T.C.
VAN YÜZÜNCÜ YIL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ZOOTEKNİ ANABİLİM DALI

LİSANS YERLEŞTİRME SINAVINDA YERLEŞME BAŞARISININ KARAR AĞAÇLARINA GÖRE 'BAGGING' VE 'BOOSTING' YÖNTEMLERİYLE SINIFLANDIRILMASI

DOKTORA TEZİ

HAZIRLAYAN: Tuğba TUĞ KAROĞLU
DANIŞMAN: Prof. Dr. Hayrettin OKUT

VAN-2018

T.C.
VAN YÜZÜNCÜ YIL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ZOOTEKNİ ANABİLİM DALI

LİSANS YERLEŞTİRME SINAVINDA YERLEŞME BAŞARISININ KARAR AĞAÇLARINA GÖRE 'BAGGING' VE 'BOOSTING' YÖNTEMLERİYLE SINIFLANDIRILMASI

DOKTORA TEZİ

HAZIRLAYAN: Tuğba TUĞ KAROĞLU

VAN-2018

KABUL VE ONAY SAYFASI

Zootekni Anabilim Dalı'nda Prof. Dr. Hayrettin OKUT danışmanlığında, Tuğba TUĞ KAROĞLU tarafından sunulan"Lisans Yerleştirme Sınavında Yerleşme Başarısının Karar Ağaçlarına Göre "Bagging" ve "Boosting" Yöntemleriyle Sınıflandırılması" isimli bu çalışma "Lisansüstü Eğitim ve Öğretim Yönetmeliği" ve "Fen Bilimleri Enstitüsü Yönergesi" nin ilgili hükümleri gereğince 04/06/2018 tarihinde aşağıdaki jüri tarafından oy birliği/~~oy çokluğu~~ ile başarılı bulunmuş ve doktora tezi olarak kabul edilmiştir.

Başkan: Prof. Dr. Hayrettin OKUT İmza:

Üye: Prof. Dr. Abdullah YEDİLOVA İmza:

Üye: Doç. Dr. Yakut GEVREKÇİ İmza:

Üye: Doç. Dr. M. Nuri ALMALI İmza:

Üye: Dr. Öğr. Üyesi Barış KAKİ İmza:

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 22/06/2018 tarih ve 2018/28-I...sayılı kararı ile onaylanmıştır.

İmza

Prof. Dr. Sıtkı SENSOY

Enstitü Müdürü

Enstitü Müdürü

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Tuğba TUĞ KAROĞLU

ÖZET

LİSANS YERLEŞTİRME SINAVINDA YERLEŞME BAŞARISININ KARAR AĞAÇLARINA GÖRE 'BAGGING' VE 'BOOSTING' YÖNTEMLERİYLE SINIFLANDIRILMASI

TUĞ KAROĞLU, Tuğba
Doktora Tezi, Zootekni Anabilim Dalı
Tez Danışmanı : Prof. Dr. Hayrettin OKUT
HAZİRAN 2018, 99 sayfa

Bu tez çalışması; 2010-2013 yılları arasında Türkiye'nin 81 ilinde Öğrenci Seçme ve Yerleştirme Merkezi (ÖSYM) tarafından düzenlenen Yükseköğrenime Giriş Sınavına (YGS) giren; 180 ve üzeri puan alarak ikinci basamak sınavı olan Lisans Yerleştirme Sınavına (LYS) girmeye hak kazanan öğrencilerin Ensemble modellerinden ikisi olan "Bagging" ve "Boosting" yöntemleri ile sınıflandırılmasının yapılması amacıyla gerçekleştirilmiştir.

Bu amaçla Türkiye İstatistik Kurumu (TÜİK)'ten ve ÖSYM arşivinden değişkenlerle ilgili veriler toplanmıştır. ÖSYM sınav kılavuzu incelenerek değerlendirmelerin yapılmasında bu kılavuzdaki esaslar dikkate alınmıştır.

Literatür taramaları yapılmış, veriler toplanmış ve MATLAB bilgisayar programında Baggingve Boosting yöntemleriyle sınıflandırma işlemleri gerçekleştirilmiş, R bilgisayar programında da Kümeleme Analizi yapılarak desteklenmiştir. Hiyerarşik kümeleme sonuçları Doğu Anadolu'daki illerin kendi arasında, Marmara, Batı Ege ve Batı Akdeniz Bölgeleri'ndeki illerin ise kendi arasında gruplaşmış olduğu gözlenmiştir. İç Anadolu ile Orta ve Doğu Karadenizde bulunan iller ile Gaziantep, Malatya ve Elazığ benzerlikleri ayrıca dikkat çekicidir.

Çalışma sonucunda "Bagging" ve "Boosting" yöntemleri sınıflandırma sonucundaki performansları karşılaştırılmış ve ayrıca kümeleme analizi ile yapılan sınıflandırmalar desteklenmiştir. Buna göre; "Bagging" ve "Boosting" yöntemlerinin hatayı düşürmede iyi bir performans sergiledikleri belirlenmiştir.

Anahtar kelimler: AdaBoost, Bagging, Boosting, Bootstrap, Ensemble Modeller, Kümeleme.

ABSTRACT

CLASSIFICATION OF REPLACEMENT SUCCESS ACCORDING TO DECISION TREES BY USING "BAGGING" AND "BOOSTING" METHODS

TUĞ KAROĞLU, Tuğba
Ph.D. Thesis, Animal Science
Supervisor: Prof. Dr. Hayrettin OKUT
June 2018, 99 pages

This study was conduted to classify the student those took Undergraduate Placement Exam (LYS) organized by Student Selection and Placement Center (OSYM). For the classification, two ensemble methods, Bagging and Boosting were considered.

Therefore, data compromised student exam scores and related variables were retrieved from the archives of Turkish Statistical Institue (TUIK) and Student and Placement Center.

The evalution has been done by considering the regulations specified in the Handbook of Student Selection and Placement Center (OSYM). MATLAB and R package programs were used to obtained clustering, corrograms, biclustering and bagging and boosting results. Hierarchical clustering results depicted that the provinces within Eastern Anatolia clustered in the same cluster group. Likewise, the provinces of Marmara, Western Eagen and Southern Mediterranean regions clustered into the same cluster in term of similarities. The provinces of Center Aanatolia and Center and Eastern Black Sea, as well as Gaziantep and Elazığ, clustered into the same cluster in terms of similarities distance was used for clustering. These classifications were supported by bagging and boosting. Frther, the conclusion has been done that bagging and boosting have a good predictive ability to classify results by decrease the sum of the squared error term.

Keywords: AdaBoost, Bagging, Boosting, Bootstrapping, Ensemble Models, Clustring.

ÖN SÖZ

Son yıllarda, ensemble sistem olarak da adlandırılan çoklu sınıflandırma sistemi, hesaplamalı zekâ ve makina öğrenme çevrelerinde oldukça büyük bir ilgiyle izlenmektedir. Ensemble sistemler birçok alandaki problemlerin çözümünde ve gerçek uygulamalarda çok etkili ve çok yönlü olduklarını kanıtladıklarından bu ilgiyi fazlasıyla hak etmektedir. İlk olarak otomatik karar verici sistemdeki sapmayı azaltmak, dolayısıyla da doğruluğu artırmak için geliştirilen Ensemble Sistem; özellik seçimi, güven aralığı, eksik özellik, aşamalı öğrenme, hata düzeltme, kategorik-orantısız veri, hareketsiz dağılımlardan kavram yanılgısını öğrenme gibi birçok değişik makina öğrenme sorununda başarılı bir şekilde kullanılmaktadır. Ensemble sistemlerde amaç doğru kararı almak için değişik görüşleri değerlendirerek, son karara varmak için bazı düşünce süreçleri aracılığıyla fikirleri bir araya getirmektir. Ensemble sistemler özel uygulamalarının birçok diğer makina öğrenimi mevcuttur.

Bu çalışmada Ensemble sistemin iki yöntemi olan “Bagging” ve “Boosting” yöntemleriyle YGS'de başarı göstererek, LYS'ye girmeye hak kazanan öğrencilerin sınıflandırılması yapılmıştır. Tez çalışmamın gerçekleşmesinde katkıda bulunan, karşılaştığım güçlükleri aşmamda yardımcı olan, önerileriyle bana ışık tutan, yol gösteren ve her konuda desteğini esirgemeyen kıymetli danışmanım saygıdeğer hocam Prof. Dr. Hayrettin OKUT'a, desteğini her zaman hissettiğim, çalışmam süresince her konuda yardımcı olan sayın hocam Prof. Dr. Abdullah YEŞİLOVA'ya, tez izleme komitemde yer alan ve yardımlarını esirgemeyen sayın hocam Doç. Dr. M. Nuri ALMALI’ya, bu çalışmanın oluşma sürecinde büyük katkıları ile bana destek olan Arş. Gör. Serdar ABUT’a ve çalışmam boyunca manevi desteğini yakından hissettiğim sevgili arkadaşım Hacer Filiz TAŞDELEN’e, her zaman yanımda olan babam Güven TUĞ’a, beni her konuda destekleyen eşim Tamer KAROĞLU'na ve doğduğu günden bu yana bu çalışmanın içinde olan sevgili kızım Derin KAROĞLU’na sonsuz teşekkürlerimi sunuyorum.

2018
Tuğba TUĞ KAROĞLU

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELER LİSTESİ

Çizelge **Sayfa**

ŞEKİLLER LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanmış bazı simge ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
B	Tekrarlanan Boostrap örneklem sayısı
err	Ağırlıklandırılmış hata
w	Ağırlık
sl	Adımuzunluğu

Kısaltmalar	**Açıklama**
AB	Avrupa Birliği
ADN	Adana
ADY	Adıyaman
AFY	Afyon
AGR	Ağrı
AKS	Aksaray
AMS	Amasya
ANK	Ankara
ANT	Antalya
ART	Artvin
AYD	Aydın
BAL	Balıkesir
BAY	Bayburt
BC	Temel Sınıflandırıcı
BLC	Bilecik
BNG	Bingöl
BTL	Bitlis

BOL	Bolu
BRD	Burdur
BRS	Bursa
CKL	Çanakkale
CKR	Çankırı
CRM	Çorum
C4.5	Bir Karar Ağacı Örneği
DEN	Denizli
DYR	Diyarbakır
DZC	Düzce
EDR	Edirne
EKK	En Küçük Kare
ELZ	Elazığ
EZN	Erzincan
EZM	Erzurum
ESK	Eskişehir
GAZ	Gaziantep
GRS	Giresun
GUM	Gümüşhane
HAK	Hakkari
HAT	Hatay
HKO	Hata Kareler Ortalaması
ISP	Isparta
IST	İstanbul
ISTHOR	İstihdam Oranı
ISZOR	İşsizlik Oranı
IZM	İzmir
İBBS	İstatistiksel Bölge Birimleri Sınıflandırması
KRS	Kars
KAH	Kahramanmaraş
KAS	Kastamonu
KAY	Kayseri

KLS	Kilis
KOC	Kocaeli
KON	Konya
KRL	Kırklareli
KRM	Karaman
KRK	Karabük
KSH	Kırşehir
KTH	Kütahya
LYS	Lisans Yerleştirme Sınavı
MAL	Malatya
MAN	Manisa
MAR	Mardin
MER	Mersin
MUG	Muğla
MUS	Muş
NEV	Nevşehir
NGD	Niğde
NN	Sinir Ağı
OGCSAY	Öğrenci Sayısı
OKLOR	Okullaşma Oranı
OKYAZBLMYN	Okuma Yazma Bilmeyen Sayısı
OKSAY	Okul Sayısı
OGTSAY	Öğretmen Sayısı
ORD	Ordu
OSM	Osmaniye
ÖSYM	Öğrenci Seçme ve Yerleştirme Merkezi
ÖSYS	Öğreni Seçme ve Yerleştirme Sistemi
PAC	Olasılıklı Yaklaşık Doğru
RIZ	Rize
SAK	Sakarya
SAM	Samsun

SAMME	Çok Sınıflı Kayıp Fonksiyon Kullanan Kademeli Toplamsal Modelleme
SRT	Siirt
SNP	Sinop
SVS	Sivas
SMV	Basit Çoğunluk Oylaması
TKR	Tekirdağ
TKT	Tokat
TRB	Trabzon
TNC	Tunceli
TÜİK	Türkiye İstatistik Kurumu
UNIMEZ	Üniversite Mmezunu Sayısı
URF	Şanlıurfa
USK	Uşak
VAN	Van
YGS	Yükseköğrenime Giriş Sınavı
YGSGIR	YGS'ye Giren Öğrenci Sayısı
YOZ	Yozgat
ZON	Zonguldak
WL	Zayıf Öğrenici
WMV	Ağırlıklandırılmış Çoğunluk Oylaması
180USTYERSAY	180 ve üstü puan alıp yerleşen sayısı
180UST OGCSAY	180 ve üstü puan alan öğrenci sayısı
180USTOGCOR	180 ve üstü puan alan öğrenci oranı

EKLER DİZİNİ

Ek **Sayfa**

1. GİRİŞ

Son yıllarda makine öğrenimi ve kompütasyonel zekâ araştırmacıları ensemble temelli sistemlerin karar vermede ne kadar yararlı olduğunu keşfetmede oldukça geç kalmışlardır. Yoğun araştırmaların sonucunda ensemble sistemler hakkında birçok çalışma yapılmıştır. Ensemble temelli sistemler, toplumlar var olduğu sürece çevremizde günlük yaşantımızın bir parçası olacaktır. Aslında ensemble temelli karar verme, bize yabancı olan bir kavram değildir. Bu sistemleri günlük hayatımızda sıklıkla kullanmaktayız. Örneğin; bir karar alma aşamasında insanların oylama yaptığı demokrasinin temeli, yetkili birini seçme ya da yeni bir kanun yapma aslında ensemble temelli karar verme üzerine dayalıdır. İster bir jüriye, ister mahkeme heyetine dayalı olsun, birçok ülkedeki yargı sistemi ensemble temelli karar vermeye dayalıdır (Zhang ve Ma, 2012).

Karar ağaçları, çok katmanlı algılayıcılar, en yakın komşuluk algoritmaları ve destek vektör makineleri en çok bilinen ve kullanılan sınıflandırma yöntemlerdir. Bu tür algoritmalar temel öğreniciler olarak bilinir ve belirli bir veri seti ile eğitilerek sınıflandırma işlemlerinde kullanılır. Bilimsel çalışmalarda temel öğrenicilerden oluşturulmuş farklı sınıflandırıcı topluluklar üzerinde çalışmalar yapılmıştır. Geniş bir kullanım alanına sahip olan ve Ensemble Model olarak adlandırılan bu sınıflandırıcı toplulukları, sınıflandırma aşamasında temel eğiticiye göre daha yüksek bir başarı performansı gösterdikleri için tercih edilmektedir. Ensemble modellerin temelinde sınıflandırıcılarının kararlarının birleştirilmesi ve komite kararının oluşturulması vardır (Bulut, 2016).

Ensemble sözlük anlamı olarak “birlik, grup, takım” anlamına gelmektedir ve birkaç farklı yöntem ile yapılan tahminlerin kombinasyonunu veren fonksiyon şeklinde ifade edilmektedir. Bu yöntemde karışık olan bir hesaplama, öncelikle basit hesaplamalara bölünür daha sonra elde edilen çözümlerin birleştirilmesi ile problem sonuca ulaşır. Bir komisyona üye seçimindeki mantık ile ensemble yöntemi arasındaki mantık aynıdır. Komisyonun her bir üyesi hem kendi alanında yetenekli olmalı hem de üyeler birbirini tamamlayıcı konumda olmalıdır. Çünkü üyeler birbirini tamamlayıcı konumda olmazsa komisyon oluşturmanın bir mantığı olmaz ve tek bir üye yeterli olur.

Fakat üyeler birbirini tamamlayıcı olduğunda, bir veya birkaç üye hata yaptığında diğer üyelerin bu hatayı düzeltme olasılığı daha fazla olur (Oza, 2006). Birçok görüş çoklu sınıflandırıcıları kullanarak öznitelik uzayın parçalanma fikrine dayalı oluşan ensemble sistemleri ilk örneklerden biri olarak kabul etmektedir. Yaklaşık on yıl sonra, sınıflandırma performansını artırmak için benzer şekilde yapılandırılmış sinir ağlarının bir grubunun kullanılabileceği gösterilmiştir.

Ensemble sınıflandırmada temel amaç, farklı sınıflandırıcılar tarafından elde edilen değerlerin bir araya getirilerek bir sonuca ulaşılmasıdır. Bu hesaplama işlemi, diğer sınıflandırıcılara belli ağırlık puanları verilerek yapılır. Böylece farklı sınıflama algoritmaları birleştirilerek hangi oranların kullanılacağı belirlenir. Ensemble sınıflandırma içerisinde en çok kullanılan yöntemler; bagging, boosting, rotasyon ormanı ve rastgele orman algoritmalarıdır (Augusty ve ark., 2013).

Bagging algoritması, var olan bir eğitim setinden yeni eğitim setleri oluşturarak temel öğreniciyi yeniden eğitmeyi amaçlayan bir yöntem olup, eğitim setinin rastgele seçimle üretilmesini amaçlar (Breiman, 1996). Söz konusu eğitim setlerini bootstrap yöntemi ile oluşturur. Bundan dolayı, bazı eğitim örnekleri yeni eğitim kümesinde yer almaz, fakat bazıları ise birden fazla yer alabilir. Topluluktaki her bir temel öğrenici birbirinden farklı örnekler içeren eğitim kümeleriyle eğitilerek, sonuçları çoğunluk oylaması ile birleştirir (Kılınç ve ark., 2015).

Boosting algoritması ise sınıflandırıcının bulmuş olduğu doğruluk değerini artırmaya çalışan bir yöntemdir. Veriye ait olan bir önceki sınıflandırıcının doğru olarak belirleyemediği veriler kullanılır ve hatalı veriler sonradan kullanılacak eğitim setine ilave edilerek daha doğru tahmin yapmaya çalışılır (Schapire ve Freund, 2012).

Bu çalışmada Ensemble sınıflandırmanın iki yöntemi olan “Bagging” ve “Boosting” yöntemleriyle ülkemizde liseyi bitiren öğrencilerin girdiği bir sınav olan Yükseköğrenime Giriş Sınavı'nda (YGS) başarı göstererek, Lisans Yerleştirme Sınavı'na (LYS) girmeye hak kazanan öğrencilerin “Lisans”, “Açıköğretim” ve “Meslek Yüksek Okulu”na yerleşen öğrencilerin sınıflandırılması il ve bölge düzeyinde, okullaşma oranı, okur yazarlık oranı, öğretmen sayısı, öğrenci sayısı vs. değişkenleri esas alınarak yapılmıştır. Bununla birlilte, 2010-2013 yıllarına ait YGS ve LYS sonuçları DÜZEY1, DÜZEY2, DÜZEY3 gibi gelişmişlik düzeyine göre bölgelere ayrılarak sınıflandırma işlemi yapılmıştır. Bagging ve Boosting yöntemlerinin

sınıflandırma sonucundaki performansları karşılaştırılmış ve ayrıca kümeleme analizi yapılarak bu sınıflandırma sonuçları desteklenmiştir.

2. KAYNAK BİLDİRİŞLERİ

Veri madenciliği, geniş veri tabanından yararlı ve önemli ölçüde bilgilerin elde edilmesini sağlayan bir tekniktir. Genel ifadeyle büyük ölçekli veriler arasından bilgisayar programı yardımıyla bilgiye ulaşabilme ve ulaşılan bilgilerin ayıklanarak, veri yığınları içerisinden geleceğe ilişkin tahminler yapılabilmesine olanak sağlayan model, süreç ve analizleri kapsar (Okut, 2015). Veri madenciliği, veri içerisindeki bağlantılardan ve ilişkilerden faydalanarak geçerli tahminler yapmak için kullanılan bir süreçtir. Burada amaç; geçmişte yapılan çalışmaları temel alarak gelecekteki davranışların tahminine yönelik karar verme modelleri ortaya çıkarmaktır (Koyuncugil ve ark., 2008).

Veri madenciliği; veri tabanı teknolojisi, istatistik, makine öğrenimi, örüntü tanıma, yapay sinir ağları, verilerin görselleştirilmesi ve uzaysal veri analizi gibi farklı disiplinlerde yer alan tekniklerin bir birleşimini içerir (Hand ve ark., 2001). Bu tekniklerden biri olan makine öğrenimi; çevresel durum gözlemleri ve geçiş tabanlı kuralların eşdeğer olduğu öğrenme işlerinin ve öğrenmenin otomasyonu olup, bir problemi ortamdan aldığı bilgiye göre modelleme özelliğine sahiptir (Akgöbek ve Çakır, 2009). Makine öğrenimi teknikleri yönlendirici (supervised) ve yönlendirici olmayan (unspervised) öğrenme metotlarından oluşur. Yönlendirici öğrenme, önceden gözlemlenmiş ve sonuçları bilinen (etiketlenmiş) verileri kullanarak, bu verileri ve sonuçlarını kapsayan bir fonksiyon oluşturmayı amaçlayan makine öğrenimi metodudur. Yönlendirici olmayan öğrenme ise, etiketlenmemiş verideki gizli yapıyı bulma işlemidir. Yani, veriler arasında var olan ama gözle görülmeyen bağıntının açığa çıkarılması işlemidir (Nizam ve Akın, 2014).

Makine öğrenme tekniklerinden biri olan yönlendirici öğrenme en çok sınıflandırma ve regresyon gibi yöntemlerde kullanılmaktadır. Sınıflama ve regresyon modellerinde kullanılan başlıca tekniklerden biri ise Karar Ağaçlarıdır. Yönlendirici öğrenme sınıflandırmasında yaygın olarak kullanılan bir algoritma türü olan Karar Ağaçları iki amaç için kullanılır. Bunlardan biri sınıflandırma amaçlı kullanılan sınıflandırma ağaçları iken diğeri ise tahminleme amaçlı kullanılan regresyon ağaçlarıdır. Karar ağaçları analizi iki aşamadan oluşmaktadır. Birinci aşama ağacı

oluşturma işlemidir. Bu aşamada bütün eğitim verileri ile ağacın kök kısmından başlanarak ağaç oluşturulur. Dönüşümlü olarak değişkenlerin verdiği bilgiler parçalanarak değişken seçimi yapılır. İkinci aşama, ağacı budama aşamasıdır. Bu aşamada ise hata, belirsiz bilgi ve aykırı değer içeren ağacın dalları uzaklaştırılır (Okut, 2015).

Uygulanması ve yorumlaması kolay olan karar ağaçları modelleri, veri setlerine sağladığı uyum, güvenirlik, oluşturulmasındaki kuralların basitliği ve sadeliği nedeniyle veri madenciliğinin sınıflama modelleri arasında yaygın kullanıma sahiptir (Gökay ve Taşkın, 2005; Kavzoğlu ve Çölkesen, 2010). Bu modeller, çok büyük hacimli veri setlerine kolayca uygulanabilir. Böylece hem daha hızlı bir uygulama olanağı elde edilmiş olur hem de elde edilen sonuçların yorumlanması ve ilişkilerin önem sırasının görsel olarak ortaya konulması diğer yöntemlere göre daha kolay ve kullanışlı olur (Myatt, 2007).Karar Ağacı modellerinin en önemli avantajları, parametrik olmayan yöntemler arasında olması nedeniyle diğer çok değişkenli tekniklerde sağlanması gereken istatistik varsayımlarının olmamasıdır. Bu durum da, elde edilen sonuçların yorumunu oldukça basitleştirerek daha somut ve kullanışlı hale getirmektedir. Bu modeller, karar alırken hangi faktörlerin göz önüne alınması gerektiği ve her bir faktörün, kararın farklı çıktıları ile geçmişte nasıl ilişkili olduğunun belirlenmesi konusunda araştırıcıya yardımcı olan modellerdir (Bounsaythip ve Esa, 2001; Albayrak ve Yılmaz, 2009; Schapire ve Freund, 2012).

Karar ağaçları, öğrenme ve sınıflama olmak üzere iki basamaklı bir işlemden oluşur. Öğrenme aşamasında verilerin bir kısmı kullanılır. Buna eğitim verisi denir. Eğitim verileri genel olarak toplam verilerin büyük kısmından oluşur. Verinin geriye kalan kısmına ise test verileri denir. Bir sınıflama algoritması tarafından eğitim verileri kullanılarak öğrenme gerçekleştirilir. Sınıflama basamağında ise test verisi, sınıflama kurallarının veya karar ağacının doğruluğunu belirlemek amacıyla kullanılır. Eğer doğruluk onaylanır durumdaysa, kullanılan kurallar, yeni verilerin sınıflanması amacıyla kullanılır (Okut, 2015).

Sınıflandırıcıların birleştirilmesi, yeniden örneklenen eğitim setleri ile sınıflandırıcıların ayrı ayrı eğitilmesi ile ortaya çıkan tahminler doğrultusunda sınıflandırma işleminin gerçekleştirilmesi işlemlerinden oluşur. Genel olarak birleştirme sonucu elde edilen sınıflandırıcı ile yapılan sınıflandırma doğruluğunun her bir

sınıflandırıcının tekil olarak kullanılmasından daha iyi sonuç verdiği ifade edilir. Çünkü tek bir sınıflandırıcı daha yüksek test hatasına sahip olabilmekte iken, sınıflandırıcıların çeşitliliği, genellikle tek bir sınıflandırıcının hatasını telafi eder. Bundan dolayı, sınıflandırıcıların kombinasyonu ile daha az test hatası elde edilmektedir (Opitz ve Mach, 1999; Pal ve Mather, 2003).

Ensemble sınıflandırmada temel amaç, önceden farklı sınıflandırıcılar tarafından elde edilen değerlerin bir araya getirilmesi ile bir sonuç üretilmesidir. Bu işlem yapılırken diğer sınıflandırıcılara belli ağırlık puanları verilerek hesaplama yapmaya çalışılır. Burada asıl problem farklı sınıflama algoritmalarını birleştirmek ve hangi oranların kullanılacağına karar vermektir. En büyük avantajı diğer yöntemlerin verilerini bir arada kullandığı için daha iyi değerler elde edebilmesidir (Augusty ve Izudheen, 2013). Ensemble sınıflandırıcıları, son yıllarda önemli bir kullanım alanına sahip olmuştur. Özellikle tekil sınıflandırıcı yapılarında oluşan hataları minimum hale getirmek ve daha hızlı bir sınıflandırma algoritması sunması gibi özellikleri, ensemble sınıflandırıcıları bu konuma taşımıştır (Lee ve ark., 2010; Tartar ve ark., 2013).

Kısmen bu ufuk açıcı çalışmalar sonucu olarak ve kısmen de bağımsız çalışmalar sayesinde ensemble sistemleri çalışmalarındaki artıştan dolayı ensemble temelli algoritmaların birçok değişik türü olan bagging, rastgele orman (karar ağaçlar topluluğu), rotasyon ormanı, birleşik sınıflandırıcı sistem, uzmanların karışımı (MoE), toplu genelleme, ortak kümelenme, çoklu sınıflandırıcılar kombinasyonu, dinamik sınıflandırıcı seçimi, sınıflandırıcı birleşimi, sinir ağlar birleşimi, sınıflandırıcı topluluğu gibi algoritmalar olarak ortaya çıkmışlardır (Dasarathy ve Sheela, 1979; Jacobs ve ark., 1991; Wolpert, 1992; Xu ve ark, 1992; Jordan ve Jacobs, 1994; Drucker ve ark, 1994; Blooch, 1996; Breiman, 1996; Woods ve ark, 1997; Kuncheva, 2001; Kuncheva, 2004). Bu algoritmalar ve genelde tüm ensemble sistemler her bir sınıflandırıcının eğitim verisinin seçimine, topluluk üyelerinin oluşturulmasında kullanılan özel süreçlere, ve toplu karar vermek için bir araya getirme kuralına bağlı olarak değişiklik gösterir (Zhand ve Ma, 2012).

Birçok durumda ensemble elemanları; sınıflandırma seçimi ve sınıflandırıcı birleşimi olmak üzere iki genel kurulumdan biri olarak kullanılır (Woods ve ark., 1997; Kuncheva, 2002; Kuncheva, 2004).

Sınıflandırıcı seçiminde, her bir sınıflandırıcı eğitilir, yeni örneklem ile bu alana yakın verilerle eğitilen sınıflandırıcı son kararı vermek için seçilir (Jacobs ve ark, 1991; Alpaydin ve Jordan, 1996; Woods ve ark 1997; Giacinto ve Roli, 2001).

Sınıflandırıcı birleşiminde, yani füzyonda tüm sınıflandırıcılar tüm özniteliksel uzay üzerinden eğitilirler ve daha sonra düşük sapma ve bundan dolayı oluşan düşük hata ile birleşik sınıflandırıcı elde etmek için biraraya getirilirler. Bagging, rastgele orman, arc-x4, Boosting ve AdaBoost bu yaklaşımın örnekleridir (Schapire, 1990; Breiman, 1996; Schapire ve Freund, 1997; Breiman, 1998; Breiman, 2001). Her bir sınıflandırıcının birleştirilmesi sadece etiketlere ya da sınıfa özgü sürekli değer verilerine bağlı olabilir. Bu nedenle sınıflandırıcı sonuçları ilk olarak [0, 1] aralığında normalleşmesi, sınıflandırıcıların her bir sınıfa verdiği destek olarak yorumlanabilir. Bu tür yorumlamalar cebir birleşim kurallarına (basit ya da ağırlıklı çoklu seçim, maksimum, minumum, toplam, üretim ya da diğer sınıfa özgü kombinasyonlara), Dempster–Shafer temelli sınıflandırıcı birleşim ya da karar şablonuna yol açar (Ho ve ark., 1994; Rogova, 1994; Lu, 1996; Kittler ve ark., 1998; Akoot ve Kittler, 1999; Tax ve ark., 2000; Kuncheva, 2001; Muhlbaier ve ark, 2001; Kuncheva, 2002).

Ensemble modelleri arasında ilk uygulamaya alınan bagging yöntemidir. Bu yöntem; tekrarlı olarak iadeli şekilde öğrenim verilerinden örnek çekerek tahmin konusunda uygun ağacı önerir. Bagging yönteminde “n” adet örnekten oluşan eğitim setinden yine “n” örnekli bir eğitim seti; iadeli, rastgele seçimle üretilir. Bu durumda bazı eğitim örnekleri yeni eğitim kümesinde yer almazken (yaklaşık % 33) bazıları birden fazla kez yer alırlar. Topluluktaki her bir temel öğrenici bu şekilde üretilmiş birbirinden farklı örnekler içeren eğitim kümeleriyle eğitilirler ve sonuçları çoğunluk oylaması ile birleştirilir. Bundan dolayı bagging yöntemi, eğitim setinin rastgele olarak yeniden oluşturarak ayrı ayrı her bir sınıflandırıcının eğitimini gerçekleştirmesini sağlar (Okut, 2015).

Bootstrap ve Aggregation'ın kısaltılmış hali olan Bagging, orijinal veri setlerinden yeniden örneklendirilerek ortaya çıkarılan ve bootstrap tarafından farklı eğitim veri setleri tarafından eğitilen sınıflandırıcıları, en son elde edilen sonuçlar olarak bir araya getirir ve optimizasyon işlemi kullanılarak bagging algoritmasının sadeleştirilmesiyle elde edilen en uygun bir ensemble modeli elde edilir. Optimizasyon işlemi ile temel sınıflandırıcıların doğruluğu ve çeşitliliğine göre en uygun (optimum)

sınıflandırıcıların nasıl seçileceği vurgulanır (Zeng ve ark. 2010). Fakat Bagging yönteminin sınıflandırma algoritmalarının performansını iyileştirilmesi için elde edilen sınıflandırıcıların kayıp değerlerekarşı ne kadar hassas olan, verilerin veri setini içerdiği kayıp değere göre değişir. Fakat oylama işlemindeki sınıflandırıcıları eğitmek için kullanılan Bootstrap yönteminde, veri setlerinin yetersizliğinin de göz önünde bulundurulması gerekir (Hsu 2013). En yaygın kullanılan ensemble yöntemlerinden olan bagging algoritması, bootsrap örnekleri tarafından ensemble sınıflandırıcılarını oluşturur ve farklı bootstrap örneklerini öğrenme seti olarak kullanarak sınıflandırmayı daha iyi hale getirr. Son zamanlarda yapılan çalışmalar Bagging yönteminin öğrenim seti içerisindeki aykırı değerlerin etkisini azalttığını göstermiştir (Biggio ve ark., 2011). Zira bootstrap örneklemesi ile söz konusu aykırı değerlerin öğrenim veri setine dahil olma olasılığı çok düşük olmaktadır.

Algoritma sınıflandırıcıların sonuçlarını iyileştirebilecek bir metot olan bagging ile ilgili birçok inceleme yapılmıştır. Bagging metodu; karar ağaçlarını üretmede algoritma sınıflandırmasını kullanmayı amaçlar. Bu metot tarafından sergilenmiş olan sınıflandırmaların daha iyiye götürülmesine yardımcı olacak minimum sayıda karar ağacı bulunmuştur (Machova ve ark. 2006). Esposito ve Saitta (2003) ise çalışmalarında ensemble öğrenmede Monte Carlo algoritmalarının faydalı olduğunu ileri sürmüştür. Çalışmalarında bu algoritmanın, bagging yönteminin ne zaman faydalı olacağının tahmin edilmesine yardımcı olduğunu ve marjini yükseltmenin, performansı iyileştirmeye nasıl yardımcı olduğunu açıklamış ve hata tahmini için, ensemble öğrenmenin yeni bir yolunu sunmuşlardır. Babu ve ark. (2013), oluşturdukları "soya fasülyesi uzman bilgi tabanı" programı çerçevesinde C4.5 sınıflandırıcının performansını artırmak üzere bagging algoritmasından yararlanarak hastalık çeşidinin belirlenmesine yönelik bir uygulama geliştirmişlerdir.

Bagging yöntemi ilk bulunduğunda, işe yaramayacağı konusunda sadece sezgisel tartışmalar ortaya atılmıştı. Bylander (2002), Bagging sınıflandırma ağacının veya regresyon tahmini performansını kanıtlamayı amaçladığında avantaja dönüşen yumuşatıcı bir işlem haline geldiğini göstermiştir. Karar ağacı olayında, bu teori Breiman'ın bagging hata olasılığını da azaltan, hata kareler ortalaması (HKO) olan varyans küçültme tekniğidir. Aynı şey bagging'in hesaplama açısından daha basitversiyonu olan subagging için de geçerlidir. Bu durum aynı zamanda tahmin

edicinin U-istatistik olduğu basit olay Buja ve Stueztle (2006)'in çalışmasında da gösterilmiştir.

Schapire (2002), çalışmasında boosting algoritmasında her bir temel öğrenicinin önceki temel öğrenicilerin doğru sınıflandıramadığı örnekler ile eğitilmekte ve temel öğrenicilerin kararlarının kendi eğitim kümeleri üzerindeki başarıları ile ağırlıklandırılarak birleştirilmekte olduğu fikrini ortaya koymuştur. Boosting yöntemi bir dizi sınıflandırıcı serisi oluşturma işlemidir (Freund ve Schapire, 1996; Nanni ve Lumini, 2006). Serinin üyesi olan her bir sınıflandırıcı için kullanılan eğitim seti, serideki bir önceki sınıflandırıcı veya sınıflandırıcıların performansına göre belirlenir (Quinlan, 1996; Maclin ve Opitz, 1997, Dietterrich, 2000).

Boosting, bir sınıflandırıcının doğruluğunu geliştirerek en iyi kullanım sağlayan bir yöntemdir. Bu sınıflandırma yöntemi, eğitim setinde tam olarak doğru sınıflandırıcıyı inşaa etmek için alt program olarak kullanılır. Boosting yöntemi; eğitim verisinde tekrar eden sınıflandırma sistemini uygular fakat her bir basamakta öğrenme dikkati, uyarlanabilen ağırlıklar kullanarak, bu setin farklı örnekleri üstünde yoğunlaşır. İlerleme bittiğinde elde edilen tekli sınıflandırıcılar, eğitim setindeki yüksek doğruluk sınıflandırıcısı olan finalle birleştirilir. Çeşitli yazarların hem teorik hem de deneysel olarak gösterdiği gibi son sınıflandırıcı böylece genel olarak test setindeki yüksek doğruluk derecesini başarır (Dietterich 2000; Banfield ve ark., 2007).

Boosting algoritmalarının pek çok versiyonu olmasına rağmen en iyi bilineni AdaBoost algoritmasıdır. Fakat sadece iki elemanlı sınıflandırma problemleri için kullanılabilir. Çoklu sınıflandırma problemleri için boosting algoritmaları arasında, AdaBoost'un en kolay ve en doğal iki çeşidi; SAMME(çok sınıflı kayıp fonksiyon kullanan kademeli toplamsal modelleme) ve AdaBoost.M1'dir (Mukherjee ve Schapire, 2011).

AdaBoost yöntemi; boosting yöntemi ile bagging yönteminin birleşimidir (Webb, 2000). İlk olarak Freud ve Schapire tarafından 1996 tarihli çalışmalarında tanıtılan AdaBoost yöntemi, zayıf sınıflandırıcılarla güçlü bir sınıflandırıcı oluşturan iki sınıflı sınıflandırma problemleri için birçok zayıf sınıflandırıcıdan oluşan bir ensemble yöntemidir. Bagging ve AdaBoost gibi oylama amaçlı sınıflandırma algoritmaları, gerçek veya yapay veri kümeleri için kesin sınıflandırıcıların doğruluğunu iyileştirmede çok etkili olmuşlardır. Bu algoritmalar yeniden gözden geçirilip Naive-Bayes uyarıcısı

ve karar ağacı ile konjeksiyonel bir kaç çeşidi karşılaştırarak, büyük bir deneysel çalışma tanımlanmıştır. Karışık yeniden ağırlıklandırma ve kombinasyon tekniklerini kullanan ve ayrıca sınıflandırma hatalarını da etkileyen bu algoritmaları ortaya çıkarmak üzere, farklı yöntemlerin nasıl etkilediğini gösterilir. Bagging yöntemi sabit olmayan metotların değişkenliğini azaltırken, Boosting yöntemi sabit olmayan metotların hem sapmasını hem de varyansını düşürür, fakat sabit olan Naive-Bayes için olan varyansı ise yükseltir. Önemli bir farkın göstergesi olan yeniden örnekleme yerine, yeniden ağırlıklandırma yöntemi kullanılırsa Arc-X4, AdaBoost algoritmasından farklı davranış sergiler. Budama yapılmayan durumda olasılık tahminleri kullanıldığında, bagging yönteminde iyileşme görülür. Oylama metotlarının hata kareler ortalaması ile oylama olmadan uygulanan metotlar karşılaştırıldığında, oylama metotlarının hata kareler ortalamasında büyük ve önemli azalmaya giden bir yol olduğu görülmüştür (Bauer ve Kohavi, 1998).

Bagging ve boosting sınıflandırma için yaygın olarak kullanılan iki ensemble yöntemidir. Ortak amaçları, rastgele tahminden bir nebze daha iyi olan tekli sınıflandırıcıları birleştiren bir sınıflandırıcının doğruluğunu iyileştirmektir. Boosting algoritmaları ailesi arasında, sadece ikiye bölme işi için kullanılmasına rağmen en iyi bilineni AdaBoost algoritmasıdır (Alafaro ve ark., 2013). Sınıflandırma modellerini geliştirmek için kullanılan buluşsal yaklaşımlar olan bagging ve boosting yöntemleri, temel öğrenme algoritmalarına verilen eğitim verisini etkisi altına alarak, çeşitli ensemble sınıflandırıcıları üretir. Bunlar; yapay ve gerçek dünya veri setlerindeki bazı algoritmaları iyileştirmede çok başarılıdırlar. Bagging ve boosting yöntemlerinin eğitim verisi aracılığıyla, sadece bir geçişe ihtiyaç duyan çevrimiçi versiyonları vardır. Boosting tek bir güçlü sınıflandırıcı meydana getirmek için güçsüz sınıflandırıcıların topluluklarını bir araya getirir. Boosting yönteminde başarılı modeller eski tahminlere fazladan ağırlık verir. Oysa bagging yönteminde her model veri setinin bootstrap örneğini kullanarak bağımsız şekilde yapılandırılır. Sonunda genel tahmin oy çokluğuyla yapılır (Kumari, 2012).

Ensemble işleminde kullanılan her bir sınıflandırıcı için bagging ve boosting yöntemleri farklı eğitim setlerinin oluşturulmasında, yeniden örnekleme tekniklerini kullanmaktadır. Bu iki yöntemin kullanımı arasındaki en temel fark boosting yönteminin tekrarlı bir şekilde birden çok sınıflandırıcı oluşturmasıdır (Webb, 2000).

Boosting yönteminin ikili sınıflandırma problemlerinde düşük hata ürettiği ve sınıflandırma performansında üstün başarı gösterdiği ortaya koyulmuştur (Schapire, 1990). Karar ağacı algoritmasının performansını iyileştirmek için, bagging ve boosting karşılaştırıldığında gürültünün hiç olmadığı veya çok az olduğu durumlarda rastgele sınıflandırma, bagging ile yarışır durumdadır. Hatta bazen rastgele sınıflandırma yönteminin daha iyi sonuç vermesi de söz konusu olmuştur. Bununla birlikte; boosting yöntemi kadar kesin sonuç vermediği belirlenmiştir. Gürültünün çok olduğu durumlarda bagging, boosting yönteminden çok daha iyiyken, rastgele sınıflandırmanın bazı durumlarda iyi olduğu tespit edilmiştir (Dietterrich 1999).

Bir diğer Ensemble modeli olan rastgele altuzaylaryönteminde temel öğrenicilere eğitim kümesindeki tüm özelliklerden ziyade bu özelliklerin bir rastgele alt kümesi verilmektedir. Bu durum, her bir temel öğrenicinin veriye farklı uzaylardan yaklaşarak ve farklı sonuçlar elde edilmesini sağlar. Temel öğrenicilerin kararları çoğunluk oylaması ile birleştirilmektedir (Ho, 1998).

Ensemble modeller için kullanılan başka bir yöntem olan rastgele oman yöntemidir. Rasgele orman yönteminin temel öğrenicileri karar ağaçlarıdır. Temel öğreniciler yine bagging ile üretilmiş eğitim örnekleriyle eğitilirler, fakat farklı olarak temel öğrenicilerin (karar ağaçlarının) her bir düğümünde veriyi bölerken, tüm özelliklerin incelenmesi yerine özelliklerin rastgele bir alt kümesi incelenir. Böylece karar ağacının üretim süresi kısalır ve karar ağaçlarının farklılığı yeni bir rastgelelik ile artırılır. Temel öğrenicilerin sonuçları yine çoğunluk oylaması ile birleştirilmektedir (Breiman, 2001).

Bagging, boosting ve rastgele orman yöntemleri, tekil sınıflandırıcıların performansını iyileştirmek için kullanılan ensemble yöntemleridir. Bu yöntemler, sınıflandırma doğruluğunu arttırarak, diğer yöntemlere göre büyük bir üstünlük sergilemektedir. Bu nedenle bagging ve boosting yöntemleri hacim ve içerik olarak büyüyen veri setlerinde kullanımı giderek yaygınlık kazanmaktadır (Bifet ve ark., 2010).

Makine öğrenimi alanında oldukça etkili olan ensemble algoritmaları elde ettikleri başarılı sonuçlarla bunu kanıtlamaktadır. Bagging, rastgele altuzaylar ve rastgele orman algoritmaları üzerinde budama ve ağırlıklandırma işlemleri incelendiğinde, bu algoritmaların başarıları arasında en fazla farklılık bulunan yöntemin

rastgele altuzaylar olduğu saptanmıştır. Tüm algoritmalar içinde en başarılı algoritmanın ise kararları ağırlıklandırılmış rastgele altuzaylar olduğu belirlenmiştir. Üç ensemble algoritmasının (temel öğrenicilerinin) kullandıkları bilgiler incelendiğinde, eğitim kümesinden en az bilgiyi kullanan üyelerin rastgele altuzaylar algoritmasında olduğu görülmüştür. Bagging algoritması üyeleri, ağacın her düğümünde örneklerin tüm özelliklerine sahiptir. Rastgele altuzaylar algoritmasında, üyeler eğitim kümesinin sadece bir bölümüne sahip iken rastgele orman algoritması üyeleri ise her düğümde, özelliklerin rastgele bir alt kümesine sahip olmuştur. Bunun sonucu olarak rastgele altuzay algoritması üyelerine ait sonuçların birbirinden farklı olması, diğer ensemble algoritmalarına göre daha fazladır. Rastgele altuzay algoritmasının diğer ensemble algoritmalarına göre üyelerinin tümüne daha az ihtiyaç duymaktadır. Çünkü rastgele altuzaylara ait grup özelliklerinin bazıları sınıflandırmaya yardımcı olmayabilir. Üyelerinin tümüne en çok ihtiyaç duyan ensemble algoritması Bagging algoritmasıdır. Bunun nedeni ise, üyelerin en çok bilgisini kullanan ve dolayısıyla temel öğrenicilerinin kullandıkları bilgilerin kesişimi büyük olan algoritmanın Bagging algoritması olmasıdır. Rastgele altuzaylar; temel öğrenicilerin karar ağırlıklandırılmalarının daha iyi sonuç verdiği tek ensemble algoritmasıdır. Tüm üyelerin başarılarının birbirine çok yakın olduğu bir durumda, kararların ağırlıklandırılması etkili bir işlem olmayacaktır. Bu durumda ensemble algoritmalarının temel öğrenicileri arasında ürettikleri farklılıklara göre büyükten küçüğe sıralaması; rastgele orman, rastgele altuzaylar, bagging şeklindedir (Amasyalı ve Ersoy, 2011).

Yönlendirici öğrenme, makine öğreniminin bir alt dalıdır. Bu etiketlenmemiş veri tabanlı yönlendirici olmayan öğrenme ile yönlendirici ve yönlendirici olmayan öğrenmenin kombinasyonu olan yarı yönlendirici öğrenme ile de çelişmez (Bishop 2006).

İkili sınıflandırmada, zayıf öğrenici rastgele tahminden daha iyi bir sınıflandırma oranı verir. Diğer yandan, daha güçlü öğrenici mükemmele yakın sınıflandırmayla değerlendirilmelidir. Bu teorik problem yüksek pratiklik ile ilgilidir. Çünkü zayıf bir öğrenici elde etmek kolay fakat güçlü öğrenici elde etmek zordur (Zhou 2012).

Makine öğreniminin lideri olarak kabul edilen Breimanın göstermiş olduğu gibi AdaBoost yöntemi Boosting standart sınıflandırıcıları arasında en iyisi olarak kabul edilmektedir (Hastie ve ark., 2009).

3. MATERYAL VE YÖNTEM

3.1. Materyal

Bu çalışmada kullanılan veri seti TÜİK (Türkiye İstatistik Kurumu) ve ÖSYM (Öğrenci Seçme ve Yerleştirme Merkezi) arşivinden alınmıştır. Veri setini oluşturan değişkenler; 2010-2013 yılları arasında 81 ilde orta öğretim kurumlarında okuyan öğrenci sayısı, orta öğretim kurumlarında çalışan öğretmen sayısı, orta öğretim kurumları sayısı, okullaşma oranı, okuma yazma bilen kişi sayısı, üniversite mezunu olan kişi sayısı, işsizlik oranı, istihdam oranı, ÖSYM'nin düzenlediği YGS'ye giren; 180 ve üzeri puan alarak ikinci basamak sınavı olan LYS'ye girmeye hak kazanan öğrencilerden Lisans, Açık Öğretim Fakültesi ve Meslek Yüksek Okuluna yerleşen öğrenci sayıları olarak kullanılmıştır. Bu amaçla TÜİK ve ÖSYM arşivinden değişkenlerle ilgili veriler toplanmıştır.

ÖSYM sınav kılavuzu incelenerek değerlendirmelerin yapılmasında bu kılavuzdaki esaslara göre incelemeler yapılmıştır. Söz konusu kılavuza göre; YGS'de en az bir puan türünde 140 ve üzeri puan alamayan adayların, YGS puanları ile bir yükseköğretim programını tercih etme ve LYS'ye girme hakkı bulunmamaktadır. YGS'de 140.00 ile 179.99 arası puan alan adaylar sadece meslek yüksekokulu ön lisans programları ile açık öğretim programlarını tercih edebilir. Bu adayların LYS'ye girme hakkı bulunmamaktadır. 180.00 ve daha fazla puan alan adaylar, LYS'ye girme hakkı kazanır ve hem meslek yüksekokulu ön lisans programları ile açık öğretim programlarını hem de YGS puanı ile öğrenci alan lisans programlarını tercih edebilir (Anonim, 2013).

TÜİK ve ÖSYM arşivinden alınan verilerle oluşturulan değişkenler şu şekilde belirlenmiştir:

* Orta öğretim kurumlarında okuyan öğrenci sayısı
* Orta öğretim kurumlarında çalışan öğretmen sayısı
* Orta öğretim kurumları sayısı
* Okullaşma oranı
* Okuma yazma bilen sayısı

* Üniversite mezunu sayısı
* İşsizlik oranı
* İstihdam oranı
* YGS'ye giren öğrenci sayısı
* YGS de 180 ve üzeri puan alan öğrenci sayısı
* LYS sonucu yerleşen öğrenci sayısı
* LYS sonucu Lisansa yerleşen öğrenci sayısı
* LYS sonucu Açık Öğretim Fakültesine yerleşen öğrenci sayısı
* LYS sonucu Ön Lisansa yerleşen öğrenci sayısı

Veri setinin MATLAB istatistik yazılım programında Bagging ve Boosting yöntemleriyle sınıflandırma işlemleri gerçekleştirilmiştir. Ayrıca R bilgisayar programında da Kümeleme Analizi yapılarak desteklenmiştir. Veri setine göre başarı oranları yeniden hesaplanmış ve yeni hesaplanan değer, bağımlı değişken olarak göz önünde tutulmuştur. Buna göre ortalamanın üstü 1'e ve ortalamanın altı 0'a eşit olacak şekilde bağımlı değişken değerleri atanmıştır.

Kullanılan verilere uygulanan Bagging ve Boosting yöntemlerinden elde edilen sonuçları vermeden önce, korrogram, bootstrap kümeleme ve iki yönlü kümeleme (biclustering) tekniklerinden elde edilen sonuçlar değerlendirilmiştir.

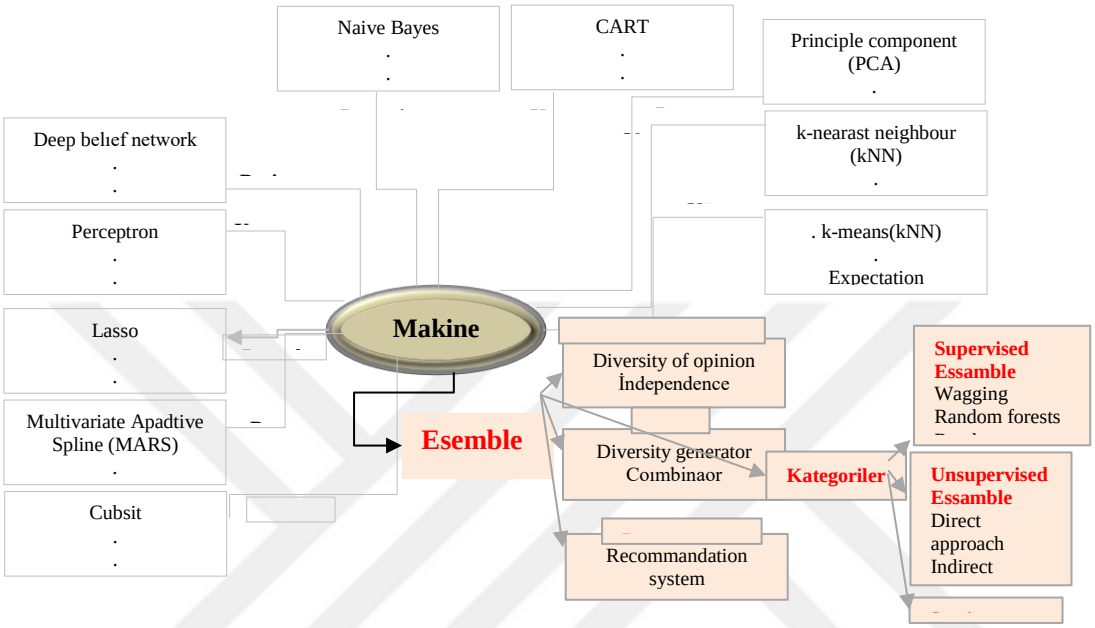
3.2. Yöntem

Tezin yöntem kısmında ana başlıklar olarak; ensemble yöntemine yönelik genel tanıtım ve ensemble yöntemlerinden olan Bagging ve Boosting yöntemlerinin tanıtımı yapılmıştır.

3.2.1. Ensemble sistemleri ve istatistiksel özellikleri

Makine öğreniminin önemli bir elemanı olan ensemble yöntemi yüksek doğrulukla sonuç elde etmede hem yönlendirici hem de yönlendirici olmayan çok etkili öğrenme algoritmalarına sahiptir (Şekil 3.1). Ensemble yöntemleri, tek bir yöntem kullanarak model uyumunu yapmaz. Aksine birçok yöntemin doğrusal kombinasyonunu kullanarak modelin uyumunu yapmaktadır. Başka bir ifadeyle ensemble yöntemleri,

birden çok model oluşturarak ve bunları birleştirerek parametre tahminini yapar ve sonuçların iyileştirilmesini sağlar. Bunun sonucu olarak ensembleyöntemleri istatistiksel modellerin tahmin ve öngörü performansını iyileştirmede çok etkili bir yöntem olarak öne çıkar (Zhou 2012).



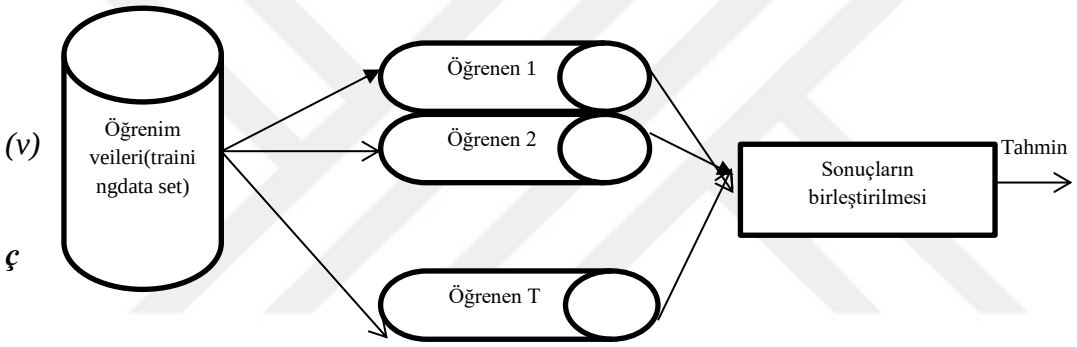
Şekil 3.1. Makine öğrenimi ve Ensemble modeli (Gollapudi, 2016, değiştirilerek alınmıştır).

Ensemble yöntemlerinin uyumunu sağladığı her bir modele; öğreniciler veya öğrenenler denilmektedir. O halde ensemble yöntemi aynı problem üzerinde birçok öğrenicinin elde ettiği bilgiyi bir araya getirerek modelin tahmin etme performansını iyileştirir (Şekil 3.2). Her bir öğrenene, temel öğrenicidenilmektedir. Temel öğrenicilerin bilgisi genel olarak öğrenim veri setinden elde edilir. Bu bilgi bir algoritma yardımı ile sağlanır ve bu algoritmaya temel öğrenim algoritması denilmektedir. Bu algoritma bir karar ağacı, bir yapay sinir ağı veya diğer bir algoritma çeşidi olabilir (Zhou, 2012). Ensemble yöntemlerinde her bir öğrenen hem aynı hem de farklı öğrenim algoritması kullanabilir. Yaygın olarak her bir öğrenen, aynı öğrenim algoritması kullanır ki buna homojen ensemble yöntemler denilmektedir.

Herhangi bir sınıflandırma hatasını kontrol edebildiğimiz iki bileşen olan; *bias*, sınıflandırıcının doğruluğunu; *varyans* ise sınıflandırıcının değişik eğitimlere tabi tutulmasını ifade eder. Bu iki bileşen arasında ters bir orantı vardır. Düşük hatalı

sınıflandırıcılar yüksek sapma eğilimi gösterir ve bunun terside geçerlidir. Diğer taraftan, ortalama almanın, sapmayı azaltan bir etkisi de söz konusudur. Bundan dolayı ensemble yöntemlerinin hedefi nispeten belirlenmiş ya da benzer hatalı birkaç sınıflandırıcı oluşturmak, bunların verilerini bir araya getirmek, ortalamasını tespit etmek ve sapmayı azaltmaktır (Zhang ve Ma, 2012).

Ensemble yöntemleri bağlamında, ensemble elemanlarını (öğrenenleri) bir araya getirmenin birçok yolu vardır. Sınıflandırıcı verilerinin ortalamasını almak bunlardan biridir. Ayrıca, sınıflandırıcı verilerini bir araya getirmek, gruptaki en iyi sınıflandırıcıdan daha iyi bir sonucu garanti eden bir sınıflandırma performansını gerektirmez. Bundan ziyade düşük performanslı bir sınıflandırıcı seçme olasılığını azaltır.



Şekil 3.2. Tipik bir ensemble mekanizması.

3.2.2. Ensemble elemanların bir araya getirilmesi

Herhangi bir ensemble temelli sistemin son adımı, her bir sınıflandırıcıyı birleştirmek için kullanılan mekanizmadır. Bu adımda kullanılan strateji kısmen grup elemanı olarak kullanılan sınıflandırıcının türüne bağlıdır. Örneğin, destek vektör mekanizması gibi bazı sınıflandırıcılar, sadece aralıklı değer sınıflandırma sonuçlarını sağlar. Çok katmanlı algılayıcılar ya da (na¨ive) Bayes sınıflandırıcıları gibi diğer sınıflandırıcılar ise sınıflandırıcının her bir sınıfa verdiği destek olarak da yorumlanan sürekli değer veren sınıfa özgü sonuçlar sağlar. Oylama temelli yaklaşımlara ek olarak, aritmetik (toplam, sonuç, ortalama) birleştiriciler ya da daha karmaşık karar modelleri gibi sınıflandırıcılar için bir dizi seçenek mevcuttur. Eğitim tamamlandıktan sonra bu

birleştiricilerin birçoğu hemen kullanılabilirken, daha karmaşık algoritma kombinasyonları için ek eğitim aşamaları gerekebilir.

Son yıllarda çok fazla sayıda ensemble temelli sınıflandırıcı geliştirilmiştir. Bunların birçoğu çok iyi şekilde temellendirilmiş olup, tahmin etme performansları iyi test edilmiş algoritmalar ve bu algoritmaların varyasyonlarından oluşmaktadır (Şekil 3.1). Bu çalışmada ensemble modellerinden, bagging ve boosting yöntemleri ele alınacaktır.

3.2.3. Bagging (BoostrapAggregating)

Leo Breiman'ın sınıflama ve regresyon tahminlerinde doğruluğu arttırmak için kullanılan bir yöntem olan bagging algoritması etkili ve aynı zamanda basit, ensemble temelli bir algoritmadır (Coşgun ve ark., 2011).Bagging, bootstrap kümelenmesini ifade etmekte olup, varyansı düşürmek için bootsrap örneklemini kullanan ve bazı tahmin edicilerin doğruluğunu artıran (sınıflandırma ve regresyonda kullanılabilir) bir tekniktir (Breiman, 1996). Bagging yöntemini daha detaylı açıklama adına bir sonraki alt başlıkta kısaca bootsrap yöntemine yer verilmiştir.

3.2.4. Bootstrap

Bootstrapörnek temelli istatistiksel bir yöntemdir. Doğruluk tahmini için kullanılan yeniden örnekleme yöntemi olarak ifade edilen Bootstrap, küçük örneklem büyüklüğünü ele alır (Efron ve Tbrishani, 1993). Bu yöntemde birçok (ayrışık olmayan) eğitim verisi, tek bir ana veri setinden yer değiştirerek rastgele çekilir. "N" örnekli bir veri setinde yerine koyarak"N" örneklemin rastgele seçilmesiyle bir bootstrap eğitim veri seti oluşturulur. Her defasında bir örnek seçilir ve yeniden seçilen örneğin seçimi eşit olasılıklı şekilde gerçekleştirilir. Çekilen örnek, eğitim setine tekrar eklenir. Böylelikle bir eğitim setinde aynı örneğin birden fazla sayıda seçilme olasılığı olduğu gibi aynı zamanda örneğin hiç çekilmemesi de olasıdır. Bu durumda her bir örnek 1/N olasılıkla seçilir. Seçilmeme olasılığı ise;

$$\left(1-\frac{1}{N}\right)^{N} \approx \exp(-1) \approx 0.368 \qquad (3.1)$$

Şeklindedir. Yani veri setlerinin % 36.8'i test setini oluştururken % 63.2'si de eğitim seti için elde edilir (Efron ve Tibshirani,1993).

Güvenli bir modelin oluşturulabilmesi için eğitim ve test setlerinin seçimi çok önemlidir. Çünkü, eğer test seti, eğitim setini iyi temsil ederse, modelin performansının doğru tahminin elde edilmesi mümkün olur.

Hata tahminini elde etmek için kullanılan rastgele bootstrap örneklem sayısı "B" olmak üzere, örnekleme yöntemi B kez tekrarlanabilir ve bootstrap örneklemlerinin her biri modeli eğitmek için kullanılır. Modelin tahmin hatasını hesaplamak için elde edilen modeller, orijinal veri setine veya örnekleme dahil olmayan veriye uygulanarak, B defa tekrarlanır ve bootstrap hata tahmini örneklem üzerindeki ortalama tahmin hatası olarak elde edilir.

Orijinal örneklem kullanıldığında, eğitim seti ile test seti benzer olacağından model göreceli olarak iyi tahminler yapacaktır. Bu durumu azaltmak için Efron ve Tibshirani (1998)'nin "0.632 tahmin edicisi" kullanılır (Grubinger ve ark. 2010). Bootstrap tahminlerindeki aşırı uyumu (overfitting) gidermek için "0.632 bootstrap hata tahmin edicisi" aşağıdaki şekilde yazılabilir (Song ve ark., 2002).

$$hata/doğruluk_{boot} = \frac{1}{B}\sum_{i=1}^{B}[(0.632 * testhata_i) + (0.368 * toplamhata_i)] \qquad (3.2)$$

Burada B; hata tahminini elde etmek için kullanılan rastgele örneklem sayısını, $testhata_i$; i. bootstrap örneklemi sonucu elde edilen modelin i. test setine uygulandığında elde edilen hatayı, $toplamhata_i$ ise; i. bootstrap ile elde edilen modelin orijinal veri setine uygulandığında elde edilen hatayı belirtmektedir.

Klasik olarak bootstrap; N sayıda örnekler ile ilgili olarak $Z = \{Z_1, \dots, Z_N\}$, P populasyonu konusunda sınırlı sayıda büyük bazı istatistikler çıkarmak için kullanılır. Bu fikir B'den, T(P)'nin tahminlerini sağlayan Z'den her biri N rastgele örnekleri yer değiştirerek B kümelerinde $Z_b^* \subseteq Z$, b = 1,…….,B eşitliğini sağlamak içindir. Bu

tahminler daha sonra, son tahmin ortalamasına çevrilir ve böylelikle varyans tahmini ve güven aralıklarını sağlamak mümkün olur. Bu yöntem aşağıda ifade edilmiştir.

3.2.4.1. Bootsrap aşamaları

Girdi: N örneklem sayısı, P populasyon, $Z = \{Z_1, Z_2, \ldots, Z_N\}$, B ise; bootstrap örneklerinin sayısı olmak üzere;

Çıktı: Populasyon istatistiğinin tahmini $\hat{T}(P)$

$b=1$ için B

Yer değiştirme ile Z'den N örneklemine eğitim uygulanır, b. bootstrap örneği için Z_b^* elde edilir.

Her Z_b^* örneği için hesaplama yapılır ve istatistiğin tahmini $\hat{T}(Z_b^*)$ olur.

En sonunda sonlandırma işlemi yapılır.

$\hat{T}(Z_1^*), \ldots, \hat{T}(Z_B^*)$'in ortalaması olarak $\hat{T}(P)$ bootstrap tahmini hesaplanır.

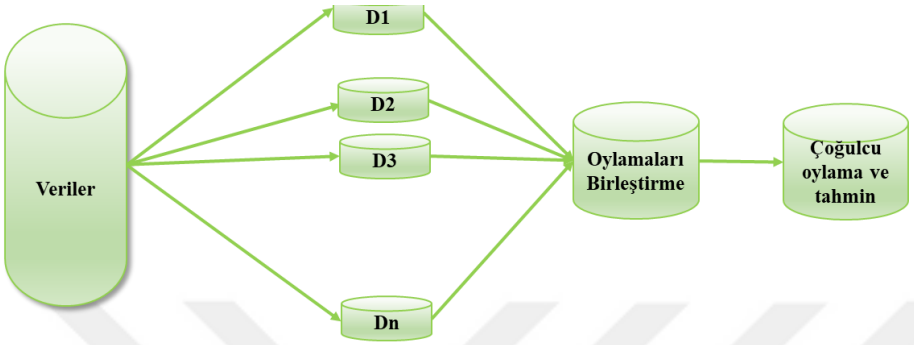
$\hat{T}(Z_1^*), \ldots, \hat{T}(Z_B^*)$ 'in varyansı örnek olarak kullanılarak, tahminin doğruluğu hesaplanır.

3.2.5. Bagging yöntemi ve çalışma sistematiği

n kadar bireyin olduğu $D = \{D_1, D_2, \ldots, D_n\}$ veri seti düşünelim. $D_i = (x_i, y_i)$, y_i sınıflandırma ya da regresyon problemlerinde bir reel sayı olmak üzere, bagging yönteminde amaç, Tkadar tahminleyicikümesini saptamak (t = 1,……, T) ve bu tahminleyicilerin kümesini birleştirerek (regresyon amaçlı modellerde ortalamasını alarak ya da sınıflandırma amaçlı modellerde çoğul oylama ile) son tahminleyici oluşturmaktır(Ferraira ve Figueiredo, 2007). Bu nedenle Bagging basitçe; T bağımsız sınıflandırıcılarını eğitir ve bunların her biri yer değiştirme yöntemleriyle örneklendirilir. n ya da n'nin bir kısmı, D eğitim verisinden örnekleme yapar ve çoğulcu oylamaya dayalı olarak karar verir (Şekil 3.3).

Beklenildiği gibi Bagging yönteminde bağımsız öğrenenlerin kombinasyonu hataların küçülmesini sağlayacaktır. Bundan dolayı, mümkün olduğu kadar bağımsız öğrenenlerin sayısını artırmak gerekir. Dolayısıyla Bagging yönteminde her seferinde kullanılan öğrenim veri setlerinin örtüşmemesi, başka bir ifadeyle her bir öğrenim veri

setinin diğerinden farklı olması çok önemlidir. Ancak sonsuz miktarda öğrenim veri seti oluşturmak mümkün olmayacığına göre, bootstrap ile oluşturulan bazı öğrenim veri setlerinde, ortak bilgiler bulunacak ve bu da öğrenicilerin tahmin performansında zayıflamaya neden olacaktır.



Şekil 3.3. Bagging yönteminin çalışma sitematiği.

Varsayalim ki öğrenim veri setimizde *"m"* kadar birey bulunsun. Bootstrap iadeli örnekleme yöntemi ile öğrenim veri seti için "*m"* büyüklüğünde örnekleme yapılır. Bu işlemi "*t"* kadar tekrar ettiğimizi düşünürsek, her biri "*m"* büyüklüğünde olan "*t"* kadar öğrenim veri setimiz oluşmaktadır. İadeli örnekleme yapıldığı için her bir "*m"* büyüklüğündeki veri setimizde bazen bir gözlem değeri birden fazla aynı örneğe girebildiği gibi, bazı gözlem değerleri de veri setine dahil olmayacaktır. Bootstrap ile oluşturulan her örnek için temel öğrenme algoritması kullanılarak temel öğrenici eğitilir. Dahasonra bagging yöntemi; sınıflandırma için çoğulcu oylama yaparak gerekli tahmini yapar. Eğer sınıflandırma değil de regresyonda olduğu gibi tahmin yapmak söz konusu ise, o zaman *t* kadar regresyon modelinden elde edilen parametre tahminlerinin ortalamasını alarak tahminleme yapar. Sınıflandırma veya regresyon için parametre tahminleri elde edildikten sonra bu tahminleri test veri setine uygulayarak performans değerlendirmesi yapar. Verilen bu açıklamalar aşağıdaki gibi özetlenir.

* Öğrenim için botstrap ile D_1, D_2,….,Dt, *t* öğrenim veri seti oluşturulur.

* Oluşturulan veri setinin öğrenimi başlatılır.

* Bir öğrenim algoritması kullanılarak (öğrenim algoritmasını Γ ile gösterelim) öğrenim sağlanır.

* $h_t = \Gamma(D_{bo})$, bootstrap ile oluşturulan her veri seti için sınıflandırma eğitimi yapılır. BuradaD_{bo} bootstrap dağılımı olmaktadır.

Coklu oylama ve sonuç

Yukarıda özetlenen aşamalar; öğrenim ve test olmak üzere iki aşama içerir. Öğrenim aşamasında her iterasyon için *t, t=1,...T,* N kadar öğrenim veri seti oluşturulur. Bu işleme bootstrap denilmektedir. Daha sonra temel bir model tercihedilir (örneğin kararağacı, yapay sinir ağları vs.) ve veriler öğrenimetabi tutulur.

Test aşamasında; her test döngüsünde T kadar öğrenim modelinden elde edilen sonuçlar birleştirilerek tahminleme yapılır. Bundan dolayı sınıflandırmayı esas alan çalışmalarda çoklu oylama, regresyon modellemelerinde ise bootstrap veri setlerinden elde edilen ortalama kullanılır.

Yukarıda verilen sistematik hem tahminleme hatasının düşük olmasını sağlar hem de modelin genelleştirme performansını iyileştirir. Bagging yöntemi değişken seçiminde, karar ağaçları gibi yöntemler için varyans azaltma tekniği olarak uyarlanmıştır ve aşırı uyumu engelleme özelliğine sahiptir (Breiman1996). Varyanstaki bu azalma ensemble olarak kullanılan sınıflandırıcı sayısı ile orantılıdır. İkili sınıflandırmalar için Bagging yöntemi aşağıda tanımlanmıştır.

Bagging nispeten küçük eğitim veri setleri problemlerinde en uygun yoldur. Bireysel sınıflandırıcılar çoğul oylama yoluyla kısımları birleştirmeden önce *"bite"* olarak isimlendirilen bu segmentlerde eğitilir (Breiman, 1999).

3.2.5.1. Sınıflandırma için Bagging işlemi

Girdi: Veri kümesi $D = \{D_1, D_2, \ldots, D_n\}$, olmak üzere B bootstrap örneklerinin sayısı $x_i \in X$ için $D_i = (x_i, y_i)$ ve $y_i \in \{-1, +1\}$ olur.

Çıktı: Son sınıflandırıcı $H: X \rightarrow \{-1, +1\}$ olur.

$b = 1$ alınsın.

D'den *n* örnekleri yer değiştirerek dizilir ve D_b^* örneği elde edilir.

Her bir bootstrap Z_b^* örneğinden H_b sınıflandırıcısı öğrenilir.

Sonlandırma işlemi yapılır.

Büyük bir çoğunluk tarafından oylanan son sınıflandırıcı ortaya konulur. $H_1, \ldots, H_\beta$ verildiğinde,

$$H(x) = sign\left(\sum_{b=1}^{B} H_b(x)\right) \tag{3.3}$$

şeklinde olur.

3.2.6. Out of Bagging

$Z = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ eğitim verisini ele alacak olursak, bu verilerden çekilen, her bir $T_{B,K} = 1, \ldots, B$ bootstrap örneklemi için model kurularak $Q\left(x, T_{B,K}\right)$ tahmini bulunmak istenirse, bagging tahmini;

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{Q}\left(x, T_{B,K}\right) \tag{3.4}$$

şeklinde tanımlanır (Hastie ve ark. 2001).

Genel olarak her bir bootstrap ağacı, orijinalinden farklılık gösterebilir ve farklı sayıda son düğüme sahip olabilir. İşte bu B tane ağaçtaki x'lerin oluşturduğu ortalama tahmin, bagging tahminini verir. Bir bootstrap örnekleminde eğitim verilerinin % 37'si örneğin dışında kalır. Bir tekrarda, örneğin çekilmeyen kısma "out of bag" veri, örneğin çekilmiş kısma ise "in bag" veri denir (Prasad ve ark. 2006).

OOB (Out of bag) veri bir ağacı budamak veya oluşturmak amacıyla kullanılmaz, fakat bagging tahminlerini genelleştirme ve düğüm hatası üzerinde daha iyi tahmin yapmasını sağlar. Eğitim verisinin örneği dışında kalan kısmını oluşturduğu ve yaklaşık % 37'lik oran aslında kullanılmayan test örnekleridir. Bundan dolayı regresyon ağaçlarında gerçek eğitim setinin cevap değerlerini kullanmak yerine, OOB tahminlerini kullanmak daha doğru regresyon ağaçlarının elde edilmesini sağlar.

OOB tahmini için her temel öğrenici için kullanılan öğrenim verilerini kaydetmemiz gerekir. OOB'yi $H^{oob}(x)$ olarak gösterelim. Bu durumda $H^{oob}(x)$;

$$H^{oob}(x) = \underset{y \epsilon Y}{\arg max} \sum_{t=1}^{T} (h_t(x) = y).(x \notin D_t) \tag{3.5}$$

Ve bagging yönteminin genelleştirme hatası:

$$err^{oob} = \frac{1}{|D|} \sum_{(x,y)\in D} (H^{oob}(x) \neq y) \tag{3.6}$$

OOB örnekler aynı zamanda, başka amaçlar için de kullanılabilir. Örneğin, karar ağaçları temel sınıflandırıcılar olarak kullanıldığında, her ağacın her düğümünün posterior olasılığı, OOB örnekler kullanılarak tahmin edilebilir. Eğer bir düğüm OOB örnekleri içermezse, "sayılamayan" olarak adlandırılır. Bir test örneği olarak, posterior olasılık, içine düştüğü sayısız olmayan düğümlerin posterior olasılığının ortalaması tarafından tahmin edilir.

3.2.7. Zayıf öğreniciler ve güçlü öğreniciler

Zayıf öğreniciler (WL) ve güçlü öğreniciler (SL) boosting yönteminin merkezinde olan temel kavramlardır. Bu kavramların kökeni olasılıklı yaklaşık doğru olarak ifade edilen PAC (probably approximately correct) öğrenim teorisine dayanır. Sınıflandırma kuralı,
$f: \chi \to \{-1, +1\}$ olan, $f \in \mathcal{F}$, $\mathcal{F}$'den $\chi \to$ {-1,+1} bazı fonksiyonlar sınıfı olan bir hipotezi göz önüne alalım. Bu hipotezin bir fonksiyon kümesini de ele alalım. Yani $\{(x_i y_i), i = 1, \ldots, N\}$ ikili küme, $y_i = f(x_i)$ ve x_iP dağılımının bazı örnekleri yeterince veri sağlandığında, güçlü bir öğrenici ve yüksek olasılıklı iyi sınıflandırıcı üretecek durumda olur. Başka bir ifadeyle, her bir P için $f \in \mathcal{F}, \varepsilon \geq 0 \; ve \delta \leq 1/2{,}1 - \delta$, den daha az olmayan olasılıklı, $\mathbb{P}_P[h(x) \neq f(x)] \leq \varepsilon$ yi sağlayan bir sınıflandırıcı $h: \chi \to \{-1, +1\}$ 'e üretir. Bununla beraber algoritma; $1/\varepsilon$, $1/\delta$, N'de en çok terimli ve χ' in boyutu olabilir (Meir ve Ratsch, 2006).

Boosting hipotezler uzayında pek çok farklı hipotezi birleştirmek yoluyla daha iyi bir hipotez ile sonuca ulaşır. Bu nedenle, boosting yöntemine ait büyük tahminleme gücü, birleştirilen hipotezin çeşitliliğinden ileri gelmektedir. Eğer boosting güçlü bir öğrenci kullanırsa, söz konusu çeşitlilik azalma eğilimi gösterecektir. Bunun sonucu olarak her iterasyonda hata giderek yok olma durumuna ulaşır. Böylece boosting karışık

modelleri, aşırı uyum ile tahmin etme tehlikesi yaratır. Bu bağlamda zayıf öğreniciye de ihtiyaç duyulmaktadır. Bu durumu ile boosting yönteminde defalarca zayıf öğrenci eğitilir. Her eğitimde ağırlıkları değiştirilir.

3.2.8. Boosting

Algoritmalar ailesi olarak boosting; WL'leri SL'lere dönüştürme anlamını taşımaktadır. Esas olarak WL şansa bağlı tahminden biraz daha iyi kabul edilmesine karşılık, SL mükemmele yakın bir performansı ifade etmektedir. Boosting zayıf sınıflandırıcılar grubundan, her biri gelişigüzel tahminden açıkça daha iyi sonuçlar veren, rastgele olarak az eğitim hatası yapabilen güçlü bir sınıflandırıcı oluşturmak için tekrarlı bir yaklaşım olarak tanımlanmıştır. Boosting SMV (Simple Majority Voting-Basit Çoğunluk Oylaması) kullanarak zayıf sınıflandırıcıların ensemble grubunu birleştirme koşuluyla, bagging yaklaşımından önemli bir noktada farklılık gösterir. Bagging yöntemi, tekli sınıflandırıcıları eğitmek için seçilen örneklerin eğitim verilerinin bootstrap ile tekrar edilmesidir. Bu durum her örneğin her bir eğitim veri setinde eşit derecede bulunma şansı anlamına gelmektedir. Oysaki boosting yönteminde her son sınıflandırıcı için eğitim veri seti, daha önce üretilmiş sınıflandırıcılar tarafından yanlış sınıflandırılan örneklere odaklanır. Bundan dolayı sınıflandırması doğru yapılmış olana daha büyük ağırlık değeri atanırken sınıflandırması iyi yapılmamış olana ise daha düşük ağırlık değeri atanmaktadır (Schapire, 1990).

İkili sınıf problemleri için tasarlanan boosting yöntemi tek seferde üç zayıf sınıflandırıcı kümesi oluşturur. İlk sınıflandırıcı h_1, bagging yöntemine benzer şekilde mevcut eğitim verisinin rastgele altkümesi üzerinde eğitilir. İkinci sınıflandırıcı h_2, yarısı yanlış sınıflandırılan ve yarısı da h_1 tarafından doğru tanımlanan orijinal veri kümesinin farklı alt kümesi üzerinde eğitilir. Böylesi bir eğitim altkümesi h_1 kararını veren "en bilgilendirici" olarak isimlendirilir. h_3 üçüncü sınıflandırıcı h_1 ve h_2'nin uyuşmadığı örneklerle eğitilir. Söz konusu bu üç sınıflandırıcı daha sonra "üç-yol çoğunluk oyu" aracılığıyla birleştirilir. Her bir sınıflandırıcının ikili sınıflandırıcı problem üzerine kurulu sınıflandırıcıdan en az beklenen $\varepsilon < 0.5$ hata oranına sahip olması şartıyla, Schapire bu üç ensemble sınıflandırıcısının eğitim hatasını, $g(\varepsilon) <$

$3\varepsilon^2 - 2\varepsilon^3$ile (burada ε üç sınıflandırıcıdan herhangi birisinin hatasıdır) sınırlandığını kanıtlamıştır (Zhang ve Ma, 2012).

3.2.8.1. Boosting algoritması

Boosting sınıflandırıcıların kombinasyonu WL'den sağlanan herhangi bir sınıflandırıcının tek başına gösterdiği performanstan çok daha iyisini gösterir. Boosting stratejisi ve sınıflandırıcıların topluluğu, tek bir güçlü sınıflandırıcı saptamaya çalışmak yerine birçok zayıf sınıflandırıcıyı elde eder ve bir şekilde bu sınıflandırıcıları bir araya getirir. Sınıflandırıcıların topluluğunu oluşturma fikri son yıllarda oldukça ilgi çekmektedir. Bunun nedeni birçok basit sınıflandırıcıyı eğitmek ve bu sınıflandırıcıları kullanarak karmaşık bir sınıflandırıcı oluşturmak için bir araya getirmenin tek bir kompleks sınıflandırıcı saptamaktan daha kolay olmasıdır. Örneğin büyük bir NN (sinir ağı) eğitmek yerine çok sayıda NN eğitebilir ve nihai sonucu oluşturmak için her birinin sonucunu bir araya getirebiliriz (Şekil 3.4).

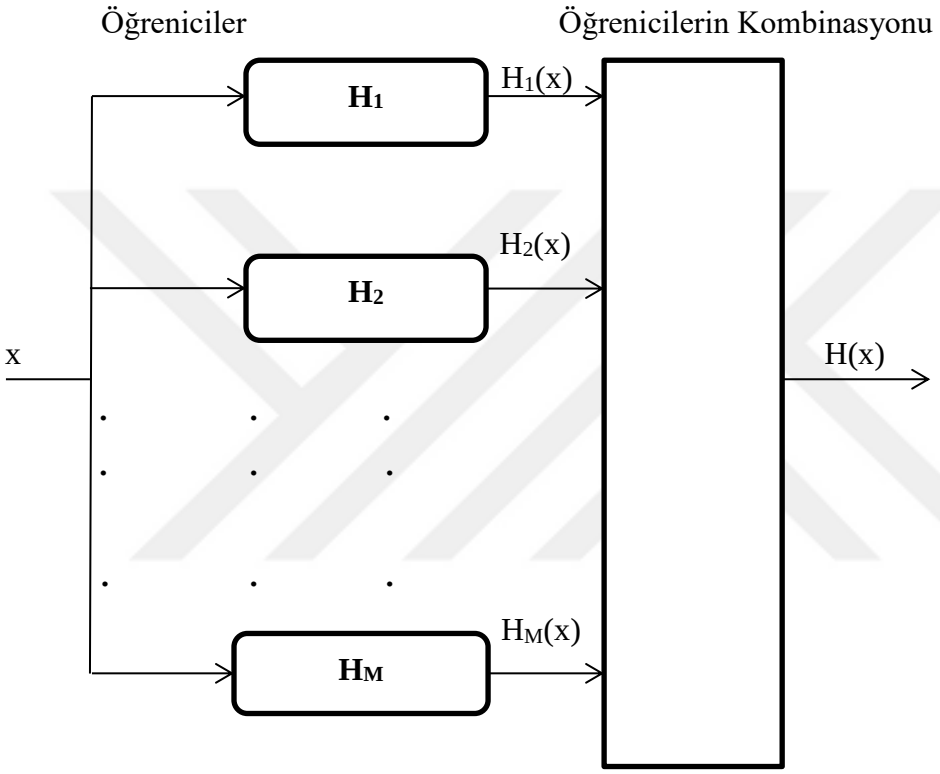
Zayıf sınıflandırıcıların sonuçları $H_m(x)$ ile $m \in \{1, \dots, M\}$ H (x) aracılığıyla elde edilen ensemble sınıflandırıcıların sonuçlarını oluşturmak için birleştirilirler.

$H_m: \aleph \to \{-1, +1\}$m'inci iki elemanlı zayıf sınıflandırıcı (m =1,…., M için) olarak kabul edelim, $x \in \aleph$ sınıflandırılacak girdi örüntüsü olsun, $H_1(x), \dots, H_M(x)$ sonuçlarını tekbir sınıf tahminine dönüştürmek için birçok seçeneğimiz vardır (James ve ark., 2017). Bu durumda; $H: \aleph \to \{-1, +1\}$ olur.$\alpha_1, \dots, \alpha_M$ bir ağırlıklar kümesidir ve eğer tüm ağırlıklar eşit ise basit çoğunluk oyu oluşur.

$$H(x) = sign\left(\sum_{m=1}^{M} \alpha_m H_m(x)\right) \tag{3.7}$$

Ensemble sınıflandırıcılarının eğitilebildiği ve birleştirildiği birçok değişik yöntem arasında boosting tekniği iyi uygulanabilir performansının yanında, bu tekniği cazip hale getiren teorik ve algoritmik özellikler de sergiler (Schapire, 2002; Meir ve Ratsch, 2006; James ve ark., 2017). Aslında Boosting, eğitim verisinin farklıversiyonlarına dayalı, "Eş. 3.7" de birleştirildiği gibi, zayıf sınıflandırıcı sıralaması üreten temel zayıf öğrenme algoritmasının tekrarlı kullanımından oluşur. Algoritmanın

her döngüsünde eğitim verisindeki her bir örneğin ağırlıklandırılması bir önceki sınıflandırıcının doğru olmasına dayanır. Böylece algoritmanın hala yanlış sınıflandırılan örneklere odaklanmasına izin verir. Boosting algoritmasının birçok çeşidi temel öğrenici seçiminde ve eğitim örneklerinin ağırlık güncellemesi için gereken kriterlerde farklılaşır. Adaptive Boosting algoritmasının kısaltması olan AdaBoost tartışmalı olarak en çok bilinen boosting algoritmasıdır (Freund ve Schapire, 1997).



Şekil 3.4. Ensemble sınıflandırıcılar konsepti.

3.2.8.2. Boosting prosedürü

İlk boosting işlemi Schapire tarafından önerilmiştir. Bu anahtar sonuç kolay ve zor öğrenmelerin denkliğidir. Zor öğrenme, WL'nin birleştirilmesi ile uygulanabilir. Boosting için işlem detaylı olarak aşağıdaki gibi tanımlanmıştır (Schapire ve Freund, 2010).

Sınıflandırma için Boosting işlemi

Girdi: $x_i \in X$ için $D_i = (x_i, y_i)$ ve $y_i \in \{-1, +1\}$ olmak üzere veri seti $D = \{D_1, D_2, \ldots, D_N\}$ olsun.

Çıktı: $H: X \rightarrow \{-1, +1\}$ bir sınıflandırıcı olmak üzere;

D'den D^*_1 $D_1^* - i$ 'i elde etmek için $L_1 < N$ örnekleri rastgele ve yer değiştirmeden seçilir.

H_1 sınıflandırıcısını sağlayarak D^*_1 , $D_1^* - i$ üzerinden WL'yi çalıştırarak H_2 sonucuna varılır.

D_i^*'i elde etmek için, H_1 tarafından yanlış sınıflandırılmış örneklerin yarısı ile D'den $L_2 < N$ örnekleri seçilir.

D_2^* üzerinden WL'yi çalıştırarakH_2 sonucuna varılır.

H_1 ve H_2anlaşmazlığı üzerinden Z' deki tüm örnekler seçilir ve D_3^* üretilir.

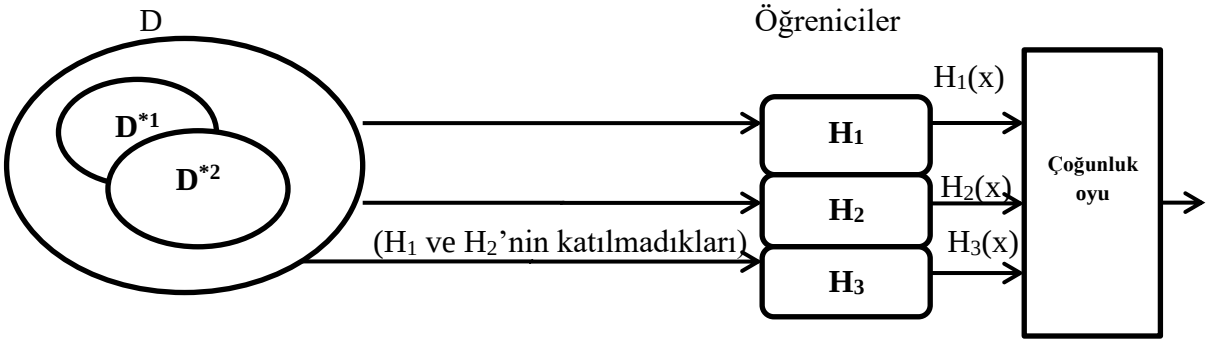
D_3^* üzerinden WL çalıştırılarak H_3 sınıflandırıcısı elde edilir.

Büyük çoğunluk oylaması olarak son sınıflandırıcı üretilir.

$$H(x) = sign\left(\sum_{b=1}^{3} H_b(x)\right) \tag{3.8}$$

Yukarıdaki algoritmada görülebileceği gibi; eğitim kümesi birbirinin yerini almadan D_1^*, D_2^*, ve D_3^*olacak şekilde rastgele üç parçaya bölünmüştür. Verilen örnek için, eğer ilk iki sınıflandırıcı (H_1 ve H_2) sınıf etiketi konusunda uzlaşır ise bu söz konusu örnek için son karardır. Uzlaşamadıkları örnekler kümesi H_3'ü saptamak için kullanılan D_3^* ile ifade edilir. Schapire bu saptama metodunun güçlü olduğunu göstermiştir. Dahası hata; bu yaklaşımı tekrarlayarak kullanma suretiyle daha da azaltılabilir. Yani her bir öğrenici, boosting yöntemi aracılığıyla kendiliğinden elde edilebilir (Şekil 3.5).

Önerilen ilk boosting yaklaşımının grafiksel ifadesine dikkat edildiğinde; herbir öğrenicinin, boosting algoritması aracılığıyla geri dönüşümlü tarzda öğrenebildiği görülür.



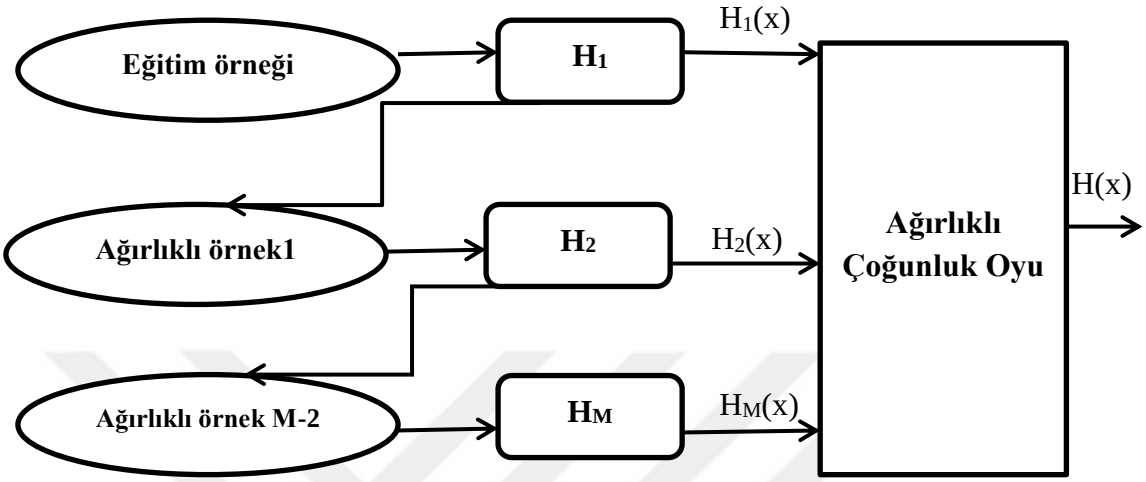
Şekil 3.5. Boosting yaklaşımının grafiksel ifadesi.

3.2.9. AdaBoost

Boosting Algoritmasının üzerindeki ilk bireysel çalışmalarından sonra, Schapire ve Freund (2010), Adaptive Boosting (AdaBoost) algoritmasını önermiştir. AdaBoost yöntemi başlangıçta AdaBoost.M1 olarak isimlendirilmiştir. Daha sonraları bu yöntemin regresyon dışında sınıflandırma amaçlı kullanılmaya başlanması ile AdaBoost olarak isimlendirilmiştir. Boosting algoritmaları, uyum eksikliklerini ölçüm şekline ve sonraki adımlarda gözlem ağırlıklarını ne şekilde seçtiklerine göre değişim gösterirler.Adaboost gibi orijinal boosting algoritmaları ikili sınıflandırma problemlerini geliştirmek için kullanılmıştır (Elith ve ark., 2008; Brownlee, 2016). AdaBoost yönteminin altında yatan temel düşünce; rastgele altörnekler yerine, bazı eğitim verilerinin ağırlıklandırılmış örneklerinin kullanılmasıdır. Aynı eğitim veri kümesi tekrarlanarak kullanılır ve bu nedenle önceki boosting yöntemlerinin aksine eğitimin çok geniş olması gerekmez.

AdaBoost algoritması çok iyi performanslı sınıflandırıcı toplulukları oluşturmak için çok popüler ve detaylı çalışılan bir yöntemdir (James ve ark., 2017). Algoritma, yapının son sınıflandırıcısını oluşturmak için WL'yi kullanarak, sınıflandırıcıların bir kümesini öğrenir. Zayıf sınıflandırıcılar önceki sınıflandırıcıların doğruluğuna bağlı ağırlıklar ile eğitim verisinin yeniden ağırlıklandırılmış versiyonunu kullanarak sıralı olarak elde edilir. Her bir eğitim veri seti, önceki sınıflandırıcıların doğru sınıflandırmasına veya yanlış sınıflandırılmasına göre ağırlıklandırılmıştır. Bu durum WL'nin her bir iterasyonda önceki zayıf sınıflandırıcılar tarafından iyi sınıflandırmayan öğreniciye odaklanmasına izin verir. Temel sınıflandırıcıları sağlamak için zayıf sınıflandırıcıları seçmek önemlidir. Bu durum, daha önce doğru sınıflandırılan

örneklerin ağırlıklarını önemli ölçüde azaltmadan öğrenmelerini sağlar. Eğer temel öğrenici çok güçlü ise yüksek doğruluğa ulaşır, sonraki iterasyonlarda belirli ağırlıkları tespit etmek için aykırı değerleri dışarıda bırakır.



Şekil 3.6. Adaptive Boosting algoritmasının grafiksel tasarımı (Hastie ve ark., 2001).

Adaptive Boosting algoritması, her bir WL eğitim veri örneklerinin farklı ağırlandırılmış versiyonu üzerine eğitilmiştir.

3.2.9.1. İkili sınıflandırma için kullanılan AdaBoost algoritması

Girdi: $x_i \in X$ için $D_i = (x_i, y_i) y_i \in \{-1, +1\}$ sınıflandırıcıların maksimum sayısı olmak üzere, veri kümesi $D = \{D_1, D_2, \ldots, D_N\}$ olsun.

Çıktı: $H: X \rightarrow \{-1, +1\}$ bir sınıflandırıcı olmak üzere;

Program kullanıma hazır hale getirilir. Ağırlıklar $w_i^1 = \frac{1}{N}, i \in \{1, \ldots, N\}\ ve\ m =$ 1'e göre kurulur.

$m \leq M$ iken, w_i^1 ağırlıklarını kullanarak Z üzerinden zayıf öğrenici için çalıştırılır. $H_m: X \rightarrow \{-1, +1\}$ sınıflandırıcısı sunulur.

err_m hesaplanır.

H_m 'nin ağırlıklandırılmış hatası $err_m = \sum_{i=1}^{N} w_i^{(m)} h\left(-y_i H_m(x_i)\right)$ olarak bulunur.

$\alpha_m = \frac{1}{2} log\left(\frac{1-err_m}{err_m}\right)$ olarak hesaplanır.

Her örnek için *i=1,..., N* olmak üzere ağırlık $v_i^{(m)} = w_i^{(m)} exp\left(-\alpha_m y_i H_m(x_i)\right)$ güncellenir.

Ağırlıkları yeniden normale döndürülür. i=1,…, N ve $w_i^{(m+1)} = v_i^{(m)} / S_m$ olmak üzere $S_m = \sum_{j=1}^{N} V_j$, $m \leftarrow m + 1$ olacak şekilde iterasyon uygulanır.

İşlem bitirilir.

Son sınıflandırıcı,

$$H(x) = sign\left(\sum_{j=1}^{M} \alpha_j H_j(x)\right) \tag{3.9}$$

Algoritmada kullanılan $h: \mathbb{R} \rightarrow \{0,1\}$ fonksiyonu Heaviside fonksiyonudur. Bu fonksiyon eğer $x \geq 0\ ise\ h(x) = 1, x < 0\ ise\ h(x) = 0$ olarak tanımlanır. Sonuç olarak, hem y_i hem de $H_m(x_i)$; {-1, +1} değerlerini aldığından, $y_i \neq H_m(x_i)$ ise $h\left(-y_i H_m(x_i)\right) = 1$ ve eğer $y_i = H_m(x_i)\ ise\ h\left(-y_i H_m(x_i)\right) = 0$ olur. err_m, m. sınıflandırıcısının ağırlıklı hata oranıdır (Ferraira ve Figueiredo, 2007).

Eğitim kümesinin ağırlıklandırılmış versiyonu üzerine zayıf bir belirleyici algoritması işletmek zayıf sınıflandırıcıyı sağlamak anlamına gelir. Diyelim ki H_m verilen *H* sınıflandırıcıların ailesine bağlı olsun. Bu durum bazı küçük pozitif ε'ler için

$$\sum_{i=1}^{N} w_i h\left(-y_i H_m(x_i)\right) \leq \frac{1}{2} - \varepsilon \tag{3.10}$$

eşitsizliği yazılır. 1,…….,N için sadece w_i= 1/N, örneğin, AdaBoost algoritmasının ilk iterasyon durumunda "Eş. 3.10" un sol tarafı ile klasik hata oranı, eğitim kümesinde kesişir. Bu türden zayıf sınıflandırıcıların oluşması boosting algoritmasının önemli bir bileşenidir (Meir ve Ratsch, 2006). Sınıflandırıcıların zayıflığı genellikle *H*'nin sadece basit sınıflandırıcıları içermesine izin vererek kontrol edilir. Örneğin, $X = \mathbb{R}^d$ durumunda, H ailesi sadece $H(x) = sign(u^T x + r)$ formunun doğrusal kuralını içerebilir. Bazen algılayıcı (perceptron) olarak ya da $H(x) = sign(ux_i + t)$ girdinin tek

elemanına bağlı kural olarak bilinen $u \in \mathbb{R}^d$ ve $r \in \mathbb{R}$ karar kökü olarak adlandırılan $u \in \{-1, +1\}$ ve $t \in \mathbb{R}$ olmaktadır.

AdaBoost.M1 algoritmasının kodu aşağıda verilmiştir. Örnek dağılım tüm eğitim veri örnekleri x_i, i=1, ...,N için her bir ardışık sınıflandırıcı (hipotez) h_t için eğitim veri alt kümelerinden S_t rastgele bir ağırlık atar. Dağılım düzgün dağılımlı (uniform) olarak başlatılır. Böylece, tüm örneklerin ilk eğitim veri setine girebilmesi eşit olasılıklı olur. h_t sınıflandırıcısının e_t eğitim hatası h_t($\varepsilon_t = \sum_i I[\![h_t(x_i \neq y_i)]\!]D_t(x_i)$, ($[\![h_t(x_i \neq y_i)]\!]$ (değişken doğru ise 1 değilse 0) tarafından yanlış sınıflandırılan örneklerin dağılım ağırlığı miktarı olarak hesaplanır. AdaBoost.M1 bu hatanın β_t'yi sağlamak için daha sonra normalize olan $1/2$'den daha az olmasını gerektirir.

Öyle ki

$0 < \beta_t < 1$ $için$ $0 < \varepsilon_t < 1/2$ olur.

3.2.9.2. AdaBoost.M1 algoritması

Girdiler: Eğitim verisi = $\{x_i, y_i\}, i = 1, \dots, N y_i \in \{\omega_1, \dots . \omega_C\}$ denetlenen öğrenici

Temel sınıflandırıcı: Ensemble ölçüsü T

$D_1(i) = 1/N$ olarak başlatılır.

$t = 1,2, \dots, T$' ye bakılır.

D_t dağılımından S_1 eğitim alt kümesini çizilir.

Temel sınıflandırıcıyı S_t üzerinden eğitilir, $h_t: X \to Y$ hipotezini alınır.

h_t hatası hesaplanır.

$$\varepsilon_t = \sum_i I[\![h_t(x_i \neq y_i)]\!]D_t(x_i) \tag{3.11}$$

Eğer $\varepsilon_t > 1/2$ ise dikkate alınmaz.

Kurulum yapılır.

$$\beta_t = \varepsilon_t/(1-\varepsilon_t) \tag{3.12}$$

Örnek dağılımı güncellenir.

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} \beta_t, & eğer h_t(x_i) = y_i \\ 1, & ise\ aksi halde \end{cases} \tag{3.13}$$

D_{t+1}'in uygun dağılım fonksiyonu olduğundan emin olmak için $Z_t = \sum_i D_t(i)$ fonksiyonu uygulanır.

Sonuç: Ağırlıklandırılmış Çoğunluk Oylaması: Verilen etiketlenmemiş *Z* örneği

$$V_c = \sum_{t:h_t(z)=w_c} log\left(\frac{1}{\beta_t}\right), \qquad c = 1,2,\ldots,C \tag{3.14}$$

Çıktı: En yüksek sınıf V_c

AdaBoost.M1 algoritması aslında "Eş. 3.13" te gösterilen dağılım güncelleme kuralıdır. Mevcut hipotez h_2 tarafından doğru sınıflandırılan örneklerin dağılım ağırlığı β_t faktörü kadar azalırken, yanlış sınıflandırılmış örneklerin ağırlığı değişmez. D_{T+1}'in uygun dağılım olduğunu anlamak için ağırlıklar güncellenip Z_t tarafından yeniden normalize edildiğinde, yanlış sınıflandırılan örneklerin ağırlıkları büyük oranda yükselir. Bundan dolayı gruba eklenen her bir yeni sınıflandırıcı ile Adaboost ağırlıklı olarak yanlış sınıflandırılan örneklere odaklanır. Her bir iterasyonda "Eş. 3.13" te *t* yanlış sınıflandırılan örneklerin ağırlığını artırırken, doğru sınıflandırılan örneklerin ise düşürür. Hem yanlış hem de doğru hesaplanan örnekler $1/2$'ye eşitlenir. Temel model öğrenme algoritması olan BC (Temel Sınıflandırıcı) $1/2$'den daha az hata gerektirdiği için daha önce yanlış sınıflandırılmış eğitim örneğinin, en az birini doğru şekilde sınıflandırmayı garantilemiş olur. AdaBoost bunu gerçekleştiremez ise işlemi tamamlayamaz, gerçekleştirir ise daha sonra WMV (ağırlıklandırılmış çoğul oylama) kullanarak birleştirilen T sınıflandırıcıları üretinceye kadar devam eder.

Tekli sınıflandırıcıların karşılıklı normalize edilmiş hatalarının AdaBoost.M1 ağırlıklı çoğul oylamasında oylama ağırlıkları olarak kullanıldığı belirtilecek olursa,

eğer bundan dolayı eğitim (lowβ_t) sırasında iyi performans gösteren sınıflandırıcıların, yüksek oylama ağırlıkları ile ödüllendirildiği görülür. Sınıflandırıcı performansı kendi eğitim verisinde sıfıra çok yakın olabileceğinden, β_t sayısal olarak sabit olamayacak kadar büyük olabilir. Oylama ağırlıklarında logaritma kullanılarak böylesi sorunlar giderilir.

$$E_{ensemble} < 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t(1-\varepsilon_t)} \tag{3.15}$$

$\varepsilon_t < 1/2$, $E_{ensemble}$ olduğu için, ensemble miktarı büyüdükçe ensemble hatası da düşer. Fakat ilginçtir ki; AdaBoost.M1'in hala ikili olmayan sınıf problemlerinde bile sınıflandırıcıların, $1/2$'den daha az hataya (ağırlıklı) sahip olmasını gerektirir. Bu düzeye ulaşmak, sınıf sayıları arttığı için oldukça zordur. Örneğin, el yazımı karakter tanımlama probleminde "1" ve "7" sayıları birbirine benzediği için sınıflandırıcı bu iki sınıfa da yüksek destek, diğerlerine ise düşük destek verebilir. AdaBoost.M1'deki hatanın aksine, AdaBoost.M2 seçilmeyen sınıflara verilen desteği kullanır ve $1/2$'den daha az olması gerekmeyen kayıp fonksiyonunu tanımlar. Fakat AdaBoost.M2'de, AdaBoost.M1'deki eğitim hatası için benzer bir üst sınır vardır. AdaBoost.R, temel olarak sınıflandırma hatalarının regresyon hataları ile değiştiren, tahmin fonksiyonu için tasarlanan diğer bir versiyondur (Freund ve Schapire, 1997).

3.2.10. Gradyan Boosting

Boosting yöntemine istatistiksel bakış fikri, fonksiyon uzayındaki en dik türevsel azalış (gradient descent) yoluyla, deneysel riski en iyi hale getiren Boosting algoritması tanımlanmıştır (Freidman 2001). Genel olarak, *Y* sonucu ile *X* tahmini değişkenlerini ilişkilendiren istatistiksel modelin regresyon fonksiyonu olan *f(.)* tahmin probleminin açılımı aşağıdaki gibi ifade edilebilir.

$$\hat{f}(.) = \underset{f(.)}{argmin}\{E_{Y,X}[\rho(Y, f(X))]\} \tag{3.16}$$

Burada *p(.)*, türevlenebilir kayıp fonksiyonunu simgeler. En yaygın kayıp fonksiyon olan L_2 kaybı $\rho(y, f(.)) = (y - f(.))^2$ ortalamanın, en küçük kare yoluyla bulunan regresyon modelini oluşturur $f(x) = E(Y|X = x)$. Pratikte gözlemlerin öğrenme örneğinde $(y_1, x_1), \dots, (y_n, x_n)$ deneysel riski, en uygun hale aşağıdaki gibi getirilir.

$$\hat{f}(.) = argmin_{f(.)} \left\{ \frac{1}{n} \sum_{i=1}^{n} \rho\big(y_i, f(x_i)\big) \right\} \tag{3.17}$$

Yönteme, rastgeleliği dahil etmek hem işlem hızını hem de doğru tahmin yapma olasılığını arttırır. Bu nedenle, alt örnek tüm eğitim veri setinden rastgele seçilir. Bu seçim model güncelleşmesini oluşturmak için gerekli olduğu gibi, aynı zamanda aşırı uyum sorununa karşı da sağlamlığı arttırır (Mayr ve ark., 2014).

Gradyan boosting yönteminin esas amacı, AdaBoost algoritmasında olduğu gibi temel öğrenicileri yeniden ağırlıklandırarak yerleştirmek değil, bir önceki iterasyonda değerlendirilen kayıp fonksiyonunu olan $u^{[m]}$'yi negatif gradyanvektörüne uydurmaktır.

$$u^{[m]} = \left(u_i^{[m]}\right)_{i=1,\dots,n} = \left(-\frac{\partial}{\partial f}\rho(y_i, f)\Big|_{f=\hat{f}^{[m-1]}(.)}\right)_{i=1,\dots,n} \tag{3.18}$$

L_2 kayıp fonksiyon ve m iterasyon sayısı olmak üzere, $\rho(y, f(.)) = \frac{1}{2}\big(y - f(.)\big)^2$ basitçe $y - f(.)$ kalanlarından yeniden yapılanmasına yardımcı olur. Bir önceki iterasyon $y - f(.)^{[m-1]}$ kadar olur.

Bu durumda, hem AdaBoost hem de gradyan boosting yöntemlerinin aynı ana fikri vurguladığı daha da netleşir. Her iki algoritmada da, tahmini zor olan problematik gözlemlere olan odaklanmayı iterasyonla değiştirme yoluyla, basit temel öğrenicinin performansını arttırır. Adaboost ile bu değişiklik, daha önce yanlış sınıflandırılmış ağırlığı yükseltilmiş gözlemler ile yapılır. Gradyan boosting ise daha önceki iterasyonlarda hesaplanmış büyük kalıntılar tarafından oluşan zor gözlemleri tanımlar (Mayr ve ark., 2014).

Başlatma

İterasyon sayacı kurulur

$m = 0$. $\hat{f}^{[0]}$ ile toplamsal tahmini başlatılır. Örneğin $\hat{f}^{[0]} \coloneqq (0)_{i=1,\dots,n}$

Bir dizi temel öğrenici özelleştirilir. $h_1(x_1), \dots, h_p(x_p)$.

Negatif gradyan uygunlaştırılır.

$m \coloneqq m + 1$ olacak şekilde kurulum yapılır.

Daha önceki yinelemede değerlendirilen kayıp fonksiyonun u negatif gradyan vektörü hesaplanır.

$$u^{[m]} = \left(u_i^{[m]}\right)_{i=1,\dots,n} = \left(-\frac{\partial}{\partial f}\rho(y_i, f)\Big|_{f=\hat{f}^{[m-1]}(.)}\right)_{i=1,\dots,n} \tag{3.19}$$

Her bir temel öğreniciye ayrı ayrı $u^{[m]}$ negatif gradyan vektörü uyarlanır.

$$u^{[m]} \xrightarrow{temelöğrenici} \hat{h}_j^{[m]}(x_j)\ j = 1, \dots, p \tag{3.20}$$

Bir Bileşeni Güncelleme

Negatif gradyan vektörüne en iyi uyan bileşen j^* olarak seçilir.

$$j^* = argmin_{1 \le j \le p} \sum_{i=1}^{n} \left(u_i^{[m]} - \hat{h}_j^{[m]}(x_j)\right)^2 \tag{3.21}$$

Toplamsal tahmin edici $\hat{f}$ fonksiyonu bu bileşen ile güncellenir.

$$\hat{f}^{[m]}(.) = \hat{f}^{[m-1]}(.) + sl.\hat{h}_{j^*}^{[m]}(x_{j^*}) \tag{3.22}$$

Burada sl küçük adım uzunluğudur $(0 < sl \ll 1)$. 0.1'de pratikteki tipik değerdir.

İterasyon

$m = m_{stop}$ olana kadar 2. adımdan 6. adıma kadar iterasyon uygulanır.

3.2.11. Boosting yönteminde “Aşırı Uyum" (Overfitting) sorunu

AdaBoost yönteminin içeriği hakkında süregelen tartışma, onun aşırı uyum özelliğidir. Aşırı uyumdan kurtulmak için algoritmanın temel görevi, eğitim örnekleri için en iyi olası sınıflandırıcıyı bulmak değil, yerine yeni bir gözlem kümesi için en iyi tahmin kuralını bulmak olmalıdır.

AdaBoost yönteminin gereğinden fazla iterasyon içerecek şekilde çalıştırılması (çok geç durdurulması) aşırı uyumu kolaylaştırabilir. Çünkü son bölümün karmaşası artar. Diğer yandan algoritmayı gerektiğinden erken durdurmak yalnızca eğitim verisinde yüksek hataya yol açmaz, aynı zamanda yeni veride daha zayıf bir tahminle sonuçlanır (yetersiz uyum (underfitting)). AdaBoost yönteminin içeriğinde, algoritmanın aşırı uyum gösterebilmesine rağmen sıklıkla aşırı uyuma dirençli olduğu da görülür (Grove ve Schuurmans, 1998; Ratsch, 2001; Bühlmann ve Hothorn, 2007; Zhou, 2012; Schapire ve Freund 2012).

AdaBoost algoritmasının son çözümündeki veri setinde olan genelleme hatası, temel öğrenicinin karmaşası ve boosting iterasyonlarının sayısı olan artı eğitim hatası ile bağlantılıdır. Bu Occam’sRazor olarak bilinen geniş çapta kabul edilmiş görüş tarafından desteklenmiştir (Blumer ve ark, 1987). Esas olarak bu görüş; tahminler ve daha karmaşık sınıflandırıcılar, eğer her ikisi de aynı miktarda bilgi taşıyorsa, daha az karmaşık olan tarafından daha iyi yapılabilir olarak açıklanabilir. Fakat bu teorik sonuç AdaBoost algoritmasının sıklıkla aşırı uyuma dirençli olduğu fikri tarafından desteklenmez. Çünkü AdaBoost’un son karmaşa çözümü Occam'sRazor'u takip eden iterasyonu durdurmasına bağlıdır ve daha zayıf sonuçlar doğurur (Zhou, 2012).

AdaBoost’un performansı genellikle doğru sınıflandırma oranının değerlendirilmesiyle ölçülmektedir ve aşırı uyuma direnç sadece bu özel kritere odaklanarak gösterilmektedir. Fakat AdaBoost tarafından uygun hale getirilen bu kriter, aslında doğru sınıflandırma oranı olmayıp üsse ait kayıp fonksiyondur ve aynı tahmin tarafından uygun hale getirilmesi gerekmeyen iki kriterdir. Bu nedenle, AdaBoost

algoritmasının aşırı uyum davranışı, sadece üsse ait kayıp fonksiyona odaklanarak, analiz edilmesi şeklinde de savunulmuştur (Bühlmann ve Hothorn, 2007).

3.2.12. Bagging ve Boosting yöntemlerinin karşılaştırılması

Bagging yöntemi varyans azaltma tekniği olarak değerlendirilip, eğitim algoritmalarının tahmin hatasını azaltmak için kullanılırken, Boosting yöntemi temel yöntemin parametre tahmininde sapma miktarını azaltır (Coşgun ve ark., 2011).

Bagging ve Boosting üzerine yapılan bir çok çalışma göstermiştir ki, Boosting algoritması Bagging algoritmasına göre daha üstündür (Elith ve ark., 2008; Brownlee, 2016).

Regresyon tahmin edicisi; Bosting yönteminde de Bagging yönteminde de eğitim setinin farklı alt setleri üzerinde eğitilir. Bagging yönteminde her bir eğitim setinin N örneğinden yerine koyarak çekilen N örneklemi üzerinde eğitilirken, eğitim setinden alınan her örneğin bootstrap örnekleminde görülme şansı eşit olur. Boosting sıralı bir yöntemdir. Bagging yönteminde olduğu gibi eğitim setinde yerine koyarak çekilen N boyutlu örneklem üzerinde eğitilir. Bu örnekler arasında hatası en fazla olan bu örneklerin, ikinci eğitim setinin üyeleri olarak öğrenme olasılıkları daha yüksek olacak şekilde düzenlenir.

Boosting, Bagging ve Bootstrap algoritmalarınınürettiklerine ve eğitim verilerini nasıl ele aldıklarına göre aralarındaki ilişki incelenmiş ve bu üç tekniğinde rastgeleörnekleme üzerine kurulduğunu vurgulanmıştır (Çizelge 3.1).

Bootstrap ve Bagging algoritmaları örneklemeyi yerine koyarak yaparken Boosting bu yöntemi kullanmaz. Bagging ve Boosting algoritmalarının ortak noktası ikisinindeherbir sınıflandırıcının çoğunluk oyları olan son sınıflandırıcıyı sağlamalarıdır (Ferraira ve Figueiredo, 2007).

Deneysel sonuçlar çok az ya da hiç sınıflandırma gürültüsü olmayan durumlar için rastgele sıralama Bagging ile eşdeğerdedir (belkide biraz daha üstündür). Fakat Boosting kadar doğru değildir. Önemli oranda sınıflandırma gürültülü durumlarda Bagging yöntemi, Boosting yönteminden çok daha etkili ve bazen de rastgele sıralamadan daha etkilidir.

Çizelge3.1. Eğitim verisi işlemleri ile Bootstrap, Bagging ve Boosting arasındaki ilişki.

	Yer değiştirme ile rastgele örnekleme	*Yer değiştirme olmaksızın rastgele örnekleme*
İstatistiksel doğruluk testini sergiler	Bootstrap	
Sınıflandırmayı Sergiler	Bagging	Boosting

Bu çalışmada MATLAB istatistik yazılım programında Bagging ve Boosting yöntemleriyle sınıflandırma yapılmıştır. Ayrıca 2010-2013 yıllarına ait YGS ve LYS sonuçları; Bakanlar Kurulu tarafından 13 mart 2001 tarihinde kabul edilmiş ve 24 Mart 2001 tarihli Resmi gazetede yayımlanan "AB (Avrupa Birliği) Müktesebatının Üstlenilmesine İlişkin Türkiye Ulusal Programı"na göre DÜZEY1, DÜZEY2 ve DÜZEY3 şeklinde gelişmişlik düzeyine göre bölgelere ayrılarak "Bagging" ve "Boosting" yöntemleri kullanılarak sınıflandırma işlemleri yapılmıştır. AB genelinde tek bir veri tabanı oluşturmak, bölgesel istatistikleri standartlaştırmak ve bunu yaparken de karşılaştırılabilir bir tablo ortaya çıkarmak için bölgelerin sahip oldukları benzer niteliklere göre oluşturulan İBBS (İstatistiki Bölge Birimleri Sınıflaması), Türkiye'de örnek bölge birimi uygulaması olarak kabul edilmiş ve Türkiye İstatistik Kurumu'nun desteği ile Devlet Planlama Teşkilatı tarafından 2002 yılında tamamlanmıştır. İBBS'yegöre Türkiye DÜZEY1 olarak 12, DÜZEY2 olarak 26, DÜZEY3 olarak da 81 İstatistiki Bölge Birimi olarak ayrılmıştır.

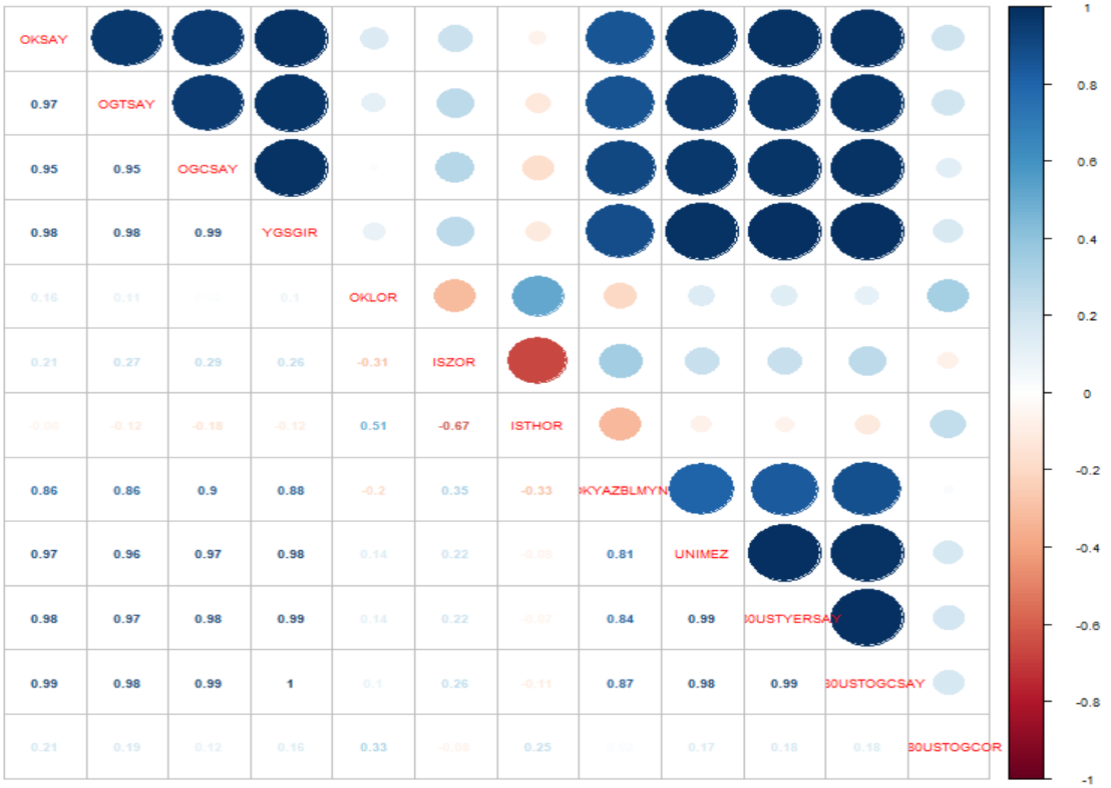
4. BULGULAR

Bu çalışmada kullanılan verilere uygulanan Bagging ve Boosting yöntemlerinden elde edilen sonuçları vermeden önce, korrogram, bootstrap kümeleme ve iki yönlü kümeleme (biclustering) tekniklerinden elde edilen sonuçlar değerlendirilmiştir.

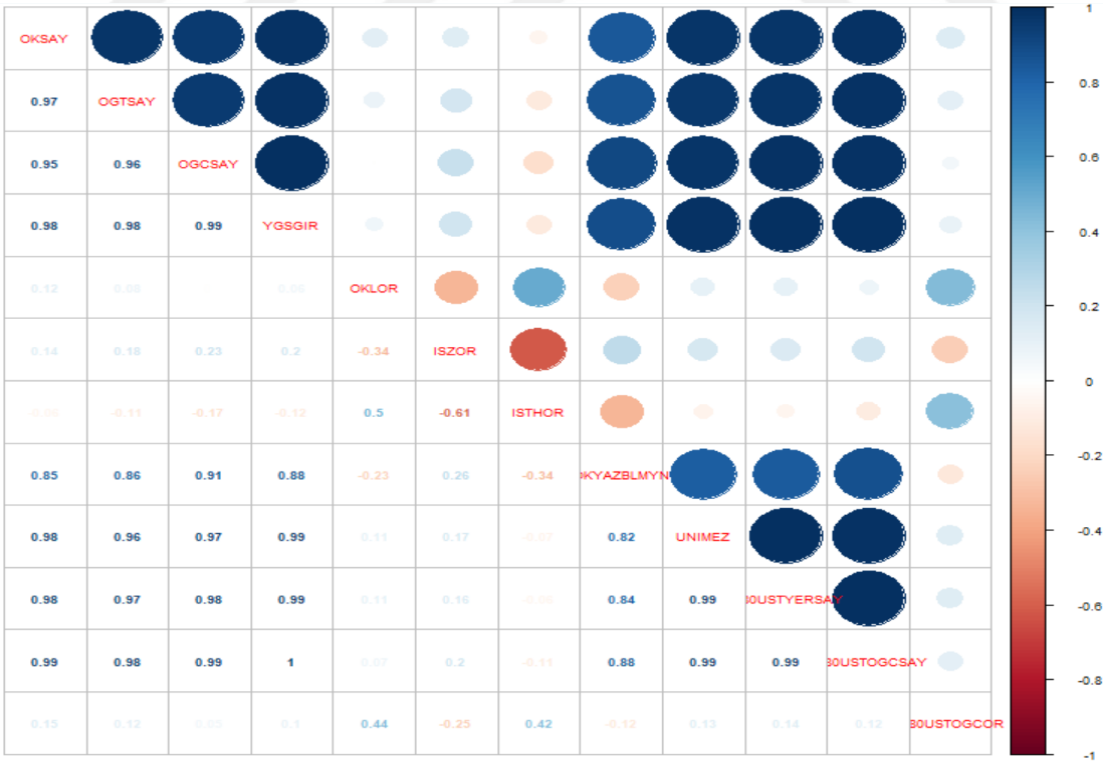
4.1. Değişkenler Arasındaki İlişkinin Korrogram Yöntemi ile İncelenmesi

Bu çalışmada kullanılan 2010-2013 yıllarına ilişkin çalışmaya konu edilen öğrenci sayısı (OGCSAY), okullaşma oranı (OKLOR), okuma yazma bilmeyen sayısı (OKYAZBLMYN), okul sayısı (OKSAY), öğretmen sayısı (OGTSAY), işsizlik oranı (ISZOR), istihdam oranı (ISTHOR), üniversite mezunu sayısı (UNIMEZ), YGS'ye giren öğrenci sayısı (YGSGIR), 180 ve üstü puan alıp yerleşen sayısı (180USTYERSAY), 180 ve üstü puan alan öğrenci sayısı (180USTOGCSAY), 180 ve üstü puan alan öğrenci oranı (180USTOGCOR) değişkenler arasındaki ilşkiler matrisi korrogram yöntem kullanılarak grafiksel olarak değerlendirilerek sonuçları verilmiştir (Şekil 4.1, Şekil 4.2, Şekil 4.3, Şekil 4.4). Şekillerde de anlaşıldığı gibi OKSAY, OGTSAY, OGCSAY, YGSGIR, OKLOR, ISZOR, ISTHOR, OKYAZBLMYN, UNIMEZ, 180USTYERSAY, 180USTOGCSAY, 180USTOGCOR değişkenleri arasında hem negatif hem de pozitif yönlü kolerasyonlar gözlenmiştir.

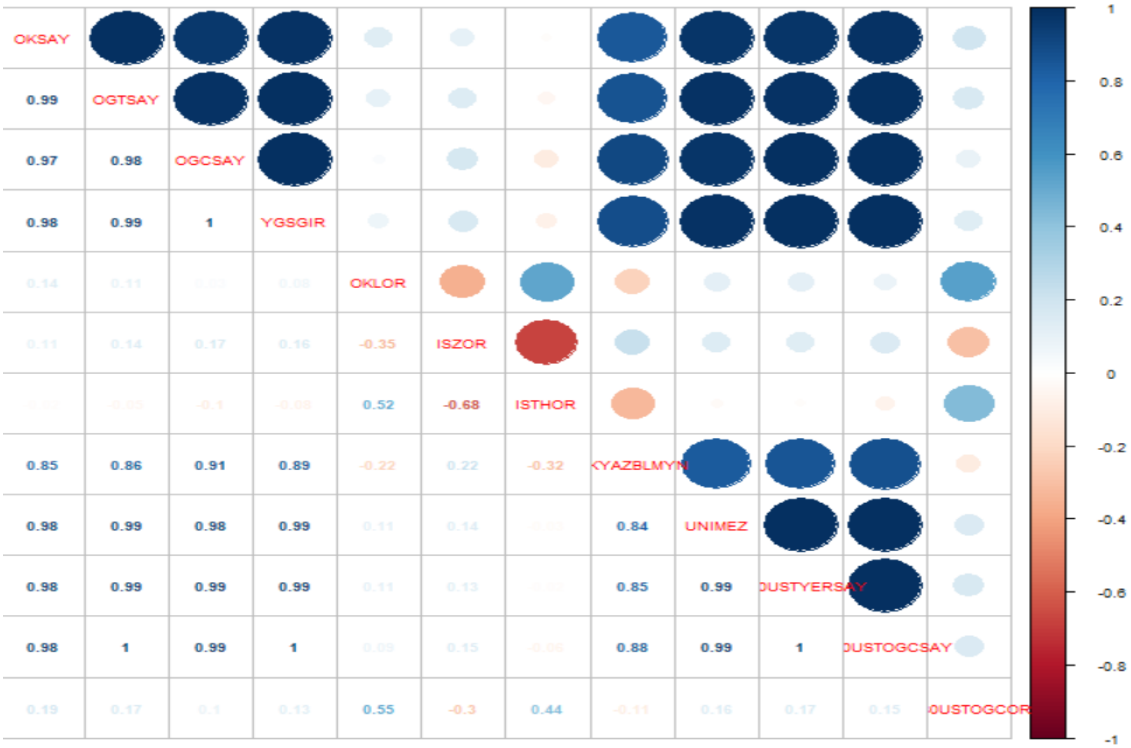
Şekiller incelendiğinde değişkenler arasında genel olarak yüksek düzeyde pozitif bir korelasyon olduğu görülmektedir. Yıllara göre değerlendirildiğinde ve ilk olarak 2010 yılına ait değişkenler arasındaki ilişki ele alındığında okul sayısı ile öğretmen sayısı arasında 0.97 oranında pozitif korelasyon olduğu, aynı şekilde öğretmen sayısı ile öğrenci sayısı ve okul sayısı ile öğrenci sayısı arasında 0.95 oranında pozitif korelasyon olduğu belirlenmiştir (Şekil 4.1). Bunun yanı sıra işsizlik oranı ile istihdam oranı gibi iki demografik değişken arasında -0.67 oranında negatif korelasyon olduğu dikkat çekmektedir. En yüksek negatif korelasyon, söz konusu bu iki değişken arasında gözlenmiştir. Ayrıca, okuma yazma bilmeyen sayısı ile istihdam oranı arasında -0.33 büyüklüğünde negatif korelasyon olduğu görülmektedir.



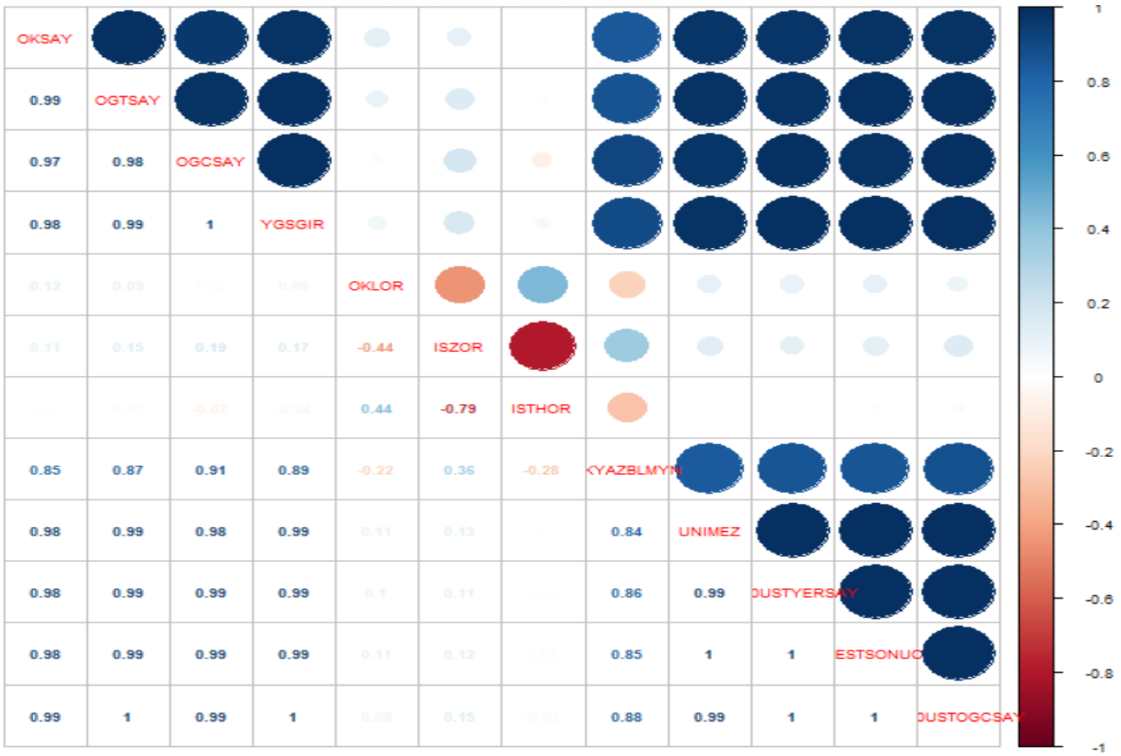
Şekil 4.1. 2010 yılına ait değişkenler arasındaki korelasyon.



Şekil 4.2. 2011 yılına ait değişkenler arasındaki korelasyon.



Şekil 4.3. 2012 yılına ait değişkenler arasındaki korelasyon.



Şekil 4.4. 2013 yılına ait değişkenler arasındaki korelasyon.

Korelasyon yapısı bakımından bireysel olarak diğer yıllar tek tek ele alındığında, 2010 yılına ilişkin değişkenler arasında var olan korelasyon matrisine benzer bir görüntünün var olduğu gözlemlenmiştir. Bu nedenle değişkenler arasındaki pozitif ve negatif korelasyon yapısının 2010 yılına paralel bir görüntü sergilediği ve çok büyük değişikler içermediği görülmektedir. Örneğin; 2010, 2011 ve 2012 yıllarında öğrenci sayısı ile YGS'de 180 puan ve üzeri puan alarak lisans, ön lisans ve açık öğretim fakültesine yerleşen öğrenci sayısı arasında 0.98 büyüklüğünde pozitif korelasyon varken 2013 yılında bu değer 0.99 olmuştur. İşsizlik ve istihdam oranı arasındaki negatif korelasyon değeri ise 2010 yılında -0.67 iken 2011 yılında -0.61, 2012 yılında -0.68 ve 2013 yılında -0.79'dur.

4.2. Yıllara Göre İllerin Kümeleme Analizine Göre İncelenmesi

R paket programı kullanılarak (pvclust kütüphanesi) yıllara göre hiyerarşik kümeleme analizi ile değerlendirme yapılmıştır. Hiyerarşik kümeleme içerisindeki her küme için çoklu-ölçekleme yoluyla, bootstrap örneklemesi için kullanılan bir *p*- değeri hesaplanır. Her bir kümeleme için hesaplanan *p*- değeri (0-1) aralığında olur. Hesaplanan *p*- değeri, kümelemenin eldeki verilen tarafından nasıl desteklendiğini belirtir. Kullanılan pvclust kütüphanesi iki farklı *p* değeri verir ve bu değerleri, kümeleme grafikleri üzerinde kırmızı ve yeşil renklerle belirtilir. Kırmızı renk ile verilen *p* değeri yaklaşık sapmasızlık (Approximately Unbiased- AU), yeşil renkte verilen *p* değeri ise bootstrap olasılığı (Bootstrap Probability- BP) anlamına gelmektedir. Kümelemenin doğruluğunu belirlemede yaklaşık sapmasızlığı ifade eden (kırmızı renkli) *p* değeri göz önünde bulundurulur.

Bootstrap örnekleme yöntemi kullanılarak yapılmış kümeleme analizi sonuçları sunulmuştur (Şekil 4.5, Şekil 4.6, Şekil 4.7, Şekil 4.8). Söz konusu şekillerde, verilen *p* değerleri ve oluşturulan kümeler 5000 bootstrap örneklemesinden elde edilerek ortalama değer kullanılmıştır.

Hiyerarşik kümelemede 81 ile göre 2010-2013 yılları arasında inceleme yapıldığında öncelikle iki kümenin oluştuğu ve bunların da kendi arasında alt kümeler oluşturduğu görülmektedir. İlk bakışta alt kümelerde yaklaşık sapmasızlık değerlerinin çok yüksek olduğu, hiyerarşik y ilerledikçe bu değerin daha da düştüğü görülmektedir.

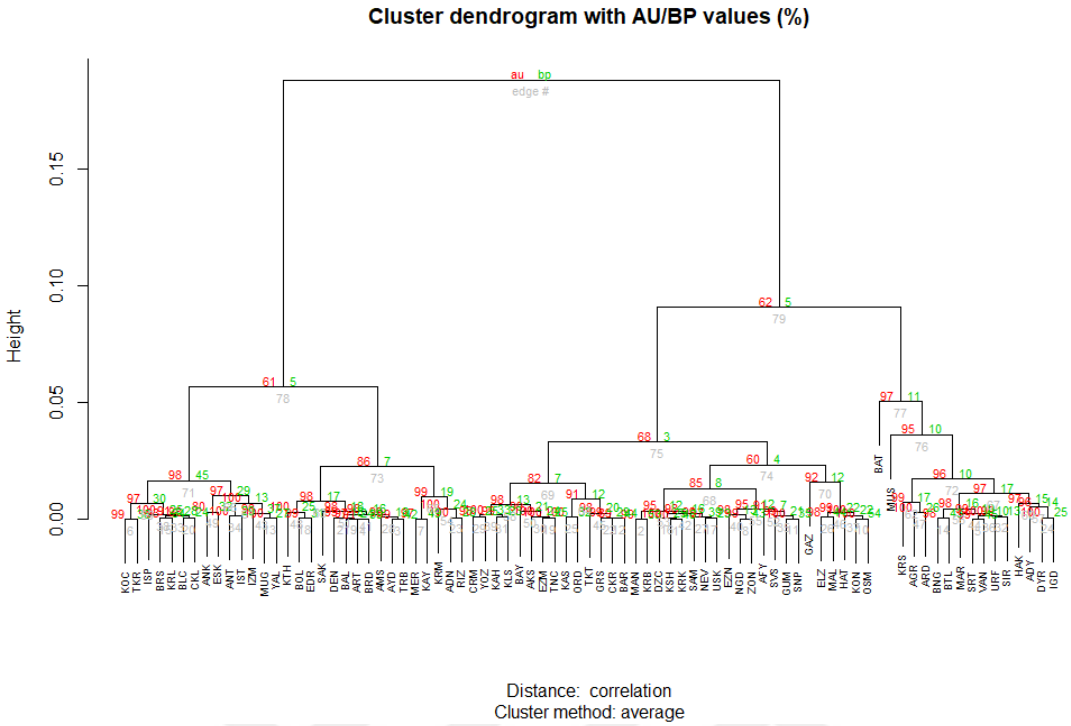
Başka bir anlatımla, alt küme oluşturmada belirsizliğin daha az olduğu, birçok alt kümeyi bir araya getirip daha büyük bir küme oluşturmada belirsizliğin de arttığı sonucuna varılmaktadır. Örneğin 2010 yılında Iğdır, Diyarbakır, Adıyaman illerinin tek bir alt kümede bulunmasına ilişkin yaklaşık sapmasızlık değeri 100 iken (sıfır belirsizlik) bu illere Hakkari'nin de dahil olması durumunda yaklaşık sapmasızlık 96'ya inmiştir. Benzer sonuç diğer alt ve üst kümelerde de gözlenmiştir. Örneğin Kkocaeli ve Tekirdağ'ın tek bir kümede bulunmasına ilişkin yaklaşık sapmasızlık 99 iken (bir birim belirsizlik), bu illere Isparta, Bursa ve Kırklareli'nin de dahil olması ile oluşacak kümelemede yaklaşık sapmasızlık 97'ye inmiştir (Şekil 4.5).

Bütün yıllar tek tek ele alınıp ve kendi aralarında karşılaştırıldığında başarı oranı ve diğer değişkenler bakımından illerin oluşturdukları alt ve üst kümelerin birbirinden çok da farklı olmadığı görülmektedir. Örneğin; 2010 yılında Batman diğerlerinden kısmen farklılık gösterse de Kars, Ağrı, Ardahan, Bingöl, Bitlis, Mardin, Siirt, Van, Şanlıurfa, Şırnak, Hakkari, Adıyaman, Diyarbakır ve Iğdır ortak özellik taşıyıp benzer özellikler göstererek, ortak bir grupta yer alarak kümelenirken, Kocaeli ve Tekirdağ, Ankara ile Eskişehir, Muğla ile Yalova kendi arasında benzer özellik göstererek kümelendiği görülür.

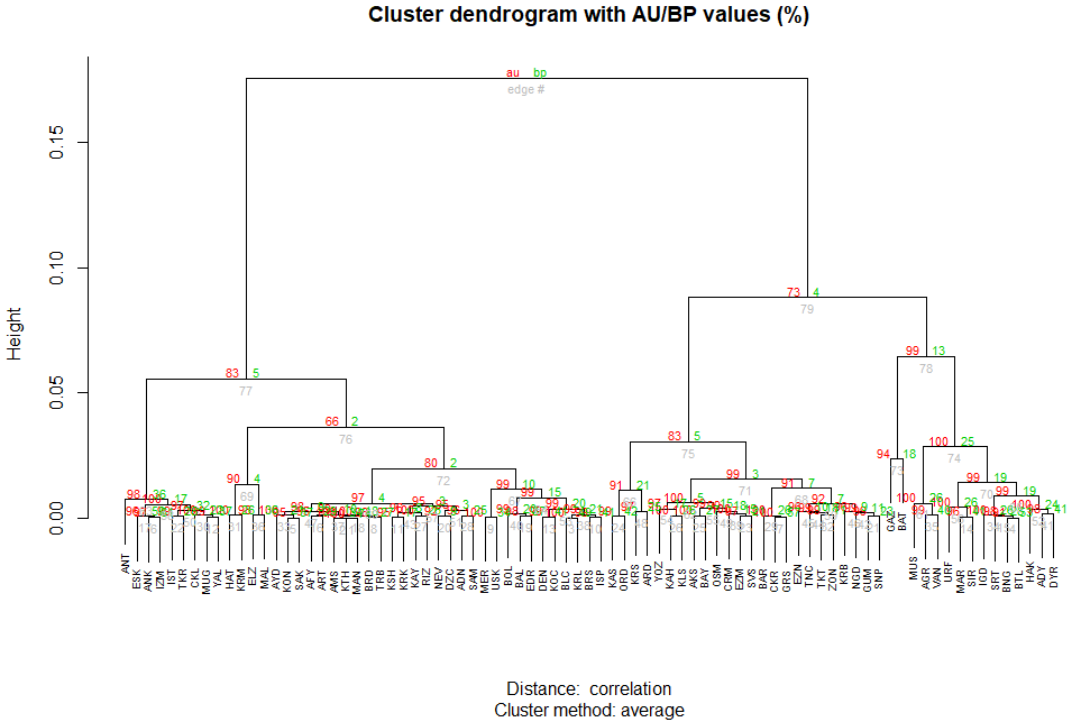
2011 yılına bakıldığında Nevşehir ve Düzce, Adana ile Samsun arasında benzer özellikler bulunmakta ve bu iki grup kendi arasında da yine benzer özellikler taşımaktadır. 2010 yılına benzer şekilde 2011 yılında da Antalya, Eskişehir-Ankara-İzmir, İstanbul-Tekirdağ, Çanakkale-Muğla-Yalova grupları da ortak özellikleri olan şehirler arasındadır.

2012 ve 2013 yıllarına bakıldığında çok belirgin bir değişikliğin olmadığı görülmektedir.

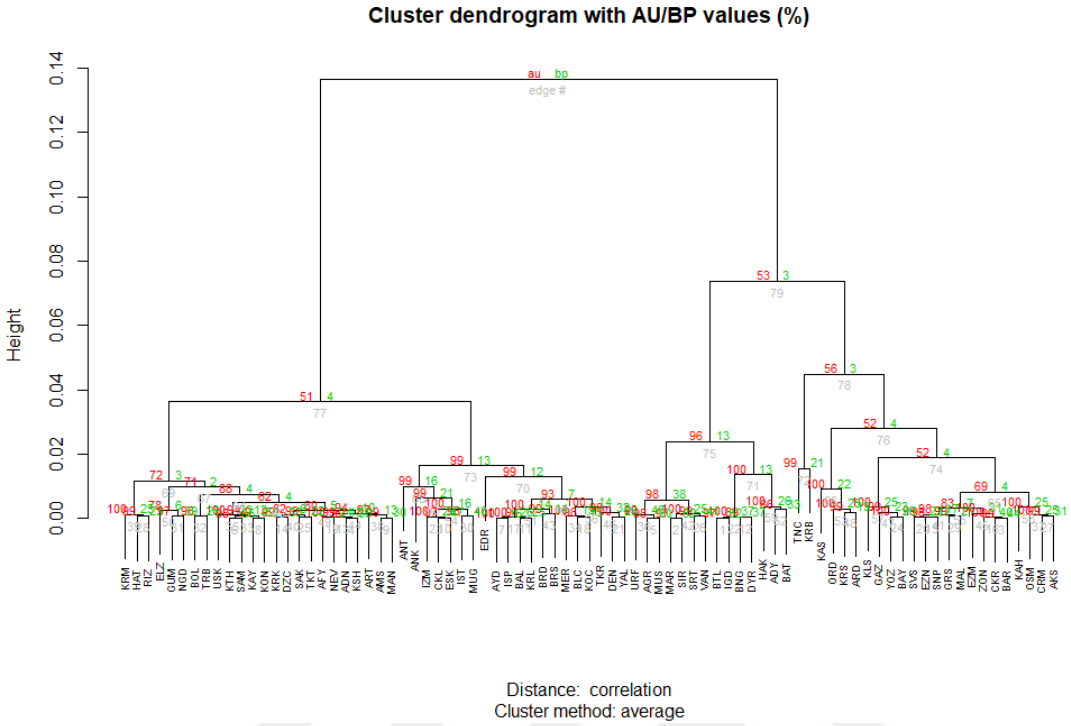
Burada en çok dikkat çeken Doğu Anadolu'da yer alan illerin başarı oranları bakımından kendi arasında, Marmara, Batı Ege ve Batı Akdeniz'de yer alan illerin ise kendi arasında gruplaşmış olmasıdır. İç Anadolu ile Orta ve Doğu Karadeniz'de bulunan iller ile Gaziantep, Malatya ve Elazığ arasında öğrencilerin başarı oranı bakımından ortak özellikler sergiledikleri dikkati çekmektedir.



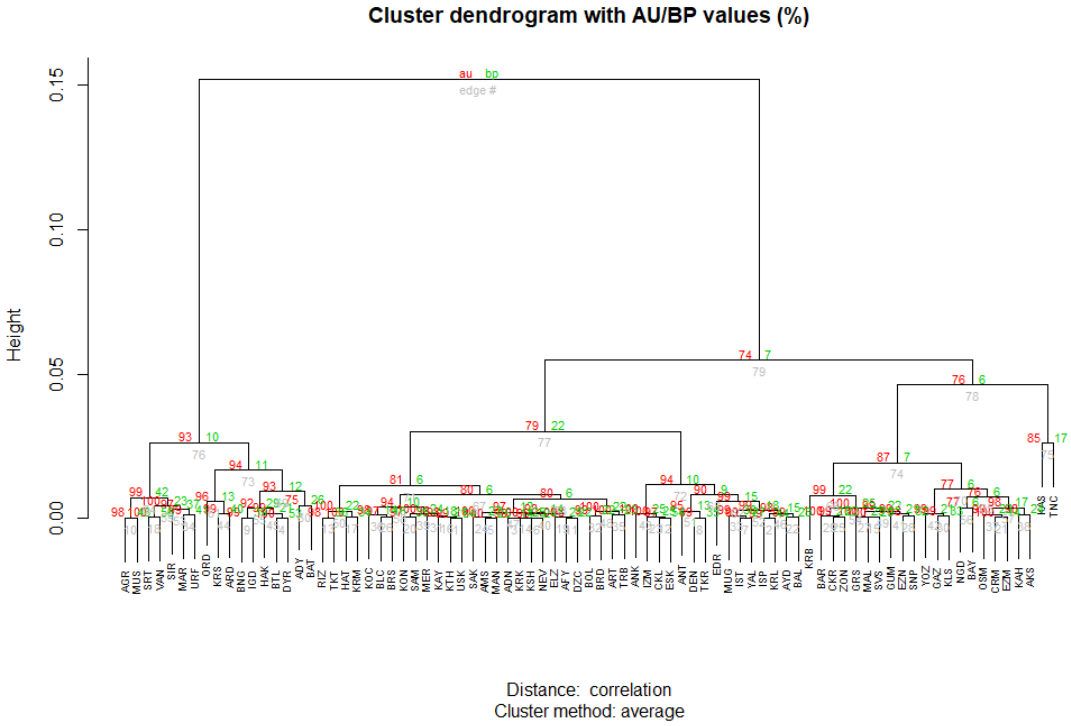
Şekil4.5. 2010 yılına ait iller arasındaki kümeleme analizi sonuçları.



Şekil4.6. 2011 yılına ait iller arasındaki kümeleme analizi sonuçları.



Şekil4.7. 2012 yılına ait iller arasındaki kümeleme analizi sonuçları.



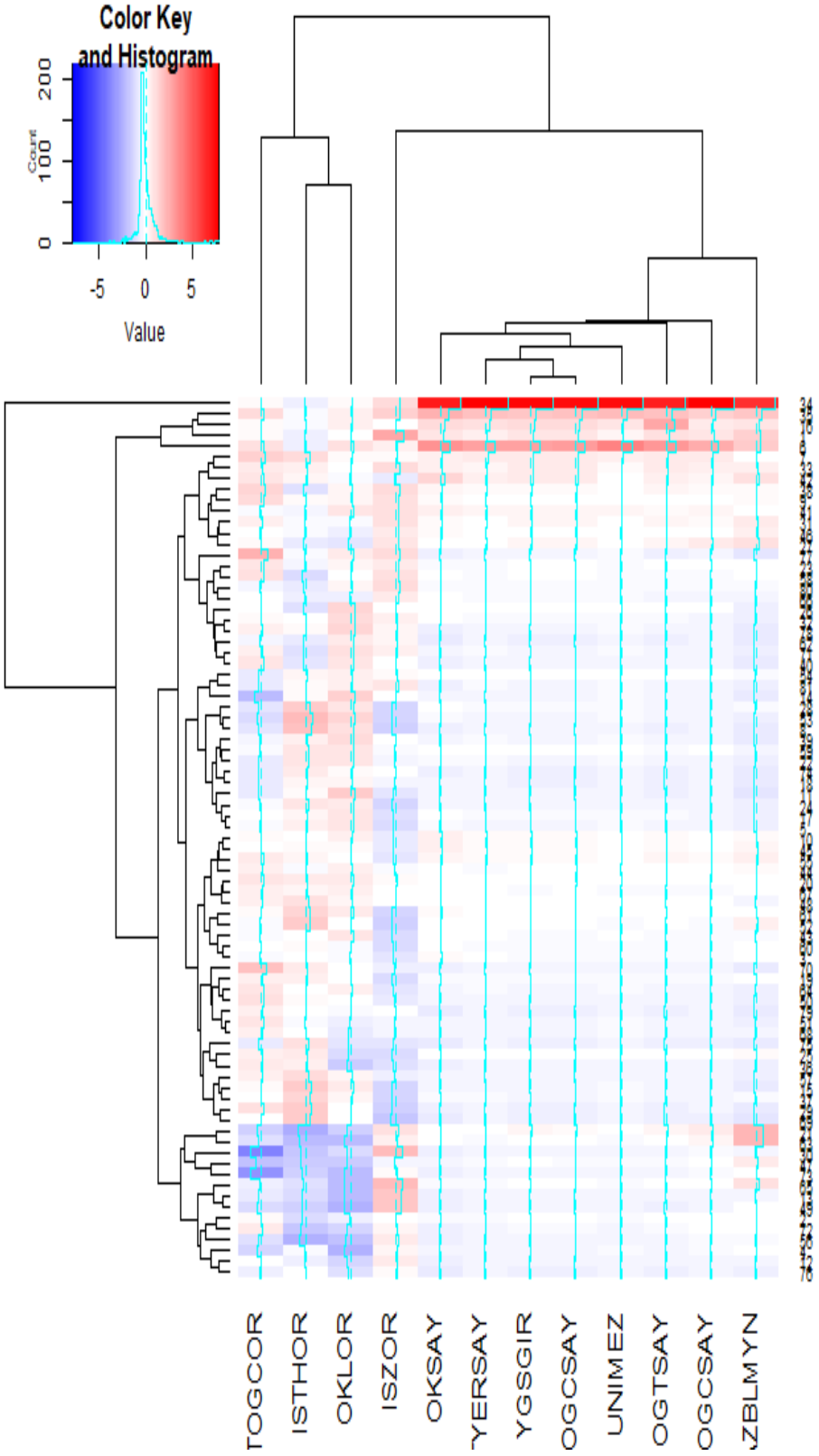
Şekil 4.8. 2013 yılına ait iller arasındaki kümeleme analizi sonuçları.

4.3. İki Yönlü Kümeleme Analizi ile İller ve Değişkenlerin Birlikte Değerlendirilmesi

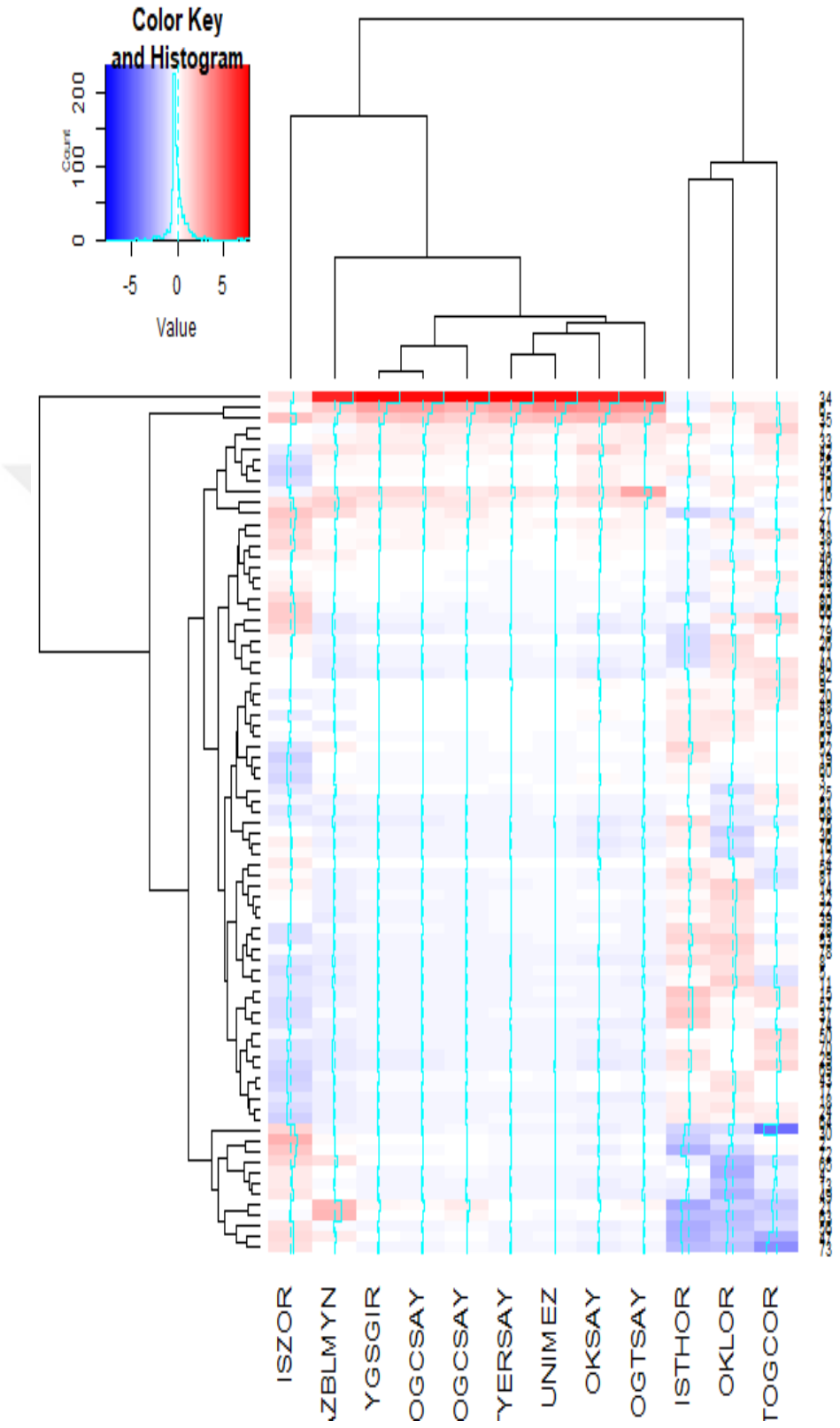
Bu kısımda, iller ile değişkenlerin birlikte hiyerarşik kümelemede konumlandığı bilgisi verilmiştir. İller plaka numaralarına göre kodlanmış olup, değişkenler de kısaltılarak grafik üzerinde gösterilmiştir. İki yönlü kümeleme oluşturulmadan önce değişkenler yeniden ölçeklendirilerek birimlerden bağımsız hale getirilmiştir. İki yönlü kümelemede koyu mavi ile gösterilen yerlerde negatif yönden, kırmızı olan bölgelerde ise pozitif yönden oluşan alt kümeler sergilenmektedir. Burada dikkat çeken kısım, dört yılda da İstanbul'un diğer illerden ayrı bir grup oluşturması ve okuma yazma bilmeyen sayısı, öğrenci sayısı, öğretmen sayısı, üniversite mezunu sayısı, öğrenci sayısı, YGS'ye giren öğrenci sayısı, 180 ve üstü puan alarak üniversiteye yerleşen öğrenci sayısı ve okur yazar sayısı ile pozitif yönde ortak bir özellik göstermesidir. Yine Ankara ve İzmir için de aynı değişkenlerin İstanbul kadar pozitif yönde ortak özellik taşımasa dahi belirgin bir şekilde pozitif ortak özellik taşıdığı görülmüştür.

2010-2013 yılları bir arada incelendiğinde özellikle Iğdır, Bingöl, Siirt, Batman, Muş, Bitlis, Van, Hakkari ve Urfa'nın; okullaşma oranı, istihdam oranı ve YGS'de 180 ve üstü puan alan öğrenci oranı değişkenleri üzerinde özellikle negatif yönde ortak bir özellik oluşturduğu saptanmıştır (Şekil 4.9, Şekil 4.10, Şekil 4.11, Şekil 4.12).

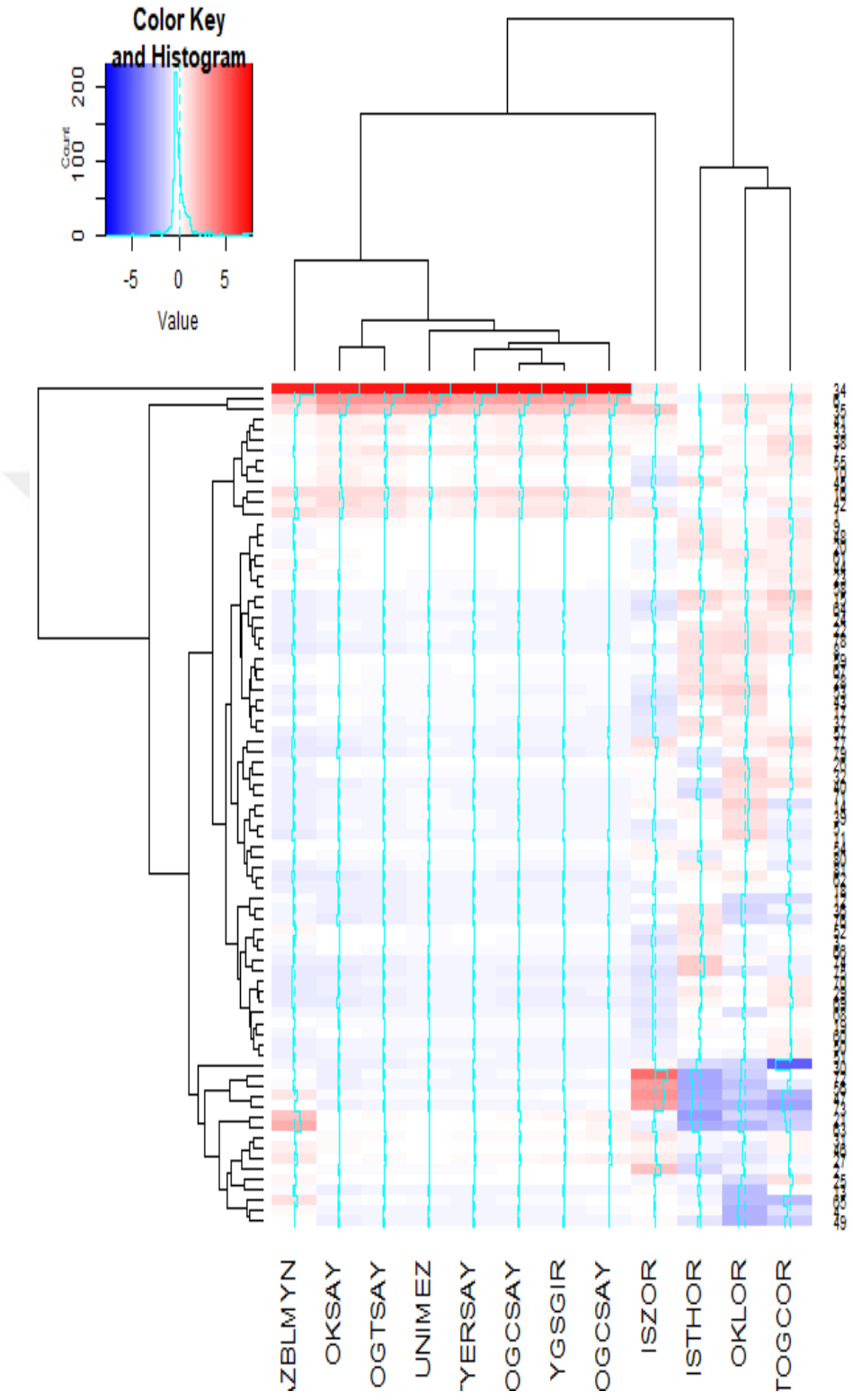
Dikkat çeken bir başka durum ise okullaşma oranı değişkeninin dört yılda da özellikle Artvin, Bilecik, Amasya, Karabük, Kırklareli, Giresun ve Rize'de pozitif yönde ortak özellik göstermiş olmasıdır. Bununla birlikte aynı değişken Şanlıurfa, Siirt, Ağrı, Van, Muş, Hakkari ve Bitlis illeri için negatif yönde ortak özellik gösterme eğiliminde olmuştur.



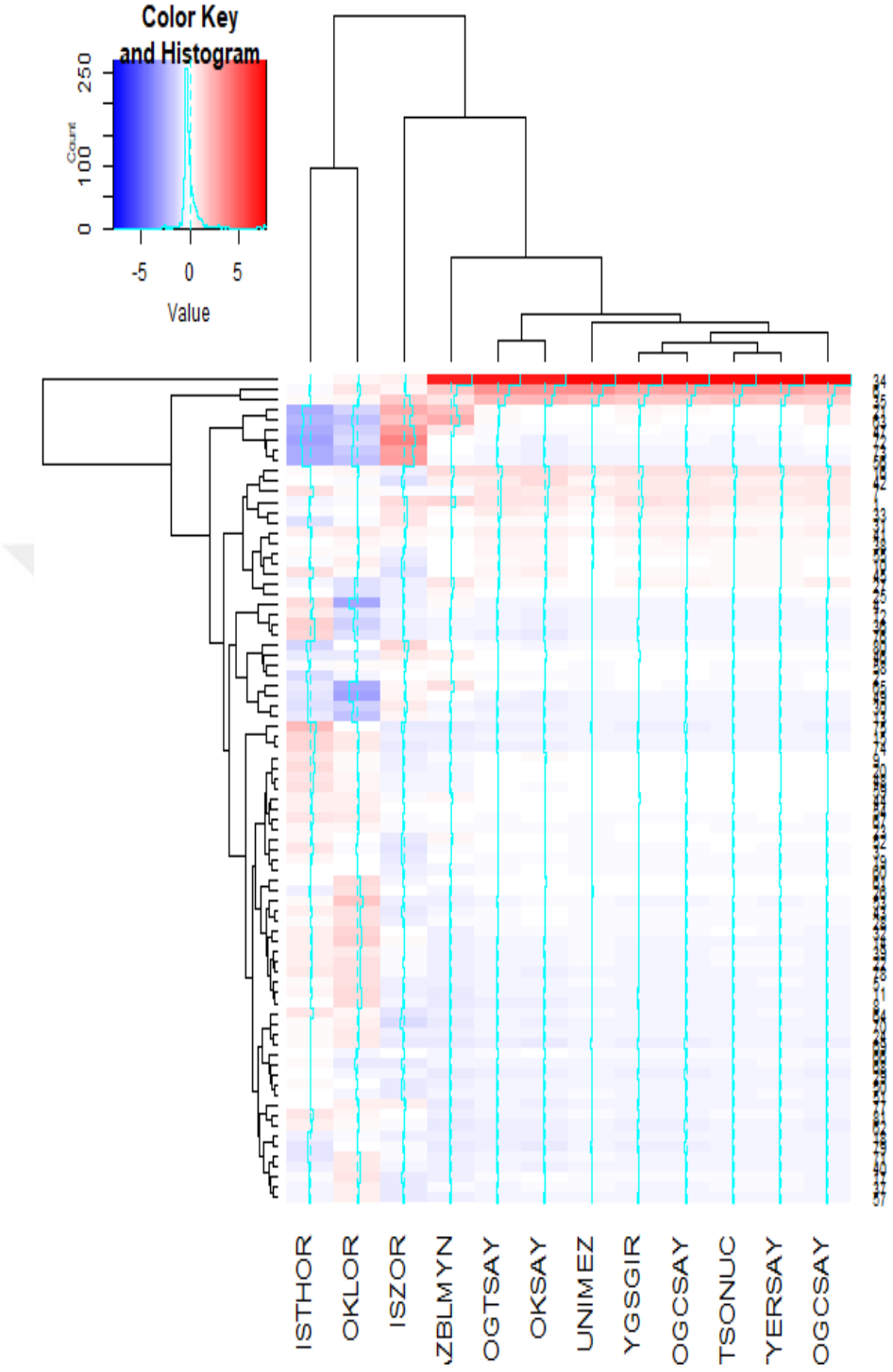
Şekil4.9. 2010 yılına ait iller ve değişkenler arasında iki yönlü kümeleme analizi sonuçları.



Şekil 4.10. 2011 yılına ait iller ve değişkenler arasında iki yönlü kümeleme analizi sonuçları.



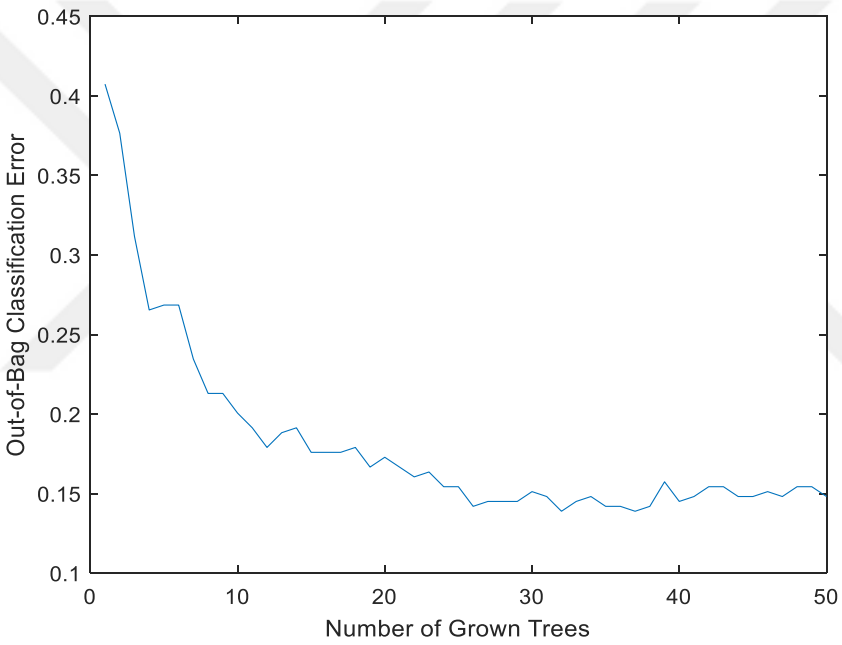
Şekil 4.11. 2012 yılına ait iller ve değişkenler arasında iki yönlü kümeleme analizi sonuçları.



Şekil 4.12. 2013 yılına ait iller ve değişkenler arasında iki yönlü kümeleme analizi sonuçları.

4.4. Bagging Yöntemiyle Sınıflandırma Sonuçları

Bagging ve Boosting sınıflandırma performanslarının daha iyi değerlendirilmesi için öğrenicilerin başarı oranları iki grup halinde gruplandırılmıştır. Buna göre ortalamanın üstünde kalan iller 1, ortalamanın altında kalan iller ise 0 olarak kodlanarak bağımlı değişkenler oluşturulmuştur. Bagging ve Boosting ensemble algoritmaları buna göre çalıştırılmıştır. Bagging ve Boosting algoritmalarının tahminleme yeteneklerini değerlendirmek amacıyla, veri seti eğitim ve test şeklinde ikiye bölünmüştür. Bu amaçla, 2010-2012 yılları arasındaki veriler eğitim (training) verileri olarak kullanılırken 2013 verileri ise test (testing) verileri olarak kullanılmıştır.

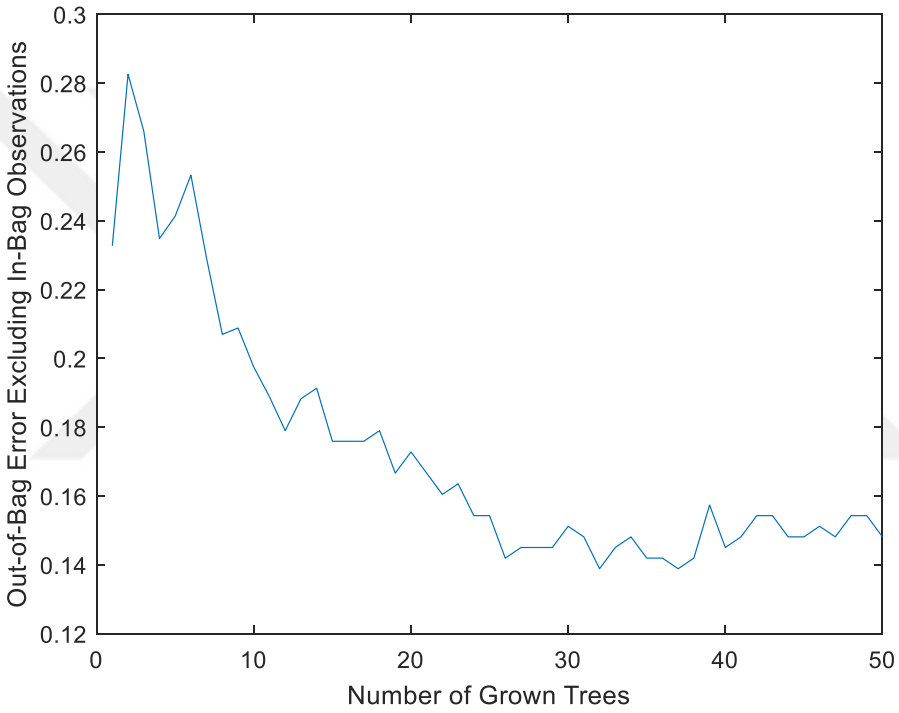


Şekil 4.13. Out-of-Bag sınıflandırma hatası.

Bagging ve Boosting gibi yönlendirici (supervised) makine öğrenim yöntemlerinin performanslarını değerlendirmede küçük hata değerleri ve hata varyansı, önemli kriterlerden biri olarak göz önünde bulundurulur. Ağaç sayısı arttırıldıkça, hatanın da azalma eğilimi ele alınarak değerlendirme yapılır (Şekil 4.13). Aşırı uyum sorunu olmadığı sürece ağaç sayısı arttıkça sınıflandırma hatasının düşmesi beklenir. Çalışmamızda ağaç sayısı 32 iken hatanın en düşük olduğu görülmüştür. Fakat genel anlamda bakıldığı zaman 25. ağaçtan sonra sınıflandırma hatasında büyük bir değişiklik olmadığı ve kısmen stabil bir duruma dönüştüğü görülmektedir. Ağaç sayısını

kararlaştırmada OOB hatasının en ciddi dirsek yaptığı yerdeki ağaç sayısı göz önünde bulundurulur. Bu çalışmada en ciddi dirsek oluşumu 10. ağaçta meydana gelmiştir (Şekil 4.13).

Pratik uygulamalarda yüzlerce ağaçla ensemble modelleri büyütülebilir. Örneğin daha hızlı gelişim süreci için 50 ağaç kullanılarak optimum yaprak sayısı belirlendiği gibi, 100 ağaçla da daha büyük bir ensemble modeli oluşturularak, özellik sayısı tahmin edilebilir.

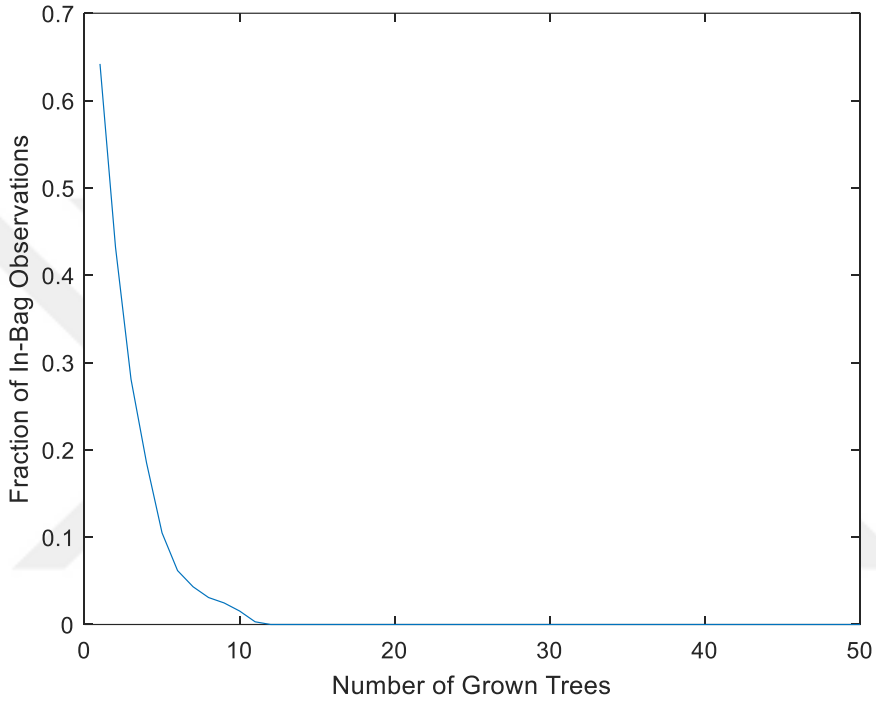


Şekil 4.14. In-Bag gözlemi dışında kalan sınıflandırma hatası.

Bagging yönteminde, tekrar edilen gözlemler uzaklaştırıldıktan sonra sınıflandırma hatasına (out- of - bag error) bakıldığı zaman In-Bag gözlemi dışında kalan sınıflandırma hatası sonuçlarının OOB sınıflandırma hatası sonuçlarından çok da farklı olmadığı görülür. Özellikle 25. ağaçtan sonra sınıflandırma hatasında çok büyük değişiklik olmamıştır. En ciddi değişim ise 10. ağaçta gözlenmiştir (Şekil 4.14).

OOB indisi gözlemleri hangi ağaçlar için out of bag olduğunu anlatan TreeBagger özelliğini gösterir. Bu özelliği kullanmak "in bag" olan tüm eğitim verilerindeki gözlemin fonksiyonunu tüm ağaçlar için kontrol edebilir. Kıvrılma

yaklaşık 2/3'te başlar ki, bu durum bootstrap tekrarı tarafından seçilen yegane fraksiyondur ve yaklaşık 10. ağaçtan sonra 0'a iner. Bagging yönteminde tekrar eden gözlemler incelendiğinde her ne kadar 25. ağaca kadar sınıflandırma hatası düşmeye başlamış olsa da, hatada ciddi şekilde düşme 11. ağaca kadar devam etmiş ve 11. ağaçtan sonra ise dalgalanma yüksek olmamıştır (Şekil 4.15).

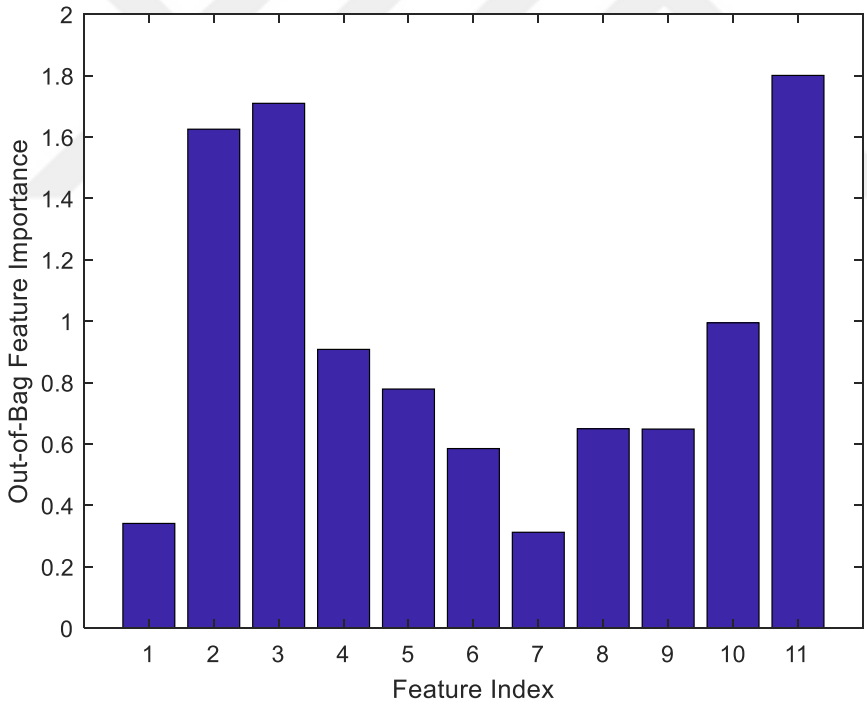


Şekil 4.15. In-Bag gözlemi içinde sınıflandırma hatası.

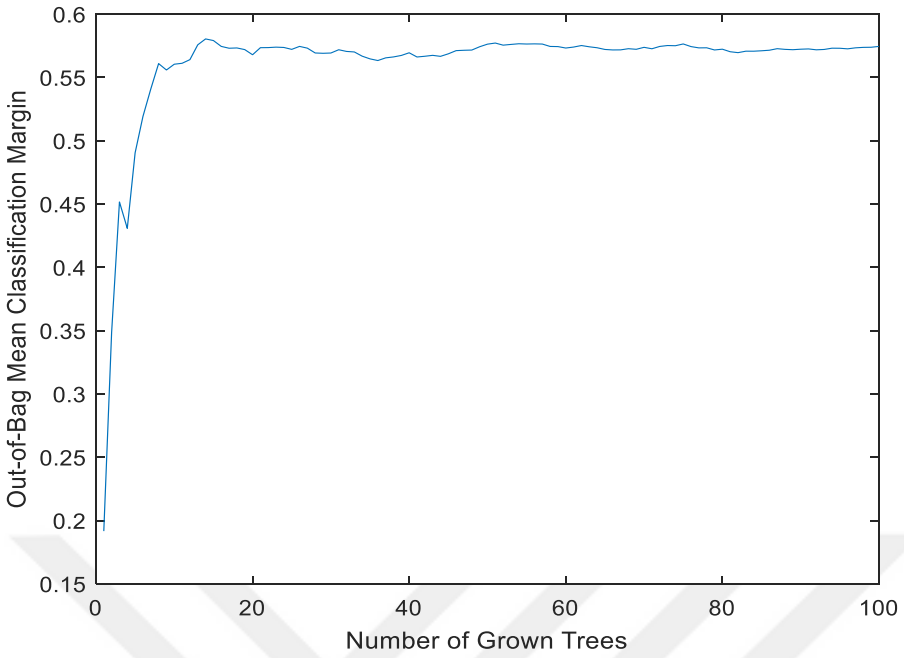
Tahmin yeteneği önemsiz özelliklerden ziyade, önemli özelliklere bağlıdır. Her bir özellik için veri seti incelendiğinde, sınıflandırmada hangi özelliklerin daha fazla etkili olduğunu gösterir. MATLAB uygulamasında *OOBPermutedVarDeltaError* komutu ile tüm ağaçların üzerinde hata kareler ortalamasında meydana gelen ortalama artışı depolar ve bu değeri her bir değişken için ağaçların üzerinden alınan standart sapmaya böler. Böylece değişkenlerin sınıflandırmaya olan katkısını belirler. Daha büyük olan bu değer, daha önemli olan değişken anlamı taşır. Daha sonra keyfi bir kesim noktası (örneğin 0.6 gibi) bir değer, eşik olarak belirlenir ve bu eşik değerin üstünde olanlar, en önemli değişken olarak belirlenir. En güçlü özellikleri kullanmak, tahmin gücünü arttırmak bagging algoritması için önemli bir strateji olmaktadır. Burada

1'den 11'e kadar kodlanmış olan özellikler şu şekilde ifade edilmiştir. 1. özellik; okul sayısı, 2. özellik; öğretmen sayısı,3. özellik; öğrenci sayısı, 4. özellik; YGS'ye giren öğrenci sayısı, 5. özellik; okullaşma oranı, 6. özellik; işsizlik oranı, 7. özellik; istihdam oranı, 8. özellik; okuma yazma bilmeyen sayısı, 9. özellik; üniversite mezunu sayısı, 10. özellik; YGS'de 180 ve üstü puan alarak yerleşen öğrenci sayısı ve 11. özellik; 180 ve üstü puan alan öğrenci sayısı (Şekil 4.16).

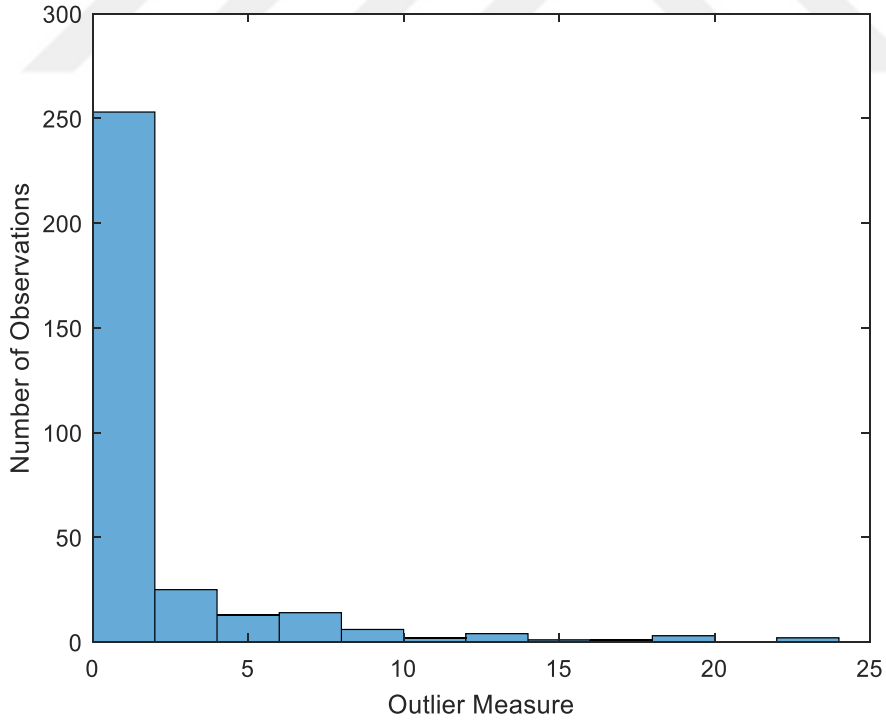
Sınıflandırmada 11. özellik olarak nitelendirilen 180 ve üstü puan alan öğrenci sayısı diğer özelliklerden daha etkili olmuştur. Bu özellikten sonra sırasıyla 3. özellik olan öğrenci sayısı ve 2. özellik olan öğretmen sayısı sınıflandırmada etkili olan özelliklerdir. Sınıflandırmada en az etkili olan özellik ise 7. özellik olarak ifade edilen istihdam oranı ve sonrasında da1. özellik olarak ifade edilen okul sayısı değişkenidir (Şekil 4.16).



Şekil 4.16. Sınıflandırmada özelliklerin etkisi.



Şekil 4.17. Ağaç sayısı arttıkça ortalama sınıflandırma hatası marjini.

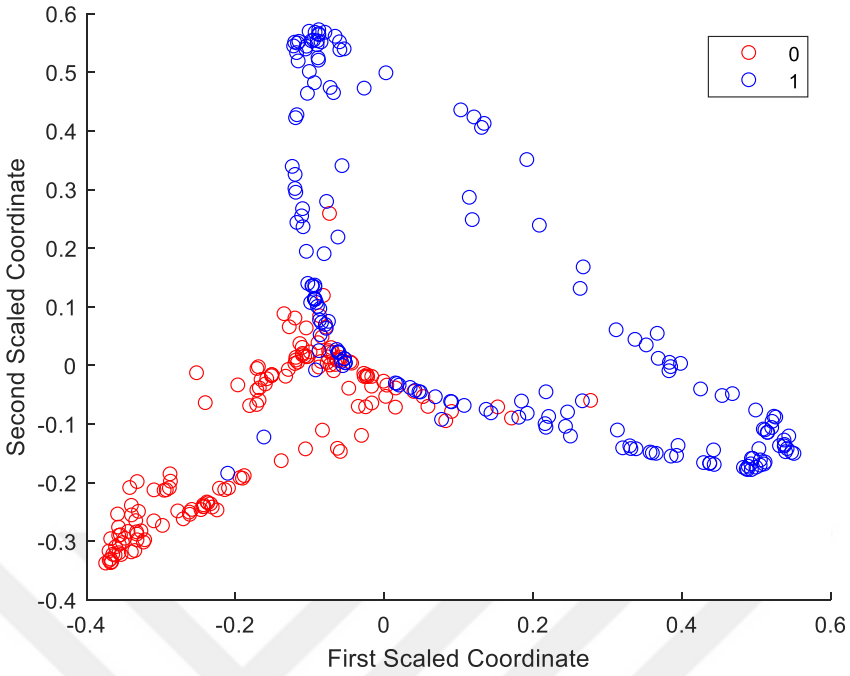


Şekil 4.18. Gözlem sayıları ile aykırı değerler arasındaki ilişki.

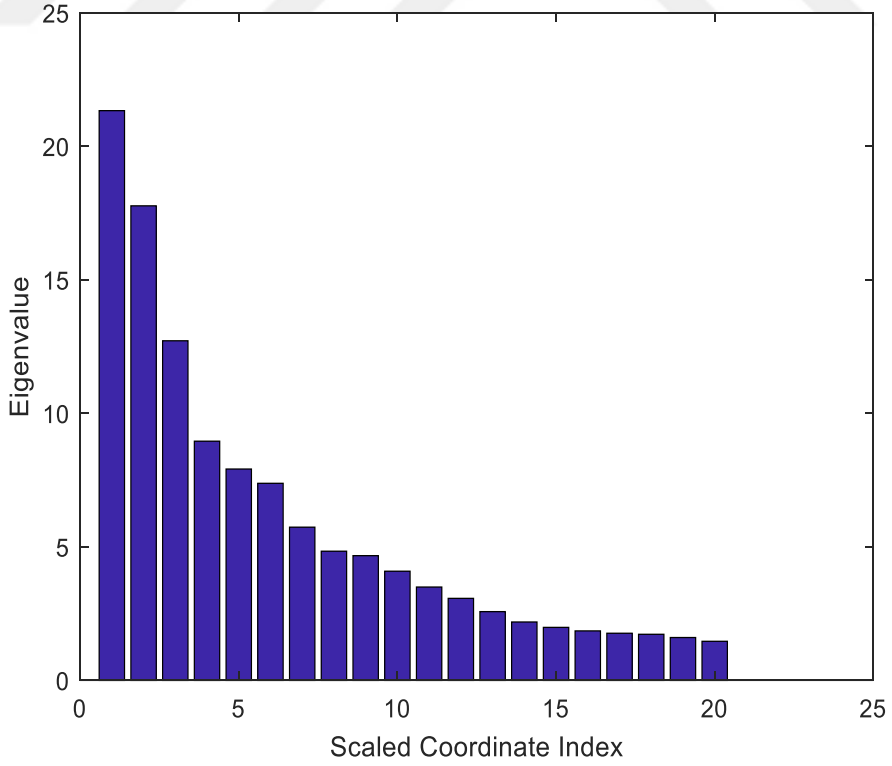
Ağaç sayısı arttıkça, eğrinin altında kalan alan % 56 -% 57 oranında 21. ağaçta ulaşıp, sonrasında da devam etmiştir. Eğri altında kalan büyüklük ağaç sayısı olarak bilgi verir (Şekil 4.17). Eğri altında kalan alan büyüdükçe sınıflandırmadaki belirsizlikte azalır. Bu durum dikkate alındığında 21. ağaç optimal ağaç sayısı olarak düşünülebirlir.

Bagging algoritması aykırı değerlere karşı boosting algoritmasına göre daha az duyarlıdır. Gözlem sayıları ile aykırı değerler arasındaki ilişki incelendiğinde, öğrenim veri setinde yer alan değerlerin büyük kısmının sıfır (0), yani aykırı değere olarak tanımlandığı, sadece bir kaç aykırı değerin olduğu, bunlarında bu değerlerin genel olarak sonucu etkilemediği görülmüştür (Şekil 4.18).

Ölçeklenmiş grafikte de yine aykırı değerler dikkat çekmektedir. Sınıflandırma yapılırken; kırmızı renkli pullar 0, mavi renkli pullar 1 olarak kodlanarak sınıflandırma yapılmıştır. Grafik incelendiğinde sınıflandırmanın doğru yapılmış olduğu görülmektedir. Belirsizliğin önemsiz duruma eriştiği bir sınıflandırmada kırmızı ve mavi pulların iki ayrı grup olarak şekilde kümelenmesi beklenir. Dahası bu iki renk arasında bir marj alanının yer alması beklenir. "0" yüksek oranda doğruluk taşımaktadır. Çünkü koordinat ekseninde ait olduğu bölgede yığılma ve kümelenme söz konusudur. "1" biraz dağınık görünse de ait olduğu bölgede kümelenmiştir. "0"ların bir kısmının grafik üzerinde "1" bölgesinde yer almaktadır. Bu da kısmen de olsa bir entropinin varlığına işaret etmektedir. Yapılan *sensitivity* analizinde bagging algoritmasının sınıflandırma performansının % 85.4 olduğu bulunmuştur. "0" ve "1"lerin bazı yerlerde iç içe geçmiş olması ve kendi kümesinin dışında olması sınıflandırmanın % 100'lük bir doğruluk olmadığını göstermektedir. Bu durum da aykırı değerlerin varlığına işaret etmektedir (Şekil 4.19).

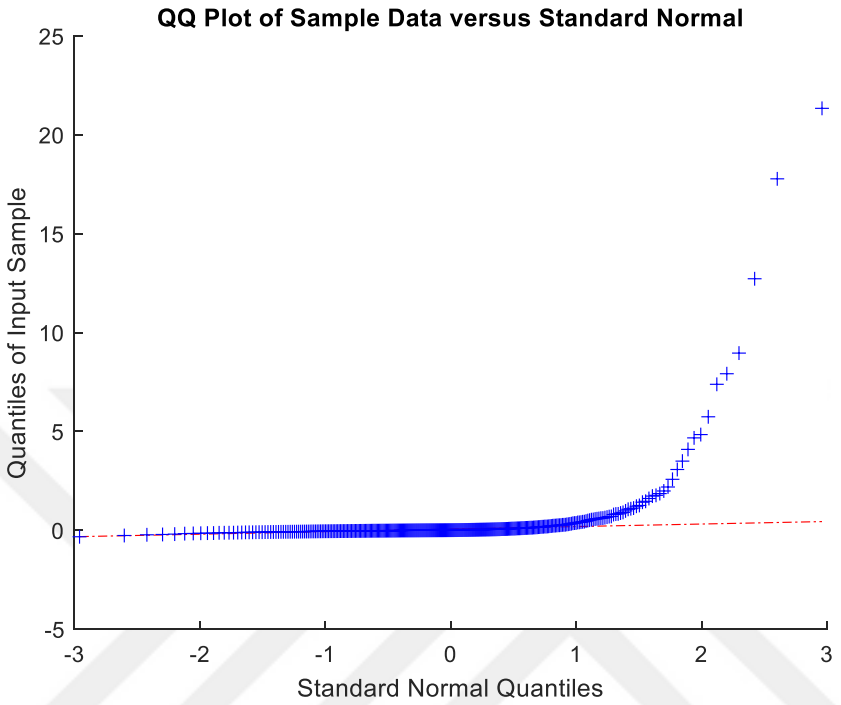


Şekil 4.19. Birinci ve ikinci koordinatlara göre ölçeklendirilmiş sonuçlar.



Şekil 4.20. Öz değerler ve ölçeklendirilmiş koordinat göstergesi.

Özdeğerler ve ölçeklendirilmiş koordinat göstergesi ve ağaç sayısının özdeğerleri dikkate alındığında, ağaç sayısının değişimi gözlenmektedir (Şekil 4.20).

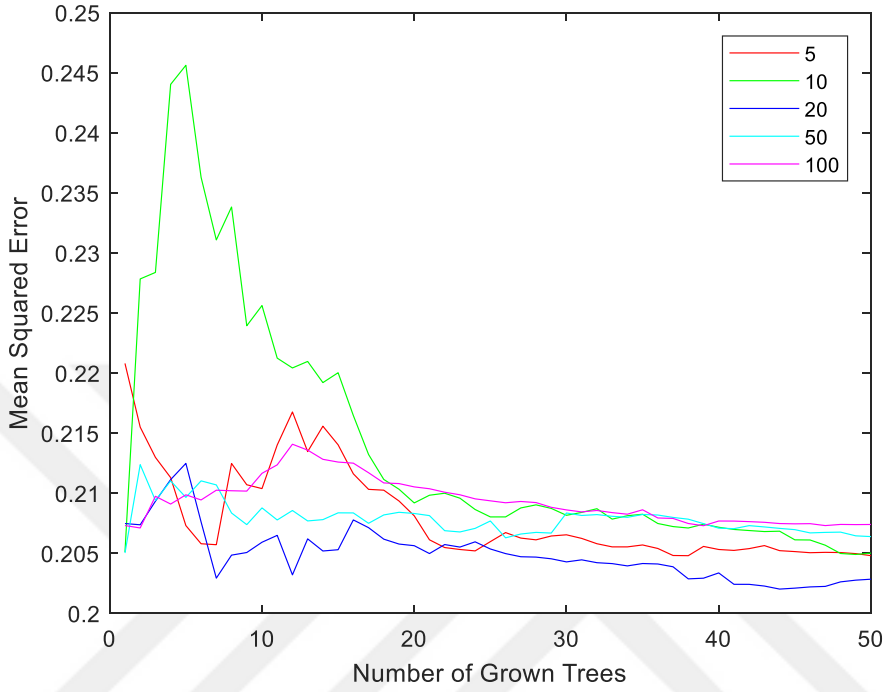


Şekil 4.21. Hataların dağılımı grafiği.

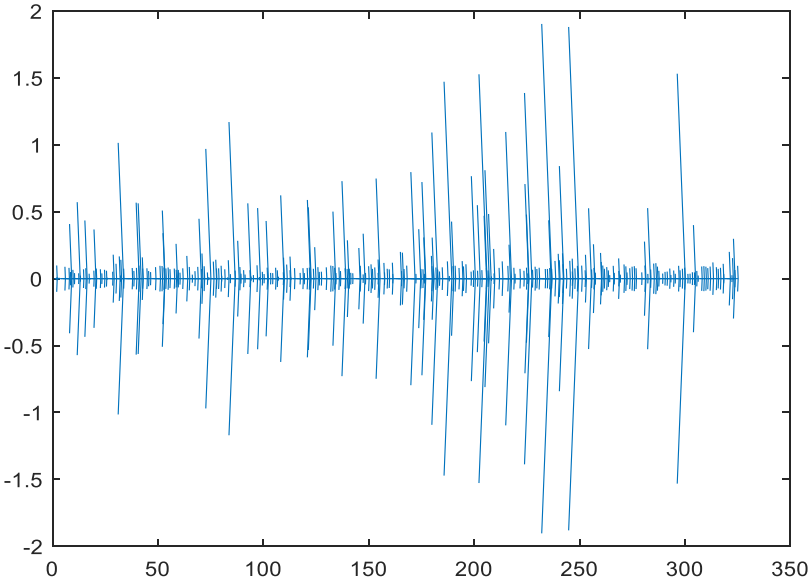
Hataların dağılım grafiği incelendiğinde hataların Gaussian dağılışını net yansıtması için bu değerlerin "0" ekseninde -3 ile +3 arasında değişmesi beklenir. Gözlenen sapmanın hata değerlerinin kısmen sağa çarpık olduğu görülmektedir (Şekil 4.21). Grafiğin doğrusal şekilde devam etmesi gerekirken 0 ile 2 arasında beklenilenin aksi yönünde bir yol izlemesi, aykırı değerlerin varlığından kaynaklanmaktadır. Fakat bu aykırı değerlerin varlığı, hatanın sıfıra yakın olmasını engellememektedir (Şekil 4.21).

Ayrıca HKO (Hata Kareler Ortalaması)'nın yaprak sayısına göre nasıl değiştiği incelenmiştir (Şekil 4.22). Regresyon için genel olarak yaprak sayısı 5 seçilir ve giriş bilgilerinin üçte biri rastgele seçilir. Bir sonraki adımda çeşitli yaprak sayısının gerilemesiyle elde edilen hatalar, optimal yaprak sayılarını karşılaştırarak doğrular. Kırmızı renkle gösterilen grafik hata kareler ortalamasının en düşük olduğu değerleri ortaya çıkarır. Şekilde farklı renklerle 5 ila 100 yaprak sayısının hata kareler ortalaması

üzerinde etkisi değerlendirilmiştir. Grafik incelendiğinde hata kareler ortalamasının en düşük 5 yaprakta elde edildiği görülür. Yaprak sayısı 5, 10 ve 20'ye kadar çıkmış olsa da, ideal yaprak sayısı 5 ila 20 yaprak arasında olmalıdır (Şekil 4.22).



Şekil 4.22. Yaprak sayısı ile Hata Kareler Ortalaması arasındaki ilişki.

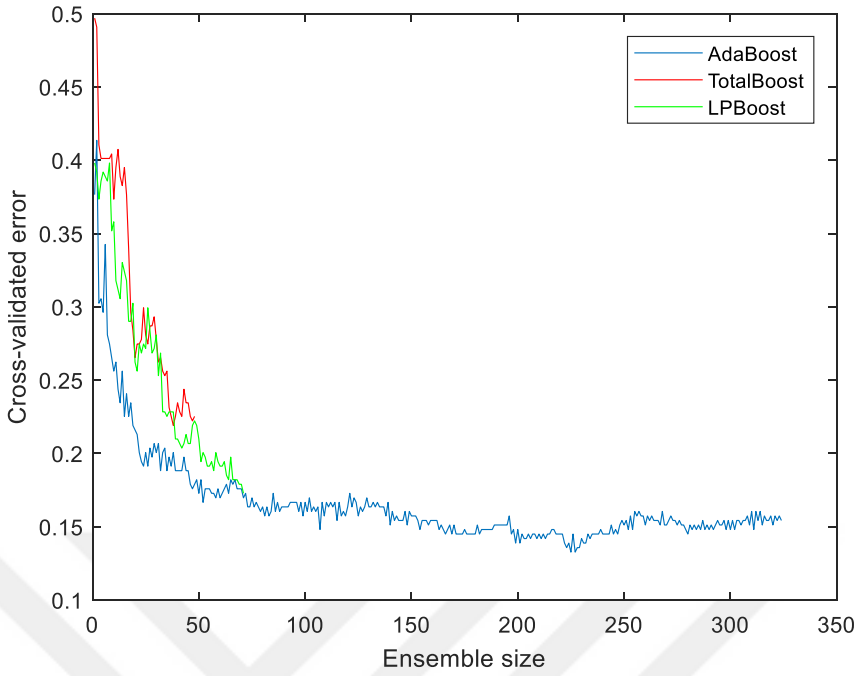


Şekil 4.23. Hata kareler ortalamasının balıksırtı grafiği ile gösterimi.

Bunların dışında HKO'nın balıksırtı grafiği ile gösterimine yer verilmiştir. Balıksırtı grafiğinde elde edilen görüntü hata için verilen diagnostiklerle uyum içerisindedir (Şekil 4.19, Şekil 4.21). Genel olarak toplamda var olan 324 gözlem değerine ait, hata değerlerinin sıfıra yakın olduğu görülür. Fakat bazı verilerde çok büyük hataların olduğu açıktır. Bunlar da daha önce dile getirildiği gibi aykırı değer adayları olarak dikkate alınmalıdır (Şekil 4.23).

4.5. Boosting Yöntemiyle Sınıflandırma

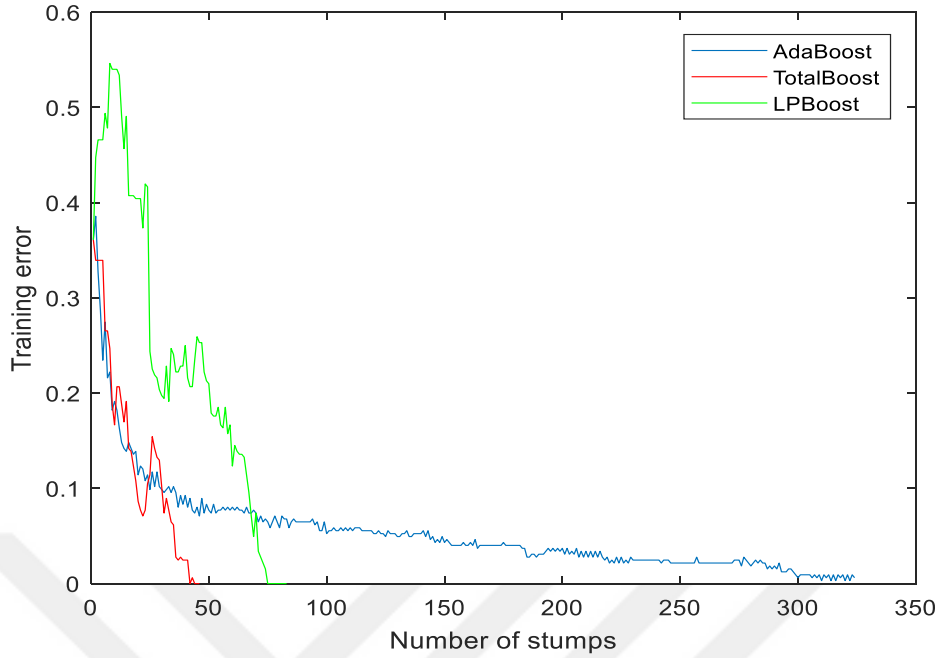
Boosting yöntemiyle sınıflandırma yapılırken, Adaboost, LP Boosting ve Total boosting olmak üzere üç tane boosting yöntemi kullanılıp hangisinin daha ideal sonuç verdiği incelenmiştir. Boosting yöntemleriyle yapılan sınıflandırma sonucu elde edilen grafikler verilmiştir (Şekil4.24, Şekil 4.25, Şekil 4.26). Üç grafikte de ortak olarak ensemble sayısı dikkate alındığında, 50'den sonra bir farklılık olduğu görülür. Test kısmındaki (testing) çapraz geçerlilik hataları (cross validation error) ile ensemble miktarının değişimi görülmektedir (Şekil 4.24). Çapraz geçerlilik hata değeri en küçük olan algoritma boosting algoritmaları arasında en iyi olanıdır. Çapraz geçerlilik hata değerinin belirlenmesinde değişik kayıp fonksiyonları kullanılır ve kayıp fonksiyonuna bağlı olarak aynı verilerde farklı sonuçlara ulaşılabilir. En yaygın kullanılan kayıp fonksiyonu J(.) hata kareler ortalaması olup gerek değer ile tahmin edilen değer arasındaki farktan hesaplanır. Regresyon ve sınıflandırma esaslı boosting algoritmalarında test için kullanılan veri setindeki çapraz geçerlilik hataların küçük olması arzu edilir.



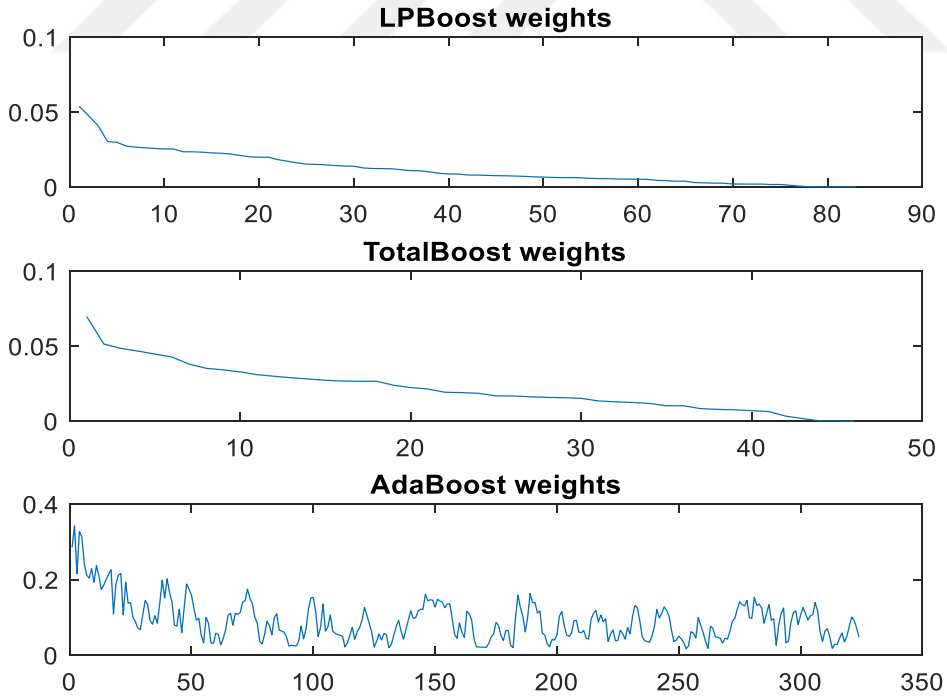
Şekil 4.24. Ensemble büyüklüğü ile çapraz geçerlilik hatası miktarının arasındaki ilişki.

Eğitim verileri ile ensemble büyüklüğü arasındaki hatalar üç ayrı boosting yöntemiyle incelendiğinde, hem test kısmındaki verilerde, hem de eğitim verilerinde, AdaBoost'un daha iyi sonuç verdiği görülmüştür. Adaboost algoritmasının ürettiği kayıp fonksiyon değerleri incelendiğinde, zaman zaman bu değerlerin 0.15'in altına düştüğü ve çoğunlukla 0.2 ile 0.15 aralığında değiştiği görülmektedir. Adaboost eğitim veri setinde bazen kayıp fonksiyonun daha fazla olduğu gözlenebilir. Bu durumu aşırı uyum olmadığı sürece normal karşılamak gerekir. Total Boost ve LP Boost yöntemlerinde aşırı uyum başlamıştır, fakat AdaBoost yönteminde aşırı uyumu kontrol altına alan parametrenin varlığından dolayı aşırı uyum tehlikesine rastlanmamıştır (Şekil 4.25).

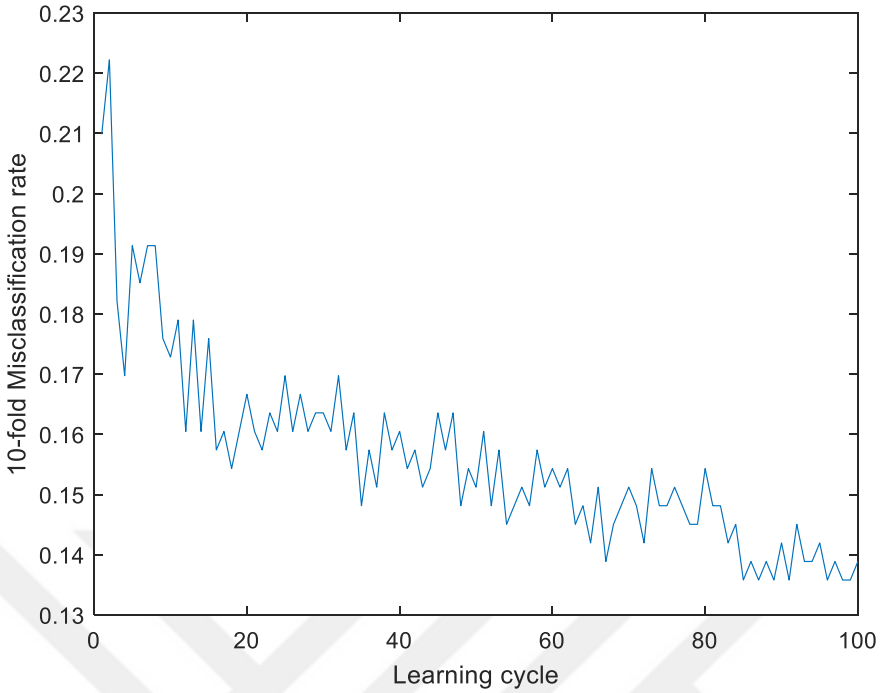
Parametrelerin boostingde kullanılan üç farklı algoritmaya göre dağılımı incelendiğinde ise, yineleme sayısı esas alındığında AdaBoost yönteminin, diğer yöntemlere göre daha iyi olduğu görülür (Şekil 4.26).



Şekil 4.25. Kök sayısı ve eğitim hatası arasındaki ilişki.



Şekil 4.26. Parametrelerin algoritmalara göre dağılımı.



Şekil 4.27. 10 segmentte yanlış sınıflandırma oranı ile öğrenme döngüsü.

Biri test geri kalan dokuzu eğitim verisi olmak üzere, 10 segmente ayrılarak yanlış sınıflandırma oranı 100 iterasyondan oluşan öğrenme döngüsü incelendiğinde sınıflandırmada çapraz geçerlilik hata değeri (cross validation) hesaplanırken, veriler 10 segmente bölünür. Bu segmentlerden dokuzu eğitim verisi olarak belirlenirken, biri test verisi olarak kullanılır. Daha sonra bir diğer segment test verisi olarak kullanılır ve geri kalan dokuz segment eğitim verisi olarak hesaplanır. Bu işlem tüm segmentler test verisi olarak hesaplanana kadar devam eder. Bu şekilde on tane çapraz hatanın ortalaması hesaplanarak sonuçlar bulunur. 100 iterasyon sonunda hatanın düşüşü görülmektedir. 100. iterasyonda hata sıfıra yaklaştığı için işlem burada kesilmiştir (Şekil 4.27).

4.6. Bagging ve Boosting Sonuçlarının Karşılaştırılması

Çalışmada önerilen yöntemlerin performansını göstermek için doğruluk, kesinlik, duyarlılık ve f-ölçütleri kullanılmıştır. Bu başarı ölçütleri aşağıdaki gibi hesaplanmaktadır.

$$Doğruluk(Accuracy) = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Kesinlik(Precision) = TP/(TP + FP)$$

$$Duyarlılık(Recall) = TP/(TP + FN)$$

$$F - \text{Ölçütü}(F - Measure) = 2(Duyarlılık * Kesinlik)/(Duyarlılık + Kesinlik)$$

Bu denklemlerde T, F, P ve N; sırasıyla doğruyu, yanlışı, pozitifi ve negatifi ifade etmektedir. Örneğin, TP doğru sınıflandırılan pozitif örnek sayısını; FN ise yanlış sınıflandırılan negatif örnek sayısını göstermektedir.

Doğruluk: Başarının tespiti için kullanılan en popüler ve basit yöntemdir ve bu oran doğru sınıflandırılmış (TP+TN) örnek sayısının, toplam örnek sayısına (TP+TN+FP+FN) oranı olarak tanımlanmaktadır.

Kesinlik: Sınıflandırıcı sonucunun kesinlik derecesini verir. Pozitif olarak etkilenen örneklerin sayısının (TP) pozitif olarak sınıflandırılmış toplam örneklere (TP+FP) oranıdır.

Duyarlılık: Pozitif olarak etiketlenmiş örneklerin (TP) gerçekten pozitif olan örneklerin (TP+FN) toplam sayısına oranıdır.

F-Ölçütü: Kesinlik ve duyarlılık metrikleri kullanılarak hesaplanmaktadır. Sistemin, kesinlik veya duyarlılık yönüne doğru optimize edilmesinde kullanılmaktadır.

Bu sonuçlar sınıflandırma işlemleri 10- katlı çapraz geçerlilik (cross validation) testine göre gerçekleştirilmiştir. Veri seti 10 parçaya bölünür, sırası ile bu 10 parçanın her biri test seti, diğerleri eğitim seti olarak kullanılarak sınıflandırma işlemi gerçekleştirilir. İşlem sonunda 10 sınıflandırma işleminin sonuçları genel başarı olarak alınır.

Çizelge 4.1. Boosting AdaBoost algoritmasından elde edilen karışıklık (Confusion) Matrisi.

Output Class		*Ortalamanın altı (0)*	*Ortalamanın üstü (1)*	*%*
	Ortalamanın altı (0)	122	27	% 81.88
	Ortalamanın üstü (1)	23	152	% 86.85
		% 84.13	% 84.91	% 84.56

Çizelge 4.2. Bagging algoritmasından elde edilen karışıklık (Confusion) matrisi.

Output Class		*Ortalamanın altı (0)*	*Ortalamanın üstü (1)*	*%*
	Ortalamanın altı (0)	122	27	% 81.88
	Ortalamanın üstü (1)	26	149	% 85.14
		% 82.43	% 84.65	% 83.64

Çizelge 4.3. Boosting AdaBoost algoritmasından elde edilen gerçek pozitif (TP), yanlış pozitif (FP), doğruluk, kesinlik, duyarlılık ve f-ölçütleri.

Sınıf	*TP oranı*	*FPoranı*	*Kesinlik*	*Duyarlılık*	*F-Ölçütü*
Ortalamanın altı (0)	0.819	0.131	0.841	0.819	0.830
Ortalamanın üstü (1)	0.869	0.181	0.849	0.869	0.859
Ortalama	0.846	0.158	0.846	0.846	0.846

Çizelge 4.4. Bagging algoritmasından elde edilen Gerçek pozitif (TP), Yanlış pozitif (FP), doğruluk, kesinlik, duyarlılık ve f-ölçütleri.

Sınıf	*TP oranı*	*FPoranı*	*Kesinlik*	*Duyarlılık*	*F-Ölçütü*
Ortalamanın altı (0)	0.819	0.149	0.824	0.819	0.822
Ortalamanın üstü (1)	0.851	0.181	0.847	0.851	0.849
Ortalama	0.836	0.166	0.836	0.836	0.836

Yukarıda verilen sonuçlar göstermiştir ki, bütün performans ölçütlerinde marjinal olarak Boosting yöntemi Bagging yönteminden daha iyi sonuçlar üretmiştir. Örneğin ortalama olarak Bagging sınıflandırma metodu ile % 83.6 başarı oranı gözlenmiştir.

Bagging ve Boosting karşılaştırmaları yapılırken Bagging için 1000 Bootstrap alarak yapılmıştır. Ancak, veri setinin büyük olmadığı bu tür uygulamalarda Bootstrap sayısı arttırılarak daha doğru bir karşılaştırma yapmak gerekmektedir.

Ayrıca Boosting yöntemi ile yapılan sınıflandırmada ortalamanın altında ve üstünde olan değerlere göre iki grup oluşturulduğunda elde edilen sınıflandırmanın kümeleme analizi sonuçlarını destekler nitelikte olduğu görülmekedir. Örneğin ilk gözlem değeri % 88 olasılıkla ortalamanın üstü gruba dahil edilirken, % 11 olasılıkla da ortalamanın altında kalan grubu temsil eder. Bu durum da Boosting yönteminin, kümeleme analizini doğruladığını gösterir.

5. TARTIŞMA VE SONUÇ

Bu çalışmada 2010-2013 yılları arasında Yükseköğretime Geçiş Sınavına girerek başarı gösteren, 180 ve üstü puan alan öğrencilerden üniversiteye yerleşen öğrencilerin Bagging ve Boosting yöntemleri ile sınıflandırılması ele alınmıştır. Veri seti oluşturulurken 81 il baz alınmış ve dört yıl için kümeleme analizleri ayrı ayrı yapılarak değerlendirilmiştir. Kümeleme analizi için R bilgisayar programı, Bagging ve Boosting yöntemleriyle sınıflandırma yapmak için ise MATLAB bilgisayar programı kullanılmıştır.

Elde edilen sonuçlar dikkate alındığında AdaBoost yönteminin karar ağacı diğer ensemble yöntemlere göre daha başarılı bir yöntem olduğu anlaşılmıştır. AdaBoost yöntemi veri setinden rastgele seçim yaparak yeni sınıflandırmalar yapmak yerine hangi örnekle hatalı sınıflandırma işlemlerinin yapıldığını belirlemektedir. Bu nedenle her bir iterasyonda hatayı düşürdüğü ve aşırı uyumdan diğer yöntemlere göre daha az etkilendiği için daha yüksek başarı sunmuştur. Yapılan çalışmalar, dengesiz veri gürültülü olduğunda Bagging yönteminin, Boosting yöntemine göre net olarak tercih edildiğini gösterir. Bunun sebebi ise Boosting yönteminin performansını düşürerek gürültü örneklerine daha fazla odaklanmasıdır. İleri gürültü örneklerinin ve dengesiz verilerin olduğu durumlarda Bagging Yöntemi Boosting yöntemine göre daha iyi bir performans sergiler (Khoshgoftaar ve ark., 2011).

Bu sonuçlar Bagging algoritması uygulanma aşamasında parametrik olmayan Bootstrap ile örnekleme yapıldığında olağan bir sonuç olduğu bir çok çalışma tarafından onaylanmaktadır (Chen ve ark., 2017). Bagging, Boosting yöntemine göre daha düşük bir performans sergilemiştir. Bunun nedeni Boosting yönteminin hata yapılan örnekler üzerinden kendini yeniden eğitebilmesidir (Kotsiantis, 2014). Ayrıca Işıkhan (2014), Regresyon Ağaçları yönteminin performansını incelemiş ve Bagging ve Boosting yöntemlerinin regresyon ağacının test seti performansı üzerinde daha iyi bir iyileştirici rol üstlendiğini görmüştür. Bagging ve Boosting algoritmalarının performansları birbiriyle kıyaslandığında ise Boosting algoritmasının, ortalama ve standart hata yönünden Bagging algoritmasına göre daha iyi tahmin yaptığı belirlenmiştir.

Ağaç sayısının artması algoritmayı daha kararlı hale getirir (Breiman 2001). Ağaç sayısı arttıkça doğru sınıflandırma yapmada algoritma performansı daha iyi bir sonuca yakınsadığı görülse de algoritma daha yavaş çalışacak ve işlem yükü artacaktır. Bu nedenle ağaç sayısının doğru belirlenmesi çok önemlidir (Arsov ve ark., 2017). Bu çalışmada da hatanın en düşük değeri 32. ağaç da gözlemlense de 25. ağaç itibariyle sınıflandırma hatasında büyük bir değişiklik olmamasından dolayı işlem yükünü hafifletmek adına 25 ağaç kullanmak yeterli olacaktır. Ayrıca hatanın belirgin olarak en çok düştüğü dirsek olarak nitelendirebileceğimiz durum 10. ağaçta olmaktadır.

Bagging algoritması ile elde edilen sonuçlar göstermiştir ki; az sayıda aykırı değerin olması durumunda söz konusu aykırı değerler sonuç üzerinde çok fazla etkili olmamaktadır (Bkz. Şekil 4.18). Veriler ölçeklendirildiğinde de benzer sonuçlar dikkati çekmektedir. Bir çok çalışma bu bulgulara benzer sonucu rapor etmiştir (Davidson ve Pan, 2016; Chen ve ark., 2017; Arsov ve ark., 2017). Zira Boosting algoritmalarını kullanan yöntemler, aykırı değerlerin bulunduğu değişkenler ve kovaryetlere göre düzeltilmiş değişkenler üzerinde zayıf sınıflandırma yapma performansı göstermektedir. Buna karşılık Bagging algoritması örneklemden kaynaklanan sapmadan dolayı da zayıf sınıflandırma performansı göstermektedir (Davidson ve Pan. 2016). Schapire ve Freund (2012), yaptıkları bir çalışmada benzer sonuçlarla sınıflandırmalar arasındaki marjin genişliği dikkate alarak elde etmişlerdir. Bu çalışmada elde edilen sonuçlara göre aykırı değer bulunduran değişkenler için daha dar bir marjin genişliği ile sınıflandırma yapıldığı sonucuna varılmıştır.

Bagging yönteminde öğrenim veri seti için Bootstrap ile örnekleme yapıldığında orijinal veri setinde bulunan bazı gözlem değerleri (X_i,Y_i) öğrenim setine dahil olmayabilir. Bu duruma OOB (Out of Bag) ismi verilmektedir. OOB sorunu parametrik olmayan Bootstrap örneklemesinden kaynaklanır. Mevcut çalışmada Bagging yönteminde tekrar eden gözlemler incelendiğinde her ne kadar 25. Ağaca kadar sınıflandırma hatası düşmeye başlamış olsa da hata da ciddi şekilde düşme 11. Ağaca kadar devam etmiş ve 11. Ağaçtan sonra ise dalgalanma yüksek olmamıştır (Bkz. Şekil 4.15). Benzer sonuç Davidson ve Pan (2016), tarafından yapılan çalışmada da rapor edilmiştir. Yapılan çalışmada parametrik olmayan Bootstrap yerine parametrik Bootstrap yöntemi ile örnekleme yapıldığında örneklemden kaynaklanan sapmanın azaldığı belirtilmiştir.

Sonuçlar incelendiğinde ağaç sayısı ve etkileşim derecesi arttıkça hata kareler ortalaması değerleri gittikçe azalmaktadır. Aşırı uyum olsa da olmasa da ağaç sayısı arttıkça hata kareler ortalaması değerlerinde azalma görülmektedir.

Ensemble yöntemleri uygulama kolaylığı ve tahminleme performansından dolayı gün geçtikçe daha fazla uygulama alanı bulmaktadır. Bagging, paralel ensemble algoritmalarına sahip olup, varyansı küçültme özelliği taşımaktadır. Buna karşılık Boosting, ardışık ensemble algoritmalarına sahip olup sapmayı küçültme özelliğine sahiptir. Arsov ve ark., (2017), Bagging ve Boosting algoritmalarının (paralel ve ardışık yaklaşımlar) ortak özelliğini taşıyan yeni bir algoritma denemişve Bagging ile Boosting sorunlarının ortadan kalktığını ve yakınsama haızının arttığını ortaya koymuşlardır.

Sonuç olarak; kendi çalışmamız ve diğer çalışmalar incelendiğinde çeşitli veri setlerindeki deney sonuçları göstermiştir ki; çeşitli temel öğrenicilerdeki aykırı değerlere karşı Bagging algoritmaları daha etkili, tahminleme sapmasına karşı ise Boosting algoritmaları daha etkilidir. Bu avantajlarına rağmen ensemble yöntemlerinin bazı zayıf yönleri de bulunmaktadır. Örneğin Bagging ve Boosting sonuçlarının yorumlanması halen sorun olarak karşımızda durmaktadır. Bununla birlikte çok büyük veri setlerinde yakınsama sorunu yaşanmaktadır. Ayrıca büyük veri setlerinde Boosting algoritması ile aşırı uyum problemi de söz konusu olmaktadır.

KAYNAKLAR

Akgöbek, Ö., Çakır, F., 2009. Veri Madenciliğinde Bir Uzman Sistem Tasarımı. ***Akademik Bilişim'09 - XI. Akademik Bilişim Konferansı Bildirileri.*** 11-13 Şubat 2009, Şanlıurfa. 801-806.

Albayrak, A., Yılmaz, Ş., 2009. Veri Madenciliği: Karar Aağacı Algoritmaları ve İMKB Verileri Üzerine bir Uygulama. ***Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*** **14***(*1) : 31-52.

Alfaro, E., Gamez, M., Garcia, N., 2013. Adabag: An R package for classification with Boosting and Bagging. ***Journal of Statistical Software,*** **54** (2): 1-35.

Alkoot, F. M., Kittler, J., 1999. Experimental evaluation of expert fusion strategies. ***Pattern Recognition Letters,*** **20** (11): 1361–1369.

Alpaydin, E., Jordan, M. I., 1996. Local linear perceptrons for classification**.** ***IEEE Transactions on Neural Networks,*** **7**(3): 788-792.

Amasyalı, M. F., Ersoy, O., 2011. Sınıflandırıcı Topluluklarında Başarı Tabanlı Budama ve Ağırlıklı Oylama. ***IEEE 19. Sinyal İşleme ve İletişim Uygulamaları Kurultayı.*** 20-22 Nisan 2011, Muğla. 194-197.

Anonim, 2013. Öğrenci Seçme ve Yerleştirme Sistemi Yükseköğretim Programları ve Kontenjanları Kılavuzu. ***http://www.osym.gov.tr.***

Arsov, N., Pavlovski, M., Basnarkov, L., Kocarev, L., 2017. Generating highly accurate prediction hypotheses through collaboratative ensemble learning. ***Scientific Reports,*** **7**(44649): 1-34.

Augusty, S. M., Izudheen, S., 2013. Ensemble Classifiers A Survey: Evaluation of Ensemble classifiers and data level methods to deal with imbalanced data problem in protein- protein interactions. ***Review of Bionformatics and Biometrics,*** **2**(1): 1-9.

Babu, M. S., Reddy, S., Ramana, B.V., Murty, R. N. V., 2013. A Web- Based soya bean expert system using Baggng algorithm with C4.5 Decission Trees. ***International Journal of Agriculture Innovations and Research,*** **1** (4): 91-96.

Banfield, R. E., Hall, L. O., Bowyer, K. W., Kegelmeyer, W. P., 2005. Ensemble diversity measures and their application to thinning**.** ***Information Fusion,*** **6** (1): 49–62.

Bauer, E., Kohavi, R., 1998. An Emperical comparison of voting classification algorithms: Bagging, Boosting and variants. ***Machine Learning,*** 1-38.

Benediktsson, J. A., Swain, P.H., 1992. Consensus theoretic classification methods. ***IEEE Transactions on Systems, Man and Cybernetics,*** **22**(4): 688–704.

Bifet, A., Holmes, G., Pfahringer, B., Frank., E., 2010. MOA: Massive online analysis. ***In: Journal of Machine Learning Resarch,*** **11** : 1601-1604.

Biggio, B., Corona, I., Fumera, G., Giacinto, G., Roli, F., 2011. Bagging Classifiers for Fighting Poisoning Attacks in Adversarial Classification Tasks. ***Springer Verlag Berlin Heidelberg,***350-359.

Bishop, C. M., 2006. Pattern Recognation and Machine Learning. ***Springer,*** New York. 758.

Bloch, I., 1996. Information combination operators for data fusion: A comparative review with classification. ***IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems andHumans,*** **26**(1): 52–67.

Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M. K., 1987. Occam's Razor. ***Information Processing Letters,*** **24** : 377-380.

Bounsaythip, C., Esa, R. R., 2001. Overview of data mining for customer behavior modeling. ***VTT Information Technology Research Report,*** **1**: 1-53.

Breiman, L., 1996. Bagging predictors. ***Machine Leraning,*** **24**(2):123-140.

Breiman, L., 1998. Arcing classifiers. ***Annals of Statistics,*** **26**(3): 801–849.

Breiman, L., 1999. Prediction games and arcing algorithms. ***Neural Computation,*** **11**:1493-1517.

Breiman, L., 2001. Random Forests. ***Machine Learning,*** **45**(1): 5-32.

Breiman, L., 2001. Using iterated bagging to debias regressions. ***Machine Learnings,*** **45** (3): 261-277.

Brownlee, J., 2016. Boosting and AdaBoost for Machine Learning. ***https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/*** Machine Learning Algorithms. Erişim tarihi:15.05.2018.

Buja, A., Stuztle W., 2006. Observation on Bagging. ***Statistica Sinica,*****16** :323-351.

Bulut, F., 2016. Sınıflandırıcı toplulukların dengesiz veri kümeleri üzerindeki performans analizleri. ***Bilişim Teknolojileri Dergisi,*** **9**(2):153-159.

Bühlmann, P., Hothorn, T., 2007. Boosting algorithms: Regularization, prediction and model fitting (with Discussion). ***Statistical Science,*****22**: 477-522.

Bylander, T., 2002. Estimating generalization error on two-class data sets using out-of-bag estimates. ***Machine Learning,*** **48** : 287–297.

Coşgun, E., Limdi, N.A., Duarte C.W., 2011. High dimensional pharmacogenetic prediction of a continuous trait using machine learning techniques with application to warfarin dose prediction in African American. ***Bioinformatics,*** **27**(10): 1384-1389.

Chen, Z., Lin, T., Chen, R., Xie Y., Xu, H., 2017. Creating diversity in ensembles using synthetic neighborhoods of training samples. ***Journal Apllied Intelligence,*** **47** (2): 570-583.

Davidson, I., Fan, W., 2006. When Efficient Model Averaging Out- Performs Boosting and Bagging. ***10th European Conference on Principles and Practice of Knowledge Discovery in Databases.*** September, 2006, Berlin, Germany, 477-486.

Dietterich, T., 1995. Overftting and Undercomputing in Machine Learning. ***ACM Computing Surveys (CSUR).27*** (3): 326-327.

Dietterich, T., 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. ***Machine Learning,*** **40**(2):139–157.

Drucker, H., Cortes, C., Jackel, L. D., LeCun, Y., Vapnik, V., 1994. Boosting and other ensemble methods. ***Neural Computation,*** **6**(6): 1289–1301.

Efron, B., Tibshirani, R., 1993. ***An Introduction to the Bootstrap.*** Chapman and Hall. London. 430.

Elith, J., Leathwick, J.R, Hastie, T. 2008. A working guide to boosted regression trees. ***Journal of Animal Ecology,*** **77**(4): 802-813.

Esposito, R., Saitta, L., 2003. Monte Carlo Theory as an Explanation of Bagging and Boosting. ***IJCAI'03 Proceedings of the 18th International Joint Conference on Artifical Intelligence.*** 09-15 Agust, 2003, Acapulco, Mexico, 499-504.

Ferreira, A., Figueiredo, M., 2007. Boosting Algorithms: A Review of Methods, Theory, and Applications, Chap, 2. ***Ensemble Machine Learning Methods and Application.*** (Editörler: C. Zhang, Y. Ma) Springer, New York. 35-85.

Freund, Y., Schapire, R., 1996. Experiments With a New Boosting Algorithm. ***In: Proceedings of the Thirteenth International Conference on Machine Learning Theory.*** MorganKaufmann Publishers Inc. San Francisco, 148-156.

Freund, Y., Schapire, R., 1996. Game theory, on-line prediction and boosting. ***In Proceedings of the Ninth Annual Conference on Computational Learning Theory,*** ACM Press. 325–332.

Freund, Y., Schapire, R., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. ***Journal of Computer and System Sciences,*** **55**(1):119–139.

Friedman, J. H., 2001. Greedy Function Approximation: A Gradient Boosting Machine. ***The Annals of Statistics,*** **29**:1189-1232.

Giacinto, G., Roli, F., 2001. Approach to the automatic design of multiple classifier systems. ***Pattern Recognition Letters,*** **22** (1) : 25–33.

Gollapudi, S., 2016. ***Practical Machine Learning.*** Published by Packt Publishing ltd. Birmingham B3 2PB, UK. 432.

Gökay,E. G., Taşkın, Ç., 2005. Veri madenciliğinde Karar Ağaçları ve bir satış analizi uygulaması. ***Eskişehir Osmangazi Üniversitesi Sosyal Bilimler Dergisi.*** **6**(2): 221-239.

Grove, A.J., Schuurmans, D., 1998. Boosting in the Limit: Maximizing the Margin of Learned Ensembles. ***In: Proceeding of the AAAI-98. John Wiley & Sons Ltd,***692-699.

Grubinger, T., Kobel, C., Pfeiffer, K.P., 2010. Regression tree construction by bootstrap: Model search for DRG-systems applied to Austrian health-data. ***BMC Medical Informatics and Decision Making,*** **10** (9): 1-11.

Hand, D., Manilla, H., Smyth, P., 2001. Principles of Data Mining. ***MIT***, USA, 546

Hastie, T., Tibshirani, R., Friedman, J., 2001. ***The Elements of Statistical Learning.*** Springer, 2nd edition, New York, NY. 737

Hastie, T., Tibshirani, R., Friedman, J., 2009. ***The Elements of Statistical Learning: Data Mining,*** Springer, 2nd edition, New York, NY. 744

Ho, T.K., Hull J.J., Srihari, S.N., 1994. Decision combination in multiple classifier systems. ***IEEE Transactions on Pattern Analysis and Machine Intelligence,*** **16**(1): 66–75.

Ho, T. K., 1998. Random subspace method for constructing decision forests. ***IEEE Transactions on Pattern Analysis and Machine Intelligence,*** **20**(8): 832–844.

Hsu, K. W., 2013. Weight-Adjusted Bagging of classification algorithms sensitive to mising values. ***International Journal of Information and Education Technology,*** **3** (5) : 560-566.

Işıkhan, S., 2014. ***Mikrodizilim Gen İfade Çalışmalarında Genelleştirme Yöntemlerinin Regresyon Modelleri Üzerine Etkisi*** (doktora tezi). Hacettepe Üniversitesi, Ankara.

James, G., Witten, D., Hastie, T., Tibshirani, R., 2017. An Introduction to Statistical Learning: with Applications in R. ***Springer Text in Statistics. 7th printing edition.*** 321.

Jacobs, R. A., Jordan, M. I., Nowlan S. J., Hinton, G. E., 1991. Adaptive mixtures of local experts. ***Neural Computation,*** **3**(1):79–87.

Jordan, M. J., Jacobs, R. A., 1994. Hierarchical mixtures of experts and the EMalgorithm. ***Neural Computation,*** **6** (2): 181–214.

Kavzoğlu, T., Çölkesen, İ., 2010. Karar Ağaçları İle uydu görüntülerinin sınıflandırılması: Kocaeli örneği. ***Harita Teknolojileri Elektronik Dergisi,*** **2** (1): 36-45.

Khoshgftaar, T. M., Hulse, J. V., Napolitano, A., 2011. Comparing Boosting and Bagging Techniques with Noisy and Imbalanced Data. ***IEEE Transactions on Systems Man and Cybernetics,*** **41** (3): 552-568.

Kılınç, D., Borandağ, E., Yücalar, F., Özçift, A., Bozyiğit, F., 2015. Yazılım Hata Kestiriminde Kollektif Sınıflandırma Modellerinin Etkisi. ***Ulusal Yazılım Mühendisliği Sempozyumu.*** Eylül 2015, İzmir. 113-121.

Kittler, J., Hatef, M., Duin, R. P. W., Mates, J., 1998. On combining classifiers. ***IEEE Transactions on Pattern Analysis and Machine Intelligence,*** **20**(3): 226–239.

Kotsiantis, S. B., 2014. Bagging and Boosting variants for handling classification problems: asurvey. ***Cambridge University Press.*** **29** (1): 78-100.

Koyuncugil, A. S., Özgülbaş, N., 2008. İMKB'de İşlem Gören KOBİ'lerin güçlü ve zayıf Yönleri : Bir CHAID Karar Ağacı uygulaması. ***Dokuz Eylül Üniversitesi İİBF Dergisi,*** **23**(1): 1-22.

Kumari, G. T., 2012. A Study of Bagging and Boosting approaches to develop meta-classifier. ***Engineering Science and Technology: An International Journal (ESTIJ),*** **2**(5): 850-855.

Kuncheva, L. I., Bezdek, J. W., Duin, R.P.W., 2001. Decision templates for multiple classifier fusion: an experimental comparison. ***Pattern Recognition,*** **34** (2): 299–314

Kuncheva, L. I., 2002. Switching between selection and fusion in combining classifiers: An experiment.***IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,*** **32**(2): 146–156.

Kuncheva, L. I., 2002. A theoretical study on six classifier fusion strategies. ***IEEE Transactions on Pattern Analysis and Machine Intelligence*** **24** (2): 281–286.

Kuncheva, L. I., 2004. Classifier ensembles for changing environments. ***5th International Workshop on Multiple Classifier Systems in Lecture Notes in Computer Science,***(Editörler: F. Roli, J. Kittler, T. Windeatt). Cagliari, Italy, 3077:1-15.

Lee, S. L.A., Kouzani, A. Z., Hu, E. J., 2010. Random forest based lungnodule classification aided bu clustering. ***Computerized Medical Imagingand Graphics,*** 34: 535-542.

Lu, Y., 1996. Knowledge integration in a multiple classifier system. ***Applied Intelligence, 6***(2): 75–86.

Machova, K., Barcak, F., Bednar, P., 2006. A Bagging method using Decision Trees in the role of base classifiers. ***Acta Polytechnica Hungarica,*** **3** (2): 121-132.

Maclin, R., Opitz, D., 1997. An empirical evaluation of bagging and boosting. ***In Proceedings of the Fourteenth National Conference on Artificial Intelligence,*** AAAI Press. 546–551.

Mayr, A., Binder, H., Gefeller, O., Schid, M., 2014. The evolution of Boosting algorithms from Machine Learning to statistical modelling. ***Methods of Information Medicine,*** **53**(6):419-27.

Meir, R., R¨atsch, G., 2006. An introduction to boosting and leveraging. ***Advanced Lectures on Machine Learning.*** (Editör: S. Mendelson, A. Smola). Springer, Verlag.119-184.

Muhlbaier, M., Topalis, A., Polikar, R., 2005. Ensemble confidence estimates posterior probability. ***6th Int. Workshop on Multiple Classifier Systems, Lecture Notes on Computer Science,*** (Editörler: N. C. Oza, R. Polikar, J. Kittler, F. Roli), Monterey, CA. 3541: 326–335,.

Mukherjee, I., Schapire, R. E., 2011. A Theory of multiclass Boosting. ***Advances in Neural Information Processing Systems,*** **23**(4): 1722-1722.

Myatt, J. G., 2007. Making Sense of Data A Practical Guide to Exploratory Data Analysis and Data Mining. ***A John Wiley& Sons, Inc., Publication.*** Canada. 258.

Nanni, L., Lumini, A., 2006. An Eperimental comparison of ensemble of classifers for biometric data. ***Neuro Computing,*** **69** : 1670-1673.

Nizam, H., Akın, S., 2014. Sosyal Medyada Makine Öğrenmesi ile Duygu Analizinde Dengeli ve Dengesiz Veri Setlerinin Performanslarının Karşılaştırılması. ***XIX. Türkiye'de İnternet Konferansı Bildirileri.*** Yaşar Üniversitesi, İzmir. 129-136.

Okut, H., 2015. Karar Ağaçları (Decision Trees), ***Lisansüstü Ders Notları.***

Opitz, D.W., Maclin, R., 1999. Popular Ensemble Methods: An Empirical Study. ***Journal of Artificial Intelligence Research,*****11**: 169-198.

Oza, N. C., 2006. Ensemble Data Mining Methods. ***Encyclopedia of Data Warehouising and Mining.*** (Idea Group Reference). 448-452.

Pal, M., Mather P.M., 2003. An assessment of the effectiveness of decision tree methods for land cover classification. ***Remote Sensing of Environment,*** **86**: 554-565.

Prasad, A.M., Iverson, L.R., Liaw, A., 2006. Newer classification and regression tree techniques: bagging and random forests for ecologica prediction. ***Ecosystems,*** **9**: 181–199.

Ratsch, G., Onoda, T., Müller, K. R., 2001. Soft Margins for AdaBoost. ***Machine Learning,*** **42** (3): 287-320.

Rogova, G., 1994. Combining the results of several neural network classifiers. ***Neural Networks,*** **7** (5): 777–781.

Quinlan, R., 1996. Bagging, Boosting, and C4.5. ***In Proceedings of the Thirteenth National Conferenceon Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference,***AAAI Press and MIT Press :725-730.

Quinlan, J. R., 2006. Bagging, Boosting and C4.5. ***AAAI-96 Proceedings,*** 725-730.

Schapire, R. E., 1990. The strength of weak learnability. ***Machine Learning,*** **5** (2):197–227.

Schapire, R. E., 1999. Theoretical views of Boosting and applications. ***Proceedings of the 10th Iınternational Conference on Algorithmic Learning Theory***,13-25.

Schapire, R. E., 2002. The boosting approach to machine learning: An overview. In MSRI Workshop Nonlinear Estimation and Classification. ***Springer-Verlag,*** Berkeley. 332.

Schapire, R. E., Freund, Y., Bartlett, P., Lee, W., 1997. Boosting the margin: A new explanation for the effectiveness of voting methods. ***In Proceedings of the 14th International Conference onMachine Learning (ICML),*** Nashville, TN. 322-330.

Schapire, R. E., Freund, Y., 2012. ***Boosting: Foundations and Algorithms.*** MIT Press, Cambridge, London, England. 528.

Song, M., Breneman, C.M., Bi, J., Sukumar, N., Bennett, K.P., Cramer, S.M., 2002. Prediction of protein retention times in anionexchange chromatography systems

using support vector regression. ***Journal of Chemical Information and Computer Sciences,*** **42**(6): 1347-1357.

Tartar, A., Kılıç, N., Akan, A., 2013. Bagging support vector machine approaches for pulmonary nodule detection.***IEEE InternationalConference on Control, Decisionand Information Technologies.*** Tunisia, 047-050.

Tax, D. M. J., Breukelen, M., Duin, R. P. W., Kittler, J., 2000. Combining multiple classifiers by averaging or by multiplying. ***Pattern Recognition,*** **33**(9): 1475–1485.

Webb, G. I., 2000. MultiBoosting: A technique for combining boosting and wagging. ***Machine Learning,*** **40**: 159-196.

Wolpert, D. H., 1992. Stacked generalization. ***Neural Networks,*** **5**(2,): 241–259.

Woods, K., Kegelmeyer, W. P. J., Bowyer, K., 1997. Combination of multiple classifiers using local accuracy estimates. ***IEEE Transactions on Pattern Analysis and Machine Intelligence,*** **19**(4) : 405–410.

Xu, L., Krzyzak, A., Suen, C.Y., 1992. Methods of combining multiple classifiers and theirapplications to handwriting recognition. ***IEEE Transactions on Systems, Man and Cybernetics,*** **22** (3): 418–435.

Zeng, X. D., Chao, S., Wang, F., 2010. Optimization of Bagging Classifiers Based on SBCB Algorithm. ***Proceedings of the ninth International Conference on Machine Learning and Cybernetics.***11-14 July 2010, Qingdao. 262-267.

Zhang, C., Ma, Y., 2012. Ensemble Learning, Chap. 1. ***Ensemble Machine Learning***(Editor: R. Polikar). 1-17.

Zhou, Z. H., 2012. ***Ensemble Methods: Foundations and Algorithms.*** Chapman & Hall/CRC Machine Learning & Pattern Recognition Series. Boca Raton, FL, United States of America. 236.

EKLER

Ek 1. Düzey 1'e göre kodlar ve gruplar.

KOD	*Düzey 1*
TR1	İstanbul
TR2	Batı Marmara
TR3	Ege
TR4	Doğu Marmara
TR5	Batı Anadolu
TR6	Akdeniz
TR7	Orta Anadolu
TR8	Batı Karadeniz
TR9	Doğu Karadeniz
TRA	Kuzeydoğu Anadolu
TRB	Ortadoğu Anadolu
TRC	Güneydoğu Anadolu

Ek 2. Düzey 2'ye göre kodlar ve gruplar.

KOD	Düzey 2	KOD	Düzey 2
TR10	İstanbul	TR71	Kırıkkale
TR21	Tekirdağ	TR72	Kayseri
TR22	Balıkesir	TR81	Zonguldak
TR31	İzmir	TR82	Kastamonu
TR32	Aydın	TR83	Samsun
TR33	Manisa	TR90	Trabzon
TR41	Bursa	TRA1	Erzurum
TR42	Kocaeli	TRA2	Ağrı
TR51	Ankara	TRB1	Malatya
TR52	Konya	TRB2	Van
TR61	Antalya	TRC1	Gaziantep
TR62	Adana	TRC2	Şanlıurfa
TR63	Hatay	TRC3	Mardin

Ek 3. Düzey 3'e göre kodlar ve gruplar.

KOD	Düzey 3	KOD	Düzey 3	KOD	Düzey 3
TR100	İstanbul	TR411	Bursa	TR621	Adana
TR211	Tekirdağ	TR412	Eskişehir	TR622	Mersin
TR212	Edirne	TR413	Bilecik	TR631	Hatay
TR213	Kırklareli	TR421	Kocaeli	TR632	Kahramanmaraş
TR221	Balıkesir	TR422	Sakarya	TR633	Osmaniye
TR222	Çanakkale	TR423	Düzce	TR711	Kırıkkale
TR310	İzmir	TR424	Bolu	TR712	Aksaray
TR321	Aydın	TR425	Yalova	TR713	Niğde
TR322	Denizli	TR510	Ankara	TR714	Nevşehir
TR323	Muğla	TR521	Konya	TR715	Kırşehir
TR331	Manisa	TR522	Karaman	TR721	Kayseri
TR332	Afyon	TR611	Antalya	TR722	Sivas
TR333	Kütahya	TR612	Isparta	TR723	Yozgat
TR334	Uşak	TR613	Burdur	TR811	Zonguldak
KOD	**Düzey 3**	**KOD**	**Düzey 3**	**KOD**	**Düzey 3**
TR812	Karabük	TR905	Artvin	TRB21	Van
TR813	Bartın	TR906	Gümüşhane	TRB22	Muş
TR821	Kastamonu	TRA11	Erzurum	TRB23	Bitlis
TR822	Çankırı	TRA12	Erzincan	TRB24	Hakkari
TR823	Sinop	TRA13	Bayburt	TRC11	Gaziantep
TR831	Samsun	TRA21	Ağrı	TRC12	Adıyaman
TR832	Tokat	TRA22	Kars	TRC13	Kilis
TR833	Çorum	TRA23	Iğdır	TRC21	Şanlıurfa
TR834	Amasya	TRA24	Ardahan	TRC22	Diyarbakır
TR901	Trabzon	TRB11	Malatya	TRC31	Mardin
TR902	Ordu	TRB12	Elazığ	TRC32	Batman
TR903	Giresun	TRB13	Bingöl	TRC33	Şırnak
TR904	Rize	TRB14	Tunceli	TRC34	Siirt

Ek 4. MATLAB İstatistiksel yazılım programında yazılan Bagging kodları.

```
%load veriler
X=x(1:324-81,:);
% 2013 ün verileri son aşamada test verisi olarak kullanılacağı düşünüldü
% bu yüzden burada son 81 veriyi almiyoruz.
Y=y(1:324-81,:);

%Asagidaki asamalarda optimal yaprak buyuklugunu belirlemek uzere degisik yaprak buyulugune gore
%hata kareler ortalamasi
% oob hatasi hesaplanir.

leaf = [5 10 20 50 100];
col = 'rgbcmy';
figure(1);
for i=1:length(leaf)
b = TreeBagger(50,X,Y,'method','r','oobpred','on',...
'cat',1:3,'minleaf',leaf(i));
plot(oobError(b),col(i));
hold on;
end
xlabel('Number of Grown Trees');
ylabel('Mean Squared Error');
legend({'5' '10' '20' '50' '100'},'Location','NorthEast');
hold off;
%buradaki grafige bakilarak en küçük MSE değerinin hangi yaprak buyuklugunde gerceklestigi bulunur.



%Katki yapan ozelliklerin onem sirasi
```

```
b = TreeBagger(100,X,Y,'Method','R','OOBVarImp','On',...
    'cat',1:3,...
    'MinLeaf',5);

% Burada min leaf değeri önceki grafikteki en küçük MSE değerini veren
% yaprak sayısı 5 olduğu için 5 alındı.



%Hata egrisi gozlenerek egitim boyunca herseyin yolunda gittigi incelenebilir
figure(2)
plot(oobError(b));
xlabel 'Number of Grown Trees';
ylabel 'Out-of-Bag Mean Squared Error';



%Asagidaki MATLAB kodlari out-of-bag permut edildiginde hata kareler ortalamasindaki (HKO)
%artisa katki yapan degiskenlerin statusu belirlenir.
%Grafikte verilen "OOBPermutedVarDeltaError" HKO'nin esembledaki butun agaclardaki
%ortalama artisi butun degiskenlerdeki standart sapmasina bolunmesi ile elde edilir.
%BUYUK DEGERLER BUYUK KATKI YAPAN DEGISDKEN ANLAMINA GELIR. Keyfi olarak bir deger ornegin 0.20 alinarak bunun uzerinde
%bulunan degerlere ait degisken daha fazla katki sagladigi anlamina gelir
figure(3)
bar(b.OOBPermutedVarDeltaError);
xlabel 'Feature Number' ;
ylabel 'Out-of-Bag Feature Importance';
idxvar = find(b.OOBPermutedVarDeltaError>0.4)
```

```
idxCategorical = find(idxvar<4);


% OOB Indisi (TreeBagger bir ozelligi) hangi agac sayisinda in-bag olanlari
%belrlemek uzere asagidaki kodlar verilmekte

finbag = zeros(1,b.NTrees);
for t=1:b.NTrees
   finbag(t) = sum(all(~b.OOBIndices(:,1:t),2));
end
finbag = finbag / size(X,1);
figure(4)
plot(finbag);
xlabel 'Number of Grown Trees';
ylabel 'Fraction of In-Bag Observations';


%Buyuyen agac sayisina bagli olarak en onemli 5 ozelligin belirlenmesi

b5v = TreeBagger(100,X(:,idxvar),Y,'Method','R',...
   'OOBVarImp','On','cat',idxCategorical,...
   'MinLeaf',1);
figure(5)
plot(oobError(b5v));
xlabel 'Number of Grown Trees';
ylabel 'Out-of-Bag Mean Squared Error';
figure(6)
bar(b5v.OOBPermutedVarDeltaError);
xlabel 'Feature Index';
ylabel 'Out-of-Bag Feature Importance';
```

```
%Outlier bulmak amaciyla proximity matrisinin kullanimi
%fillProximities:
b5v = fillProximities(b5v);


%Outlier yapisina karsilik normalizastyon islemiute deviation for the entire sample.
figure(7)
hist(b5v.OutlierMeasure);
xlabel 'Outlier Measure';
ylabel 'Number of Observations';


%Verilerdeki klastur sayisini bulma
figure(8)
[~,e] = mdsProx(b5v,'colors','k');
[s,e] = mdsProx(b5v,'colors','rb');
xlabel('1st Scaled Coordinate');
ylabel('2nd Scaled Coordinate');


%Eaigen degerleri dikkata alinarak degiskenlerin onem durumu
figure(9);
bar(e(1:20));
xlabel('Scaled Coordinate Index');
ylabel('Eigenvalue');


% Ensemble Configuration'un save edilmesi
c = compact(b5v);
```

```
%Son 81 veri test ediliyor

testX=inputs(324-80:324,idxvar);
testY=outputs(324-80:324,:);
[predClass] = c.predict(testX);


xx=[1:1:81];
figure
plot(xx,predClass,xx,testY);
```

Ek 5. MATLAB istatistiksel yazılım programında yazılan Boosting kodları.

```
%essemble sifilandirmanin olususturulmasi
%Dikkat karar agaclari boostingde default olarak
% kullanilam zayif ogrenme yontemidir.
rng default % For reproducibility
T =324;
treeStump = templateTree('MaxNumSplits',1);
adaStump = fitcensemble(X,Y,'Method','AdaBoostM1','NumLearningCycles',T, ...
   'Learners',treeStump);
totalStump = fitcensemble(X,Y,'Method','TotalBoost','NumLearningCycles',T, ...
   'Learners',treeStump);
lpStump = fitcensemble(X,Y,'Method','LPBoost','NumLearningCycles',T, ...
   'Learners',treeStump);
figure;
plot(resubLoss(adaStump,'Mode','Cumulative'));
hold on
plot(resubLoss(totalStump,'Mode','Cumulative'),'r');
plot(resubLoss(lpStump,'Mode','Cumulative'),'g');
hold off
xlabel('Number of stumps');
ylabel('Training error');
legend('AdaBoost','TotalBoost','LPBoost','Location','NE');
```

```
%Her 3 essemble icerisinde agac sayisinin belirlenmesi

[adaStump.NTrained totalStump.NTrained lpStump.NTrained]

%C her 3 essemble algoritmasina croos validation durumu

cvlp = crossval(lpStump,'KFold',5);
cvtotal = crossval(totalStump,'KFold',5);
cvada = crossval(adaStump,'KFold',5);

figure;
plot(kfoldLoss(cvada,'Mode','Cumulative'));
hold on
plot(kfoldLoss(cvtotal,'Mode','Cumulative'),'r');
plot(kfoldLoss(cvlp,'Mode','Cumulative'),'g');
hold off
xlabel('Ensemble size');
ylabel('Cross-validated error');
legend('AdaBoost','TotalBoost','LPBoost','Location','NE');


%%%%%%% Cross validatin hatasinin küçültecek hiper parametrelerin elde edilisi
rng default
Mdl = fitcensemble(X,Y,'OptimizeHyperparameters','auto',...
    'HyperparameterOptimizationOptions',struct('AcquisitionFunctionName',...
    'expected-improvement-plus'))
```

```
%%%% AdaBoost LPBoost and TotalBoost agirliklarinin (parametre tahminlerinin)
durumu.


cada = compact(adaStump);

clp = compact(lpStump);

ctotal = compact(totalStump);


figure

subplot(3,1,1)

plot(clp.TrainedWeights)

title('LPBoost weights')

subplot(3,1,2)

plot(ctotal.TrainedWeights)

title('TotalBoost weights')

subplot(3,1,3)

plot(cada.TrainedWeights)

title('AdaBoost weights')
```

Ek 6. Bootstrap yoluyla elde edilmiş olan sınıflandırma olasılıkları.

grup 1'e dahil olma olasılığı	grup 2'ye dahil olma olasılığı	Dahil olduğu sınıfın tahmini	Gerçek değer (y)
0.11627907	0.88372093	1	1
0.052631579	0.947368421	1	1
0.081081081	0.918918919	1	1
0.476190476	0.523809524	1	1
0.055555556	0.944444444	1	1
0.135135135	0.864864865	1	1
0.09375	0.90625	1	1
0	1	1	1
0.032258065	0.967741935	1	1
0.023809524	0.976190476	1	1
0.727272727	0.272727273	0	1
0.125	0.875	1	1
0.181818182	0.818181818	1	1
0	1	1	1
0.028571429	0.971428571	1	1
0.321428571	0.678571429	1	1
0	1	1	1
0.022222222	0.977777778	1	1
0.176470588	0.823529412	1	1
0.028571429	0.971428571	1	1
0.571428571	0.428571429	0	1
0.028571429	0.971428571	1	1
0	1	1	1
0.108108108	0.891891892	1	1
0.078947368	0.921052632	1	1
0.184210526	0.815789474	1	1
0.135135135	0.864864865	1	1
0	1	1	1
0.1	0.9	1	1
0.822222222	0.177777778	0	0
0.21875	0.78125	1	1
0.046511628	0.953488372	1	1
0.068181818	0.931818182	1	1
0.243243243	0.756756757	1	1
0.244897959	0.755102041	1	1
0.083333333	0.916666667	1	1
0.025641026	0.974358974	1	1

grup 1’e dahil olma olasılığı	grup 2’ye dahil olma olasılığı	Dahil olduğu sınıfın tahmini	Gerçek değer (y)
0.051282051	0.948717949	1	1
0.033333333	0.966666667	1	1
0	1	1	1
0.162162162	0.837837838	1	1
0.121212121	0.878787879	1	1
0.096774194	0.903225806	1	1
0.261904762	0.738095238	1	1
0	1	1	1
0.060606061	0.939393939	1	1
0.615384615	0.384615385	0	1
0	1	1	1
0.545454545	0.454545455	0	1
0	1	1	1
0	1	1	1
0	1	1	1
0	1	1	1
0	1	1	1
0.105263158	0.894736842	1	1
0.558823529	0.441176471	0	1
0	1	1	1
0	1	1	1
0.047619048	0.952380952	1	1
0	1	1	1
0.024390244	0.975609756	1	1
0.081081081	0.918918919	1	1
0.685714286	0.314285714	0	1
0.083333333	0.916666667	1	1
0.512820513	0.487179487	0	1
0.051282051	0.948717949	1	1
0.058823529	0.941176471	1	1
0	1	1	1
0.142857143	0.857142857	1	1
0.019607843	0.980392157	1	1
0	1	1	1
0.666666667	0.333333333	0	1
0.897435897	0.102564103	0	0
0	1	1	1
0.111111111	0.888888889	1	1
0.162790698	0.837209302	1	1

grup 1'e dahil olma olasılığı	grup 2'ye dahil olma olasılığı	Dahil olduğu sınıfın tahmini	Gerçek değer (y)
0	1	1	1
0.075	0.925	1	1
0	1	1	1
0.063829787	0.936170213	1	1
0.102564103	0.897435897	1	1
0.266666667	0.733333333	1	1
0.175	0.825	1	1
0	1	1	1
0.297297297	0.702702703	1	1
0.162162162	0.837837838	1	1
0.117647059	0.882352941	1	1
0.048780488	0.951219512	1	1
0.085714286	0.914285714	1	1
0.051282051	0.948717949	1	1
0.096774194	0.903225806	1	1
0.157894737	0.842105263	1	0
0.1875	0.8125	1	1
0.333333333	0.666666667	1	1
0.230769231	0.769230769	1	1
0	1	1	1
0.228571429	0.771428571	1	1
0.078947368	0.921052632	1	1
0.026315789	0.973684211	1	1
0.119047619	0.880952381	1	1
0	1	1	1
0.125	0.875	1	0
0	1	1	1
0.095238095	0.904761905	1	1
0	1	1	1
0.068965517	0.931034483	1	1
0.081081081	0.918918919	1	1
0.2	0.8	1	1
0.027777778	0.972222222	1	1
0.039215686	0.960784314	1	1
0.731707317	0.268292683	0	0
0	1	1	1
0.034482759	0.965517241	1	1
0.117647059	0.882352941	1	1
0.371428571	0.628571429	1	1

grup 1'e dahil olma olasılığı	grup 2'ye dahil olma olasılığı	Dahil olduğu sınıfın tahmini	Gerçek değer (y)
0.258064516	0.741935484	1	1
0	1	1	1
0.058823529	0.941176471	1	1
0	1	1	1
0.02173913	0.97826087	1	1
0	1	1	1
0.333333333	0.666666667	1	1
0.022222222	0.977777778	1	1
0.028571429	0.971428571	1	1
0.022222222	0.977777778	1	1
0	1	1	1
0.060606061			
0.078947368	0.921052632	1	0
0	1	1	1
0.129032258	0.870967742	1	0
0	1	1	1
0.027027027	0.972972973	1	1
0.028571429	0.971428571	1	1
0.057142857	0.942857143	1	1
0	1	1	1
0	1	1	1
0.352941176	0.647058824	1	0
0	1	1	1
0.024390244	0.975609756	1	1
0.03030303	0.96969697	1	1
0	1	1	1
0.157894737	0.842105263	1	1
0.212765957	0.787234043	1	1
0.34375	0.65625	1	0
0.028571429	0.971428571	1	1
0.05	0.95	1	0
0.045454545	0.954545455	1	1
0	1	1	1
0	1	1	1
0	1	1	1
0.027027027	0.972972973	1	1
0.093023256	0.906976744	1	1
0.243902439	0.756097561	1	1
0.878787879	0.121212121	0	0

grup 1'e dahil olma olasılığı	grup 2'ye dahil olma olasılığı	Dahil olduğu sınıfın tahmini	Gerçek değer (y)
0.022222222	0.977777778	1	1
0.093023256	0.906976744	1	1
0.228571429	0.771428571	1	1
0.076923077	0.923076923	1	1
0.069767442	0.930232558	1	1
0.088235294	0.911764706	1	1
0	1	1	1
0	1	1	0
0.466666667	0.533333333	1	0
0.620689655	0.379310345	0	0
1	0	0	0
0.931034483	0.068965517	0	0
0.971428571	0.028571429	0	0
0.375	0.625	1	1
0.722222222	0.277777778	0	1
0.777777778	0.222222222	0	1
0.714285714	0.285714286	0	1
0.72972973	0.27027027	0	0
0.457142857	0.542857143	1	0
1	0	0	0
0.897435897	0.102564103	0	0
0.891891892	0.108108108	0	0
0.411764706	0.588235294	1	1
0.5	0.5	0	0
0.810810811	0.189189189	0	0
0.744680851	0.255319149	0	0
0.906976744	0.093023256	0	0
0.303030303	0.696969697	1	0
0.72972973	0.27027027	0	0
0.977272727	0.022727273	0	0
0.55	0.45	0	1
0.3125	0.6875	1	1
0.738095238	0.261904762	0	1
0.761904762	0.238095238	0	0
0.757575758	0.242424242	0	0
0.916666667	0.083333333	0	0
0.805555556	0	1	
1	0	0	0
0.324324324	0.675675676	1	0

grup 1'e dahil olma olasılığı	grup 2'ye dahil olma olasılığı	Dahil olduğu sınıfın tahmini	Gerçek değer (y)
0.925	0.075	0	0
0.393939394	0.606060606	1	0
0.333333333	0.666666667	1	0
0.717948718	0.282051282	0	1
0.548387097	0.451612903	0	0
0.75	0.25	0	0
0.475	0.525	1	1
0.916666667	0.083333333	0	0
0.636363636	0.363636364	0	1
0.897435897	0.102564103	0	0
0.675675676	0.324324324	0	1
0.823529412	0.176470588	0	0
0.973684211	0.026315789	0	0
0.413793103	0.586206897	1	0
0.861111111	0.138888889	0	0
0.935483871	0.064516129	0	0
0.848484848	0.151515152	0	1
0.970588235	0.029411765	0	0
0.676470588	0.323529412	0	0
0.913043478	0.086956522	0	1
0.4	0.6	1	0
1	0	0	0
0.780487805	0.219512195	0	0
0.684210526	0.315789474	0	1
1	0	0	0
0.441176471	0.558823529	1	1
0.59375	0.40625	1	1
0.545454545	0.454545455	0	0
0.710526316	0.289473684	0	0
0.971428571	0.028571429	0	0
0.647058824	0.352941176	0	0
0.710526316	0.289473684	0	0
0.870967742	0.129032258	0	1
0.891891892	0.108108108	0	0
0.973684211	0.026315789	0	0
0.833333333	0.166666667	0	0
0.825	0.175	0	0
0.65	0.35	0	1
0.695652174	0.304347826	0	1

grup 1’e dahil olma olasılığı	grup 2’ye dahil olma olasılığı	Dahil olduğu sınıfın tahmini	Gerçek değer (y)
0.95	0.05	0	0
0.827586207	0.172413793	0	0
1	0	0	0
0.6875	0.3125	0	0
0.5	0.5	0	0
0.615384615	0.384615385	0	0
0.405405405	0.594594595	1	1
0.470588235	0.529411765	1	1
0.696969697	0.303030303	0	0
0.586956522	0.413043478	0	0
0.888888889	0.111111111	0	0
0.741935484	0.258064516	0	0
0.825	0.175	0	0
0.833333333	0	0	0
0.944444444	0.055555556	0	0
1	0	0	0
0.088888889	0.911111111	1	0
0.735294118	0.264705882	0	0
0.707317073	0.292682927	0	0
1	0	0	0
0.8	0.2	0	0
0.657894737	0.342105263	0	0
0.939393939	0.060606061	0	0
0.928571429	0.071428571	0	0
0.928571429	0.071428571	0	0
0.444444444	0.555555556	1	0
0.75862069	0.24137931	1	0
1	0	0	0
0.848484848	0.151515152	0	0
1	0	0	0
0.965517241	0.034482759	0	0
0.810810811	0.189189189	0	0
1	0	0	0
0.692307692	0.307692308	0	0
0.432432432	0.567567568	1	0
0.606060606	0.393939394	0	0
0.935483871	0.064516129	0	0
0.609756098	0.390243902	0	0
0.977272727	0.022727273	0	0

grup 1'e dahil olma olasılığı	grup 2'ye dahil olma olasılığı	Dahil olduğu sınıfın tahmini	Gerçek değer (y)
0.641025641	0.358974359	0	0
1	0	0	0
0.71875	0.28125	0	0
1	0	0	0
0.763157895	0.236842105	0	0
0.513513514	0.486486486	0	0
0.305555556	0.694444444	1	0
0.951219512	0.048780488	0	0
1	0	0	0
0.926829268	0.073170732	0	1
0.72972973	0.27027027	0	0
0.714285714	0.285714286	0	0
0.864864865	0.135135135	0	0
0.675675676	0.324324324	0	0
1	0	0	0
0.970588235	0.029411765	0	0
0.829268293	0.170731707	0	0
0.666666667	0.333333333	0	0
1	0	0	0
0.896551724	0.103448276	0	0
0.96875	0.03125	0	0
0.827586207	0.172413793	0	0
0.757575758	0.242424242	0	0
0.545454545	0.454545455	0	0
1	0	0	0
1	0	0	0
0.540540541	0.459459459	0	0
0.975	0.025	0	0
0.341463415	0.658536585	1	0
0.707317073	0.292682927	0	0
0.931818182	0.068181818	0	0
0.894736842	0.105263158	0	0
1	0	0	0
0.76	0.24	0	0
0.96875	0.03125	0	0
0.953488372	0.046511628	0	0
0.885714286	0.114285714	0	0
0.96875	0.03125	0	0
0.931034483	0.068965517	0	0

grup 1’e dahil olma olasılığı	grup 2’ye dahil olma olasılığı	Dahil olduğu sınıfın tahmini	Gerçek değer (y)
0.944444444	0.055555556	0	0
0.464285714	0.535714286	1	0
0.6875	0.3125	0	0
0.970588235	0.029411765	0	0
0.829268293	0.170731707	0	0
1	0	0	0
0.657142857	0.342857143	0	0
0.888888889	0.111111111	0	0
0.538461538	0.461538462	0	0
0.586206897	0.413793103	0	1
0.617647059	0.382352941	0	0
0.892857143	0.107142857	0	0
0.923076923	0.076923077	0	0
0.931034483	0.068965517	0	0

ÖZ GEÇMİŞ

Tuğba TUĞ KAROĞLU 1979 yılında Van'da doğdu. İlk okulu Van Atatürk İlkokulu'nda, orta okul ve liseyi Vangölü Anadolu Lisesi'nde okudu. 1997 yılında başladığı Yüzüncü Yıl Üniversitesi Eğitim Fakültesi Matematik Öğretmenliği Bölümü'nden 2001 yılında birincilikle mezun oldu. Aynı yıl Matematik Bölümü'nde başladığı yüksek lisans eğitimini 2004 yılında tamamladı. 2011 yılında Yüzüncü Yıl Üniversitesi Fen Bilimleri Enstitüsü Zootekni Anabilim Dalı Biyometri-Genetik Bilim Dalı'nda doktora eğitimine başladı. Milli Eğitim Bakanlığına bağlı çeşitli okullarda Matematik öğretmeni ve yöneticilik görevlerinde bulundu. Halen İpekyolu İMKB Fen Lisesi'nde Okul Müdürü olarak görevine devam etmektedir. Evli ve bir çocuğu vardır.

T.C
VAN YÜZÜNCÜ YIL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
LİSANSÜSTÜ TEZ ORİJİNALLİK RAPORU

Tarih: 25/06/2018

Tez Başlığı / Konusu:

Lisans Yerleştirme Sınavında Yerleşme Başarısının Karar Ağaçlarına Göre 'Bagging' ve 'Boosting' Yöntemleriyle Sınıflandırılması

Yukarıda başlığı/konusu belirlenen tez çalışmamın Kapak sayfası, Giriş, Ana bölümler ve Sonuç bölümlerinden oluşan toplam 72 sayfalık kısmına ilişkin, 11/06/2018 tarihinde şahsım/tez danışmanım tarafından TURNİTİN intihal tespit programından aşağıda belirtilen filtreleme uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 2 (iki) dir.

Uygulanan filtreler aşağıda verilmiştir:

- Kabul ve onay sayfası hariç,
- Teşekkür hariç,
- İçindekiler hariç,
- Simge ve kısaltmalar hariç,
- Gereç ve yöntemler hariç,
- Kaynakça hariç,
- Alıntılar hariç,
- Tezden çıkan yayınlar hariç,
- 7 kelimeden daha az örtüşme içeren metin kısımları hariç (Limit inatch size to 7 words)

Van Yüzüncü Yıl Üniversitesi Lisansüstü Tez Orijinallik Raporu Alınması ve Kullanılmasına İlişkin Yönergeyi inceledim ve bu yönergede belirtilen azami benzerlik oranlarına göre tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Gereğini bilgilerinize arz ederim.

11/06/2018
Tarih ve İmza

Adı Soyadı: Tuğba TUĞ KAROĞLU
Öğrenci No: 10910220064
Anabilim Dalı: Zootekni
Programı: Biyometri-Genetik
Statüsü: Y. Lisans ☐ Doktora ☒

DANIŞMAN ONAYI
UYGUNDUR

(Prof. Dr. Hayrettin OKUT)

ENSTİTÜ ONAYI
UYGUNDUR

Prof. Dr. Fatih SEN
Enstitü Müdürü

(Unvan, Ad Soyad, İmza)