T.C.
İSTANBUL KÜLTÜR ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MİKROSERVİS EKOSİSTEMİNDE SERVİS KEŞFİ MEKANİZMASI

YÜKSEK LİSANS TEZİ

Ahmet Vedat TOKMAK

2000005655

Anabilim Dalı: Bilgisayar Mühendisliği

Programı: Bilgisayar Mühendisliği

Tez Danışmanı: Doç. Dr. Akhan Akbulut

OCAK 2023

T.C.
İSTANBUL KÜLTÜR ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MİKROSERVİS EKOSİSTEMİNDE SERVİS KEŞFİ MEKANİZMASI

YÜKSEK LİSANS TEZİ

Ahmet Vedat TOKMAK

2000005655

Anabilim Dalı: Bilgisayar Mühendisliği

Programı: Bilgisayar Mühendisliği

Tez Danışmanı: Doç. Dr. Akhan Akbulut

Jüri Üyesi: Prof. Dr. Özgür Koray Şahingöz

Jüri Üyesi: Dr. Öğr. Üyesi İsmail Koç

OCAK 2023

ÖNSÖZ

Tez çalışmamdaki katkıları ve desteğinden dolayı değerli tez danışmanım Doç. Dr. Akhan AKBULUT'a çok teşekkür ederim. Her zaman yanımda olan özellikle çıkmaza girdiğim zamanlarda desteğini hissettiren sevgili eşim Elif NASIR TOKMAK'a çok teşekkür ederim.

Ocak 2023

Ahmet Vedat TOKMAK

İÇİNDEKİLER

KISALTMALAR

SOA	: Servis Odaklı Mimari (Service Oriented Architecture)
SVM	: Destek Vektör Makinesi (Support Vector Machine)
DNS	: Alan Adı Sunucusu (Domain Name Server)
IP	: İnternet Protokolü
RF	: Rassal Orman (Random Forest)
LDA	: Latent Dirichlet Allocation
BTM	: Bi-Term Konu Modeli (Bi-Term Topic Model)
WSDL	: Web Servisleri Açıklama Dili (Web Services Description Language)
OWL	: Web Ontoloji Dili (Web Ontology Language)
DT	: Karar Ağacı (Decision Tree)
KNN	: K-en Yakın Komşular (K-Nearest Neighbors)
TS	: Tabu Arama (Tabu Search)
SLR	: Sistematik Literatür Taraması (Systematic Literature Review)
LSTM	: Uzun Kısa Vadeli Hafıza Ağları (Long- Short Term Memory)
bi-LSTM	: Çift Yönlü LSTM
QWS	: Web Servis Kalitesi (Quality Of Web Service)
SHAP	: Shapley Katkısı Açıklamaları (Shapley Additive Explanations)
ROC	: İşlem Karakteristik Eğrisi (Receiver Operating Characteristics)
AUC	: ROC Eğrisinin Altında Kalan Kısım (Area Under The Curve)
API	: Uygulama Geliştirme Arayüzü (Application Programming Interface)

TABLO LİSTESİ

ŞEKİL LİSTESİ

Üniversite : **T.C. İstanbul Kültür Üniversitesi**
Enstitü : **Lisansüstü Eğitim Enstitüsü**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Program : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Doç. Dr. Akhan AKBULUT**
Tez Türü ve Tarihi : **Yüksek Lisans – Ocak 2023**

ÖZET

MİKROSERVİS EKOSİSTEMİNDE SERVİS KEŞFİ MEKANİZMASI

Günümüzde teknolojinin hızla gelişmesi ile yazılım-yoğun sistemler her zamankinden daha fazla hayatımıza dahil olmakta, bu sistemlerde çoğunlukla tercih edilen monolitik yazılım mimarisinin ihtiyacı karşılamakta yetersiz kaldığı görülmektedir. Servis Odaklı Mimari (SOA), uygulama geliştirme dili, platform bağımsız kullanımı ve yüksek ölçeklenebilirlik avantajları nedeniyle monolitik mimari yerine tercih edilmeye başlanmıştır. SOA'nın en güncel uygulaması olan mikroservis mimarisinin yazılım mimarisi olarak kullanımının yaygınlaşması, mikroservisler için keşif problemini beraberinde getirmiştir. Mikroservislerin etkin kullanımı için ilk olarak erişilmek istenen mikroservise ait IP ve Port bilgilerine takiben mikroservisin ilgili yazılım bileşeninin aktif olup olmadığı bilgisine ihtiyaç vardır. Aynı servisi sunan çok sayıda mikroservis tespit edilmesi durumunda, mikroservisler arasından hizmet kalitesi en yüksek olanın seçilmesi gerekir. Bir mikroservisin kalitesi; başarı, verim, gecikme zamanı, tepki süresi gibi belirli parametrelerle belirlenir. Bu çalışma kapsamında mikroservis kalitesinin tahmin edilebilmesi için sistematik literatür taramasıyla yapılan çalışmalarda öne çıkan SVM, Karar Ağacı, Rassal Orman, KNN ve Naive Bayes sınıflandırma algoritmalarının etkili olduğu gözlemlenmiştir. Yaptığımız araştırma çalışmasının bir diğer bulgusu olarak; ilgili algoritmalarla birlikte önerilen Gradyan Artırma, XGBoost, LightGBM ve CatBoost yükseltme algoritmalarını kullanan ampirik çalışmalar yapılmıştır. Geliştirilen modellerin en uygun hiperparametre değerlerinin tespit edilmesi için Grid Search, Random Search, Bayes Search, Halvin Grid Search ve Halvin Random Search olarak beş farklı yöntem kullanılmıştır. Deneylerde gerçek dünyadan elde edilen 2507 mikroservise ait trafik

verisini barındıran QWS veriseti kullanılmıştır. Mikroservis kalitesinin tahmin edilmesinde en iyi sonuç %90.42'lik genel doğruluk oranı ile CatBoost algoritmasıyla elde edilmiştir.

Anahtar Kelimeler: Mikroservis, Servis Keşfi, Yükseltme Algoritması, CatBoost, LightGBM, XGBoost, Gradyan Artırma

University	**: T.C. İstanbul Kültür University**
Institute	**: Institute of Sciences**
Department	**: Computer Engineering**
Program	**: Computer Engineering**
Thesis Advisor	**: Assoc. Prof. Akhan AKBULUT**
Degree Awarded And Date	**: MA – Ocak 2023**

ABSTRACT

SERVICE DISCOVERY MECHANISM IN THE MICROSERVICE ECOSYSTEM

Currently, the rapid advancement of technology has led to the expansion of information systems, and the monolithic software design was previously inadequate to satisfy the requirements. SOA (Service Oriented Architecture) has begun to be favored over monolithic design as a result of its scalability and independence from language and platform usage. The rapid expansion in the number of microservices, which is the result of SOA deployment, has brought about the microservices discovery challenge. The IP address and port number of the microservice, as well as whether or not the microservice is running, are required to utilize microservices. In the event that there are numerous microservices, the one with the highest quality should be chosen. Several characteristics impact the success of a microservice, including efficiency, delay time, and reaction time. Under the scope of this study, the classification algorithms SVM, Decision Tree, Random Forest, KNN, and Naive Bayes, which are popular in systematic literature reviews, were determined to predict microservice quality. Experiments were conducted utilizing the proposed boosting algorithms Gradient Boosting, XGBoost, LightGBM, and CatBoost as well as the determined methods. Grid Search, Random Search, Bayes Search, Halvin Grid Search, and Halvin Random Search were employed to identify the optimal hyperparameter values for their algorithms. In the studies, the QWS dataset containing data from 2507 real-world microservices was utilized, and the CatBoost algorithm yielded the best accuracy rate of 90.42 percent.

Keywords: Microservice, Service Discovery, Boosting Algorithm, CatBoost, LightGBM, XGBoost, Gradient Boosting

1. GİRİŞ

Günümüzde bilişim teknolojilerinin kullanımının her geçen gün artması beraberinde kullanılan uygulamaların kapasitesinin de artmasına imkân sağlamaktadır. Uygulamaların monolitik yapıda tasarlanması, küçük ölçekli uygulamalar için avantaj olarak görünse de orta ve büyük ölçekli uygulamalarda bu yaklaşım ölçeklenebilirlik sorunlarına sebebiyet vermektedir. Servis odaklı mimari (Service Oriented Architecture-SOA) uygulamaların tek bir bileşen yerine farklı görevlere odaklanmış küçük iş parçacıklarına ayrılması yaklaşımıdır. SOA, dil ve platform kullanımı açısından bağımsız oluşu ve daha iyi ölçeklenebilir olmasından kaynaklı olarak gün geçtikçe daha çok tercih edilmektedir. SOA, monolitik mimarinin sınırlarıyla ilgili sorunları ele almak için şu anda mikroservisleri kapsayacak şekilde gelişmektedir [1]. Kullanılan servislerin başka uygulamalarda tekrar kullanılabilir oluşu sayesinde zaman ve maliyet tasarrufu sağlanmaktadır. Servislerin birbirinden bağımsız kuruluma sahip oluşu ise servislerin birinde ortaya çıkacak bir sorunda diğer servislerin etkilenmemesini veya minimum düzeyde etkilenmesini, bakım ve hata giderme sürecinin de daha kolay olmasını sağlamıştır.

1.1 Problem Tanımı

İnternet ağında yaşanan gelişmeler sonucunda kullanılan mikroservislerin sayısı da büyük bir hızla artmaktadır. Büyük uygulamalar, farklı işlevleri yerine getiren birçok mikroservisten oluşur, bu nedenle her mikroservis için doğru dili seçmek, tüm uygulamayı tek bir dille yazmaktan daha kolay ve verimlidir. Bu özelliğinden dolayı mikroservisler geliştiriciler tarafından daha çok tercih edilmektedir. Fakat geniş internet ağı ve web servislerinin dinamik yapısı beraberinde temel problem olan servis keşfini getirmektedir. Servis keşfi mikroservislerin en temel problemidir. Servis keşfi genel olarak üç kavram üzerine odaklanır: Bunlardan birincisi keşif (Discovery); servisin konumu yani ip ve port bilgisi, ikincisi durum kontrolü (Healthcheck); servisin çalışıp çalışmadığı ve son olarak yük dengeleme (Load Balancing); gelen isteğin en uygun durumdaki servise yönlendirilmesidir. Servis keşif sürecinin performansını artırarak en uygun servisin en hızlı şekilde

bulunmasını sağlamak için en çok tercih edilen yöntem ve teknikler benzer özellikteki servislerin sınıflandırılması ve kümelenmesidir [2].

Bilişim teknolojilerindeki hızlı gelişim beraberinde servis keşfinde daha önce kullanılan yaklaşımların ihtiyacı tam olarak karşılayamamasına sebep olmuştur. Sürekli büyüyen ve anlık olarak değişen mikroservis ekosisteminde kullanılan sınıflandırma yaklaşımlarının da sürekli güncellenmesi veya yeni yaklaşımların bulunması gerekmektedir. Özellikle e-ticaret, finans ve bankacılık gibi mikroservislerin sayısının, durumunun ve konumlar anlık olarak değişebildiği büyük sektörlerde kullanıcı memnuniyeti için doğru servisin bulunması kadar bunun için geçen sürede çok önemlidir. Bu tez konusunun seçilmesindeki en büyük motivasyon bu ihtiyacın karşılanmasına yönelik çalışmaların sınırlı sayıda olmasıdır.

Literatürde mikroservisler için kullanılan keşif yaklaşımları incelendiğinde en çok üzerinde çalışılan konunun benzer işlevi yerine getiren servisler arasında tercih yapmak olduğu görülmektedir. Örneğin bir e-ticaret sitesinde ödeme işlemi için birden fazla servis kullanıldığını varsayalım. Ödeme işlemi gerçekleştirileceği zaman kullanıcının mevcut servislerden hangisine yönlendirileceğinin bilinmemesi temel sorunu oluşturmaktadır. Bu durumda aynı görevi yerine getiren servislerin sınıflandırılarak belirlenmesi ve belli ölçütlere göre içlerinden en iyi hizmet kalitesine sahip olanın tahmin edilmesi gerekmektedir. Yapılan tahminin doğruluğu ve hızı, servis keşfinde kullanılan yaklaşımın başarısını göstermektedir. Daha önce yapılan çalışmalarda birçok sınıflandırma yöntemi kullanılmış ve bunlardan iyi sonuçlar elde edilmiştir. Genelde en yaygın kullanılan algoritmalar Lojistik Regresyon, Rassal Orman (Random Forest-RF) [3], Destek Vektör Makinesi (Support Vector Machine-SVM) [4], Karar ağacı (Decision Tree- DT) [2] ve Naive Bayes'dir [5]. Bu algoritmalar ile yapılan deneyler sonucunda en iyi sonuç Rassal Orman algoritması ile elde edilmiştir. Kullanılan yöntem ve algoritmalar farklı olsa da temel amaç servislerin en iyi şekilde sınıflandırılması ve kullanılacak servisin kalitesinin doğru olarak tahmin edilmesidir.

Bu çalışmanın motivasyonu, mikroservis ekosisteminde servisin kalitesinin tahmin edilerek, kullanıma alınacak servislerin belirlenmesidir. Sınıflandırma probleminin çözümünde yükseltme (Boosting) algoritmaları kullanılarak sonuçların iyileştirilmesi hedeflenmektedir. Yükseltme algoritmaları Rassal Orman algoritması

gibi temelde bir topluluk öğrenmesi yöntemidir. Rassal Orman algoritmasından farklı olarak yükseltme algoritmalarında sınıflandırıcılar bir önceki sınıflandırıcının tahminini dikkate alarak yeni tahmin üretir ve bu süreç modelden daha fazla gelişme kaydedilemeyinceye kadar devam eder [6]. Gradyan Artırma, XGBoost, LightGBM ve CatBoost gibi yükseltme algoritmaları servis keşfi sürecinde servis kalitesinin tahmin edilmesi aşamasında daha iyi doğruluk oranları sağlanması amaçlanmaktadır. Bu algoritmaları güçlü kılan en önemli özellik, daha önceki tahminde yapılan hataları en aza indirgeme eğiliminde olmasıdır.

1.2 Tezin Amacı

Bu çalışmada makine öğrenmesi yöntemleri ve 2.507 gerçek web servis uygulaması için bir dizi ölçümden oluşan QWS [7] veriseti kullanılarak motivasyonumuz doğrultusunda servis kalitesini en iyi şekilde tahmin etmek için deneyler yapılmıştır. Literatür taramasında öne çıkan beş algoritma ile dört yükseltme algoritmasından elde edilen sonuçlar karşılaştırılarak en yüksek başarımı sunan yapay öğrenme modeli bulunacaktır. Bu çalışmada kabul edilmesi ve reddedilmesi gereken hipotezler aşağıdaki gibidir.

H0: Servis kalitesi tahmininde kullanılacak yapay öğrenme modelinin %90 ve üzeri doğruluk oranı ile tahminde bulunması.

H1: Servis kalitesi tahmininde kullanılacak yapay öğrenme modelinin %90 ve üzeri doğruluk oranı ile tahminde bulunamaması.

H0 ve H1'in doğrulamasını yapmak için tezin yöntem kısmında detaylandırılan dokuz araştırma sorusu belirlenmiş olup, bu soruların yanıtlarını arayan çalışmalar perspektifinde tez çalışmaları gerçekleştirilmiştir.

1.3 Tezin Organizasyonu

Bu tez altı bölümden oluşmaktadır. Birinci bölümde, problemin tanımı yapılmış ve araştırmanın amacı ve önemi anlatılmıştır. İkinci bölümde, tezin konusu olan mikroservislere ait temel kavramların tanımı ve daha önce yapılan çalışmalardan bahsedilmiştir. Üçüncü bölümde, literatürde yayınlanmış yayınlar üzerinde yapılan sistematik literatür taramasının sonuçlarından, önerilen yöntem olan yükseltme algoritmalarının özelliklerinden, deneylerde kullanılan verisetinden ve deneylerde

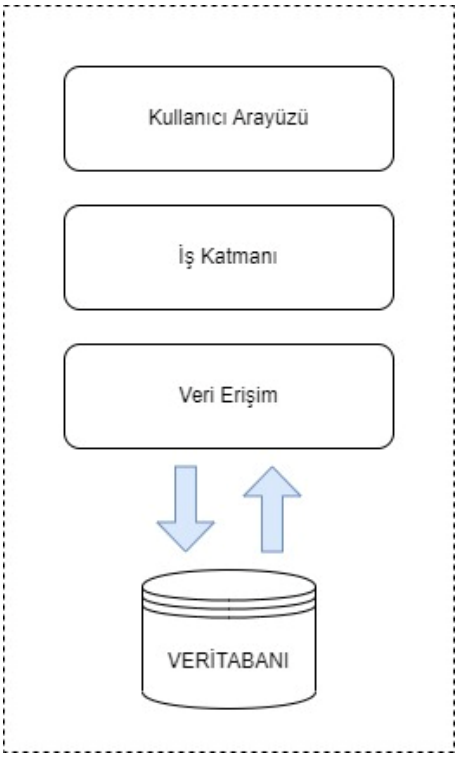
kullanılan hiperparametre ayarlama yöntemlerinden bahsedilmiştir. Dördüncü bölümde literatürden seçilen ve önerilen yükseltme algoritmalarına ait sonuçlar anlatılmıştır. Beşinci bölümde yapılan deneyler üzerine tartışma yer almış; altıncı bölümde ise elde edilen bulgular üzerinden sonuçlar ele alınmıştır.

2. TEMEL BİLGİLER VE LİTERATÜR TARAMASI

Bu bölümde tez kapsamındaki çalışmalarda kullanılan temel kavramlar ele alınmıştır. Bunlardan ilki servis odaklı mimari geliştirmeden önce kullanılan monolit mimaridir. Monolit mimarinin güçlü ve zayıf yönleri detayları ile anlatılmıştır. İkinci olarak ise çalışmamıza temel oluşturan mikroservislerin dayanağı olan SOA'nın çalışma mantığı ile avantaj ve dezavantajları ortaya konmuştur. Daha sonra mikroservis keşif aşamasının çalışmasına yönelik kavramlar olan mikroservis ekosistemi, mikroservis keşfi ve servis kaydının detayları anlatılmıştır.

2.1 Monolitik mimari

Katmanlı mimariyi yaklaşımını kullanan bir uygulama genellikle üç seviyeden oluşur. Bunlar son kullanıcı seviyesinden itibaren: Kullanıcı arayüzü, iş katmanı ve veri erişim arayüzüdür. Kullanıcı arayüzü, kullanıcının etkileşimde bulunduğu bölümken iş katmanı ise kullanıcı isteklerini yerine getiren algoritmaları barındırır. Veri erişim arayüzü ise yapılan işlemler sonucu oluşan verilerin depolandığı bölüm olarak tanımlanır. Monolotik mimari birden çok bileşen içeren tek bir kod tabanına sahip mimaridir [8]. Tüm uygulamayı ayağa kaldıran tek bir derleme dosyası bulunur. Şekil 2.1, bu modelin yapısını göstermektedir.



Şekil 2.1. Monolitik Mimari

Monolotik Mimarinin Avantajları

- Geliştirmesi basit: Geliştiriciler için mikroservislere oranla daha kolaydır.
- Daha kolay hata ayıklama ve uçtan uca test etme: Monolitik bir uygulama tek bir bölünmez proje olduğundan, uçtan uca testleri çok daha hızlı şekilde gerçekleştirilebilir [9].
- Kolay dağıtım: Monolitik uygulamaların tek bir parça oluşu ile bağlantılı bir avantajı ise kolay dağıtımdır. Tek bir parçayı dağıtmak onlarca servisi dağıtmaktan çok daha kolaydır [10].

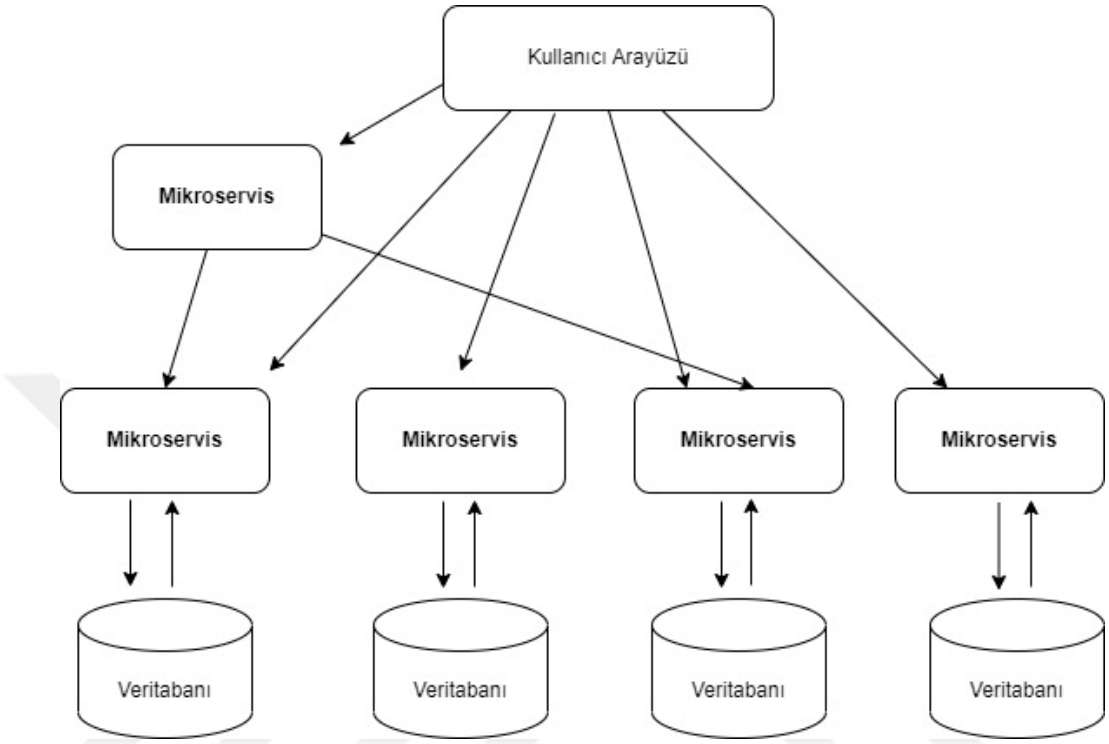
Monolitik Mimarinin Dezavantajları

- Karmaşıklık: Bir monolitik uygulama büyüdükçe kullanılan kod miktarının artmasıyla kullanılan kodların anlaşılması zorlaşmaktadır.
- Yüksek bakım maliyeti: Yazılan her kod tüm uygulamayı etkileyeceği için uygulama üzerinde değişiklik yapmak zordur.
- Ölçeklenebilirlik (Scalable) sınırlı: Bileşenleri bağımsız olarak ölçekleyemezsiniz. Tüm uygulamayı ölçeklemeniz gerektiği için darboğazlardan etkilenir [10].
- Yeni teknoloji entegre etmekte sınırlı: Monolitik bir uygulamaya yeni bir teknolojinin entegre edilmesi tüm uygulamayı etkileyeceği için oldukça zahmetli bir iştir. Küçük bir alanda değişiklik yapılacak olsa bile monolit yapıdan dolayı yapılacak değişiklik diğer bölümleri de etkileyecektir hatta bu değişiklikler birçok alanda kayba sebep olabilir [9].
- Başlatma süresi: Uygulama büyüdükçe yapılan değişiklikler sonrası uygulamayı yeniden başlatma süresi de uzamaktadır.

2.2 Servis Odaklı Mimari (SOA)

Servis Odaklı Mimari, günümüzde kullanılan uygulamaların hızla büyümesi ve kullanılan teknolojilerin değişmesi ve gelişmesiyle birlikte yazılım mimarisinde en yaygın kullanılan yöntemdir. SOA'daki amaç monolitik uygulama oluşturmak yerine uygulamanın daha küçük, birbirine bağlı servislere bölmektir. SOA şu anda monolitik mimarinin sınırlarıyla ilgili sorunları ele almak için mikroservisleri kapsayacak şekilde gelişmektedir [1]. Mikroservis mimarisinde Şekil 2.2'de görüldüğü üzere tüm işlemleri yapan tekbir iş katmanı yerine birbirinden bağımsız farklı işlevleri yerine getiren mikroservisler bulunmaktadır. Mikroservislerin kullandığı veritabanları

yapılan işlemlere göre aynı olabileceği gibi farklı da olabilir. Birbirinden bağımsız olan mikroservislerin bu yapısı, herhangi bir servisteki arızanın sistemin bütününü etkilemesi yerine o servisle sınırlı kalmasını sağlar ve servislerin geliştirilmesini de kolaylaştırır.



Şekil 2.2. Mikroservis Mimarisi

Servis Odaklı Mimarinin Avantajları

- Bağımsız bileşenler: Her bir servis bağımsız bir şekilde dağıtılabilir ve güncellenebilir yapısı daha fazla esneklik sağlar. Bir serviste ortaya çıkacak bir hata uygulamanın genelini etkilemez. Uygulamaya yeni bir özellik eklemek veya uygulamadaki mevcut bir özelliği değiştirmek monolotik uygulamalara kıyasla daha kolaydır.
- Anlaşılabilirlik: Daha küçük ve daha basit bileşenlere ayrılan bir mikroservis uygulamasının anlaşılması ve yönetilmesi daha kolaydır [9].
- Yüksek ölçeklenebilirlik: Her bir servis bağımsız olarak ölçeklendirilebilir. Bu nedenle tüm uygulamanın ölçeklenmesi gerekmez [10]. Uygulamanın belli bir alanında yani tek bir mikroserviste yapılan güncelleme sonrası sadece o servisin ölçeklenmesi yeterli olacağından kaynak tüketimi noktasında tasarruf sağlar.

- Teknoloji Çeşitliliği: Uygulamanın monolotik mimaride olduğu gibi tek bir platformda geliştirilme zorunluluğu yoktur. Geliştirme takımları servislerin geliştirileceği teknolojileri bağımsız olarak polygloth yapıda uygularlar.
- Yeniden kullanılabilirlik: Bağımsız olarak geliştirdiğiniz bir servisi ihtiyacınıza göre ufak değişiklikler ile başka bir projede de kullanılabilir. Örneğin projenizde kullandığınız bir servisi başka bir projenizde direkt olarak konumlandırabilir ve kullanılabilirsiniz. Böylece bir servis geliştirmede zamandan ve maliyetten tasarruf edebilirsiniz.

Servis Odaklı Mimarinin Dezavantajları

- Orkestrasyon karmaşıklığı: Bir mikroservis mimarisi dağıtık bir sistem olduğundan, tüm modüller ve veritabanları için ayrı ayrı yapılandırma yapmanız gerekir [10].
- Sistem dağıtımı: Bir mikroservis mimarisi, birden fazla servis ve veritabanından oluşan karmaşık bir yapıdır. Bu nedenle tüm bağlantıların dikkatle ele alınması ve her servis için ayrı bir dağıtım süreci gerektirir.
- Yönetim ve izlenebilirliğin zorluğu: Bir mikroservis uygulaması oluştururken birden fazla durumla ile ilgilenmeniz gerekecektir. Harici yapılandırmayı, ölçütleri takip edebilme, durum kontrol mekanizmaları ve mikroservislerin yönetilebildiği ortamlar gerekir.
- Test etmenin zorluğu: Bağımsız olarak konuşlandırılan birçok servis test sürecini oldukça zorlaştırır. Geleneksel test yöntemleri yerine kaos mühendisliği uygulanır ki bu daha maliyetli bir yaklaşımdır.

Küçük ölçekli veya küçük bir kitleye hitap eden uygulamalarda monolitik mimari uygun olsa da orta ve büyük ölçekli projeler için mikroservis mimarisi tercih edilmelidir. Mikroservis mimarisinin dezavantajları olsa da sağladığı avantajlar dikkate alındığında bu dezavantajlar göze alınabilir. Özellikle sürekli ve büyük bir hızla gelişen teknoloji servis odaklı mimariyi vazgeçilemez hale getirmektedir.

2.3 Mikroservis Ekosistemi

Mikroservis ekosistemi monolitik yapılardan mikroservis mimarilerine geçişler sonucunda oluşmaktadır. Yüksek ölçeklenebilirlik yapısı, kolay ve hızlı ürünlerin oluşması, yönetiminin parçalara ayrılmasından dolayı mimarilerin en önemli

avantajlarıdır [10]. Mikroservislerin en fazla kullanıldığı sektörler; yazılım, finans, danışmanlık şirketleri, e-ticaret firmaları, telekomünikasyondur. Bir proje büyüklüğüne göre yüzlerce mikroservisten oluşabilir. Yüzlerce mikroservisin oluşturduğu bu ekosistem beraberinde büyük bir sorun olan servis keşfini getirmektedir.

2.4 Mikroservis Keşfi

Mikroservis mimarisine geçişle beraber uygulamalar monolit olarak yapılmak yerine birbirinden bağımsız olarak çalışan tek bir işe odaklı mikroservisler şeklinde tasarlanmaya başlandı. Birbirinden bağımsız olarak ve farklı ekipler tarafından geliştirilen servislerin bir bütünü oluştururken birbirlerinden haberdar olmaları gerekir. Statik düzende bu çok büyük bir problem olarak görünmese de mikroservis mimarisinin getirmiş olduğu bağımsızlık ve ölçeklenebilirlik özellikleri süreci dinamik bir hala çevirmiştir. Örneğin uygulamadaki bir servisi oluşturan ekip servisin konumunu değiştirebilir veya kullanılan bir servisin ihtiyaca göre kopya sayıları artırılıp azaltılabilir [11]. Servislerin birbirleri ile haberleşebilmeleri için birbirlerinin konumlarını yani ip adreslerini ve port numaralarını bilmeleri gerekir. Servislerin birbirlerinin konumlarının bulma sürecine servis keşfi denir.

Mikroservis keşfinin ilk uygulaması servislerin konum bilgilerinin tek bir dosyaya kaydedilmesi yöntemidir. Fakat internetin hızlı bir şekilde büyümesi, servis sayılarının ve servislerin yaşam sürelerinin hızlı bir şekilde değişmesi kullanıcıları farklı teknolojiler ve stratejiler kullanmaya zorlamaktadır. Servis keşfi için genel olarak üç temel yaklaşım kullanılmaktadır. Birincisi mevcut alan adı sunucusu (Domain Name server-DNS) yapısını kullanmak, İkinci yaklaşım, Apache Zookeeper, Consul gibi mevcut güçlü tutarlı bir anahtar değer veri deposunu kullanmaktır. Son olarak Netfliks Eureka gibi servis keşfi için optimize edilmiş tasarımlar sunan teknolojiler kullanmaktır. Sürekli büyüyen mikroservis ekosisteminde servislerin haberleşmesi için birbirlerini bulmaları olmazsa olmazdır. Bulunamayan bir servisin varlığının bir önemi yoktur. Servis keşif süreci bu yüzden çok önemlidir. Service keşfi temel olarak üç kavram üzerinde durmaktadır. Bunlar:

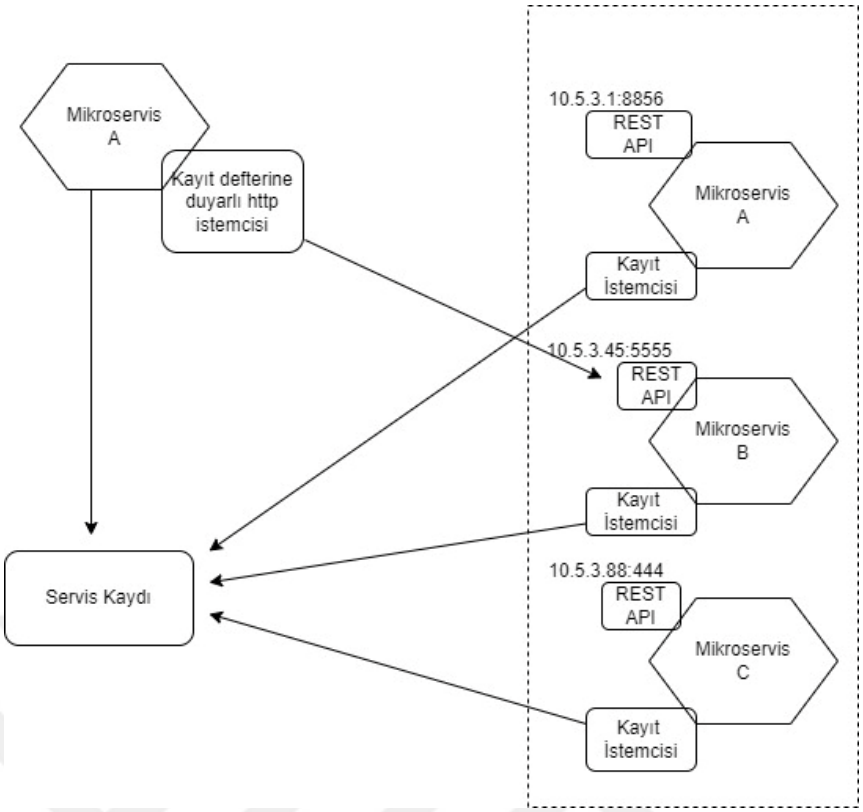
- **Keşif:** Servislerin dinamik bir ortamda diğer servisler ile iletişim kurabilmeleri için, birbirlerinin IP ve port bilgilerine ihtiyacı vardır. Keşif ise bunu sağlamaktadır.

- **Durum Kontrol:** Çalışan servislerin sistemde kalmasını çalışmayan servislerin ise dinamik bir şekilde sistem dışı kalmaları sağlanmaktadır.
- **Yük Dengeleme:** Bir servise gelmiş olan isteğin, aynı işi yapan servislere dinamik olarak dağıtılmasını sağlamaktır.

Mikroservis keşif sürecinde kullanılan iki tür model bulunmaktadır. Bunlardan birincisi kullanılabilir servislere ait ağ konumlarının ve servisler arasındaki yük dengeleme işlemlerinde sorumluluğun istemcide olduğu istemci taraflı servis keşfi, ikincisi ise bu işlemlerin bir yük dengeleyici ile yapıldığı sunucu taraflı servis keşfi modelidir. Bu modeller aşağıda detaylı olarak anlatılmıştır.

İstemci taraflı servis keşfi

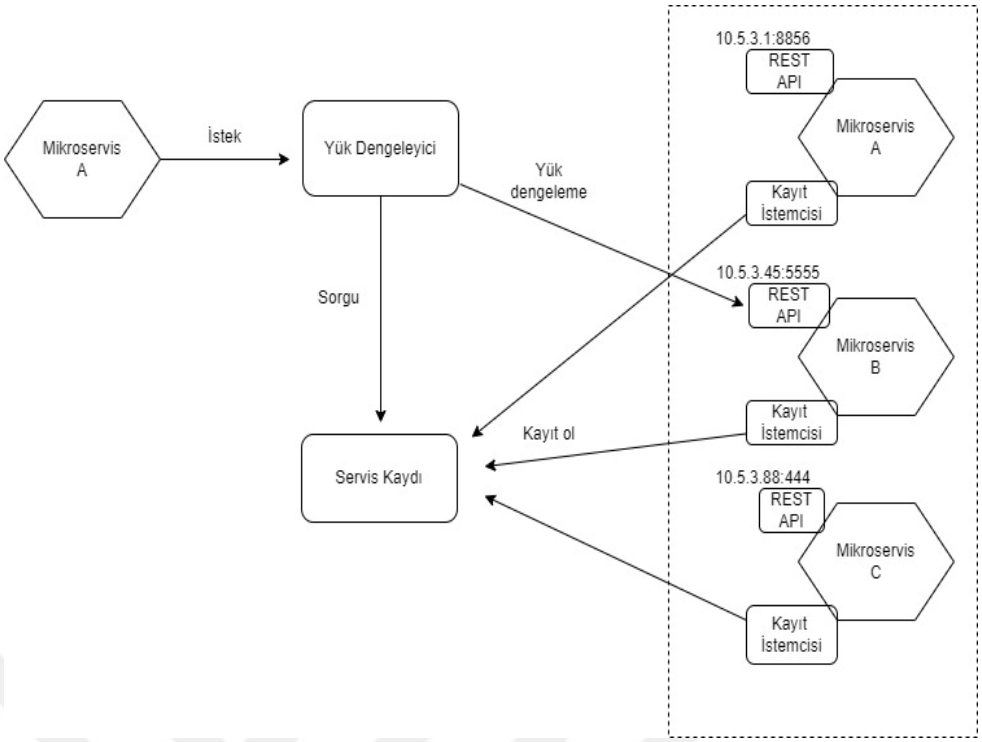
İstemci tarafında keşif kullanılırken, kullanılabilir servis örneklerinin ağ konumlarını ve bunlar arasında yük dengeleme isteklerini belirlemek istemcinin sorumluluğundadır [12]. Her servis başlatıldığında bir servis kayıt defterine kaydedilir ve periyodik olarak her servisin çalışıp çalışmadığı kontrol edilir. Çalışan servisler kayıt defterinde kalmaya devam ederken çalışmayan servisler kayıt defterinden silinir. İstemci, kullanılabilir servislerin bir veritabanı olan servis kayıt defterini sorgular. Daha sonra mevcut servisler arasında en uygun servisi bulmak için bir yük dengeleme algoritması kullanır [11] ve bir istekte bulunur. Şekil 2.3'de görüldüğü üzere istemci taraflı servis keşfinde, tüm servisler önce bir servis kayıt defterine kaydedilir. Mikroservislerden biri başka bir servisle iletişime geçmeden önce servis kayıt defterine gidip iletişim kurmak istediği servise ait bilgileri aldıktan sonra bu servisle iletişime geçer.



Şekil 2.3. İstemci taraflı servis keşfi mimarisi

Sunucu taraflı servis keşfi

İstemci, bir yük dengeleyici aracılığıyla bir servise istekte bulunur. Yük dengeleyici, servis kayıt defterini sorgular ve her isteği kullanılabilir bir servis örneğine yönlendirir [11]. Burada da istemci taraflı keşifte olduğu gibi her servis aktif olunca servis kayıt defterine kaydedilir ve pasif olunca buradan silinir. Sunucu taraflı keşfin en büyük yararı istemciyi servis keşif sürecinden soyutlamasıdır. İstemci sadece yük dengeleyici ile iletişime geçer ve isteğini ona iletir. Şekil 2.4'te görüldüğü üzere sunucu taraflı keşifte de tüm servisler öncelikle servis kayıt defterine kaydedilir. İstemci taraflı keşiften farklı olarak bir servis başka bir servisle iletişime geçmek istediği zaman direkt olarak servis kaydına gitmez bunun yerine arada servisler arasında yönlendirmeyi sağlayan bir yük dengeleyiciyle iletişime geçer. Yük dengeleyici servis kayıtlarına bakarak kendisine gelen isteği en uygun servise yönlendirme işini yapar.



Şekil 2.4. Sunucu taraflı servis keşif mimarisi

2.5 Servis Kaydı

Servis keşif sürecinin önemli bir parçasıdır. Servis kayıt defteri mikroservis ekosisteminde olan servislerin en güncel konumlarını tutar. Servislerin konumları, aktif veya pasif durumları sürekli değişebildiği için servis kaydının kullanılabilir ve güncel olması gerekir. Bir servisin kaydının oluşması iki şekilde olur: Birincisi kendi kendine kayıt yani servis kendisin servis kayıt defterine kaydettirmekten ve silmekten sorumludur. İkincisi ise üçüncü taraf kayıt, servis kayıt şirketi olarak bilinen başka bir sistem bileşeni, kaydı işler [12]. Servis kayıt şirketi, sistemdeki değişiklikleri izler yeni kullanılabilir bir servis fark ettiğinde servis kayıt defterine kaydeder. Sonlandırılan servisleri ise kayıt defterinden siler.

Servis kayıtlarının örnekleri şunları içerir:

- Netflix Eureka: Servis örneklerini kaydetmek ve sorgulamak için bir REST API sağlar. POST Bir hizmet örneği, bir istek kullanarak ağ konumunu kaydeder. PUT Her 30 saniyede bir, bir istek kullanarak kaydını yenilemesi gerekir. Bir kayıt, bir HTTP DELETE isteği kullanılarak veya örnek kaydı zaman aşımına uğrayarak kaldırılır.

- Etcd: Paylaşılan yapılandırma ve servis keşfi için kullanılan, yüksek düzeyde kullanılabilir, dağıtılmış, tutarlı, anahtar/değer deposu. Etcd kullanan iki önemli proje Kubernetes ve Cloud Foundry'dir.
- Consul: Servisleri keşfetmek ve yapılandırmak için bir araç. İstemcilerin servisleri kaydetmesine ve keşfetmesine olanak tanıyan bir uygulama programlama arayüzü (Application Programming Interface-API) sağlar. Consul, servis kullanılabilirliğini belirlemek için sağlık kontrolleri yapabilir.
- Apache Zookeeper – Dağıtılmış uygulamalar için yaygın olarak kullanılan, yüksek performanslı bir koordinasyon hizmeti. Apache Zookeeper, başlangıçta Hadoop'un bir alt projesi olarak ilk kez tanıtıldı, ancak şimdi üst düzey bir proje olarak bağımsız bir üründür.

2.6 Literatür taraması

Literatürde bu alanda yapılan çalışmaları incelediğimizde servis keşif sürecinde araştırmacıların en çok benzer işi yapan servisler arasında en doğru servisi ulaşmaya çalıştığı görülmektedir. Avantajlarından dolayı daha çok tercih edilmeye başlanan mikroservis mimarisinde benzer işleri yapan servislerden en doğru olana en hızlı şekilde ulaşmak en büyük zorluktur. Bu aşamada araştırmacıların ilk başvurduğu yöntem benzer görevdeki servislerin sınıflandırılması ve kümelenmesidir.

Web servislerinin kalitesinin değerlendirilmesi, bir uygulama için bir web servisinin seçilmesinde büyük önem taşımaktadır [13]. Web servislerinin kalite özellikleri BPNN, PNN, GMDH, CART, J48, TreeNet ve SVM vb. sınıflandırıcılar kullanılarak benzer özellikteki servisleri en iyi sonucu elde etmek için sınıflandırılmıştır [14]. Araştırmacılar sınıflandırma ve kümeleme için farklı yaklaşımlar kullanarak en uygun servise ulaşma aşamasında çözümler üretmeye çalışmaktadır. Kalite parametrelerine göre dinamik olarak oluşturulan tanıma modülleri aracılığıyla web servislerinin sınıflandırılması yaklaşımı daha az karmaşık görevler kullanarak seçim problemine bir çözüm sağlar [15].

Servislerin seçimini kolaylaştırmak için servislerin özniteliklerini kullanmak en yaygın yöntem olmakla beraber bu yöntemde çok farklı teknikler kullanılmaktadır. Tabu Arama (Tabu Search-TS) algoritması, uyarlanabilir belleğin yanı sıra duyarlı keşif kullanır [16]. Komşu arama ile benzer şekilde başlayıp ve belirlenen bir kriteri karşılayana kadar bir noktadan diğerine yinelemeli olarak ilerler. Naive Bayes, Bayes

teorisine dayanan yaygın olarak kullanılan bir sınıflandırma yöntemidir [17]. Bir Bayes sınıflandırıcısı, bir sınıfın belirli bir özelliğinin diğer özellikler ile ilgisi olmadığını varsayar. Rassal Orman, çok sayıda karar ağacının bir araya getirilmesine dayanan bir sınıflandırma ve regresyon yöntemidir [18]. K-Means algoritması, bir kümenin ortalaması ile kümedeki noktalar arasındaki kare hatası hesaplayarak kümelemenin kalitesini belirler. Algoritmanın amacı, tüm K kümeleri üzerinde karesi alınmış hatanın toplamını en aza indirmektir [19]. Destek Vektör Makinesi, veri madenciliği alanında kullanılan bir makine öğrenmesi tekniğidir. Algoritma, doğrusal ve doğrusal olmayan ikili sınıflandırma problemini çözen denetimli bir öğrenme algoritması olarak tanımlanır. Görüldüğü üzere yapılan çalışmaların tamamı farklı teknikler ve algoritmalar kullanılsa da özniteliklere göre en iyi servis sınıflandırması yapmaktır.

Servislerin gruplanmasında ve sıralanmasında kullanılan diğer yaklaşımlar ise yapay sinir ağları [20], yapay arı kolonisi algoritması [21] ve farklı yaklaşımların hibrit [22] olarak kullanıldığı yaklaşımlardır. Bütün yaklaşımlarda ortak amaç en doğru servise en hızlı şekilde ulaşmak için servislerin gruplanması ve sıralanmasıdır. Servis sayısını hızla artışı ve dinamik bir ortam olmasından dolayı servis keşfi üzerine yapılan çalışmalar devam edecektir.

3. YÖNTEM

Bu bölümde servis keşfi için belirlenen temel araştırma soruları ile bu sorunlara karşı bulunan veya önerilen çözüm önerileri anlatılmıştır. Karşımıza çıkan ve cevaplanması gereken dokuz soru Tablo 3.1'de yer listelenmektedir. Bu sorulardan ilk altısının cevaplarını bulmak için sistematik literatür taraması (Systematic Literature Review-SLR) yapılarak literatürde yer alan yayınlar incelenmiştir. Daha önce yapılan çalışmalar bir yandan bize yol gösterirken bir yandan da çalışmamızın sonunda bize karşılaştırma imkânı sağlamıştır. Bazı soruların cevaplanması için literatürde yer alan çalışmalardan faydalanmak tez çalışmasına daha fazla odaklanma imkânı vermiştir.

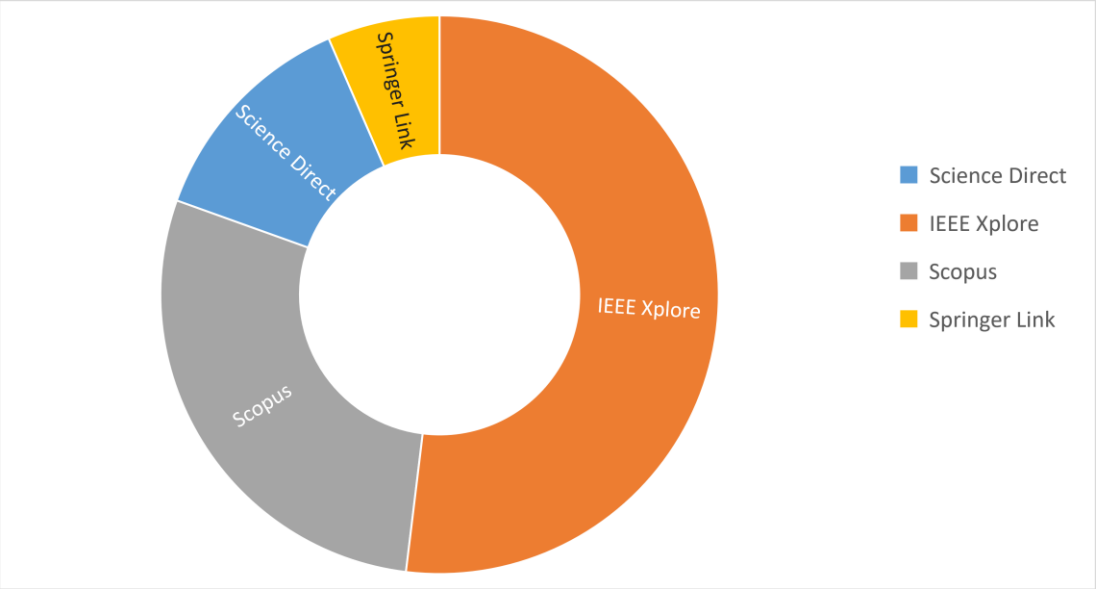
Tablo 3.1. Araştırma soruları

	Araştırma Soruları
S1	Servis Keşfine yönelik hangi yaklaşım kullanılmaktadır?
S2	Servis keşfi kullanım amacı nedir?
S3	Kullanılan Meta-sezgisel yaklaşım ve algoritmalar nelerdir?
S4	Kullanılan yaklaşım bulut tabanlı mıdır?
S5	Kullanıcı sorgusu öncesinde yapılan işlemler nelerdir?
S6	Zorluklar ve Çözüm önerileri nelerdir?
S7	Kullanılabilecek alternatif sınıflandırma algoritmaları nelerdir?
S8	Kullanılan sınıflandırma algoritmalarında en iyi hiperparametre kombinasyonları nasıl elde edilir?
S9	Sınıflandırma algoritmaları hangi verisetinde test edilecektir?

Yapılan SLR çalışması ile araştırma sorularından ilk altısına cevap verilmiştir. SLR çalışması sonucu elde edilen veriler aşağıda detaylı olarak anlatılmıştır. Tez çalışmasında kullanılacak alternatif yani önerilen sınıflandırma algoritmaları, bu algoritmalarda kullanılacak en iyi hiperparametre kombinasyonları belirlemek için faydalanılacak hiperparametre ayarlama yöntemleri de aşağıda detaylı olarak anlatılmıştır. Son olarak da deneylerin yapılacağı test ortamı ve kullanılan verisetleri aşağıda detaylı olarak anlatılmıştır.

3.1 Sistematik Literatür Taraması

Sistematik literatür taraması ile servis keşfi için belirlene ilk altı soruya literatürde yapılmış çalışmalardan çözüm yöntemleri bulunmuştur. SLR çalışmasın da ilk olarak Science Direct, IEEE, Scopus, Springer Link gibi yaygın olarak kullanılan veritabanlarında "Servis keşfi" ve "Web servislerinde seviş keşfi" anahtar kelimeleri ile arama yapılarak servis keşfi ile ilgili yayınları toplanmıştır, ikinci aşamada ise bulunan çalışmalar analiz edilmiştir. Seçilen araştırma soruları için bulunan sonuçlar seçilen yönteme de kaynak teşkil etmiştir. Literatürde servis keşfi ile ilgili son 10 yılda yayınlanmış çalışmalardan belirli kriterlere göre 46 yayın seçilmiştir. Şekil 3.1, seçilen yayınlarının veritabanlarına göre dağılımını göstermektedir. Seçim kriteri olarak tam metine ulaşma, dil olarak İngilizce ve deneysel sonuç içerme kriterleri uygulanmıştır. Seçilen her bir yayın araştırma soruları dikkate alınarak detaylı bir şekilde incelenmiş ve bulunan sonuçlar aşağıda detaylı olarak paylaşılmıştır.

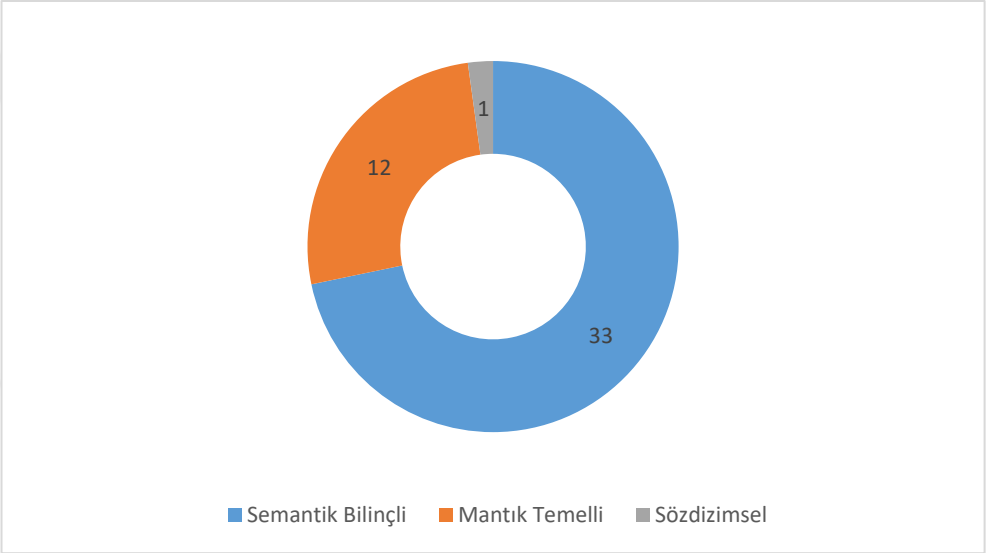


Şekil 3.1. Seçilen yayınların veritabanlarına göre dağılımı

3.1.1 Servis Keşfine yönelik hangi yaklaşım kullanılmaktadır?

Sözdizimsel [S1], Semantik bilinçli [S4, S5, S10, S15, S23] ve mantık temelli yaklaşımlar [S16-S19] literatürde baskın keşif teknikleridir. Sözdizimsel yaklaşımlar anahtar kelime eşleştirmesine göre çalışan en eski yöntemdir. Sözdizimsel yaklaşımda kullanıcının işi çok zordur. Çünkü doğru anahtar kelimeleri kullanmazsa istediği

servisi bulması çok zordur. Sözdizimsel yaklaşımdan sonra ortaya çıkan ve servislerin anlamsal tanımıyla ilgilenen semantik yaklaşım kullanıcının istediği servisleri daha hızlı ve güvenilir bir şekilde bulmasına imkân sağlar. Semantik temelli yaklaşımlarda türeyen ontoloji temelli yaklaşımda ise servis ve kullanıcı bağlamında kullanıcının en doğru servisi bulmasına imkân sağlar. Seçilen makaleler incelendiğinde Semantik bilinçli yaklaşım 33, mantık temelli yaklaşım 12 ve sözdizimsel yaklaşım 1 kez kullanılmıştır. Şekil 3.2, incelenen yayınlarda servis keşfine yönelik kullanılan yaklaşımların dağılımını göstermektedir. Bu sonuçlar bize web servis keşfinde en yaygın kullanılan yaklaşımın semantik bilinçli yaklaşım olduğunu göstermiştir.



Şekil 3.2. Servis keşfine yönelik kullanılan yaklaşımların dağılımı

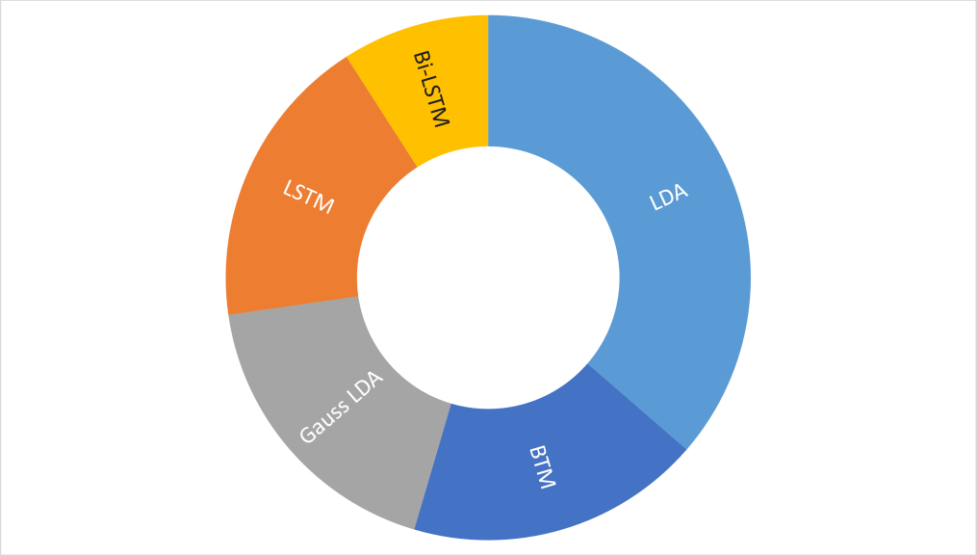
3.1.2 Web Servis keşfinin kullanım amacı nedir?

Web Servis keşfi temel olarak üç kavram üzerinde durmaktadır. Bunlar: Keşif [S2-S46], durum kontrol ve yük dengelemedir [S1]. Seçilen yayınların tamamı keşif kavramı üzerine odaklandığı belirlenirken sadece bir yayında yük dengelemeye de yer verildiği görülmüştür. Mikroservis ekosisteminde yer alan çok sayıda servis arasından keşfedilemeyen servislerin bir önemi yoktur. Bu da keşif sürecinin önemini ortaya koymaktadır. Literatürden seçilen yayınlara da keşif sürecinin öneminin yansıdığı görülmüştür. İncelen yayınların tamamında servislerin keşif sürecinde servisin bulunulması üzerine yoğunlaşıldığı görülmüştür. Bu da artan servis sayılarıyla birlikte

en önemli önceliğin servislerin mikroservis ekosisteminde bulunulması olduğunu göstermiştir.

3.1.3 Kullanılan Meta-sezgisel yaklaşım ve algoritmalar nelerdir?

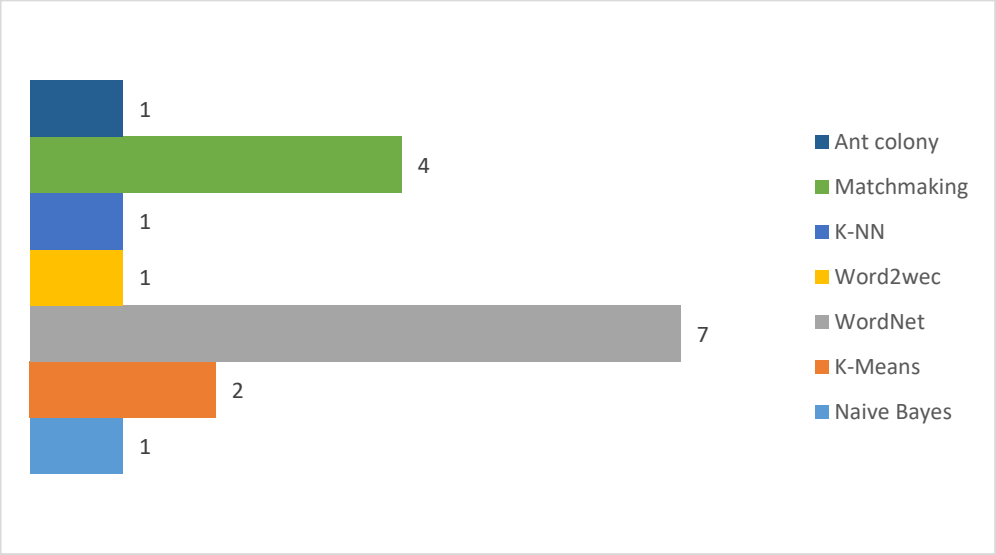
Web servislerinin gelişen teknoloji ile birlikte büyük bir hızla arttığı düşünüldüğünde kullanıcıların en doğru servisi en hızlı şekilde bulması her geçen gün zorlaşmaktadır. Araştırmacılar servis keşfini daha ileri taşıyabilmek için farklı yöntem ve algoritmalar kullanmaktadır. SLR çalışmasında web servis keşfi için servislerin ortak özelliklerine göre sınıflandırılması en doğru servisin en hızlı şekilde bulunabilmesi için çok önemlidir. LDA (Latent Dirichlet Allocation) [S5, S8, S24], BTM (Bi-term Topic Model) [S6, S34], Uzun Kısa Vadeli Hafıza Ağları (Long- Short Term Memory- LSTM) [S3], çift yönlü LSTM (Bi-LSTM) [S37] ve Gauss LDA [S24] Web servislerinin kümelenmesi veya sınıflandırılması için kullanılan en yaygın yöntemlerdir. İncelenen yayınlarda en yaygın olarak tercih edilen sınıflandırma yöntemlerinin LDA ve BTM olduğu görülmüştür. Şekil 3.3'de yayınlarda web servis keşfi sürecinde kümeleme için kullanılan yöntemler gösterilmektedir.



Şekil 3.3. Kümeleme ve Sınıflandırma Yaklaşımları

Servisler gruplanırken anahtar kelimeler ve servis açıklamalarında kullanılan kelimeler dikkate alınarak servisler arasındaki benzerliklere göre sınıflandırılmaktadır. Seçilen anahtar kelimelerin ile yeterli olmayan açıklama metinleri keşif süreci için sorun teşkil etmektedir. Kullanıcının arama yaparken kullandığı doğal dil ile burada kullanılan dilin farklılıkları ise başka bir sorundur. İncelenen yayınlarda araştırmacılar

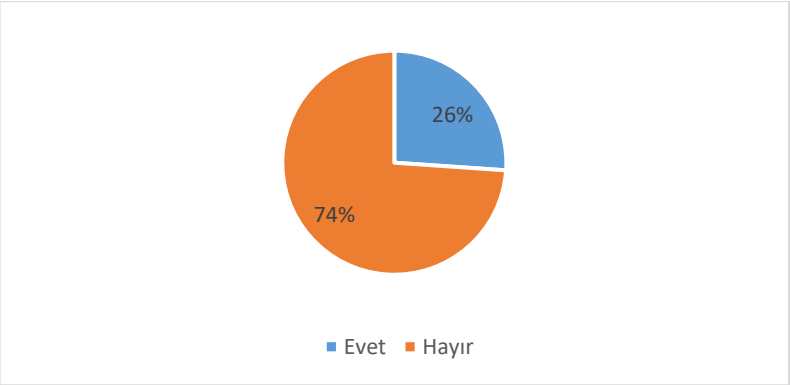
bu sorunları aşabilmek için farklı algoritmalar kullandığı tespit edilmiştir. SLR çalışmasında en çok karşılaşılan algoritmaların WordNet [S6, S20, S21], Çöpçatan [S12, S21, S29] ve K-Means [S2, S12] olduğu görülmüştür. Şekil 3.4, SLR çalışmasında tespit edilen algoritmaları göstermektedir.



Şekil 3.4. Algoritma türleri

3.1.4 Kullanılan yaklaşım bulut bilişim tabanlı mıdır?

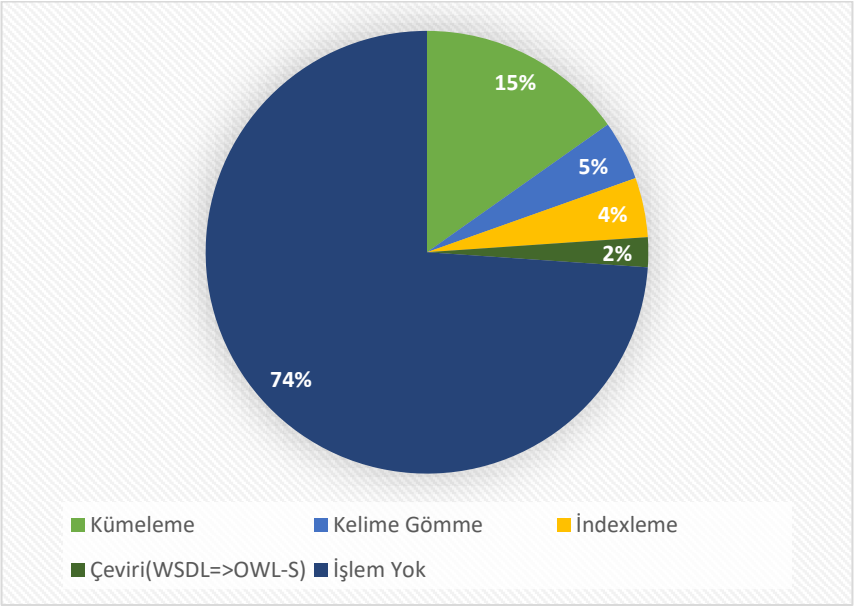
Ağ alt yapısındaki gelişmeler ile internete erişim hızı ve kalitesi her geçen gün artmaktadır. Bulut bilişim de internetteki bu gelişimle birlikte hızla büyümektedir. Servis sağlayıcılar ve kullanıcılar, güvenlik ve maliyeti dikkate alarak Bulut teknolojisini her geçen gün daha fazla kullanmaya başlaması da kaçınılmazdır. Servis sağlayıcılarının bulut teknolojisine geçmesi kullanıcılar açısından kullanılabilecek servis sayısının ve niteliğinin artmasını sağlayacaktır. Seçilen yayınlarda servislerin keşfi için kullanılan yaklaşımların Bulut Bilişim bağlantıları incelenmiştir [S10-S14, S22]. Web servislerinin her geçen gün bulut teknolojisini kullanımının artmasına rağmen SLR çalışmasında incelenen yayınlara bunun yansıması görülmemiştir. Şekil 3.5, incelen yayınların bulut bilişim kullanımını göstermektedir.



Şekil 3.5. Bulut Bilişim kullanım grafiği

3.1.5 Kullanıcı sorgusu öncesinde yapılan işlemler nelerdir?

Web servisleri için üç ana grup vardır. Bunlar servis sağlayıcılar, kullanıcılar ve servis sağlayıcılar ile kullanıcılar arasında bağlantıyı kuran aracılardır. Servis sağlayıcılar servislerini aracılar üzerinden yayınlar. Bu da aracılar üzerinde çok sayıda servisin olmasına ve karmaşıklığa yol açabilmektedir. İncelenen yayınlara bakıldığında kullanıcı sorgu yapmadan önce bazı işlemlerin yapıldığı görülmüştür. Bunlar servislerin kendi aralarında benzerliklerine göre sınıflandırılması, servisleri indexleme [S35], eksik açıklama metinleri için kelime gömme [S40] ve WSDL [S42] den OWL-S 'e [S3, S7, S14] çeviridir. İncelene yayınların çoğunluğunda bu işlemleri kullanıcı sorgusundan sonra kullanıcı isteğine göre yapıldığı görülmüştür. Şekil 3.6, kullanıcı sorgusu öncesi yapılan işlemlerin dağılımını göstermektedir.



Şekil 3.6. Sorgu Öncesi Yapılan İşlemler

3.1.6 Zorluklar ve Çözüm Önerileri Nelerdir?

Araştırmacılar için bu zorlukların öngörülebilir olması ve çözüm yöntemleri konusunda fikir sahibi olunabilmesi çok önemlidir. Yayınlarda rapor edilen zorluklar incelendiği zaman bu zorlukların genel olarak veriseti ve kullanıcı sorgusu olarak iki grupta toplandığı görülmektedir. Genel olarak incelenen yayınlarda araştırmacıların karşısına iki kategoride beş sorunun çıktığı görülmüştür. Bazı zorlukların çözüm önerilerinin birden fazla olması araştırmacıların daha iyiye veya özgün çözüm yöntemlerine yöneldiğini göstermektedir.

SLR çalışmasında seçilen yayınların araştırmacılarının karşısına ilk olarak çıkan sorun üzerinde çalışabilecekleri bir verisetinin bulunmamasıdır. Araştırmacılar için yapacakları çalışmaya uygun bir veriseti bulunması çok önemli, çünkü bir verisetini yeniden oluşturmak oldukça zahmetli ve maliyetli bir çalışma gerektirir. İncelenen yayınlarda araştırmacılarımız bu sorunu halka açık bir veriseti kullanarak çözüme kavuşturmuşlardır [S2, S4, S22, S30, S31, S33]. Genel olarak kullanılan verisetleri SAWSDL-TC3, OWLS-TC4 ve ProgrammableWeb'e ait verisetleridir. Veriseti ile ilgili karşılaşılan ikinci sorun verisetinde bulunan hatalı içeriğe sahip servisler ve replika servislerin varlığıdır [S15, S18, S20]. Çalışmanın güvenliği için incelenen yayınları hazırlayan araştırmacılar verisetinde bulunan hatalı içeriğe sahip servisleri ve replika servisleri araştırmadan çıkarma yoluna gitmeyi tercih ettikleri görülmüştür.

Veriseti ile karşılaşılan son sorun verisetinde bulunan yeterli açıklamaya sahip olmayan servislerdir [S13, S40]. Servislerin yeterli açıklamaya sahip olmaması servis keşfi sürecinde doğru servisin tahmin edilmesini zorlaştırmaktadır. Bu sorunu aşmak için araştırmacılar servis açıklama bölümüne bu bölümde yer alan diğer kelimeler ile benzer anlam ve sözdizimsel özelliğe sahip kelimeler ekleyerek [S40] servis tanımını geliştirdikleri görülmüştür. Araştırmacıların tercih ettiği diğer bir çözüm önerisinin ise servisin ait olduğu kategoriye ait rastgele cümleler ile servisin tanımını geliştirmek olduğu tespit edilmiştir [S13].

Gündelik dil kullanımı ve sorgu esnasında kullanılan kısaltmalar servis keşfi aşamasında karşılaşılan diğer bir sorundur [S3, S11, S12, S18, S20, S33, S36, S40]. Kullanılan sorguda geçen gereksiz kelimelerin ve kısaltmaların varlığı derin öğrenme algoritmaları ve makine öğrenmesi algoritmaları kullanımında hem süreci uzatmakta

hem de çıkacak sonucun doğruluğunu etkilemektedir. Araştırmacılar bu sorunu sorguda yer alan gereksiz kelimeleri ve kısaltmaları kaldırarak çözdükleri görülmüştür.

Sorguda kullanılan kısa (yetersiz) veri girişi karşılaşılan başka bir sorundur [S2, S5, S7, S12, S40]. Sorgudaki yetersiz veri girişi kullanıcının isteğinin tam olarak anlaşılmasını engeller. Sorgu sonucu olarak kullanıcının karşısına çok fazla servisin dönmesi veya hiçbir servisin bulunamaması muhtemeldir. Bu soruna çözüm olarak araştırmacılar iki çözüm yolu kullandıkları tespit edilmiştir. Bunların birincisi sorguda kullanılan anahtar kelimelerin eş anlamlılarını ve çeşitli varyanslarını kullanmak [S5, S7, S40], ikincisi ise eş anlamlı kelimelerden ve kısaltmalardan oluşan uzlaşı kütüphanesi kullanmaktır [S2, S12]. Seçilen yayınlarda karşılaşılan zorluklar ve bu zorluklara karşı önerilen çözüm önerileri Tablo 3.2'de listelenmiştir.

Tablo 3.2. Zorluklar ve Çözüm Önerileri

Kategori	**Zorluklar (1–5)**	**Çözüm Önerileri (1–7)**
Veriseti	1.Veriseti edinme	1.Halka açık veriseti kullanma
	2.Hatalı içeriğe sahip servisler ve replikalar	2.Hatalı veya eksik içeriğe sahip servisleri ve replikaları kaldırma
	3.Verisetinde bulunan servislere ait yetersiz açıklama	3.Aktif kelime ile benzer anlam ve sözdizimsel özelliğe sahip kelimeler kullanılarak servis tanımını geliştirme
		4.Servisin ait olduğu kategoriye ait seçilen rastgele cümle ekleme
Sorgu	4.Kullanıcının doğal dil girişi (Gündelik Dil Kullanımı)	5.Dolgu sözcükleri (Stop words) ve kısaltmaları kaldırma
	5.Sorguda kısıtlı veri girişi	6.Sorguda anahtar kelimelerin eş anlamlılarını / varyasyonlarını kullanma
		7.Uzlaşı Kütüphanesi (eş anlamlı kelimeler ve kısaltmalardan oluşan) kullanma

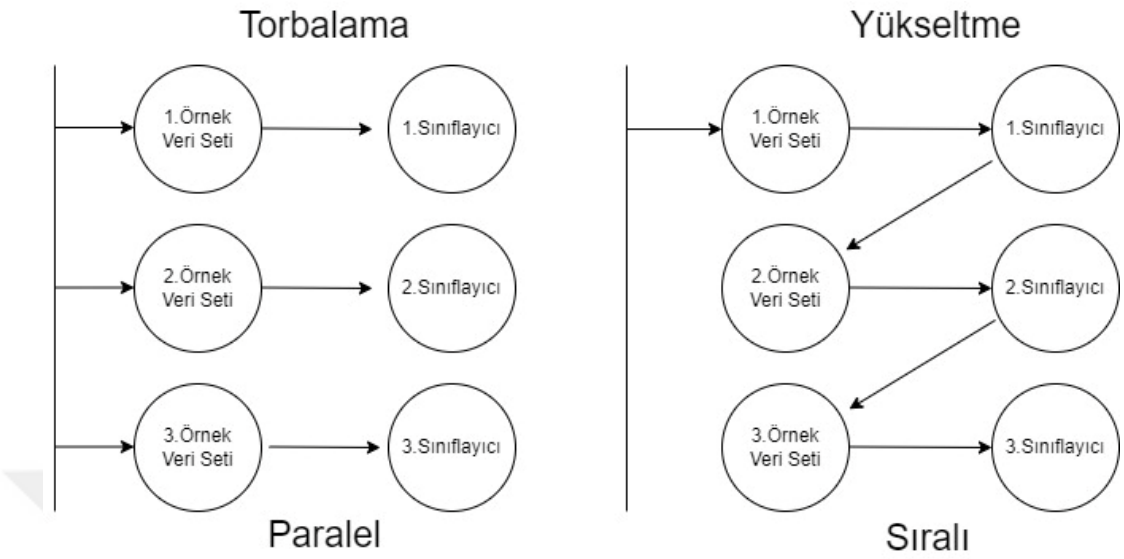
3.2 Önerilen Sınıflandırma Yöntemi

Sistematik literatür taraması ile elde ettiğimiz sonuçlar ışığında servis keşfi için sınıflandırma yöntemlerinin önemi açıkça görülmektedir. Artan servis sayısı ve özellikle benzer işleve sahip birden çok servis oluşu en doğru servisin keşfini zorlaştırmaktadır. Yedinci araştırma sorumuza cevap olarak literatürde yer alan sınıflandırma algoritmalarına karşı yükseltme algoritmalarını öneriyoruz. Yükseltme (Boosting) topluluk ilkesine göre çalışan sıralı bir tekniktir yani bu yöntem yeni veriyi sıralı şekilde oluşturur. Topluluk öğrenme yöntemi aynı öğrenme algoritmasını kullanarak birden fazla modeli eğitmek amacıyla kullanılan bir makine öğrenmesi yöntemidir. Öğrenmede hatanın başlıca nedenleri yanlılık, gürültü ve varyanstan kaynaklanmaktadır. Topluluk öğrenme yöntemi bu faktörlerin minimum seviyeye indirilmesine sağlar. Bu yöntem makine algoritmalarının doğruluğunu ve kararlılığını geliştirmek amacıyla tasarlanmıştır. Bu yöntem Torbalama (Bagging) ve Yükseltme (Boosting) tekniği olmak üzere ikiye şekilde uygulanmaktadır [23].

Torbalama yöntemi 1996 yılında Breiman tarafından geliştirilmiştir [23]. Orijinal verisetinden elde edilen önyükleme örneklerine tahminciler uygulanarak bir topluluk oluşturulur. Oluşturulan alt örneklemler orijinal verisetindeki sayı ile aynı olacağı için bazı gözlemler önyükleme sonucunda oluşturulan örneklemlerde yer almayabilirken bazıları iki veya daha fazla kez yer alabilir. Tahminlerin birleştirilmesinde regresyon ağaçlarında ortalama alınır, sınıflandırma ağaçlarında ise sonuçlar oylama ile belirlenir. Torbalama yöntemi, düşük yanlılık ve yüksek varyanslı olan değişkenleri daha uygun hale getirirken değişkenlerin tahmin geçerliliğini de arttır. Deneysel sonuçlara göre torbalama yöntemi, tekil ağaçlara göre daha verimli sonuçlar vermektedir.

Yükseltme yönteminde temel amaç verisetine farklı ağırlıklar verilerek oluşturulan ağaçlar topluluğundan çıkarımlar yapmaktır. Başlangıçta tüm gözlemler eşit ağırlığa sahip iken ağaç topluluğu büyümeye başladıkça, ağırlıklandırmalar tekrar düzenlenir. Yükseltme algoritmalarında ağaçlar birbirlerine bağlı ve her bir sınıflandırıcı, önceki sınıflandırıcıların başarısını dikkate alınarak eğitilir. Yükseltmenin amacı, en doğru hedefin seçilme olasılığını arttırmaktır [24]. Şekil 3.7, torbalama ve yükseltme algoritmalarının çalışma şeklini göstermektedir. En çok

bilinen yükseltme algoritmaları Gradyan Artırma, XGBoost, LightGBM ve CatBoost' dur. Bu algoritmalar aşağıda detaylı olarak açıklanmıştır.



Şekil 3.7. Topluluk öğrenme yöntemleri çalışma şekli

3.2.1 Gradyan Artırma Algoritması

Gradyan Artırma, Friedman tarafından 2001 yılında tanıtılan güçlü bir makine öğrenme tekniğidir. Yükseltme, zayıf öğrenicileri öğreniciye dönüştürme yöntemidir [6]. Bunu aşamalı, eklemeli ve sıralı olarak yapar. Gradyan Artırma 'de öncelikli olarak ilk karar ağacı oluşturulur. Sonrasında tahmin hataları göz önüne alınarak yeni ağaçlar oluşturulur. Bu durum modelden daha fazla gelişme kaydedilemeyinceye kadar devam eder. Hem Regresyon hem de sınıflandırma modellerini eğitmek için kullanılabilir. Gradyan Artırma 'da başarıyı artırmak için doğru hiperparametreleri ayarlamak çok önemlidir. Bu modelde kullanılabilecek hiperparametreler 3 kategoride toplanmaktadır.

İlk kategoride modeldeki her bir ağacı etkileyen ağaca özgü hiperparametreler yer almaktadır [25]. Yükseltme algoritmalarının temelinde karar ağaçları yer aldığından modelin başarısı için oluşturulan ağacın derinliği, yaprak sayısı gibi özellikleri modelin oluşturulması aşamasında çok önemlidir. Ağaca özgü hiperparametreler Tablo 3.3'de açıklanmıştır.

Tablo 3.3. Gradyan artırma algoritması ağaca özgü hiperparametreler

Tür	Hiperparametre	Açıklama
Ağaca özgü Hiperparametreler	**min_samples_split**	Bölme için dikkate alınması gereken bir düğümde gerekli olan minimum örnek sayısını tanımlar.
	min_samples_leaf	Bir uç düğüm veya yaprakta gereken minimum örnekleri tanımlar.
	min_weight_fraction_leaf	Bir tamsayı yerine toplam gözlem sayısının kesri olarak tanımlanır.
	max_depth	Bir ağacın maksimum derinliğini ayarlamak için tanımlanır.
	max_leaf_nodes	Bir ağaçtaki maksimum uç düğüm veya yaprak sayısını tanımlar.
	max_features	En iyi ayrımı ararken dikkate alınması gereken özelliklerin sayısını tanımlar.

İkinci kategoride ise model geliştirildikten sonra modelin performansını yükseltmek için kullanılan hiperparametreler yer almaktadır. Bu hiperparametreler modeldeki yükseltme işlemini etkiler [25]. Yükseltme hiperparametrelerinin ayarlanmasında hiperparametre ayarlama yöntemlerinin kullanılması en uygun değerlerin bulunmasında önemlidir. Gradyan artırma algoritmasında kullanılan yükseltme hiperparametreleri Tablo 3.4'de açıklanmıştır. Son kategoride ise kayıp fonksiyonu, çıktı ve rasgele sayı üretilmesi gibi genel işlevleri ayarlamaya yarayan hiperparametreler yer almaktadır. Bu hiperparametreler modelin genel işlevi için kullanılır [25]. Gradyan artırma algoritmasında genel işlevleri ayarlamada kullanılan hiperparametreler Tablo3.5'de açıklanmıştır.

Tablo 3.4. Gradyan artırma algoritması yükseltme hiperparametreleri

Tür	Hiperparametre	Açıklama
Yükseltme Hiperparametreleri	**learning_rate**	Bu, her ağacın nihai sonuç üzerindeki etkisini belirler.
	n_estimators	Modellenecek sıralı ağaçların sayısı tanımlanır.
	Subsample	Her ağaç için seçilecek gözlem oranı tanımlanır.

Tablo 3.5. Gradyan artırma algoritması genel hiperparametreler

Tür	Hiperparametre	Açıklama
Genel Hiperparametreler	**Loss**	Her bölmede en aza indirilecek kayıp fonksiyonunu ifade eder.
	İnit	Bu hiperparametre, çıktının başlatılmasını ayarlar.
	random_state	Her seferinde aynı rasgele sayıların üretilmesi için rasgele sayı çekirdeği tanımlar.
	Verbose	Model tamamlandığında yazdırılacak çıktı türü tanımlanır.
	warm_start	Bunu kullanarak, bir modelin önceki uyumlarına ek ağaçlar eklenebilir.
	presort	Daha hızlı bölmeler için verilerin önceden sıralanıp sıralanmayacağını tanımlanır.

3.2.2 XGBoost (eXtreme Gradyan Artırma) Algoritması

Tianqi Chen ve Carlos Guestrin tarafından 2016 yılında geliştirilmiştir. Paralel işleme, ağaç budama, eksik değerleri işleme ve aşırı sapmayı(overfitting) önlemek için düzenleme yoluyla optimize edilmiş [24] yani hız ve performans için tasarlanmış Gradyan Artırma çerçevesi kullanan, karar ağacı tabanlı bir uygulamadır. Aşırı uyumu azalttığı ve genel performansı artırdığı için düzenli artırma tekniği olarak da bilinir. XGBoost algoritması kullanılırken farklı hiperparametrelerin göz önünde bulundurulması gerekir. XGBoost algoritmasında yer alan hiperparametreler modelin öğrenme görevi, performansını artırmak ve genel işlevlerini ayarlamak için kullanılan hiperparametreler olmak üzere 3 kategori altında toplanır [26]. Tablo3.6 da XGBoost algoritmasına ait hiperparametreler açıklanmaktadır.

Tablo 3.6. XGBoost algoritması hiperparametreleri

Tür	Hiperparametre	Açıklama
GENEL	**booster**	Her yinelemede çalıştırılacak model türünü seçmek için kullanılır.
	silent	Sessiz mod etkinleştirildiğinde 1'e ayarlanır, yani çalışan hiçbir mesaj yazdırılmaz.
	nthread	Bu, paralel işleme için kullanılır ve sistemdeki çekirdek sayısı girilmelidir.
YÜKSELTİCİ	**eta**	Her adımda ağırlıkları küçülterek modeli daha sağlam hale getirir.
	min_child_weight	Bir çocukta gerekli olan tüm gözlemlerin ağırlıklarının minimum toplamını tanımlar.
	gama	Gama, bir bölme yapmak için gereken minimum kayıp azaltmayı belirtir.
	max_delta_step	Maksimum delta adımında, her ağacın ağırlık tahmininin olmasına izin verir.
	colsample_bytree	Her ağaç için rasgele örneklenecek sütunların oranını belirtir.
	colsample_bylevel	Her düzeyde, her bölme için sütunların alt örnek oranını belirtir.
	lambda	XGBoost 'un düzenlileştirme bölümünü işlemek için kullanılır.
	reg_alpha	Algoritmanın daha hızlı çalışması için çok yüksek boyutluluk durumunda kullanılabilir
ÖĞRENME GÖREVİ	**Objective**	Bu, minimize edilecek kayıp fonksiyonunu tanımlar.
	eval_metric	Doğrulama verileri için kullanılacak ölçüttür.
	seed	Tekrarlanabilir sonuçlar üretmek ve ayrıca parametre ayarı için kullanılabilir.

3.2.3 LightGBM Algoritması

LightGBM, Microsoft DMTK (Distributed Machine Learning Toolkit) projesi kapsamında 2017 yılında geliştirilmiş bir yükseltme algoritmasıdır. Diğer yükseltme algoritmaları ile karşılaştırıldığında yüksek işlem hızı, büyük verileri işleyebilmesi, daha az kaynak (RAM) kullanımı, yüksek tahmin oranı, paralel öğrenme ve GPU öğrenimini desteklemesi gibi avantajları vardır [27]. Yapılan çalışmalara göre diğer modellere göre 20 kat daha hızlı olduğu sonucuna ulaşılmıştır [28].

LightGBM, histogram tabanlı çalışan bir algoritmadır. Sürekli değere sahip olan değişkenleri kesikli hale getirerek hesaplama maliyetini azaltır. Bu yöntem sayesinde hem eğitim süresi kısalmakta hem de kaynak kullanımı düşmektedir. Karar ağaçlarında öğreniminde seviye odaklı veya yaprak odaklı olarak iki strateji kullanılabilir. Seviye odaklı stratejide ağaç büyürken ağacın dengesi korunur. Yaprak odaklı stratejide ise kaybı azaltan yapraklardan bölünme işlemi devam eder. LightGBM bu özelliği sayesinde diğer yükseltme algoritmalarından ayrılmaktadır. Model yaprak odaklı strateji ile daha az hata oranına sahip olur ve daha hızlı öğrenir. Ancak yaprak odaklı büyüme stratejisi veri sayısının az olduğu durumlarda modelin aşırı öğrenmeye yatkın olmasına sebebiyet verir. Bu nedenle algoritma büyük verilerde kullanılmak için daha uygundur. Ayrıca ağaç derinliği, yaprak sayısı gibi parametreler optimize edilip aşırı öğrenmenin önüne geçmeye çalışılabilir.

LightGBM ayrıca diğer algoritmalardan farklı iki teknik kullanmaktadır. Bunlar, veri örneklerinin ve değişkenlerin sayısı ile ilgilenen Gradyan Tabanlı Tek Yönlü Örnekleme ve Özel Değişken Paketi'dir. Gradyan Tabanlı Tek Yönlü, karar ağaçlarının doğruluk oranını korurken veri sayısını azaltmayı amaçlamaktadır. Özel Değişken Paketi, doğruluk oranına zarar vermeden değişken sayısını azaltmayı ve buna bağlı olarak model eğitiminin verimliliğini arttırmayı amaçlamaktadır. Özetle, Gradyan tabanlı örnekleme daha az önemli sayılabilecek verileri ihmal ederek bilgi kazanımını hesaplamak için veri boyutunu azaltırken, Özel Değişken Paketi, boyutsallığı azaltmak için değişkenleri bir araya getirir. Bu iki işlev ile LightGBM eğitim sürecinin verimliliğini arttırır. LightGBM algoritmasına ait en önemli hiperparametreler Tablo 3.7'de açıklanmıştır.

Tablo 3.7. LightGBM algoritması hiperparametreleri

Hiperparametre	Açıklama
lambda_l1	L1/l2 için Lambda_l1 kontrolü aşırı sığdırmayla mücadele etmek için kullanılır.
max_depth	Eğitilen her ağacın maksimum derinliğini kontrol eder.
num_iterations	Num_iterations, güçlendirme yinelemelerinin sayısını belirtir.
learning_rate	Bu, her ağacın nihai sonuç üzerindeki etkisini belirler.
early_stopping_rounds	Son erken durdurma turundan sonra doğrulama metriği gelişmiyorsa, bu parametre eğitimi durduracaktır.
subsample	Ağaç oluşturma yinelemesi başına kullanılan satırların yüzdesini belirlenmesini sağlar.
Boosting	Artırma yöntemini belirlemek için kullanılır. Gbdt, dart ve goss değerlerini alır.

3.2.4 CatBoost Algoritması

Catboost, Yandex şirketi tarafından geliştirilmiş olan Gradyan Artırma tabanlı açık kaynak kodlu bir makine öğrenmesi algoritmasıdır. Gradyan Artırma algoritmasının performansını arttırmak amacıyla geliştirilen XGBoost ve LightGBM'e alternatif olarak Nisan 2017 tarihinde tanıtılmıştır. Yüksek öğrenme hızı hem sayısal hem kategorik hem de metin verileri ile çalışılabilmesi, GPU desteği ve görselleştirme seçenekleri sunması diğer algoritmalardan en çok ayrılan özellikleridir [29].

CatBoost algoritması, veri hazırlığı evresini kısaltması sayesinde diğer modellere göre daha iyi bir konuma sahiptir. Boş veriler ile çalışabilir ve kategorik verilere kodlama uygular. Kendine has bir kodlama metoduna sahip olması sebebiyle kategorik veriler ile yüksek performanslı olarak çalışabilir. Yani veri hazırlığı yaparken ayrıca bir kodlama işlemi yapılmasına gerek yoktur hatta kodlama yapılmaması tavsiye edilir. Ayrıca Catboost simetrik ağaçlar kurması sayesinde çok derin ağaçlar kurmadan yüksek tahmin oranı yakalar ve aşırı öğrenme sorununu aşar. Eğer aşırı öğrenme meydana gelirse algoritma hedefe ulaşmadan önce öğrenmeyi durdurur.

Algoritma GPU ile öğrenim yapabilmektedir. Bu sayede öğrenim süresi kısalmaktadır. Algoritmanın GPU üzerinde çalışabilmesi için NVIDIA Driver versiyon 418 ya da üstü ekran kartına sahip olunması gerekmektedir. Ayrıca öğrenme sırasında CatBoost bellek kopyaları alarak çalışır eğer bilgisayarın bir anda kapanması gibi beklenmedik bir sorun yaşanırsa öğrenme aşaması kaldığı yerden devam eder. Bellek kopyası alınması durumu parametrelerle kontrol edilebilir. Bu özellik büyük verilerle çalışırken öğrenim süresi çok uzun olacağı için önemlidir. Ayrıca CatBoost, parametre ayarı için esnek bir arayüz sağlar ve farklı görevlere uyacak şekilde yapılandırılabilir [30]. Tablo 3.8'de CatBoost algoritmasında öne çıkan hiperparametreler açıklanmıştır.

Tablo 3.8. CatBoost algoritması hiperparametreleri

Hiperparametre	Açıklama
iterations	Makine öğrenimi problemlerini çözerken oluşturulabilecek maksimum ağaç sayısını ifade eder.
learning_rate	Öğrenme oranı. Gradyan adımını azaltmak için kullanılır.
depth	Ağacın derinliğini ayarlamak için kullanılır.
l2_leaf_reg	Maliyet fonksiyonunun L2 düzenlileştirme dönemindeki katsayıyı ifade eder.
random_strength	Ağaç yapısı seçildiğinde bölmeleri puanlamak için kullanılacak rastgelelik miktarı tanımlanır.
bagging_temperature	Bayes önyüklemesinin ayarlarını tanımlar.
border_count	Sayısal özellikler için bölme sayısını tanımlar.

3.3 Hiperparametre Ayarlama Yöntemleri

Makine öğrenimi modelleri için doğru doğru hiperparametreleri seçmek çıkacak sonuçların doğruluğu için çok önemlidir. Hiperparametreler doğru bir şekilde ayarlanmazsa model parametrelerimiz, kayıp fonksiyonunu indiremediği için yetersiz sonuçlar üretir. Bu da modelimizin daha fazla hata yaptığı yani uygulamada doğruluğun daha kötü çıkacağı anlamına gelir. Hiperparametre ayarlaması yapılmadan önce parametreler ve hiperparametreler arasındaki farkın bilinmesi gerekir. Model parametreleri model tarafından verilen verilerden tahmin edilen parametrelerdir. Bir öğrenme algoritması, verilen veri kümesi için model parametrelerini öğrenir veya

tahmin eder, ardından öğrenmeye devam ederken bu değerleri güncellemeye devam eder. Hiperparametreler ise algoritmanın kendisine özgüdür yani model tarafından verilen verilerden tahmin edilemeyen parametrelerdir [31].

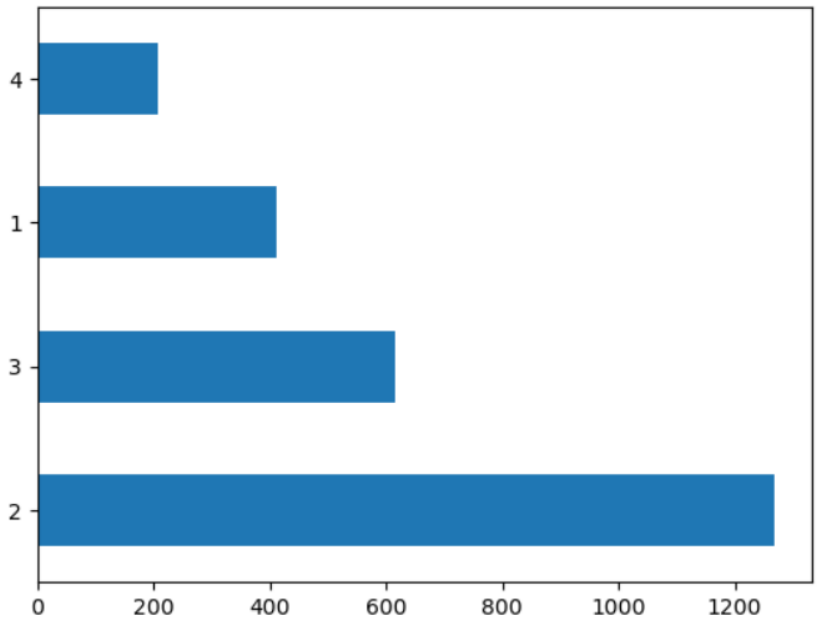
Hiperparametre ayarı, oluşturulan modelin performansını en üst seviyeye çıkaran doğru hiperparametre kombinasyonunu belirleme işlemidir. Tek bir eğitim sürecinde birden çok deneme çalıştırır. Her denemede, eğitim seçilen hiperparametreler için belirlenen sınırlar içinde ayarlanan değerlerle eksiksiz bir şekilde yürütülür. Bu işlemin sonunda, modelin en iyi sonuçları vermesi için gerekli en uygun hiperparametre değerlerinin kümesini verir. Oluşturulan modelin en iyi sonucu vermesi için önemli bir adımdır. Random Search [32], Grid Search, Halvin Grid Search [33], Halvin Random Search ve Bayes Search çalışmamızda kullanılan hiperparametre optimizasyon teknikleridir.

3.4 Test Ortamı ve Veriseti

Bu tez çalışması kapsamında oluşturulan tüm modeller için yapılan deneylerin yapıldığı bilgisayarın özellikleri elde edilen sonuçların geçerliliği için çok önemlidir. Daha iyi veya daha kötü özelliklere sahip bilgisayarlarda farklı sonuçların bulunması kuvvetle muhtemeldir. Bu tez kapsamında yapılan tüm deneyler için kullanılan bilgisayarın özellikleri şunlardır: İşlemci 11.Nesil Intel Core i7-11800H 2.30GHz, Ram 16GB DDR4 3200MHz, Ekran kartı 4GB NVIDIA GeForce RTX 3050 ve kullanılan işletim sistemi Windows 10 Profesyonel (64 Bit) işletim sistemidir. Programlama editörü olarak da Jupyter Notebook [34] tercih edilmiştir.

Günümüzde web servis sağlayıcılarının ve web servislerinin sayısının her geçen gün artmasıyla web servis kalitesinin önemi de daha da artmaktadır. Servis kalitesinin başarılı bir şekilde değerlendirebilmek için kullanıcının servisten beklentilerinin ne olduğunu bilmek gerekir. Son araştırma sorumuz cevap olarak bu çalışmada yer alan sınıflandırma algoritmalarını test etmek ve karşılaştırmak için halka açık olan QWS (Web Servis Kalitesi-Quality of Web Service) [7] verisetinin iki versiyonu da kullanılmıştır. Deneyler QWS verisetinin ikinci versiyonu ile yapılmıştır. En yüksek değerlere sahip modelimiz son olarak birinci versiyonu ile tekrar test edilmiştir. QWS verisetinin birinci versiyonu 365 ikinci versiyonu ise 2.507 gerçek Web servis uygulaması için bir dizi QWS ölçümünden oluşur. QWS veriseti 13 parametreden oluşmaktadır. WsRF (Web Services Resource Framework) [35]

parametresi ilk 9 parametre dikkate alınarak oluşturulmuştur. Sınıflandırma parametresi ise QWS verisetinde, WsRF genel kalite derecelendirmesine göre 1.Platin (Yüksek Kalite), 2.Altın, 3.Gümüş, 4.Bronz (Düşük Kalite) olacak şekilde oluşturulmuştur. Qws verisetinde bulunan sınıfların ağırlıkları Şekil 3.8'de gösterilmektedir. Sınıflandırma, aynı işlevi yerine getiren çeşitli servisler arasında ayrım yapmak için yardımcı olabilir. QWS parametreleri Tablo 3.9'de gösterilmiştir.



Şekil 3.8. Qws veriseti sınıf ağırlıkları

Tablo 3.9. QWS Parametreler

ID	**Parametre**	**Tanım**
1	Tepki Süresi	İstek gönderme ve yanıt alma zamanı
2	Kullanılabilirlik	Başarılı çağrılar/toplam çağrılar
3	Verim	Toplam çağrı sayısı / süre
4	Başarı Oranı	Yanıt mesajı / istek mesajı
5	Güvenilirlik Oran	Hata mesajı sayısı/toplam mesaj sayısı
6	Uyma	Bir WSDL'nin bir belirtimi takip ettiği kapsam
7	En İyi Uygulamalar	WS-I Temel Profilini takip etme kapsamı
8	Gecikme	Belirli bir isteği işleme koyma süresi
9	Belgeler	WSDL'deki dokümantasyon ölçüsü
10	WsRF	Web Servis Kalite Sıralaması
11	Servis Sınıflandırması	Servis sunma niteliklerini temsil eden düzeyler
12	Servis adı	Web Servis adı
13	WSDL Adresi	(WSDL) dosyasının Web üzerindeki konumu

4. BULGULAR

Bu bölümde ilk olarak sistematik literatür taraması sonucu tespit edilen sınıflandırma algoritmalarından Rassal Orman, Destek Vektör Makinesi, Karar Ağacı, K-en Yakın Komşular ve Naive Bayes sınıflandırma algoritmalarına ait elde edilen sonuçlar açıklanmıştır. Daha sonra ise topluluk öğrenme ilkesine göre çalışan yükseltme algoritmaları Gradyan Artırma, XGBoost, LightGBM ve CatBoost algoritmalarına ait sonuçlar açıklanmıştır. Her algoritmanın eğitimi QWS verisetinin ikinci versiyonu ile tek tek bağımsız olarak yapılmıştır. Veriseti %70 eğitim ve %30 test olmak üzere ikiye ayrılmıştır ve elde edilen en iyi sonuçlar gösterilmiştir.

Deneylerde ilk olarak QWS verisetinde bulunan servis kalite özelliklerinin (İlk dokuz Parametre) SHAP (SHapley Additive ExPlanations) tekniği ile model tahmininde yaratığı etki yani tahmindeki ağırlıkları belirlenmiştir. SHAP tekniği verisetinde yer alan her bir değer için ayrı Shapley değeri hesaplar ve her Shapley değeri, ilişkilendirildiği verinin tahminde yarattığı etkiyi belirtmektedir. Bir özelliğe ait tüm Shapley değerlerinin toplamı, o özelliğin modelin tahmini için toplam katkısını belirtmektedir. Oluşturulan modelin eğitim ve tahmin sürelerini kısaltmak için parametrelerin tahmindeki ağırlıkları dikkate alınarak ilk 6 parametre ile deneyler gerçekleştirilmiştir.

Sınıflandırma algoritmaları için hiperparametre değerlerinin kombinasyonu çok önemlidir. İkinci aşamada her bir sınıflandırma algoritmasının eğitim ve test doğruluk değerini artırmak için hiperparametre ayarlama yöntemleri kullanılarak en iyi hiperparametre kombinasyonları veya hiperparametre değerleri bulunmuştur. Bulunan hiperparametre değerleri ile elde edilen doğruluk oranları her sınıflandırma algoritması için ayrı ayrı gösterilmiştir. Son olarak modele ait Karmaşıklık Matrisi (Confusion Matrix), Sınıflandırma raporu ve işlem karakteristik eğrisi (Receiver Operating Characteristics-ROC) ile bu eğrinin altında kalan kısım (Area Under The Curve-AUC) kullanılarak modele ait sonuçlar anlatılmıştır.

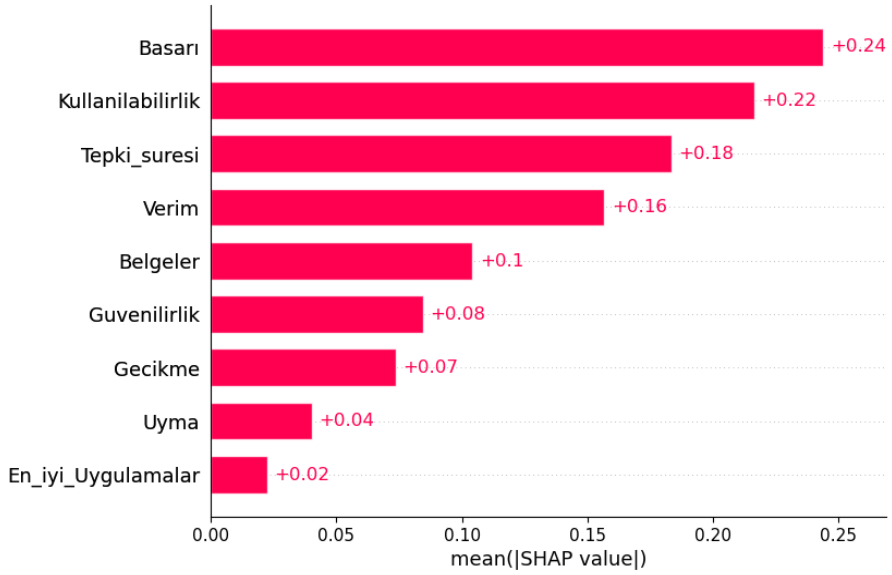
Sınıflandırma problemlerinin değerlendirilmesinde kullanılan Karmaşıklık matrisi öngörülen ve tahmini değerleri gösteren dört farklı kombinasyona sahip bir tablodur. Karmaşıklık matrisi ile elde edilebilecek ölçütlerden bazıları şunlardır. İlk olarak doğruluk yani doğru olarak sınıflandırılan örneklerin yüzdesi, ikinci olarak

Duyarlılık (Recall) yani pozitif olarak tahmin etmemiz gereken işlemlerin ne kadarını pozitif olarak tahmin ettiğimizi gösteren ölçüt ve Kesinlik(precision), pozitif olarak tahminlendiğimiz değerlerin gerçekten kaç adedinin pozitif olduğunu göstermektedir. Doğruluk, Kesinlik ve Duyarlılık ölçütlerini Sınıflandırma raporu (classification report) ile ayrı ayrı kod yazmadan tek seferde gösterilmiştir. Sınıflandırma raporunda yer alan F1Score kesinlik ve duyarlılığın harmonik ortalamasıdır. Modelin kesinliğini ve sağlamlığını göstermektedir.

AUC ve ROC eğrisi sınıflandırma modellerinin başarısını gösterir. ROC bir olasılık eğrisidir ve AUC ayrılabilirliğin derecesini veya ölçüsünü temsil eder [36]. AUC, ROC eğrisinin altında kalan alandır. AUC yüksek ise model tahmin etmekte daha iyi demektir. ROC-AUC genellikle ikili sınıflandırma modellerini destekler fakat bizim modellerimiz çok sınıflı olduğu için bire karşı diğerleri yaklaşımı kullanılarak ROC-AUC değerleri hesaplanmıştır. Bu yöntemde, bir sınıfı "olumlu" kabul ederken, diğerleri "negatif" kabul edilir. Verisetimiz 4 sınıflı olduğu için dört farklı ROC-AUC puanı elde edilmiştir ve model puanı için bu puanların ortalaması alınmıştır. Histogramlar ne kadar ayrı ise ROC eğrisi o kadar iyidir ve ortalama puan ne kadar yüksekse model tahminini o kadar iyi yapıyor demektir.

4.1 Naive Bayes Sınıflandırma Algoritması

Naive Bayes sınıflandırma algoritmasının temeli Bayes teoremine dayanır. Bayes teoremi 1812 yılında Thomas Bayes tarafından bulunan koşullu olasılık hesaplama formülüdür [37]. Naive Bayes sınıflandırması olasılık ilkelerine göre tanımlanmış bir dizi hesaplama ile, sisteme sunulan verilerin sınıfını yani kategorisini tespit etmeyi amaçlar. Modelimiz Naive Bayes algoritması ile çalıştırıldıktan sonra QWS veriseti parametrelerinin ağırlıkları tespit edilmiştir. Parametrelerin model tahmini üzerindeki ağırlıkları Şekil 4.1'de gösterilmektedir. En yüksek ağırlığa sahip ilk altı parametre olan Başarı, Kullanılabilirlik, Tepki süresi, Verim, Belgeler ve güvenirlilik deneylerde kullanılmak üzere seçilmiştir.



Şekil 4.1. Naive Bayes algoritmasında QWS veriseti parametrelerinin ağırlıkları

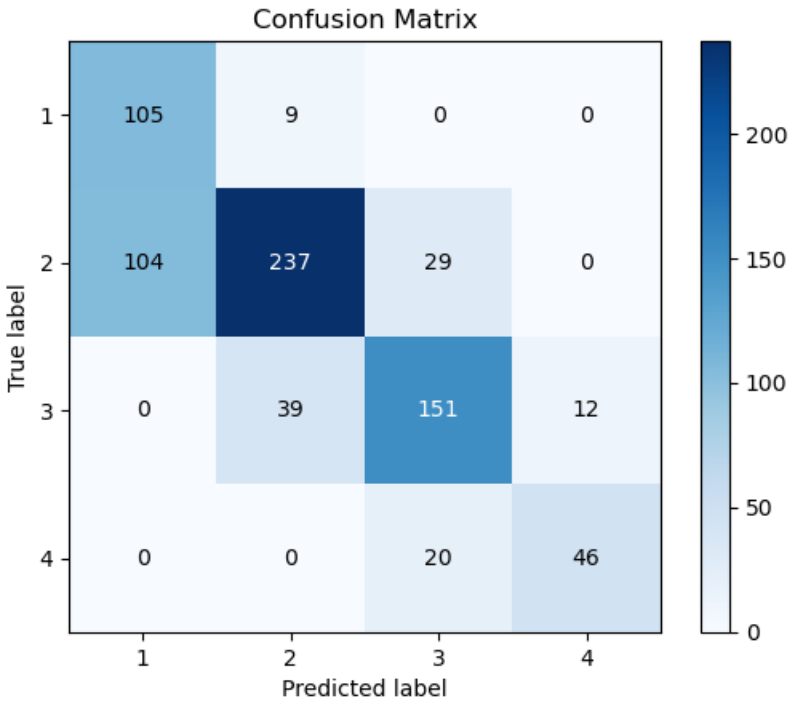
Naive Bayes algoritması ile oluşturduğumuz modeli iyileştirmek için eğriyi genişletmekte kullanılan bir stabilite hesaplaması olan "var_smoothing" hiperparametresinin ayarlanması gerekmektedir. Bu hiperparametre ile dağılım ortalamasından uzakta olan örnekler de hesaba katıldığı için modelin kararlılığının artması sağlanmıştır. Beş farklı hiperparametre ayarlama yöntemi kullanılarak 0 ile -9 arasında eşit aralıklı 100 hiperparametre değeri denenmiş ve beş farklı yöntemle elde edilen hiperparametre değerleri ve bu değerlere ait doğruluk oranları Tablo 4.1' de listelenmiştir. Elde dilen hiperparametre değerleri farklı olmasına rağmen doğruluk oranı değişim göstermemiştir.

Tablo 4.1. Naive Bayes algoritmasına ait hiperparametre değerleri ve doğruluk oranları

Hiperparametreler	**Grid Search**	**Random Search**	**Halvin Grid Search**	**Halvin Random Search**	**Bayes Search**
var_smoothing [0 - (-9)]	8.1113083078e-08	1.873817422e-08	1.2328467394e-09	4.32876128108e-08	1.23284673944e-09
Doğruluk Oranı (%)	71.67	71.67	71.67	71.67	71.67

Naive Bayes algoritmasına ile oluşturduğumuz modele ait karmaşıklık matrisi ile modelin her bir sınıfı tahmin etmekteki başarısı ayrı ayrı görülmektedir. Modele ait

karmaşıklık matrisi Şekil 4.2'de gösterilmiştir. Matrise göre sınıf bazında tahminleri incelendiğinde ikinci sınıfa ait tahminlerin daha kesin olduğu görülmüştür. Karmaşıklık matrisinden elde edilen modele ait sınıflandırma raporunda sınıf bazında ana sınıflandırma ölçütlerinin bir temsilini gösterilmektedir. Bu rapor incelendiğinde en yüksek kesinliğin (precision) %83 ile ikinci sınıfa ait olduğu ve ortalama kesinliğinde %72 olduğu görülmüştür. Geri çağırma değerleri yani olumlu örnekleri doğru bir şekilde bulma yeteneği incelendiğinde en iyi sonucun %92 ile birinci sınıfa ait olduğu ve ortalamanın %75 olduğu görülmüştür. F1 puanına yani hassaslığın ağırlıklı harmonik ortalaması dikkate alındığında en iyi sonucun %75 ile üçüncü sınıfa ait olduğu ortalamanın ise %72 olduğu görülmektedir. Modelin doğruluk oranı da raporda %72 olarak tespit edilmiştir. Modele ait sınıflandırma raporu Şekil 4.3'de gösterilmiştir.



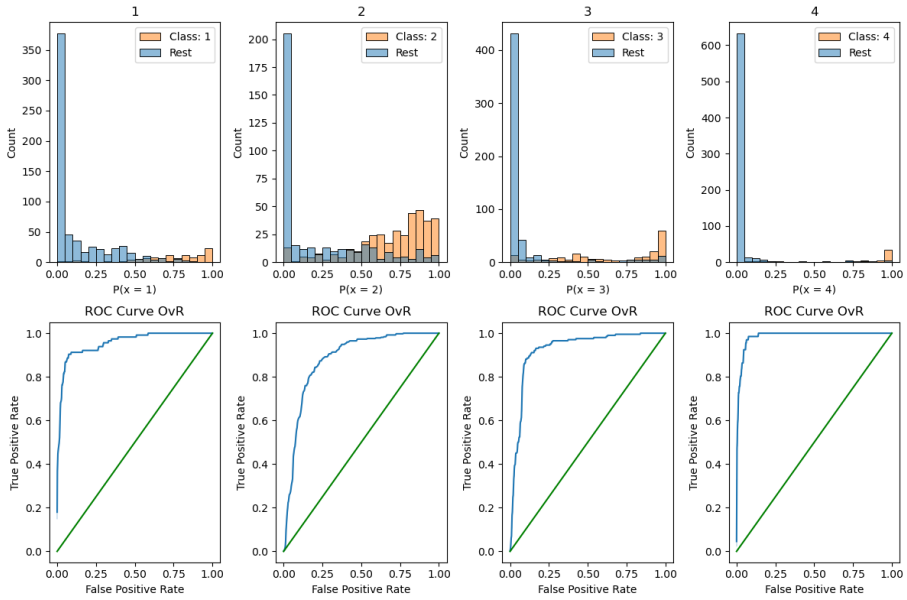
Şekil 4.2. Naive Bayes karmaşıklık matrisi

Sınıflandırma Raporu

	precision	recall	f1-score	support
1	0.50	0.92	0.65	114
2	0.83	0.64	0.72	370
3	0.76	0.75	0.75	202
4	0.79	0.70	0.74	66
accuracy			0.72	752
macro avg	0.72	0.75	0.72	752
weighted avg	0.76	0.72	0.72	752

Şekil 4.3.Naive Bayes Sınıflandırma raporu

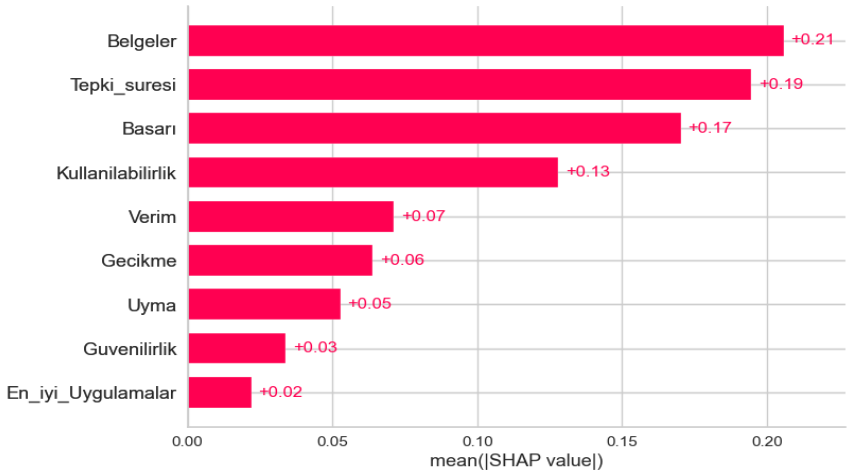
Modelimizde dört farklı sınıf olduğu için her bir sınıfa ait ROC-AUC değerleri farlı oluşmuştur. Modelimize ait ROC-AUC değerleri 1.Sınıf için 0.9556, 2.Sınıf için 0.8806, 3.Sınıf için 0.9253, 4. Sınıf için 0.9852 ve ortalama 0.9367 olarak hesaplanmıştır. ROC-AUC değerlerine bakıldığında modelimizi genel olarak iyi olduğu yani ayırt ediciliğinin iyi olduğu ve iyi tahmin yaptığı görülmektedir. Bununla beraber Şekil 4.4'de yer alan ROC eğrisi grafikleri incelendiğinde her tüm eğrilerin dikliğinin iyi olduğu yani ideal nokta olan sol üst köşeye yakınlıklarının iyi olduğu görülmektedir.



Şekil 4.4. Naive Bayes ROC eğrisi

4.2 K-en Yakın Komşular Sınıflandırma Algoritması

KNN algoritmaları, 1967 yılında T. M. Cover ve P. E. Hart tarafından önerilmiştir [38]. Algoritma, sınıfları belli olan bir örnek veri kümesindeki verilerden yararlanılarak kullanılmaktadır. Kısaca KNN algoritması sınıflandırılmak istenen bir veriyi daha önceki verilerle olan yakınlık ilişkisine göre sınıflandıran bir algoritmadır. Modelimiz KNN algoritması ile çalıştırıldıktan sonra QWS veriseti parametrelerinin ağırlıkları tespit edilmiştir. Parametrelerin model tahmini üzerindeki ağırlıkları Şekil 4.5'de gösterilmiştir. En yüksek ağırlığa sahip ilk altı parametre olan Belgeler, Tepki süresi, Başarı, Kullanılabilirlik, Verim ve Uyma deneylerde kullanılmak üzere seçilmiştir.



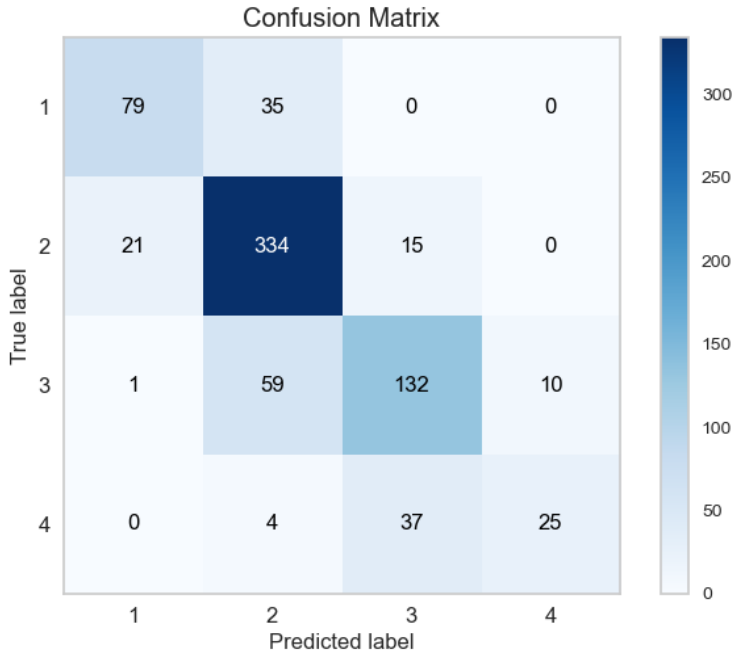
Şekil 4.5. KNN algoritmasında QWS veriseti parametrelerinin ağırlıkları

KNN algoritması ile oluşturduğumuz modeli iyileştirmek için en uygun K değerinin tek tek denenerek bulunması gerekir. Kısaca "n_neighbors" hiperparametresinin ayarlanması gerekmektedir. Beş farklı hiperparametre ayarlama yöntemi kullanılarak 1 ile 50 arasında tüm hiperparametre değeri denenmiş ve beş farklı yöntemle elde edilen hiperparametre değerleri ve bu değerlere ait doğruluk oranları Tablo 4.2' de listelenmiştir. Farklı yöntemler ile elde edilen hiperparametre değerleri kullanılarak elde edilen modele ait en iyi doğruluk oranının Halvin Random Search ile elde edilen K=10 değeriyle elde edildiği ve en iyi doğruluk oranın %75.79 olduğu tespit edilmiştir.

Tablo 4.2. KNN algoritmasına ait hiperparametre değerleri ve doğruluk oranları

Hiperparametreler	Grid Search	Random Search	Halvin Grid Search	Halvin Random Search	Bayes Search
n_neighbors [1 - 50]	9	5	1	10	9
Doğruluk Oranı (%)	75	73.67	75.13	75.79	75

KNN algoritmasına ile oluşturduğumuz modele ait karmaşıklık matrisi ile modelin her bir sınıfı tahmin etmekteki başarısı ayrı görülmektedir. Modele ait karmaşıklık matrisi Şekil 4.6'da gösterilmektedir. Matrise göre sınıf bazında tahminleri incelendiğinde ikinci sınıfa ait tahminlerin daha kesin olduğu görülmüştür. Karmaşıklık matrisinden elde edilen modele ait sınıflandırma raporunda sınıf bazında ana sınıflandırma ölçütlerinin bir temsilini gösterilmektedir. Bu rapor incelendiğinde en yüksek kesinliğin %78 ile birinci sınıfa ait olduğu ve ortalama kesinliğinde %75 olduğu görülmüştür. Geri çağırma değerleri yani olumlu örnekleri doğru bir şekilde bulma yeteneği incelendiğinde en iyi sonucun %90 ile ikinci sınıfa ait olduğu ve ortalamanın %66 olduğu görülmektedir. F1 puanına yani hassaslığın ağırlıklı harmonik ortalaması dikkate alındığında en iyi sonucun %83 ile ikinci sınıfa ait olduğu ortalamanın ise %69 olduğu görülmüştür. Modelin doğruluk oranı da raporda %76 olarak görülmektedir. Modele ait sınıflandırma raporu Şekil 4.7'de görüntülenmektedir.



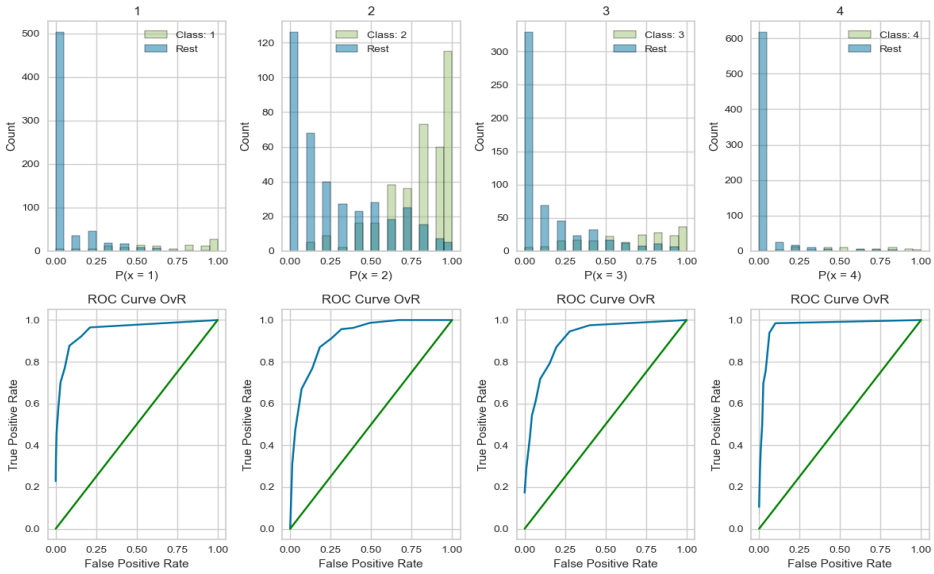
Şekil 4.6. KNN karmaşıklık matrisi

Sınıflandırma Raporu

	precision	recall	f1-score	support
1	0.78	0.69	0.73	114
2	0.77	0.90	0.83	370
3	0.72	0.65	0.68	202
4	0.71	0.38	0.50	66
accuracy			0.76	752
macro avg	0.75	0.66	0.69	752
weighted avg	0.75	0.76	0.75	752

Şekil 4.7.KNN Sınıflandırma raporu

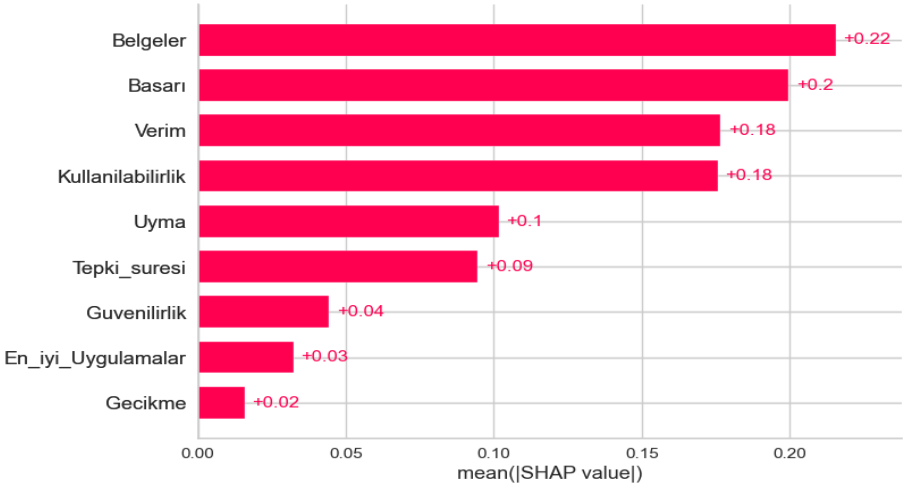
Modelimizde dört farklı sınıf olduğu için her bir sınıfa ait ROC-AUC değerleri farlı oluşmuştur. Modelimize ait ROC-AUC değerleri 1.Sınıf için 0.9503 2.Sınıf için 0.9155, 3.Sınıf için 0.9112, 4. Sınıf için 0.9689 ve ortalama 0.9365 olarak hesaplanmıştır. ROC-AUC değerlerine bakıldığında modelimizi genel olarak iyi olduğu yani ayırt ediciliğinin iyi olduğu ve iyi tahmin yaptığı görülmektedir. Bununla beraber Şekil 4.8'de yer alan ROC eğrisi grafikleri incelendiğinde her tüm eğrilerin dikliğinin iyi olduğu yani ideal nokta olan sol üst köşeye yakınlıklarının iyi olduğu görülmektedir.



Şekil 4.8. KNN ROC eğrisi

4.3 Rassal Orman Sınıflandırma Algoritması

Rassal Orman algoritması hiperparametre ayarı yapmadan bile, çoğu zaman iyi bir sonuç üretebilen, esnek, kullanımı kolay bir makine öğrenmesi algoritmasıdır. Rassal Orman algoritması; birden çok karar ağacı üzerinden her bir karar ağacını farklı bir gözlem örneği üzerinde eğiterek çeşitli modeller üretip, sınıflandırma oluşturmanızı sağlamaktadır [39]. Modelimizi Rassal Orman algoritması ile çalıştırıldıktan sonra QWS veriseti parametrelerinin ağırlıkları tespit edilmiştir. Parametrelerin model tahmini üzerindeki ağırlıkları Şekil 4.9'da gösterilmiştir. En yüksek ağırlığa sahip ilk altı parametre olan Belgeler, Başarı, Verim, Kullanılabilirlik, Uyma ve Tepki süresi deneylerde kullanılmak üzere seçilmiştir.



Şekil 4.9. Rassal Orman algoritmasında QWS veriseti parametrelerinin ağırlıkları

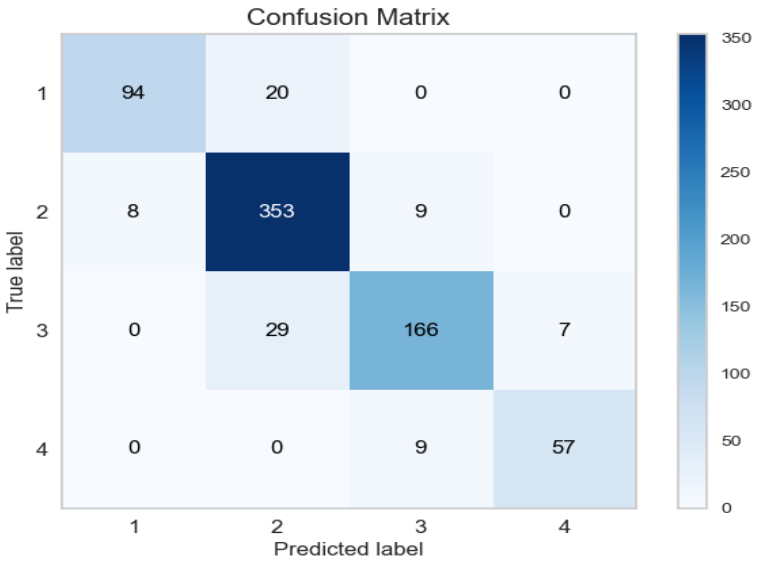
Rassal Orman algoritması ile oluşturduğumuz modeli iyileştirmek en iyi hiperparametre kombinasyonlarının bulunması gerekir. Dört farklı hiperparametrenin en iyi kombinasyonlarını bulmak için beş farklı hiperparametre ayarlama yöntemi kullanılmıştır. Hiperparametrelere ait denenen değerler ile beş farklı yöntemle elde edilen hiperparametre değerlerinin kombinasyonları ve bu kombinasyonlara ait elde edilen doğruluk oranları Tablo 4.3' de listelenmiştir. Farklı yöntemler ile elde edilen hiperparametre değerleri kullanılarak elde edilen modele ait en iyi doğruluk oranının Grid Search ile bulunan hiperparametre değerleri kombinasyonuyla elde edildiği ve en iyi sonucun %89.62 olduğu tespit edilmiştir.

Tablo 4.3. Rassal Orman algoritmasına ait hiperparametre değerleri ve doğruluk oranları

Hiperparametreler	**Grid Search**	**Random Search**	**Halvin Grid Search**	**Halvin Random Search**	**Bayes Search**
max_depth [2,5,8,10]	10	10	10	8	10
max_features [2,5,8]	2	2	5	5	2
n_estimators [10,500,1000]	500	500	1000	500	1000
min_samples_split [2,5,10]	5	10	2	2	2
Doğruluk Oranı (%)	89.62	88.43	89.09	88.69	88.96

Rassal Orman algoritmasına ile oluşturduğumuz modele ait karmaşıklık matrisi ile modelin her bir sınıfı tahmin etmekteki başarısı ayrı görülmektedir. Modele ait karmaşıklık matrisi Şekil 4.10'da gösterilmiştir. Matrise göre sınıf bazında tahminleri incelendiğinde ikinci sınıfa ait tahminlerin daha kesin olduğu görülmektedir. Karmaşıklık matrisinden elde edilen modele ait sınıflandırma raporunda sınıf bazında ana sınıflandırma ölçütlerinin bir temsilini gösterilmektedir. Bu rapor incelendiğinde En yüksek kesinliğin %92 ile birinci sınıfa ait olduğu ve ortalama kesinliğinde %90 olduğu görülmüştür. Geri çağırma değerleri yani olumlu örnekleri doğru bir şekilde bulma yeteneği incelendiğinde en iyi sonucun %95 ile ikinci sınıfa ait olduğu ve ortalamanın %87 olduğu görülmüştür. F1 puanına yani hassaslığın ağırlıklı harmonik ortalaması dikkate alındığında en iyi sonucun %91 ile ikinci sınıfa ait olduğu

ortalamanın ise %88 olduğu görülmüştür. Modelin doğruluk oranı da raporda %89 olduğu görülmüştür. Modele ait sınıflandırma raporu Şekil 4.11'de gösterilmiştir.



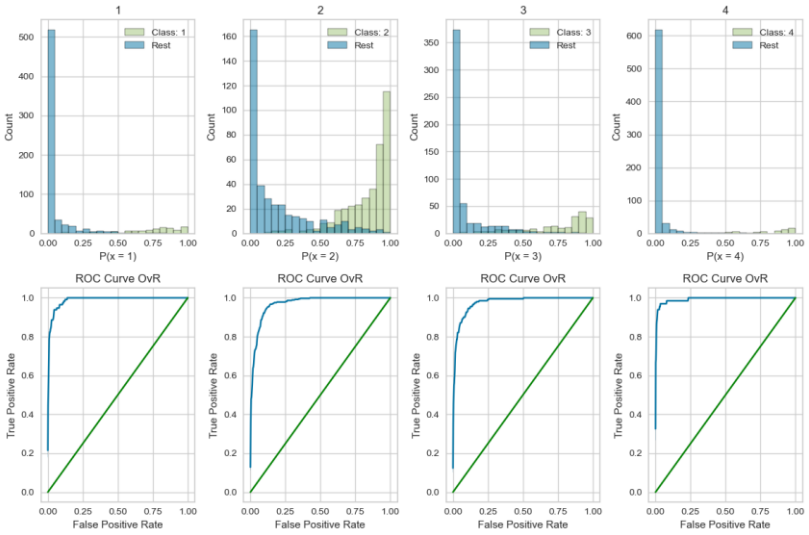
Şekil 4.10. Rassal Orman karmaşıklık matrisi

Sınıflandırma Raporu

	precision	recall	f1-score	support
1	0.92	0.82	0.87	114
2	0.88	0.95	0.91	370
3	0.90	0.82	0.86	202
4	0.89	0.86	0.88	66
accuracy			0.89	752
macro avg	0.90	0.87	0.88	752
weighted avg	0.89	0.89	0.89	752

Şekil 4.11. Rassal Orman Sınıflandırma raporu

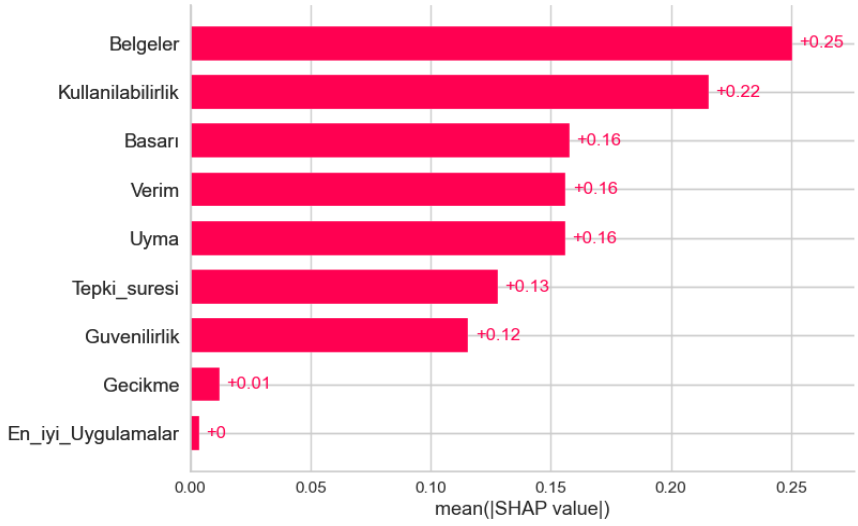
Modelimizde dört farklı sınıf olduğu için her bir sınıfa ait ROC-AUC değerleri farklı oluşmuştur. Modelimize ait ROC-AUC değerleri 1.Sınıf için 0.9879 3 2.Sınıf için 0.9693, 3.Sınıf için 0.9750, 4. Sınıf için 0.9925 ve ortalama 0.9812 olarak hesaplanmıştır. ROC-AUC değerlerine bakıldığında modelimizi genel olarak iyi olduğu yani ayırt ediciliğinin çok iyi olduğu ve çok iyi tahmin yaptığı görülmektedir. Bununla beraber Şekil 4.12'de yer alan ROC eğrisi grafikleri incelendiğinde her tüm eğrilerin dikliğinin çok iyi olduğu yani ideal nokta olan sol üst köşeye yakınlıklarının çok iyi olduğu görülmektedir.



Şekil 4.12. Rassal Orman ROC eğrisi

4.4 Destek Vektör Makinesi Sınıflandırma Algoritması

Destek Vektör Makineleri 1963 yılında Vladimir Vapnik ve Alexey Chervonenkis tarafından temelleri atılan istatiksel öğrenme teorisine dayalı bir gözetimli öğrenme algoritmasıdır [40]. Temel olarak bir doğru ile iki sınıfa ait verileri birbirinden en uygun şekilde ayırarak iki sınıfının noktalarının doğruya maksimum uzaklıkta olmasını amaçlar. Modelimizi SVM algoritması ile çalıştırıldıktan sonra QWS veriseti parametrelerinin ağırlıkları tespit edilmiştir. Parametrelerin model tahmini üzerindeki ağırlıkları Şekil 4.13'de gösterilmektedir. En yüksek ağırlığa sahip ilk altı parametre olan Belgeler, Kullanılabilirlik, Başarı, Verim, Uyma ve Tepki süresi deneylerde kullanılmak üzere seçilmiştir.



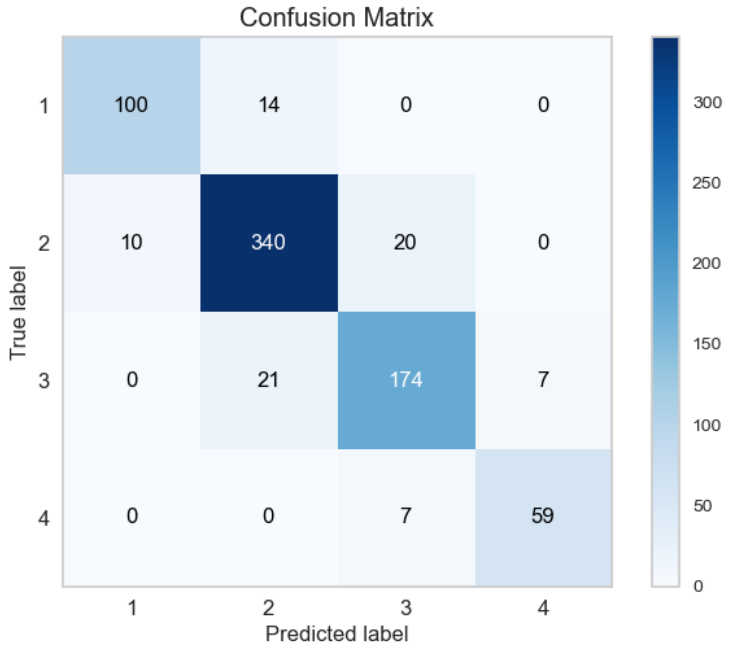
Şekil 4.13. SVM algoritmasında QWS veriseti parametrelerinin ağırlıkları

SVM algoritması ile oluşturduğumuz modeli iyileştirmek en iyi hiperparametre kombinasyonlarının bulunması gerekir. SVM de C hiperparametresi yanlış sınıflandırmayı temsil eden ceza parametresidir. C hiperparametresi, SVM optimizasyonuna ne kadar hatanın göze alınabileceğini söyler. Karar sınırı ile yanlış sınıflandırma dengesi bu şekilde kontrol edilebilir. Karar sınırına olan mesafe ile doğru orantılıdır. C hiperparametresi için en iyi değerleri bulmak için beş farklı hiperparametre ayarlama yöntemi kullanılmıştır. C hiperparametresi için 1 ile 10 arasındaki tüm tamsayılar beş farklı hiperparametre ayarlama yöntemi ile denenmiş elde edilen hiperparametre değerleri ve bu kombinasyonlara ait doğruluk oranları Tablo 4.4' de listelenmiştir. Farklı yöntemler ile elde edilen hiperparametre değerleri kullanılarak elde edilen modele ait en iyi doğruluk oranının Halvin Random Search ile bulunan C=9 değeri ile elde edildiği en iyi sonucun %89.62 olduğu tespit edilmiştir.

Tablo 4.4. SVM algoritmasına ait hiperparametre değerleri ve doğruluk oranları

Hiperparametreler	**Grid Search**	**Random Search**	**Halvin Grid Search**	**Halvin Random Search**	**Bayes Search**
C [1-10]	4	4	3	9	4
Doğruluk Oranı (%)	89.22	89.22	89.22	89.48	89.22

SVM ile oluşturduğumuz modele ait karmaşıklık matrisi ile modelin her bir sınıfı tahmin etmekteki başarısı ayrı görülmektedir. Modele ait karmaşıklık matrisi Şekil 4.14'de gösterilmektedir. Matrise göre sınıf bazında tahminleri incelendiğinde ikinci sınıfa ait tahminlerin daha kesin olduğu görülmektedir. Karmaşıklık matrisinden elde edilen modele ait sınıflandırma raporunda sınıf bazında ana sınıflandırma ölçütlerinin bir temsilini gösterilmektedir. Bu rapor incelendiğinde en yüksek kesinliğin %91 ile birinci ve ikinci sınıfa ait olduğu ve ortalama kesinliğinde %89 olduğu görülmektedir. Geri çağırma değerleri yani olumlu örnekleri doğru bir şekilde bulma yeteneği incelendiğinde en iyi sonucun %92 ile ikinci sınıfa ait olduğu ve ortalamanın %89 olduğu görülmektedir. F1 puanına yani hassaslığın ağırlıklı harmonik ortalaması dikkate alındığında en iyi sonucun %91 ile ikinci sınıfa ait olduğu ortalamanın ise %89 olduğu görülmektedir. Modelin doğruluk oranı da raporda %89 olarak görülmektedir. Modele ait sınıflandırma raporu Şekil 4.15'de görüntülenmektedir.



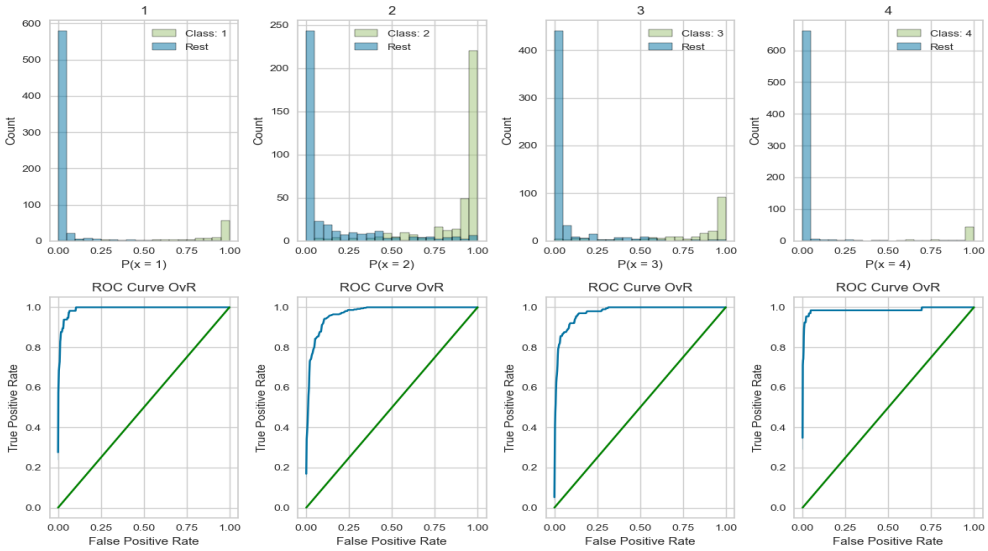
Şekil 4.14. SVM karmaşıklık matrisi

Sınıflandırma Raporu

	precision	recall	f1-score	support
1	0.91	0.88	0.89	114
2	0.91	0.92	0.91	370
3	0.87	0.86	0.86	202
4	0.89	0.89	0.89	66
accuracy			0.89	752
macro avg	0.89	0.89	0.89	752
weighted avg	0.89	0.89	0.89	752

Şekil 4.15. SVM Sınıflandırma raporu

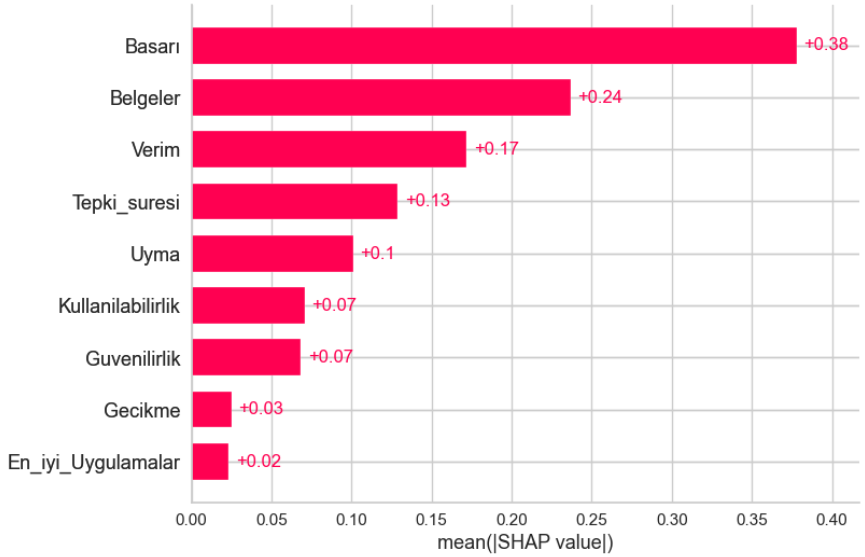
Modelimizde dört farklı sınıf olduğu için her bir sınıfa ait ROC-AUC değerleri farklı oluşmuştur. Modelimize ait ROC-AUC değerleri 1. Sınıf için 0.9912 3 2. Sınıf için 0.9702, 3. Sınıf için 0.9746, 4. Sınıf için 0.9862 ve ortalama 0.9806 olarak hesaplanmıştır. ROC-AUC değerlerine bakıldığında modelimizi genel olarak iyi olduğu yani ayırt ediciliğinin çok iyi olduğu ve çok iyi tahmin yaptığı görülmektedir. Bununla beraber Şekil 4.16'da yer alan ROC eğrisi grafikleri incelendiğinde tüm eğrilerin dikliğinin çok iyi olduğu yani ideal nokta olan sol üst köşeye yakınlıklarının çok iyi olduğu görülmektedir. Bu da modelin sağlamlığını göstermektedir.



Şekil 4.16. SVM ROC eğrisi

4.5 Karar Ağacı Sınıflandırma Algoritması

Karar Ağaçları, sınıflandırma problemlerinde kullanılan en popüler makine öğrenmesi algoritmalarından biridir. Karar ağacı algoritması, veri özelliklerinden basit kurallar çıkarıp daha sonra bu kuralları öğrenerek bir değişkenin değerini tahmin etmektir. Bu algoritma eksik değerleri desteklemediğinden dolayı modeli oluşturmadan önce eksik değerlerin çözülmesi gerekir. Modelimizi Karar Ağacı Sınıflandırma Algoritması ile çalıştırıldıktan sonra QWS veriseti parametrelerinin ağırlıkları tespit edilmiştir. Parametrelerin model tahmini üzerindeki ağırlıkları Şekil 4.17'de gösterilmektedir. En yüksek ağırlığa sahip ilk altı parametre olan Başarı, Belgeler, Verim, Tepki süresi, Uyma ve Kullanılabilirlik deneylerde kullanılmak üzere seçilmiştir.



Şekil 4.17. Karar Ağacı algoritmasında QWS veriseti parametrelerinin ağırlıkları

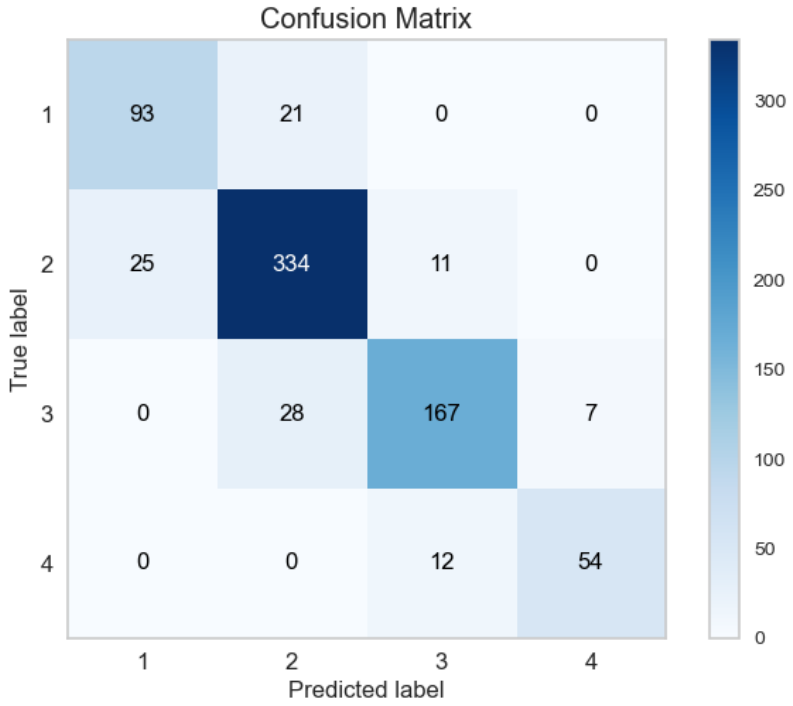
Karar Ağacı Algoritması ile oluşturduğumuz modeli iyileştirmek en iyi hiperparametre kombinasyonlarının bulunması gerekir. Dört farklı hiperparametrenin en iyi kombinasyonlarını bulmak için beş farklı hiperparametre ayarlama yöntemi kullanılmıştır. Hiperparametrelere ait denenen değerler ile beş farklı yöntemle elde edilen hiperparametre değerlerinin kombinasyonları ve bu kombinasyonlara ait doğruluk oranları Tablo 4.5' de listelenmiştir. Tabloda görüldüğü üzere farklı yöntemlere farklı hiperparametre değerleri hesaplamasında en iyi sonucun Halvin Random Search ile bulunan hiperparametre değerleri kombinasyonuyla elde edildiği ve en iyi doğruluk oranın %86.17 olduğu görülmektedir.

Tablo 4.5. Karar Ağacı algoritmasına ait hiperparametre değerleri ve doğruluk oranları

Hiperparametreler	**Grid Search**	**Random Search**	**Halvin Grid Search**	**Halvin Random Search**	**Bayes Search**
max_depth [1-10]	9	9	9	9	9
min_samples_split [2-50]	5	8	5	4	9
Doğruluk Oranı (%)	85.37	85.23	85.37	86.17	84.44

Karar Ağacı Algoritması ile oluşturduğumuz modele ait karmaşıklık matrisi ile modelin her bir sınıfı tahmin etmekteki başarısı ayrı görülmektedir. Modele ait

karmaşıklık matrisi Şekil 4.18'de gösterilmektedir. Matrise göre sınıf bazında tahminleri incelendiğinde ikinci sınıfa ait tahminlerin daha kesin olduğu görülmektedir. Karmaşıklık matrisinden elde edilen modele ait sınıflandırma raporunda, sınıf bazında ana sınıflandırma ölçütlerinin bir temsilini gösterilmektedir. Bu rapor incelendiğinde en yüksek kesinliğin %89 ile dördüncü sınıfa ait olduğu ve ortalama kesinliğinde %86 olduğu görülmektedir. Geri çağırma değerleri yani olumlu örnekleri doğru bir şekilde bulma yeteneği incelendiğinde en iyi sonucun %90 ile ikinci sınıfa ait olduğu ve ortalamanın %84 olduğu görülmektedir. F1 puanına yani hassaslığın ağırlıklı harmonik ortalaması dikkate alındığında en iyi sonucun %89 ile ikinci sınıfa ait olduğu ortalamanın ise %85 olduğu görülmektedir. Modelin doğruluk oranı da raporda %86 olarak görülmektedir. Modele ait sınıflandırma raporu Şekil 4.19'da görüntülenmektedir.



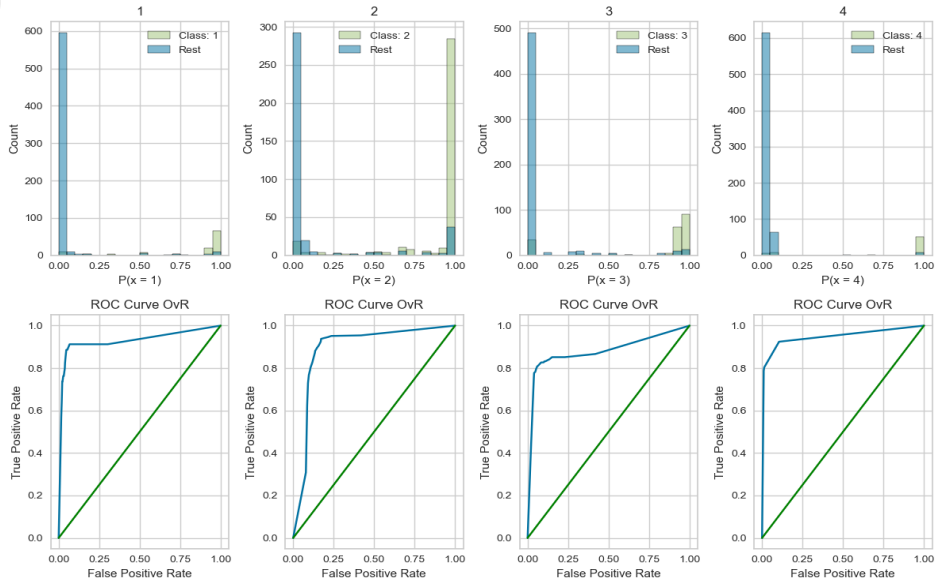
Şekil 4.18. Karar Ağacı karmaşıklık matrisi

Sınıflandırma Raporu

	precision	recall	f1-score	support
1	0.79	0.82	0.80	114
2	0.87	0.90	0.89	370
3	0.88	0.83	0.85	202
4	0.89	0.82	0.85	66
accuracy			0.86	752
macro avg	0.86	0.84	0.85	752
weighted avg	0.86	0.86	0.86	752

Şekil 4.19. Karar Ağacı Sınıflandırma raporu

Modelimizde dört farklı sınıf olduğu için her bir sınıfa ait ROC-AUC değerleri farklı oluşmuştur. Modelimize ait ROC-AUC değerleri 1. Sınıf için 0.9281 3 2. Sınıf için 0.8905, 3. Sınıf için 0.8793, 4. Sınıf için 0.9468 ve ortalama 0.9112 olarak hesaplanmıştır. ROC-AUC değerlerine bakıldığında modelimizin ayırt ediciliğinin iyi olduğu ve iyi tahmin yaptığı görülmektedir. Bununla beraber Şekil 4.20'de yer alan ROC eğrisi grafikleri incelendiğinde 1. ve 4. sınıflara ait eğrilerin dikliğinin iyi olduğu yani ideal nokta olan sol üst köşeye yakınlıklarının çok iyi olduğu görülmektedir.

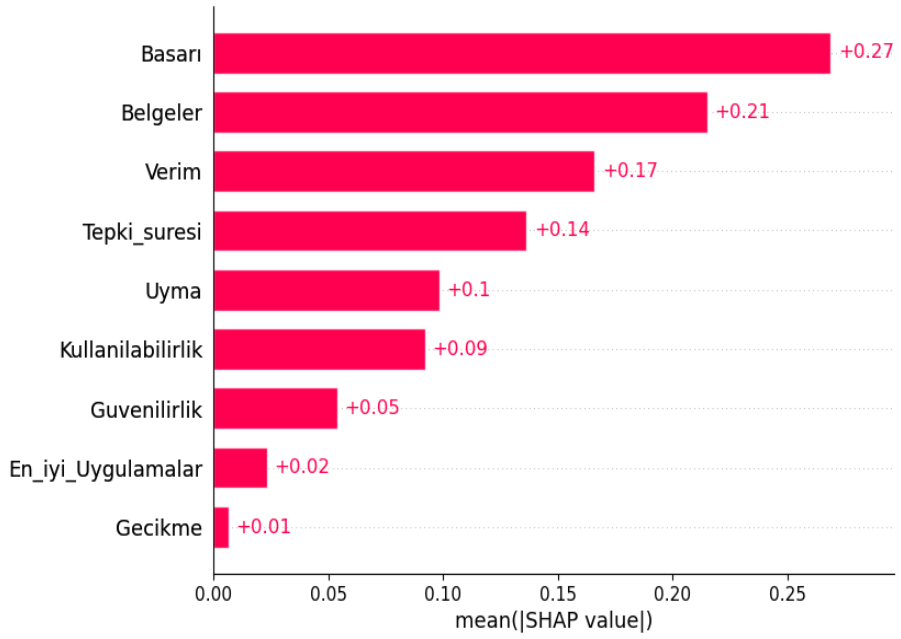


Şekil 4.20. Karar Ağacı ROC eğrisi

4.6 Gradyan Artırma Sınıflandırma Algoritması

Gradyan Artırma algoritması QWS verisetinde bulunan ilk dokuz parametre kullanılarak çalıştırılmış ve parametrelerin model tahmini üzerindeki ağırlıkları tespit

edilmiştir. Şekil 4.21, bu parametreleri ve bunların model üzerindeki etkisinin büyüklüğünü göstermektedir. Nihai modelimizde kullanmak üzere ağırlığı en yüksek altı parametre olan Başarı, Belgeler, Verim, Tepki süresi, Uyma ve Kullanılabilirlik deneylerde kullanılmak üzere seçilmiştir.



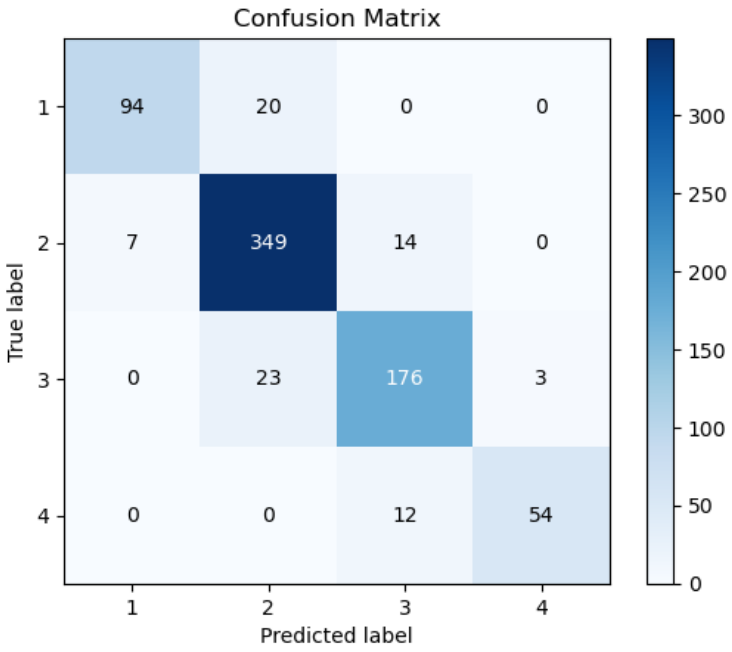
Şekil 4.21. Gradyan Artırma algoritması parametre ağırlıkları

Daha sonra modelimizi geliştirmek için 5 farklı hiperparametre ayarlama yöntemi kullanılarak çalıştırılmış ve en uygun hiperparametre değerleri ile bu değerler ile elde edilen doğruluk oranları elde edilmiştir. Tüm hiperparametreler ile hiperparametrelere ait değerler ile hiperparametre ayarlama yöntemleriyle elde edilen değerler kombinasyonu ve bu kombinasyonlara ait doğruluk oranları Tablo 4.6'da gösterilmektedir. En iyi sonuç Grid Search yöntemiyle elde edilen hiperparametre değerleri kombinasyonu ile hesaplanan %90.02'lik sonuçtur.

Tablo 4.6. Gradyan Artırma algoritması hiperparametre değerleri ve doğruluk oranları

Hiperparametreler	Grid Search	Random Search	Halvin Grid Search	Halvin Random Search	Bayes Search
learning_rate [0.001, 0.01, 0.1, 0.05]	0.05	0.1	0.1	0.05	0.05
n_estimators [100,150,200,500]	150	100	150	500	200
max_depth [3,5,10]	10	5	5	10	5
max_features ["sqrt", "log2"]	log2	sqrt	sqrt	log2	log2
min_samples_split [2,3,5,10]	3	5	5	3	2
Subsample [0.6-1.0]	0.6	0.8	0.6	0.6	0.6
Doğruluk Oranı (%)	90.02	89.49	89.36	89.49	89.36

Gradyan Artırma algoritmasına ile oluşturduğumuz modele ait karmaşıklık matrisi ile modelin her bir sınıfı tahmin etmekteki başarısı ayrı görülmektedir. Modele ait karmaşıklık matrisi Şekil 4.22'de gösterilmektedir. Matrise göre sınıf bazında tahminleri incelendiğinde ikinci sınıfa ait tahminlerin daha kesin olduğu görülmektedir. Karmaşıklık matrisinden elde edilen modele ait sınıflandırma raporunda sınıf bazında ana sınıflandırma ölçütlerinin bir temsilini gösterilmektedir. Bu rapor incelendiğinde en yüksek kesinliğin %95 ile dördüncü sınıfa ait olduğu ve ortalama kesinliğinde %92 olduğu görülmektedir. Geri çağırma değerleri yani olumlu örnekleri doğru bir şekilde bulma yeteneği incelendiğinde en iyi sonucun %96 ile ikinci sınıfa ait olduğu ve ortalamanın %87 olduğu görülmektedir. F1 puanına yani hassaslığın ağırlıklı harmonik ortalaması dikkate alındığında en iyi sonucun %92 ile ikinci sınıfa ait olduğu ortalamanın ise %89 olduğu görülmektedir. Modelin doğruluk oranı da raporda %90 olarak görülmektedir. Modele ait sınıflandırma raporu Şekil 4.23'de görüntülenmektedir.



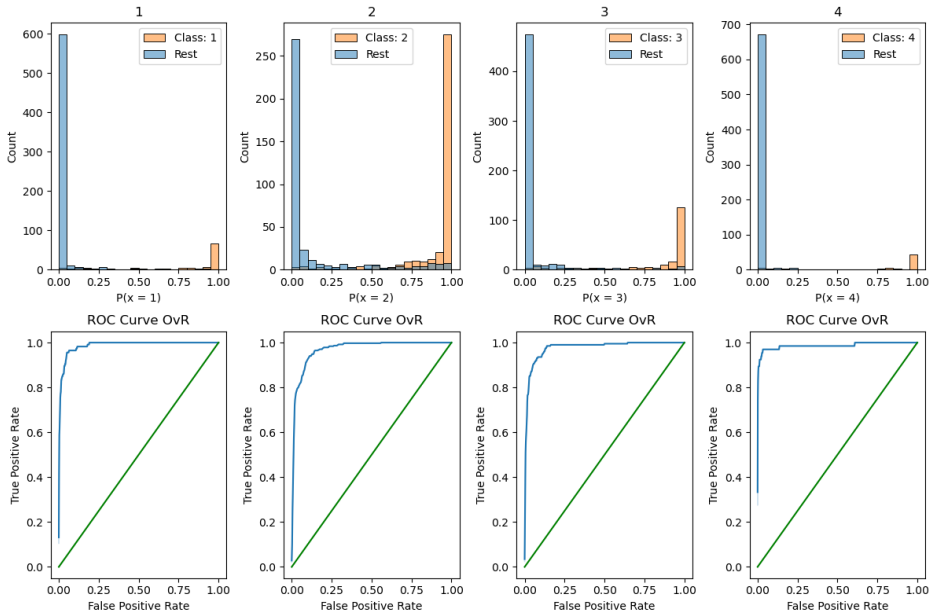
Şekil 4.22. Gradyan Artırma algoritması karmaşıklık matrisi

Sınıflandırma RAporu

	precision	recall	f1-score	support
1	0.93	0.82	0.87	114
2	0.89	0.96	0.92	370
3	0.90	0.86	0.88	202
4	0.95	0.85	0.90	66
accuracy			0.90	752
macro avg	0.92	0.87	0.89	752
weighted avg	0.90	0.90	0.90	752

Şekil 4.23. Gradyan Artırma algoritması sınıflandırma raporu

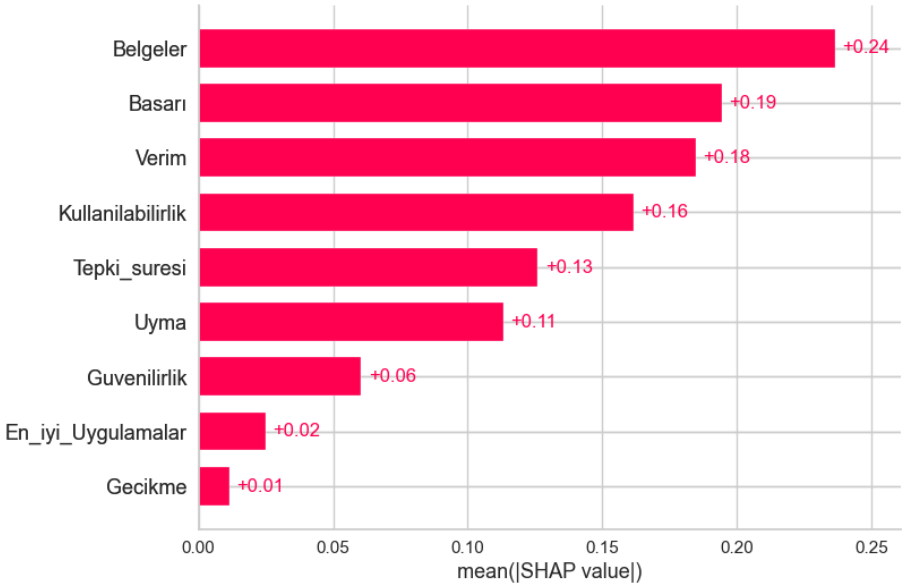
Modelimizde dört farklı sınıf olduğu için her bir sınıfa ait ROC-AUC değerleri farklı oluşmuştur. Modelimize ait ROC-AUC değerleri 1. Sınıf için 0.9872 3 2. Sınıf için 0.9691, 3. Sınıf için 0.9766, 4. Sınıf için 0.99867 ve ortalama 0.9799 olarak hesaplanmıştır. ROC-AUC değerlerine bakıldığında modelimizi genel olarak iyi olduğu yani ayırt ediciliğinin iyi olduğu ve iyi tahmin yaptığı görülmektedir. Bununla beraber Şekil 4.24'da yer alan ROC eğrisi grafikleri incelendiğinde 1. ve 4. sınıflara ait eğrilerin dikliğinin iyi olduğu yani ideal nokta olan sol üst köşeye yakınlıklarının çok iyi olduğu görülmektedir.



Şekil 4.24. Gradyan Artırma ROC eğrisi

4.7 XGBoost Sınıflandırma Algoritması

XGBoost Algoritması QWS verisetinde bulunan ilk dokuz parametre kullanılarak çalıştırılmış ve parametrelerin model tahmini üzerindeki ağırlıkları tespit edilmiştir. Şekil 4.25, bu parametreleri ve bunların model üzerindeki etkisinin büyüklüğünü göstermektedir. Nihai modelimizde kullanmak üzere ağırlığı en yüksek altı parametre olan Belgeler, Başarı, Verim, Kullanılabilirlik, Tepki süresi ve Uyma deneylerde kullanılmak üzere seçilmiştir.



Şekil 4.25. XGBoost algoritması parametre ağırlıkları

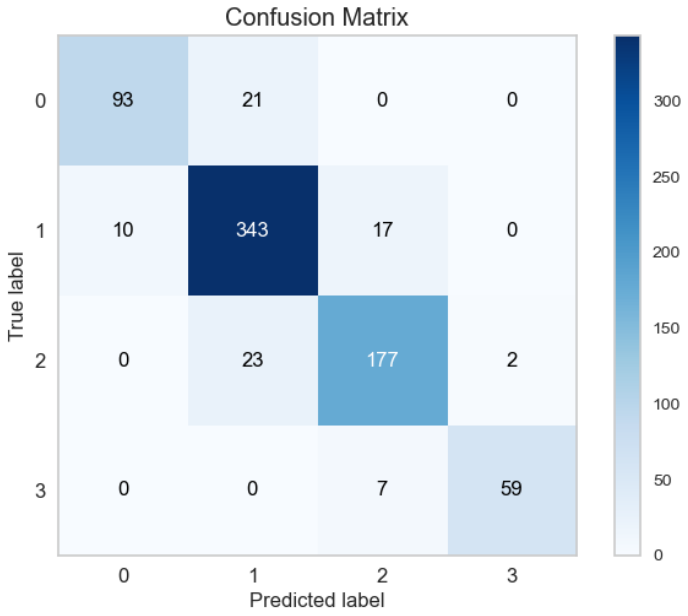
Daha sonra modelimizi geliştirmek için 5 farklı hiperparametre ayarlama yöntemi kullanılarak çalıştırılmış ve en uygun hiperparametre değerleri ile bu değerler ile elde edilen doğruluk oranları elde edilmiştir. Tüm hiperparametreler ile hiperparametrelere ait değerler ile hiperparametre ayarlama yöntemleriyle elde edilen değerler kombinasyonu ve bu kombinasyonlara ait doğruluk oranları Tablo 4.7'de gösterilmektedir. En iyi sonuç Halvin Random Search yöntemiyle elde edilen hiperparametre değerleri kombinasyonu ile hesaplanan %89.36'lık sonuçtur.

Tablo 4.7. XGBoost algoritması hiperparametre değerleri ve doğruluk oranları

Hiperparametreler	**Grid Search**	**Random Search**	**Halvin Grid Search**	**Halvin Random Search**	**Bayes Search**
min_child_weight [1,3,5]	3	1	1	1	3
Gamma [0.1,0.2,0.3,0.4,0.5]	0.1	0.4	0.2	0.3	0.4
max_depth [3,5,7,9]	3	5	3	5	3
colsample_bytree [0.6,0.7,0.8,0.9,1.0]	0.9	0.8	0.9	0.9	1.0
reg_alpha [0, 0.001, 0.005, 0.01, 0.05]	0.01	0.05	0.001	0.05	0.05
Subsample [0.6,0.7,0.8,0.9,1.0]	0.9	0.7	0.8	0.7	0.9
Doğruluk Oranı (%)	88.96	88.82	89.09	89.36	89.22

XGBoost algoritması ile oluşturduğumuz modele ait karmaşıklık matrisi ile modelin her bir sınıfı tahmin etmekteki başarısı ayrı görülmektedir. Modele ait karmaşıklık matrisi Şekil 4.26'da gösterilmektedir. Matrise göre sınıf bazında tahminleri incelendiğinde ikinci sınıfa ait tahminlerin daha kesin olduğu görülmektedir. Karmaşıklık matrisinden elde edilen modele ait sınıflandırma raporunda sınıf bazında ana sınıflandırma ölçütlerinin bir temsilini gösterilmektedir. Bu rapor incelendiğinde en yüksek kesinliğin %97 ile dördüncü sınıfa ait olduğu ve ortalama kesinliğinde %91 olduğu görülmektedir. Geri çağırma değerleri yani olumlu örnekleri doğru bir şekilde bulma yeteneği incelendiğinde en iyi sonucun %93 ile ikinci sınıfa ait olduğu ve ortalamanın %88 olduğu görülmektedir. F1 puanına yani hassaslığın ağırlıklı harmonik ortalaması dikkate alındığında en iyi sonucun %93 ile dördüncü sınıfa ait olduğu ortalamanın ise %89 olduğu görülmektedir. Modelin

doğruluk oranı da raporda %89 olarak görülmektedir. Modele ait sınıflandırma raporu Şekil 4.27'de görüntülenmektedir.



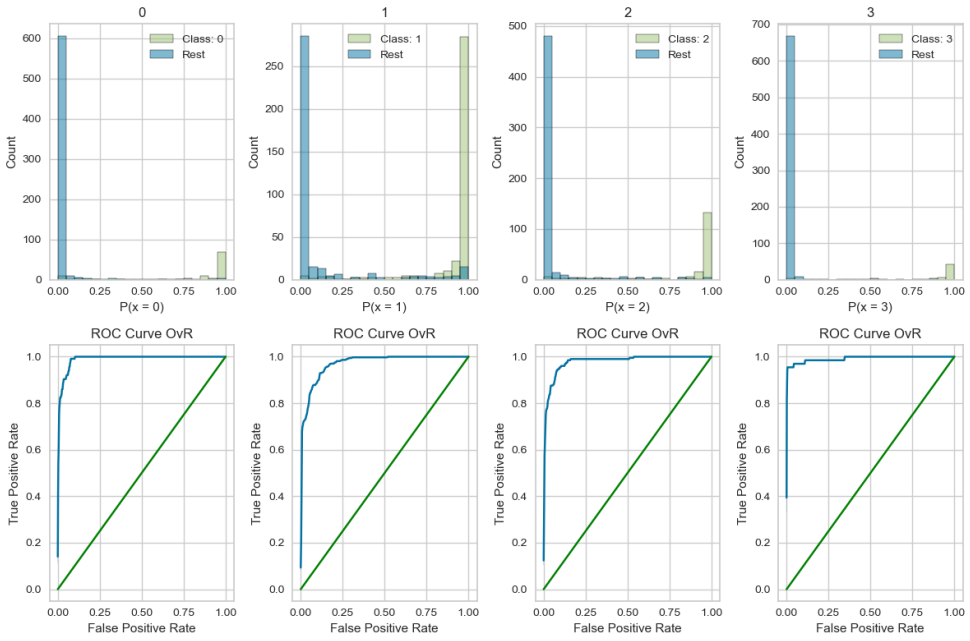
Şekil 4.26. XGBoost algoritması karmaşıklık matrisi

Sınıflandırma Raporu

	precision	recall	f1-score	support
0	0.90	0.82	0.86	114
1	0.89	0.93	0.91	370
2	0.88	0.88	0.88	202
3	0.97	0.89	0.93	66
accuracy			0.89	752
macro avg	0.91	0.88	0.89	752
weighted avg	0.89	0.89	0.89	752

Şekil 4.27. XGBoost algoritması sınıflandırma raporu

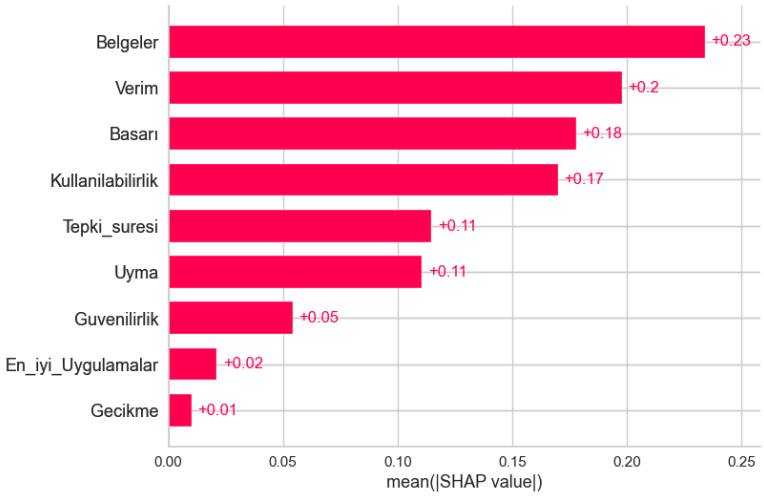
Modelimizde dört farklı sınıf olduğu için her bir sınıfa ait ROC-AUC değerleri farklı oluşmuştur. Modelimize ait ROC-AUC değerleri 1.Sınıf için 0.9888 3 2.Sınıf için 0.9701, 3.Sınıf için 0.9782, 4. Sınıf için 0.9920 ve ortalama 0.9823 olarak hesaplanmıştır. ROC-AUC değerlerine bakıldığında modelimizi genel olarak iyi olduğu yani ayırt ediciliğinin iyi olduğu ve iyi tahmin yaptığı görülmektedir. Bununla beraber Şekil 4.28'de yer alan ROC eğrisi grafikleri incelendiğinde tüm sınıflara ait eğrilerin dikliğinin iyi olduğu yani ideal nokta olan sol üst köşeye yakınlıklarının çok iyi olduğu görülmektedir



Şekil 4.28. XGBoost ROC eğrisi

4.8 LightGBM Sınıflandırma Algoritması

LightGBM algoritması QWS verisetinde bulunan ilk dokuz parametre kullanılarak çalıştırılmış ve parametrelerin model tahmini üzerindeki ağırlıkları tespit edilmiştir. Şekil 4.29, bu parametreleri ve bunların model üzerindeki etkisinin büyüklüğünü göstermektedir. Nihai modelimizde kullanmak üzere ağırlığı en yüksek altı parametre olan Belgeler, Verim, Başarı, Kullanılabilirlik, Tepki süresi ve Uyma deneylerde kullanılmak üzere seçilmiştir.



Şekil 4.29. LightGBM algoritması parametre ağırlıkları

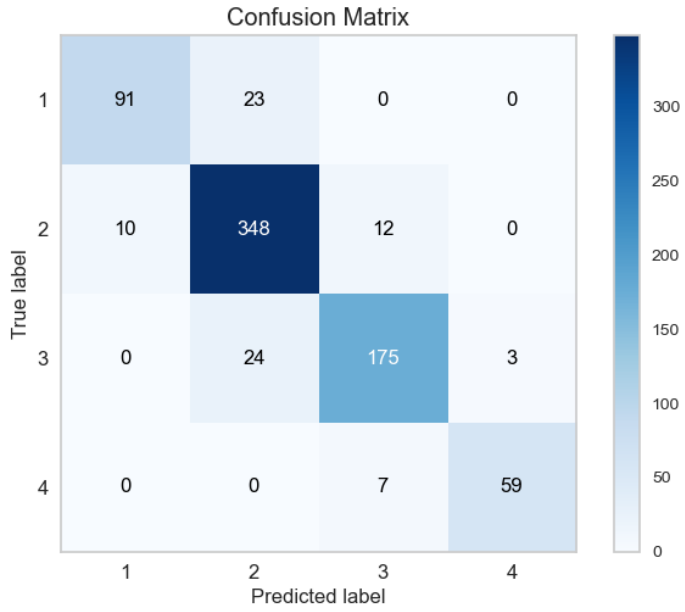
Daha sonra modelimizi geliştirmek için 5 farklı hiperparametre ayarlama yöntemi kullanılarak çalıştırılmış ve en uygun hiperparametre değerleri ile bu değerler ile elde edilen doğruluk oranları elde edilmiştir. Tüm hiperparametreler ile hiperparametrelere ait değerler ile hiperparametre ayarlama yöntemleriyle elde edilen değerler kombinasyonu ve bu kombinasyonlara ait doğruluk oranları Tablo 4.8'de gösterilmektedir. En iyi sonuç Grid Search ve Halvin Random Search yöntemleriyle elde edilen hiperparametre değerleri kombinasyonları ile hesaplanan %89.49'luk sonuçtur.

Tablo 4.8. LightGBM algoritması hiperparametre değerleri ve doğruluk oranları

Hiperparametreler	**Grid Search**	**Random Search**	**Halvin Grid Search**	**Halvin Random Search**	**Bayes Search**
feature_fraction [0.6,0.7,0.8,0.9,1.0]	1.0	0.7	1.0	0.6	0.9
Boosting ['gbdt', 'dart', 'goss']	goss	goss	dart	gbdt	gbdt
max_depth [3, 4, 5,6]	6	5	5	4	6
learning_rate [0.1,0.01,0.02,0.05]	0.1	0.1	0.1	0.1	0.05
reg_lambda [0, 1e-1, 1, 5, 10, 20, 50, 100]	1	0.1	0	10	5
Subsample [0.6, 0.8, 1.0]	0.6	0.8	0.8	0.8	0.6
Doğruluk Oranı (%)	89.49	88.29	88.82	89.49	88.56

LightGBM Algoritması ile oluşturduğumuz modele ait karmaşıklık matrisi ile modelin her bir sınıfı tahmin etmekteki başarısı ayrı görülmektedir. Modele ait karmaşıklık matrisi Şekil 4.30'da gösterilmektedir. Matrise göre sınıf bazında tahminleri incelendiğinde ikinci sınıfa ait tahminlerin daha kesin olduğu görülmektedir. Karmaşıklık matrisinden elde edilen modele ait sınıflandırma raporunda sınıf bazında ana sınıflandırma ölçütlerinin bir temsilini gösterilmektedir. Bu rapor incelendiğinde en yüksek kesinliğin %95 ile dördüncü sınıfa ait olduğu ve ortalama kesinliğinde %91 olduğu görülmektedir. Geri çağırma değerleri yani olumlu örnekleri doğru bir şekilde bulma yeteneği incelendiğinde en iyi sonucun %94 ile

ikinci sınıfa ait olduğu ve ortalamanın %87 olduğu görülmektedir. F1 puanına yani hassaslığın ağırlıklı harmonik ortalaması dikkate alındığında en iyi sonucun %92 ile dördüncü sınıfa ait olduğu ortalamanın ise %89 olduğu görülmektedir. Modelin doğruluk oranı da raporda %89 olarak görülmektedir. Modele ait sınıflandırma raporu Şekil 4.31'de görüntülenmektedir.



Şekil 4.30. LightGBM algoritması karmaşıklık matrisi

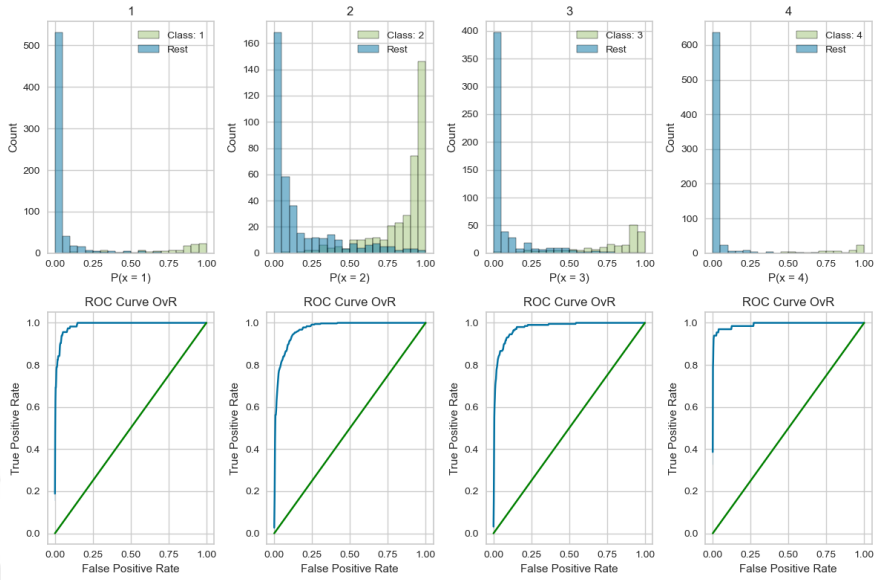
Sınıflandırma Raporu

	precision	recall	f1-score	support
1	0.90	0.80	0.85	114
2	0.88	0.94	0.91	370
3	0.90	0.87	0.88	202
4	0.95	0.89	0.92	66
accuracy			0.89	752
macro avg	0.91	0.87	0.89	752
weighted avg	0.90	0.89	0.89	752

Şekil 4.31. LightGBM algoritması sınıflandırma raporu

Modelimizde dört farklı sınıf olduğu için her bir sınıfa ait ROC-AUC değerleri farklı oluşmuştur. Modelimize ait ROC-AUC değerleri 1. Sınıf için 0.9885, 2. Sınıf için 0.9700, 3. Sınıf için 0.9758, 4. Sınıf için 0.9925 ve ortalama 0.9817 olarak hesaplanmıştır. ROC-AUC değerlerine bakıldığında modelimizin ayırt ediciliğinin iyi olduğu ve iyi tahmin yaptığı görülmektedir. Bununla beraber Şekil 4.32'de yer alan

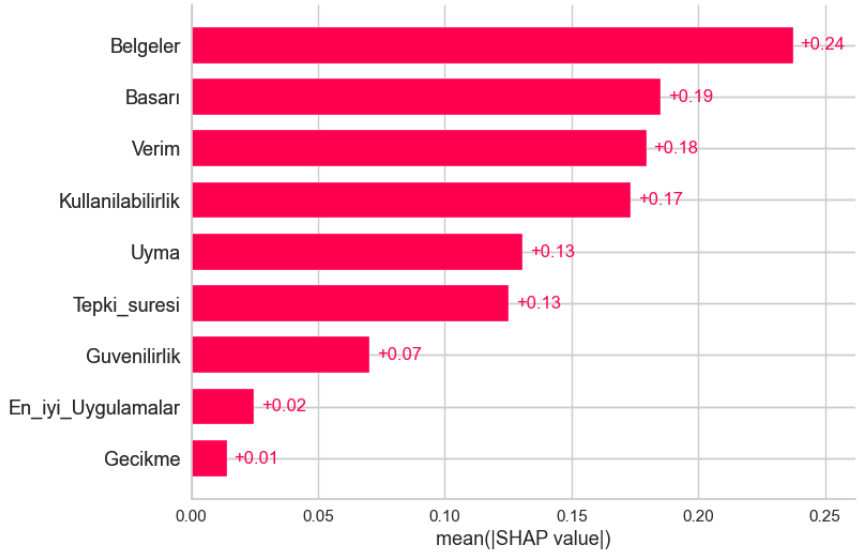
ROC eğrisi grafikleri incelendiğinde tüm sınıflara ait eğrilerin dikliğinin iyi olduğu yani ideal nokta olan sol üst köşeye yakınlıklarının çok iyi olduğu görülmektedir



Şekil 4.32. LightGBM ROC eğrisi

4.9 CatBoost Sınıflandırma Algoritması

CatBoost Algoritması, QWS verisetinde bulunan ilk dokuz parametre kullanılarak çalıştırılmış ve parametrelerin model tahmini üzerindeki ağırlıkları tespit edilmiştir. Şekil 4.33, bu parametreleri ve bunların model üzerindeki etkisinin büyüklüğünü göstermektedir. Nihai modelimizde kullanmak üzere ağırlığı en yüksek altı parametre olan Belgeler, Başarı, Verim, Kullanılabilirlik, Uyma ve Tepki süresi deneylerde kullanılmak üzere seçilmiştir.



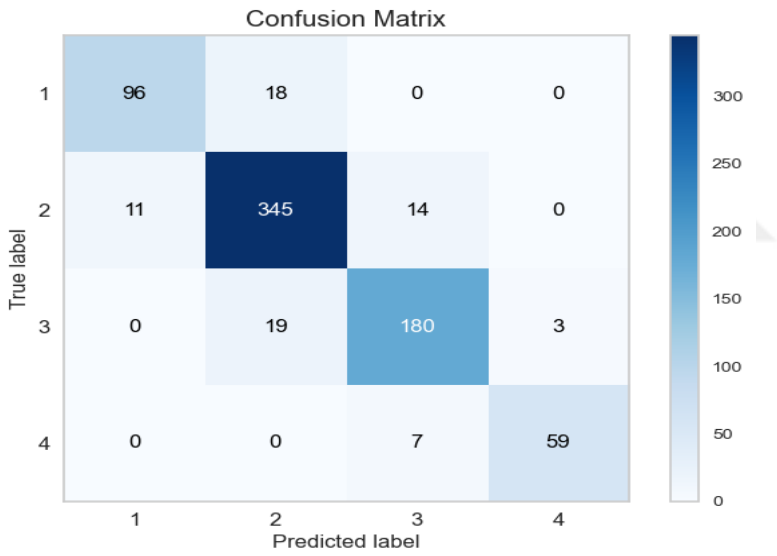
Şekil 4.33. CatBoost algoritması parametre ağırlıkları

Daha sonra modelimizi geliştirmek için 5 farklı hiperparametre ayarlama yöntemi kullanılarak çalıştırılmış ve en uygun hiperparametre değerleri ile bu değerler ile elde edilen doğruluk oranları elde edilmiştir. Tüm hiperparametreler ile hiperparametrelere ait değerler ile hiperparametre ayarlama yöntemleriyle elde edilen değerler kombinasyonu ve bu kombinasyonlara ait doğruluk oranları Tablo 4.9'da gösterilmektedir. En iyi sonuç Bayes Search ve Halvin Grid Search yöntemleriyle elde edilen hiperparametre değerleri kombinasyonları ile hesaplanan %90.42'lik sonuçtur.

Tablo 4.9. CatBoost algoritması hiperparametre değerleri ve doğruluk oranları

Hiperparametreler	**Grid Search**	**Random Search**	**Halvin Grid Search**	**Halvin Random Search**	**Bayes Search**
İterations [200,500]	500	500	500	500	489
learning_rate [0.01,0.05, 0.1]	0.1	0.1	0.1	0.1	0.05
Depth [3,5,8]	3	8	3	8	8
border_count [32,5,10,20,50,100,200]	200	200	100	200	200
l2_leaf_reg [3,1,5,10,100]	3	5	3	5	10
Doğruluk Oranı (%)	90.02	90.02	90.42	90.02	90.42

CatBoost Algoritması ile oluşturduğumuz modele ait karmaşıklık matrisi ile modelin her bir sınıfı tahmin etmekteki başarısı ayrı görülmektedir. Modele ait karmaşıklık matrisi Şekil 4.34'de gösterilmektedir. Karmaşıklık matrisinden elde edilen modele ait sınıflandırma raporunda sınıf bazında ana sınıflandırma ölçütlerinin bir temsilini gösterilmektedir. Bu rapor incelendiğinde en yüksek kesinliğin %95 ile dördüncü sınıfa ait olduğu ve ortalama kesinliğinde %91 olduğu görülmektedir. Geri çağırma değerleri yani olumlu örnekleri doğru bir şekilde bulma yeteneği incelendiğinde en iyi sonucun %93 ile ikinci sınıfa ait olduğu ve ortalamanın %89 olduğu görülmektedir. F1 puanına yani hassaslığın ağırlıklı harmonik ortalaması dikkate alındığında en iyi sonucun %92 ile ikinci ve dördüncü sınıfa ait olduğu ortalamanın ise %90 olduğu görülmektedir. Modelin doğruluk oranı da raporda %90 olarak görülmektedir. Modele ait sınıflandırma raporu Şekil 4.35'de görüntülenmektedir.



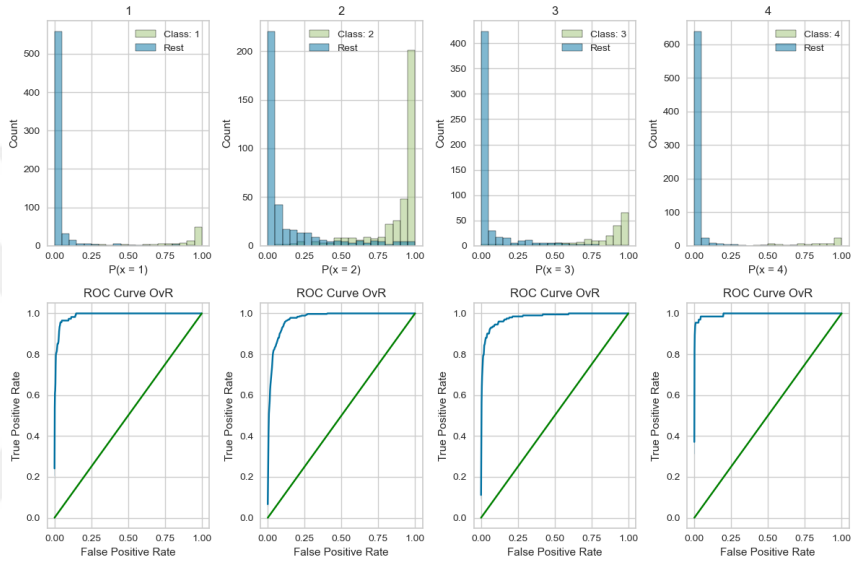
Şekil 4.34. CatBoost algoritması karmaşıklık matrisi

Sınıflandırma Raporu

	precision	recall	f1-score	support
1	0.90	0.84	0.87	114
2	0.90	0.93	0.92	370
3	0.90	0.89	0.89	202
4	0.95	0.89	0.92	66
accuracy			0.90	752
macro avg	0.91	0.89	0.90	752
weighted avg	0.90	0.90	0.90	752

Şekil 4.35. CatBoost algoritması sınıflandırma raporu

Modelimizde dört farklı sınıf olduğu için her bir sınıfa ait ROC-AUC değerleri farklı oluşmuştur. Modelimize ait ROC-AUC değerleri 1. Sınıf için 0.9896, 2. Sınıf için 0.9729, 3. Sınıf için 0.9792, 4. Sınıf için 0.9954 ve ortalama 0.9843 olarak hesaplanmıştır. ROC-AUC değerlerine bakıldığında modelimizi genel olarak iyi olduğu yani ayırt ediciliğinin iyi olduğu ve iyi tahmin yaptığı görülmektedir. Bununla beraber Şekil 4.36'da yer alan ROC eğrisi grafikleri incelendiğinde tüm sınıflara ait eğrilerin dikliğinin iyi olduğu yani ideal nokta olan sol üst köşeye yakınlıklarının çok iyi olduğu görülmektedir.



Şekil 4.36. CatBoost ROC eğrisi

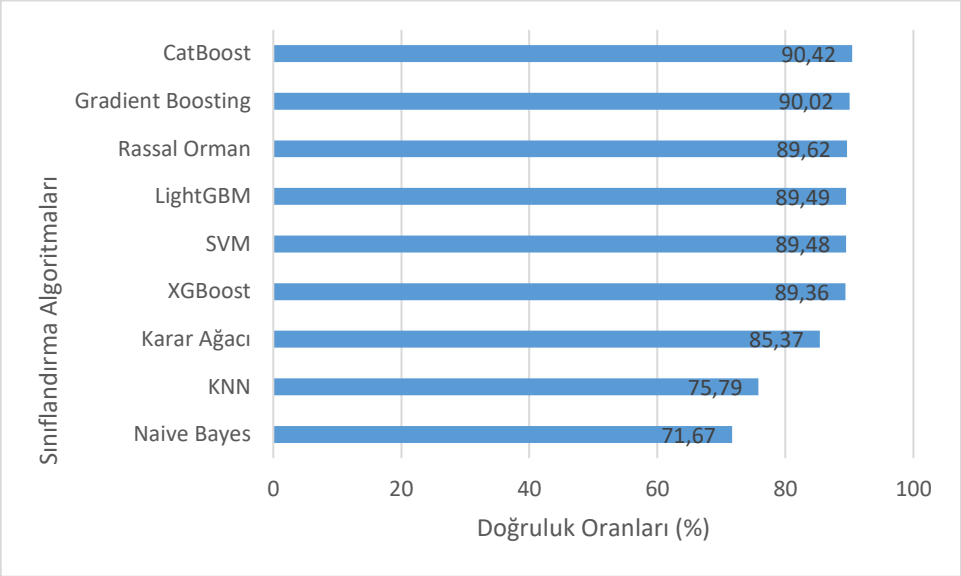
5. TARTIŞMA

Yapılan çalışmada mikroservis ekosisteminde servis keşfi ile ilgili literatürde yapılan çalışmalar, mikroservis keşfi ile ilgili kullanılan yöntemler ve faydalanılan algoritmalar incelenmiştir. Mikroservis ekosisteminde doğru ve en verimli servisi seçmek için sınıflandırma algoritmalarının kullanıldığı görülmüştür. Yapılan sistematik literatür taraması sonucunda başarılı olan sınıflandırma algoritmalarından Rassal Orman, SVM, Decision Tree, KNN ve Naive Bayes algoritmaları önerilen sınıflandırma algoritmaları ile karşılaştırma yapılmak üzere seçilmiştir. Bu algoritmalara alternatif olarak temelinde topluluk öğrenme ilkesine dayalı olarak çalışan yükseltme algoritmaları sunulmaya çalışılmıştır.

İlk olarak literatürden seçilen algoritmaları kendi aralarında değerlendirildiğinde en başarılı algoritmanın Rassal Orman algoritması olduğu görülmüştür. Naive Bayes ve KNN algoritmaları ise deneylerimizde en düşük sonuçların üretildiği algoritmalardır. Rassal Orman ve SVM algoritmalarının diğerlerine göre daha başarılı oluşunun sebepleri incelenmiştir. SVM'nin sınıflandırma problemleri için kullanılan verileri analiz eden ilişkili öğrenme algoritmalarına sahip bir model olduğu karşımıza çıkmaktadır. Rassal Orman Algoritması'nın ise topluluk öğrenmesi ilkesine göre çalışan farklı algoritmaların sonucunda en uyumlu sonucu üreten bir model olduğu karşımıza çıkmaktadır. Buradan yola çıkarak topluluk ilkesine göre çalışan algoritmalar ile ilişkili veriler üzerinde daha iyi çalışan algoritmaların mikroservisleri sınıflandırmada daha başarılı sonuçlar üreteceği söylenebilir.

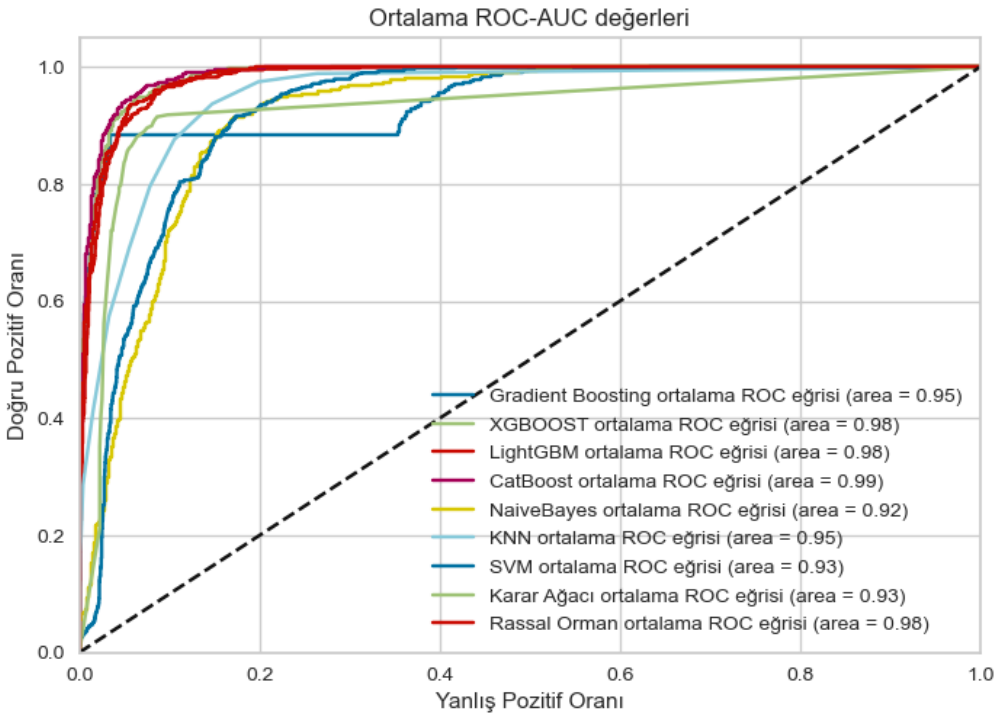
Literatürden seçilen algoritmalara ait sonuçlar ışığında önerilen yükseltme algoritmalarının doğru bir seçim olduğu görülmektedir. Yükseltme algoritmalarının Rassal Orman Algoritması gibi temelinde bir topluluk öğrenmesi yöntemi olması ve bu algoritmaların bir önceki öğrenmede yapılan hataları düzelterek devam etmesi seçim sebeplerindendir. Elde edilen sonuçlara bakıldığı zaman yükseltme algoritmalarının diğer algoritmalara yakın ve onlardan daha iyi sonuçlar ürettiği görülmektedir. Dokuz algoritmanın sonuçları karşılaştırıldığında, ilk sırayı CatBoost algoritmasının aldığı görülmektedir. Şekil 5.1'de deneylerde kullanılan sınıflandırma algoritmaları ile bu algoritmalara ait sonuçları gösterilmektedir. CatBoost algoritması

2017 yılında Yandex tarafından geliştirilmiştir ve bu algoritmalar arasında en son geliştirilen algoritmadır.



Şekil 5.1. Sınıflandırma algoritmaları ile elde edilen en iyi sonuçlar

Gradyan Artırma algoritması, 1999 yılında geliştirildikten sonra XGBoost ve LightGBM algoritmaları 2016 yılında bu algoritmadan geliştirilmiştir. CatBoost algoritması ise 2017 yılında bu algoritmalara alternatif olarak geliştirilen son algoritmadır. Yükseltme algoritmalarının sınıflandırmadaki başarısı bu algoritmaların günümüzde de sürekli geliştirilmeye devam edilmesinin ana sebebidir. Bunun en güzel örneği de Yandex'in geliştirmiş olduğu ve çalışmamızda en başarılı sonucu üreten CatBoost algoritmasıdır. Ayrıca ROC grafiklerindeki AUC değerleri de bu durumu doğrulamaktadır. Tüm algoritmalardan elde edilen AUC değerleri incelendiğinde en iyi sonucun yine CatBoost algoritmasına ait olduğu görülmektedir. Şekil 5.2'de ortalama ROC-AUC değerlerini gösterilmektedir. Yükseltme algoritmalarının kullanımı kolay olmasına karşılık sonuçların iyileştirilmesi için en iyi hiperparametre kombinasyonun bulunması zahmetli ve zor bir iştir. Gradyan Artırma algoritmasının kendisinden geliştirilen XGBoost ve LightGBM 'den çok az farkla da olsa daha iyi sonuç üretmesi verilerle ilişkili olabileceği gibi hiperparametre kombinasyonuyla da ilişkili olabilir.



Şekil 5.2. Ortalama ROC-AUC değerleri

En iyi sonucu elde ettiğimiz CatBoost algoritması, XGBoost ve LightGBM'e alternatif olarak geliştirilmiştir. Adı "Category" ve "Boosting" kelimelerinden türetilmiştir. Yüksek öğrenme hızı hem sayısal hem kategorik hem de metin verileri ile çalışılabilmesi, GPU desteği gibi özellikleri CatBoost algoritmasını diğerlerinden ayıran en önemli özelliklerdir. Kendine has bir kodlama yöntemi olduğu için kategorik verilerle daha yüksek performans ile çalışabilir. Veriler için ön işlemeye ihtiyacı yoktur, hatta ön işlem yapılmaması tavsiye edilir. Ayrıca simetrik ağaçlar kurması sayesinde çok derin ağaçlar kurmadan yüksek tahmin oranı yakalar bu da aşırı öğrenmenin önüne geçer. Bu algoritmanın GPU ile öğrenim yapabilmesi ve çalışırken bellek kopyaları alması diğer iyi özellikleridir. Bu özellikler sayesinde büyük veriler ile daha rahat işlem yapabilmektedir.

Genel olarak bakıldığında yükseltme algoritmalarından elde edilen doğruluk oranlarının yüksek olduğu görülmüştür. Ayrıca elde edilen AUC değerlerinin yüksek oluşu, bu algoritmaların ayırt ediciliğinin yüksek olduğunu göstermiştir. Yükseltme algoritmalarının sağladığı avantajların başında kolay anlaşılabilir ve uygulanabilir olması gelir. Bu algoritmalar, veri ön işlemi gerektirmez ve eksik verilerin ele alınması için yerleşik rutinlere sahiptir. Yükseltme algoritmaları sıralı olarak yani bir önceki

öğrenmeye göre çalıştığı için makine öğrenimi modellerinde yaygın olan yüksek sapmaları azaltır ve eğitim süreci sırasında tahmin doğruluğunu artıran özelliklere öncelik verir. Bu da veri özniteliklerini azaltmaya ve büyük veri kümelerini verimli şekilde kullanmaya yardımcı olur. Bu algoritmanın dezavantajları ise her bir model, bir öncekinin hatalarını düzeltmeye çalıştığından, uç değerlere sahip sonuçları saptırabilir. Ayrıca yükseltme algoritma diğer süreçlerden daha karmaşık olduğundan, bu algoritmaları gerçek zamanlı uygulamalarda kullanırken de zorlanılabilir.

5.1 Geçerliliğe Yönelik Şartlar ve Tehditler

Bu tez çalışmasında yapılan deneyler belli şartlar ve koşullarda yapılmıştır. Yapılan deneylerin geçerlilik şartları aşağıda belirtilmiştir.

- Bu çalışmadaki deneyler QWS veriseti ile yapılmıştır ve elde edilen sonuçlar bu veriseti için geçerlidir.
- Bu çalışmadaki deneylerde her algoritma için verisetindeki parametrelerin ağırlıkları hesaplanmış ve ağırlığı en yüksek altı parametre kullanılmıştır. Bundan dolayı algoritmalar için elde edilen sonuçlar seçilen parametrelerde geçerlidir.
- Her algoritmada sadece belirli hiperparametrelerin değerleri incelenmiş ve ayarlanmıştır. Diğer hiperparametre değerleri varsayılan olarak kullanılmıştır. Elde edilen sonuçlar bu hiperparametreler için verilen değerlerde geçerlidir.

Yapılan deneyler sonucu elde edilen sonuçlarını olumsuz etkileyebilecek bazı unsurlar vardır. Bu unsurlar geçerliliğe yönelik tehditler olarak ele alınmaktadır ve aşağıda listelenmektedir.

- Uygulayıcının seçilen parametrelerin değiştirmesi veya veriseti üzerinde değişiklik yapması sonucu değiştirebilir.
- Uygulayıcının farklı hiperparametrelerin değerlerini değiştirilmesi veya seçilen hiperparametrelere farklı değerler verilmesi sonucu değiştirebilir.
- Test ortamının değiştirilmesi ortaya çıkan eğitim ve test sürelerinin değiştirebilir.

6. SONUÇ

Bu çalışmada mikroservis ekosisteminde en ideal mikroservisleri bulmak için literatürde yer alan sınıflandırma algoritmaları ile önerilen yükseltme algoritmalarının karşılaştırılması yapılmıştır. Deneylerde veriseti olarak QWS verisetinin ikinci versiyonu %70 eğitim ve %30 test olmak üzere ikiye ayrılarak kullanılmıştır. QWS veriseti 2.507 gerçek web servis uygulaması için bir dizi QWS ölçümünden oluşur. Servisler kalite derecelendirmesine göre 1. Platin (Yüksek Kalite), 2. Altın, 3. Gümüş, 4. Bronz (Düşük Kalite) olacak şekilde sınıflandırılmıştır. Servis kalitesinin tahmini için verisetinden her algoritmada model üzerindeki ağırlıkları dikkate alınarak ilk altı parametre seçilmiştir. Dokuz parametre kullanımı yerine altı parametre kullanımı zamandan tasarruf yapılmasını sağlamıştır. Sistematik literatür taraması ile literatürde yapılan çalışmalar incelenmiş ve yükseltme algoritmalarıyla karşılaştırılacak sınıflandırma algoritmaları belirlenmiştir. Bunlar Rassal Orman, SVM, Karar Ağacı, KNN ve Naive Bayes algoritmalarıdır.

Çalışmanın ilk aşamasında literatürde yer alan 5 farklı sınıflandırma algoritması ile deneyler yapılmıştır. Her algoritma model tahmini üzerindeki ağırlığı en fazla olan ilk altı parametre seçilerek önce varsayılan hiperparametre değerleri ile çalıştırılmıştır. Her algoritmanın üreteceği en iyi sonucu tespit etmek için 5 farklı hiperparametre ayarlama yöntemiyle algoritmaların en iyi hiperparametre değerlerinin kombinasyonu bulunmuştur. Deneyler sonucunda elde edilen en düşük doğruluk oranı Naive Bayes algoritması ile elde edilen %71.67 ve en iyi doğruluk oranı ise Rassal Orman algoritması ile elde edilen %89.62 olarak saptanmıştır.

Çalışmanın ikinci aşamasında yükseltme algoritmaları olan Gradyan Artırma, XGBoost, LightGBM ve CatBoost algoritmaları, Qws verisetinden model tahmini üzerindeki ağırlığı en fazla olan ilk altı parametre seçilerek önce varsayılan hiperparametre değerleri ile çalıştırılmıştır. Daha sonra her algoritmanın üreteceği en iyi sonucu tespit etmek için 5 farklı hiperparametre ayarlama yöntemiyle algoritmaların en iyi hiperparametre değerlerinin kombinasyonu bulunmuştur. Yapılan deneylerde Gradyan Artırma ile %90.02, XGBoost ile %89.36, LightGBM ile %89.49 ve CatBoost ile %90.42'lik doğruluk oranları elde edilmiştir. Yükseltme algoritmalarında elde edilen sonuçlar genel olarak incelendiğinde bu algoritmaların başarısı ortaya çıkmıştır.

CatBoost algoritması dokuz parametre ile çalıştırıldığında geçen ortalama eğitim süresi 2334ms, ortalama test süresi 2.669ms ve ortalama doğruluk oranı %93.14 iken altı parametre ile çalıştırıldığında ortalama eğitim süresi 2200ms, ortalama test süresi 2.446ms ve ortalama doğruluk oranı %90.42 olarak ölçülmüştür. Parametre sayısının azaltılması bize eğitim süresi için ortalama %3.2 ve test için ortalama %8.35 avantaj sağlamıştır. Doğruluk oranında ise %2.92 kayıp olmuştur. Kazanılan süreyi kaybedilen doğruluk oranı ile karşılaştırınca mili saniyelik kazanımlar için yaklaşık yüzde üçlük kaybın doğru olamayacağına kanaat getirilmiştir. Son olarak QWS verisetinin ikinci versiyonu ile elde ettiğimiz modelimizi verisetinin birinci versiyonu ile test ettiğimizde %80 doğruluk oranı elde ettik.

Bu çalışma, mikroservis ekosisteminde en iyi servisi bulmak için topluluk ilkesine göre çalışan temelde karar ağaçları tabanlı algoritmalar olan yükseltme algoritmalarının başarısını göstermiştir. Elde edilen doğruluk oranları yükseltme algoritmalarının başarısını ve elde edilen AUC değerleri yükseltme algoritmalarının ayırt ediciliğini doğrulamıştır. Literatürden belirlenen algoritmalar içinde en iyi sonucu elde eden Rassal Orman algoritmasının da topluluk öğrenme yöntemlerinden olması yükseltme algoritmaları ile elde edilen sonuçların tesadüf olmadığını göstermiştir. Sürekli genişleyen mikroservis ekosisteminde istenilen niteliklere sahip servislerin bulması için yükseltme algoritmaları daha çok kullanılacaktır ve yeni geliştirilecek algoritmalar yüksek olasılıkla bu algoritmalardan geliştirilecektir.

KAYNAKÇA

[1] Akbulut, A., & Perros, H. G. (2019). Performance analysis of microservice design patterns. *IEEE Internet Computing*, *23*(6), 19-27.

[2] Olewy, N. A. H. H., & Hadi, A. K. (2021, December). Classifying Quality of Web Services Using Machine Learning Classification and Cross Validation Techniques. In *2021 2nd Information Technology To Enhance e-learning and Other Application (IT-ELA)* (pp. 125-130). IEEE.

[3] Yuan-jie, L., & Jian, C. (2012, May). Web service classification based on automatic semantic annotation and ensemble learning. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum* (pp. 2274-2279). IEEE.

[4] Wang, H., Shi, Y., Zhou, X., Zhou, Q., Shao, S., & Bouguettaya, A. (2010, October). Web service classification using support vector machine. In *2010 22nd IEEE International conference on tools with artificial intelligence* (Vol. 1, pp. 3-6). IEEE.

[5] Liu, J., Tian, Z., Liu, P., Jiang, J., & Li, Z. (2016, June). An approach of semantic web service classification based on Naive Bayes. In *2016 IEEE International Conference on Services Computing (SCC)* (pp. 356-362). IEEE.

[6] E. R. Muratlar, «Gradient Boosted Regresyon Ağaçları,» 24 Ocak 2020. [Çevrimiçi]. Available: https://www.veribilimiokulu.com/gradient-boosted-regresyon-agaclari/. [Erişildi: 21 Kasım 2022].

[7] E. A.-M. v. Q. Mahmoud, «The QWS dataset,» 2007. [Çevrimiçi]. Available: https://qwsdata.github.io. [Erişildi: 10 Kasım 2022].

[8] Al-Debagy, O., & Martinek, P. (2018, November). A comparative review of microservices and monolithic architectures. In *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)* (pp. 000149-000154). IEEE.

[9] A. Kharenko, «Monolithic vs. Microservices Architecture,» 2015 Ekim 9. [Çevrimiçi]. Available: https://medium.com/adopting-microservices-architecture/monolithic-vs-microservices-architecture-5c4848858f59. [Erişildi: 15 Kasım 2022].

[10] G. Ayrancıoğlu, «Monolitik Mimari ve Microservice Mimarisi Arasındaki Farklar,» 2019 Ekim 16. [Çevrimiçi]. Available:

https://gokhana.medium.com/monolitik-mimari-ve-microservice-mimarisi-aras%C4%B1ndaki-farklar-bd89ac5b094a. [Erişildi: 2022 Kasım 16].

[11] A. E. Özçelik, «Microservice Mimarilerinde Service Discovery,» 8 Eylül 2020. [Çevrimiçi]. Available: https://medium.com/i%CC%87yi-programlama/microservice-mimarilerinde-service-discovery-7a6ebceb1b2a. [Erişildi: 17 Kasım 2022].

[12] C. Richardson, «Service Discovery in a Microservices Architecture,» 2015 Ekim 12. [Çevrimiçi]. Available: https://www.nginx.com/blog/service-discovery-in-a-microservices-architecture/. [Erişildi: 2022 Kasım 16].

[13] Al-Masri, E., & Mahmoud, Q. H. (2008). Toward quality-driven web service discovery. *IT Professional*, *10*(3), 24-28.

[14] Mohanty, R., Ravi, V., & Patra, M. R. (2010). Web-services classification using intelligent techniques. *Expert Systems with Applications*, *37*(7), 5484-5490.

[15] Adarme, M., & Jimeno, M. (2021). QoS-Based Pattern Recognition Approach for Web Service Discovery: Ar_WSDS. *Applied Sciences*, *11*(17), 8092.

[16] Swamy, K. (2015, June). Web service classification using multi-Layer perceptron optimized with Tabu search. In *2015 IEEE International Advance Computing Conference (IACC)* (pp. 290-294). IEEE.

[17] Mustafa, A. S., & Kumaraswamy, Y. S. (2014, November). Data mining algorithms for Web-services classification. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)* (pp. 951-956). IEEE.

[18] Das, M. S., Govardhan, A., & Lakshmi, D. V. (2019). Classification of web services using data mining algorithms and improved learning model. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *17*(6), 3191-3202.

[19] Chippa, M., Priyadarshini, A., & Mohanty, R. (2019, April). Application of machine learning techniques to classify web services. In *2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)* (pp. 1-7). IEEE.

[20] Al-Masri, E., & Mahmoud, Q. H. (2009, October). Discovering the best web service: A neural network-based solution. In *2009 IEEE International Conference on Systems, Man and Cybernetics* (pp. 4250-4255). IEEE.

[21] Chandra, M., & Niyogi, R. (2019). Web service selection using modified artificial bee colony algorithm. *IEEE Access*, *7*, 88673-88684

[22] Mhlanga, S. T., Chiyangwa, T. B., Lall, M., & Ojo, S. (2019, December). A hybrid-based architecture for web service selection. In *2019 IEEE International Conference on Engineering, Technology and Education (TALE)* (pp. 1-8). IEEE.

[23] Ş. Ay, «Ensemble Learning — Bagging ve Boosting,» 16 Aralık 2019. [Çevrimiçi]. Available: https://medium.com/deep-learning-turkiye/ensemble-learning-bagging-ve-boosting-50643428b22b. [Erişildi: 2 Aralık 2022].

[24] O. Samur, «Boosting Algoritmaları,» 17 Kasım 2020. [Çevrimiçi]. Available: https://www.datasciencearth.com/boosting-algoritmalari/. [Erişildi: 21 Kasım 2022].

[25] A. Jain, «Complete Machine Learning Guide to Parameter Tuning in Gradient Boosting (GBM) in Python,» 21 Şubat 2016. [Çevrimiçi]. Available: https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/. [Erişildi: 6 Ocak 2023].

[26] A. Jain, «Complete Guide to Parameter Tuning in XGBoost with codes in Python,» 1 Mart 2016. [Çevrimiçi]. Available: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/. [Erişildi: 24 11 2022].

[27] E. R. Muratlar, «LightGBM,» 29 Nisan 2020. [Çevrimiçi]. Available: https://www.veribilimiokulu.com/lightgbm/. [Erişildi: 21 Kasım 2022].

[28] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*.

[29] E. R. Muratlar, «CatBoost Nedir? Diğer Boosting Algoritmalarından Farkı Nelerdir?,» 6 Ağustos 2020. [Çevrimiçi]. Available: https://www.veribilimiokulu.com/catboost-nedir-diger-boosting-algoritmalarindan-farki-nelerdir/. [Erişildi: 21 Kasım 2022].

[30] CatBoost, «Parameter tuning,» [Çevrimiçi]. Available: https://catboost.ai/en/docs/concepts/parameter-tuning. [Erişildi: 24 Kasım 2022].

[31] A. B. Shahul ES, «Hyperparameter Tuning in Python: a Complete Guide,» 14 Ekim 2022. [Çevrimiçi]. Available: https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide. [Erişildi: 23 Kasım 2022].

[32] B. Chen, «A Practical Introduction to Grid Search, Random Search, and Bayes Search,» 6 Eylül 2021. [Çevrimiçi]. Available: https://towardsdatascience.com/a-practical-introduction-to-grid-search-random-search-and-bayes-search-d5580b1d941d. [Erişildi: 30 Kasım 2022].

[33] Scikit-learn developers, «Tuning the hyper-parameters of an estimator,» [Çevrimiçi]. Available: https://scikit-learn.org/stable/modules/grid_search.html. [Erişildi: 30 Kasım 2022].

[34] J. N. community, «Free software, open standards, and web services for interactive computing across all programming languages,» Jupyter, 2014. [Çevrimiçi]. Available: https://jupyter.org/. [Erişildi: 6 Ocak 2023].

[35] Al-Masri, E., & Mahmoud, Q. H. (2007, August). Qos-based discovery and ranking of web services. In *2007 16th international conference on computer communications and networks* (pp. 529-534). IEEE.

[36] S. Narkhede, «Understanding AUC - ROC Curve,» 26 Haziran 2018. [Çevrimiçi]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. [Erişildi: 3 Aralık 2022].

[37] E. Hatipoglu, «Machine Learning — Classification — Naive Bayes — Part 11,» 13 Temmuz 2018. [Çevrimiçi]. Available: https://medium.com/@ekrem.hatipoglu/machine-learning-classification-naive-bayes-part-11-4a10cd3452b4. [Erişildi: 3 Aralık 2022].

[38] E. K. Ulgen, «Makine Öğrenimi Bölüm-2 (k-En Yakın Komşuluk),» 16 Ekim 2017. [Çevrimiçi]. Available: https://medium.com/@k.ulgen90/makine-%C3%B6%C4%9Frenimi-b%C3%B6l%C3%BCm-2-6d6d120a18e1. [Erişildi: 03 Aralık 2022].

[39] Ecemiş, A., Dokuz, A. Ş., & Çelik, M. Çeşitli Veri Kümeleri Üzerinde Veri Madenciliği Algoritmalarının Performansının Değerlendirilmesi, 2019, *International Turkic World Congress on Science and Engineering*, Niğde Tükiye

[40] Ayhan, S., & Erdoğmuş, Ş. (2014). Destek vektör makineleriyle sınıflandırma problemlerinin çözümü için çekirdek fonksiyonu seçimi. *Eskişehir Osmangazi Üniversitesi İktisadi ve İdari Bilimler Dergisi*, *9*(1), 175-201.

EK 1. SLR ÇALIŞMASINDA SEÇİLEN YAYINLAR

[S1] Houmani, Z., Balouek-Thomert, D., Caron, E., & Parashar, M. (2020, May). Enhancing microservices architectures using data-driven service discovery and QoS guarantees. In 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID) (pp. 290-299). IEEE.

[S2] Garba, S., Mohamad, R., & Saadon, N. A. (2022). Self-adaptive mobile web service discovery framework for Dynamic Mobile Environment. Journal of Systems and Software, 184, 111120.

[S3] Guodong, L., Zhang, Q., Ding, Y., & Zhe, W. (2020). Research on service discovery methods based on knowledge graph. IEEE Access, 8, 138934-138943.

[S4] Fang, M., Wang, D., Mi, Z., & Obaidat, M. S. (2018). Web service discovery utilizing logical reasoning and semantic similarity. International Journal of Communication Systems, 31(10), e3561.

[S5] Sun, C., Lv, L., Tian, G., Wang, Q., Zhang, X., & Guo, L. (2020). Leverage label and word embedding for semantic sparse web service discovery. Mathematical Problems in Engineering, 2020.

[S6] Wang, J., Gao, P., Ma, Y., He, K., & Hung, P. C. (2017). A web service discovery approach based on common topic groups extraction. IEEE Access, 5, 10193-10208.

[S7] Huang, Z., & Zhao, W. (2020). Combination of ELMo representation and CNN approaches to enhance service discovery. IEEE Access, 8, 130782-130796.

[S8] Li, C., Zhang, R., Huai, J., Guo, X., & Sun, H. (2013, June). A probabilistic approach for web service discovery. In 2013 IEEE International Conference on Services Computing (pp. 49-56). IEEE.

[S9] Adarme, M., & Jimeno, M. (2021). QoS-Based Pattern Recognition Approach for Web Service Discovery: Ar_WSDS. Applied Sciences, 11(17), 8092.

[S10] Dantas, J. R. V., Lira, H. A., Muniz, B. D. A., Nunes, T. M., & Farias, P. P. M. (2015, October). Semantic web services discovery adopting serin. In 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (pp. 387-394). IEEE.

[S11] Pushpa, C. N., Deepak, G., Kumar, A., Thriveni, J., & Venugopal, K. R. (2020, July). OntoDisco: improving web service discovery by hybridization of ontology focused concept clustering and interface semantics. In 2020 IEEE international conference on electronics, computing and communication technologies (CONECCT) (pp. 1-5). IEEE.

[S12] Madhu, B. G., & Sinchana, J. (2019, July). Mining Underlying Interface Semantics Based Web Services Discovery Approach. In 2019 1st International Conference on Advances in Information Technology (ICAIT) (pp. 453-461). IEEE.

[S13] Li, S., Luo, H., & Zhao, G. (2020, October). bi-hptm: An effective semantic matchmaking model for web service discovery. In 2020 IEEE International Conference on Web Services (ICWS) (pp. 433-440). IEEE.

[S14] Fethallah, H., Ismail, S. M., Mohamed, M., & Zeyneb, T. (2017, December). An outranking model for web service discovery. In 2017 International Conference on Mathematics and Information Technology (ICMIT) (pp. 162-167). IEEE.

[S15] Cheng, B., Li, C., & Chen, J. (2018). Semantics Mining&Indexing-Based Rapid Web Services Discovery Framework. IEEE Transactions on Services Computing, 14(3), 864-875.

[S16] Samir, S., Sarhan, A., & Algergawy, A. (2017, May). Context-based web service discovery framework with qos considerations. In 2017 11th international conference on research challenges in information science (rcis) (pp. 146-155). IEEE.

[S17] Eddine, A. D., M'hamed, B. F., Sifaqes, T. A., & Hamza, Y. O. U. S. F. I. (2017, December). Evaluation tool for contextual similarity measures in Web services discovery approaches. In 2017 International Conference on Mathematics and Information Technology (ICMIT) (pp. 168-176). IEEE.

[S18] Cheng, B., Zhao, S., Li, C., & Chen, J. (2016, June). MISDA: web services discovery approach based on mining interface semantics. In 2016 IEEE International conference on Web services (ICWS) (pp. 332-339). IEEE.

[S19] Naim, H., Aznag, M., Quafafou, M., & Durand, N. (2016, June). Probabilistic approach for diversifying web services discovery and composition. In 2016 IEEE International conference on Web services (ICWS) (pp. 73-80). IEEE.

[S20] Cheng, B., Zhao, S., Li, C., & Chen, J. (2016). A web services discovery approach based on mining underlying interface semantics. IEEE Transactions on Knowledge and Data Engineering, 29(5), 950-962.

[S21] Hammami, R., Bellaaj, H., & Kacem, A. H. (2016, June). A novel approach for semantic web service discovery. In 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (pp. 250-252). IEEE.

[S22] Wu, Y., Yan, C., Ding, Z., Liu, G., Wang, P., Jiang, C., & Zhou, M. (2015). A multilevel index model to expedite web service discovery and composition in large-scale service repositories. IEEE Transactions on Services Computing, 9(3), 330-342.

[S23] Seghir, N. B., Kazar, O., Rezeg, K., & Bourekkache, S. (2017). A semantic web services discovery approach based on a mobile agent using metadata. International Journal of Intelligent Computing and Cybernetics.

[S24] Tian, G., Liu, P., Peng, Y., & Sun, C. (2018). Tagging augmented neural topic model for semantic sparse Web service discovery. Concurrency and Computation: Practice and Experience, 30(16), e4448.

[S25] Duan, L., & Tian, H. (2017). Collaborative web service discovery and recommendation based on social link. Future Internet, 9(4), 63.

[S26] Rodriguez-Mier, P., Pedrinaci, C., Lama, M., & Mucientes, M. (2015). An integrated semantic web service discovery and composition framework. IEEE transactions on services computing, 9(4), 537-550.

[S27] Khan, G., Sengupta, S., Sarkar, A., & Debnath, N. C. (2015, July). Web service discovery in enterprise cloud bus framework: T vector based model. In 2015 IEEE 13th International Conference on Industrial Informatics (INDIN) (pp. 1672-1677). IEEE.

[S28] Moradyan, K., Bushehrian, O., & Akbari, R. (2015, November). A query ontology to facilitate web service discovery. In 2015 2nd International conference on knowledge-based engineering and innovation (KBEI) (pp. 202-206). IEEE.

[S29] Kamath, S., & Ananthanarayana, V. S. (2014, December). Change propagation based incremental data handling in a Web service discovery framework. In 2014 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT) (pp. 000474-000479). IEEE.

[S30] Lo, W., Yin, J., & Wu, Z. (2015, June). Accelerated sparse learning on tag annotation for web service discovery. In 2015 IEEE International Conference on Web Services (pp. 265-272). IEEE.

[S31] Alnahdi, A., Liu, S. H., & Melton, A. (2015, June). Enhanced web service matchmaking: A quality of service approach. In 2015 IEEE World Congress on Services (pp. 341-348). IEEE.

[S32] Alshafaey, M. S., Saleh, A. I., & Alrahamawy, M. F. (2021). A new cloud-based classification methodology (CBCM) for efficient semantic web service discovery. Cluster Computing, 24(3), 2269-2292.

[S33] Baskara, A. R., & Sarno, R. (2017, October). Web service discovery using combined bi-term topic model and WDAG similarity. In 2017 11th International Conference on Information & Communication Technology and System (ICTS) (pp. 235-240). IEEE.

[S34] Omidipoor, M., Toomanian, A., Neysani Samany, N., & Mansourian, A. (2020). Knowledge discovery web service for spatial data infrastructures. ISPRS International Journal of Geo-Information, 10(1), 12.

[S35] Abdelli, A., Serrai, W., & Mokdad, L. (2022). A novel and efficient index based web service discovery approach. Computer Standards & Interfaces, 80, 103586.

[S36] Ye, H., Cao, B., Chen, J., Liu, J., Wen, Y., & Chen, J. (2019, December). A web services classification method based on GCN. In 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom) (pp. 1107-1114). IEEE.

[S37] Ye, H., Cao, B., Peng, Z., Chen, T., Wen, Y., & Liu, J. (2019). Web services classification based on wide & Bi-LSTM model. IEEE Access, 7, 43697-43706.

[S38] Sha, J., Du, Y., & Qi, L. (2019). A user requirement oriented Web service discovery approach based on logic and threshold Petri net. IEEE/CAA Journal of Automatica Sinica, 6(6), 1528-1542.

[S39] Sullivan, A., & Candra, M. Z. C. (2019, November). Web Service Recommendation System using History and Quality of Service. In 2019

International Conference on Data and Software Engineering (ICoDSE) (pp. 1-6). IEEE.

[S40] Tian, G., Zhao, S., Wang, J., Zhao, Z., Liu, J., & Guo, L. (2019). Semantic sparse service discovery using word embedding and Gaussian LDA. IEEE Access, 7, 88231-88242.

[S41] Zeshan, F., Mohamad, R., Ahmad, M. N., Othman, M. B., Elhag, A. A. M., Hussain, S. A., ... & Babar, I. (2019). Context-Aware ontology and web services discovery for distributed embedded real-time systems. Malaysian Journal of Computer Science, 32(3), 186-208.

[S42] Zhang, L. (2014, May). OWL-S based web service discovery in distributed system. In 2014 IEEE Workshop on Electronics, Computer and Applications (pp. 882-885). IEEE.

[S43] Benghida, A., & Boufaida, M. (2014, March). Web services discovery in P2P networks based on clustering. In 2014 1st International Conference on Information and Communication Technologies for Disaster Management (ICT-DM) (pp. 1-7). IEEE.

[S44] Du, Y. Y., & Qin, X. (2014). Multi-strategy web service discovery for smart government. In Applied Mechanics and Materials (Vol. 536, pp. 625-631). Trans Tech Publications Ltd.

[S45] Han, X. Y., & Li, X. M. (2014). A Service Discovery Algorithm Based on Network Topology Management. In Applied Mechanics and Materials (Vol. 513, pp. 1182-1186). Trans Tech Publications Ltd.

[S46] Tang, X., Luo, D., & Huan, J. (2017, August). An Optimized Method for Web Service Discovery Based on Semantics. In ITITS (pp. 1-7).