.5 olacak şekilde değiştirilerek başarı oranı artırılmıştır. Güncelleme formülünde değişiklik yapılmadığı zaman sınıflayıcı yapı hata oranı 0.5 üzerine çıktığı için çalışmayı durdurmaktadır. Logaritmik ifadenin üssü sırasıyla 0.9, 1.2, 1.5 ve 1.9 yapılmış ve optimum üs değeri 1.5 olarak bulunmuştur.

Topluluk sınıflandırıcı AdaBoost yapısında temel öğrenici sayısı iki farklı yöntemle elde edilerek sistemin başarısı gözlenmiştir. İlk yöntemde, sınıflandırılan verinin özellik sayısı kadar temel öğrenici kullanılmıştır. 17 özellik çıkarıldıysa 17 temel öğrenici, 70 özellik çıkarıldıysa 70 temel öğrenici kullanılmıştır. İkinci yöntemde

temel öğrenici sayısı manuel olarak ayarlanmıştır. 5, 8 ve 10 adet temel öğrenici EKG verisinden çıkartılan 17, 33, 43 ve 70 özellik ile test edilmiştir.

Çizelge 6.1 ve Çizelge 6.2' de literatürde yapılan çalışmalar ile tez çalışmasında elde edilen sonuçlar karşılaştırılmıştır.

Çizelge 6.1 Ayrık veriler için literatürde elde edilen sonuçlar ile tez çalışmasında elde edilen sonuçların karşılaştırılması

Veri	Yayın	Doğruluk (%)	Yapı
Göğüs kanseri	Neshat M. ve ark. (2008)	95,3	DVM
	Rodriguez J.J. & Maudes J. (2007)	95,08	Standart AdaBoost
	Tez çalışması (2014)	98,36	SÖS-AdaBoost-DVM
	Tez çalışması (2014)	92,45	TBA-AdaBoost-DVM
Pima Diyabet	Rodriguez J.J. & Maudes J. (2007)	74,93	Standart AdaBoost
	Neshat M. ve ark. (2008)	77,9	DVM
	Tez çalışması (2014)	63,51	SÖS-AdaBoost-DVM
	Tez çalışması (2014)	69,23	TBA-AdaBoost-DVM
Karaciğer Düzensizliği	Neshat M. ve ark. (2008)	70,8	DVM
	Pham, D. T ve ark. (2000)	55,9	RULES-4
	Tez çalışması (2014)	53,84	SÖS-AdaBoost-DVM
	Tez çalışması (2014)	-	TBA-AdaBoost-DVM

Çizelge 6.2 EKG sınıflandırmada literatürde elde edilen sonuçlar ile tez çalışmasında elde edilen sonuçların karşılaştırılması

Veri	Yayın	Sınıf Sayısı	Doğruluk (%)	Yapı
EKG	Kao ve ark. (2012)	2	91,95	KDKNN-AdaBoost
	Xie B. & Minn H. (2012)	10	77,53	Standart Adaboost
	Tez çalışması (2014)	3	97,37	SÖS-AdaBoost-DVM
	Tez çalışması (2014)	3	71,05	TBA-AdaBoost-DVM

6.2 Öneriler

Gerçeklenen uygulamalarda elde edilen sonuçlara bakıldığında temel öğrenici sayısı özellik sayısına bağlı olmadan, manuel değiştirildiğinde daha yüksek bir başarı elde edilebilmektedir. Bundan dolayı çeşitli denemeler yaparak optimum temel öğrenici sayısı bulunmalıdır.

AdaBoost algoritmasında ağırlık güncelleme formülünde logaritmik ifadenin üs değerinin optimum değeri bulunmalıdır.

Kullanılan verilerde özellik çıkarma yöntemi olarak SÖS algoritmasının TBA algoritmasından oldukça üstün olduğu görülmektedir. Bundan dolayı EKG ve göğüs kanseri, pima diyabet vb. biyomedikal verilerde özellik çıkarma yöntemi olarak SÖS algoritmasının kullanılması uygun olabilir.

Tez çalışmasında önerilen SÖS-AdaBoost-DVM hibrit yapısının genelleştirilebilmesi için bu yapı farklı biyomedikal verilerle de test edilmelidir. Bu şekilde, önerilerin sınıflandırıcı yapının performansı daha sağlıklı bir şekilde değerlendirilebilir.

KAYNAKLAR

Âlvarez, 1., Görriz, J. M., Ramirez, J., Salas-Gonzalez, D., Löpez, M., Segovia, F., Prieto, B., 2009, Alzheimer's diagnosis using eigenbrains and support vector machines, *In Bio-lnspired Systems: Compulational and Ambient Intelligence*, Springer Berlin Heidelberg, 973-980.

Barstugan, M. & Ceylan, R., 2014, Comparison of Decision Tree and SVM Based AdaBoost Algorithms on Biomedical Benchmark Datasets, *MBEC2014: 6th European Conference of the International Federation for Medical and Biological Engineering*, 63.

Barstugan, M. & Ceylan, R., 2014, Topluluk Sınıflandırıcıları ve Örüntü Sınıflandırma Uygulamaları, Yüksek Lisans Seminer Çalışması, 23.

Beltrân, N. H., Duarte-Mermoud, M. A., Soto Vicencio, V. A., Salah, S. A., & Bustos, M. A., 2008, Chilean Wine classification using volatile organic compounds data obtained with a fast GC analyzer, *Instrumentation and Measurement*, IEEE Transactions on, 57(11), 2421-2436.

Beygelzimer, A., Langford, J., & Zadrozny, B., 2005, Weighted one-against-all, In *AAAI*, 720-725.

Bui, L., Tran, D., Huang, X., & Chetty, G., 2011, Face recognition based on Gabor features, *In Visual Information Processing (EUVIP)*, 3rd European Workshop on IEEE, 264-268.

Chen, J., Wang, 12., Yan, S., Shan, S., Chen, X., & Gao, W., 2007, Enhancing human face detection by resampling examples through manifoids, *Systems, Man and Cybernetics, Part A: Systems and Humans*, IEEE Transactions on, 37(6), 1017-1028.

De Marsico, M., Nappi, M., 'Riccio, D., & Wechsler, H., 2013, Robust Face Recognition for Uncontrolled Poşe and Illumination Changes, *Systems, Man, and Cybernetics: Systems*, IEEE Transactions on, 43(1), 149-163.

Deng, K., 1999, OMEGA: On-Line Memory-BasedGeneral Purpose System Classifier, *PHD Thesis, Chapter 7*, 120-122.

Erkilinc, M. S., & Sahin, F., 2011, Camera control with EMG signals using Principal Component Analysis and support vector machines, *In Systems Conference (SysCon)*, 2011 IEEE International, 417-421.

Florea, L., Oprisescu, S., & Florea, C., 2012, Mouth area analysis by the use of selected spectral energies, *In Communications (COMM)*, 2012 91h International Conference on IEEE, 103-106.

Freund, Y. and Robert E. S., 1995, A desicion-theoretic generalization of on-line learning and an application to boosting, *Computational learning theory*. Springer Berlin Heidelberg.

Gao, J., Rao, N., Yang, Y., & Wei, W., 2012, A New Lie Detection Method Based on Small-number of P 300 Responses, *In Biomedical Engineering and Biotechnology (ICBEB)*, 2012 International Conference on IEEE, 662-665.

Garg, A., Pavlovic, V., & Rehg, J. M., 2003, Boosted learning in dynamic Bayesian networks for multimodal speaker detection, *Proceedings of the IEEE, 91(9)*, 1355-1369.

Guo, G., & Dyer, C. R., 2005, Learning from examples in the small sample case: face expression recognition, *Systems, Man, and Cybernetics, Part B: Cybernetics*, IEEE Transactions on, 35(3), 477-488.

Han, X., 2010, Nonnegative principal component analysis for cancer molecular pattern discovery, *Computational Biolow and Bioinformatics*, IEEE/ACM Transactions on, 7(3), 537-549.

Isa, S. M., Fanany, M. 1., Jatmiko, W., & Arymurthy, A. M., 2011, Sleep apnea detection from ECG signal: analysis on optimal features, principal components, and nonlinearity, *In Bioinformatics and Biomedical Engineering (ICBBE)*, 2011 5th International Conference on IEEE, 1-4.

Jiangtao, L., & Zhenpu, G., 2010, Identification of the mineral oil fluorescence spectroscopy based on the ICA and SVM, *In Computer Design and Applications (ICCDA)*, 2010 International Conference on IEEE, 506-508.

Kanaan, L., Merheb, D., Kauas, M., Francis, C., Amoud, H., & Honeine, P., 2011, PCA and KPCA of ECG signals with binary SVM classification, *In Signal Processing Systems (SIPS)*, 2011 IEEE Workshop on IEEE, 344-348.

Kao, T. P., Wang, J. S., Lin, C. W., Yang, Y. T., & Juang, F. C., 2012, Using Bootstrap AdaBoost with KNN for ECG-based automated obstructive sleep apnea detection. In *Neural Networks (IJCNN)*, 1-5.

Kallas, M., Francis, C., Kanaan, L., Merheb, D., Honeine, P., & Amoud, H., 2012, Multi-class SVM classification combined with kernel PCA feature extraction of ECG signals, *In Telecommunications (ICT)*, 2012 191h International Conference on IEEE, 1-5.

Kégl, B., 2009, Introduction to AdaBoost, 11-14.

Kubat, M., Holte, R. C., & Matwin, S., 1998, Machine learning for the detection of oil spills in satellite radar images, *Machine learning*, 30(2-3), 195-215.

Kulkarni, S. R., Harman, G., 2011, Statistical learning theory: a tutorial, *Wiley Interdisciplinary Reviewx: Computational Statistics*, 3(6), 543-556.

Kutscher, S. Algorithms for ECG Feature Extraction: an Overview, 2013.

Lu, Y., Tian, Q., Sanchez, M., Neary, J., Feng, L., & Wang, Y., 2007, Learning microarray gene expression data by hybrid discriminant analysis. *MultiMedia*, IEEE, 14(4), 22-31.

Marcano-Cedeño, A., Quintanilla-Domínguez, J., Cortina-Januchs, M. G., & Andina, D., 2010, Feature selection using sequential forward selection and classification applying artificial metaplasticity neural network. In *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, IEEE, 2485-2850.

Melgani, F., Bazi, Y., 2008, Classification of electrocardiogram signals with support vector machines and particle swarm optimization, *Information Technology in Biomedicine, IEEE Transactions on 12(5),* 667-677.

Neshat, M., Yaghobi, M., & Naghibi, M., 2008, Designing an expert system of liver disorders by using neural network and comparing it with parametric and nonparametric system, In *Systems, Signals and Devices, IEEE SSD 5th International Multi-Conference on IEEE*, 1-6.

Pham, D. T., Dimov, S. S., & Salem, Z., 2000, Technique for selecting examples in inductive learning, In *European symposium on intelligent techniques, Aachen, Germany*, 119-127.

Provost, F. J., Fawcett, T., & Kohavi, R., 1998, The case against accuracy estimation for comparing induction algorithms, In *ICML*, Vol.98, 445-453.

Rodríguez, J. J., & Maudes, J., 2008, Boosting recombined weak Classifiers, *Pattern Recognition Letters*, 29(8, 1049-1059.

Roychowdhury S., Koozekanani D., Parhi, K., 2012, DREAM: Diabetic Retinopathy Analysis using Machine Learning, *In Biomedical and Health Informatics*, IEEE Journal, 1-12.

Rückstieβ, T., Osendorfer, C. and Smagt P., 2011, Sequential Feature Selection for Classification, *AI 2011: Advances in Artificial Intelligence*, 132-141.

Sanjeev, R.K. and Gilbert H., 2001, Statistical Learning Theory: A Tutorial, 16-21.

Smith, L., 2002, A tutorial on Principal Component Analysis, 1-27.

Subasi, A., & Gursoy, M. I., 2010, EEG signal classification using PCA, ICA, LDA and support vector machines, *Expert Systems with Applications*, *37*(12), 8659-8666.

Sun, Z., Zheng, C. H., Gao, Q. W., Zınang, J., & Zhang, D. X., 2012, Tumor Classification Using Eigengene-Based Classifier Committee Learning Algorithm, *Signal Processing Lelters*, IEEE, 19(8), 455-458.

Surawicz, B., & Knilans, T., 2008, Chou's electrocardiography in clinical practice: adult and pediatric, *Elsevier Health Sciences*.

Tao, H., Rui, L., & Mei-Juan, Z., 2010, Face recognition under complex conditions, *In Electrical and Control Engineering (ICECE)*, 2010 International Conference on IEEE 960-963.

Theodoridis, S., Pikrakis, A., Koutroumbas, K., & Cavouras, D., 2010, *Introduction to Pattern Recognition: A Matlab Approach: A Matlab Approach*, Academic Press, 79-80.

Tofighi, A., & Monadjemi, S. A., 2011, Face detection and recognition using skin color and AdaBoost algorithm combined with Gabor features and SVM classifier, *In Multimedia and Signal Processing (CMSP)*, 2011 International Conference on IEEE, 141-145.

Vapnik, V. N., 1999, An overview of statistical learning theory, *Neural Networks*, *IEEE Transactions on*, *10*(5), 988-999.

Xie, B., & Minn, H., 2012, Real-time sleep apnea detection by classifier combination, *Information Technology in Biomedicine, IEEE Transactions on, 16*(3), 469-477.

Ye, C., Coimbra, M. T., & Vijaya Kumar, B. V. K., 2010, Arrhythmia detection and classification using morphological and dynamic features of ECG signals, *In Engineering in Medicine and Biolow Society (EMBC)*, 2010 Annual International Conference of the IEEE, 1918-1921.

Yun, G. & Reeves, S.J., 1998, Efficient computation for sequential forward observation selection in image reconstruction. In *Image Processing ICIP 98. Proceedings*, 1998 International Conference on IEEE, 380-384.

Zhang, T., Liu, J., Liu, S., Xu, C., & Lu, H., 2011, Boosted exemplar learning for action recognition and annotation, *Circuits and Systems for Video Technology*, IEEE Transactions on, 21(7), 853-866

Zhao, Q., & Principe, J. C., 2001, Support vector machines for SAR automatic target recognition, *Aerospace and Electronic Systems*, IEEE Transactions on, 37(2), 643-654.

Zhou, J., Gao, D., & Zhang, D., 2007, Moving vehicle detection for automatic traffic monitoring, *Vehicular Technologv*, IEEE Transactions on, 56(1), 51-59.

İNTERNET KAYNAKLARI

[1] http://drsuleymanaysel.com/17/19/kardiyolojide-tani-yontemleri/elektrokardiyografi ekg/

[2] http://www.metealpaslan.com/ekg/rbbb.htm

[3] http://www.metealpaslan.com/ekg/lbbb.htm

EKLER

EK-1 SÖS-AdaBoost-DVM Yapısı ile EKG Verisi Sınıflandırma Yazılımı

```
tic
clear all
clc
load ecgnormal.dat;
load ecgrbb.dat;
load ecglbb.dat;
giris_son = [];     giris_s   = []; son_veri   = [];
train_target=zeros(1141,1);
train_target(1:758,1)=1;
train_target(759:1141,1)=-1;
hedef=train_target;
T=10;
sayi_ozellik_1 =0; % çıkartılan ozellik sayısı
sayi_ozellik_2 =0; % çıkartılan ozellik sayısı
sayi_ozellik_3 =0; % çıkartılan ozellik sayısı
%%% ECG Normal Verisi ile Sağ Dal Bloku'nun ayrımı
for q=0:25:175
   % 10'arlı gruplar hâlinde özellikler değerlendiriliyor
   giris=[ecgnormal(q+1:q+25,1:758)'; ecgrbb(q+1:q+25,1:383)'];

   ozellik =zeros(25,1);
   son_veri = zeros(1141,25);
   b=0; % Çıkartılan özellik sayısı
    for i=1:10
     c = cvpartition(hedef,'holdout',.1);
     % Her veri eğitim ve testte kullanılarak en etkin özellikler seçiliyor
     giris_egitim = giris(training(c,1),:);
    hedef_egitim = hedef(training(c,1));
    giris_test = giris(test(c,1), :);
    hedef_test = hedef(test(c,1));
    f = @(giris_egitim, hedef_egitim, giris_test, hedef_test) sum(hedef_test ~= classify(giris_test,
giris_egitim, hedef_egitim));
     fs = sequentialfs(f,giris,hedef);
      for k=1:25
     % Her seçim işlemi sonucunda seçilen özelliklerin
     % kaç defa seçildiği kaydediliyor.
        if fs(k) == 1
          ozellik(k)= ozellik(k)+1;
        end
     end
   end
   % Seçilen özellikler ile yeni giriş verisi oluşturuluyor
   for m=1:25
    if ozellik(m) >= 3 % 10 veya daha fazla seçilen özellikler
       b=b+1;
       son_veri(:,b) = giris(:,m);
    end
   end
     y=find(son_veri(1,:));
   giris_s= son_veri(:,y);

   giris_son = [giris_son giris_s];
end
  giris=giris_son;
  [S, sayi_ozellik_1] = size(giris);
```

```
sayi_ozellik_1
alpha = [];
 % Her bir zayıf öğrenicinin ağırlığı
% Alfa değeri ise her bir temel öğrenicinin ağırlığı olarak kullanılıyor.
% En iyi alpha değerini tutuyor.
for n=1:1:10 %Cross-validation'ın başladığı yer.
 c = cvpartition(hedef,'holdout',.1);
   % Sınıf bilgisi hedef_test kullanılarak eğitim seti ile test seti
   % rastgele parçalara ayırıldı. Bölme oranı %10. Yani 683 adet verinin
   % %90'ı eğitime, %10'u teste ayrılıyor.
% Özellik sayısı "w" bulunuyor.
[h w] = size(giris);
% Eğitim seti oluşturuluyor.
giris_egitim = giris(training(c,1),:);
hedef_egitim = hedef(training(c,1));
% Test seti oluşturuluyor.
giris_test = giris(test(c,1), :);
hedef_test = hedef(test(c,1));
      err(1) = inf;
   % İlk hata sonsuz alınıyor. Daha sonra geçici hata ile değişiyor.
     for j=1:w;

        % Her bir özelliğin ilk ağırlıkları oluşturuluyor.
        D = ones(size(hedef_egitim))/length(hedef_egitim);

        % SVM'ye eğitim için giriş ve hedef verileri tanıtılarak bir SVM
        % sınıflayıcı yapı oluşturuluyor. Daha sonra bu yapı test
        % edilerek yapının geçici hatası elde ediliyor.
        svmStruct(1,j) = svmtrain(giris_egitim(:,j),hedef_egitim(:,1));
        siniflandirma_sonucu= svmclassify(svmStruct(1,j),giris_egitim(:,j));
        temp = sum(D.*(siniflandirma_sonucu ~= hedef_egitim));   % özelliğin sınıflandırılma hatası

        % Geçici hata bir önceki hatadan küçük ise geçici hata, yeni hata
        % oluyor. En iyi özellikte ise en az hatayı veren özellik tutuluyor.
        if temp < err(1)
          err(1) = temp;
          en_iyi_ozellik(1,1) =j;
        end
         % Hata 0.5'ten büyük ise algoritma gereği işlem sonlandırılıyor.
        if(err(1) >= 1/2)
          disp('stop b/c err>1/2')
          break;
        end
        % Hata 0 ise 10^-6 yapılıyor. Sebebi, hata 0 olduğunda alpha
        % değerinin sonsuz olması.
        if err(1) == 0
           err(1) = 0.000001;
        end
        % Her bir özellik için ağırlık hesaplanıyor.
        alpha(j) = 1/2 * log((1-err(1))/err(1));
        % Elde edilen yeni ağırlıkla, D ağırlıkları güncelleniyor.
        D = D.* exp((-alpha(j).*hedef_egitim.* svmclassify(svmStruct(1, j),giris_egitim(:,j))).^(1.5));
        D = D./sum(D);
        % Güncellenen D ağırlıkları yeni bir matriste tutuluyor. Bu
        % ağırlıklar ilk yapıya âit. İkinci yapı, yani ikinci temel
        % öğrenici sınıflandırma yaparken her bir özelliğinin ilk
        % ağırlığı olarak, ilk temel sınıflandırıcının son ağırlıklarını
        % alacak. Bu şekilde her seferinde ilk ağırlık
        % D= 1/ length(hedef_egitim) olmayacak. Her özelliğin kendi
        % ağırlığınca D değeri güncellenecek.
```

```
      agirliklar(:,j) = D;
      agirliklar(:,j) = abs(agirliklar(:,j));
  end
   % Birinci temel öğrenicinin ağırlığı tutuluyor. Bu ağırlık en az
   % hatayı veren özelliğin ağırlığı.
  alfa(1)=alpha(en_iyi_ozellik(1,1));
  for t=1:T
     % Her temel öğrenicinin ilk hatası her zaman sonsuz olarak
     % başlayıp, daha sonra geçici hata ile güncelleniyor.
     err(t)=inf;
     for j=1:w
      % İlk ağırlıklar, bir önceki temel sınıflandırıcının en az hatayı
      % veren ağırlıkları olarak alınıyor.
      D=agirliklar(:,j);
      % SVM'ye eğitim için giriş ve hedef verileri tanıtılarak bir SVM
      % sınıflayıcı yapı oluşturuluyor. Daha sonra bu yapı test
      % edilerek yapının geçici hatası elde ediliyor.
      svmStruct(t,j) = svmtrain(giris_egitim(:,j),hedef_egitim(:,1));
      siniflandirma_sonucu= svmclassify(svmStruct(t,j),giris_egitim(:,j));
      temp = sum(D.*(siniflandirma_sonucu ~= hedef_egitim));
      if temp < err(t)
        err(t) = temp;
        en_iyi_ozellik(t,1) =j;
      end
      if(err(t) >= 1/2)
        disp('stop b/c err>1/2')
        t
        j
        break;
      end
      if err(t) == 0
          err(t) = 0.000001;
      end
      alpha(j) = 1/2 * log((1-err(t))/err(t));
      D = D.* exp((-alpha(j).*hedef_egitim.* svmclassify(svmStruct(t, j),giris_egitim(:,j))).^(1.5));
      D = D./sum(D);
      agirliklar_2(:,j) = D;
     end
     alfa(t)=alpha(en_iyi_ozellik(t,1));
     agirliklar = agirliklar_2;
     agirliklar = abs(agirliklar);
  end

% Eğitim başarısı test edilirken kullanılacak satır sayısı elde ediliyor.
[x y] = size(giris_egitim);
  % Her bir SVM yapısına eğitim verisi verilerek yapılar test ediliyor.
  % Öncelikle parametreleri yukarıda eğitimle belirlenmiş olan birinci
  % temel öğrenici ile her özellik ayrı ayrı test ediliyor. Her özelliğin
  % test sonucu toplanıyor. Sign fonksiyonu bu sonuçlar {-1,+1} formatına
  % getirilerek birinci temel sınıflandırıcının sonucu (X x 1 boyutunda)
  % elde edilip, başarısı hesaplanıyor.
  for j=1:t
     finalLabel = zeros(size(hedef_egitim));
     for k=1:w;  % Özellik sayısı kadar.
        finalLabel = alfa(j) * svmclassify(svmStruct(j, k), giris_egitim(:,k));
        tfinalLabel = sign(finalLabel);
        son(:,k) = tfinalLabel;
     end
```

```
    for o=1:x % (satır sayısı)
        bir_sayisi=0;
        for p=1:w
            if son(o,p) == 1
              bir_sayisi = bir_sayisi+1;
            end
        end
        if bir_sayisi > (y/2)
            tfinalLabel(o,1) =1;
            % Her satırdaki "1" sayısı "-1" sayısından büyükse o satır
            % yani örüntü +1 demektir.
        else
            tfinalLabel(o,1) =-1;
            % Değilse sonuç -1 demektir.
        end
    end
    dogru=0; yanlis=0;
    for l=1:x
        if tfinalLabel(l,1) == hedef_egitim(l,1)
            dogru= dogru+1;
        else
            yanlis=yanlis+1;
        end
    end
    % "t" adet temel öğrenicinin başarısı tutuluyor. "t" adet temel
    % öğrenicinin başarısı egitim_basari matrisi ile tutuluyor. Bu
    % matriste 9 adet değer tutuluyor.
    egitim_basari(1,j)= dogru *100/x;
end
% cross-validation yapıldığı için, her bir cross-validation için
% 9 temel öğrenicinin ortalaması alınarak, cross-validation adedince
% eğitim başarısı elde ediliyor.
egitim_basari_son(1,n)=mean(egitim_basari);
% Her bir SVM yapısına test verisi verilerek yapılar test ediliyor.
% Öncelikle parametreleri yukarıda eğitimle belirlenmiş olan birinci
% temel öğrenici ile eğitimde hiç kullanılmamış olan, sisteme tamamen
% yeni olan her özellik, ayrı ayrı test ediliyor. Her özelliğin test
% sonucu toplanıyor. Sign fonksiyonu bu sonuçlar {-1,+1} formatına
% getirilerek birinci temel sınıflandırıcının sonucu elde edilip,
% başarısı hesaplanıyor.
for m=1:t
     finalLabel_t = zeros(size(hedef_test));
    for a=1:w;  % Özellik sayısı kadar.
        finalLabel_t = alfa(m) * svmclassify(svmStruct(m, a), giris_test(:,a));
        tfinalLabel_t = sign(finalLabel_t);
        son_t(:,a) = tfinalLabel_t;
        % Her özellik değeri -1,+1 olarak toplanmadan tutuluyor
    end
    for o=1:size(hedef_test) % (satır sayısı)
        bir_sayisi=0;
        for y=1:w
            if son_t(o,y) == 1
              bir_sayisi = bir_sayisi+1;
            end
        end
        [A B] = size(giris_test);
        if bir_sayisi > (B/2)
            tfinalLabel_t(o,1) =1;
            % Her satırdaki "1" sayısı "-1" sayısından büyükse o satır
            % yani örüntü +1 demektir.
```

```
        else
           tfinalLabel_t(o,1) =-1;
           % Değilse sonuç -1 demektir.
        end
      end
      % Sistemin duyarlılık, belirlilik, doğruluk değerlerinin elde
      % edilmesi için her bir temel sınıflandırıcının confusion
      % matrisleri elde ediliyor.
      TP_normal=0; TN_normal=0;
      FP_normal=0; FN_normal=0;
          for p=1:size(hedef_test)
              if  hedef_test(p,1)== 1
                  if tfinalLabel_t(p,1) ==1
                    TP_normal = TP_normal+1;
                  end
                  if tfinalLabel_t(p,1) ==-1
                    FN_normal= FN_normal+1;
                  end
              end
               if  hedef_test(p,1)== -1
                  if tfinalLabel_t(p,1) ==1
                       FP_normal = FP_normal+1;
                  end
                  if tfinalLabel_t(p,1) ==-1
                       TN_normal= TN_normal+1;
                  end
               end
          end
      sensitivity(1,m)= TP_normal / (TP_normal +FN_normal)*100;
      specifity(1,m)  = TN_normal/ (TN_normal + FP_normal) *100;
      accuracy(1,m)   = (TP_normal+ TN_normal) / (TP_normal +FP_normal +TN_normal
+FN_normal)*100;
   end
   % Her cross-validation için temel öğrenicilerin ortalama başarısı
   % tutuluyor.
   sensitivity_s(1,n)= mean(sensitivity);
   specifity_s(1,n)  = mean(specifity);
   accuracy_s(1,n)   = mean(accuracy);
end % cross-validation sonu
% cross-validation adedince elde edilen sonuçların ortalaması alınarak
% tek bir test başarısı elde ediliyor.
egitim_basari_1 = mean(egitim_basari_son);
sensitivity_son_1= mean(sensitivity_s);
specifity_son_1  = mean(specifity_s);
accuracy_son_1   = mean(accuracy_s);
 giris_son = [];
  train_target=zeros(473,1);
train_target(1:90,1)=-1;
train_target(91:473,1)=1;
hedef=train_target;
%%% ECG Sağ Dal Bloku ile Sol Dal Bloku'nun ayrımı
for q=0:25:175
   % 10'arlı gruplar hâlinde özellikler değerlendiriliyor
   giris=[ecglbb(q+1:q+25,1:90)'; ecgrbb(q+1:q+25,1:383)'];

   ozellik =zeros(25,1);
   son_veri = zeros(473,25);
   b=0; % Çıkartılan özellik sayısı

   for i=1:10
```

```
      c = cvpartition(hedef,'holdout',.1);
      % Her veri eğitim ve testte kullanılarak en etkin özellikler seçiliyor

     giris_egitim = giris(training(c,1),:);
     hedef_egitim = hedef(training(c,1));
     giris_test = giris(test(c,1), :);
     hedef_test = hedef(test(c,1));
     f = @(giris_egitim, hedef_egitim, giris_test, hedef_test) sum(hedef_test ~= classify(giris_test,
giris_egitim, hedef_egitim));
     fs = sequentialfs(f,giris,hedef);
      for k=1:25
      % Her seçim işlemi sonucunda seçilen özelliklerin
      % kaç defa seçildiği kaydediliyor.
         if fs(k) == 1
            ozellik(k)= ozellik(k)+1;
         end
      end
    end
    % Seçilen özellikler ile yeni giriş verisi oluşturuluyor
    for m=1:25
     if ozellik(m) >=6 % 10 veya daha fazla seçilen özellikler
        b=b+1;
        son_veri(:,b) = giris(:,m);
     end
    end

    y=find(son_veri(1,:));
    giris_s= son_veri(:,y);

    giris_son = [giris_son giris_s];
end
   giris=giris_son;
  [S, sayi_ozellik_2] = size(giris);
    sayi_ozellik_2
for n=1:1:10 %Cross-validation'ın başladığı yer.
     c = cvpartition(hedef,'holdout',.1);
    % Sınıf bilgisi hedef_test kullanılarak eğitim seti ile test seti
    % rastgele parçalara ayırıldı. Bölme oranı %10. Yani 683 adet verinin
    % %90'ı eğitime, %10'u teste ayrılıyor.

% Özellik sayısı "w" bulunuyor.
[h w] = size(giris);
    % Eğitim seti oluşturuluyor.
 giris_egitim = giris(training(c,1),:);
hedef_egitim = hedef(training(c,1));
 % Test seti oluşturuluyor.
giris_test = giris(test(c,1), :);
hedef_test = hedef(test(c,1));
        err(1) = inf;
    % İlk hata sonsuz alınıyor. Daha sonra geçici hata ile değişiyor.
        for j=1:w;
            % Her bir özelliğin ilk ağırlıkları oluşturuluyor.
         D = ones(size(hedef_egitim))/length(hedef_egitim);
                % SVM'ye eğitim için giriş ve hedef verileri tanıtılarak bir SVM
         % sınıflayıcı yapı oluşturuluyor. Daha sonra bu yapı test
         % edilerek yapının geçici hatası elde ediliyor.
         svmStruct(1,j) = svmtrain(giris_egitim(:,j),hedef_egitim(:,1));
         siniflandirma_sonucu= svmclassify(svmStruct(1,j),giris_egitim(:,j));
         temp = sum(D.*(siniflandirma_sonucu ~= hedef_egitim));   % özelliğin sınıflandırılma hatası
```

```
    % Geçici hata bir önceki hatadan küçük ise geçici hata, yeni hata
    % oluyor. En iyi özellikte ise en az hatayı veren özellik tutuluyor.
    if temp < err(1)
      err(1) = temp;
      en_iyi_ozellik(1,1) =j;
    end
    % Hata 0.5'ten büyük ise algoritma gereği işlem sonlandırılıyor.
    if(err(1) >= 1/2)
      disp('stop b/c err>1/2')
      break;
    end
    % Hata 0 ise 10^-6 yapılıyor. Sebebi, hata 0 olduğunda alpha
    % değerinin sonsuz olması.
    if err(1) == 0
       err(1) = 0.000001;
    end
    % Her bir özellik için ağırlık hesaplanıyor.
    alpha(j) = 1/2 * log((1-err(1))/err(1));
     % Elde edilen yeni ağırlıkla, D ağırlıkları güncelleniyor.
    D = D.* exp((-alpha(j).*hedef_egitim.* svmclassify(svmStruct(1, j),giris_egitim(:,j))).^(1.5));
    D = D./sum(D);
    % Güncellenen D ağırlıkları yeni bir matriste tutuluyor. Bu
    % ağırlıklar ilk yapıya âit. İkinci yapı, yani ikinci temel
    % öğrenici sınıflandırma yaparken her bir özelliğinin ilk
    % ağırlığı olarak, ilk temel sınıflandırıcının son ağırlıklarını
    % alacak. Bu şekilde her seferinde ilk ağırlık
    % D= 1/ length(hedef_egitim) olmayacak. Her özelliğin kendi
    % ağırlığınca D değeri güncellenecek.
    %
    agirliklar_3(:,j) = D;
    agirliklar_3(:,j) = abs(agirliklar_3(:,j));
end
 % Birinci temel öğrenicinin ağırlığı tutuluyor. Bu ağırlık en az
 % hatayı veren özelliğin ağırlığı.

alfa(1)=alpha(en_iyi_ozellik(1,1));
for t=1:T
   % Her temel öğrenicinin ilk hatası her zaman sonsuz olarak
   % başlayıp, daha sonra geçici hata ile güncelleniyor.
   err(t)=inf;
   for j=1:w

    % İlk ağırlıklar, bir önceki temel sınıflandırıcının en az hatayı
    % veren ağırlıkları olarak alınıyor.
    D=agirliklar_3(:,j);
    % SVM'ye eğitim için giriş ve hedef verileri tanıtılarak bir SVM
    % sınıflayıcı yapı oluşturuluyor. Daha sonra bu yapı test
    % edilerek yapının geçici hatası elde ediliyor.
    svmStruct(t,j) = svmtrain(giris_egitim(:,j),hedef_egitim(:,1));
    siniflandirma_sonucu= svmclassify(svmStruct(t,j),giris_egitim(:,j));
    temp = sum(D.*(siniflandirma_sonucu ~= hedef_egitim));
    if temp < err(t)
      err(t) = temp;
      en_iyi_ozellik(t,1) =j;
    end
       if(err(t) >= 1/2)
      disp('stop b/c err>1/2')
      t
      j
      break;
```

```
        end
           if err(t) == 0
            err(t) = 0.000001;
        end
        alpha(j) = 1/2 * log((1-err(t))/err(t));
        D = D.* exp((-alpha(j).*hedef_egitim.* svmclassify(svmStruct(t, j),giris_egitim(:,j))).^(1.5));
        D = D./sum(D);
        agirliklar_4(:,j) = D;
      end
       alfa(t)=alpha(en_iyi_ozellik(t,1));
       agirliklar_3 = agirliklar_4;
       agirliklar_3 = abs(agirliklar_3);
  end
% Eğitim başarısı test edilirken kullanılacak satır sayısı elde ediliyor.
[x y] = size(giris_egitim);
  % Her bir SVM yapısına eğitim verisi verilerek yapılar test ediliyor.
  % Öncelikle parametreleri yukarıda eğitimle belirlenmiş olan birinci
  % temel öğrenici ile her özellik ayrı ayrı test ediliyor. Her özelliğin
  % test sonucu toplanıyor. Sign fonksiyonu bu sonuçlar {-1,+1} formatına
  % getirilerek birinci temel sınıflandırıcının sonucu (X x 1 boyutunda)
  % elde edilip, başarısı hesaplanıyor.
  for j=1:t
     finalLabel = zeros(size(hedef_egitim));
     for k=1:w;  % Özellik sayısı kadar.
        finalLabel = alfa(j) * svmclassify(svmStruct(j, k), giris_egitim(:,k));
        tfinalLabel = sign(finalLabel);
        son_2(:,k) = tfinalLabel;
     end
       for o=1:x % (satır sayısı)
        bir_sayisi=0;
        for p=1:w
            if son_2(o,p) == 1
              bir_sayisi = bir_sayisi+1;
            end
        end
        if bir_sayisi > (y/2)
           tfinalLabel(o,1) =1;
           % Her satırdaki "1" sayısı "-1" sayısından büyükse o satır
           % yani örüntü +1 demektir.
        else
           tfinalLabel(o,1) =-1;
           % Değilse sonuç -1 demektir.
        end
     end

     dogru=0; yanlis=0;
      for l=1:x
           if tfinalLabel(l,1) == hedef_egitim(l,1)

            dogru= dogru+1;
        else
            yanlis=yanlis+1;
        end
     end
     % "t" adet temel öğrenicinin başarısı tutuluyor. "t" adet temel
     % öğrenicinin başarısı egitim_basari matrisi ile tutuluyor. Bu
     % matriste 9 adet değer tutuluyor.
     egitim_basari(1,j)= dogru *100/x;
  end
```

```
% cross-validation yapıldığı için, her bir cross-validation için
% 9 temel öğrenicinin ortalaması alınarak, cross-validation adedince
% eğitim başarısı elde ediliyor.
egitim_basari_son(1,n)=mean(egitim_basari);
% Her bir SVM yapısına test verisi verilerek yapılar test ediliyor.
% Öncelikle parametreleri yukarıda eğitimle belirlenmiş olan birinci
% temel öğrenici ile eğitimde hiç kullanılmamış olan, sisteme tamamen
% yeni olan her özellik, ayrı ayrı test ediliyor. Her özelliğin test
% sonucu toplanıyor. Sign fonksiyonu bu sonuçlar {-1,+1} formatına
% getirilerek birinci temel sınıflandırıcının sonucu elde edilip,
% başarısı hesaplanıyor.
for m=1:t
    finalLabel_t = zeros(size(hedef_test));
   for a=1:w;  % Özellik sayısı kadar.
      finalLabel_t = alfa(m) * svmclassify(svmStruct(m, a), giris_test(:,a));
      tfinalLabel_t = sign(finalLabel_t);
      son_t_2(:,a) = tfinalLabel_t;
      % Her özellik değeri -1,+1 olarak toplanmadan tutuluyor
   end
   for o=1:size(hedef_test) % (satır sayısı)
      bir_sayisi=0;
      for y=1:w
          if son_t_2(o,y) == 1
            bir_sayisi = bir_sayisi+1;
          end
      end
      [A B] = size(giris_test);
      if bir_sayisi > (B/2)
         tfinalLabel_t(o,1) =1;
         % Her satırdaki "1" sayısı "-1" sayısından büyükse o satır
         % yani örüntü +1 demektir.
      else
         tfinalLabel_t(o,1) =-1;
         % Değilse sonuç -1 demektir.
      end
   end
         % Sistemin duyarlılık, belirlilik, doğruluk değerlerinin elde
   % edilmesi için her bir temel sınıflandırıcının confusion
   % matrisleri elde ediliyor.
   TP_rbb=0; TN_rbb=0;
   FP_rbb=0; FN_rbb=0;
       for p=1:size(hedef_test)
           if  hedef_test(p,1)== 1
                if tfinalLabel_t(p,1) ==1
                  TP_rbb = TP_rbb+1;
                end
                if tfinalLabel_t(p,1) ==-1
                  FN_rbb= FN_rbb+1;
                end
           end
            if  hedef_test(p,1)== -1
                if tfinalLabel_t(p,1) ==1
                     FP_rbb = FP_rbb+1;
                end
                if tfinalLabel_t(p,1) ==-1
                     TN_rbb= TN_rbb+1;
                end
            end
       end
```

```
        sensitivity(1,m)= TP_rbb / (TP_rbb +FN_rbb)*100;
        specifity(1,m)  = TN_rbb/ (TN_rbb + FP_rbb) *100;
        accuracy(1,m)   = (TP_rbb+ TN_rbb) / (TP_rbb +FP_rbb +TN_rbb +FN_rbb)*100;
    end
    % Her cross-validation için temel öğrenicilerin ortalama başarısı
    % tutuluyor.
    sensitivity_s(1,n)= mean(sensitivity);
    specifity_s(1,n)  = mean(specifity);
    accuracy_s(1,n)   = mean(accuracy);
end % cross-validation sonu
% cross-validation adedince elde edilen sonuçların ortalaması alınarak
% tek bir test başarısı elde ediliyor.
egitim_basari_2 = mean(egitim_basari_son);
sensitivity_son_2= mean(sensitivity_s);
specifity_son_2  = mean(specifity_s);
accuracy_son_2   = mean(accuracy_s);
giris_son = [];
   train_target=zeros(848,1);
train_target(1:90,1)=1;
train_target(91:848,1)=-1;
hedef=train_target;
    %%% ECG Sol Dal Bloku ile Normalin Ayrımı
for q=0:25:175
    % 10'arlı gruplar hâlinde özellikler değerlendiriliyor
    giris=[ecglbb(q+1:q+25,1:90)'; ecgnormal(q+1:q+25,1:758)'];
    ozellik =zeros(25,1);
    son_veri = zeros(848,25);
    b=0; % Çıkartılan özellik sayısı
     for i=1:10
      c = cvpartition(hedef,'holdout',.1);
      % Her veri eğitim ve testte kullanılarak en etkin özellikler seçiliyor
     giris_egitim = giris(training(c,1),:);
     hedef_egitim = hedef(training(c,1));
     giris_test = giris(test(c,1), :);
     hedef_test = hedef(test(c,1));
     f = @(giris_egitim, hedef_egitim, giris_test, hedef_test) sum(hedef_test ~= classify(giris_test,
giris_egitim, hedef_egitim));
      fs = sequentialfs(f,giris,hedef);
       for k=1:25
       % Her seçim işlemi sonucunda seçilen özelliklerin
       % kaç defa seçildiği kaydediliyor.
          if fs(k) == 1
            ozellik(k)= ozellik(k)+1;
          end
       end
    end
    % Seçilen özellikler ile yeni giriş verisi oluşturuluyor
    for m=1:25
     if ozellik(m) >= 3 % 10 veya daha fazla seçilen özellikler
        b=b+1;
        son_veri(:,b) = giris(:,m);
     end
    end
    y=find(son_veri(1,:));
    giris_s= son_veri(:,y);

    giris_son = [giris_son giris_s];
end
  giris=giris_son;
  [S, sayi_ozellik_3] = size(giris);
```

```
sayi_ozellik_3
  for n=1:1:10 %Cross-validation'ın başladığı yer.
 c = cvpartition(hedef,'holdout',.1);
    % Sınıf bilgisi hedef_test kullanılarak eğitim seti ile test seti
    % rastgele parçalara ayırıldı. Bölme oranı %10. Yani 683 adet verinin
    % %90'ı eğitime, %10'u teste ayrılıyor.
% Özellik sayısı "w" bulunuyor.
[h w] = size(giris);
    % Eğitim seti oluşturuluyor.

giris_egitim = giris(training(c,1),:);
hedef_egitim = hedef(training(c,1));
% Test seti oluşturuluyor.
giris_test = giris(test(c,1), :);
hedef_test = hedef(test(c,1));
    err(1) = inf;
    % İlk hata sonsuz alınıyor. Daha sonra geçici hata ile değişiyor.
       for j=1:w;

         % Her bir özelliğin ilk ağırlıkları oluşturuluyor.
         D = ones(size(hedef_egitim))/length(hedef_egitim);

         % SVM'ye eğitim için giriş ve hedef verileri tanıtılarak bir SVM
         % sınıflayıcı yapı oluşturuluyor. Daha sonra bu yapı test
         % edilerek yapının geçici hatası elde ediliyor.
         svmStruct(1,j) = svmtrain(giris_egitim(:,j),hedef_egitim(:,1));
         siniflandirma_sonucu= svmclassify(svmStruct(1,j),giris_egitim(:,j));
         temp = sum(D.*(siniflandirma_sonucu ~= hedef_egitim));   % özelliğin sınıflandırılma hatası
         % Geçici hata bir önceki hatadan küçük ise geçici hata, yeni hata
         % oluyor. En iyi özellikte ise en az hatayı veren özellik tutuluyor.
         if temp < err(1)
           err(1) = temp;
           en_iyi_ozellik(1,1) =j;
         end
         % Hata 0.5'ten büyük ise algoritma gereği işlem sonlandırılıyor.
         if(err(1) >= 1/2)
           disp('stop b/c err>1/2')
           break;
         end
         % Hata 0 ise 10^-6 yapılıyor. Sebebi, hata 0 olduğunda alpha
         % değerinin sonsuz olması.
         if err(1) == 0
            err(1) = 0.000001;
         end
                  % Her bir özellik için ağırlık hesaplanıyor.
         alpha(j) = 1/2 * log((1-err(1))/err(1));
                        % Elde edilen yeni ağırlıkla, D ağırlıkları güncelleniyor.
         D = D.* exp((-alpha(j).*hedef_egitim.* svmclassify(svmStruct(1, j),giris_egitim(:,j))).^(1.5));
         D = D./sum(D);
         % Güncellenen D ağırlıkları yeni bir matriste tutuluyor. Bu
         % ağırlıklar ilk yapıya âit. İkinci yapı, yani ikinci temel
         % öğrenici sınıflandırma yaparken her bir özelliğinin ilk
         % ağırlığı olarak, ilk temel sınıflandırıcının son ağırlıklarını
         % alacak. Bu şekilde her seferinde ilk ağırlık
         % D= 1/ length(hedef_egitim) olmayacak. Her özelliğin kendi
         % ağırlığınca D değeri güncellenecek.
         %
         agirliklar_5(:,j) = D;
         agirliklar_5(:,j) = abs(agirliklar_5(:,j));
    end
```

```
    % Birinci temel öğrenicinin ağırlığı tutuluyor. Bu ağırlık en az
    % hatayı veren özelliğin ağırlığı.
   alfa(1)=alpha(en_iyi_ozellik(1,1));
   for t=1:T
      % Her temel öğrenicinin ilk hatası her zaman sonsuz olarak
      % başlayıp, daha sonra geçici hata ile güncelleniyor.
      err(t)=inf;
      for j=1:w
       % İlk ağırlıklar, bir önceki temel sınıflandırıcının en az hatayı
       % veren ağırlıkları olarak alınıyor.
       D=agirliklar_5(:,j);
       % SVM'ye eğitim için giriş ve hedef verileri tanıtılarak bir SVM
       % sınıflayıcı yapı oluşturuluyor. Daha sonra bu yapı test
       % edilerek yapının geçici hatası elde ediliyor.
       svmStruct(t,j) = svmtrain(giris_egitim(:,j),hedef_egitim(:,1));
       siniflandirma_sonucu= svmclassify(svmStruct(t,j),giris_egitim(:,j));
       temp = sum(D.*(siniflandirma_sonucu ~= hedef_egitim));
       if temp < err(t)
          err(t) = temp;
          en_iyi_ozellik(t,1) =j;
       end
       if(err(t) >= 1/2)
          disp('stop b/c err>1/2')
          t
          j
          break;
       end
       if err(t) == 0
           err(t) = 0.000001;
       end
       alpha(j) = 1/2 * log((1-err(t))/err(t));
       D = D.* exp((-alpha(j).*hedef_egitim.* svmclassify(svmStruct(t, j),giris_egitim(:,j))).^(1.5));
       D = D./sum(D);
       agirliklar_6(:,j) = D;
      end
       alfa(t)=alpha(en_iyi_ozellik(t,1));
       agirliklar_5 = agirliklar_6;
       agirliklar_5 = abs(agirliklar_5);
   end
% Eğitim başarısı test edilirken kullanılacak satır sayısı elde ediliyor.
[x y] = size(giris_egitim);
   % Her bir SVM yapısına eğitim verisi verilerek yapılar test ediliyor.
   % Öncelikle parametreleri yukarıda eğitimle belirlenmiş olan birinci
   % temel öğrenici ile her özellik ayrı ayrı test ediliyor. Her özelliğin
   % test sonucu toplanıyor. Sign fonksiyonu bu sonuçlar {-1,+1} formatına
   % getirilerek birinci temel sınıflandırıcının sonucu (X x 1 boyutunda)
   % elde edilip, başarısı hesaplanıyor.
   for j=1:t
      finalLabel = zeros(size(hedef_egitim));
      for k=1:w;  % Özellik sayısı kadar.
         finalLabel = alfa(j) * svmclassify(svmStruct(j, k), giris_egitim(:,k));
         tfinalLabel = sign(finalLabel);
         son_3(:,k) = tfinalLabel;
      end
      for o=1:x % (satır sayısı)
         bir_sayisi=0;
         for p=1:w
             if son_3(o,p) == 1
                bir_sayisi = bir_sayisi+1;
             end
```

```
        end
        if bir_sayisi > (y/2)
            tfinalLabel(o,1) =1;
            % Her satırdaki "1" sayısı "-1" sayısından büyükse o satır
            % yani örüntü +1 demektir.
        else
            tfinalLabel(o,1) =-1;
            % Değilse sonuç -1 demektir.
        end
    end
    dogru=0; yanlis=0;
    for l=1:x
        if tfinalLabel(l,1) == hedef_egitim(l,1)
            dogru= dogru+1;
        else
            yanlis=yanlis+1;
        end
    end
    % "t" adet temel öğrenicinin başarısı tutuluyor. "t" adet temel
    % öğrenicinin başarısı egitim_basari matrisi ile tutuluyor. Bu
    % matriste 9 adet değer tutuluyor.
    egitim_basari(1,j)= dogru *100/x;
end
% cross-validation yapıldığı için, her bir cross-validation için
% 9 temel öğrenicinin ortalaması alınarak, cross-validation adedince
% eğitim başarısı elde ediliyor.
egitim_basari_son(1,n)=mean(egitim_basari);
% Her bir SVM yapısına test verisi verilerek yapılar test ediliyor.
% Öncelikle parametreleri yukarıda eğitimle belirlenmiş olan birinci
% temel öğrenici ile eğitimde hiç kullanılmamış olan, sisteme tamamen
% yeni olan her özellik, ayrı ayrı test ediliyor. Her özelliğin test
% sonucu toplanıyor. Sign fonksiyonu bu sonuçlar {-1,+1} formatına
% getirilerek birinci temel sınıflandırıcının sonucu elde edilip,
% başarısı hesaplanıyor.
for m=1:t
     finalLabel_t = zeros(size(hedef_test));
    for a=1:w;  % Özellik sayısı kadar.
        finalLabel_t = alfa(m) * svmclassify(svmStruct(m, a), giris_test(:,a));
        tfinalLabel_t = sign(finalLabel_t);
        son_t_3(:,a) = tfinalLabel_t;
        % Her özellik değeri -1,+1 olarak toplanmadan tutuluyor
    end
    for o=1:size(hedef_test) % (satır sayısı)
        bir_sayisi=0;
        for y=1:w
            if son_t_3(o,y) == 1
              bir_sayisi = bir_sayisi+1;
            end
        end
        [A B] = size(giris_test);
        if bir_sayisi > (B/2)
            tfinalLabel_t(o,1) =1;
            % Her satırdaki "1" sayısı "-1" sayısından büyükse o satır
            % yani örüntü +1 demektir.
        else
            tfinalLabel_t(o,1) =-1;
            % Değilse sonuç -1 demektir.
        end

    end
```

```
        % Sistemin duyarlılık, belirlilik, doğruluk değerlerinin elde
        % edilmesi için her bir temel sınıflandırıcının confusion
        % matrisleri elde ediliyor.
        TP=0; TN=0;
        FP=0; FN=0;
            for p=1:size(hedef_test)
                if  hedef_test(p,1)== 1
                    if tfinalLabel_t(p,1) ==1
                      TP = TP+1;
                    end
                    if tfinalLabel_t(p,1) ==-1
                      FN= FN+1;
                    end
                end
                 if  hedef_test(p,1)== -1
                    if tfinalLabel_t(p,1) ==1
                        FP = FP+1;
                    end
                    if tfinalLabel_t(p,1) ==-1
                        TN= TN+1;
                    end
                 end
            end
        sensitivity(1,m)= TP / (TP +FN)*100;
        specifity(1,m)  = TN/ (TN + FP) *100;
        accuracy(1,m)   = (TP+ TN) / (TP +FP +TN +FN)*100;
    end
        % Her cross-validation için temel öğrenicilerin ortalama başarısı
    % tutuluyor.
    sensitivity_s(1,n)= mean(sensitivity);
    specifity_s(1,n)  = mean(specifity);
    accuracy_s(1,n)   = mean(accuracy);
end % cross-validation sonu
% cross-validation adedince elde edilen sonuçların ortalaması alınarak
% tek bir test başarısı elde ediliyor.
egitim_basari_3 = mean(egitim_basari_son);
sensitivity_son_3= mean(sensitivity_s);
specifity_son_3  = mean(specifity_s);
accuracy_son_3   = mean(accuracy_s);
basari_egitim= (egitim_basari_3 + egitim_basari_2 + egitim_basari_1)/3
basari_sens = (sensitivity_son_3 + sensitivity_son_2 + sensitivity_son_1)/3
basari_spec = (specifity_son_3 + specifity_son_2  +specifity_son_1)/3
basari_acc  = (accuracy_son_3 + accuracy_son_2 + accuracy_son_1)/3
toc
```

EK-2 TBA Yazılımı

```
function [Y, lbd, cos2, CC] = pca(X, P, labels)
[N, L] = size(X) ;
if (P < 1)
  error('PCANA: must return at least 1 principal component.')
end
if (P > L)
  error('PCANA: can"t get more principal components than dimensions.')
end
if (nargin == 2)
  verbose = 0 ;
end
R = corrcoef(X) ;
[V, S] = eig(R) ;
[l, idx] = sort(diag(S)') ;
l = fliplr(l) ;
idx = fliplr(idx) ;
% Coordonees centrees reduites.
XCR = (X - ones(N,1)*mean(X)) / diag(std(X)) ;
PC = (XCR * V(:,idx) ) ;
% P first principal components:
Y = PC(:, 1:P) ;
% P first portions of inertia:
lbd = l(1:P) / sum(l) ;
% Squared cosines of projection vs. original data:
cos2 = sum((Y.^2)') ./ sum((XCR.^2)') ;
% Correlations of features with PCs:
CC = V(:,idx) * diag(sqrt(l)) ;
if (nargin == 3)
  figure
  subplot(2,2,1) ;
  plot(PC(:,1), PC(:,2), '+') ;
  if (size(labels, 1) == N)
    for i = 1:N
      text(PC(i,1), PC(i,2), [' ', labels(i,:)]) ;
    end
  end
  xlabel('First principal axis') ;
  ylabel('Second principal axis') ;
  title('Data projection on the first two principal axes')
  subplot(2,2,2) ;
  plot(cos(2*pi*(0:100)/100), sin(2*pi*(0:100)/100)) ;
  hold on, plot([-1, 1], [0, 0]), plot([0, 0], [-1, 1]) ;
  plot(CC(:,1), CC(:,2), 'x') ;
  for k = 1:L
    text(CC(k,1)+.03, CC(k,2)+.03, num2str(k)) ;
  end
  xlabel('First PC') ;
  ylabel('Second PC') ;
  title('Correlation of features with first 2 PCs')
  subplot(2,2,3) ;
  hh = plot(l/sum(l), '--') ;
  set(hh, 'LineWidth', 3) ;
  hold on, hh = plot(cumsum(l)/sum(l)) ;
  set(hh, 'LineWidth', 3) ;
  axis([1 L 0 1]) ;
  plot([2 2], [0 1]) ;
  xlabel('PC no.') ;
  ylabel('Inertia') ;
```

```
  legend('PC inertia', 'Cumulated inertia') ;
  title('Repartition of inertia on the PCs') ;
  subplot(2,2,4) ;
  hold on
  hh = plot(acos(sqrt(cos2))/pi*180, 1:N, '+') ;
  set(hh, 'LineWidth', 3) ;
  hold on
  for i = 1:N
    hh = plot([0, acos(sqrt(cos2(i)))/pi*180], [i, i], '-') ;
    set(hh, 'LineWidth', 3) ;
  end
  set(gca, 'Box', 'on') ;
  set(gca, 'YLim', [1 N]) ;
  if (size(labels, 1) == N)
    set(gca, 'YTick', 1:N) ;
    set(gca, 'YTickLabels', labels) ;
  end
  xlabel('Angle (deg)')
  title('Angle of data with the PC plan')
end
CC = CC(:,1:P) ;
```

EK-3 TBA-AdaBoost-DVM Yapısı ile Göğüs Kanseri Verisi Sınıflandırma Yazılımı

```
tic
clear all
clc
veri = xlsread('breastcancer.xlsx'); % Sınıflandırılacak veri
giris = veri(:,1:9);
Y = pca(giris, 4); % 4 özellik çıkarıldı
veriseti = [Y veri(:,10)];
giris = veriseti(:,1:4);
hedef = veriseti(:,5);
% Sınıflandırılma yapacak göğüs kanseri verisinin sınıfları SVM için {-1,+1}
% olarak değiştirildi. AdaBoost da {-1,+1} olarak çalıştığı için sınıflarda
% başka bir % değişiklik yapılmadı.
[N M] = size(giris);
% Verinin girişleri 0 ve 1 aralığına normalize ediliyor
giris=(giris-repmat(min(giris),N,1))./(repmat(max(giris),N,1)-repmat(min(giris),N,1));
  for m=1:683

      if hedef(m,1) == 2
         hedef(m,1) =1;
      end

      if hedef(m,1) == 4
         hedef(m,1) = -1;
      end
  end
alpha = [];
% Her bir zayıf öğrenicinin ağırlığı
% Alfa değeri ise her bir temel öğrenicinin ağırlığı olarak kullanılıyor.
% En iyi alpha değerini tutuyor.
for s=1:1 % cross-validation 10 defa çalıştırılacak.
for n=1:1:10 %Cross-validation'ın başladığı yer.
 c = cvpartition(hedef,'holdout',.1);
    % Sınıf bilgisi hedef_test kullanılarak eğitim seti ile test seti
    % rastgele parçalara ayırıldı. Bölme oranı %10. Yani 683 adet verinin
    % %90'ı eğitime, %10'u teste ayrılıyor.
% Özellik sayısı "w" bulunuyor.
[h w] = size(giris);
% Eğitim seti oluşturuluyor.
giris_egitim = giris(training(c,1),:);
hedef_egitim = hedef(training(c,1));
 % Test seti oluşturuluyor.
giris_test = giris(test(c,1), :);
hedef_test = hedef(test(c,1));
    err(1) = inf;
    % İlk hata sonsuz alınıyor. Daha sonra geçici hata ile değişiyor.
    for j=1:w;
        % Her bir özelliğin ilk ağırlıkları oluşturuluyor.
        D = ones(size(hedef_egitim))/length(hedef_egitim);
        % SVM'ye eğitim için giriş ve hedef verileri tanıtılarak bir SVM
        % sınıflayıcı yapı oluşturuluyor. Daha sonra bu yapı test
        % edilerek yapının geçici hatası elde ediliyor.
        svmStruct(1,j) = svmtrain(giris_egitim(:,j),hedef_egitim(:,1));
        siniflandirma_sonucu= svmclassify(svmStruct(1,j),giris_egitim(:,j));
        temp = sum(D.*(siniflandirma_sonucu ~= hedef_egitim));   % özelliğin sınıflandırılma hatası
        % Geçici hata bir önceki hatadan küçük ise geçici hata, yeni hata
        % oluyor. En iyi özellikte ise en az hatayı veren özellik tutuluyor.
        if temp < err(1)
          err(1) = temp;
```

```
        en_iyi_ozellik(1,1) =j;
      end
      % Hata 0.5'ten büyük ise algoritma gereği işlem sonlandırılıyor.
      if(err(1) >= 1/2)
        disp('stop b/c err>1/2')
        break;
      end
      % Hata 0 ise 10^-6 yapılıyor. Sebebi, hata 0 olduğunda alpha
      % değerinin sonsuz olması.
      if err(1) == 0
         err(1) = 0.000001;
      end
      % Her bir özellik için ağırlık hesaplanıyor.
      alpha(j) = 1/2 * log((1-err(1))/err(1));
      % Elde edilen yeni ağırlıkla, D ağırlıkları güncelleniyor.
      D = D.* exp((-alpha(j).*hedef_egitim.* svmclassify(svmStruct(1, j),giris_egitim(:,j))).^(0.4));
      D = D./sum(D);
      % Güncellenen D ağırlıkları yeni bir matriste tutuluyor. Bu
      % ağırlıklar ilk yapıya âit. İkinci yapı, yani ikinci temel
      % öğrenici sınıflandırma yaparken her bir özelliğinin ilk
      % ağırlığı olarak, ilk temel sınıflandırıcının son ağırlıklarını
      % alacak. Bu şekilde her seferinde ilk ağırlık
      % D= 1/ length(hedef_egitim) olmayacak. Her özelliğin kendi
      % ağırlığınca D değeri güncellenecek.
      %
      agirliklar(:,j) = D;
      agirliklar(:,j) = abs(agirliklar(:,j));
end
 % Birinci temel öğrenicinin ağırlığı tutuluyor. Bu ağırlık en az
 % hatayı veren özelliğin ağırlığı.
alfa(1)=alpha(en_iyi_ozellik(1,1));
for t=2:w
    % Her temel öğrenicinin ilk hatası her zaman sonsuz olarak
    % başlayıp, daha sonra geçici hata ile güncelleniyor.
    err(t)=inf;
    for j=1:w
     % İlk ağırlıklar, bir önceki temel sınıflandırıcının en az hatayı
     % veren ağırlıkları olarak alınıyor.
     D=agirliklar(:,w);
     % SVM'ye eğitim için giriş ve hedef verileri tanıtılarak bir SVM
     % sınıflayıcı yapı oluşturuluyor. Daha sonra bu yapı test
     % edilerek yapının geçici hatası elde ediliyor.
     svmStruct(t,j) = svmtrain(giris_egitim(:,j),hedef_egitim(:,1));
     siniflandirma_sonucu= svmclassify(svmStruct(t,j),giris_egitim(:,j));
     temp = sum(D.*(siniflandirma_sonucu ~= hedef_egitim));
     if temp < err(t)
        err(t) = temp;
        en_iyi_ozellik(t,1) =j;
     end
     if(err(t) >= 1/2)
        disp('stop b/c err>1/2')
        break;
     end
     if err(t) == 0
         err(t) = 0.000001;
     end
     alpha(j) = 1/2 * log((1-err(t))/err(t));
     D = D.* exp((-alpha(j).*hedef_egitim.* svmclassify(svmStruct(t, j),giris_egitim(:,j))).^(0.4));
     D = D./sum(D);
     agirliklar_2(:,j) = D;
```

```
        end
        alfa(t)=alpha(en_iyi_ozellik(t,1));
        agirliklar = agirliklar_2;
        agirliklar = abs(agirliklar);
    end
% Eğitim başarısı test edilirken kullanılacak satır sayısı elde ediliyor.
[x y] = size(giris_egitim);
    % Her bir SVM yapısına eğitim verisi verilerek yapılar test ediliyor.
    % Öncelikle parametreleri yukarıda eğitimle belirlenmiş olan birinci
    % temel öğrenici ile 9 özellik ayrı ayrı test ediliyor. Her özelliğin
    % test sonucu toplanıyor. Sign fonksiyonu bu sonuçlar {-1,+1} formatına
    % getirilerek birinci temel sınıflandırıcının sonucu (X x 1 boyutunda)
    % elde edilip, başarısı hesaplanıyor.
    for j=1:t
        finalLabel = zeros(size(hedef_egitim));
        for k=1:w;  % Özellik sayısı kadar.
            finalLabel = finalLabel + alfa(j) * svmclassify(svmStruct(j, k), giris_egitim(:,k));
            tfinalLabel = sign(finalLabel);
        end
        dogru=0; yanlis=0;
        for l=1:x
            if tfinalLabel(l,1) == hedef_egitim(l,1)
                dogru= dogru+1;
            else
                yanlis=yanlis+1;
            end
        end
        % "t" adet temel öğrenicinin başarısı tutuluyor. "t" adet temel
        % öğrenicinin başarısı egitim_basari matrisi ile tutuluyor. Bu
        % matriste 9 adet değer tutuluyor.
        egitim_basari(1,j)= dogru *100/x;
    end
    % cross-validation yapıldığı için, her bir cross-validation için
    % 9 temel öğrenicinin ortalaması alınarak, cross-validation adedince
    % eğitim başarısı elde ediliyor.
    egitim_basari_son(1,n)=mean(egitim_basari);
    % Her bir SVM yapısına test verisi verilerek yapılar test ediliyor.
    % Öncelikle parametreleri yukarıda eğitimle belirlenmiş olan birinci
    % temel öğrenici ile eğitimde hiç kullanılmamış olan, sisteme tamamen
    % yeni olan 9 özellik, ayrı ayrı test ediliyor. Her özelliğin test
    % sonucu toplanıyor. Sign fonksiyonu bu sonuçlar {-1,+1} formatına
    % getirilerek birinci temel sınıflandırıcının sonucu elde edilip,
    % başarısı hesaplanıyor.
        for m=1:t
         finalLabel_t = zeros(size(hedef_test));
        for a=1:w;  % Özellik sayısı kadar.
            finalLabel_t = finalLabel_t + alpha(m) * svmclassify(svmStruct(m, a), giris_test(:,a));
            tfinalLabel_t = sign(finalLabel_t);
        end
        % Sistemin duyarlılık, belirlilik, doğruluk değerlerinin elde
        % edilmesi için her bir temel sınıflandırıcının confusion
        % matrisleri elde ediliyor.
        TP=0; TN=0;
        FP=0; FN=0;
            for p=1:size(hedef_test)
                if  hedef_test(p,1)== 1
                    if tfinalLabel_t(p,1) ==1
                      TP = TP+1;
                    end
                    if tfinalLabel_t(p,1) ==-1
```

```
                    FN= FN+1;
                end
            end
            if  hedef_test(p,1)== -1
                if tfinalLabel_t(p,1) ==1
                    FP = FP+1;
                end
                if tfinalLabel_t(p,1) ==-1
                    TN= TN+1;
                end
            end
        end
        sensitivity(1,m)= TP / (TP +FN)*100;
        specifity(1,m)  = TN/ (TN + FP) *100;
        accuracy(1,m)   = (TP+ TN) / (TP +FP +TN +FN)*100;
    end
    % Her cross-validation için temel öğrenicilerin ortalama başarısı
    % tutuluyor.
    sensitivity_s(1,n)= mean(sensitivity);
    specifity_s(1,n)  = mean(specifity);
    accuracy_s(1,n)   = mean(accuracy);
end  % cross-validation sonu
% cross-validation adedince elde edilen sonuçların ortalaması alınarak
% tek bir test başarısı elde ediliyor.
egitim_basari(1,s) = mean(egitim_basari_son);
sensitivity_son(1,s)= mean(sensitivity_s);
specifity_son(1,s)  = mean(specifity_s);
accuracy_son(1,s)   = mean(accuracy_s);
end % 10 defa çalıştırılan cross-validation sonu
egitim_basari_son  = mean(egitim_basari)
basari_sensitivity = mean(sensitivity_son)
basari_specifity   = mean(specifity_son)
basari_accuracy    = mean(accuracy_son)
toc
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Mücahid Barstuğan
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Samsun / 24.09.1988
Telefon	**:**	0536 694 55 72
Faks	**:**	
e-mail	**:**	mubarstugan@selcuk.edu.tr

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	**Düzce Fen Lisesi, Merkez, DÜZCE**	**2006**
Üniversite	:	**Kocaeli Üniversitesi, İzmit, KOCAELİ**	**2010**
Yüksek Lisans	:	**Selçuk Üniversitesi, Selçuklu, KONYA**	**2014**
Doktora	:		

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2011	**Na-De Elektronik A.Ş.**	**Ar-Ge Mühendisi**

UZMANLIK ALANI

YABANCI DİLLER

Japonca (YDS 2014 – 30)
İngilizce (KPDS 2010 -71)