Gözetimli Öğrenme Algoritmalarının İstatistiksel Yöntemlerle Karşılaştırılması

Proje No: 109E186

Prof. Dr. Ethem Alpaydın
Doç. Dr. Olcay Taner Yıldız
Murat Semerci

Mart 2013
İstanbul

Önsöz

Birden çok gözetimli öğrenme algoritmasının birden çok veri kümesi üzerinde karşılaştırılarak belirli bir başarı ölçütüne göre en iyisinin istatistiksel olarak anlamlı biçimde bulunması, ya da daha genel bir tanımla iyiden kötüye doğru sıralanması, hem örüntü tanıma, hem veri madenciliği açısından önemli bir konudur. Bu projede farklı başarı ölçütlerini kullanarak, istenen sayıda öğrenme algoritmasını istenen sayıda veri kümesi üzerinde karşılaştıran ve sıralayan yeni istatistiksel yöntemler önerdik. Bu proje Tübitak tarafından desteklenmiştir.

İçindekiler

Tablo Listesi

Şekil Listesi

Özet

Birden çok gözetimli öğrenme algoritmasının birden çok veri kümesi üzerinde karşılaştırılarak belirli bir başarı ölçütüne göre en iyisinin istatistiksel olarak anlamlı (şans eseri oluşamayacak kadar büyük) biçimde bulunması, ya da daha genel bir tanımla iyiden kötüye doğru sıralanması, hem örüntü tanıma, hem veri madenciliği açısından önemli bir konudur. Birden çok algoritmayı karşılaştırmak, birden çok veri kümesi üzerinde karşılaştırma yapmak, ya da hata yerine başka ölçütler kullanmak ancak son yıllarda yapay öğrenme yazınında yer almaya başlamıştır.

Bu projede farklı başarı ölçütlerini kullanarak, istenen sayıda öğrenme algoritmasını istenen sayıda veri kümesi üzerinde karşılaştıran ve iyiden kötüye doğru sıralayan yeni istatistiksel yöntemler önerdik. Bu bağlamda, hem belli bir dağılım varsayan, hem de dağılımdan bağımsız sınamaların çok değişkenli hallerini iki ya da daha çok algoritmayı bir ya da daha çok veri kümesi üzerinde sıralayacak şekilde önerdik. Aynı zamanda, sınıflandırma hatası yanında menteşe yitimi ve ϵ-duyarlıklı yitim gibi başka ölçütlerin de bu sınamalarda kullanılabileceğini gösterdik. Son olarak bu çalışmalarımızı standart yapay öğrenme veri tabanları yanında biyoinformatik ve konuşma tanıma alanlarında uyguladık.

Önerdiğimiz yöntemler ve ölçütler, araştırmacılara sadece kendi algoritmalarının başarımlarını başkaları ile karşılaştırmada fayda sağlamayacak, aynı zamanda veri madenciliği gibi kullanıcıların yapay öğrenme konusunda uzman olmadıkları uygulamalarda birden fazla aday algoritmayı karşılaştırırken (veya sıralarken) otomatik yöntemlere olan ihtiyacı da karşılayacaktır.

Anahtar Kelimeler: Yapay öğrenme, örüntü tanıma, veri madenciliği, istatistiksel sınama, gözetimli öğrenme, istatistik.

Abstract

Finding the best of multiple supervised learning algorithms, or in the general case, ordering them from best to worst with statistical significance (with differences large enough that they could not have been due to chance), by comparing them on multiple data sets using different cost metrics is an important problem both in pattern recognition and data mining. Comparing multiple classification algorithms, making comparisons on multiple data sets, or using cost metrics other than misclassification error are topics that have become important only in recent years in the machine learning literature.

In this project, we developed novel statistical methods to compare multiple classification algorithms on multiple data sets and order them from best to worst. In order to accomplish this, we proposed multivariate counterparts of the univariate parametric and nonparametric tests both to compare two or more algorithms on an arbitrary number of data sets. While at the same time, we showed that performance metrics other than the error rate such as hinge and ϵ-sensitive loss can be used with these tests. We applied our approaches both to standard machine learning data sets and bioinformatics and speech processing applications.

Our proposed tests and performance metrics not only will allow researchers to compare their favorite algorithm's performance with existing ones, but especially in data mining applications where users are not necessarily experts in machine learning, an automated method such as ours will allow choosing the best of, or order, a number of candidate learning algorithms for a given application.

Keywords: Machine learning, pattern recognition, data mining, statistical tests, supervised learning, statistics.

Bölüm 1

Giriş

Bu projede

- Birden çok veri kümesi üzerinde gözetimli öğrenme algoritmalarını maliyet duyarlı olarak karşılaştırmada kullanmak için Multi^{2}Test adlı bir yöntem geliştirdik (Ulaş ve diğerleri, 2012).
- İki veya daha fazla sınıflandırma algoritmasını karşılaştırmak için çok değişkenli istatistiksel sınamaları geliştirdik ve önerdik (Yıldız ve diğerleri, 2011).
- Üstteki çalışmalarımızda kullandığımız sınamalar, çok değişkenli normal dağılımı varsayıyor ve bu varsayımın geçerliliği özellikle küçük verilerde kuşkulu olabiliyor. Projede ayrıca aynı amaç için dağılımdan bağımsız yöntemlerin kullanılması üzerine çalıştık.
- Çekirdek işlevli modellerin kullandığı menteşe ve ϵ-duyarlı yitim için özel denence sınamaları geliştirdik. Böyle bir sınama, çıktılar arasındaki farkı daha hassas olarak belirlemeyi ve böylesi çekirdek tabanlı yöntemleri daha doğru karşılaştırmayı sağlamaktadır (Yıldız ve Alpaydın, 2012).
- Biyoinformatik uygulamalarına özel olarak sınıflandırmada kullanılan başarım ölçütlerini, deney tasarımı yöntemlerini ve istatistiksel sınama türlerini inceledik ve üç önemli biyoinformatik dergisinde son iki yılda yayımlanmış 1,500'ün üzerinde makaleden derlediğimiz verilerle karşılaştırdık (İrsoy ve diğerleri, 2012a).
- Uygulama olarak yaptığımız bir çalışmada yeni bir karar ağacı mimarisi önerdik. Önerdiğimiz yapının iç düğümlerinde verilen kararlar bilinen karar ağaçlarının aksine kesin 0/1 değil, 0 ile 1 arasında sürekli bir sayıdır, her düğümde çocuklardan birine gitmek yerine, hepsine, bir geçit işlevinin belirlediği farklı olasılıklarla gidilir. Bu halinde henüz üzerinde çalıştığımız, ve bu çalışmayı bu projeyle ilişkilendirecek nokta, düğüm türleri arasında istatistiksel bir sınamayla seçim yapmak olacaktır (İrsoy ve diğerleri, 2012b).
- Tübitak 109E142 nolu projeyle ortak bir çalışmada farklı sınıflandırıcı ve sıralayıcıların başarılarını karşılaştırmak için istatistiksel sınama yöntemleri kullandık (Dikici ve diğerleri, 2013).

Bölüm 2

Genel Bilgiler

2.1 Başarım Ölçütleri

2.1.1 Hata Dizeyi ve Başarım Ölçütleri

İki sınıflı öğrenme kümesi üzerinde $f(x|\phi)$ sınıflandırıcımızı eğittikten sonra, geçerleme kümesinden seçilen bir x örneğini $f(x|\phi) \geq \theta$ ise artı, $f(x|\phi) < \theta$ ise eksi sınıfa atarız. x girdisinin gerçek sınıfına bağlı olarak dört durum söz konusudur ve başarımı ölçmek için bu durumların geçerleme kümesi içinde kaç kez geçtiğini sayarız (Tablo 2.1):

- Doğru artı (da): Hem gerçek sınıfı hem kestirilen sınıfı artı olan örnek sayısı.
- Yanlış eksi (ye): Gerçek sınıfı artı, kestirilen sınıfı eksi olan örnek sayısı.
- Yanlış artı (ya): Gerçek sınıfı eksi, kestirilen sınıfı artı olan örnek sayısı.
- Doğru eksi (de): Hem gerçek sınıfı hem kestirilen sınıfı eksi olan örnek sayısı.

Tablo 2.1: 2×2 hata dizeyi

	Kestirim		
Gerçek	+	-	Toplam
+	da	ye	a
-	ya	de	e
Toplam	a'	e'	N

Burada N, geçerleme kümesindeki örnek sayısı, a bunun içindeki artı, e ise eksi sayısıdır ($a + e = N$). Benzer biçimde, a' sınıflandırıcının artı karar verdiği örnek sayısı, e' sınıflandırıcının eksi karar verdiği örnek sayısıdır ($a' + e' = N$).

Sıklıkla bir yapay öğrenme uygulamasında, birden çok aday öğrenme algoritması içinden birini seçmemiz gerekir. Gözetimli öğrenmede modeller genelde hatalarına göre karşılaştırılır, ama bu her zaman en iyi sonucu vermez, çünkü hata, yanlış artıyla yanlış eksi arasında ayrım yapmaz. Bu yüzden hangi tür hataya odaklandığımıza göre farklı hata ölçütleri önerilmiştir. Örneğin örüntü tanımada bizim için doğru artı oranı ve yanlış artı oranı önemliyken bilgi getiriminde artı örneklere yoğunlaşırız ve kesinlik ve anma adını

verdiğimiz ölçütleri kullanırız. Yaygın kullanılan başarım ölçütleri şunlardır:

$$\begin{array}{ll} \textit{Hata oranı} = \frac{ya+ye}{N} & \textit{Başarı} = \frac{da+de}{N} \\ \textit{Doğru-artı oranı} = \frac{da}{a} & \textit{Yanlış-artı oranı} = \frac{ya}{e} \\ \textit{Anma} = \frac{da}{a} & \textit{Kesinlik} = \frac{da}{a'} \\ \textit{Duyarlılık} = \frac{da}{a} & \textit{Özgünlük} = \frac{de}{e} \end{array} \tag{2.1}$$

Doğru artı oranı, duyarlılık ve anmaya, yanlış-artı-oranı da $1-$özgünlük'e eşittir.

Öğrenme algoritmalarını karşılaştırırken sınıflandırma hatası üzerinde bir sınama yaptığımız zaman bazı farkları anlayamayabiliriz. Yukarıda bahsettiğimiz gibi hata, yanlış artı ve yanlış eksi arasında fark gözetmediği için bu sınamalar algoritmaların yanlış artı ve yanlış eksi çıktıları arasındaki farkı ayırt edemezler. İki sınıflandırıcının öğrenme hataları aynı olduğu halde, bir tanesinin tüm hatası yanlış artılar yüzünden gerçekleşirken ötekinin tüm hatası yanlış eksiler yüzünden olabilir. Karşılaştırma ölçütümüzün yalnızca hata olduğu durumda bu iki, çok farklı sınıflandırıcı arasındaki farkı ayırt edemeyiz.

Şekil 2.1(a)'da ortalamaları 2 ve 3'te olan iki normal dağılımı (eksi ve artı) ve her iki dağılımdan çekilen 100'er örneği görmekteyiz. Girdi değeri belirli bir eşiğin üzerinde olduğunda artı sınıfı seçen bir sınıflandırıcımız var. Bu karar eşiğini 2'den 3'e doğru değiştirdiğimizde farklı sınıflandırıcılar elde ediyoruz. Şekil 2.1(b)'de hata değişmemekte, yanlış artıler azalırken yakın oranda yanlış eksiler de artmaktadır. Şekil 2.1(c) ve Şekil 2.1(d)'de karar sınırı değiştiğinde (doğru artı, yanlış artı) ve (kesinlik, çağırılabilirlik) gibi ölçüt çiftlerinin farklılaştığını görebiliyoruz. Bu yüzden hata üzerinde çalışan bir istatistiksel sınama bu sınıflandırıcılar arasındaki farkı yakalayamazken, (doğru artı, yanlış artı) ya da (kesinlik, çağırılabilirlik) gibi ölçüt çiftleri üzerinde çalışan bir iki değişkenli sınama farkı yakalayabilecektir.

2.1.2 Kayıp, Risk ve Karar Sınırı

Tablo 2.2: 2×2 kayıp dizeyi

	Kestirim	
Gerçek	$+$	$-$
$+$	0	λ
$-$	1	0

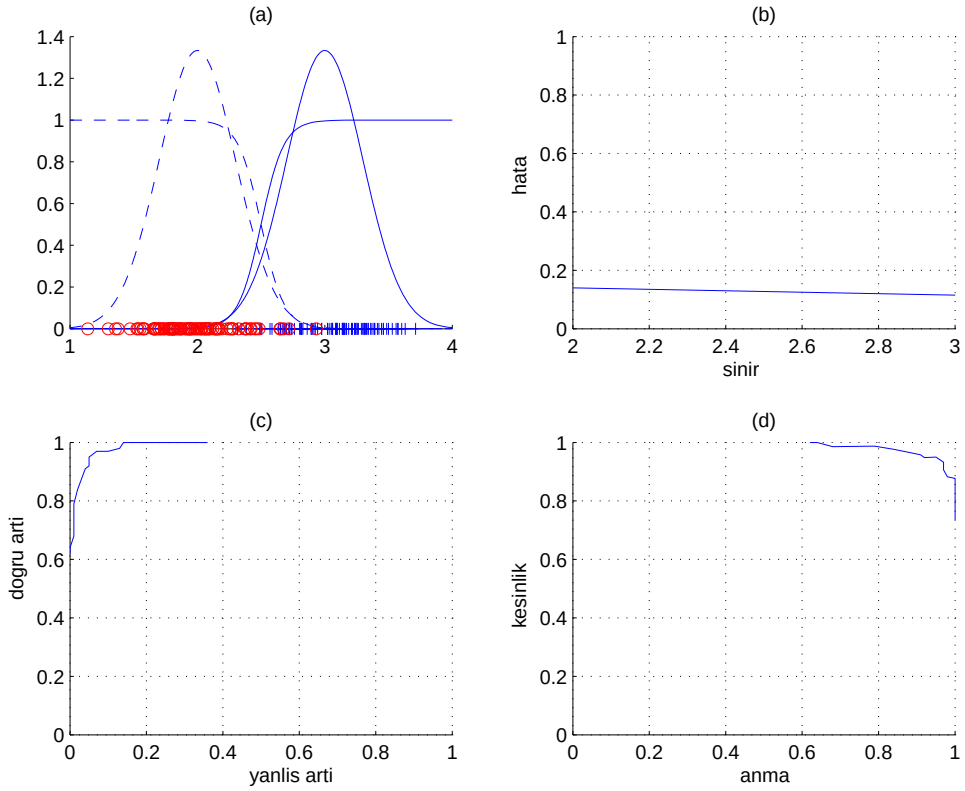
Artı ve eksi sınıflar arasında karar verirken çoğunlukla eşik değeri olarak $\theta = 0.5$ kullanılır. Bu da artı sınıf için kestirilen sonsal olasılığın eksi sınıf için kestirilen sonsal olasılıktan büyük olmasına karşılık gelir. Tablo 2.2'deki kayıp dizeyi verildiğinde, $\boldsymbol{x}$ örneğini artı sınıfa atamanın riski:

$$R(+|\boldsymbol{x}) = 0 \cdot P(+|\boldsymbol{x}) + 1 \cdot P(-|\boldsymbol{x}) = P(-|\boldsymbol{x})$$

ve eksi sınıfa atamanın riski de

$$R(-|\boldsymbol{x}) = \lambda P(+|\boldsymbol{x}) + 0 \cdot P(-|\boldsymbol{x}) = \lambda P(+|\boldsymbol{x})$$

olur. $R(+|\boldsymbol{x}) < R(-|\boldsymbol{x})$ ya da $P(-|\boldsymbol{x}) < \lambda P(+|\boldsymbol{x})$ ise artı sınıfı seçeriz ve $P(+|\boldsymbol{x}) + P(-|\boldsymbol{x}) = 1$ olduğundan $\boldsymbol{x}$'i artı sınıfa atamak ancak



Şekil 2.1: Sınıflandırma hatasının öğrenme algoritmalarını karşılaştırmada en iyi ölçüt olmadığını gösteren bir örnek.

$$P(+|\boldsymbol{x}) > \frac{1}{1+\lambda} \tag{2.2}$$

sağlandığında en düşük riskli harekettir. Bu da, 0.5 eşiğinin $\lambda = 1$ değerine karşılık geldiğini gösterir (Alpaydın, 2010); eğer yanlış artı ve yanlış eksinin farklı maliyetleri varsa, o duruma uygun bir θ seçmemiz gerekir.

2.1.3 Performans eğrileri ve bu eğrilerin altında kalan alanlar

Bazı durumlarda kayıp dizeyini bilmeyebilir ve θ değiştikçe başarım ölçütlerinin değerlerinin nasıl değiştiğini izlemek isteyebiliriz. Bu durumda, bunları θ'nın bir işlevi olarak çizmek ve genel davranışı görmek isteriz. Alıcı işletim özellikleri eğrisi, doğru artı oranı ile yanlış artı oranının bir eğrisidir. Benzer şekilde, kesinlik-anma ya da duyarlılık-özgünlük eğrileri de çizilebilir. Çalışmamızda bu eğrileri genelleştirerek başarım eğrileri kavramını önerdik ve daha önce verilmiş işlem karakteristiği eğrisi çizme algoritmasını (Fawcett, 2006) genelleştirdik. Sınıflandırıcılara karşılık gelen eğrileri karşılaştırmak zordur ve başarım eğrisini tek bir değere indirgeyerek özetlemek önerilmiştir. Bu durumda yapılan, eğrinin altında kalan alanı eğrinin ardışık noktalarının oluşturduğu yamukların alanlarının toplamı olarak kestirmektir (Fawcett, 2006).

Alıcı işletim özellikleri eğrisi ve altında kalan alan

Alıcı işletim özellikleri eğrisi, doğru artı oranı ile yanlış artı oranını betimler. (0,0) noktasından çizime başlanır. θ azaldıkça, doğru artılar ve beraberinde yanlış artıların sayısı artar. Sınıflandırıcı başarılıysa, doğru artıların artışı yanlış artıların artışından fazla olacaktır ve bu durumda eğri, sol üst köşeye yakın geçer. θ daha da azaldıkça, artı olarak sınıflandırmak giderek kolaylaşır ama yanlış artıların sayısı artar. (0,0) noktası bütün örnekleri eksi olarak sınıflandırır. (1,1) noktası tam tersini gösterir ve sınıflandırıcı bütün örnekleri artı olarak sınıflandırır.

İki sınıflandırıcıyı karşılaştırırken, bir sınıflandırıcıya ait alıcı işletim özellikleri eğrisi her zaman (bütün eşik değerleri için) ikinci sınıflandırıcıya ait alıcı işletim özellikleri eğrisinin üzerindeyse, birinci sınıflandırıcıyı ikinciye yeğleriz. Bazı durumlarda, birinci sınıflandırıcıya ait alıcı işletim özellikleri eğrisi uzayın bir parçasında (bazı eşik değerleri için) ikinci sınıflandırıcıya ait alıcı işletim özellikleri eğrisinin üzerinde, uzayın diğer bir kısmında ise altında olabilir. Bu da farklı kayıp koşullarında farklı sınıflandırıcıların tercih edileceği anlamına gelir.

Kesinlik-Anma eğrisi ve altında kalan alan

Kesinlik-anma eğrisi çoğunlukla bilgi erişiminde kullanılır (Zweig ve Campbell, 1993). Bir sorgu verildiğinde, veritabanında saklanmış kayıtlardan bazıları konuyla ilgili (artı), bazıları da değildir (eksi). Bir sorguyu gösteren x verildiğinde, bazı artı (ilgili) örnekler ve yanlışlıkla bazı eksi (ilgisiz) örnekler getirilir. Kesinlik, ilgili ve getirilmiş dökümanların tüm getirilen dökümanlara oranı, anma ise ilgili ve getirilmiş dökümanların tüm ilgili dökümanlara oranıdır. Yine karar eşiğini değiştirerek bir dizi kesinlik ve anma değeri elde edilebilir ve bunları birleştirerek bir kesinlik-anma eğrisi çizilebilir.

Eşik değeri θ, 1'e yaklaştıkça, getirilen belgelerin sayısı azalır ve çoğunun ilgili olması beklenir; böylece kesinlik, 1'e yaklaşır. İlgili belgelerin çok azı getirildiğinden anma küçük olacaktır. θ azaldıkça, getirilen belgelerin hepsi ilgili olmayacak ve kesinlik azalacak, fakat ilgili belgelerden daha fazlasını getirdiğimizden anma artacaktır. Alıcı işletim özellikleri eğrisindeki gibi, iki sınıflandırıcıyı karşılaştırırken bir sınıflandırıcıya ait kesinlik-anma eğrisi her zaman ikinci sınıflandırıcıya ait eğrinin üzerindeyse birinci sınıflandırıcıyı ikinci sınıflandırıcıya yeğleriz.

Alıcı işletim özellikleri eğrisi ile kesinlik-anma eğrileri farklı ölçütleri kullandığından altlarında kalan alan da farklı olabilir. Dolayısıyla alıcı işletim özellikleri eğrisinin altında kalan alana göre daha iyi olan bir sınıflandırıcı, kesinlik-anma eğrisinin altında kalan alana göre daha kötü bir sınıflandırıcı olabilir.

Kesinlik-anma eğrisi, sınıf oranına duyarlıyken alıcı işletim özellikleri eğrisi değildir (Davis ve Goadrich, 2006). a/e oranı değiştikçe, her iki satırdan değerler kullandığı için kesinlik değişir, ama doğru artı ve yanlış artı oranları tek bir satırdan değer kullandıkları için değişmeyebilir.

Kesinlik-anma ve alıcı işletim özellikleri eğrilerinin farklı uygulama alanları vardır. Kesinlik-anma eğrisinde, temel olarak artı örnekleri ne kadar iyi sınıflandırdığımızla ilgilenirken, alıcı işletim özellikleri eğrisinde, yanlış artı oranını azaltmaya çalışır ve doğru eksi oranını da artırmak isteriz. Örneğin bilgi erişiminde, verilen bir sorgu için ilgisiz bir çok belgeyi veri tabanına eklemenin sorgunun başarım sonucunu etkilememesini isteriz.

2.2 MultiTest

MultiTest (Yıldız ve Alpaydın, 2006), maliyet duyarlı bir yöntem olup sınıflandırma algoritmalarını hata oranına göre sıralar ve hatalarının eşit olması durumunda algoritmaların maliyetlerini bu eşitlikleri bozmak için kullanır. İki algoritmanın hata oranı aynı olduğunda, maliyet ölçütüne göre daha basit olanını yeğliyoruz. İkili istatistiksel sınamasının sonucuna göre, bu önkabulümüzü terk edip daha pahalı (karmaşık) olan algoritmayı yeğliyebiliyoruz (eğer hatanın azlığı istatistiksel olarak anlamlı ise).

Sınıflandırma algoritmalarını basitten karmaşığa maliyet ölçütüne göre 1, 2, ..., L olarak dizelim ve bunları bir çizgenin düğümleri olarak düşünelim. Daha sonra tüm ikili öğrenme algoritmaları için sıfır denencesi

$$H_0 : \mu_i \leq \mu_j$$

olan istatistiksel sınamayı uyguluyoruz. İstatistiksel sınama reddederse, daha maliyetli olan j. algoritmanın daha ucuz i. algoritmadan istatistiksel olarak daha az hatalı olduğu sonucuna varıp, i düğümünden j düğümüne bir ok koyuyoruz.

$L(L-1)/2$ istatistiksel sınamanın ardından, çizgede sadece istatistiksel sınamanın reddettiği durumlar için oklar bulunmaktadır. j düğümüne gelen okların sayısı j algoritmasından daha pahalı fakat istatistiksel olarak j'nin daha az hata yaptığı algoritmaların sayısını, j düğümünden giden okların sayısı da j algoritmasından daha pahalı fakat istatistiksel olarak j'den daha az hatalı algoritmaların sayısı göstermektedir. Çizge oluşturulduğunda ilingisel sıralamayla algoritmaları bu hata ve maliyet bilgisini kullanan biçimde sıralıyoruz (Örnek, Şekil 2.2'de verilmiştir).

2.3 Dağılıma Bağlı Sınamalar

2.3.1 İkili Karşılaştırma

İki sınıflandırma algoritmamız olduğunu varsayalım. Bu iki algoritmayı k tane öğrenme ve geçerleme katı üzerinde eğitip geçerlediğimizi ve k tane ayrı karıştırma dizeyi M_{ij}, $i=1, 2$, $j=1, \ldots, k$ hesapladığımızı düşünelim.

Tek Değişkenli Durum

Sınıflandırma hatası üzerinde karşılaştırma yapacaksak, her iki algoritma ve bütün k katlar için hatayı $e_{ij} = ya_{ij} + ye_{ij}$ olarak hesaplarız ve her katta sınıflandırıcıların hataları arasındaki fark

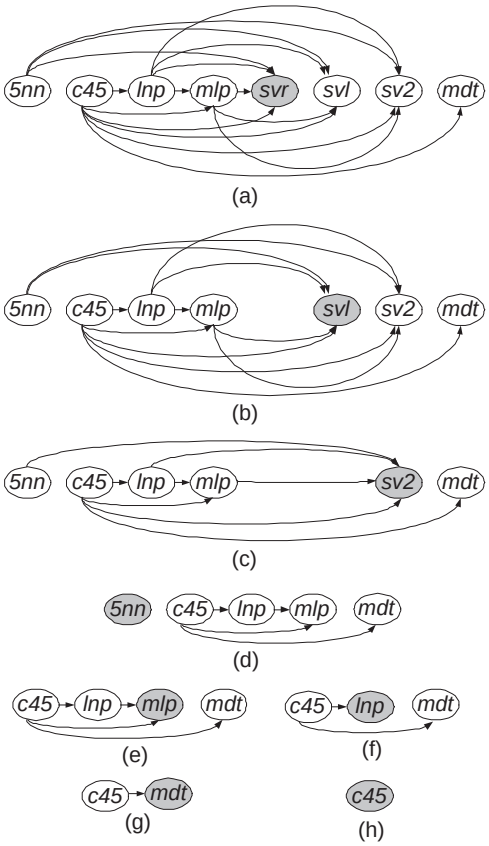
$$d_j = e_{1j} - e_{2j}$$

olur. Bu farkın 0 ortalamalı bir dağılımdan geldiğini sınarız:

$$H_0 : \mu_d = 0 \text{ karşı } H_1 : \mu_d \neq 0$$

Tek değişkenli eşli t sınaması için ortalama ve standart sapmayı hesaplayalım:

$$\overline{d} = \sum_{j=1}^{k} d_j/k \ , \ s_d = \frac{\sum_j (d_j - \overline{d})^2}{k-1}$$



Şekil 2.2: MultiTest yöntemi ile *optdigits* veri kümesi üzerinde, maliyet ölçütü öğrenme zamanı olarak alındığında üretilmiş çizge üzerinde yapılan ilingisel sıralama.

Bu iki algoritmanın beklenen sınıflandırma hataları aynı olduğunu belirten sıfır denencesine göre biliyoruz ki,

$$t' = \sqrt{k}\frac{\overline{d}}{s_d} \tag{2.3}$$

sınama istatistiği $k-1$ serbestlik derecesiyle t dağılır. Sıfır denencesi, $|t'| > t_{\alpha/2,k-1}$ ise $(1-\alpha)100$ % güvenle reddedilir.

Çok Değişkenli Durum

İki algoritmayı karşılaştırmak için tek bir ölçüt kullanmak yerine birkaç ölçüt kullanmak istersek sayılları değil yöneyleri karşılaştıran bir sınamaya ihtiyaç duyarız. Öyle bir durumda, p boyutlu dağılımların ortalamalarını karşılaştırırız, yani sıfır denencesi $H_0 : \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 = \mathbf{0}$ olan bir istatistiksel sınama yapmamız gerekir. (doğru artı, yanlış artı) ya da (kesinlik, anma) ölçütleri üzerinde bir karşılaştırma yapacaksak $p = 2$, tüm 2×2 karıştırma dizeyi üzerinden karşılaştırma yapacaksak, $p = 4$ olur.

$\boldsymbol{x}_{ij}$ ile p başarım değeri içeren bir başarım yöneyini gösterelim. Çok değişkenli eşli Hotelling sınaması için, eşli

$$\boldsymbol{d}_j = \boldsymbol{x}_{1j} - \boldsymbol{x}_{2j}$$

fark yöneylerini hesaplar ve fark yöneylerinin p değişkenli 0 merkezli bir normal dağılımdan

geldiğini kontrol eden istatistik sınamanın sıfır ve karşıt denencesi

$$H_0 : \boldsymbol{\mu}_d = \mathbf{0} \text{ karşı } H_1 : \boldsymbol{\mu}_d \neq \mathbf{0}$$

olacaktır. Ortalama fark yöneyi ve eşdeğişinti dizeyi ise

$$\overline{\boldsymbol{d}} = \sum_{j=1}^{k} \boldsymbol{d}_j / k \ , \ \mathbf{S}_d = \frac{1}{k-1} \sum_j (\boldsymbol{d}_j - \overline{\boldsymbol{d}})(\boldsymbol{d}_j - \overline{\boldsymbol{d}})^T$$

olacaktır. Bu iki algoritmanın beklenen başarım ölçüt yöneylerinin eşit olduğunu belirten sıfır denencesine göre biliyoruz ki (Rencher, 1995)

$$T'^2 = k\overline{\boldsymbol{d}}^T \mathbf{S}_d^{-1} \overline{\boldsymbol{d}} \tag{2.4}$$

sınama istatistiği p ve $k-1$ serbestlik dereceleriyle Hotelling T^2 dağılır. Sıfır denencesi, $T'^2 > T^2_{\alpha,p,k-1}$ ise $(1-\alpha)100$ % güvenle reddedilir.

Çok değişkenli sınamanın sıfır denencesi reddedilirse, p artçı tek değişkenli istatistik sınama uygulayarak hangi değişkenlerin çok değişkenli sıfır denencesinin reddedilmesine neden olduğu anlaşılabilir. Örneğin (kesinlik, anma) üzerine bir çok değişkenli sınama sıfır denencesini reddederse farkın, kesinlik mi, anma yüzünden mi olduğu tek değişkenli bir istatistiksel sınama ile anlaşılabilir.

2.3.2 Değişkenlik Çözümlemesi

$L > 2$ öğrenme algoritmasını karşılaştırırken aynı ortalama başarıma sahip olup olmadıkları sınanır. Tek değişkenli durumda karıştırma dizeyleri hata değerlerine indirgenir ve karşılaştırma yapılır; çok değişkenli durumdaysa başarım değerlerinden oluşan yöneyler karşılaştırılır.

Tek Değişkenli Durum

L algoritma verildiğinde, sıfır ile karşı denencesi

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_L \text{ karşılık } H_1 : \text{ en az bir } r, s \text{ ikilisi için } \mu_r \neq \mu_s$$

olacaktır. $e_{ij}, i = 1, \ldots, L, j = 1, \ldots, k$ ile i algoritmasının j geçerleme katındaki hatasını göstersin. $e_{i\cdot} = \sum_j e_{ij}/k$ ile i algoritmasının ortalama hatasını, $e_{\cdot\cdot} = \sum_i e_{i,\cdot}/L$ ile genel ortalamayı gösterelim. Tek değişkenli değişkenlik çözümlemesi (Univariate ANOVA)

$$\begin{aligned} F' &= \frac{MSH}{MSE} = \frac{SSH/(L-1)}{SSE/L(k-1)} \\ &= \frac{(\sum_i e_{i\cdot}^2/k - e_{\cdot\cdot}/Lk)(L-1)}{(\sum_{i,j} e_{ij}^2 - \sum_i e_{i\cdot}^2/k)/L(k-1)} \end{aligned} \tag{2.5}$$

sıfır denencesine göre $L-1, L(k-1)$ serbestlik dereceli F dağılımlıdır. Sıfır denencesi $F' > F_{\alpha,L-1,L(k-1)}$ ise $(1-\alpha)100$ % güvenle reddedilir. ANOVA reddederse ve en az iki algoritmanın istatistiksel olarak birbirinden farklı olduğunu biliyorsak, bölüm 2.3.1'deki istatistiksel sınamayı artçı sınama olarak uygulayıp hangi ikilinin hatadaki farka neden olduğunu anlayabiliriz.

Çok Değişkenli Durum

L algoritma verildiğinde, sıfır ile karşı denencesi

$$H_0 : \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \cdots = \boldsymbol{\mu}_L \text{ vs. } H_1 : \text{ en az bir } r, s \text{ ikilisi için } \boldsymbol{\mu}_r \neq \boldsymbol{\mu}_s$$

olacaktır. $\boldsymbol{x}_{ij}$ ile $i = 1, \ldots, L, j = 1, \ldots, k$, i algoritmasının j geçerleme katındaki p boyutlu başarım yöneyini göstersin. Çok değişkenli MANOVA öbeklerarası ve öbekiçi eşdeğişinti dizeylerini kullanır:

$$\begin{aligned}
\mathbf{H} &= k \sum_{i=1}^{L} (\overline{\boldsymbol{x}}_{i\cdot} - \overline{\boldsymbol{x}}_{\cdot\cdot})(\overline{\boldsymbol{x}}_{i\cdot} - \overline{\boldsymbol{x}}_{\cdot\cdot})^T \\
\mathbf{E} &= \sum_{i=1}^{L} \sum_{j=1}^{k} (\boldsymbol{x}_{ij} - \overline{\boldsymbol{x}}_{i\cdot})(\boldsymbol{x}_{ij} - \overline{\boldsymbol{x}}_{i\cdot})^T
\end{aligned}$$

Bu durumda

$$\Lambda' = \frac{|\mathbf{E}|}{|\mathbf{E} + \mathbf{H}|} \tag{2.6}$$

sınama istatistiği sıfır denencesine göre $p, L - 1, L(k - 1)$ serbestlik dereceleri ile Wilks dağılımından gelir (Rencher, 1995). MANOVA reddederse, her değişken üzerinde ayı ayrı olmak üzere p tane ANOVA sınaması yapılabilir. Ayrıca, bölüm 2.3.1'deki istatistiksel sınamayı artçı sınama olarak uygulayıp hangi ikilinin hatadaki farka neden olduğunu anlayabiliriz.

2.4 Dağılımdan Bağımsız Sınamalar

Dağılımdan bağımsız sınamada sıra dönüşümü yapılarak örneklerin mutlak değerleri yerine sıra değerleri karşılaştırılır (Kvam ve Vidakovic, 2007). Örneğin iki örneklemli sınamada değerler arasındaki fark yerine kimin daha küçük, kimin daha büyük olduğuna bakılır. Aşağıda bu sınamaların önce tek değişkenli, sonra çok değişkenli sürümlerini kısaca özetliyoruz.

2.4.1 Dağılımdan Bağımsız Tek Değişkenli Sınamalar

İki Örneklemli Sınamalar

Her $j = 1, \ldots, k$ kat için iki algoritmayı da j. öğrenme kümesi üzerinde eğittiğimizi, j. geçerleme kümesi üzerinde sınadığımızı ve $i = 1,2$ için x_{ij} başarım değerini (örneğin hata) elde ettiğimizi varsayalım. Eşli sınama yapıyor, yani tüm algoritmalar için aynı öğrenme ve geçerleme kümelerini kullanıyoruz. Yapmak istediğimiz iki x_j örneklem kümesinin aynı dağılımdan mı, yoksa iki farklı dağılımdan mı geldiğini anlamaya çalışmaktır.

Wald-Wolfowitz Sınaması Her biri k büyüklüğünde iki örneklem verildiğinde önce bunları birleştirip oluşan $2k$ sayıyı sıralarız, öyle ki en iyi sayı 1 sıra numarasını, ikinci 2 sıra numarasını, vs. alır. Sıralama yapılırken her sayının ait olduğu örneklem unutulmaz. Bir akış ardışık olarak aynı örneklemden gelen sayı dizisinin uzunluğunu göstermek üzere, sıralı $2k$ sayının içindeki toplam akış sayısı R, Wald-Wolfowitz sınamasının istatistiğini

oluşturur. Eğer söz konusu iki dağılım arasında istatistiksel olarak anlamlı bir fark yoksa sık sık bir örneklemden ötekine geçiş olmasını ve dolayısıyla R sayısının yüksek olmasını bekleriz. Asimptotik olarak

$$W = \frac{R - k - 1}{\sqrt{\frac{k(k-1)}{(2k-1)}}}$$

standart normal dağılımdan gelir. Eğer $W < Z_\alpha$ ise sıfır denencesini reddederiz.

Kolmogorov-Smirnov Sınaması İki dağılımdan her biri k büyüklüğünde iki örneklem verildiğinde yine önce bunları birleştirip oluşan $2k$ sayıyı sıralarız, öyle ki en iyi sayı 1 sıra numarasını, ikinci 2 sıra numarasını, vs. alır. Bu $2k$ sayı içinde, 1 ve $2k$ arasındaki herhangi bir i değeri için, birinci örneklemden i. değerden daha iyi olan kaç öğe olduğunu ve ikinci örneklemden i. değerden daha iyi kaç öğe olduğunu sayarız. Eğer iki dağılım arasında istatistiksel bir fark yoksa bu iki sayının tüm olası i değerleri için yakın olmasını bekleriz. Yaptığımız, her i için bu sayaçları hesaplayıp farklarını bulmak ve en büyüğünü belirlemektir. İki dağılım aynıysa, bu en büyük fark küçük olmalıdır. Kolmogorov-Smirnov sınamasının (Friedman ve Rafsky, 1979) altında yatan düşünce budur.

İki örneklemi X_1, X_2, birleştirilmiş değerleri de $x_{(j)}, j = 1, \ldots, 2k$ olarak gösterelim. Öğrenme algoritmalarını karşılaştırırken bu değerler k geçerleme kümesi üzerinde alınan başarım değerlerine karşılık gelecektir. Birinci örneklem içinde $x_{(i)}$'den daha iyi olan öğelerin sayısı

$$s^1_{(i)} \equiv \#_{x_{(j)} \in X_1}\{x_{(j)} \leq x_{(i)}\},\ i = 1, \ldots, 2k.$$

$s^2_{(i)}$ benzer şekilde tanımlanır. Mutlak fark ise

$$d_{(i)} = \frac{|s^1_{(i)} - s^2_{(i)}|}{k},\ i = 1, \ldots, 2k.$$

olarak tanımlıdır. İki dağılım arasında istatistiksel olarak anlamlı fark bulunmadığını söyleyen sıfır denencesine göre, bu farkların en büyüğü $2k$ serbestlik dereceli Kolmogorov dağılımından gelir. Eğer $K_{k,k,d} < \alpha$ ise sıfır denencesini reddederiz.

Çok Örneklemli Sınama

Eğer iki tane değil de $L > 2$ algoritmayı karşılaştırıyorsak çok örneklemli sınama yapmamız gerekir. Her $j = 1, \ldots, k$ kat için L algoritmayı da j. öğrenme kümesi üzerinde eğittiğimizi, j. geçerleme kümesi üzerinde sınadığımızı ve $i = 1, 2, \ldots, L$ için x_{ij} başarım değerini elde ettiğimizi varsayalım. Yapmak istediğimiz L tane x_j kümesinin aynı dağılımdan mı yoksa L farklı dağılımdan mı geldiğini anlamaya çalışmaktır.

Kruskal-Wallis Sınaması Elimizde L tane örneklem ve her birinde k değer olduğu için, bu değerleri sıraladığımızda en iyi başarım değeri 1 sıra numarasını, ikinci 2 sıra numarasını, ve sonuncu Lk sıra numarasını alır. r_{ij}, i. algoritmanın k. öğrenme başarımının sıra numarasını göstersin. Bütün ortalamaların aynı olduğu sıfır denencesine göre Kruskal-Wallis sınama istatistiği

$$X = \frac{12}{Lk(Lk+1)} \sum_{i=1}^{L} k\overline{R}_i^2 - 3(Lk+1) \tag{2.7}$$

$L-1$ dereceli ki-kare dağılımından gelir. Eğer $X > \chi^2_{\alpha,L-1}$ ise sıfır denencesini reddederiz.

2.4.2 Dağılımdan Bağımsız Çok Değişkenli Sınamalar

Gerek Kolmogorov-Smirnov ve Wald-Wolfowitz, gerekse Kruskal-Wallis sınamaları tek değişkenli örneklemler için tanımlanmış sınamalardır. Eğer yalnızca hata üzerinden değil birden çok başarım ölçütü üzerinden sınıflandırma algoritmalarını karşılaştırmak istersek her üç sınamanın da çok değişkenli örneklemler üzerinde uygulanabilecek şekilde genelleştirilmesi gerekir.

Bu sınamaları çok boyutlu veride uygulayabilmek için p boyutlu başarım yöneylerini sıralayabilmemiz gerekir. Bu amaçla önerilen yaklaşım, bunlardan bir ağaç oluşturmak ve bu ağaç üzerinde sıralama yapmaktır. Bunun için öncelikle başarım yöneylerini p boyutlu uzayda noktalar olarak düşünürüz. Ardından iki (ya da L) örnekleme ait $2k$ (ya da Lk) yöneylerinin düğümleri olduğu bir çizge tanımlanır ve bu düğümler arasındaki kenarların ağırlıkları karşılık gelen yöneyler arasındaki Euclid uzaklığına eşit olur. Daha sonra bu çizgenin en küçük kapsayan ağacı bulunur. Bu ağaç birbirine yakın olan yöneyleri birbirine bağlayacaktır (Friedman ve Rafsky, 1979).

İkili Sınamalar

Wald-Wolfowitz Sınaması Wald-Wolfowitz sınamasını genelleştirmek için, elde edilen ağaçtaki kenarlardan iki ucu farklı örneklemden olanlar silinir. Kalan bağlı parça sayısı bize Wald-Wolfowitz sınama istatistiği olan R sayısını verecektir.

Kolmogorov-Smirnov Sınaması Kolmogorov-Smirnov sınamasını genelleştirmek için, önce ağaçta çapı en yüksek olan düğüm belirlenir. Bir düğümün çapı, o düğümle başlayan en yüksek yolun uzunluğudur. Çapı en yüksek olan düğüm 1 sıra numarasını almak üzere, öteki düğümler yükseklik-öncelikli gezme algoritmasına göre sıra numarası alırlar. Yükseklik-öncelikli gezme algoritması özyinelemeli olarak şöyle tanımlanır: Önce kök düğüm ziyaret edilir, daha sonra yüksekliği en fazla olan çocuk ve tüm soyu ziyaret edilir, en sonunda da yüksekliği en düşük olan çocuk ve soyu ziyaret edilir. Düğümleri ziyaret etme sırası örneklerin sıra numaralarını belirleyecektir. Elde edilen sıra numaraları Kolmogorov-Smirnov sınamasında kullanılır (Friedman ve Rafsky, 1979).

Çok Örneklemli Sınama

Kruskal-Wallis Sınaması Elimizde iki algoritma yerine L tane algoritma olduğunda, yukarıda anlattığımız gibi Lk düğümlü bir çizge oluştururuz. Yine bu çizgenin en küçük kapsayan ağacı bulunur ve yükseklik-öncelikli gezme algoritmasına göre bu düğümlere sıra numarası verilir. Son olarak bu sıra numaraları kullanılarak Kruskal-Wallis sınaması uygulanır.

Bölüm 3

Gereç ve Yöntem

3.1 Maliyet Duyarlı Sınama İçin Multi^{2}Test Yöntemi

Multi^{2}Test yöntemi temel olarak MultiTest yönteminin birden çok veri kümesi üzerinde genelleştirilmesidir (Ulaş ve diğerleri, 2012). Önce MultiTest yöntemi (Bölüm 2.2'de anlatıldığı gibi bir ikili istatistiksel sınama ve maliyet ölçütü kullanılarak) ayrı ayrı veri kümeleri üzerinde uygulanır ve her algoritma için her veri kümesinde bir sıra numarası (1 en iyiyi, L en kötüyü gösterecek biçimde) belirlenir. Ardından, bu sıra numaraları, algoritmaları sıralamayan fakat ikişer ikişer istatistiksel farkları veren artçı bir istatistiksel sınamaya verilir. Sonra bu artçı istatistiksel sınamanın sonuçlarını ve ilk aşamada kullandığımız maliyet ölçütünü kullanan ikinci bir MultiTest'le algoritmaların son sırası oluşturulur. Bu ikinci MultiTest'te oluşturulan çizgenin okları artçı sınamanın sonuçlarına göre konulmaktadır.

Multi^{2}Test'in ikinci aşamasına örnek olarak, isimleri A, B, C, D olan dört sınıflandırıcımız ve bunların maliyete göre sırasının $C < A < D < B$ olduğunu varsayalım. Eğer artçı sınamaya göre A, C'den ve B, D'den istatistiksel olarak daha iyiyse oluşan çizge Şekil 3.1'de verilmiştir. İlingisel sıralama algoritması bu aşamada uygulandığında dört sınıflandırıcı, 1:A, 2:C, 3:B, 4:D olarak sıralanır.

Şekil 3.1: Örnek probleme göre Multi^{2}Test yönteminin ikinci aşamasında oluşturulan çizge.

3.2 Farklı Yitimler İçin Sınamalar

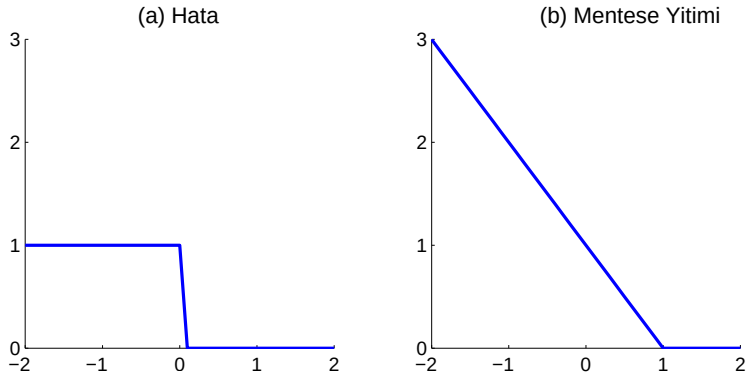
3.2.1 Menteşe Yitim Tabanlı Sınama

Yazındaki istatistiksel sınamalar, örneğin, eşli t sınaması, 5×2 çapraz geçerlenmiş t sınaması (Dietterich, 1998) gibi, 0/1 yitime karşılık gelen sınıflandırma hatasını kullanır. Destek yöney makineleri (Vapnik, 1995) eğitimde menteşe yitimini kıstas alır. Menteşe yitimi sınıflandırıcının verdiği kararın ayırtacın yalnızca doğru tarafında olmasına değil, kenar payı değerine de bakar. Bu çalışmamızda menteşe yitim tabanlı sınama için yöntemler önerdik (Yıldız ve Alpaydın, 2012).

x^t girdisi için çekirdek sınıflandırıcı çıktısını $f(x^t) \in \Re$ ve istenen çıktıyı $r^t \in \{-1, +1\}$ ile gösterelim. Bu durumda hata ve menteşe yitimi aşağıdaki gibi tanımlanacaktır (Şekil 3.2):

$$\text{Hata} = \begin{cases} 0 & \text{eğer } \text{sgn}(f(x^t)) = y^t \\ 1 & \text{aksi halde} \end{cases} = \begin{cases} 0 & \text{eğer } f(x^t)y^t \geq 1 \\ 1 & \text{aksi halde} \end{cases} \tag{3.1}$$

$$\text{Menteşe Yitimi} = \begin{cases} 0 & \text{eğer } f(x^t)y^t \geq 1 \\ 1 - f(x^t)y^t & \text{aksi halde} \end{cases} \tag{3.2}$$



Şekil 3.2: $y^t = 1$ için $f(x^t)$ cinsinden (a) sınıflandırma hatası ve (b) menteşe yitimi.

Sınıflandırma hatası yalnızca sınıflandırıcının çıktısının sınırın doğru tarafında olup olmadığına bakar; menteşe yitimi iki açıdan farklıdır:

(1) sınırın doğru tarafında ama kenar payının içinde olan örnekler de cezalandırılır,

(2) yanlış sınıflandırılmış örnekler yanlış tarafın ne kadar içinde olduklarıyla doğru orantılı olarak cezalandırılır.

İki çekirdek sınıflandırıcıyı bunları da göz önüne alarak karşılaştırmak davranışlarını daha iyi ayırt etmemizi sağlayacaktır. Gerek farklı çekirdekleri, gerekse de farklı tür girdileri içeren çekirdekleri kullanan çekirdek sınıflandırıcıları karşılaştırırken bizim baktığımız aralarında istatistiksel anlamda fark olup olmadığıdır. Örneğin, yeni önerilmiş bir çekirdeğin eldeki çekirdeklere göre iyileştirme getirip getirmediğini anlamak isteriz.

İstatistiksel sınama yaparken, iki algoritmayı birden fazla eğitim ve geçerleme kümesi üzerinde çalıştırır ve geçerleme kümesi üzerindeki sonuçlarının dağılımları aralarında istatistiksel olarak anlamlı (yani şansla oluşamayacak kadar büyük) bir fark olup olmadığına göre karşılaştırırız. Genellikle farklı (eğitim, geçerleme) veri kümesi çiftleri elde etmek için k-kat çapraz geçerleme kullanılır.

Deneylerimizde geçerleme kümeleri çok küçük değildi ve bu durumda yine merkezi limit kuramına göre menteşe değerlerinin de normal olduğunu varsayabiliriz. Normallik sınaması ile deneysel olarak baktığımızda da menteşe değerlerinin normal dağılımdan geldiğini söyleyebileceğimizi gördük. Gelmediği durumda, örneğin çok küçük veri kümelerinde, dağılımdan bağımsız sınama yöntemlerinin kullanılması gerekir.

3.2.2 ϵ-Duyarlı Yitim Tabanlı Sınama

x^t girdisi için çekirdek bağlanım algoritmasının çıktısını $f(x^t) \in \Re$ ve istenen normalleştirilmiş çıktıyı $y^t \in \Re$ ile gösterelim. Bu durumda kare hata ve ϵ-duyarlı yitim aşağıdaki gibi tanımlanacaktır (Vapnik, 1995):

$$\text{kare hata} = |y^t - f(x^t)|^2 \tag{3.3}$$

$$\epsilon\text{-duyarlı yitim} = \begin{cases} 0 & \text{if } |y^t - f(x^t)| \leq \epsilon \\ |y^t - f(x^t)| - \epsilon & \text{aksi halde} \end{cases} \tag{3.4}$$

Kare hatası yalnızca bağlanım algoritmasının çıktısı ile gerçek çıktı arasındaki farkı, bu fark ne olursa olsun cezalandırır; ϵ-duyarlı yitim iki açıdan farklıdır:

(1) bağlanım algoritmasının çıktısı ile gerçek çıktı arasındaki fark belirli bir sınırın altındaysa (epsilon duyarlılık) cezalandırma yapılmaz,

(2) bağlanım algoritmasının çıktısı ile gerçek çıktı arasındaki fark sınırın üstündeyse örnekler yanlış tarafın ne kadar içinde olduklarıyla doğrusal orantılı olarak cezalandırılır.

Menteşe yitimde olduğu gibi iki çekirdek bağlanım algoritmasını ϵ-duyarlı yitime göre karşılaştırmak davranışlarını daha iyi ayırt etmemizi sağlayacaktır.

3.3 Biyoinformatikte Sınıflandırma Deneylerinin Tasarımı Ve Sonuçların Çözümlenmesi

3.3.1 İstatistiksel Sınamalar

Sınıflandırma deneylerinde en sık kullanılan dört farklı denence sınaması senaryosu vardır (Tablo 3.1):

1. Bir veri kümesi üzerinde belirli bir başarım ölçütü üzerinden karşılaştırmak istediğimiz iki algoritmamız vardır. Bu en sık kullanılan senaryo türüdür. Örneğin iki algoritma, hata ya da eğri altında kalan alana göre karşılaştırmak istenir. Veya, aynı algoritmanın iki farklı türü karşılaştırılmak istenir; örneğin sinir ağı ile sınıflandırma yapmadan önce öznitelik seçiminin istatistiksel olarak fark yaratıp yaratmayacağı incelenir.

2. Bir veri kümesi üzerinde $L > 2$ algoritmayı belirli bir başarım ölçütüne göre karşılaştırmak isteriz. Bunlar farklı algoritmalar olabileceği gibi aynı algoritmanın farklı sürümleri de olabilir; örneğin sınıflandırıcının önündeki farklı öznitelik çıkarımı algoritmalarını karşılaştırmak isteyebiliriz.

3. $M > 1$ veri kümesi üzerinde iki algoritmayı bir başarım ölçütüne göre karşılaştırmak isteriz. Örneğin, elimizde M farklı kanser veri kümesi olsun ve farklı özelliklerinden dolayı bu veri kümelerini tek bir veri kümesi halinde birleştiremeyeceğimizi düşünelim. Yapmamız gereken her iki algoritmayı bütün veri kümeleri üzerinde eğitmek ve sınamaktır. Ardından, bu iki algoritmanın her bir veri kümesi üzerindeki başarımları karşılaştırılır ve bu karşılaştırmalar birleştirilip tek bir genel sonuç elde edilir.

4. $M > 1$ veri kümesi üzerinde $L > 2$ algoritmayı belirli bir başarım ölçütüne göre karşılaştırmak isteriz. Bu en genel senaryodur.

Tablo 3.1: Farklı karşılaştırma senaryoları ve kullanılan sınamalar.

Algoritma sayısı	Veri Kümesi Sayısı $M = 1$	$M > 1$
$L = 2$	5×2 cv F sınaması	Wilcoxon işaretli sıra sınaması
$L > 2$	ANOVA + 5×2 katlı F sınaması	Friedman ve Nemenyi sınaması

Bir Veri Kümesi Üzerinde İki Algoritmayı Karşılaştırma

Toplam hata sayısı 0/1 olaylarının bir toplamı olup, binom dağılımından gelmektedir. Geçerleme kümesi çok küçük olmadığı sürece, merkezi limit teoremi gereği, binom dağılımı normal dağılıma yakınsar ve normal dağılım varsayan sınamaları kullanabiliriz. İki algoritmanın beklenen başarım değerlerini karşılaştırdığımız zaman sıfır ile karşıt denenceleri

$$H_0 : \mu_1 = \mu_2 \text{ karşıt } H_1 : \mu_1 \neq \mu_2 \tag{3.5}$$

olup eşli durumda farkların ortalamasının 0 olduğunu sınarız (Bölüm 2.3.1):

$$H_0 : \mu_d \equiv \mu_1 - \mu_2 = 0 \text{ karşıt } H_1 : \mu_d \neq 0. \tag{3.6}$$

Dietterich (Dietterich, 1998) çalışmasında McNemar'ın sınaması ve k-kat çapraz geçerlemeli t sınaması da dahil olmak üzere çeşitli dağılıma bağlı sınamaları karşılaştırmıştır. Daha sonra 5×2 çapraz geçerlemeyi ve eşli t sınamasını önermiş, ve bu sınamanın Tip 1 ve Tip 2 hatalarının düşük olduğunu göstermiştir. 5×2 çapraz geçerlemeli F sınaması (Alpaydın, 1999) eşli t sınamasının gelişmiş bir halidir.

Bir Veri Kümesi Üzerinde $L > 2$ Algoritmayı Karşılaştırma

Değişkenlik çözümlemesi (ANOVA) bütün dağılımların aynı ortalamaya sahip olup olmadığını sınar (Bölüm 2.3.2):

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_L \text{ karşıt } H_1 : \text{ en az bir } r \neq s \text{ için } \mu_r \neq \mu_s. \tag{3.7}$$

şeklindedir.

Sınama sıfır denencesini reddetmezse bütün dağılımlar aynı ölçüde iyidir. Sınama reddederse, herhangi bir yerde eşitsizlik olduğunu anlarız. Nerede olduğunu anlamak için bir dizi ikili artçı sınama yapar ve hizipleri belirleriz. Bir hizibin içindeki herhangi iki algoritma arasında istatistiksel bir fark yoktur. Hizipleri belirlemek için önce L algoritmayı ortalama başarımlarına göre sıralar ve en iyi ile en kötü algoritmayı ikili olarak aralarında istatistiksel olarak fark var mı diye sınarız. Sınama reddederse, sonuncu algoritma hariç $L - 1$ algoritmayı alır ve birinci ile $L - 1$. algoritmaları ikili karşılaştırır; aynı zamanda birinci algoritma hariç olmak üzere ikinci ile L. algoritmaları ikili karşılaştırırız. Sınamalar reddettiği sürece, her iki taraftan özyinelemeli olarak birinci ve sonuncuyu dışarıda bırakacak şekilde devam ederiz. Herhangi bir aşamada sınama reddedemezse, o grup algoritmanın altını çizer ve daha fazla ilerlemeyiz.

$M > 1$ Veri Kümesi Üzerinde İki Algoritmayı Karşılaştırma

Farklı veri kümeleri üzerinden değerler hesapladığımızda bu veri kümelerinin başarımları aynı dağılımdan veya normal dağılımdan gelmedikleri için, dağılıma bağlı bir sınama kullanamayız. Bu yüzden, farklı veri kümeleri üzerinden başarım ortalamaları hesaplamak da mantıklı değildir. Bu durumda, farklı veri kümelerinin kaç tanesinde iki algoritmanın hangisinin daha iyi olduğunu sınayan dağılıma bağlı olmayan sınama yapabiliriz. Bu veri kümelerinin bazılarında birinci algoritma daha iyi iken, bazılarında ikinci algoritma daha iyi, bazılarında ise ikisi eşit iyilikte olabilir. İşaret sınamasında, sıfır denencesine göre bu iki algoritma aynı başarımı gösterdiklerinde kazanma/kaybetme/beraberlik sayılarının olası olup olmadığını kontrol ederiz.

Wilcoxon sınaması işaret sınamasının daha ileri bir sürümü olup sadece kazanma sayılarını değil bir algoritma bir veri kümesi üzerinde ötekinden iyi olduğunda, iki algoritma arasındaki başarım farkını da kullanabilmektedir.

$M > 1$ Veri Kümesi Üzerinde $L > 2$ Algoritmayı Karşılaştırma

İkiden fazla algoritma birden fazla veri kümesi üzerinde karşılaştırıldığında, elimizde her veri kümesi üzerinde kazanma/kaybetme/beraberlik sonuçları değil, ortalama başarımlarına göre belirlenmiş bir sıra numarası olur. Bu aşamada, M veri kümesi üzerinden alınmış ortalama sıra numaralarının istatistiksel olarak farklı olup olmadığı sınanır.

Friedman sınaması ANOVA sınamasının dağılım varsaymayan bir sürümü olup farklar yerine sıra numaralarını kullanmaktadır (Demsar, 2006). Friedman sınaması reddederse, Nemenyi sınamasını artçı sınama olarak kullanır ve ardışık algoritmaların ortalama sıralamaları arasındaki farkın istatistiksel olarak anlamlı olup olmadığını sınarız.

3.4 Ayrımcı Dil Modelleme İstatistiksel Sınama

Konuşma tanırken Ayrımcı Dil Modelleme (ADM)'de amaç daha iyi örnekleri daha kötü örneklerden ayırmaktır. Örnekler için akustik ses girdisinin öznitelik yöneyi, x, ve aday denencesi, y, birlikte $\Phi(x, y)$ olarak gösterilir. Eğitim kümesindeki her örnek için üretilen aday denenceler akustik puanlarına göre bir N-listesinde sıralanır. Amaç, bu listeyi yeniden sözcük hatalarına göre sıralayabilmektir; olası her yazıya dönüştürmedeki sözcük hata sayısı hedef sıra olarak kabul edlir. Bu uygulama, sıra sayılı bağlanıma benzemektedir: Eğitim kümesindeki örnekler sınıf etiketi yerine sıralara, r_a, atanmaktadırlar. Fakat sıra sayılı bağlanımdan farklı olarak tekrar-sıralama N-listedeki aynı cümleye ait örnekleri sıralar.

Bir tekrar-sıralama senaryosunda amaç en iyi ağırlık yöneyini, $\boldsymbol{w}$, belirlemektir; öyle ki aynı N-listesindeki iki denenceden, a ve b, eğer a daha az sözcük hatasına sahipse, yeniden sıralandığında daha üst bir sıraya atansın (listenin üst sıralarına daha yakın olması). Model çıktılarının arasında fark, tanımlanmış bir ayrım eşik değerinden daha büyük olmalıdır ($\lambda > 0$):

$$r_a \succ r_b \iff \langle \mathbf{w}, \mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b) \rangle > \lambda \tag{3.8}$$

Burada sıralar, sayısal sıralamanın tersidir; örneğin, $r_a = 1$ ve $r_b = 2$ ise $r_a \succ r_b$ olarak tanımlıdır.

Sıralama Algılayıcısı Tekrar-sıralamada doğrusal algılayıcı kullanıldığında kenar payı sıralara bağlıdır ve sıraların farkıyla orantılıdır (Shen ve Joshi, 2005):

$$r_a \succ r_b \Longleftrightarrow \langle \mathbf{w}, \mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b) \rangle \geq \tau g(r_a, r_b) \tag{3.9}$$

τ pozitif bir çarpan $g()$ ise aşağıdaki gibi tanımlanmış kenar payı işlevidir:

$$g(r_a, r_b) = \begin{cases} \dfrac{1}{r_a} - \dfrac{1}{r_b} & , r_a \succ r_b \\ 0 & , r_a \prec r_b \end{cases} \tag{3.10}$$

Seçilen bu sıralama işlevi, listenin altındaki denencelerin arasında listenin üstündekilere göre daha büyük bir ayrım yapmasını amaçlamaktadır. Ayrıca aşağıdaki kenar payısıra ilişkisinin korunmasını sağlar:

$$r_a \succ r_b \succ r_c \Longleftrightarrow \begin{cases} g(r_a, r_c) > g(r_a, r_b) \\ g(r_a, r_c) > g(r_b, r_c) \end{cases} \tag{3.11}$$

Önceki çalışmadan (Arısoy ve diğerleri, 2012) farklı olarak biz öğrenme oranı çarpanı, ν, uyguluyoruz ve güncelleme kuralını aşağıdaki gibi tanımlıyoruz:

$$\mathbf{w} = \mathbf{w} + \eta g(r_a, r_b)(\mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b)) \tag{3.12}$$

Öğrenme oranı her dönem sonrasında sönüm oranı ile çarparak küçültülür.

Sıralayıcı MIRA MIRA (Crammer ve Singer, 2003), her sınıf için bir tane asıl örnek eğiten bir sınıflandırma algoritmasıdır. Verilen örneğin doğru sınıf asıl örneği ile çarpımının en yüksek değer olması hedeflenir. Doğru sınıf çarpımının öteki sınıf nokta çarpımlardan farkı kenar payı olarak tanımlanır ve bu değer olabildiğince artırılmaya çalışılır. MIRA asıl örnekleri güncellerken öğrenme oranı modelin sıradaki örnekte yaptığı hataya göre belirlenir.

Biz bu çalışmada MIRA'yı değiştirerek sıralayıcı MIRA tanımladık (Dikici ve diğerleri, 2013). Bir asıl örneği eğitirken, asıl örnek güncellemelerini kenar payı işlevini sağlamayan ikililer üzerinden tanımlıyoruz. Model eğitimi aşağıdaki biçimde yapılmaktadır:

$$\mathbf{w} \leftarrow \mathbf{w} + \tau_{ab}(\mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b)) \tag{3.13}$$

$$\tau_{ab} = G'''\left(\frac{g(r_a, r_b) - \langle \mathbf{w}, \mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b) \rangle}{\| \mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b) \|^2}\right) \tag{3.14}$$

$$G'''(u) = \begin{cases} 0 & , u < 0 \\ u & , 0 \leq u \leq g(r_a, r_b) \\ g(r_a, r_b) & , g(r_a, r_b) < u \end{cases} \tag{3.15}$$

Sıralayıcı DYM Sınıflandırıcı Destek Yöney Makinesi (DYM)'nin bir türevi olan bu sıralayıcı, örnek ikililerinin oluşturduğu kısıtlara göre (yüksek sıralı örneğin ayrım değeri düşük sıralı örneğinkinden yukarıda olmalıdır) en büyük kenar paylı üstün-düzlemi bulmaya çalışır.

$$\begin{aligned} \min_{\mathbf{w}} \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{a,b} \xi_{ab} \\ \text{s.t. } \langle \mathbf{w}, \mathbf{\Phi}(x_a, y_a) - \mathbf{\Phi}(x_b, y_b) \rangle \geq 1 - \xi_{ab} \\ \forall (a,b) \in P,\, \xi_{ab} > 0. \end{aligned} \tag{3.16}$$

Buradaki C, ödünleşim değeri, P ise $r_a \succ r_b$ olarak tanımlı ikililer kümesidir. Bir bakıma ikililer arasında tanımlanmış bir sınıflandırma problemi olarak da düşünülebilir.

Biz bu çalışmada farklı sıralayıcı algoritmaların aralarında ve sınıflandırıcılarla karşılaştırdık ve bu karşılaştırmada istatistiksel sınama yöntemlerini kullanarak farkların istatistiksel olarak anlamlı olup olmadığına baktık.

Bölüm 4

Bulgular

4.1 Farklı Başarım Ölçütlerinin Karşılaştırılması

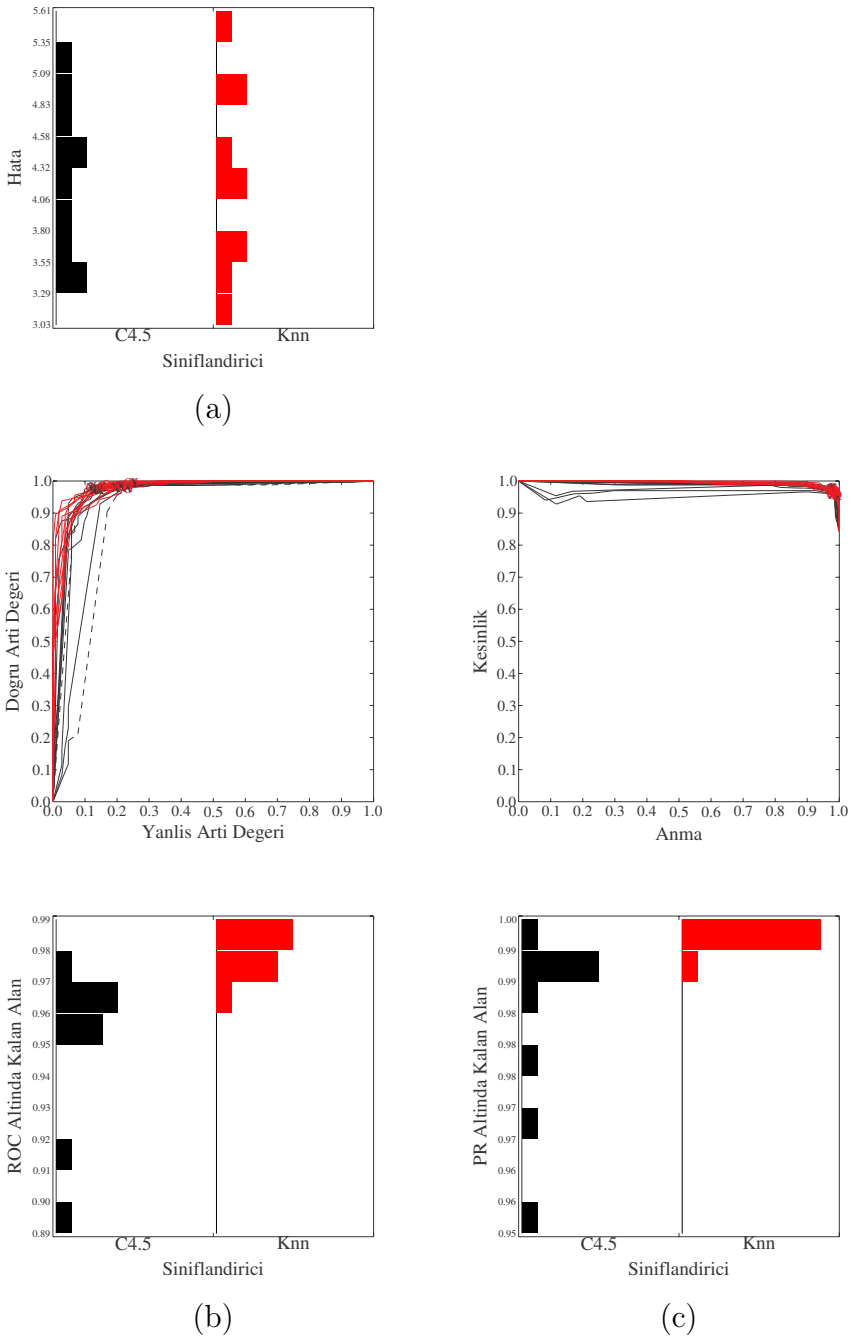
Şekil 4.1, hata sınamasının iki sınıflandırıcının eşit başarımı gösterdiğini söyleyen sıfır denencesini kabul ettiği, fakat AUC-ROC (alıcı işletim özellikleri eğrisinin altında kalan alanı kullanan sınama) ile AUC-PR (kesinlik-anma eğrisinin altında kalan alanı kullanan sınama) sınamalarının reddettiği bir örneği göstermektedir. Sınıflandırıcılar c45 (karar ağacı) ve knn (en yakın k komşu) olup veri kümesi UCI veri bankasından (Blake ve Merz, 2000) alınan *musk2* verisidir. İlk alt şekilde, hata dağılımları üst üste gelmektedir; bu da hata sınamasının kararını desteklemektedir. AUC-ROC ve AUC-PR dağılımları birbirinden yeterince ayrık olduklarından AUC-ROC ve AUC-PR sınamaları sıfır denencesini reddetmektedirler. Eğrilerin üstünde 0.5 karar eşiğine karşılık gelen nokta işaretlenmiştir ki bu hata sınamasının kullandığı değere karşılık gelmektedir. Her iki eğri türünde de knn, c45'in hep üstündedir, dolayısıyla bu iki sınıflandırıcı tüm olası karar sınırları için farklı davranmaktadır. Fakat sadece 0.5 karar sınırında birbirine yakınlaşırlar ve bu da hata sınamasının sıfır denencesini kabul etmesine neden olur. Bu sonuçtan, AUC-ROC ve AUC-PR sınamalarının hata sınamasının fark edemediği eşitsizlikleri fark edebildiğini, dolayısıyla güçlerinin daha yüksek olduğunu görebiliyoruz.

Şekil 4.2'de hatanın sıfır denencesini reddettiği fakat AUC-ROC ile AUC-PR sınamalarının kabul ettiği bir örneği görüyoruz. İlk alt şekile bakarsak c45 ve doğrusal sınıflandırıcının (lda) *musk2* veri kümesi üzerinde hata dağılımlarının birbirinden yeterince ayrık olduğu görülmektedir. Buna karşın hem AUC-ROC hem AUC-PR dağılımlarında her iki sınıflandırıcı arasında istatistiksel bir fark görünmemektedir. Eğriler kesişmektedir, ve kesişim noktasının solunda c45 daha iyi, sağında ise lda daha iyidir. Her ne kadar hata sınaması bu iki sınıflandırıcı farklı dese de (çünkü 0.5 karar eşiğinde birbirlerinden ayrıklar), bütün kayıp değerleri üzerinden ortalama aldığımızda (eğri altında kalan alana bakınca) aralarında bir fark görünmemektedir, yani bu tür sınamaların daha düşük tip 1 hatası vardır.

4.2 Multi^{2}Test Sonuçları

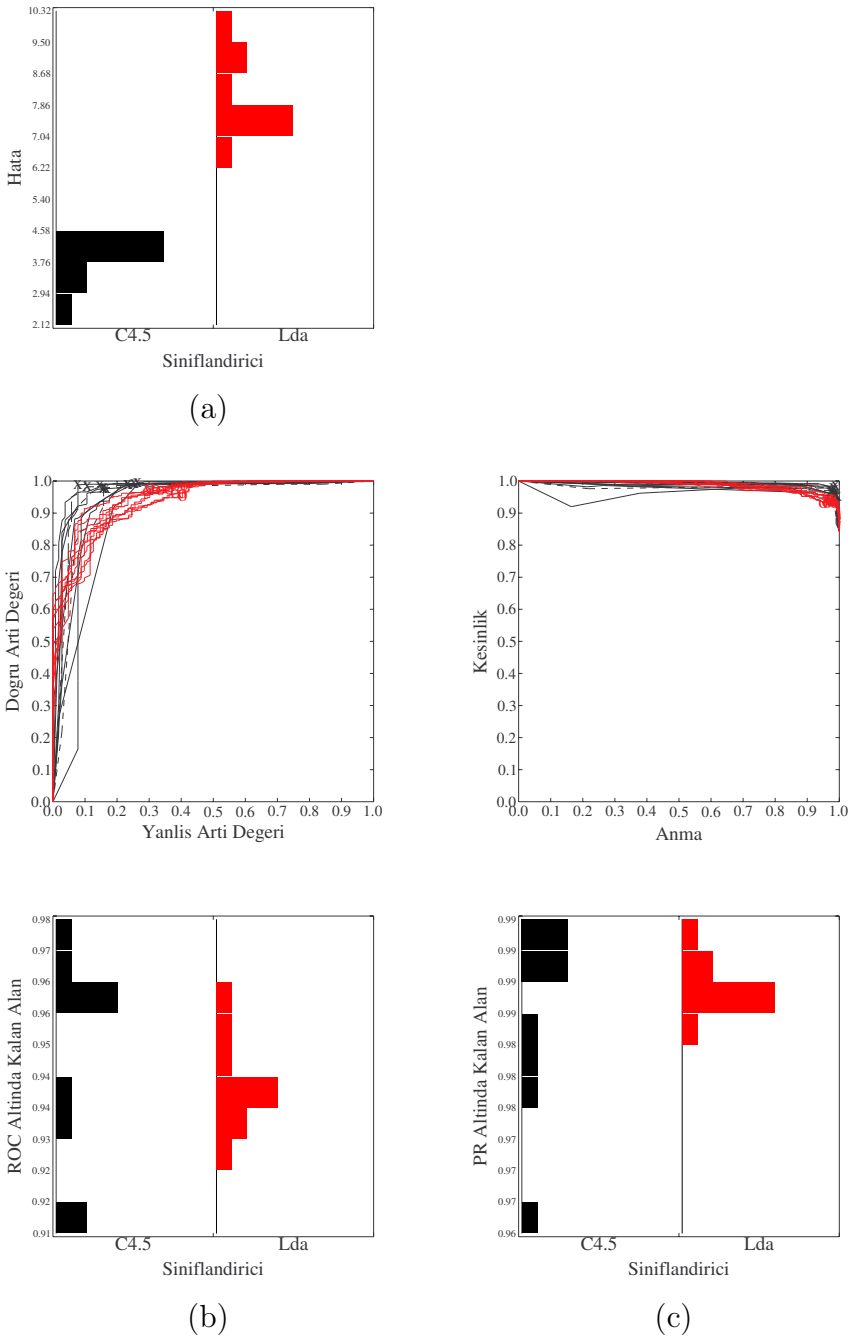
4.2.1 Deney Kurulumu

Multi2Test ile ilgili deneylerde 38 veri kümesi kullandık (Ulaş ve diğerleri, 2012). Bu veri kümelerinden 35 tanesini UCI veri bankası'ndan (Blake ve Merz, 2000), 3 tanesini de Delve



Şekil 4.1: Hata sınamasının sıfır denencesini kabul ettiği fakat AUC-ROC ve AUC-PR sınamalarının reddettiği bir örnek.

veri bankası'ndan (Hinton, 1996) aldık. Kullandığımız öğrenme algoritmaları, C4.5 karar ağacı, çok değişkenli karar ağacı (mdt) (Yıldız ve Alpaydın, 2000), çok katmanlı yapay sinir ağları (mlp), doğrusal yapay sinir ağı (lnp), destek yöney makinelerinin (Chang ve Lin, 2001) doğrusal (svl), Gauss (svr) ve ikinci dereceden polinom (sv2) çekirdek kullanan sürümleri ve en yakın 5 komşu (5nn)'dur.



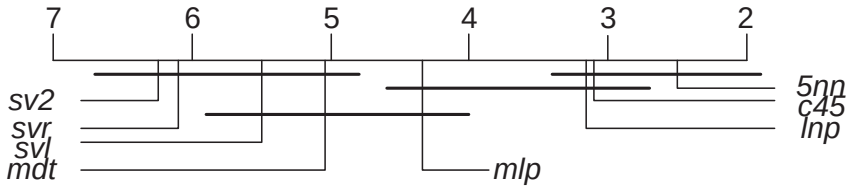
Şekil 4.2: Hata sınamasının sıfır denencesini reddettiği fakat AUC-ROC ve AUC-PR sınamalarının kabul ettiği bir örnek.

4.2.2 Sonuçlar

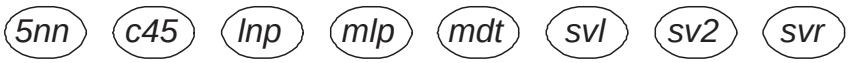
MultiTest için maliyet ölçütü olarak öğrenme zamanı kullanıldığında algoritmaların üç öbek olarak kümelendiğini görüyoruz (Şekil 4.3 ve Tablo 4.1). Maliyeti yüksek destek yöney makineleri ile çok değişkenli karar ağacı bir öbek oluştururken daha hızlı olan 5nn, c45 ve lnp'den istatistiksel olarak farklılar. Ortadaki mlp, kendisinden yavaş olan sv2, svr ve en hızlı 5nn'den istatistiksel olarak farklı iken diğer algoritmalar ile aralarında

Tablo 4.1: Multi^{2}Test kullanılarak karşılaştırılan algoritmaların ortalama sıraları

c45	*mdt*	*mlp*	*lnp*	*svl*	*sv2*	*svr*	*5nn*
3.11	5.05	4.37	3.13	5.50	6.24	6.11	2.50



Şekil 4.3: Bergman-Hommel yöntemine göre öğrenme algoritmalarının grafik olarak gösterimi



Şekil 4.4: Multi^{2}Test yönteminin ikinci aşaması sonucunda oluşam MultiTest çizgesi

istatistiksel bir fark bulunmamaktadır.

Bu sonuçların ardından elimizde hâlâ bir sıralama bulunmamaktadır. Multi^{2}Test'in ikinci aşamasını ortalama maliyetleri kullanarak uyguluyoruz. Ortalama öğrenme zamanına göre, ön sıralama 5nn < c45 < lnp < mlp < mdt < svl < sv2 < svr olarak bulunur. Bu ön sıralamanın sonuçlarını (Şekil 4.3) kullanarak Şekil 4.4'teki çizgeye ulaşıyoruz. Hiçbir sınama sonucunun ön sıralama sonucunu bozmadığını görüyoruz. Sonuç olarak sıralama, 1: 5nn, 2:c45, 3:lnp, 4: mlp, 5:mdt, 6:svl, 7:sv2, 8:svr olmaktadır.

4.3 Dağılıma Bağlı Sınamalar

4.3.1 Deney Kurulumu

27 tanesi UCI veri bankasından (Blake ve Merz, 2000), 3 tanesi Delve veri bankasından (Hinton, 1996) ve altı tanesi de biyoinformatik veri kümesi (Statnikov ve diğerleri, 2005) olmak üzere 36 tane iki sınıflı veri kümesi kullandık. 10-katlı çapraz geçerleme ile öğrenme ve geçerleme kümelerini oluşturduk. Başarım karşılaştırmasında kullanmak üzere 5 öğrenme algoritması belirledik. c45: C4.5 karar ağacı öğrenme algoritması, svm: Doğrusal çekirdek kullanan destek yöney makinesi (Chang ve Lin, 2001), lda: Doğrusal sınıflandırıcı, qda: İkinci dereceden sınıflandırıcı, knn: k = 20-en yakın komşu sınıflandırıcısı.

4.3.2 Sonuçlar

Tek değişkenli ile çok değişkenli sınamaların karşılaştırılması

Deneylerimizin birinci bölümünde, hatayı kullanan tek değişkenli k-katlı eşli t sınaması (UniErr) ile (doğru artı, yanlış artı) çiftini kullanan önerdiğimiz çok değişkenli eşli sına-

masını (MultiTF) karşılaştırdık.

Şekil 4.5'te tek değişkenli sınamanın sıfır denencesini reddedemediği, MultiTF sınamasının ise sıfır denencesini reddettiği bir örneği görüyoruz. Şekil 4.5(a)'da her iki algoritma için 10 çalıştırma sonucunda *breast* veri kümesinde elde edilen (doğru artı, yanlış artı) dağılım çizimini ve bu dağılımların üzerine oturtulan iki boyutlu normal dağılımları görüyoruz. Şekil 4.5(b)'de sınıflandırıcıların hata çubukçizitlerinin benzer olduğunu görüyoruz. Lda sınıflandırıcısının yanlış artı sayısı fazlayken, qda sınıflandırıcısının yanlış eksi sayısı fazladır. Şekil 4.5(c)'de fark yöneyleri üzerine oturtulan normal dağılımın merkezinin (0,0) noktasından uzak olduğunu ve bu yüzden çok değişkenli istatistiksel sınamanın sıfır denencesini reddettiğini görüyoruz. Şekil 4.5(d)'de hata farklarının çubukçizitinde merkez noktanın 0'a yakın olduğunu, bu yüzden tek değişkenli istatistiksel sınamanın sıfır denencesini reddedemediğini görüyoruz.

Çok değişkenli sınamaların karşılaştırılması

Deneylerimizin ikinci bölümünde, çok değişkenli sınamalarda farklı ölçütlerin kullanılmasının etkisini görmek için (kesinlik, anma) üzerinde tanımlanan çok değişkenli istatistiksel sınama ile (MultiPR), (doğru artı, yanlış artı) üzerinde tanımlanan çok değişkenli istatistiksel sınamayı (MultiTF) karşılaştırdık.

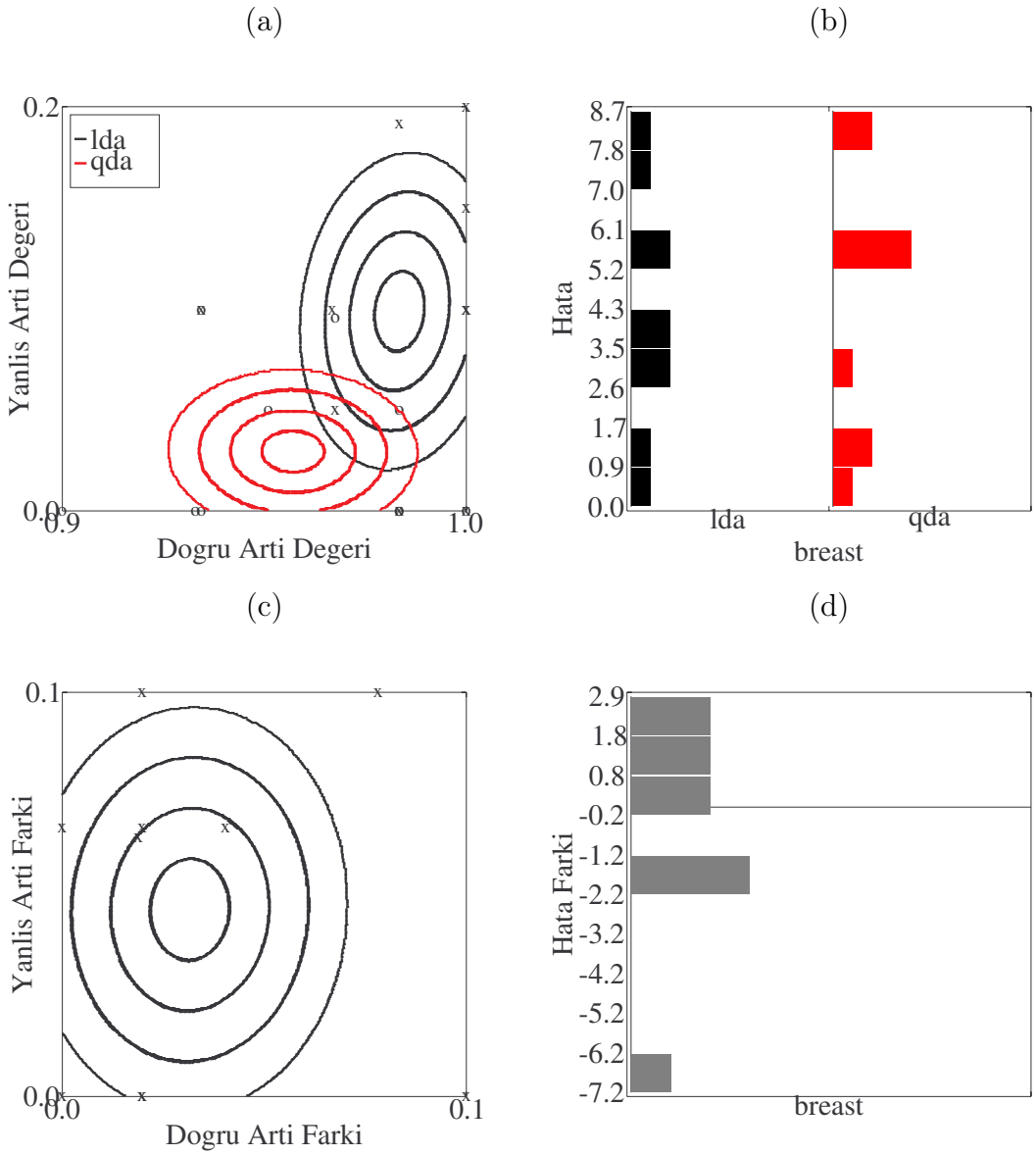
Şekil 4.6, *pima* veri kümesinde MultiTF sınamasının sıfır denencesini reddettiği, fakat MultiPR sınamasının sıfır denencesini reddedemediği bir örneği göstermektedir. Şekil 4.6(a) ve (b)'de, doğru artı ile anma aynı olduğundan x eksenleri aynıdır; bu iki şeklin y eksenindeli farkları kararların neden farklı olduğunu anlamamızı sağlar. (Doğru artı, yanlış artı) çiftine (Şekil 4.6(a)) göre c45 ve qda sınıflandırıcılarının ortalaması birbirine yakın olmasına rağmen, aralarındaki fark standart sapmaya göre fazla olduğundan (dolayısıyla olasılığı çok küçük olduğundan) sıfır denencesi reddedilmiştir. Ama (kesinlik, anma) uzayında (Şekil 4.6(b)) her iki sınıflandırıcı birbirine yakın olduğundan MultiPR sınaması sıfır denencesini reddetmemiştir. Kesinlik hesaplanırken, toplam artıya atanan örnek sayısını a' değerine bölüyoruz, yanlış artı değerini hesaplarken ise toplam eksi e değerine bölüyoruz; eğer e değeri a' değerinden daha büyük olursa yanlış artı değerinin değişintisi küçülmekte ve fark istatistiksel olarak anlamlı olmaktadır.

Bunu Şekil 4.6(d) ile 4.6(e)'yi karşılaştırarak da görebiliyoruz: (d)'de fark yöneylerinin ortalaması (0, 0)'a uzak olup sıfır denencesini reddediyor, (e)'de ise fark yöneylerinin ortalaması (0, 0)'a çok yakın olup sıfır denencesini reddedemiyoruz.

Birden çok sınıflandırıcının karşılaştırılması

Deneylerimizin üçüncü bölümünde, tek değişkenli ve çok değişkenli sınamaları kullanarak $L > 2$ sınıflandırıcıyı karşılaştırdık. Tek değişkenli durumda, ANOVA sıfır denencesini reddettiğinde, $L(L-1)/2$ tek değişkenli artçı istatistiksel sınamalar yapılarak hangi ikililer arasında fark olduğunu ve hangi sınıflandırıcı hiziplerinin oluştuğunu görebiliriz (Bir sınıflandırıcı hizibindeki sınıflandırıcıların herhangi ikisi arasında istatistiksel fark yoktur).

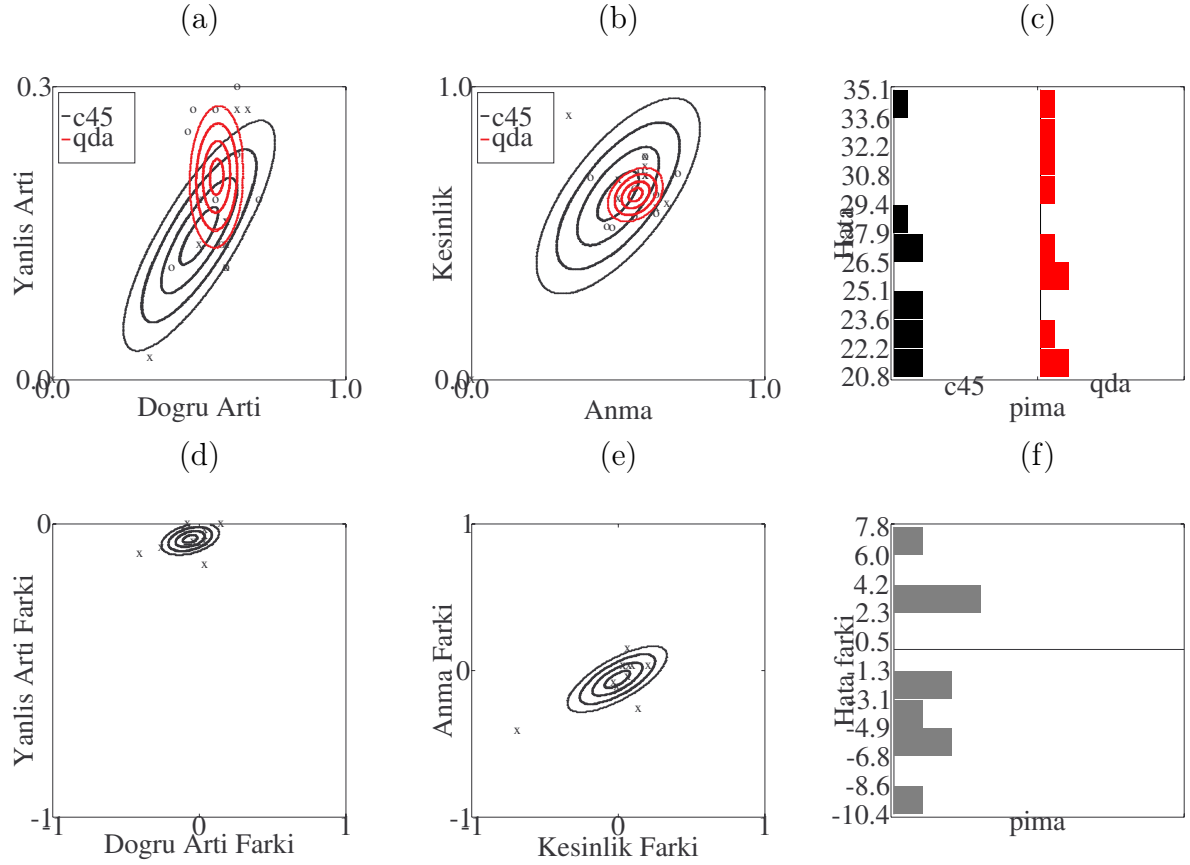
Şekil 4.7'de *breast* veri kümesi üzerinde sınıflandırıcıların başarımlarını görebiliyoruz. Hem ANOVA, hem MANOVA sınamaları sıfır denencesini reddetmektedir. Artçı sınama sonuçlarına göre, tek değişkenli sınama dört sınıflandırıcıdan oluşan (knn, lda, qda, svm) kümesini bulmakta olup, çok değişkenli sınamaların her ikisi de (MultiTF ve MultiPR) üç sınıflandırıcıdan oluşan (knn, lda, svm) hizibini bulmaktadırlar.



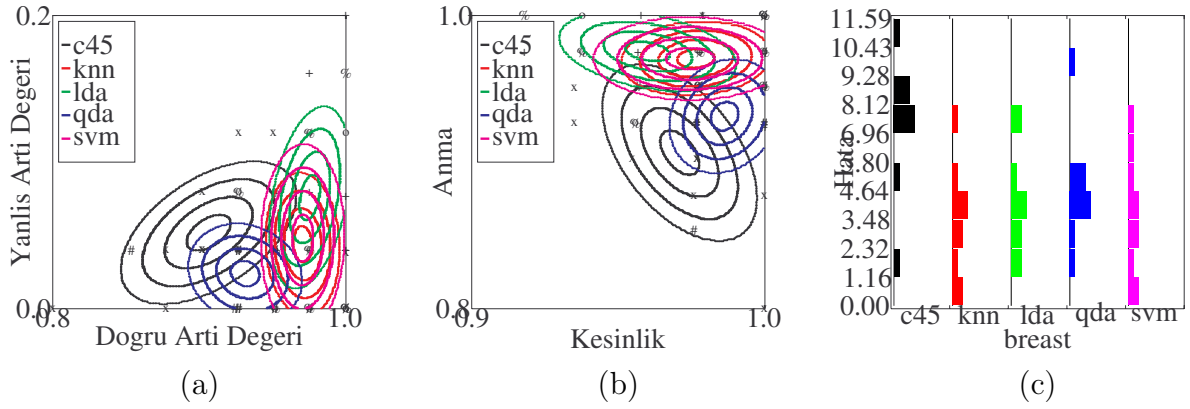
Şekil 4.5: Tek değişkenli istatistiksel sınamanın sıfır denencesini reddedemediği fakat çok değişkenli MultiTF sınamasının sıfır denencesini reddettiği bir örnek.

2×2 karıştırma dizeyi kullanarak karşılaştırma

(Doğru artı, yanlış artı) ya da (kesinlik, anma) ölçütlerini kullanmak yerine, tüm 2×2 karıştırma dizeyini kullanarak aynı çok değişkenli sınamaları dört boyutta da uygulayabiliriz. Dikkat edersek, dizey dört değer içerdiği halde, a ve e değerleri sabit olduğundan gerçek boyut sayısınun iki olduğunu ve dört boyuta geçmenin gereksiz olduğunu anlarız. Zaten, ikili karşılaştırma deneylerinde, 2740 durumdan 2582 tanesinde 2×2 karıştırma dizeyinin kertesinin iki olduğunu gördük; sadece 98 durumda kerte birdir, ki bu durum doğru artı/doğru eksi değerinin bütün katlarda aynı olduğu durumdur.



Şekil 4.6: MultiTF çok değişkenli sınamasının sıfır denencesini reddettiği MultiPR çok değişkenli sınamasınını ise sıfır denencesini reddedemediği bir örnek.



Şekil 4.7: Breast veri kümesi üzerinde 5 sınıflandırıcının karşılaştırılması

4.4 Dağılımdan Bağımsız Sınamalar

4.4.1 Sınamaların Sınanması

Yapay Veri Kümesi Üzerinde Deneyler

Birinci deney düzeneğinde amaç, Kolmogorov-Smirnov ve Wald-Wolfowitz sınamalarının yapay veri kümeleri üzerinde karşılaştırmaktır. Bu amaçla, p boyutlu, eşdeğişinti dizeyleri

I ve merkezleri $(\frac{0.2k}{\sqrt{p}}, \frac{0.2k}{\sqrt{p}}, \ldots, \frac{0.2k}{\sqrt{p}})$ olan normal dağılımdan gelen 1000 veri kümesi oluşturuldu. Ayrıca kontrol amaçlı yine p boyutlu, eşdeğişinti dizeyi **I** ve merkezi (0, 0, ..., 0) olan ve normal dağılımdan gelen 1000 veri kümesi daha oluşturuldu. Burada k sayısı merkezden uzaklığı gösteren parametre olup 1'den 10'a kadar değiştirilmektedir. k arttıkça veri kümeleri kontrol kümelerinden uzaklaşacak ve bu iki dağılımın aynı olduğunu söyleyen denence her iki sınama tarafından daha çok reddedilecektir.

Şekil 4.8'de sol sütunda Kolmogorov-Smirnov sınamasının, sağ sütunda Wald-Wolfowitz sınamasının sonuçları görülmektedir. İstisnaları olmakla birlikte, genel olarak boyut (p) arttıkça sınamaların sıfır denencesini (eşitliği) daha az reddettiklerini, yani farkı yakalamakta daha kötü olduklarını (yüksek tip 2 hatası) görüyoruz. Görüldüğü gibi veri kümesinin büyüklüğü (N) arttıkça, sınamalar daha iyi sonuç vermekte ve boyutlar (p) arasındaki fark azalmaktadır. Wald-Wolfowitz sınamasının boyut sayısından daha az etkilendiği söylenebilir.

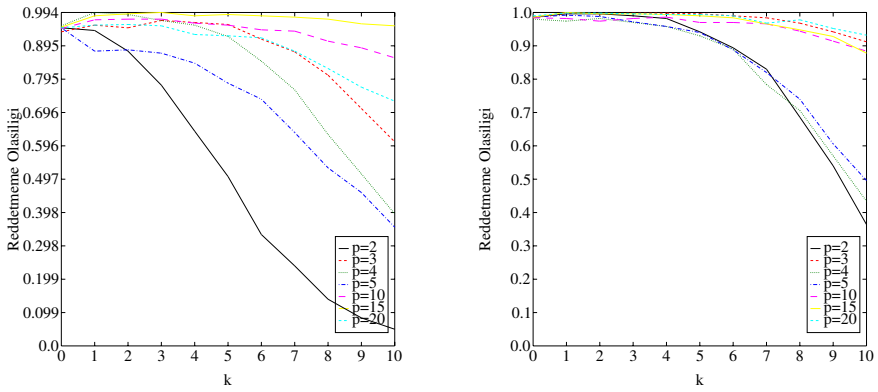
Gerçek Veri Kümesi Üzerinde Deneyler

İkinci deney düzeneğinde amaç, Kolmogorov-Smirnov ve Wald-Wolfowitz sınamalarının bir veri kümesi üzerinde eğitilmiş sınıflandırıcılarla karşılaştırmaktır. Bu amaçla UCI veri bankasından *pendigits* adlı veri kümesi alınmış, 10 sınıflı bu veri kümesinden, (karıştırma dizeyi üzerinden çıkarılan doğru artı değeri, yanlış artı değeri, çağırılabilirlik gibi ölçütler iki sınıflı problemler üzerinde tanımlı olduğu için) yalnızca 1 ve 7 sınıfları çekilerek iki sınıflı yeni bir veri kümesi elde edilmiştir. İki sınıflı veri kümesinin her boyutu belirli bir $p = 0.1k$ olasılığı ile bozularak yeni veri kümeleri oluşturulmuş ve gerek bozulmamış, gerekse bozulmuş veri kümeleri üzerinde en yakın 1-komşu algoritması 5 kere 2 kat yöntemiyle 100 kere çalıştırılmıştır. Çalıştırılan en yakın 1-komşu algoritmalarının çok boyutlu başarımları hem Kolmogorov-Smirnov hem Wald-Wolfowitz sınamaları ile karşılaştırılmıştır. Burada k sayısı bozulma olasılığını değiştiren parametre olduğundan k arttıkça bozulmuş veri kümeleri bozulmamış veri kümelerinden uzaklaşacak ve bu veri kümeleri üzerinde çalışan en yakın 1-komşu algoritmasının başarımının aynı olduğunu söyleyen denence her iki sınama tarafından daha çok reddedilecektir.

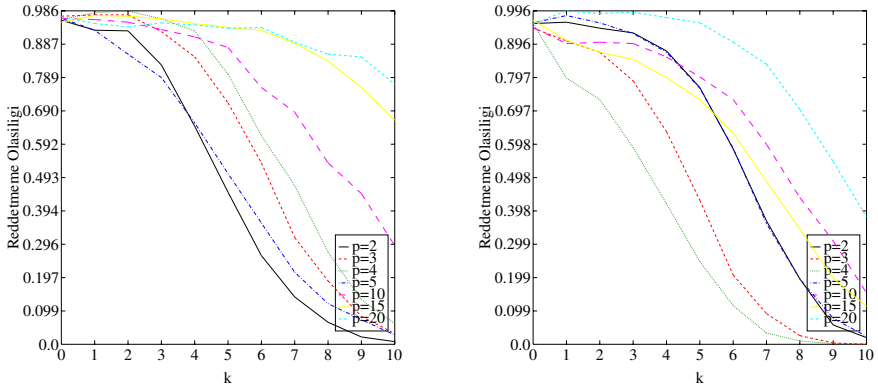
Şekil 4.9'da sol sütunda Kolmogorov-Smirnov sınamasının, sağ sütunda da Wald-Wolfowitz sınamasının sonuçları görülmektedir. Görüldüğü gibi sınamalarda kullanılan ölçütler değiştirildiğinde sınamaların sıfır denencesini reddetme oranları pek fazla değişmemektedir. Özellikle Wald-Wolfowitz sınamasında bu böyledir. Kolmogorov-Smirnov sınamasında ise karıştırma dizeyi ile ikili ölçütler arasında farklılıklar göze çarpmaktadır. Son olarak görüyoruz ki k parametresi arttıkça Kolmogorov-Smirnov sınaması Wald-Wolfowitz sınamasına göre sıfır denencesini daha hızla reddetmektedir.

4.4.2 Çok Değişkenli ve Tek Değişkenli Sınamaların Karşılaştırılması

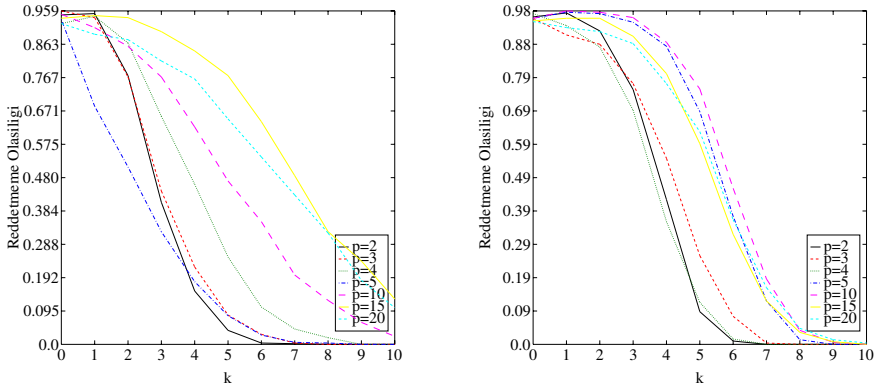
Bu kısımda hata ölçütünü kullanan tek değişkenli k-kat eşli t sınamasını önerdiğimiz iki boyutlu (doğru artı, yanlış artı) ölçütlerini kullanan çok değişkenli sınama ile *breast* veri kümesi üstünde karşılaştırdık. Şekil 4.10'da tek değişkenli sınamanın sıfır denencesini reddedemediği fakat çok değişkenli sınamanın sıfır denencesini reddettiği bir durumu görüyoruz.



$N = 10$



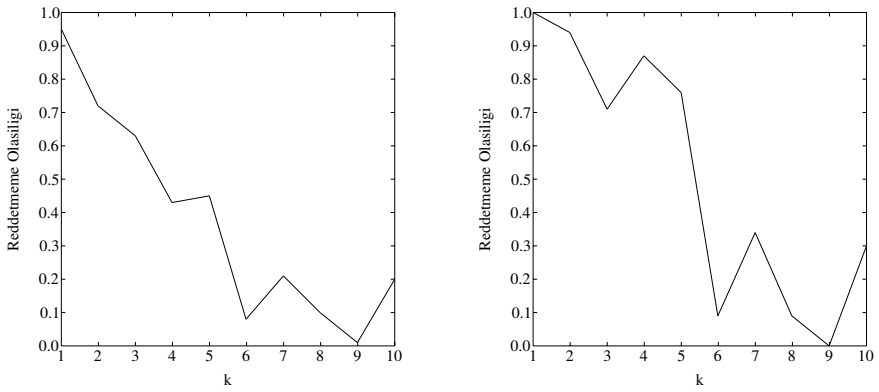
$N = 30$



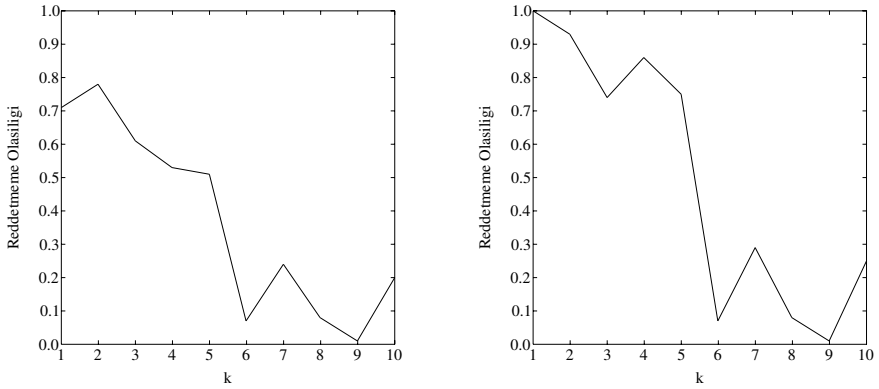
$N = 100$

Şekil 4.8: Örneklem büyüklüğü (N) ve boyut sayısına (p) göre (sol) Kolmogorov-Smirnov, (sağ) Wald-Wolfowitz sınamalarının eşitliği reddetmeme olasılıkları.

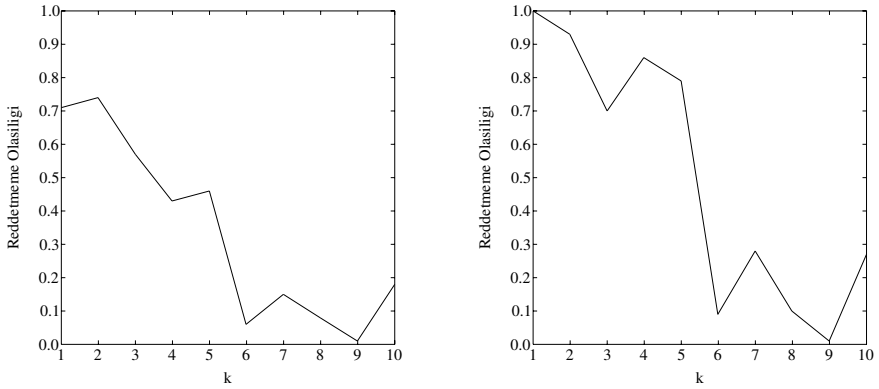
Şekil 4.10(a)'da lda algoritmasının yanlış artı değerinin yüksek, qda'in doğru artı değerinin düşük olduğunu görüyoruz. Şekil 4.10(d)'de ise ikili farkların eşdeğişinti dizeyinin çizimi görülmektedir. Merkez (0,0) noktasından uzak olduğundan Hotelling T^2 sınaması sıfır denencesini reddetmektedir. Şekil 4.10(b)'deki ağaçta lda sıra değerlerinin qda sıra de-



4 boyutlu: (doğru artı, doğru eksi, yanlış artı, yanlış eksi)



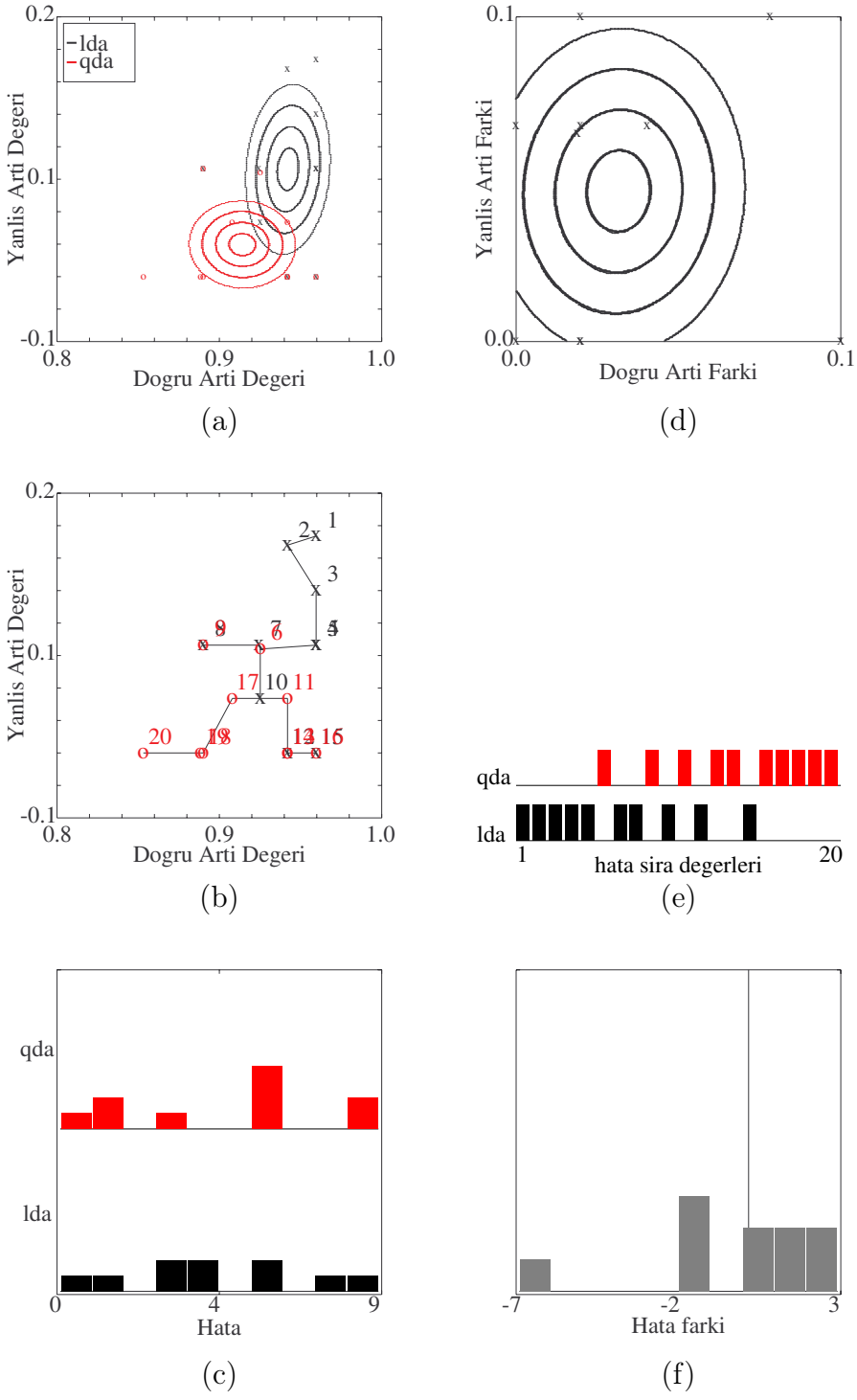
2 boyutlu: (doğru artı, yanlış artı)



2 boyutlu: (kesinlik, çağırılabilirlik)

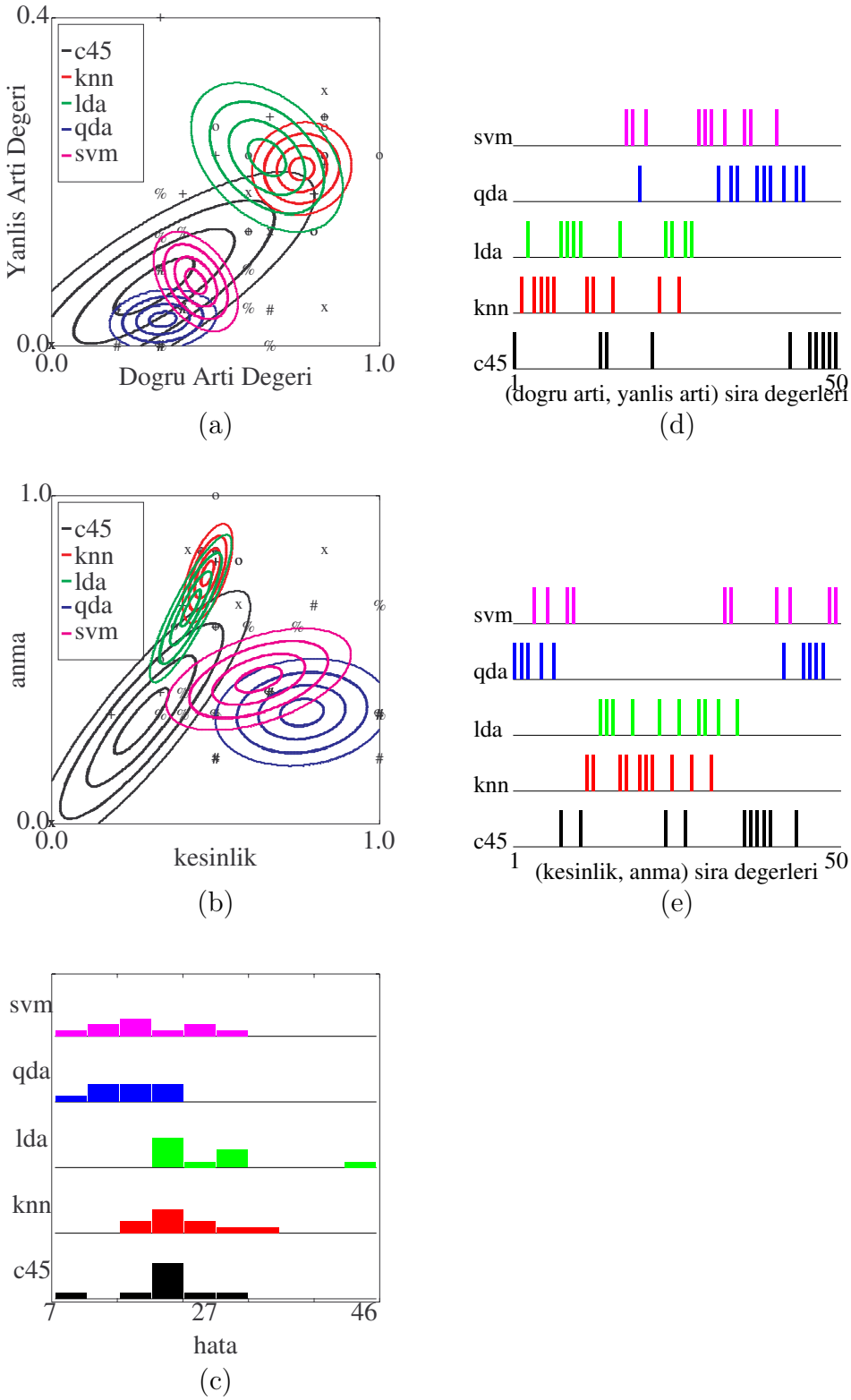
Şekil 4.9: En yakın 1-komşu algoritması kullanılarak *pendigits* verisi üzerinde bozulmuş ve bozulmamış verilerle eğitilmiş sınıflandırıcıların karşılaştırılması. (Sol) Kolmogorov-Smirnov, (sağ) Wald-Wolfowitz sınama sonuçları.

ğerlerinden ayrık olduğunu, bu yüzden Şekil 4.10(e)'de görüldüğü gibi lda algoritmasının qda algoritmasına göre daha düşük sıra numaraları aldığını ve dolayısıyla Kolmogorov-Smirnov sınamasının sıfır denencesini reddettiğini görüyoruz. Şekil 4.10(c)'de ise her iki



Şekil 4.10: Hata kullanan tek değişkenli sınama ve (doğru artı, yanlış artı) kullanan iki değişkenli dağılıma bağlı ve dağılımdan bağımsız sınama sonuçlarının karşılaştırılması.

algoritmanın hata değerleri görülmektedir. Bu veri kümesi üzerinde sadece hata değerlerine göre karşılaştırma yapıldığında lda ve qda algoritmalarının hata farkları (bkz. Şekil 4.10(f)) 0 etrafında kümelenmekte ve k-kat t sınaması sıfır denencesini reddetmemektedir. Yani toplam hataya göre iki algoritma arasındaki fark bulunamazken iki boyutlu sınama



Şekil 4.11: Beş algoritmanın karşılaştırılma sonuçları.

bu farkı yakalayabilmektedir. Bu örnekte de görüldüğü gibi çok değişkenli sınamalar tek değişkenli sınamaların yakalayamayadığı farkları bulabildiği için güçleri daha yüksektir,

bir başka deyişle tip 2 hataları daha düşüktür (Yıldız ve diğerleri, 2011).

4.4.3 İkiden Çok Algoritmanın Karşılaştırılması

Bu kısımda ikiden çok algoritmayı çok değişkenli sınama ile karşılaştırdık. Şekil 4.11'de *spect* veri kümesi üzerinde beş algoritmanın sonuçları gösterilmektedir. Şekil 4.11(a) ve (d)'de (doğru artı, yanlış artı) ölçütleriyle dağılıma bağlı ve dağılımdan bağımsız karşılaştırma yapılırken, Şekil 4.11(b) ve (e)'de (kesinlik, anma) ölçütlerine göre karşılaştırma yapılmaktadır — (d) ve (e)'de ağaçlar gösterilmemiş, yalnızca bu ağaçlardan çıkarılmış sıra numaraları gösterilmiştir. Şekil 4.11(c)'de ise hata ölçütüne göre karşılaştırma yapılmaktadır. Anova ve Kruskal-Wallis sınamaları beş algoritmanın başarımının eşit olduğunu söyleyen sıfır denencesini reddetmektedir. Ardından yapılan ikili sınamalarla arasında fark olmayan öğelerden oluşan hizipler bulunmuştur. Dağılımdan bağımsız sınamaya göre (doğru artı, yanlış artı) ölçütleri üzerinde (c4.5, qda) ve (lda, knn) hizipleri elde edilmektedir. (Kesinlik, anma) ölçütleri üzerinde de (qda, svm) ile (lda, knn) hizipleri elde edilmektedir. Hiziplerdeki farklar, farklı ölçütlerin farklı özellikleri ölçmesi yüzündendir.

4.5 Farklı Yitimler İçin Sınamalar

4.5.1 Menteşe Yitim Tabanlı Sınama

UCI veri tabanından aldığımız 11 veri kümesi üzerinde geliştirdiğimiz menteşe yitim tabanlı sınamaları denedik ve hata tabanlı sınamayla karşılaştırdık. Doğrusal, ikinci ve üçüncü dereceden çokterimli ve Gauss çekirdekli dört farklı destek yöney makinesi kullandık. Bütün çekirdeklerin değerleri normalleştirildi. 10-kat çapraz geçerleme kullandık ve bütün sınamalarda $\alpha = 0.05$ alındı.

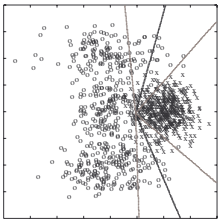
Normallik Sınaması

Menteşe yitim değerleri üzerinde dağılıma bağlı t sınaması uygulayabilmek için normallik varsayımının geçerli olduğundan emin olmalıyız. Bütün çekirdekler için normallik sınamasının (Mardia, 1970) tek değişkenli sürümünü kullandık ve sınamanın sıfır denencesini kaç kere reddettiğini saydık. Her veri kümesi üzerinde, 10-kat deneyimizi 10 kere tekrarladık; Tablo 4.2'deki sayılar $11 \times 10 = 110$ deney üzerindeki oranları göstermektedir. Gördüğümüz gibi, menteşe yitimi için sıfır denencesinin reddetme oranı, hata (0/1 yitimi) için sıfır denencesinin reddetme oranına çok yakındır. Bu da menteşe yitimi değerlerini karşılaştırırken dağılıma bağlı t sınamasını uygulayabileceğimizi gösterir. Aslında çekirdek türünün sonuçlar üzerinde önemli bir etken olduğu hissedilmektedir; bu, ileride üzerinde çalışacağımız bir konu olabilir.

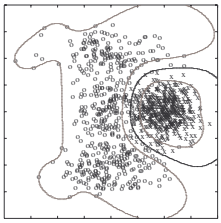
Yapay Veri Üzerinde Karşılaştırma

Şekil 4.12'de iki boyutlu yapay bir veri üzerinde ayırtaç ve kenar paylarını görmekteyiz. Doğrusal ve Gauss çekirdeklerinin karşılaştırıldığı bu durumda, hata sınaması eşitliği reddetmezken menteşe yitimine dayalı sınama, iki çekirdeğin başarımları arasında istatistiksel olarak anlamlı bir fark bulmaktadır. Şekil 4.12(a) ve (b)'de gördüğümüz gibi, iki çekirdek sınır açısından farklı olmayan ayırtaçlar bulmalarına rağmen kenar payları oldukça farklıdır.

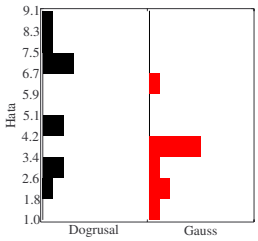
(a) Doğrusal Çekirdek



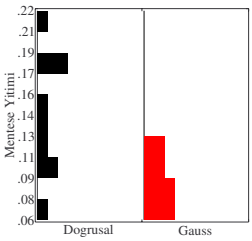
(b) Gauss Çekirdeği



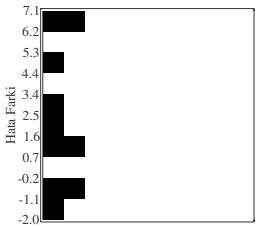
(c) Hata çubukçizitleri



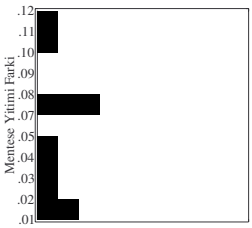
(d) Menteşe yitimi çubukçizitleri



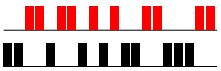
(e) Hata farkı çubukçiziti



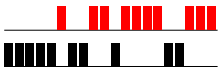
(f) Menteşe yitimi farkı çubukçiziti



(g) Hata sıra değerleri



(h) Menteşe yitimi sıra değerleri



Şekil 4.12: Doğrusal ve Gauss çekirdeklerini kullanan destek yöney makinelerinin yapay veri üzerinde karşılaştırılması.

Tablo 4.2: Farklı çekirdek tiplerine göre hata ve menteşe yitim değerlerinin normallik sınamasını ret oranları.

Çekirdek	Hata	Menteşe Yitimi
Doğrusal	0.136	0.109
İkinci Derece	0.009	0.018
Üçüncü Derece	0.009	0.000
Gauss	0.055	0.045

Şekil 4.12(c)'de hata çubukçizitlerinin üst üste geldiğini, Şekil 4.12(e)'de de eşli farklarının çubukçizitinde ortalamalarının 0'dan uzak olmadığını görmekteyiz. Bu yüzden dağılıma bağlı hata sınaması ortalamaların eşitliğini reddetmemektedir. Benzer biçimde Şekil 4.12(d) ve (f)'ye bakarsak, menteşe yitimine dayalı dağılıma bağlı sınamasının eşitliği neden reddetiğini görebiliriz: Şekil 4.12(f)'de tüm farklar artıdır. Benzer biçimde Şekil 4.12(g)'de, sıra numaraları hata için daha üst üste iken menteşe yitimi için Şekil 4.12(h)'de görüldüğü gibi değerlerin daha az üst üste bindiğini görmekteyiz. Bu yüzden dağılıma bağlı olmayan sınama da hataya göre reddetmezken menteşe yitimine göre reddetmektedir.

Genel Sonuçlar

Bütün çekirdekler ve bütün veri kümeleri için, hem hata hem menteşe yitimi için ikili karşılaştırmalar yaptık ve sınama sonuçlarını karşılaştırdık. 11 veri kümesinde, her biri için 10 bağımsız deney ve dört çekirdekle $4 \times 3 \ / \ 2 = 6$ farklı ikili karşılaştırma olduğundan toplam 660 karşılaştırma yaptık.

Tablo 4.3'te, hata ve menteşe yitimine dayalı dağılıma bağlı sınamaların yüzde (26.4 + 33.3=) 59.7 oranla aynı kararı verdiklerini gözlüyoruz. Farklı karar verdiklerinde, yüzde 33.6 oranla menteşe yitimine dayalı sınamanın istatistiksel bir fark bulup eşitliği reddettiğini, buna karşılık hataya dayalı sınamanın eşitliği reddetmediğini, yüzde 6.7 oranla ise tersi durumu gözlüyoruz. Bu da gösteriyor ki, durumların yaklaşık olarak üçte birinde menteşe yitimi cinsinden sınıflandırıcılar arasında bir fark var, ama eğer karşılaştırmalarda hata yitimi kullanılırsa bu fark anlaşılamıyor (Yıldız ve Alpaydın, 2012).

Dağılıma bağlı olmayan sınama kullanıldığında Tablo 4.4'teki sonuçları elde ettik. Dağılıma bağlı olmayan sınama daha geniş aralıklar tanımlamakta ve genel olarak daha az reddetmektedir. Her iki sınamanın da kabul ettiği durum sayısı bu kez yüzde 45.8'dir. Deneylerin yüzde 14.5'inde menteşe yitimine dayalı sınama eşitliği redderken hataya dayalı sınama reddetmez. Bu da yine menteşe yitimine dayalı sınamanın hataya dayalı sınamanın ayırt edemediği durumları ayırt edebildiğini göstermektedir.

Örnek Deney 1

Hata ve menteşe yitimine dayalı sınamaların aynı karar verdikleri duruma bir örnek olarak, *mammographic* veri kümesi üzerinde ikinci dereceden çokterimli ve Gauss çekirdeklerinin karşılaştırma sonuçlarını Şekil 4.13'te veriyoruz. Şekil 4.13(a) ve (b)'de gördüğümüz üzere, hata ve menteşe çubukçizitleri fazla üst üste gelmemekte ve her ikisi de ortalamaların eşit olduğunu iddia eden sıfır denencesini reddetmektedir. Hata ve menteşe sıra numaraları da yine üst üste gelmemekte (Şekil 4.13(c) ve (d)), dolayısıyla dağılıma bağlı olmayan

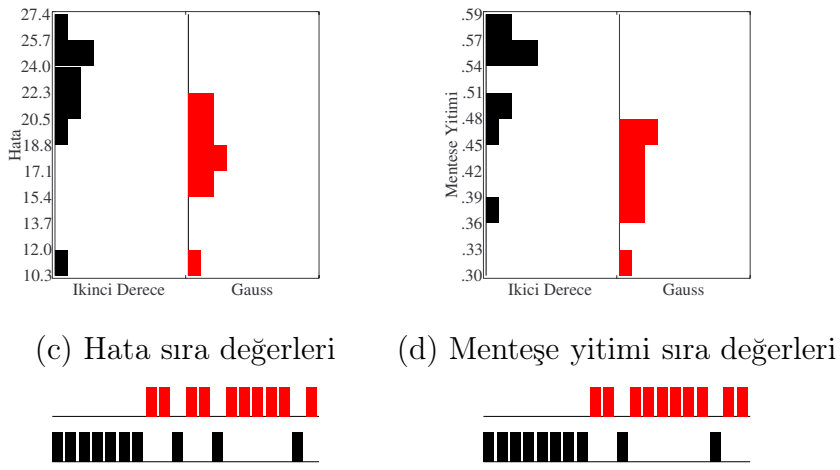
Tablo 4.3: Hata ve menteşe yitimine dayalı dağılıma bağlı sınamanın aynı/farklı karar verme yüzdeleri

	Menteşe Yitimi	
Hata	Kabul	Ret
Kabul	26.4	33.6
Ret	6.7	33.3

Tablo 4.4: Hata ve menteşe yitimine dayalı dağılıma bağlı olmayan sınamanın aynı/farklı karar verme yüzdeleri

	Menteşe Yitimi	
Hata	Kabul	Ret
Kabul	45.8	14.5
Ret	7.1	32.6

(a) Hata çubukçizitleri (b) Menteşe yitimi çubukçizitleri



(c) Hata sıra değerleri (d) Menteşe yitimi sıra değerleri

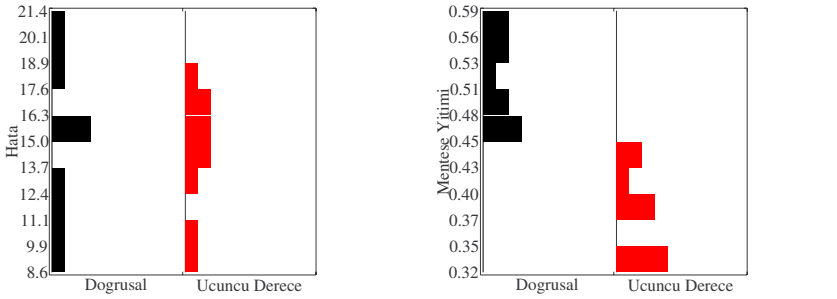
Şekil 4.13: Mammographic veri kümesi üzerinde ikinci dereceden ve Gauss çekirdeklerinin karşılaştırılması.

sınama da sıfır denencesini reddetmektedir.

Örnek Deney 2

Hata ve menteşe yitimine dayalı sınamaların farklı karar verdikleri duruma bir örnek olarak, *credit* veri kümesi üzerinde doğrusal ve üçüncü dereceden çokterimli çekirdeklerin karşılaştırmalarını Şekil 4.14'te veriyoruz. Sınıflandırıcılar hata cinsinden istatistiksel olarak anlamlı farklı olmadıkları halde menteşe yitimi cinsinden anlamlı olarak farklıdır. Hataya dayalı dağılıma bağlı sınama, Şekil 4.14(a)'da gördüğümüz üzere hata dağılımları üst üste geldiği için eşitlik denencesini reddetmezken, menteşe yitimine dayalı dağılıma

(a) Hata çubukçizitleri (b) Menteşe yitimi çubukçizitleri



(c) Hata sıra değerleri (d) Menteşe yitimi sıra değerleri

Şekil 4.14: Credit veri kümesi üzerinde doğrusal ve üçüncü dereceden öokterimli çekirdeklerin karşılaştırılması.

bağlı sınama Şekil 4.14(b)'de gördüğümüz üzere menteşe yitimi dağılımları üst üste gelmediği için ortalamaların eşit olduğunu iddia eden sıfır denencesini reddetmektedir. Hataya dayalı dağılıma bağlı olmayan sınama da sıra numaraları üst üste geldiğinden reddetmezken (Şekil 4.14(c)), menteşe yitimi cinsinden sıra numaraları ayrıktır ve dolayısıyla menteşe yitimine dayalı dağılıma bağlı olmayan sınama sıfır denencesini reddetmektedir (Şekil 4.14(d)).

4.5.2 ϵ Duyarlı Yitim Tabanlı Sınama

UCI veri tabanından aldığımız 9 veri kümesi üzerinde geliştirdiğimiz ϵ-duyarlı yitim tabanlı sınamaları denedik ve ortalama kare hata tabanlı sınamayla karşılaştırdık. Yine doğrusal, ikinci ve üçüncü dereceden çokterimli olmak üzere üç farklı destek yöney makinesi kullandık.

Normallik Sınaması

Her veri kümesi üzerinde, 10-kat deneyimizi 10 kere tekrarladık; Tablo 4.5'teki sayılar $9 \times 10 = 90$ deney üzerindeki oranları göstermektedir. Gördüğümüz gibi, ϵ-duyarlı yitim için sıfır denencesinin reddetme oranı ortalama kare hata yitimi için sıfır denencesinin reddetme oranına çok yakındır. Bu da ϵ-duyarlı yitim değerlerini karşılaştırırken dağılıma bağlı t sınamasını uygulayabileceğimizi gösterir. Burada da önemli bir etken olduğu hissedilmektedir.

Genel Sonuçlar

Bütün çekirdekler ve bütün veri kümeleri için, hem kare hata hem ϵ-duyarlı yitim için ikili karşılaştırmalar yaptık ve sınama sonuçlarını karşılaştırdık. 9 veri kümesinde, her biri için 10 bağımsız deney ve üç çekirdekle $3 \times 2 \ / \ 2 = 3$ farklı ikili karşılaştırma olduğundan toplam 270 karşılaştırma yaptık.

Tablo 4.5: Farklı çekirdek tiplerine göre kare hata ve ϵ-duyarlı yitim değerlerinin normallik sınamasını ret oranları

Çekirdek	Kare hata	ϵ-duyarlı yitim
Doğrusal	0.000	0.000
İkinci Derece	0.078	0.067
Üçüncü Derece	0.033	0.022

Tablo 4.6'da, kare hata ve ϵ-duyarlı yitime dayalı dağılıma bağlı sınamaların yüzde (7.4 + 86.7=) 94.1 oranında aynı kararı verdiklerini gözlüyoruz. Farklı karar verdiklerinde, yalnızca yüzde 0.7'sinde ϵ-duyarlı yitime dayalı sınamanın istatistiksel olarak anlamlı bir fark bulup eşitliği reddettiğini, buna karşılık kare hataya dayalı sınamanın eşitliği reddetmediğini, yüzde 5.2'sinde ise bunun tersi bir durumu gözlüyoruz. Bu da gösteriyor ki, durumların yaklaşık olarak yüzde birinde ϵ-duyarlı yitim cinsinden bağlanım algoritmaları arasında bir fark var, ama eğer karşılaştırmalarda kare hata kullanılırsa bu fark anlaşılamıyor (Yıldız ve Alpaydın, 2012).

Dağılıma bağlı olmayan sınama kullanıldığında Tablo 4.7'deki sonuçları elde ettik. Dağılıma bağlı olmayan sınama daha geniş aralıklar tanımlamakta ve genel olarak daha az reddetmektedir. Her iki sınamanın da aynı karar verdiği durum sayısı bu kez yüzde 78.5'tir. Deneylerin yüzde 10.7'sinde ϵ-duyarlı yitime dayalı sınama eşitliği redderken kare hataya dayalı sınama reddetmez. Bu da yine ϵ-duyarlı yitime dayalı sınamanın kare hataya dayalı sınamanın ayırt edemediği durumları ayırt edebildiğini göstermektedir.

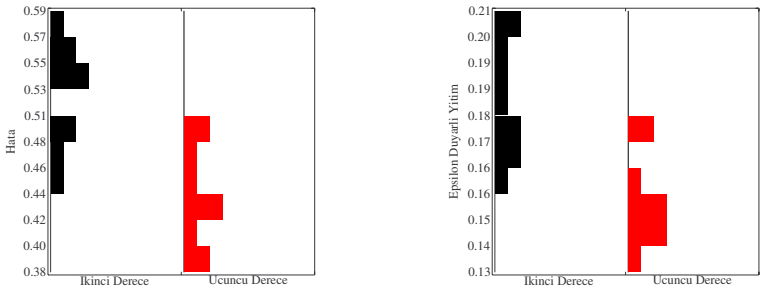
Tablo 4.6: Kare hata ve ϵ-duyarlı yitime dayalı dağılıma bağlı sınamanın aynı/farklı karar verme yüzdeleri

	ϵ-duyarlı Yitim	
Kare hata	Kabul	Ret
Kabul	7.4	0.7
Ret	5.2	86.7

Tablo 4.7: Kare hata ve ϵ-duyarlı yitime dayalı dağılıma bağlı olmayan sınamanın aynı/farklı karar verme yüzdeleri

	ϵ-duyarlı Yitim	
Kare hata	Kabul	Ret
Kabul	4.4	10.7
Ret	10.7	74.1

(a) Kare hata çubukçizitleri (b) ϵ-duyarlı yitim çubukçizitleri



(c) Kare hata sıra değerleri (d) ϵ-duyarlı yitim sıra değerleri

Şekil 4.15: *Abalone* veri kümesi üzerinde ikinci dereceden ve üçüncü dereceden çekirdeklerin karşılaştırılması.

Örnek Deney 1

Kare hata ve ϵ-duyarlı yitime dayalı sınamaların aynı karar verdikleri duruma bir örnek olarak, *abalone* veri kümesi üzerinde ikinci dereceden çokterimli ve üçüncü dereceden çokterimli çekirdeklerinin karşılaştırmalarını Şekil 4.15'te veriyoruz. Şekil 4.15(a) ve (b)'de gördüğümüz üzere, kare hata ve ϵ-duyarlı yitim çubukçizitleri fazla üst üste gelmemekte ve her ikisi de ortalamaların eşit olduğunu iddia eden sıfır denencesini reddetmektedir. Kare hata ve ϵ-duyarlı yitim sıra numaraları da yine üst üste gelmemekte (Şekil 4.15(c) ve (d)), dolayısıyla dağılıma bağlı olmayan sınama da sıfır denencesini reddetmektedir.

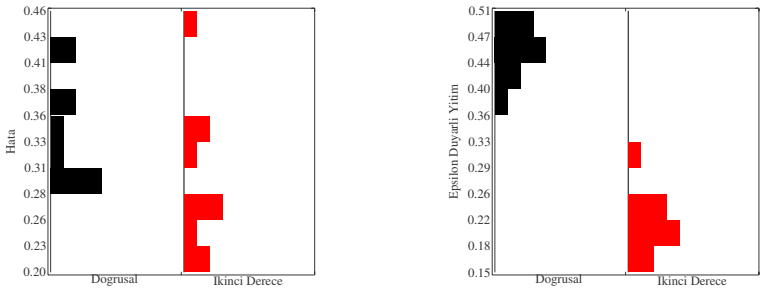
Örnek Deney 2

Kare hata ve ϵ-duyarlı yitime dayalı sınamaların farklı karar verdikleri duruma bir örnek olarak *concrete* veri kümesi üzerinde doğrusal ve ikinci dereceden çokterimli çekirdeklerin karşılaştırmalarını Şekil 4.16'da veriyoruz. Sınıflandırıcılar kare hata cinsinden istatistiksel olarak farklı olmadıkları halde ϵ-duyarlı yitim cinsinden istatistiksel olarak farklıdır. Kare hataya dayalı dağılıma bağlı sınama Şekil 4.16(a)'da gördüğümüz üzere kare hata dağılımları üst üste geldiği için eşitlik denencesini reddetmezken, ϵ-duyarlı yitime dayalı dağılıma bağlı sınama Şekil 4.16(b)'de gördüğümüz üzere ϵ-duyarlı yitim dağılımları üst üste gelmediği için ortalamaların eşit olduğunu iddia eden sıfır denencesini reddetmektedir. Kare hataya dayalı dağılıma bağlı olmayan sınama da ϵ-duyarlı yitime dayalı dağılıma bağlı olmayan sınama da sıra numaraları ayrıktır ve dolayısıyla her iki dağılıma bağlı olmayan sınama da sıfır denencesini reddetmektedir (Şekil 4.16(c), 4.16(d)).

$L > 2$ Bağlanım Algoritmasının MultiTest ile Karşılaştırılması

Bütün veri kümeleri için üç farklı çekirdeğe dayalı bağlanım algoritmasını, hem kare hata hem de ϵ-duyarlı yitim için karşılaştırdık. 9 veri kümesinde, her 10 bağımsız deney için MultiTest algoritmasının verdiği sıralamaları raporladık.

(a) Kare hata çubukçizitleri (b) ϵ-duyarlı yitim çubukçizitleri



(c) Kare hata sıra değerleri (d) ϵ-duyarlı yitim sıra değerleri

Şekil 4.16: *Concrete* veri kümesi üzerinde doğrusal ve ikinci dereceden çokterimli çekirdeklerin karşılaştırılması.

Tablo 4.8'de, kare hata yitimine dayalı dağılıma bağlı sınama kullanarak uygulanan MultiTest yönteminin 10 bağımsız deneyde ürettiği sıralamalar verilmektedir. Altı veri kümesinde (*abalone*, *add10*, *boston*, *concrete*, *puma8nh*, *puma8nm*) üçüncü dereceden çekirdek ikinci dereceden çekirdekten; ikinci dereceden çekirdekte doğrusal çekirdekten istatistiksel olarak daha iyi sonuç vermektedir. İki veri kümesinde (*puma8nh*, *puma8nm*) ise üçüncü dereceden çekirdek, hem ikinci hem doğrusal çekirdekten istatistiksel olarak daha iyi sonuç verirken, ikinci dereceden çekirdek doğrusal çekirdekten istatistiksel olarak daha iyi değildir. Dolayısıyla bu iki veri kümesinde MultiTest yöntemi 3-1-2 sıralamasını üretmektedir.

Tablo 4.8: Hata yitimine dayalı dağılıma bağlı sınama kullanarak uygulanan MultiTest yönteminin 10 bağımsız deneyde ürettiği sıralamalar (1: doğrusal, 2: ikinci derece, 3: üçüncü derece çekirdek)

	Sıralamalar					
Veri Kümesi	123	132	231	213	312	321
abalone	0	0	1	0	1	8
add10	0	0	0	0	0	10
boston	0	0	0	0	0	10
california	0	0	7	1	0	2
concrete	0	0	0	1	1	8
puma8fh	0	0	0	0	10	0
puma8fm	0	0	0	0	10	0
puma8nh	0	0	0	0	0	10
puma8nm	0	0	0	0	0	10

4.6 Biyoinformatik Yazınındaki Sınıflandırma Deneylerinin Taranması

Yapay öğrenmenin biyoinformatik alanındaki uygulama pratiklerini araştırmak için üç biyoinformatik dergisinde 2010 ve 2011 yıllarında yayımlanan makaleleri taradık ve yazılım, uygulama notları ve özetler hariç tüm makaleleri derledik. Bu makaleler için yapay öğrenme ile ilgili olanlarla, bu makalaler içinden de sınıflandırma ile ilgili olanları belirledik.

Sonuçlar gösterdi ki (İrsoy ve diğerleri, 2012a), iki yıl içinde yayımlanmış bu makalelerden %40'ı yapay öğrenme ile ilgiliyken, yapay öğrenme ile ilgili makalelerinde de %41'i sınıflandırma ile ilgilidir. Bu yüksek yüzdeler biyoinformatik araştırma alanında oldukça yüksek miktarda sınıflandırma yapıldığını ve sınıflandırıcıların başarımlarını karşılaştıran çalışmamızın biyoinformatik alanında yayımlanan makalelerin yaklaşık %16'sını ilgilendirdiğini ortaya koymaktadır.

Sınıflandırma kullanan makalelerden, veri kümesinin özelliklerini (sınıf sayısı ve girdi boyut sayısı), öğrenme yönteminin özelliklerini (boyut azaltma yapılmış/yapılmamış, sınıflandırma algoritması) ve kullanılan istatistiksel yöntembilimi (çapraz geçerleme yöntemi, başarım ölçütleri, istatistiksel sınama) belirledik. Bu makalelerden çıkardığımız sonuçlar genel olarak şöyle özetlenebilir:

- Çoğu sınıflandırma problemi iki sınıflı sınıflandırma problemidir. Bu karıştırma dizeyi tabanlı başarım ölçütlerinin birçok durumda uygulanabildiğini gösterir.

- Beklenildiği gibi, başarı/hata oranı en çok kullanılan başarım ölçütüdür. Artı örnekler üzerinde yoğunlaşılması gerektiğinde, kesinlik ve anma da kullanılmaktadır. Alcı işletim özellikleri eğrisinin altında kalan alan biyoinformatik topluluğunda sıkça kullanılırken, bu değerleri veren makalelerden sadece %51'i gerçek eğrileri göstermektedir. Kesinlik-anma eğrileri daha az sıklıkla kullanılmaktadır.

- Veri kümeleri çoğunlukla 1000 örnekten küçük olup böyle durumlarda veriden hesaplanan istatistiklerin değişkenliği yüksek olmaktadır. Uygun geçerleme ve denence sınamaları belirlemenin önemi burada öne çıkmaktadır.

- Biyoinformatik uygulamaları büyük boyutlu girdiler içermektedir. Makalelerin %20'sinde kullanılan verilerdeki boyut sayısı 10000'den fazladır. Bu sebeple küçük veride yüksek boyut sayısından dolayı ezberleme önemli bir problemdir. Çoğu uygulamada yüksek boyut olduğundan farklı boyut azaltma tekniklerinin uygulanmış olması şaşırtıcı değildir. Beklenildiği gibi boyut sayısı arttıkça, boyut azaltma yöntemlerinin kullanılma oranı da artmaktadır.

- Destek yöney makineleri ile karar ağaçları (çoğunlukla rassal ormanlar) şu anda bilinen en iyi öğrenme algoritmaları olduğundan biyoinformatik uygulamalarında da en sık kullanılan algoritmalardır. Destek yöney makineleri küçük veri kümelerinde iyi çalıştığından ve rassal orman algoritması da yüksek boyutta iyi çalıştığından biyoinformatikte bu algoritmaların seçimi şaşırtıcı değildir. Küçük veri kümelerinde, tekdışarı-bırak kullanılırken, k-kat ve $k_1 \times k_2$ kat çapraz geçerleme deneylerin %70'inde kullanılmaktadır. Bu da biyoinformatik alanında deneylerin birden çok kez tekrarlanması gerektiğinin anlaşıldığını gösterir.

- k-kat çapraz geçerleme ve öteki yeniden örnekleme yöntemleri sıklıkla kullanılmasına rağmen farklı sınıflandırıcıların başarımlarının istatistiksel sınamalarla karşılaştırıldığı durumlar tüm durumların yalnızca %21'ini oluşturmaktadır. Bazı makaleler başarım ölçütlerinin standart sapmalarını gösterip herhangi bir sınama uygulamazken, bazıları tek bir başarım değeri üzerinden bir algoritmanın diğerinden iyi olduğunu söyleyebilmektedirler. Bu da istatistiksel sınamaların biyoinformatik topluluğunda yeterince benimsenmediğini ve bizimki gibi çalışmalara gereksinim duyulduğunu göstermektedir.

4.7 Ayrımcı Dil Modellemede Sıralama Ve Sınıflandırma Yaklaşım Sonuçları

Daha önce derlenmiş ve üzerinde çalışılmış veri kümesi üzerinde altı farklı algoritma uyguladık (Arısoy ve diğerleri, 2012). Algılayıcı (Per), DYM, ve MIRA olmak üzere üç tane sınıflandırıcı ve sıralayıcı algılayıcı (PerRank), Sıralayıcı DYM (SVMrank) ve MIRArank olmak üzere de üç tane sıralayıcı yöntem deniyoruz (Dikici ve diğerleri, 2013).

109E142 nolu proje ile ortak olarak yöntemlerin başarımını sınamak için 10-kat çapraz geçerleme yaparak eşli t-sınaması ve veri kümesinin tamamını kullanarak MAPSSWE sınamasını uyguluyoruz. Bu iki sınama türü de yöntemleri ikili olarak karşılaştırmaktadır. MAPSSWE, iki model arasında cümlelerin hata farkıyla birlikte yazıya dökülen cümlenin de uzunluğunu da hesaba katmaktadır. Yöntemleri uygularken geçerleme kümesinde en iyi ortalama sonuçları veren parametrelerle kurulan modeller kullanılmaktadır.

Tablo 4.9: Sınama veri kümesinde 10–kat sözcük hata oranları

Per	MIRA	SVM	PerRank	SVMrank	MIRArank
21.87 ± 0.06	21.93 ± 0.05	22.19 ± 0.12	21.47 ± 0.05	21.63 ± 0.09	21.81 ± 0.03

Tablo 4.10: Sınama veri kümesinde 10 kat çapraz geçerleme t sınama sonuçları (p değerleri)

	MIRArank	PerRank	Per	SVMrank	SVM
MIRA	2.2×10^{-5}	6×10^{-9}	0.0418	2.4×10^{-5}	3.2×10^{-4}
SVM	9.3×10^{-6}	1.8×10^{-8}	2×10^{-5}	4×10^{-7}	
SVMrank	7.8×10^{-4}	2.1×10^{-3}	1×10^{-4}		
Per	0.9×10^{-4}	3×10^{-8}			
PerRank	1.9×10^{-9}				

Tablo 4.9'da uygulanan 10-kat çapraz geçerleme koşmalarının ortalama ve sapma değerleri verilmektedir. Tablo 4.10'da ise geçerleme sonuçları üzerinde yapılan t sınamalarından elde edilen p değerleri verilmektedir. Verilen p değerlerinden ikili karşılaştırmaların modeller arasında istatistiksel olarak anlamlı farklılıklar bulduğu anlaşılıyor. Tablo 4.10'a dayanarak algoritmaların sözcük hata oranları arasında sıralayabiliriz.

Tablo 4.11'de sınama kümesinde en iyi modellerle elde edilen sınama sonuçları verilmektedir. Bu modeller üzerinde yapılan MAPSSWE sınaması sonuçları ise Tablo 4.12'de gösterilmektedir. Bu sonuçlar önerilen sıralama yöntemlerinin sınıflandırma yöntemlerine göre istatistiksel olarak anlamlı bir üstünlüğe sahip olduğunu gösterir. Bu deneylerde

Tablo 4.11: Sınama kümesinde sözcük hata oranları

Per	MIRA	SVM	PerRank	MIRArank	SVMrank
21.84	21.83	22.08	21.47	21.74	21.61

Tablo 4.12: Sınama veri kümesinde MAPSSWE sınaması sonuçları (p değerleri)

	MIRArank	PerRank	Per	SVMrank	SVM
MIRA	0.638	0.002	0.795	0.121	0.156
SVM	0.014	< 0.001	0.085	0.004	
SVMrank	0.168	0.226	0.131		
Per	0.764	0.003			
PerRank	0.007				

kullanmamış olsak da bu uygulamada dağılımdan bağımsız sınama ve maliyet bilinçli sınamanın kullanılabileceğini görüyoruz.

Sınama sonuçları algoritmaların sıralayıcı sürümlerinin sınıflandırıcı sürümlerine göre daha az sözcük hatası verdiğini göstermiştir. Sakıncası ise sıralama yöntemlerinin eğitim sürelerinin daha uzun olmasıdır, çünkü eğitim süresince daha fazla güncelleme yapılır. Bir başka farklılık ise sıralayıcıların daha fazla öznitelik kullanmasıdır; bunun açıklaması da yine çok daha fazla günceleme yapılmasıdır. Sınıflandırıcılar her cümle için sadece bir kez güncelleme yaptıklarından daha az sayıda özniteliğin katsayısı etkilenir. Sıralayıcıda ise düşük sıradaki denenceler üzerinde de güncelleme yapıldığından ve bu örnekler esas cümleden daha farklı öznitelikler kullanabildiklerinden daha çok sayıda özniteliğin katsayısında değişime neden olur.

Sıralayıcı MIRA ile sıralayıcı algılayıcı benzer olmalarına rağmen sıralayıcı algılayıcı istatistiksel olarak daha üstün bir başarım göstermektedir. Bunun olası nedeni öğrenme oranını hesaplarken fark normu kullanarak yapılan normalleştirme olabilir. Bu yüzden daha küçük öğrenme oranlarıyla yapılan güncellemelerden dolayı düzeltici etkileri azalmaktadır.

Sıralayıcı DYM'nin sıralayıcı algılayıcı kadar başarılı olmasının nedeni algılayıcının kenar payı işlevi kullanarak daha üst sıradaki aday denencelere daha alt sıradakilere göre daha fazla önem vermesidir. DYM için bütün yanlış sıradaki denenceler aynı öneme sahiptir. Sıralayıcı algılayıcı bu yüzden daha az sözcük hatalı adaylara önem vererek sınama veri kümelerinde daha iyi adayların üst sıralarda dönmesini sağlayan, daha ayrımcı bir üstün-düzlem bulur.

İleriki çalışmalar, farklı kenar payı işlevleri kullanmak ve öznitelik seçerek veriyi önceden işlemek yönünde olabilir. Öznitelik seçmek eğitim zamanını azaltacaktır. Kenar payı işlevi ise örnek ikililerine farklı önemler vererek daha farklı ayrımcı üstün-düzlemlere ve böylece daha iyi başarıma sahip olası modellere ulaşmamızı sağlayabilir.

Bölüm 5

Tartışma ve Sonuçlar

5.1 Multi^{2}Test

Yazında, iki sınıflandırma algoritmasını tek bir veri kümesi üzerinde (5×2 cv t sınaması (Dietterich, 1998), 5×2 cv F sınaması (Alpaydın, 1999) gibi) ya da birden çok öğrenme algoritmasını birden çok veri kümesi üzerinde karşılaştıran (Demsar, 2006) istatistiksel sınamalar vardır. Bu sınamalar, ikili karşılaştırmalar yapabilmekte ve aynı hataya sahip algoritma alt kümelerini belirleyebilmekte, fakat kullanıcıya bir sıralama sunamamakta, ve dolayısıyla hangi algoritmanın en iyi olduğunu belirleyememektedirler.

Önceki çalışmamızda (Yıldız ve Alpaydın, 2006), birden çok sınıflandırıcıyı bir veri kümesi üzerinde hem genelleştirme hatası, hem de maliyet cinsinden sıralayabilen MultiTest yöntemini önermiştik. O yöntemde, eşit hataya sahip olan (istatistiksel olarak aralarında fark olmayan) algoritmaları sıralarken maliyeti ek bilgi olarak kullanmıştık. Bu projede geliştirdiğimiz Multi^{2}Test yöntemiyle (önceki çalışmamızın bir genelleştirilmesi olarak) birden çok öğrenme algoritmasının en iyisini seçebiliyor, ya da iyiden kötüye sıralayabiliyoruz (Ulaş ve diğerleri, 2012).

Multi^{2}Test yöntemi, hata üzerinde ikili istatistiksel sınamalar kullanan ve ek olarak, maliyet ölçütü olarak zaman ya da bellek karmaşıklığını gözönüne alan sıralamalar üretebilmektedir. Robotik gibi zaman/bellek kısıtı olan uygulamalarda, ya da biyoinformatik gibi bazı girdilerin pahalı deneylerle elde edilebildiği uygulamalarda Multi^{2}Test algoritmasının, başarım yanında maliyeti de kullanabilmesi sayesinde önemli olacağını düşünüyoruz.

5.2 Başarı Ölçütleri

Yazındaki istatistiksel sınamalar çoğunlukla sınıflandırma hata oranını karşılaştırmada kullanılmakta, dolayısıyla yanlış artı ve yanlış eksilerin eşit kayba sahip olduğunu varsaymaktadır. Başarım eğrileri (alıcı işletim özellikleri eğrisi), kesinlik-anma eğrisi, ve bu eğrilerin altında kalan alansa, farklı kayıp değerleri üzerinde genel başarım hesaplamada kullanılabilir.

Yaptığımız çalışmalar sonucunda, eğri altında kalan alan tabanlı sınamalar ile hata tabanlı sınamanın yüksek oranda aynı karar verdiklerini ama, farklı karar verdiklerinde, eğri altında kalan alan tabanlı sınamaların hata tabanlı sınamalar gibi tek bir kayıp değeri değil de bir çok kayıp değerini gözönünde bulundurduğu için daha güvenilir olduğunu gördük. Ayrıca alıcı işletim özellikleri eğrisi ile kesinlik-anma eğrisine dayanan sınamaların farklı karar verdiği durumlar olduğunu gözlemledik. Bu gibi durumlarda iki sınıflı

bir problemimiz varsa alıcı işletim özellikleri eğrisini, sadece artı sınıfa odaklı bir bilgi erişimi uygulamasındaysa kesinlik-anma eğrisini (ve altında kalan alanı) kullanmak akla yatkındır.

5.3 Çok Değişkenli İstatistiksel Sınamalar

Birden çok ölçütü kullanarak karşılaştırma yapabilen çok değişkenli istatistiksel sınamalar üzerinde çalıştığımız bir başka konuydu. K tane karıştırma dizeyinden (doğru artı, yanlış artı) ya da (kesinlik, anma) gibi iki boyutlu çok değişkenli istatistikleri toplayarak iki değişkenli sınamalar yaptık. Tüm 2×2 karıştırma dizeyini kullanarak ya da başka hata ölçütlerinden oluşan yöneyler kullanarak çok değişkenli başka sınamalar tanımlanabilir. Yapay öğrenme yazınında kullanılan istatistiksel sınamaların tamamı tek değişkenli olup, bildiğimiz kadarıyla, çok değişkenli istatistiksel sınamaların yapay öğrenme algoritmalarının karşılaştırılmasında kullanılması ilktir (Yıldız ve diğerleri, 2011).

Çok değişkenli sınamalardan Hoteling T^2 ve MANOVA, örneklerin normal (Gauss) dağıldığını varsayar. Her ne kadar bu varsayım çoğunlukla doğru olsa da özellikle küçük veri kümelerinde uygun olmayabilir. Dağılımdan bağımsız sınama örnekler için böyle bir dağılım varsaymaz ve bu açıdan uygulama alanı daha geniştir (Kvam ve Vidakovic, 2007); yine de belirtmek gerekir ki varsayımının geçerli olduğu durumda dağılım tabanlı sınama, dağılımdan bağımsız sınamadan tip 1 ve tip 2 hatası açısından daha başarılıdır. Çünkü dağılımdan bağımsız sınama daha geniş güven aralıkları tanımlar ve bu yüzden reddetmeye daha az eğilimlidir.

Dağılımdan bağımsız sınamada en çok kullanılan Kolmogorov - Smirnov ile Wald - Wolfowitz sınamalarıdır (Smirnov, 1939), (Friedman ve Rafsky, 1979). Biz bu sınamaları hem yapay, hem de gerçek veriyle ve kendi uygulama alanımız olan yapay öğrenme yöntemlerinin karşılaştırılmasında denedik. Özet olarak verirsek deneyler sonunda, her iki sınamanın da beklenildiği gibi girdi boyutu arttıkça daha kötü bir başarım gösterdiğini, ama bizim kullandığımız gibi düşük boyutlarda (2 ya da 4) başarıyla kullanılabileceğini gördük.

5.4 Farklı Yitimler İçin Sınamalar

Sınıflandırıcı başarımını ölçerken sıklıkla sınıflandırma hatası kullanılır ama çekirdek tabanlı destek yöney makineleri sınıflandırmada menteşe yitimini küçültmek için eğitilir ve dolayısıyla onların karşılaştırılması 0/1 yitime karşılık gelen hata cinsinden değil aynı kıstas cinsinden yapılmalıdır. Menteşe yitimi 0/1 yitiminden iki açıdan farklıdır ve daha çok bilgi verir:

(1) kenar payına düşen örnekleri yeterince güvenilirlikle sınıflandırılmadıkları için cezalandırır – oysa 0/1 yitimi o örnekleri cezalandırmaz.

(2) sınırın yanlış tarafındaki örneklerin yitimleri 0/1 yitiminde aynıdır ve hep 1'dir. Buna karşılık menteşe yitimi sınıra olan uzaklıklarına göre o örnekleri cezalandırır.

Benzer şekilde bağlanımda çekirdek tabanlı destek yöney makineleri ϵ-duyarlı yitimi küçültmek için eğitilir ve dolayısıyla onların karşılaştırılması ortalama kare yitime karşılık gelen hata cinsinden değil aynı kıstas cinsinden yapılmalıdır. ϵ-duyarlı yitim kare hatasından iki açıdan farklıdır ve daha çok bilgi verir:

(1) ϵ-duyarlı yitimde bağlanım algoritmasının çıktısı ile gerçek çıktı arasındaki fark ϵ sınırının altındaysa cezalandırma yapılmaz – oysa kare hatası o örnekleri cezalandırır.

(2) Kare hatasında örnekler gerçek değere olan uzaklıklarının karesi ile orantılı olarak cezalandırılır. Buna karşılık ϵ-duyarlı yitim sınıra olan uzaklıklarına göre o örnekleri cezalandırır.

Bu özellikler sayesinde menteşe (ϵ-duyarlı) yitimi kullanmak sınıflandırıcı (bağlanım algoritmaları) hakkında daha güvenilirlikle karar vermeyi sağlar ve hata cinsinden ayırt edilemeyen sınıflandırma (bağlanım) algoritmalarının davranışlarını ayırt etmek mümkün olabilir. Nitekim deneylerimizde aldığımız sonuçlar da menteşe (ϵ-duyarlı) yitimine dayalı sınamanın hataya dayalı sınamanın fark bulamadığı durumlarda fark bulabildiğini göstermiştir (Yıldız ve Alpaydın, 2012).

Hata için olduğu gibi menteşe ve ϵ-duyarlı yitim için de normallik varsayımının doğru olduğunu gösterdik ve bu sayede dağılıma dayalı t sınamasını deneylerimizde kullandık. Küçük veri kümeleri için, ayrık örnekler söz konusu olduğunda normallik varsayımı geçerli olmayabilir ve böyle bir durumda sıra değerlerine bağlı Kolmogorov-Smirnov sınaması kullanılabilir.

5.5 Biyoinformatik ve Konuşma Tanıma Uygulamaları

Birçok biyoinformatik uygulamasında, veri ile eğitilmiş algoritmaların başarımlarının ölçülmesi ve karşılaştırılmasında rastsallıktan etkilenmeyen sonuçların çıkarılması önemlidir. Bunun için deneylerin uygun tasarımı ve sonuçların istatistiksel sınamalarla çözümlenmesi, sonuçların istatistiksel olarak anlamlı olması için gereklidir. Bu proje kapsamında, sınıflandırmada kullanılan başarım ölçütlerini, deney tasarımı yöntemlerini ve istatistiksel sınama türlerini inceledik. Ayrıca üç önemli biyoinformatik dergisinde son iki yılda yayımlanmış 1,500'ün üzerinde makaleden derlediğimiz verilerle karşılaştırdık.

Araştırma sonuçlarımız gösterdi ki, deney tasarımının temelleri biyoinformatik alanında anlaşılmış gibi gözükmesine rağmen (örneğin tek bir öğrenme kümesi kullanmak yerine çapraz geçerleme kullanmak, hata yerine başka başarım ölçütleri kullanmak gibi), sınıflandırma algoritmalarını karşılaştırırken istatistiksel sınama kullanan makaleler tüm makalelerin ancak yüzde 21'ini oluşturmaktadır. Çalışmamızın bu parçasında, biyoinformatik yazınında sıklıkla karşılaşılan dört farklı senaryoyu her senaryo için doğru istatistiksel yöntembilimi göstererek inceledik (İrsoy ve diğerleri, 2012a).

Tübitak 109E142 nolu proje ile ortak çalışarak ayrımcı dil modelleme (ADM) için sınıflandırıcı yöntemlerinden farklı olarak sıralayıcıları da kullandık ve istatistiksel olarak sınadık (Dikici ve diğerleri, 2013). Yapılan istatistiksel test sonuçları algoritmaların sıralayıcı sürümlerinin sınıflandırıcı sürümlerine göre daha az sözcük hatası verdiğini göstermiştir. Sakıncası ise sıralama yöntemlerinin eğitim sürelerinin daha uzun olmasıdır. Çünkü eğitim süresince daha fazla güncelleme yapılır.

5.6 Gelecek Çalışmalar

Gelecek projemizde aşağıdaki konularda çalışmayı düşünüyoruz:

- Bu projede iki veya daha fazla sınıflandırma algoritmasını karşılaştırmak için çok değişkenli istatistiksel sınamalar geliştirdik. Dağılımdan bağımsız tek değişkenli sınamaları çok değişkenli sınamalara genelleştirmek için çizge tabanlı algoritmalar kullanılır ama bu algoritmalar sezgisel olup verideki ufak değişikliklerden etkilenebilirler. Çok değişkenli gürbüz sınamalar önermek gelecekte planladığımız çalışmalar arasındadır.

- Destek yöney makinelerinde kullanılan gerek menteşe yitimi gerekse ϵ-duyarlı yitimlerine dayalı denence sınamaları geliştirdik. Algoritmaların kendilerine has ölçütlere göre karşılaştırılması aralarında var olabilecek farklılıkları daha hassas şekilde belirlemeyi sağlamaktadır. Örneğin en yakın komşu algoritmalarının (değişik k değerleri için) başarımlarını gerek sınıflandırma gerekse bağlanım problemleri üzerinde karşılaştırmak için sıralama tabanlı yitim işlevleri önermek gelecekte planladığımız çalışmalar arasındadır.

- Kullandığımız MultiTest ve Multi^{2}Test yöntemleri şu anda dağılıma bağlı eşli sınama kullanır. Bunlarda dağılımdan bağımsız sınama kullanarak özellikle küçük ve gürültülü veri kümelerinde daha güvenilir sınamalar çıkarılabilir.

Kaynakça

Alpaydın, E., Combined 5×2 cv F test for comparing supervised classification learning classifiers, *Neural Computation*, 11, pp: 1975–1982, (1999).

Alpaydın, E., *Introduction to Machine Learning*, The MIT Press, (2010).

Arısoy, E., Saraçlar, M., Roark, B., Shafran, I., Discriminative language modeling with linguistic and statistically derived features, *IEEE Transactions on Audio, Speech, and Language Processing*, 20, pp: 540–550, (2012).

Blake, C., Merz, C., UCI repository of machine learning databases, (2000).

Chang, C.-C., Lin, C.-J., LIBSVM: a library for support vector machines, (2001).

Crammer, K., Singer, Y., Ultraconservative online algorithms for multiclass problems, *Journal of Machine Learning Research*, 3, pp: 951–991, (2003).

Davis, J., Goadrich, M., The relationship between precision-recall and roc curves, *International Conference on Machine learning*, (2006), pp: 233–240.

Demsar, J., Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research*, 7, pp: 1–30, (2006).

Dietterich, T. G., Approximate statistical tests for comparing supervised classification learning classifiers, *Neural Computation*, 10, pp: 1895–1923, (1998).

Dikici, E., Semerci, M., Saraçlar, M., Alpaydın, E., Classification and ranking approaches to discriminative language modeling for asr, *IEEE Transactions on Audio, Speech, and Language Processing*, 21, pp: 291–300, (2013).

Fawcett, T., An introduction to ROC analysis, *Pattern Recognition Letters*, 27, pp: 861–874, (2006).

Friedman, J. H., Rafsky, L. J., Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests, *Annals of Statistics*, 7, pp: 697–717, (1979).

Hinton, G. H., Delve project, data for evaluating learning in valid experiments, (1996).

İrsoy, O., Yıldız, O. T., Alpaydın, E., Design and analysis of classifier learning experiments in bioinformatics: Survey and case studies, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9, pp: 1663–1675, (2012a).

İrsoy, O., Yıldız, O. T., Alpaydın, E., Soft trees, *21st International Conference on Pattern Recognition*, (2012b), pp: 1819–1822.

Kvam, P. H., Vidakovic, B., *Nonparametric Statistics with Applications to Science and Engineering*, John Wiley, (2007).

Mardia, K. V., Measures of multivariate skewness and kurtosis with applications, *Biometrika*, 57, pp: 519–530, (1970).

Rencher, A. C., *Methods of Multivariate Analysis*, Wiley and Sons, New York, (1995).

Shen, L., Joshi, A. K., Ranking and reranking with perceptron, *Machine Learning*, pp: 73–96, (2005).

Smirnov, N., On the estimation of the discrepancy between empirical curves of distribution for two independent samples, *Bull. Math. Univ. Moscow*, 2, pp: 3–6, (1939).

Statnikov, A., Aliferis, C., Tsamardinos, I., Hardin, D., Levy, S., A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis, *Bioinformatics*, 21, pp: 631–643, (2005).

Ulaş, A., Yıldız, O. T., Alpaydın, E., Cost-conscious comparison of supervised learning algorithms over multiple data sets, *Pattern Recognition*, 45, pp: 1772–1781, (2012).

Vapnik, V., *The Nature of Statistical Learning Theory*, Springer Verlag, New York, (1995).

Yıldız, O. T., Alpaydın, E., Linear discriminant trees, *17th International Conference on Machine Learning*, (2000), pp: 1175–1182.

Yıldız, O. T., Alpaydın, E., Ordering and finding the best of $K > 2$ supervised learning algorithms, *IEEE Transactions on Pattern Analysis Machine Intelligence*, 28,3, pp: 392–402, (2006).

Yıldız, O. T., Alpaydın, E., Statistical tests using hinge/epsilon-sensitive loss, *Proceedings of the International Conference on Computer and Information Sciences*, (2012), pp: 153–160.

Yıldız, O. T., Aslan, Ö., Alpaydın, E., Multivariate statistical tests for comparing classification algorithms, *Learning and Intelligent Optimization (LION) Conference, Rome, Italy, January, LNCS*, volume 6683, (2011), pp: 1–15. Springer.

Zweig, M. H., Campbell, G., Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine, *Clinical chemistry*, 39, pp: 561–577, (1993).

Ek A

Türkçe İngilizce Sözlük

Türkçe	**İngilizce**
akış	run
alıcı işletim özellikleri	receiver operating characteristic (ROC)
anma	recall
artçı	post-hoc
artı	positive
asıl örnek	prototype
bağlanım	regression
bağlı parça	connected component
başarım	performance
bilgi erişimi	information retrieval
çap	eccentricity
çekirdek	kernel
çok değişkenki	multivariate
değişkenlik çözümlemesi	analysis of variance
denence	hypothesis
deney tasarımı	experiment design
destek yöney makinesi	support vector machine
dizey	matrix
doğrusal algılayıcı	perceptron
duyarlılık	sensitivity
düğüm	node
eksi	negative
en küçük kapsayan ağaç	minimum spanning tree
ϵ-duyarlı yitim	ϵ-sensitive loss
eşdeğişinti	covariance
eşli	paired
geçerleme	validation
gözetimli	supervised
hizip	clique
ilingisel	topological
işaret sınaması	sign test
karar ağacı	decision tree
kat	fold

Türkçe	**İngilizce**
kenar payı	margin
kesinlik	precision
maliyet duyarlı	cost-conscious
menteşe yitimi	hinge loss
merkezi limit teoremi	central limit theorem
öbek	group
ölçüt	metric
örneklem	sample
özgünlük	specificity
özyinelemeli	recursive
serbestlik derecesi	degrees of freedom
sıfır denencesi	null hypothesis
sınama	test
sınıflandırıcı	classifier
sıra dönüşümü	rank transformation
soy	descendant
tek değişkenli	univariate
veri kümesi	data set
yol	path
yöney	vector