Dicle University
Journal of Engineering
https://dergipark.org.tr/tr/pub/dumf
duje.dicle.edu.tr

Araştırma Makalesi / Research Article

Hiyerarşik Bölücü Kümeleme ile Çok Sınıflı Sınıflandırma Performansının Geliştirilmesi: Kümeleme Algoritmaları Üzerine Karşılaştırmalı Bir Analiz

Improving Multiclass Classification Performance with Hierarchical Divisive Clustering: A Comparative Analysis on Clustering Algorithms

Celal Alagöz*

Sivas Bilim ve Teknoloji Üniversitesi, Bilgisayar Mühendisliği Bölümü, celal.alagoz@gmail.com
ORCID: https://orcid.org/0000-0001-9812-1473

MAKALE BİLGİLERİ

Makale Geçmişi:

Geliş 2 Şubat 2024
Revizyon 5 Mayıs 2024
Kabul 14 Haziran 2024
Online 30 Hazirn 2024

Anahtar Kelimeler:

Çok Sınıflı Sınıflandırma, Hiyerarşik Sınıflandırma, Otomatik Hiyerarşi Üretme, Kümeleme Algoritmaları, Hiyerarşik Bölücü Kümeleme

ÖZ

Çok sınıflı sınıflandırma problemini hiyerarşik biçime dönüştürerek performans gelişimi elde etmek araştırılan bir konudur. Böylesi bir dönüşümü otomatik olarak gerçekleştirmede kritik bir bileşen, hiyerarşi oluşturma sürecidir. Bu çalışma, standard kümeleme algoritmaları olan K-Medoids (K-Med), K-Means ve Gaussian Karışım Modelleri'in (GMM) bir hiyerarşi inşa etme tekniği olan Hiyerarşik Bölücü Kümeleme (HDC) yönteminde bölücü işlevinde kullanılmasıyla otomatik olarak elde edilen hiyerarşilerin çok sınıflı veri setlerinin sınıflandırma performansına etkisini incelemektedir. İki farklı performans metriği olan F1 skoru ve doğruluk skoru kullanılarak yapılan analizler, K-Means ve GMM'nin K-Med'e göre genellikle daha etkili olduğunu göstermektedir. Ancak, sonuçlar performans iyileştirmesi ve öğrenme veriminin, sınıf sayısına ve veri setinin bazı karakteristik özelliklerine bağlı olarak değişebildiğini göstermektedir. Sonuçta, hiyerarşik biçime dönüştürmenin sınıflandırma performansına önemli ölçüde etkisi olduğu ve farklı veri setlerinin farklı tepkiler verdiği bulunmuştur. Çalışmanın kısıtları ve gelecek araştırmalar için sağladığı potansiyel de tartışılmıştır. Bu çalışma, kümeleme algoritmalarının HDC yöntemindeki rolünü ve gelecekteki araştırmalara bakan yönlerini anlamak için önemli bir katkı sağlamaktadır.

ARTICLE INFO

Article history:

Received 2 February 2024
Received in revised form 5 May 2024
Accepted 14 K-June 2024
Available online 30 June 2024

Keywords:

Multiclass Classification, Hierarchical Classification, Automated Hierarchy Generation, Clustering Algorithms, Hierarchical Divisive Clustering

Doi: 10.24012/dumf.1430306

ABSTRACT

Transforming a multi-class classification problem into a hierarchical form to achieve performance improvement is a topic of research. An essential aspect in automating such transformation is the process of hierarchy building. This research examines how standard clustering algorithms, including K-Medoids (K-Med), K-Means, and Gaussian Mixture Models (GMMs), contribute as partitioning functions within the Hierarchical Divisive Clustering (HDC) method, a technique for building hierarchies, and their effects on the classification performance of multi-class datasets. Analyses conducted using two different performance metrics, namely F1 score and accuracy score, indicate that K-Means and GMM are generally more effective compared to K-Med. However, the findings indicate that performance improvement and learning efficiency may vary depending on the number of classes and some characteristics of the dataset. It is found that hierarchical transformation significantly influences classification performance, and different datasets exhibit different responses. The study also discusses its limitations and future research directions. This study contributes to understanding the role of clustering algorithms in Hierarchical Divisive Clustering and potential avenues for future research.

* Sorumlu Yazar

Giriş

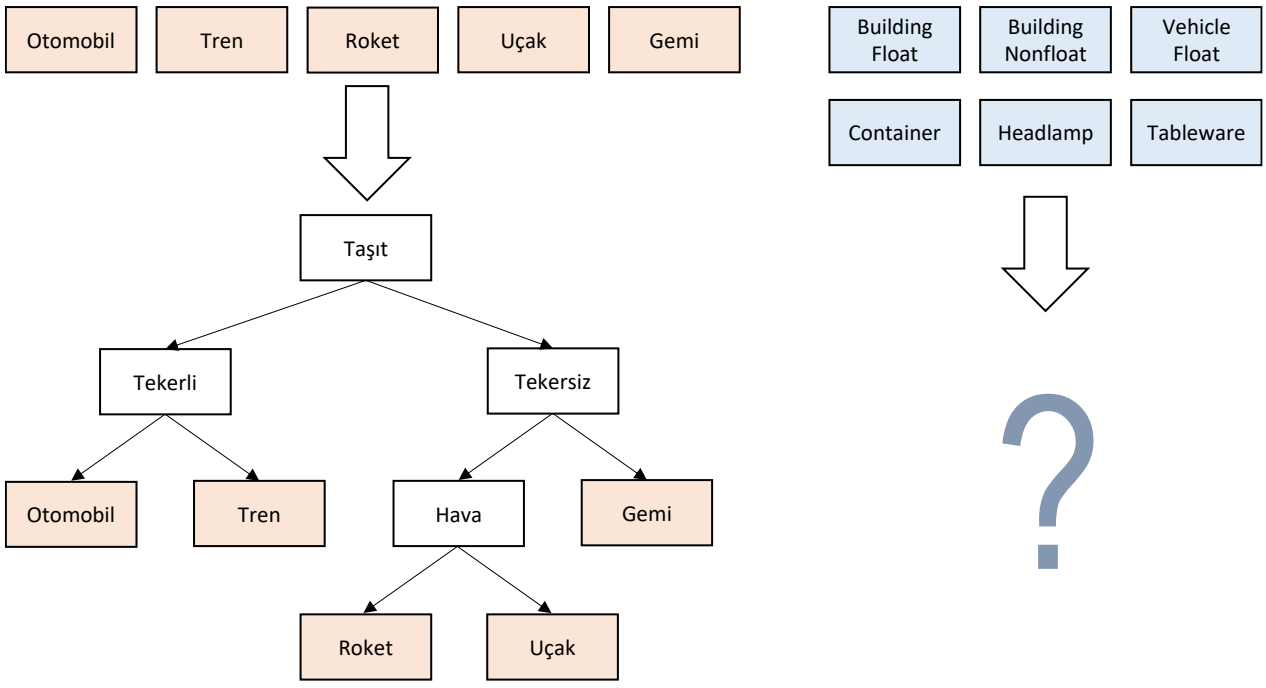
Hiyerarşik Sınıflandırma (HC), veri veya nesnelerin bir ağaç benzeri yapı içinde düzenlendiği Makine Öğrenme (ML) alanında bir sınıflandırma yaklaşımıdır. Bu yöntem, kategorilerin ve alt kategorilerin bir hiyerarşi içinde organize edildiği bir yapı oluşturmayı amaçlar. HC, özellikle karmaşık ve çok katmanlı veri setlerini işlemek için kullanışlıdır.

Bazı veri setleri hazır ve tanımlanmış bir hiyerarşi yapısı ile gelir. Çünkü bazı taksonomilerde sınıflar veya etiketler arasındaki hiyerarşi ilişkisi sezgisel ve iyi kurulmuş olduğundan yerleşmiş bir aşinalık hissiyle bağlantılıdırlar. Örneğin, Şekil 1'deki sınıf hiyerarşisine bakıldığında, hiyerarşinin anlaşılması genellikle zor değildir. Örnekteki hiyerarşi basit olduğu kadar güçlüdür, bu nedenle hiyerarşinin aksini veya alternatifini düşünmek zordur.

Ağaç benzeri yapı içerisindeki hiyerarşide her kategori veya sınıf bir düğüm ile ifade edilir. Bu düğümler, dallar aracılığıyla birbirlerine bağlanır. Bu yapı genellikle, üst düzey kategorilerden başlayarak giderek daha spesifik alt kategorilere doğru dallanır. Üst düzey kategorilere doğru ilerledikçe, ortak özellikler ve karakteristikler artar. Örneğin, Şekil 1'deki "Taşıt" kategorisi içinde "Gemi" ve "Tren" arasında ortak özellikler bulunurken, bunları ayıran spesifik özellikler de vardır. Başka bir deyişle, ağaç benzeri hiyerarşideki düğümler arasındaki ilişki, bir atasal ilişkiye, yani bir soy ilişkisine benzetilebilir. Genellikle, ağacın kökü en yukarıda olduğu için aşağıya doğru ilerlerken evlat düğümlere, yukarı doğru ilerlerken ise atalara ulaşılır. Bu, hiyerarşinin en yalın ve temel algısıdır. Ancak, daha karmaşık ve detaylı ilişki ağları da mevcut olmasına rağmen, kullanışsız oldukları için genellikle önemsenmemiştir [1].

Bu sebepten dolayı, bu çalışmadaki analizde ağaç tipi ve daha spesifik olarak ikili ağaç tipi hiyerarşi incelendi ve bu hiyerarşi yapısı içinde ikili ağaç yapısı özellikle ele alındı. Literatürde ikili ağaç tipi, iç içe geçmiş dikotomiler olarak da adlandırılır [2]. İkili ağaç şeklindeki sınıf hiyerarşisi yapısında ata (non-leaf) düğümler ve (leaf) yaprak düğümler bulunur. Her ata düğümün tam olarak iki evladı, her evladın tam olarak bir atası ve bir kardeşi bulunur. Ağacın başlangıçta ve en yukarıda yer alan kök düğümün atası yoktur. Diğer taraftan, en uç ve en aşağı nokasındaki yaprak düğümlerin evladı yoktur. Böylece, ata düğümler birden fazla sınıfı içerir, dolayısıyla bu sınıf etiketlerini ve karşılık gelen veri örneklerini kapsayan bir üst kategori gibidirler. Bu durumda, kök düğüm çok sınıflı sınıflandırma problemindeki bütün orijinal sınıf etiketlerini (düz etiketler) ve karşılık gelen veri örneklerini içerir. Buna karşılık, yaprak düğümlerin her biri düz etiketlerden sadece birini içerir. Böyle bir hiyerarşide düz etiketli sınıf sayısını n olarak adlandıralım. O halde bu hiyerarşide toplam $2n-1$ düğüm ve bunlardan $n-1$ tanesi ata düğüm olacaktır. Bununla birlikte toplam $2(n-1)$ dal olacaktır.

Bir hiyerarşi yapısı ile birlikte işlenen veri setleri literatürde yaygın şekilde çalışılmıştır. Sözgelimi veri setlerinin sınıf etiketleri arasında yapısal bir ilişki hazır bulunmaktadır. Örnek olarak metin madenciliği ve analizi alanında metin sınıflandırma [3],[4], metinden ve de özellikle de web ortamından bilgi çekme ve bağlam çıkarma [5] gibi görevlerde metnin en temel birimi olan kelimeler arasında hem semantik hem de dilbilgisel hiyerarşi yapıları kullanılmıştır. Görüntü analizi alanında görüntü etiketleme [6] ve açıklama [7], sahne anlama [8] ve görüntü bölütleme [9] görevelerinde gerek zamansal, gerek mekansal ve gerek ise semantik hiyerarşiyi kullanmak avantaj sağlamıştır. Bunların yanında Biyoinformatik disiplininde DNA, RNA, proteinler ve metabolitler ile bunların değişimleri gibi temel aktörlerin eylemlerini ve etkileşimlerini tanımlamayı amaçlayan araştırma alanı olan fonksiyonel genomik analizinde gerek gen işlevlerini kategorize etmek [10] gerekse hücre veya bireyin gözlemlenebilir bir dizi özelliğini (fenotipi), temel genetik özellikler arasındaki fonksiyonel



Şekil 1. Basit bir hiyerarşi örneği (solda). Her hangi bir hiyerarşi ile gelmeyen sınıf etiketleri (sağda).

etkileşimle (genotipi) ve çevresel koşullarla ilişkilendirmek için [11] hiyerarşi düzenin dikkae almak faydalı olmaktadır.

Bir hiyerarşi yapısı eşliğinde çalışılan veri setlerinin yanı sıra, sadece düz etiket bilgilerine sahip olduğumuz bir veri setini alıp hiyerarşik bir yapıya dönüştürme fikri araştırılan bir konudur. Bu araştırmalar "Böyle bir dönüşüm nasıl gerçekleştirilir?" ve "Bu dönüşümün veri analizine, özellikle de sınıflandırma performansına nasıl etki eder veya bu etki nasıl ölçülür?" gibi sorular etrafında şekillenmiştir. Sezgiye dayalı olarak, el ile basit ve sorunsuz bir şekilde bir hiyerarşi oluşturmak suretiyle ilk sorunu çözmenin mümkün olmadığı görünmektedir. Çünkü bir veri setinin etiketlerine baktığımızda, algımızda oluşan yüksek seviye semantik ipuçlarını, mevcut ölçüm kısıtlarıyla elde edilmiş olan nicel açıdan daha karmaşık özelliklerden çıkarmak veya çözümlemek zordur. Bu tür düşük seviyeli bilgileri daha hızlı ve tutarlı bir şekilde işlemek, bilgisayarlar veya diğer tabirle hesap makineleri için daha uygundur. Öte yandan, yüksek seviye semantik algımız itibarıyla etiketler arasında her zaman açık bir hiyerarşi bulunmayabilir. Örneğin, Şekill 1'de düz etiketleri verilen cam tipleri (*glass*) veri setinde, sınıflar arasında net bir hiyerarşi yapısı oluşturmak her zaman mümkün olmayabilir. Bir açıdan cam tipinin yapısal olarak float veya non-float olması esas alınarak bir üst kategori tavsiye edilebilirken başka bir açıdan cam tipinin işlevsel olarak pencere için olup olmadığı bir üst kategori olarak tavsiye edilebilir. Hangisinin daha iyi bir taksonomi olduğu en azından Şekil 1'deki "Taşıt" örneğindeki gibi açık değildir. Ayrıca, yüksek seviye semantik yardımıyla oluşturulan hiyerarşinin sınıf tahmini performansını artırdığına dair bir kanıt bulunmamıştır; aksine, böyle elle oluşturulan hiyerarşinin daha düşük sınıflandırma isabetliliği sonuçları doğurduğu bulunmuştur [12]-[14]. Bu nedenle, bilgisayarın veri setinde gördüğüne dayanarak oluşturulan hiyerarşiler, sınıflandırma performansı açısından daha pratik ve işlevseldir. Bu tür hiyerarşik yapıların oluşturulmasına ise otomatik hiyerarşi üretme denir. Böylelikle eğitim seti kullanılarak kullanıcı tanımı veya uzamanlık alanı bilgisi gerektirmeden bir hiyerarşi oluşturulur ve etiketi bilinmeyen yeni veri örneklerinin sınıfı bu hiyerarşi üzerinden tahmin edilir.

Otomatik hiyerarşi üretme bağlamında çeşitli yöntemler benimsenmiştir. Bunlar arasından veri analizi açısından en hafif ve tavizkar olanı rastgele hiyerarşi üretmektir. Ancak düz etiketli sınıf sayısı arttıkça mümkün olan farklı hiyerarşi yapısı sayısı üssel olarak arttığından bu seçenek uygulanabilir görünmemektedir. Sözgelimi, n adet düz etiketi olan bir veri setinin mümkün olan birbirinden farklı ikili ağaç hiyerarşik yapı sayısı T aşağıdaki gibi özyinelemeli olarak ifade edilebilir:

$$T[n] = \binom{n}{n-1} T[n-1] + \binom{n}{n-2} T[n-2] + \cdots + \binom{n}{\lceil n/2 \rceil} T[\lceil n/2 \rceil] \quad (1)$$

Burada $\lceil . \rceil$ küsüratlı sayıyı bir üst tamsayıya yuvarlama operatörünü ifade eder. Bu hesaba göre, örneğin, düz etiketli 10 sınıflı bir çoklu sınıf veri setinden yaklaşık 20 milyon farklı ikili ağaç hiyerarşi yapısı üretileceği anlamına gelir, ki bu çok geniş bir arama uzayıdır ve hesaplamayı mümkün kılmaz. Diğer yandan, rastgele üretilen ikili ağaç hiyerarşi yapılarından en iyilerini seçmek için meta-sezgisel algoritmalar kullanılmıştır [15].

Diğer otomatik hiyerarşi üretme yöntemleri veri örneklerini kullanarak hiyerarşi üretir. Otomatik hiyerarşi üretme sürecinde sınıflar arasında yapısal bir ilişki olduğu şeklinde bir ön kabul olduğundan, veri örneklerinin analizindeki amaç, bu ilişkiyi belirginleştirmektir. ML alanında hemen akla gelen ilişki türü bir mesafe ölçüsü veya iyi tanımlanmış bir ölçev (metrik) ile ifade edilen benzersizlik ilişkisidir. Bu nedenle, sınıflar arasındaki benzersizlik ölçüsünün veri örnekleri üzerinden belirlenmesi gerekmektedir. Bu bağlamda, veri örnekleri, bağlı oldukları sınıfları temsil edebilirler. Diğer bir yaklaşım ise, veri örneklerinin iç yapısını kullanmaktan ziyade, veri örnekleri üzerinde çalışan bir operatörün -bir sınıflandırıcı bu görev için en elverişli ve pratik model olarak düşünülebilir- tetkiki ve yönlendirmesiyle sınıflar arasındaki ilişkiyi belirlemektir. Veri seti iç yapısını kullanarak elde edilen benzersizlik ölçüsü temsilci tabanlı benzersizlik diye adlandırılırken bir sınıflandırıcının veri örnekleri üzerindeki performansını kullanarak elde edilen benzersizlik ölçüsü sınıflandırıcı tabanlı benzersizlik diye adlandırılır [16].

Temsilci tabanlı benzersizlik yaklaşımında, bir sınıfla ilişkili veri örnekleri o sınıfı ya toplu olarak ya da bir çeşit soyutlama yoluyla temsil edebilir. Toplu temsil yaklaşımında, sınıfla ilişkili veri örnekleri bir dağılım oluşturur ve dağılımlar arasında bilgi teorisi çerçevesinde diverjans veya mesafe ölçüleri kullanılır [17]. Alternatif olarak, soyutlama yaklaşımında, sınıfla ilişkili veri örneklerinin ağırlık merkezi (centroid) hesaplanarak ilgili sınıfın veri uzayında bir temsili elde edilir [18],[19].

Sınıfların temsilini elde ettikten sonraki evre hiyerarşi inşa etmektir. Bu işlem iki metod aracılığıyla gerçekleştirilebilir: hiyerarşik toplayıcı kümeleme (HAC) ve HDC. HAC, aşağıdan yukarı doğru ilerler; ilk aşamada, tüm sınıflar kendi ayrı kümelerindedir ve sınıflar arasındaki mesafenin de üzerinde, HAC algoritmasında ayrıca tanımlanan bir çeşit meta-mesafe ile sınıf toplulukları arası mesafe belirlemesiyle sınıflar kademeli birleşerek son adımda tek bir küme haline gelirler. Bu bağlamda, HAC algoritmasında kullanılan çeşitli meta-mesafeler bulunmaktadır. Diğer yandan, HDC algoritması yukarıdan aşağı doğru ilerler; ilk aşamada bütün sınıflar tek bir kümede toplanmışken, bütün sınıflar kendi ayrı kümesinde olana dek bir bölücü fonksiyon aracılığıyla sınıf kümeleri kademeli olarak bölünür. Burada bölücü fonksiyon için en uygun ve olası aday standard ML kümeleme algoritmalarıdır.

Hiyerarşiyi elde ettikten sonraki aşamada sınıflandırma görevinde hiyerarşiden nasıl istifade edileceği belirlenir. Bu kapsamda, sınıflandırıcı veya sınıflandırıcıları hiyerarşinin farklı bölgelerine empoze etme yöntemleri vardır [1]. Bunun için iki ana yaklaşım vardır. Birincisi, bir sınıflandırıcının hiyerarşideki bütün etiketleri aynı anda değerlendirdiği Global Sınıflandırma (GC) şeması; ikincisi ise hiyerarşinin farklı strateji yaklaşımlarına bağlı olarak farklı noktalarına lokal sınıflandırıcıların konumlandırıldığı ve bunların senkronize şekilde çalıştığı lokal sınıflandırma varyasyonlarıdır.

Hiyerarşi kalitesini ölçmenin farklı yolları mümkündür. Denetimsiz (unsupervised) bir yaklaşımla, hiyerarşi yapısındaki kümelemelerin, bunlar ata düğümlerde bulunur, kollektif kalitesi hesaplanabilir. Bu kapsamda, kümeleme

kalitesi, farklı kümelerdeki sınıfların birbirinden ne kadar kolay ayırt edilebildiği ve aynı kümelerdeki sınıfların birbirinden ne kadar zor ayırt edilebildiğiyle orantılıdır. Hiyerarşi içerisindeki kümelemelerin bütünü düşünüldüğünde ata düğümlerin hiyerarşi içierisindeki stratejik konumu gibi faktörler de hesaba katılabilir. Bununla birlikte, daha pragmatik bir yaklaşımla, HC perfromansıyla düz sınıflandırma (FC) performansı karşılaştırılarak hiyerarşi kalitesi denetimli (supervised) biçimde hesaplanabilir.

Bu çalışmada, sınıf koşullu ortalamalar (CCMs) aracılığıyla sınıf temsili elde edildi. Temel sınıflandırıcı olarak son teknoloji sınıflandırma algoritmalaraından eXtreme Gradient Boosting (XGBoost) kullanıldı. Hiyerarşi inşa aşamasında HDC algoritması kullanıldı ve bu süreçte hiyerarşi kalitesi açısından kritik öneme sahip bölücü fonksiyon için standard kümeleme algoritmalarından K-Means, Gaussian Mixture Method (GMM) ve K-Medoids (K-Med) algoritmaları kullanılarak bunların hiyerarşi kalitesine etkileri karşılaştırmalı olarak incelendi. HC her ata üğümde bir sınıflandırıcı (LCPN) şemasında çalıştırıldı. Hiyerarşi kalitesi HC perfromansıyla FC performansı karşılaştırılarak ölçüldü. Deneyler, 6 tane gerçek dünya veri seti kullanılarak gerçekleştirildi.

Önerilen yaklaşım, Python dilinde standart programlama kütüphaneleri ve araçları kullanılarak uygulanmıştır, bu da uygulanmasını nispeten kolay hale getirmektedir. Kaynak kodunun açık kaynak[1] olarak sunulması ve çalışma materyallerindeki detaylı açıklamalar, hiyerarşi üretme için kullanılan hiyerarşik kümeleme yöntemlerinin yanısıra hiyerarşik sınıflandırma uygulamasını keşfetmekle ilgilenen araştırmacılar ve uygulayıcılar için yaklaşımın erişilebilirliğini ve kullanılabilirliğini artırmaktadır. Bu makaledeki sonuçlar tekrarlanabilir niteliktedir ve kod, farklı sınıflandırıcılar ve farklı veri kümeleri ile çalışacak şekilde kolayca uyarlanabilir.

Metnin yapısı şu şekildedir: Materyal ve metod bölümünde önce çalışmada kullanılan otomatik hiyerarşi üretme aşamaları açıklanmış, ardından performans ölçme ve değerlendirme yöntemleri tanıtılmıştır. Bulgular ve tartışma bölümünde, üretilen hiyerarşilerin performansları analiz edilmiş ve bulgular yorumlanmıştır. Son olarak sonuç bölümünde çalışmada elde edilen dikkate değer sonuçlara vurgu yapılarak çalışma özetlenmiştir.

Materyal ve Metod

Bu bölümde, gelecekteki açıklamaların daha iyi anlaşılması için bazı notasyonlar tanıtılmıştır. N örnek ve m özniteliğe sahip bir veri kümesi D olsun, her bir veri örneği $x^i = \{x^{i1}, x^{i2}, \ldots, x^{im}\}$ olarak temsil edilsin. Sınıf sayısı ve dolayısıyla düz etiket sayısı n olan böyle bir veri setinde düz etiketler $C = \{c_1, c_2, \ldots, c_n\}$ kümesi ile ifade edilir.

İkili ağaç tipi hiyerarşi yapısını göstermek için içiçe geçmiş küme $P = \{P_1, P_2, \ldots, P_{n-1}\}$ kullanılır. Burada P_i, i'inci ata düğümü gösteren bir küme olup $P_i = \{P_{i,0}, P_{i,1}\}$ ifadesinde gösterildiği gibi alt kümeleri içerir. Burada $P_{i,0}$ ve $P_{i,1}$ evlat düğümler olup sırasıyla sol ve sağ evlatları gösterir ve her biri hiyerarşik yapıya göre düz etiketler içeren bir kümedir.

Sınıflar Arası Benzersizlik Elde Etme

Sınıflar arası benzersizlik elde etmek için ön şart her bir sınıfın uzayda bir nesne olarak tanımlanmasıdır. Bu tanıma uygun olarak uzaydaki mesafe üzerinden benzersizlik ölçüsü elde edilmesi yolu sağlanmış olur. Bu çalışmada CCMs yoluyla ön şart sağlandı. Yalnız, CCMs hesaplanmadan önce veri örnekleri üzerinde boyut azaltma işlemi uygulandı. Sınıflar uzayda tanımlandıktan sonra, sınıflar arası mesafe, hiyerarşi inşa sürecinde kullanılacak metriğe bağlıdır.

Doğrusal Ayrımcı Analizi

CCMs'ı hesaplamadan önce boyut azaltmanın uygulanması, hiyerarşik kümelemenin kalitesini artırmak için değerli bir strateji olarak hizmet eder [20]. Bu yaklaşım, verideki gürültünün etkisini azaltırken en ayırt edici özellikleri seçmeye odaklanır. Bu amaçla, Doğrusal Ayrımcı Analizi (LDA) boyut azaltma tekniği olarak seçilir.

LDA, orijinal olarak 1936'da Fisher tarafından önerilen [21], güçlü bir boyut azaltma yöntemi ve etkili bir sınıflandırma aracı olarak hizmet eder. Temel amacı, farklı sınıflar arasındaki kritik ayırt edici bilgiyi korurken bir veri kümesinin boyutunu azaltmaktır. Esasen, LDA, sınıf merkezleri arasındaki mesafeyi maksimize ederken her sınıf içindeki varyansı minimize edecek şekilde yeni bir boyut seti tanımlar, orijinal özelliklerin lineer kombinasyonları olarak ifade edilir.

En yüksek toplam varyansı belirleyen Temel Bileşen Analizinin (PCA) aksine, LDA, sınıf merkezleri arasındaki mesafeyi maksimize etmek ve sınıflar arasında yayılımı minimize etmek için yeni eksenler oluşturur. Bu yaklaşım, genel değişkenliği yakalamaktan ziyade sınıflar arasındaki ayrımı artırmak amacıyla avantaj sağlar.

LDA'nın ayırt edici özelliği, boyut azaltma sürecinde sınıf etiketlerini içermesidir. Sınıf ile ilişkili varyans üzerinde odaklanarak, LDA, sınıflar arasında ayırt etmeye yarayan en uygun özellikleri belirler. Bu denetimli özellik, daha etkili kümelenme ve sınıflandırma sonuçlarına katkıda bulunur.

Bu bağlamda boyut azaltma, veriyi yüksek boyutlu uzaydan, $\mathbb{R}^m$, daha düşük boyutlu uzaya, $\mathbb{R}^k$, dönüştürmeyi içerir. Azaltılmış boyut sayısı k için uygun değerin seçimi otomatik bir süreç olarak uygulanır. Gerçek boyut azaltma işleminden önce, kullanılan boyut azaltma algoritması tüm m bileşeni kullanarak eğitilir. Bu eğitim süreci, her bir bileşenin açıklanan varyans yüzdesini sağlar. Sonuç olarak, tüm bileşenlerin açıklanan varyanslarının toplamı 1.0'a eşittir. k seçimi, açıklanan varyansın kümülatif toplamının 0.95'i aşması veya eşit olması durumunda belirlenir. Başka bir deyişle, bileşenlerin varyansları toplamının en azından 0.95 olduğu bileşen sayısı k olarak belirlenir. Bu yöntem, verideki varyansın önemli bir kısmının daha düşük boyutlu temsilinde korunduğundan, boyut azaltmayı bilgi koruması ile dengelemeye yardımcı olur.

Sınıf Koşullu Ortalamalar

Sınıf koşullu ortalamalar, her sınıf içindeki özniteliklerin ortalama değerlerini belirtir. Formel olarak her sınıf $c_j, (1 \leq j \leq n)$ için sınıf koşullu ortalamalar şu şekilde hesaplanabilir:

$$\mu(c_j) = \{\mu(c_j^1), \mu(c_j^2), \ldots, \mu(c_j^k)\} \quad (2)$$

[1] https://github.com/alagoz/hge_extended

Algoritma 1. HDC Algoritma Adımları

1. Başlatma:
- **a.** Boş bir küme, $P = \{\}$, tanımla.
- **b.** Toplam nesne (sınıf) sayısı n olmak üzere bütün nesneleri içeren $C_0 = \{c_0, c_1, .., c_{n-1}\}$ kök kümeyi P birleşik kümesine yerleştir. Böylece başlangıç biçimlenimi $P = \{C_0\}$ ve $|P| = 1$ şeklinde olacaktır.
- **c.** Bir sonraki bölünecek küme C_0 olduğundan $i = 0$ ayarla.

2. Bölme Adımı:
- **a.** $|P_i| > 2$ ise P_i kümesini bölücü fonksiyon $f_c(.)$ yardımıyla böl ve evlat kümeleri elde et. Böylece $P_{i,0}, P_{i,1} \leftarrow f_c(P_i)$
- **b.** $|P_i| \leq 2$ ise bölme için kontrol edilecek küme indisini güncelle: $i = i + 1$

3. Güncelleme Adımı:
- **a.** Evlat kümelerden çapı (eleman sayısı) en büyük olanı tespit et:

$$P_{i,j} = \underset{j \in \{0,1\}}{\operatorname{argmax}} \{P_{i,0}, P_{i,1}\}$$

- **b.** Evlatları çaplarına göre birleşik kümeye yerleştir:

$$|P_{i,j}| > 1 \text{ ise } P_{|P|+1} = P_{i,j}$$

$$|P_{i,\{0,1\}\setminus j}| > 1 \text{ ise } P_{|P|+2} = P_{i,\{0,1\}\setminus j}$$

- **c.** Bölme için kontrol edilecek küme indisini güncelle:

$$i = i + 1$$

4. Adım 2 ve 3'ü Tekrarla:
Bölme ve güncelleme adımlarını $n - 1$ kez tekrarla.

Burada, $\mu(c_j^i)$, c_j sınıfına ait tüm örneklerin üzerinde bulunduğu j özelliğinin ortalamasını temsil eder. Sonuç olarak, bir sınıfa ait olan tüm veri örnekleri, k-boyutlu bir uzayda vektörler olarak temsil edilir ve bu uzayda tek bir noktaya konsolide edilir. HDC bağlamında, bu sınıf koşullu ortalamalar kümeleme algoritmasının girdi verileri olarak hizmet eder.

Hiyerarşik Bölücü Kümeleme

HDC yukarıdan aşağı çalışan bir algoritma olup benimsenen temel prensip bütün nesneleri kapsayan bir kümeyi özyinelemeli olarak daha küçük kümelere bölmektir. Çalışmada uygulanan hiyerarşik bölücü sınıflandırmanın amacı otomatik hiyerarşi üretme olduğundan ve hiyerarşik sınıflandırmada kullanılan sınıflandırıcılar her ata düğümde konumlandığından oluşturulan hiyerarşi özellikle ata düğüm bilgilerini içermek için tasarlanmıştır.

Çalışmada kullanılan HDC adımları Algoritma 1'de verilmiştir. Algoritmanın amacı ata düğümlerden oluşan bir P üst kümesi oluşturmaktır. Her ata düğüm, kendi atasından gelen sınıfların iki kümeye bölünmesiyle oluşan kümeleri içeren bir bir birleşik kümedir. Ata-evlat ilişkisi P kümesi içeriği itibarıyla dolaylı olarak verili iken bu çalışma için geliştirilen programda açık bir şekilde de kodlanmıştır.

Algoritmanın etkililiğinde en temel faktör aralarındaki benzersizliğin olabildiğince yüksek ve heterojen olduğu alt kategoriler oluşacak şekilde üst kategorileri bölmektir. Burada bölücü fonksiyonun kümeleme yetneği önem arz ediyor. Metnin devamında bu çalışmada seçilen fonksiyonlar tanıtılmıştır.

Bölücü Fonksiyon Olarak Kümeleme Algoritmaları

Hiyerarşi inşa etmek için kullanılan HDC algoritmasının etkililiğinde en temel faktör birbirinden olabildiğince benzersiz ve heterojen alt kategoriler oluşacak şekilde üst kategorileri bölerek alt kategorileri oluşturmaktır. Bu görev için denetimsiz öğrenme yöntemiyle çalışan kümeleme algoritmaları başat adaylardandır. Bu algoritmalar denetimsiz çalışmaları itibarıyla veri noktalarının iç yapısının keşfine göre çalışırlar.

Farklı kümeleme modeline sahip çok sayıda kümeleme algoritmaları mevcuttur. Kümeleme yaklaşımlarına göre mesafe ve ağırlık merkezi, dağılım, yoğunluk, grafik-temelli veya bağlantı modelleri başlıca kümeleme algoritma kategorileridir. Bu çalışma çerçevesinde kümelenmesi amaçlanan nesneler olarak sınıfların tipik özellikleri i) sınıf sayısı ortalama bir veri setindeki örnek sayısından çok daha düşük, ii) veri örneklerinin ortalaması olarak ifade edilen sınıflar mesafe modelinde kullanılmaya daha meyilli olduğu için en yaygın olarak kullanılan K-Means algoritmasını kullanmak en beklenen tercih olarak gözüküyor. K-Means algoritmasına ek olarak aynı kümeleme yaklaşımında çalışan ve aykırı örneklere daha dayanıklı olan K-Medoids algoritması da tercih edildi. Son olarak, her ne kadar örnek setinin küçüklüğü itibarıyla dağılım modeli oluşturmak zorlama bir yöntem gibi olsa da karşılaştırma amacıyla GMM algoritması da deneylerde kullanıldı. Sırada bu algoritmaların nispeten kısa tanıtımları vardır.

K-Means Algoritması

K-means, benzer veri noktalarını bir araya getirme görevi olan kümeleme için kullanılan popüler bir denetimsiz makine öğrenimi algoritmasıdır. Algoritma, bir veri kümesini K kümesine bölmeyi amaçlar, her bir veri noktası en yakın ortalamaya sahip kümelere aittir.

Algoritma 2. K-Means Algoritma Adımları

1. Başlatma:
- **a.** Küme sayısını $r = 2$ olarak tanımla.
- **b.** r küme merkezini temsil eden rastgele başlatılmış r küme merkezini seçin.

2. Atama Adımı:
Her veri noktasını, en yakın ortalamaya sahip kümesine ata. Yakınlık genellikle Öklidyen mesafe kullanılarak ölçülse de, diğer mesafe metrikleri de kullanılabilir.

3. Güncelleme Adımı:
Her kümenin merkezini, o kümeye atanmış tüm veri noktalarının ortalamasını alarak yeniden hesapla.

4. Adım 2 ve 3'ü Tekrarla:
Atama ve güncelleme adımlarını yakınsama sağlanana kadar tekrarla. Yakınsama, merkez noktaları artık önemli ölçüde değişmiyorsa veya önceden belirlenmiş bir iterasyon sayısına ulaşıldığında gerçekleşir.

Algoritma, küme içindeki karelerin toplamını minimize etmeyi amaçlar, bu da her küme içindeki veri noktalarını mümkün olduğunca benzer yapmaya çalışır. Küme sayısı seçimi önemlidir ve genellikle alan bilgisi veya deneme yanılma gerektirir. Bu çalışmada amaç ikili ağaç hiyerarşisi üretmek olduğu için küme sayısı 2 olarak ayarlanmıştır.

K-means, merkez noktalarının başlangıç konumuna duyarlıdır. Farklı başlatmalar farklı nihai kümeleri üretebilir. Algoritma, bir yerel minimuma ulaşabilir ve bu sorunu azaltmak için farklı başlatmalarla birden fazla çalıştırma yapılabilir. K-Means uygulama adımları Algoritma 2'de verilmiştir. Noktalar arası mesafe için farklı mesafe metrikleri kullanılabilir. Bu çalışmada "euclidean" mesafe tercih edilmiştir.

K-Medoids Algoritması

K-Med algoritması [22], K-Means algoritmasının bir varyasyonu olan kümeleme algoritmasıdır. Hem K-Means hem de K-Med, bir veri kümesini r kümesine bölmeyi amaçlasa da, temel fark kümelerin merkezlerini nasıl temsil ettiklerindedir. K-Med uygulama adımları Algoritma 3'te verilmiştir.

K-Means'ten farklı olarak, K-Med medoid kullanır, bu da bir kümenin içindeki tüm diğer noktalara olan benzerliklerin toplamını en aza indiren veri noktasıdır. Medoid, bir küme içinde, tüm diğer noktalara olan benzerliklerin toplamını en aza indiren veri noktasıdır. Özellikle aykırı değerlerin bulunduğu durumlarda, medoid ortalama'dan daha güvenilir bir ölçüdür.

K-Med algoritmasında benzersizlik ölçüsü (veya mesafe metriği) seçimi kritiktir. Verinin doğasına bağlı olarak Euclidean mesafe, Manhattan mesafe veya diğer benzerlik ölçüleri kullanılabilir. Ortalama yerine temsilci bir nokta (medoid) kullanılması nedeniyle K-means'a göre aykırı değerlere daha dayanıklıdır. Ortalamanın anlamlı veya temsilci bir merkez olamayacağı durumlar için uygundur.

Diğer yandan K-means'a göre hesaplama maliyeti daha yüksektir, özellikle medoid değiş tokuşu içerdiği için. Ayrıca başlangıç medoidlerin seçimine duyarlıdır.

Algoritma 3. K-Medoids Algoritma Adımları

1. *Başlatma:*
   a. Küme sayısı K'yi seç.
   b. K küme merkezini temsil eden rastgele başlatılmış K küme merkezini seçin.
2. *Atama Adımı:*
   Her veri noktasını, en yakın ortalamaya sahip kümesine ata. Yakınlık genellikle Öklidyen mesafe kullanılarak ölçülse de, diğer mesafe metrikleri de kullanılabilir.
3. *Güncelleme Adımı:*
   Her kümenin merkezini, o kümeye atanmış tüm veri noktalarının ortalamasını alarak yeniden hesapla.
4. *Adım 2 ve 3'ü Tekrarla:*
   Atama ve güncelleme adımlarını yakınsama sağlanana kadar tekrarla. Yakınsama, merkez noktaları artık önemli ölçüde değişmiyorsa veya önceden belirlenmiş bir iterasyon sayısına ulaşıldığında gerçekleşir.

Algoritma 4. EM Algoritma Adımları

1. *E adımı:*
   Her veri noktasının her bir Gauss bileşenine ait olma olasılığını (sorumluluk) tahmin et.
2. *M adımı:*
   Sorumluluklara bağlı olarak parametreleri (ortalama, kovaryans ve ağırlıklar) güncelle.
3. *Adım 1 ve 2'yi Tekrarla:*
   E ve M adımlarını yakınsama sağlanana kadar tekrarla. Yakınsama, parametreler artık önemli ölçüde değişmiyorsa veya önceden belirlenmiş bir iterasyon sayısına ulaşıldığında gerçekleşir.

Özetle, K-Medoids algoritması, K-Means algoritmasının bazı sınırlamalarına çözüm sunan dayanıklı bir kümeleme algoritmasıdır, özellikle aykırı değerlerin küme merkezlerini önemli ölçüde etkileyebileceği durumlarda kullanılır. Medoid, kümeler için daha anlamlı bir merkez sağlamak adına geniş bir uygulama alanında kullanılır.

Gauss Karışım Modelleri Algoritması

GMMs, bir karışım modelini temsil eden, çeşitli Gauss (normal) dağılımların bir karışımını içeren olasılık modelleridir. Bir GMM, verinin birkaç Gauss dağılımının karışımı tarafından oluşturulduğunu varsayar. Karışımdaki her Gauss bileşeni, verideki bir küme veya gizli alt grupu temsil eder.

GMM'lerin olasılık yoğunluk fonksiyonu, bireysel Gauss bileşenlerinin Olasılık Dağılım Fonksiyonlarının (PDF) ağırlıklı toplamıdır. Matematiksel olarak şu şekilde ifade edilebilir:

$$P(x) = \sum_{i=1}^{k} \pi_i \mathcal{N}(x|\mu_i, \Sigma_i) \quad (3)$$

Burada π_i ağırlığı, yani her Gauss bileşeninin genel karışıma katkısını temsil eder. μ_i ortalamayı, yani her Gauss bileşeninin merkezini belirler. Σ_i kovaryansı, yani her Gauss dağılımının yayılmasını veya şeklini belirtir. GMM'lerin eğitilmesi bu parametrelerin ayarlanması ile karakterizedir. Bu yaygın olarak yinelemeli biçimde Maksimum Olabilirlik Tahmini (MLE) problemini gizli parametreler aracılığıyla çözen Beklenti - Maksimizasyon (EM) algoritmasıyla sağlanır. EM uygulama adımları Algoritma 4'te verilmiştir.

GMM'ler başlangıca duyarlıdır. Ortalama başlatmak için genellikle K-Means kümeleme veya hiyerarşik kümeleme kullanılır, kovaryanslar için kimlik matrisleri, ve ağırlıklar eşit olabilir.

GMM'ler belirli bir veri noktasının belirli bir kümeye ait olma olasılığını ölçen olasılık temellidir. GMM'ler genellikle veri noktalarının birden çok kümeye farklı olasılıklarla ait olabileceği yumuşak kümeleme için kullanılır. Diğer yandan, ilk parametre seçiminin sonuçları etkilemesi şeklinde başlangıca duyarlılık hassasiyeti ve bileşen sayısının arttığı oranda GMM'leri eğitme maliyetinin artması GMM'nin kısıtlarındandır.

Özet olarak GMM'ler, verinin altında yatan veri dağılımını anlamak veya olasılıksal kümeleme gerçekleştirmek gereken çeşitli alanlarda uygulama bulan çok yönlü modellerdir.

Tablo 1. Deney için kullanılan veri setleri

Veri Seti	n	m	N	F1 Skoru	Doğruluk Skoru
LED-display-domain-7digit	*10*	*7*	*500*	*0.683±0.044*	*0.684±0.044*
ecoli	*8*	*7*	*336*	*0.739±0.113*	*0.859±0.025*
analcatdata_broadwaymult	*7*	*3*	*267*	*0.144±0.036*	*0.212±0.045*
glass	*6*	*9*	*214*	*0.721±0.075*	*0.780±0.055*
eucalyptus	*5*	*14*	*641*	*0.621±0.033*	*0.628±0.030*
prnn_cushings	*4*	*2*	*27*	*0.629±0.211*	*0.700±0.163*

Deney Kurgusu ve Performans Değerlendirme

Temel Sınıflandırıcı

Çalışmada temel sınıflandırıcı olarak son teknoloji ML sınıflandırıcı modellerinden XGBoost kullanıldı [23]. XGBoost, özellikle yapılandırılmış/tablo verileri ve regresyon problemleri için tasarlanmış geniş çapta benimsenen açık kaynaklı bir ML kütüphanesidir. Denetimli (supervised) öğrenme görevleri için tasarlanmıştır. Gradyan arttırma yönteminden yararlanarak zayıf öğrenicilerin (genellikle karar ağaçları) ardışık olarak birleşimini oluşturur. XGBoost, karmaşık modelleri cezalandırarak aşırı uyumlanmayı azaltan L1 (Lasso) ve L2 (Ridge) düzenleme terimlerini içerir. Model basitleştirme için "ağaç budama" ve hiperparametre ayarlaması için yerleşik çapraz doğrulama gibi dikkate değer özellikler bulunur. XGBoost, özellik seçimi ve model yorumlanabilirliğine yardımcı olan bir özellik önem puanı sağlar.

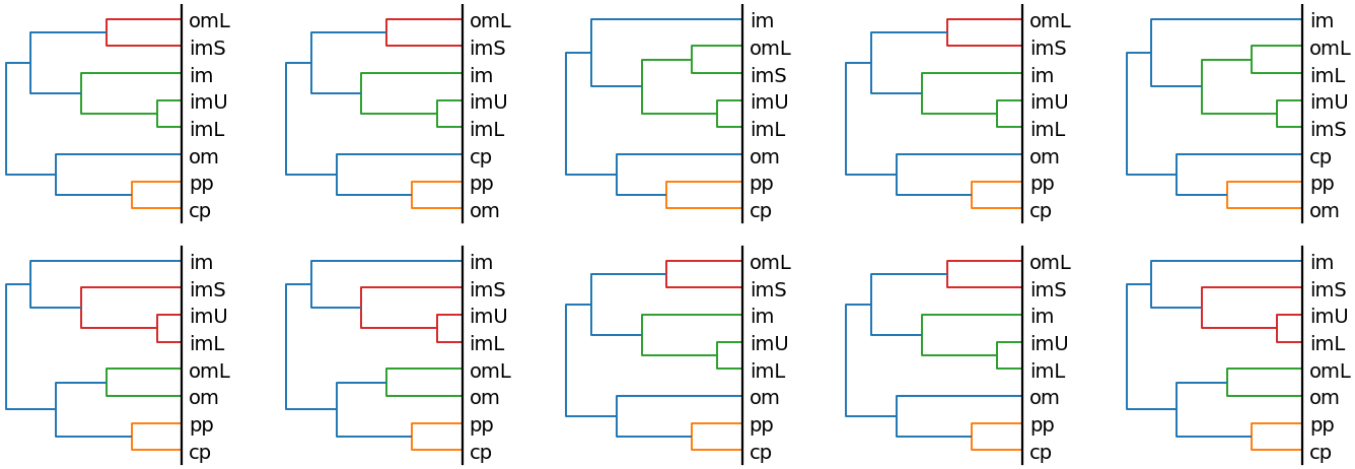
Kullanılan Veri Setleri

Kümele algoritmalarının HC perfromansına etkisini ölçmek için çok sınıflı gerçek dünya veri setleri kullanıldı. Veri setleri ML araştırma ve deneylerini kolaylaştırmayı amaçlayan açık kaynaklı bir çevrimiçi platform olan OpenML deposundan [24] elde edildi. Farklı sınıf sayısına ait 6 veri seti Python ortamında OpenML API [25] aracılığıyla indirildi. Kullanılan veri setlerine genel bir bakış Tablo 1'de gösterildi.

Performans Değerlendirme

Sınıflandırma modellerinin görünmeyen veri üzerindeki performansını değerlendirmek için Monte Carlo çapraz doğrulama yöntemi kullanıldı. Tekrar edilmiş rassal alt örnekleme olarak ta bilinen Monte Carlo çapraz doğrulama sürecinde veri seti birden fazla iterasyon için rassal eğitim ve test setlerine bölünür. Eğitim ve test setlerinin farklı çalıştırmalarada yeniden üretilebilirliğini sağlamak için rassal karıştırma (shuffling) nüvesi (seed) iterasyon numarası olarak belirlendi. Bu çalışmada Monte Carlo 20 kez çalıştırıldı ve her defasında test kümesinin oranı %20 tutuldu. Böylelikle her yinelemede farklı bir eğitim seti oluştuğundan potansiyel olarak –her ne kadar tercih edilmese de– farklı bir hiyerarşi üretilebilir. Şekil 2'de bir örnek vaka çalışması olarak "ecoli" veri seti kullanılarak 10 yinelemede üretilen hiyerarşiler gösterildi. Dikkat edildiğinde bazı yinelemelerde eş hiyerarşilerin üretildiği gözlemleniyor. Bir veri setinde farklı eğitim setleri kullanılması durumunda üretilen hiyerarşilerin emsalliğinin değişmemesi oranında otomatik hiyerarşi kararlığından söz edilebilir. Bu kararlılık en azından sezgisel olarak hiyerarşi kalitesinin yüksek olduğu anlamına gelebilir. Yine de bu kanaatin, yani hiyerarşik kararlılık ile hiyerarşi kalitesi arasında korelasyon olduğunun yeterli, yani istatistiksel anlamı (statistical significance) gösterecek miktarda deneysel testlerle doğrulanması gerekmektedir.

Performans değerlendirme metriği için doğruluk ve F1 skoru kullanıldı. Her iki metrik te çoklu Monte Carlo çalıştırmasından elde edilen ortalama ve ± sembolünü takip eden standard sapma ile gösterildi. Küsurat 3 dijit ile gösterildi. Doğruluk, sınıflandırıcının genel doğruluğunu, doğru tahmin edilen örneklerin toplam örneklerle oranını



Şekil 2. Monte Carlo sürecinde *ecoli* veri set için üretilen hiyerarşiler: Düz etiketler hücre içerisinde protein konumunu gösterir: cp (cytoplasm), im (inner membrane without signal sequence), pp (perisplasm), imU (inner membrane, uncleavable signal sequence) 35 om (outer membrane) 20 omL (outer membrane lipoprotein) 5 imL (inner membrane lipoprotein) 2 imS (inner membrane, cleavable signal sequence) 2

Tablo 2. Performans iyileşmesi sonuçları.

Veri Seti	n	Skor	FC	K-Med	K-Means	GMM
LED-display-domain-7digit	*10*	*F1*	*0.683±0.044*	***0.692±0.042***	***0.713±0.037*** ↑	***0.713±0.037*** ↑
		Doğruluk	*0.684±0.044*	***0.695±0.043*** ↑	***0.711±0.039*** ↑	***0.711±0.039*** ↑
ecoli	*8*	*F1*	*0.739±0.113*	***0.743±0.089***	***0.825±0.064*** ↑	***0.825±0.064*** ↑
		Doğruluk	*0.859±0.025*	*0.855±0.022*	***0.867±0.025***	***0.867±0.025***
analcatdata broadwaymult	*7*	*F1*	*0.144±0.036*	***0.160±0.041***	***0.159±0.038***	***0.158±0.038***
		Doğruluk	*0.212±0.045*	***0.233±0.054*** ↑	***0.233±0.051*** ↑	***0.232±0.051*** ↑
glass	*6*	*F1*	*0.721±0.075*	*0.686±0.093*	*0.687±0.093*	*0.687±0.093*
		Doğruluk	*0.780±0.055*	*0.756±0.064* ↓	*0.762±0.069*	*0.762±0.069*
eucalyptus	*5*	*F1*	*0.621±0.033*	***0.646±0.042*** ↑	***0.648±0.037*** ↑	***0.648±0.037*** ↑
		Doğruluk	*0.628±0.030*	***0.649±0.039*** ↑	***0.651±0.034*** ↑	***0.651±0.034*** ↑
prnn cushings	*4*	*F1*	*0.629±0.211*	***0.759±0.173*** ↑	***0.716±0.170***	***0.716±0.170***
		Doğruluk	*0.700±0.163*	***0.775±0.142***	***0.750±0.144***	***0.750±0.144***

* *Ortalama skorun FC performansından yüksek olması kalın yazıtipi ile farkın istatistiksel anlamın olması ise* ↑ *sembolüyle gösterildi.*

ölçerek nicelendirir. Sınıflandırıcının performansının sezgisel bir değerlendirmesini sağlar.

Diğer yandan ikili sınıflandırma görevlerinde yaygın olarak kullanılan F1 skoru, hassasiyet (precision) ve geri çağırma (recall) arasında denge sağlayan, bir sınıflandırıcının performansının tek bir sayısal ölçüsünü sunan bir metriktir. Hassaiyet, model tarafından yapılan pozitif tahminlerin doğru olma oranını ölçerken, recall (sensitivity olarak da bilinir) modelin veri kümesindeki tüm pozitif örnekleri doğru bir şekilde tanıma yeteneğini ölçer. Sınıflar arasında dengesizlik olduğu durumlarda özellikle yararlı olup yanlış pozitifler ve yanlış negatiflerin kapsamlı bir değerlendirmesini sağlar.

Çoklu sınıflandırma bağlamında, tahmin edilecek iki sınıfın ötesinde daha fazla sınıf olduğunda, sınıflandırıcının performansını değerlendirmek için F1 skoru genişletilebilir. Ancak, çoklu sınıflı durumlar için F1 skorunun hesaplanması, ikili sınıflandırma durumuna kıyasla bazı ayarlamalar gerektirir. Bu ayarlar her sınıfa ya da her örneğe eşit önem verilmesine veya her sınıfın önemi frekansıyla orantılı olmasına göre değişir. Bu çalışmada her sınıfa eşit önem verildiğinden çoklu sınıf durumuna ayarlamak için Makro-Ortalama F1 skoru yaklaşımı benimsenmiştir. Bu yaklaşımda, her sınıf için F1 skorları ayrı ayrı hesaplanır ve ardından tek bir F1 skoru elde etmek için ortalanır. Her sınıf, büyüklüğüne bakılmaksızın hesaplamada eşit ağırlığa sahiptir.

Hiyerarşi kullanmanın performansa olan etkisini ölçmek için öğrenme verimi tanımlandı. Öğrenme verimi basitçe HC ortalama F1 skorunun FC ortalama F1 skoruna oranının 100 katı olarak değerlendirildi. Oranın >100 olması performansta artma, <100 olması ise azalma olduğunu gösterir.

İstatistiksel Test

Bu çalışmada bir veri seti üzerinde hiyerarşik ve düz sınıflandırma performanslarını karşılaştırmak amaçlanmaktadır. Bilidiği gibi bir veri seti için Monte Carlo çapraz doğrulama yöntemiyle 20 kez HC ve FC skoru üretiliyor. Bu iki süreçte elde edilen skorlar arasında istatistiksel anlame sahip bir fark olduğunu göstermek için parametrik olmayan bir istatistiksel hipotez testi, Wilcoxon işretli sıralama testi (Wilcoxon signed-rank test) kullanıldı [26]. İkili ölçümleri karşılaştırmak için kullanılan parametrik çiftli T-testi (paired T-test) özelinde kabul edilen verilerin normal dağıldığı gibi varsayımları üstlenmemek için bu yöntem tercih edildi. Ancak, bu test eşleştirilmiş gözlemler arasındaki farkların sürekli olduğunu ve en azından sıralı (ordinal) bir ölçekte ölçüldüğünü varsayar. Testte kullanılan nul hipotezi şu şekilde ifade edilir:

H0: Popülasyonda eşleştirilmiş gözlemler arasında fark yoktur.

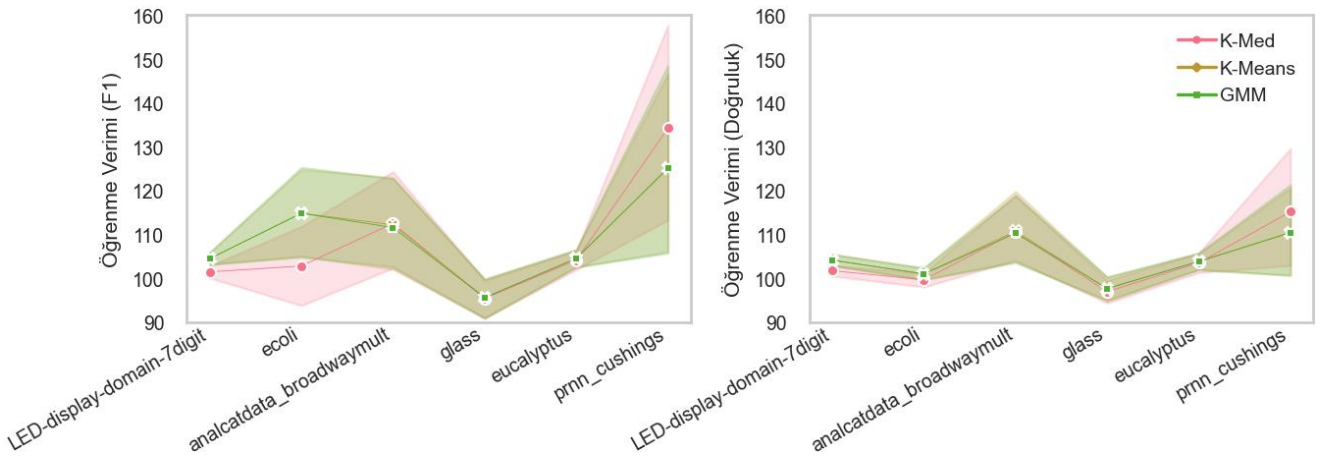
Test sürecinde elde edilen test istatistiği W şu şekilde ifade edilir:

$$W = \sum_{i=1}^{N_r} sgn(x_{2,i} - x_{1,i}) R_i \quad (4)$$

Burada N_r popülasyondaki gözlem sayısını –bu, çalışmada Monte Carlo çapraz doğrulama yönteminin kaç kez çalıştırıldığına karşılık gelir–, $sgn(.)$ işaret fonksiyonunu, $x_{1,i}$ ve $x_{2,i}$ i'inci gözlemde karşılaştırılan örnekleri –bu çalışmada FC ve HC skorlarına karşılık gelir– ve son olarak R, x_1 ve x_2 arasındaki mutlak farkların mutlak büyüklüğüne dayalı sıralamayı ifade eder. Test istatistiği W değeri işaretli sıralama dağılımıyla karşılaştırarak p değeri üretilir. Bu çalışmada nul hipotezi reddetmek için $p < 0.05$ seçildi.

Bulgular ve Tartışma

Bu çalışma standard kümeleme algoritmalarından K-Med, K-Means ve GMM'nin hiyerarşik bölücü kümele yönteminde bölücü işlevinde kullanılmasıyla otomatik olarak elde edilen hiyerarşilerin çok sınıflı veri setlerinin sınıflandırma performansına etkisini 6 gerçek dünya veri seti üzerinde deney yaparak incelemektedir. Temel sınıflandırıcı algoritması XGBoost, Python dilinde xgboost paketi



Şekil 3. F1 ve doğruluk skoru için elde edilen öğrenme verimlerinin bölücü fonksiyonlar, veri setleri ve performans metrikleri açısından karşılaşırması. Monte Carlo sürecinde meydana gelen öğrenme verimlerinin ortalaması işaretçilerle, güvenlik aralığı ise gölgelendirilmiş alan ile vurgulandı

kullanılarak çalıştırıldı. Her hangi bir veri seti özelinde sınıflandırıcının hiperparametre ayarı yapılmadı; onun yerine pakette tanımlı öndeğer parametreler kullanıldı.

Performans İyileşmesi Analizi

HC ile FC F1 ve doğruluk skoru açısından performans karşılaştırması Tablo 2'de özetlendi. Başlıca dikkat çeken gözlemlerden biri K-Means ve GMM *analcatdata broadwaymult* veri setinde görünen ufak fark hariç bütün durumlarda aynı performansı göstermesi oldu. Sonuçlar toplamda 6 veri setinin içerisinden kaç tanesinde perfromans iyileşmesi elde edildiği bakımından incelendiğinde K-Med kullanılarak üretilen hiyerarşilerde F1 skoru açısından 2'si istatistiksel anlama sahip 5 veri setinde ortalama skorda artış gözlemlenirken doğruluk skoru açısından 3'ü istatistiksel anlama sahip 4 veri setinde ortalama skorda artış gözlemlendi. K-Means ve GMM kullanılarak üretilen hiyerarşilerde hem F1 hem doğruluk skoru açısından 3'ü istatistiksel anlama sahip 5 veri setinde ortalama skorda artış gözlemlendi. Bu sayılar hiyerarşi oluşturma için K-Means ve GMM'nin K-Med'e nispeten daha terich edilebilir kümeleme algoritmaları oabileceğini ima ediyor.

Sonuçlar veri setlerinin sınıf sayıları ile ilişkisi çerçevesinde incelendiğinde, F1 skoru açısından, K-Med kullanımında 4 ve 5 sınıflı *eucalyptus* ve *prnn cushings* gibi daha az sınıflı veri setlerinde perfromansta istatistiksel anlama sahip gelişme elde edildi. K-Means ve GMM kullanımında 10 ve 8 sınıflı *LED-display-domain-7digit* ve *ecoli* gibi nispeten daha çok sınıflı veri setleri ile birlikte 5 sınıflı *eucalyptus* veri setinde perfromansta istatistiksel anlama sahip gelişme elde edildi. Diğer yandan doğruluk skoru açısından, K-Med kulanımında 10, 7 ve 5 sınıflı *LED-display-domain-7digit, analcatdata broadwaymult* ve *eucalyptus* gibi hem çok hem az sınıf sayılı veri setlerinde istatistiksel anlama sahip performans gelişimi elde edildi. K-Means ve GMM kullanımında da aynı veri setlerinde istatistiksel anlama sahip performans gelişimi elde edildi. Bu sonuçlar K-Med bölücü fonksiyonunun daha az sınıf sayılı veri setlerinde kullanılmasını daha çok sınıf sayılı veri setlerinde kullanılmasından daha avantajlı olabileceğine işaret etse de bu gözlemi istatistiksel anlamda desteklemek için yeterli sayıda veri seti kullanılmamıştır.

Hiyerarşik biçime dönüştürmenin sınıflandırma performansına etkisi veri setleri açısından değerlendirildiğinde her veri setinin hiyerarşik biçime yatklığı açısından farklı karakter sergilediği gözlemlendi. Spesifik olarak veri setleri tek tek incelediğinde *eucalytpus* veri seti gerek hiyerarşi üretme tasarısı gerekse kullanılan performans metriği farketmeksizin istatistiksel anlamlı performans gelişimi gösterdi. Bu bulgu, bu veri setinin hiyerarşik biçime dönüşmeye elverişli olduğunu gösterir. Benzer şekilde, *LED-display-domain-7digit* veri seti de tek bir istisna dışında her durumda istatistiksel anlamlı performans gelişimi gösterdiğinden bu veri setinin de hiyerarşik biçime dönüşmeye yatkın olduğu söylenebilir. Diğer taraftan *glass* veri tipi her durumda, hatta K-Med ve doğruluk skoru surumunda istatistiksel anlamlı olacak şekilde performans gerilemesi gösterdi. Bu bulgu, bu veri setinin hiyerarşik biçime pek yatkın olmadığını gösterir. Ancak yine de bu veri setinin başka bir hiyerarşik biçime dönüşmesi durumunda efektif olarak performansta iyileştirme elde edilmesinin mümkün olmadığı iddia edilemez; çünkü farklı bir hiyerarşi üretme konfigürasyonu ve farklı bir temel sınıflandırıcı kullanılması durumunda performans iyileşmesi deneysel olarak bildirilmiştir [20]. Belki bu veri seti için verimli ve efektif bir hiyerarşinin bu çalışmada kullanılan hiyerarşi üretme yöntemler marifetiyle henüz bulunamadığı kaydı düşülebilir. Diğer veri setlerinde nispeten değişken hiyerarşi yatkınlığı karakterleri gözlemlendi. Spesifik olarak *ecoli* veri setinde K-Means ve GMM kullanılarak üretilen hiyerarşiler daha kaliteli olmaya yatkın iken *prnn cushing* veri setinde K-Med kullanılarak üretilen hiyerarşilerin daha verimli olduğu izlenimi oluştu. *Analcatdata broadwaymult* veri setinde ise genelde efektif hiyerarşi elde etme elverişliliği gözlemlendi ama bu etkinlik sadece doğruluk performans metriği açısından istatistiksel anlam kazandı.

Öğrenme Verimi Analizi

Şimdiye kadarki analizlerde performans iyileştirmesinin olup olmaması ve bu iyileştirmenin istatistiksel anlamının olup olmaması dikkate alınırken iyileştirmenin hangi oranda olduğu incelenmedi. İyileşme oranını nicel olarak gösteren öğrenme verimlerinin bölücü fonksiyonlar, veri setleri ve performans metrikleri arasında karşılaştırmasını içeren grafik Şekil 3'te gösterildi. İlk göze çarpan genel olarak F1 skoru kriterinde doğruluk skoru kriterine göre genelde daha yüksek verim elde edilmesidir.

Bölücü fonksiyonlar eşliğinde elde edilen öğrenme verimi açısından sonuçlar genelde K-Means ve GMM kümeleme

algoritmalarının daha yüksek verime yol açtığını gösterdi. Daha detaylı incelendiğinde hem F1 hem doğruluk kriterine göre 10 sınıflı *LED-display-domain-7digit* ve 8 sınıflı *ecoli* veri setlerinde K-Means ve GMM daha iyi iken 7 sınıflı , 6 sınıflı ve 5 sınıflı ver setlerinde yakın verim gözlemlendi. Sadece 4 sınıflı veri setinde K-Med biraz daha iyiydi.

Veri setlerinde elde edilen öğrenme verimi açısından incelendiğinde F1 skoru kriterinde *prnn cushings*, *ecoli* ve *analcatdata broadwaymult* veri setlerinde diğer veri setlerine nispeten daha yüksek verim elde edildi; sadece *ecoli* ve K-Med kombinasyonu bu trendin haricindeydi. Doğruluk skoru kriterine göre ise *prnn cushings* ve *analcatdata broadwaymult* veri setlerinde daha yüksek verim gözlemlendi.

Kısıtlar ve Gelecek Çalışmalar

Çalışma standard kümeleme algoritmalarının otomatik hiyerarşi üretme mekanizmasında bölücü foknsiyon olarak kullanılmasının farklı açılardan kapsamlı analizini içerse de hiyerarşi üretme ve HC çok farklı yönler ve bileşenler içerdiğinden bazı kısıtlar ve ihmaller bulundurmaktadır.

Öncelikle, her bir veri setinin kendi içinde anlamlı bir sonuç çıkaracak düzeyde istatistiksel test yapılmasına rağmen çalışmada toplamda sadece 6 veri seti kullanıldı. Her ne kadar bu veri seti örnekleri üzerinden gerek sınıf sayısı açısından gerekse performans değerledirme metriği açısından bulgular analiz edildiyse de, bu çalışmada incelenen hiyerarşi üretme yöntemlerinin, genel olarak her hangi bir veri seti üzerinde etkisini istatistiksel anlamda göstermek için 6'dan çok daha fazla veri seti örneğine ihtiyaç olduğundan, çalışma bu açıdan eksiklik içermektedir. Gelecekte daha büyük sayıda veri seti ile çalışmak hedeflenmektedir.

Hiyerarşi inşa etmek sınıflar arasındaki yapısal bağı keşfetmeye bağlıdır. Bu bağın ortaya çıkarılması için gerekli olan sınıf temsili için bu çalışmada CCMs kullanılsa bile veri örneklerinden farklı sınıf temsili elde etme seçenekleri de inceleme kapsamına dahi edilebilir. Farklı sınıf temsili seçenekleri literatürde araştırılmış ve denenmiştir. Sınıflara özgü veri noktalarının dağılımını kullanarak sınıfa özgü dağılımlar arasında benzersizlik oluşturma [17] bunlardan biridir. Başka seçenekler eğitim seti üzerinde sınıflandırıcı çalıştırarak snıflar arası ilişkiye dair çıkarım yapmaktır [27-29].

Çalışmada sadece HDC düşünüldü, HAC düşünülmedi. Ayrıca, temel sınıflandırıcı olarak sadece XGBoost kullanıldı. Gelecekte hiyerarşi üretme ve HC'nin zikredilen bileşenleri de incelemeye katılıp daha kapsamlı analizler yapılabilir.

Özet

Çalışma bulgularına dayanarak, aşağıdaki sonuçlar çıkarılabilir:

- K-Means ve GMM, HDC yönteminde kullanılarak elde edilen hiyerarşilerin sınıflandırma performansını iyileştirmede potansiyel olarak daha etkili olabilir. Bu algoritmalar, genellikle K-Med'e göre daha tercih edilebilir olabilir.

- Farklı sınıf sayılarına sahip veri setlerinde, performans iyileştirmesi ve öğrenme verimi farklılık göstermektedir. Daha az sınıflı veri setlerinde K-Med'in, daha çok sınıflı veri setlerinde ise K-Means ve GMM'nin daha etkili olabileceği gözlemlenmiştir.

- Hiyerarşik biçime dönüşüm, sınıflandırma performansını önemli ölçüde etkileyebilir ve bu etki veri setinin özelliklerine bağlıdır. Bazı veri setleri için hiyerarşik biçime dönüşüm daha etkili olurken, diğerleri için performans gerilemesine neden olabilir.

- Çalışmanın kısıtları ve eksiklikleri göz önüne alındığında, gelecekte daha geniş kapsamlı çalışmalar ve daha fazla veri seti kullanılarak yapılan analizlerle sonuçların daha iyi genelleştirilmesi hedeflenmelidir.

- Bu çalışma, otomatik hiyerarşi üretme mekanizmasında kümeleme algoritmalarının kullanımının sınıflandırma performansı üzerindeki etkisini incelemiştir. Ancak, daha fazla araştırma, farklı kombinasyonlar ve yöntemlerin incelenmesini gerektirmektedir.

Sonuç olarak, bu çalışma, kümeleme algoritmalarının HDC yönteminde nasıl kullanılabileceğini ve sınıflandırma performansı üzerindeki etkilerini anlamak için önemli bir adım atmıştır. Gelecekteki araştırmaların, daha geniş veri setleri ve farklı yöntemlerle yapılan analizlerle bu konuda daha fazla bilgi sağlaması beklenmektedir.

Kaynaklar

[1] C. N. Silla, and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data mining and knowledge discovery,* vol. 22, pp. 31-72, Jan. 2011, doi: https://doi.org/10.1007/s10618-010-0175-9

[2] E. Frank and S. Kramer, "Ensembles of Nested Dichotomies for Multi-Class Problems," in *Proc. 21st Int'l Conf. Machine Learning (ICML '04)*, Banff, Alberta, Canada, p. 39, 2004, doi: https://doi.org/10.1145/1015330.1015363

[3] F. Sebastiani, "Machine learning in automated text categorization," in *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1-47, Mar. 2002, doi: https://doi.org/10.1145/505282.505283

[4] A. Sun, E. P. Lim, and W. K. Ng, "Hierarchical text classification methods and their specification," in *Cooperative internet computing*, The Springer International Series in Engineering and Computer Science, vol 729. Springer, Boston, MA, USA: 2003, ch. 14, pp. 236-256, doi: https://doi.org/10.1007/978-1-4615-0435-1_14

[5] S. Dumais and C. Hao, "Hierarchical classification of web content," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, Athens, Greece. 2000, pp. 256-263, doi: https://doi.org/10.1145/345508.345593

[6] I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski, "Hierarchical annotation of medical images," *Pattern Recognition,* vol. 44, no. 10-11, pp. 2436-2449, Oct. 2011, doi: https://doi.org/10.1016/j.patcog.2011.03.026

[7] L. Li, S. Jiang and Q. Huang, "Learning Hierarchical Semantic Description Via Mixed-Norm Regularization for Image Understanding," *IEEE Transactions on Multimedia*, vol. 14, no. 5, pp. 1401-1413, Oct. 2012, doi: https://doi.org/10.1109/TMM.2012.2194993

[8] J. Spehr, D. Rosebrock, D. Mossau, R. Auer, S. Brosig and F. M. Wahl, "Hierarchical scene understanding for intelligent vehicles," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, Baden-Baden, Germany, 2011, pp. 1142-1147, doi: https://doi.org/10.1109/IVS.2011.5940566

[9] P. Arbeláez, M. Maire, C. Fowlkes and J. Malik, "Contour Detection and Hierarchical Image Segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898-916, May 2011, doi: https://doi.org/10.1109/TPAMI.2010.161

[10] H. Blockeel, L. Schietgat, J. Struyf, S. Dzeroski and A. Clare, "Decision trees for hierarchical multilabel classification: A case study in functional genomics", in *Proc. 10th Eur. Conf. Principle Pract. Knowl. Discovery Databases: Lecture Notes in Computer Science()*, vol 4213. Springer, Berlin, Heidelberg, doi: https://doi.org/10.1007/11871637_7

[11] A. Breschi, M. Muñoz-Aguirre, V. Wucher, C. A. Davis, D. Garrido-Martín, S. Djebali, J. Gillis et al., "A limited set of transcriptional programs define major cell types," *Genome research*, vol. 30, no. 7, pp. 1047-1059, Jul. 2020, doi: https://doi.org/10.1101/gr.263186.120

[12] K. Punera, S. Rajan and J Ghosh, "Automatically learning document taxonomies for hierarchical classification," *Special interest tracks and posters of the 14th international conference on World Wide Web*, pp. 1010-1011, 2005, doi: https://doi.org/10.1145/1062745.1062843

[13] M. Sun, W. Huang and S. Savarese, "Find the Best Path: An Efficient and Accurate Classifier for Image Hierarchies," *2013 IEEE International Conference on Computer Vision*, Sydney, NSW, Australia, 2013, pp. 265-272, doi: https://doi.org/10.1109/ICCV.2013.40

[14] H. Lei, K. Mei, N. Zheng, P. Dong, N. Zhou, and J. Fan, "Learning group-based dictionaries for discriminative image representation," *Pattern Recognition*, vol. 47, no. 2, pp. 899-913, Feb. 2014, doi: https://doi.org/10.1016/j.patcog.2013.07.016

[15] V. Melnikov and E. Hüllermeier, "On the effectiveness of heuristics for learning nested dichotomies: An empirical analysis", *Mach. Learn.*, vol. 107, no. 8, pp. 1537-1560, Sep. 2018, doi: https://doi.org/10.1007/s10994-018-5733-1

[16] P. del Moral, S. Nowaczyk, A. Sant'Anna, S. Pashami, "Pitfalls of assessing extracted hierarchies for multi-class classification," Pattern Recognition, vol. 136, pp.109-225, Apr. 2023, doi: https://doi.org/10.1016/j.patcog.2022.109225

[17] K. Punera, S. Rajan and J. Ghosh, "Automatic Construction of N-ary Tree Based Taxonomies," *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, Hong Kong, China, 2006, pp. 75-79, doi: https://doi.org/10.1109/ICDMW.2006.35

[18] B. Larsen, C. Aone,"Fast and effective text mining using linear-time document clustering," in *Proc.5th ACM SIGKDD (KDD'99)*, San Diego, CA, USA, 1999, pp. 16–22, doi: https://doi.org/10.1145/312129.312186

[19] T. Li, S. Zhu and M. Ogihara,"Hierarchical document classification using automatically generated hierarchy," *J. Intell. Inf. Syst.*, vol. 29, no. 2, pp. 211-230, Oct. 2007, doi: https://doi.org/10.1007/s10844-006-0019-7

[20] C. Alagoz,"Performance Improvement in Multi-class Classification via Automated Hierarchy Generation and Exploitation through Extended LCPN Schemes," arXiv preprint arXiv:2310.20641, Oct. 2023, doi: https://doi.org/10.48550/arXiv.2310.20641

[21] R.A. Fisher, "The Use of Multiple Measures in Taxonomic Problems", *Ann. Eugenics*, vol. 7, pp. 179-188, Sep. 1936, doi: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

[22] L. Kaufman and P. J. Rousseeuw,"Finding Groups in Data: An Introduction to Cluster Analysis," New York, NY, USA:Wiley, 1990, doi: https://doi.org/10.1002/9780470316801

[23] J. H. Friedman, "Greedy function approximation: A gradient boosting machine", *Ann. Statist.*, pp. 1189-1232, Oct. 2001.

[24] J. Vanschoren, J. N. van Rijn, B. Bischl and L. Torgo, "OpenML: Networked science in machine learning", *ACM SIGKDD Explor. Newslett.*, vol. 15, no. 2, pp. 49-60, Jun. 2014, doi: https://doi.org/10.1145/2641190.2641198

[25] M. Feurer et al., "OpenML-Python: An extensible Python API for OpenML," *J. Mach. Learn. Res.*, vol. 22, no. 100, pp. 1-5, 2021.

[26] F. Wilcoxon, "Individual Comparisons by Ranking Methods," in: Kotz, S., Johnson, N.L. (eds) Breakthroughs in Statistics. Springer Series in Statistics. Springer, New York, NY, 1992, doi: https://doi.org/10.1007/978-1-4612-4380-9_16

[27] S. Godbole, "Exploiting confusion matrices for automatic generation of topic hierarchies and scaling up multi-way classifiers," *Technical report*, IIT Bombay, 2002.

[28] S. Bengio, J. Weston and D. Grangier, "Label embedding trees for large multi-class tasks," *Proc. Neural Inform. Process. Syst.*, pp. 163-171, 2010.

[29] D. Silva-Palacios, C. Ferri and M. J. Ramírez-Quintana, "Probabilistic class hierarchies for multiclass classification", *J. Comput. Sci.*, vol. 26, pp. 254-263, May 2018, doi: https://doi.org/10.1016/j.jocs.2018.01.006