# ŞİFRELİ AĞ TRAFİĞİNİN İÇERİK AÇISINDAN SINIFLANDIRILMASI

**Ramazan BOZKIR**

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

# ŞİFRELİ AĞ TRAFİĞİNİN İÇERİK AÇISINDAN SINIFLANDIRILMASI

Ramazan BOZKIR
0000-0002-0032-4270

Doç. Dr. Murtaza CİCİOĞLU
(Danışman)

Dr. Cengiz TOĞAY
(İkinci Danışman)

YÜKSEK LİSANS
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2022
**Her Hakkı Saklıdır**

# ÖZET

Yüksek Lisans

ŞİFRELİ AĞ TRAFİĞİNİN İÇERİK AÇISINDAN SINIFLANDIRILMASI

**Ramazan BOZKIR**

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

**Danışman:** Doç. Dr. Murtaza CİCİOĞLU

Günümüzde internet kullanımının yaygınlaşması mevcut bilgisayar ağları altyapısının verimli ve güvenli bir şekilde yönetilmesini gerektirmektedir. Ayrıca, her geçen gün internet uygulamalarının kullanımındaki artış ile büyük veri hacmini oluşturan ağ trafikleri de ortaya çıkmaktadır. Büyük verilerin işlenebilmesi için performans odaklı yöntemlerin kullanılması gerekmektedir. Ağ trafiği verilerinin, ağ yönetimi ve ağ güvenliği gibi birçok çalışma alanındaki uygulamalar için sınıflandırma ihtiyacı bulunmaktadır. Ağ trafiğinin şifreli olması ve VPN kullanımı gibi uygulamalar ağ trafiği sınıflandırma sürecini zorlaştırmaktadır.

Bu tez çalışmasında şifreli ağ trafiğinin sınıflandırılması için gerçek-zamanlı sistemlere kolay ve hızlı uygulanabilir performans-odaklı yeni bir platform geliştirilmiştir. Sınıflandırma sürecinde makine öğrenmesi tekniklerinden yararlanılmıştır. Deney tabanlı makine öğrenmesi tekniklerinin etkili bir şekilde uygulanabilmesi için süreç yönetim gerçekleştirilmiştir. Platformun tasarlanmasında güncel ve performanslı olan veri işleme için Apache Spark, öznitelik çıkarımı için NFStream ve süreç yönetimi için MLflow yazılım teknolojileri kullanılmıştır. Ayrıca, bu çalışma literatüre "pattern byte" isimli yeni bir öznitelik kazandırmıştır. Önerilen platform ile gerçekleştirilen deney kapsamında uygulama ve uygulama türlerine göre ağ trafiği makine öğrenmesi algoritmaları ile sınıflandırılmaktadır. Makine öğrenmesi algoritmalarından GBTree, LightGBM ve XGBoost algoritmalarının kullanılması sonucunda performans sonuçları değerlendirildi. Performans sonuçlarının değerlendirilmesi doğruluk, duyarlılık, kesinlik ve F1 skorları ile incelenmektedir. İncelenen sonuçlarda uygulama sınıflandırmasında GBTree, LightGBM, XGBoost algoritmaları sırasıyla yaklaşık %98, %89 ve %99 F1 skorlarına ulaşmaktadır. Uygulama türlerine göre sınıflandırmada ise tüm algoritmalar %99 F1 skoruna ulaşmaktadır. Sonuç olarak, algoritmalar arasında XGBoost algoritması her iki sınıflandırma probleminde %99'un üzerinde F1 skoru ile en iyi sonuca ulaştığı görülmüştür.

**Anahtar Kelimeler:** Ağ trafiği sınıflandırma, ağ trafiği, makine öğrenmesi, topluluk yöntemleri
**2022, vii + 54 sayfa.**

## ABSTRACT

MSc

CONTENT CLASSIFICATION OF ENCRYPTED NETWORK TRAFFIC

**Ramazan BOZKIR**

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

**Supervisor:** Assoc. Prof. Dr. Murtaza CİCİOĞLU

Today, the widespread use of the Internet requires efficient and secure management of the existing computer network infrastructure. Network traffic as big data emerges as a result of increasing internet applications day by day. Therefore, performance-oriented methods should be used to process big data. Network traffic data needs to be classified for applications in many workspaces, such as network management and security. Network traffic encryption and applications such as VPN increases the complexity of network traffic classification process.

In this thesis, a new performance-oriented platform has been developed for the classification of encrypted network traffic, which can be easily and quickly applied to real-time systems. Machine learning techniques were used in the classification process. Process management was carried out in order to apply experiment-based machine learning techniques effectively. Apache Spark for data processing, NFStream for feature extraction, and MLflow software technologies for process management were used in the design of the platform. In addition, this study has brought a new feature called “pattern byte” to the literature. Within the scope of the experiment carried out with the proposed platform, network traffic is classified by machine learning algorithms according to the application and application types. Performance results were evaluated as a result of using GBTree, LightGBM, and XGBoost algorithms from machine learning algorithms. Evaluation of performance results is examined by accuracy, recall, precision, and F1 scores. In the results examined, GBTree, LightGBM, and XGBoost algorithms achieve F1 scores of approximately 98%, 89%, and 99% in application classification. In classification according to application types, all algorithms reach 99% F1 scores. As a result, among the algorithms, it was seen that the XGBoost algorithm achieved the best result with an F1 score of over 99% in both classification problems.



**Key words:** Network Traffic Classification, Network Traffic, Machine Learning, Ensemble Methods
**2022, vii + 54 pages.**

## ÖNSÖZ ve/veya TEŞEKKÜR

Tez çalışmamız için gereken tüm akademik bilgi ve tecrübelerini samimi bir duygu ile aktaran, karşılaştığım problemler ile de yakından ilgilenen üzerimdeki emeği çok fazla olan değerli ve saygı değer hocam Doç. Dr. Murtaza CİCİOĞLU'na sonsuz teşekkürlerimi sunarım.

Çalışmamızın geliştirilmesine tüm süreçlerde, yazılım sektöründeki bilgi ve tecrübelerini paylaşan ve ilgisini hiçbir zaman esirgemeyen çok değerli ve saygı değer hocam Dr. Cengiz TOĞAY'a teşekkürlerimi sunarım.

Tez çalışmamız için 12 aylık bir süre boyunca "Yurt İçi Öncelikli Alanlar Yüksek Lisans Burs Programı" kapsamında verilen maddi destekten dolayı TÜBİTAK'a teşekkürlerimi sunarım.

Hayatım boyunca maddi ve manevi desteklerini üzerimden hiçbir zaman esirgemeyen aileme; başta annem Zarife BOZKIR ve babam Yusuf BOZKIR olmak üzere, ablam Nur İpek BOZKIR SİPAHİ ve kardeşim Mehmet BOZKIR'a en içten duygularımla sonsuz sevgilerimi ve teşekkürlerimi sunarım.

Ve son olarak, iyi günde kötü günde daima yanımda olan manen kardeş olarak saydığım, çok değer verdiğim ve çokça samimiyet duyduğum ağabeyim Sinan ÇANGA ve kardeşim Rasim KADİRHAN'a teşekkürü bir borç bilirim.

Ramazan BOZKIR
30/05/2022

## İÇİNDEKİLER

**Sayfa**

ÖZET ..... vi
ABSTRACT ..... vii
ÖNSÖZ ve/veya TEŞEKKÜR ..... viii
SİMGELER ve KISALTMALAR DİZİNİ ..... x
ŞEKİLLER DİZİNİ ..... xi
ÇİZELGELER DİZİNİ ..... xii
1. GİRİŞ ..... 1
2. KAYNAK ARAŞTIRMASI ..... 5
2.1. Uygulama Sınıflandırma ..... 6
2.2. Uygulama Türü Sınıflandırma ..... 7
2.3. Zararlı Yazılım Analizi ..... 9
3. MATERYAL ve YÖNTEM ..... 10
3.1. Materyal ..... 10
3.1.1. Veri Kümesi ..... 10
3.1.2. Yazılım Teknolojileri ..... 11
3.1.3. Deney Ortamı ..... 12
3.2. Ağ Trafiği Sınıflandırması ..... 13
3.2.1. Öznitelik çıkarımı ..... 14
3.2.2. Yeni bir öznitelik ..... 15
3.2.3. Veri hazırlama ..... 19
3.2.4. Makine öğrenmesi ..... 20
3.2.5. Makine öğrenmesi değerlendirme metrikleri ..... 26
3.2.6. Hiper-parametre optimizasyonu ..... 28
3.2.7. Süreç yönetimi ..... 30
4. BULGULAR ..... 34
5. TARTIŞMA ve SONUÇ ..... 49
KAYNAKLAR ..... 51
ÖZGEÇMİŞ ..... 55

## SİMGELER ve KISALTMALAR DİZİNİ

| **Simgeler** | **Açıklama** |
|---|---|
| $F$ | Ağ akışı |
| $P$ | Ağ paketi |
| $M$ | Tur sayısı |
| $L$ | Kayıp fonksiyonu |
| $\gamma$ | Tahmin sınıfı |
| $I$ | Alt küme gösterimi |
| $R$ | Bölge |
| $b$ | Tahmin değeri |
| $\nu$ | Öğrenme oranı |

| **Kısaltmalar** | **Açıklama** |
|---|---|
| QoS | Servis kalitesi (Quality of service) |
| QoE | Deneyim kalitesi (Quality of experience) |
| FS | Öznitelik seçimi (Feature selection) |
| CV | Çapraz-doğrulama (Cross-validation) |
| XGBoost | Aşırı Gradyan Artırma (Extreme Gradient Boosting) |
| LightGBM | Hafif Gradyan Artırma Makinesi (Light Gradient Boosting Machine) |
| GBTree | Gradyan Artırma Ağacı (Gradient Boosting Tree) |
| DT | Karar Ağacı (Decision Tree) |
| TLS | Taşıma Katmanı Güvenliği (Transport Layer Security) |
| P2P | Eşler arası (Peer-to-peer) |
| IP | İnternet Protokolü |
| VoIP | IP üzeriden ses verisi aktarımı (Voice over Internet Protocol) |
| Streaming | Gerçek zamanlı veri akışı |
| FT | Dosya transferi |
| VPN | Sanal özel ağ (Virtual private network) |

# ŞEKİLLER DİZİNİ


| | | Sayfa |
|---|---|---|
| Şekil 3.1. | Önerilen sistemin genel yapısı | 13 |
| Şekil 3.2. | NFStream aracının öznitelik hesaplama süreci | 15 |
| Şekil 3.3. | Yeni bir öznitelik olan "pattern_byte" için hesaplama süreci | 18 |
| Şekil 3.4. | Karar ağacının genel yapısı | 22 |
| Şekil 3.5. | Artırma tekniği ile topluluk öğrenme yönteminin genel yapısı | 23 |
| Şekil 3.6. | Karmaşıklık matrisi gösterimi | 27 |
| Şekil 3.7. | Erken durdurma tekniğinin temsili gösterimi | 30 |
| Şekil 3.8. | MLFlow kullanıcı arayüzünde deneylerin listelendiği sayfanın ekran görüntüsü | 31 |
| Şekil 3.9. | MLFlow kullanıcı arayüzünde modellerin listelendiği sayfanın ekran görüntüsü | 32 |
| Şekil 3.10. | MLFlow kullanıcı arayüzünde örnek interaktif görselleştirmelerin ekran görüntüleri | 32 |
| Şekil 4.1. | Görev A'da hedef sınıflar için veri dağılımı (sınıf verisi $< 2500$) | 37 |
| Şekil 4.2. | Görev A'da hedef sınıflar için veri dağılımı (sınıf verisi $\geq 2500$) | 37 |
| Şekil 4.3. | Görev B'de hedef sınıflar için veri dağılımı | 38 |
| Şekil 4.4. | Görev A'da sınıflandırma algoritmalarının değerlendirme skorları A) varsayılan B) öznitelik seçimi C) çapraz-doğrulama D) öznitelik seçimi ve çapraz-doğrulama | 42 |
| Şekil 4.5. | Görev B'de sınıflandırma algoritmalarının değerlendirme skorları A) varsayılan B) öznitelik seçimi C) çapraz-doğrulama D) öznitelik seçimi ve çapraz-doğrulama | 44 |
| Şekil 4.6. | Görev A'da XGBoost algoritmasının öznitelik seçimi ve çapraz-doğrulama durumundaki karmaşıklık matrisi | 47 |
| Şekil 4.7. | Görev B'de XGBoost algoritmasının öznitelik seçimi ve çapraz-doğrulama durumundaki karmaşıklık matrisi | 48 |

# ÇİZELGELER DİZİNİ

**Sayfa**


Çizelge 3.1. "pattern_byte" öznitelik çıkarımına ilişkin sözde kodu ..... 16
Çizelge 3.2. Gradyan artırma algoritmasının sözde kodu (Friedman, 2001)..... 24
Çizelge 3.3. Hiper-parametre optimizasyonunda kullanılan parametreler ve açıklamaları ..... 29
Çizelge 4.1. Öznitelik tablosu..... 35
Çizelge 4.2. Çapraz-doğrulama işleminde algoritmalar için kullanılan parametreler ve değer aralıkları..... 40
Çizelge 4.3. Görev A'nın sınıflandırma algoritmalarında kullanılan parametrelerin, varsayılan ve çapraz-doğrulama sonrasında belirlenen değerleri..... 41
Çizelge 4.4. Görev B'nin sınıflandırma algoritmalarında kullanılan parametrelerin, varsayılan ve çapraz-doğrulama sonrasında belirlenen değerleri..... 43
Çizelge 4.5. Görev A'da sınıflandırma algoritmalarının tüm değerlendirme skorları..... 45
Çizelge 4.6. Görev B'de sınıflandırma algoritmalarının tüm değerlendirme skorları..... 45
Çizelge 4.7. Görev A için sınıflandırma algoritmalarının, öznitelik seçimi ve çapraz-doğrulama durumunda öznitelik önemi en yüksek ilk 10 öznitelik..... 46
Çizelge 4.8. Görev B için sınıflandırma algoritmalarının, öznitelik seçimi ve çapraz-doğrulama durumunda öznitelik önemi en yüksek ilk 10 öznitelik..... 46

# 1. GİRİŞ

Ağ trafiğinin sınıflandırılması ağ yönetimi ve ağ güvenliği konularında oldukça önemlidir. Ağ trafiği sınıflandırması, ağ trafiğinin karakteristik özellikleri ile kullanılan uygulamaların ilişkisini ortaya koymayı amaçlamaktadır. Sınıflandırma sayesinde anormallik tespiti, servis kalitesi (Quality of Service, QoS), deneyim kalitesi (Quality of Experience, QoE), ağ kaynak kullanımı ve paylaşımı gibi konulardaki problemlere çözüm aranmaktadır.

Dijitalleşen dünyada Covid-19 etkisiyle beraber internet kullanımı hızla yaygınlaşmaktadır. Dünya nüfusunun, 2019 yılında %54'ü interneti kullanmaktayken, bu oran 2021 yılında %63'e ulaşmıştır (International Telecommunication Union, 2021). Cisco tarafından yayınlanan istatistiksel verilere göre, dünya genelinde oluşan ağ trafiği hacminde 2016'dan 2021'e kadar %26 büyüme gözlemlenmiştir (Cisco Systems, 2021). Ağ trafiği hacmindeki artış, internet ağlarında gerçekleştirilen operasyonların yönetim ve analiz süreçlerini zorlaştırmaktadır. Ağ trafiği operasyonları; hizmet kalitesinin artırılması, saldırı tespiti (Zeng vd., 2019), zararlı yazılım analizi (Arivudainambi vd., 2019), hata tespiti ve anormallik tespiti (Fernandes vd., 2018) gibi işlemlerden oluşmaktadır. Ağ trafiğinin sınıflandırılması, ağ trafiğini engelleme, önceliklendirme, kaynak yönetimi gibi eylemlerin yürütülmesine olanak sağlamaktadır. Ağ yönetimi ve güvenliği konularındaki çalışmaların başarısı, ağ trafiğini sınıflandırma başarısı ile doğru orantılı olduğu düşünülmektedir. Dolayısıyla ağ trafiği sınıflandırma sürecinin doğru, etkili ve performans odaklı olması önem arz etmektedir. Örneğin; zararlı yazılım analizi yapılan bir sistemde kullanılan zayıf bir ağ trafiği sınıflandırma yaklaşımı büyük zararlara yol açabilmektedir.

Ağ trafiğinin sınıflandırılması için geliştirilen çözüm yöntemleri üç farklı yaklaşım olarak gruplandırılmaktadır. Bunlar bağlantı noktası tabanlı (port-based), yük tabanlı (payload-based) ve makine öğrenmesi tabanlı yaklaşımlardır. 0 ile 1023 arasındaki bağlantı noktası numaraları, yaygın kullanılan TCP/IP uygulamaları için ayrılmış ve iyi bilinen bağlantı noktaları (well-known ports) olarak isimlendirilmektedir. Ağ trafiğinin sınıflandırılması için kullanılan yöntemlerden biri olan bağlantı noktası tabanlı yaklaşım, iyi bilinen

uygulamalar tarafından kullanılan standart bağlantı noktalarının kontrol edilmesine dayanmaktadır. Ancak, günümüzde mevcut uygulamaların dinamik bağlantı noktasına sahip olmalarından dolayı bu yöntem çok etkili değildir (Zhang, J. vd., 2015). Ayrıca, bağlantı noktası tabanlı yaklaşımın başarısını, sanal özel ağ (Virtual Private Network, VPN) oturumları önemli ölçüde azaltmaktadır (Bu vd., 2020).

Yük tabanlı yaklaşım, derin paket incelemesi olarak da adlandırılmaktadır. Derin paket incelemesi, önceden tanımlanan tasarım kalıpları ile ağ paketlerinin yük verilerinde doğru ve minimum gecikme ile analiz etme işlemidir (Finsterbusch vd., 2014). Tasarım kalıplarının güncel ve analiz işleminin yüksek performanslı olması gerekmektedir. Ayrıca, oturum ve uygulama katmanı (OSI modeli katmanları) içeriklerinin doğrudan analiz edilmesi kurumsal gizlilik politikalarının veya ilgili gizlilik mevzuatının ihlali anlamına gelebilmektedir (Nguyen ve Armitage, 2008). Bunun yanında, yük ve bağlantı noktası tabanlı yaklaşımlar ağ trafiğinin şifreli olması durumunda yetersiz kalmaktadır. Farklı ağlar için geliştirilen açık kaynak trafik sınıflandırma modülleri olan OpenDPI, nDPI, IPP2P, HiPPIE, libprotoident ve L7-filter çözümleri yük tabanlı yaklaşımın uygulandığı en uygun yöntemlerdendir (Finsterbusch vd., 2014).

Google, Ocak 2021'de web trafiğinin %95'inin şifreli olarak gerçekleştiği raporunu sunmuştur (Google, 2022). Yük tabanlı ve bağlantı noktası tabanlı yaklaşımların, gelişen ve yaygınlaşan şifreleme teknolojileri (SSL veya TLS) ile sağlanan ağ trafiğinin sınıflandırılmasına uygun olmadıkları düşünülmektedir. Bu nedenle, bu tez çalışmasında, ağ trafiği sınıflandırma problemi için üçüncü yöntem olan makine öğrenimi yaklaşımı önceki yaklaşımların eksikliklerini gidermek amacıyla kullanılmıştır.

Makine öğrenimi tabanlı yaklaşım, istatiksel tabanlı yaklaşım olarak da adlandırılmaktadır. Makine öğrenimi, tecrübelerden yola çıkarak otomatik öğrenme sağlayabilen ve bilgi tabanını iyileştirme ve geliştirme yeteneğine sahip bir yaklaşımdır (Nguyen ve Armitage, 2008). Ağ trafiğinde temel bilgiyi ağ paketleri oluşturmaktadır. Ağ paketi, bilgisayar ağlarında iki cihaz arasındaki alıcı ve göndericiye ait bilgileri ve iletilen bilgiyi içeren veri birimidir. Bilgisayar ağlarında iki cihazın, belirli bir süre içerisinde aldıkları ve gönderdikleri paketlerin bir araya gelmesi bir ağ akışını ifade

etmektedir. Ağ trafiği, ağ akışlarından çıkarılan istatiksel özellikler sayesinde daha kapsamlı tanımlanabilmektedir.

Makine öğrenmesindeki tasarım süreçleri, standart yazılım geliştirme yaşam döngüsü ile sağlıklı bir biçimde sürdürülememektedir (Zaharia vd., 2018). Bu süreçler öngörülemeyen yeni sorunların çözülmesini gerektirmektedir. Bu sorunlara, gereksinimlerin ve uygulanan yöntemin sürekli değişikliğe uğraması ile yazılım geliştirme sürecinin büyük bir kısmını oluşturan deneyler örnek olarak verilebilir. Bu bağlamda uçtan uca makine öğrenmesi yaşam döngüsünün yönetilebilir olması amacıyla açık kaynaklı bir platform olan MLflow (Zaharia vd., 2018), süreç yönetim aracı kullanılmıştır. Bölüm 3.2.6'da çalışmamızdaki MLflow süreç yönetimi ile ilgili detaylara değinilecektir.

Bu tez çalışmasında uçtan uca şifreli ağ trafiğinin sınıflandırılması sürecinin yönetimine imkan sunan güncel, verimli ve etkili bir platform geliştirilmiştir.

Tez çalışmasının ana katkıları ve literatürdeki diğer çalışmalardan farklarını sıralayacak olursak;

- Ağ uygulamalarının sınıflandırılması için Apache Spark (Zaharia vd., 2016), NFStream (Aouini ve Pekar, 2022), MLflow (Zaharia vd., 2018) ve PostgreSQL (The PostgreSQL Global Development Group, 2021) teknolojileri kullanılarak yeni ve tümleşik bir platformun geliştirilmesi,
- Ağ akışlarına ilişkin özniteliklerin çıkarılmasında gerçek dünya ağ analizleri için geliştirilen performans odaklı, esnek ve taşınabilir bir araç olan NFStream teknolojinin kullanılması,
- Büyük veriyi oluşturan ağ trafiklerinde kullanılacak makine öğrenmesi algoritmaları için veri hazırlama sürecinde Apache Spark teknolojisinin kullanılması,
- Deney tabanlı olan makine öğrenmesi süreçlerinin yeniden uygulanabilir ve yönetilebilir olması için açık kaynak bir süreç yönetim aracı olan MLflow yapısının kullanılması,

- Çeşitli makine öğrenmesi algoritmalarının geliştirilen platform üzerinde uygulanabilmesi (implementation) için yeni bir arayüzün tasarlanması ve bu sayede esnek bir mimari geliştirilmesi,
- Uygulamaların ağ trafiğinde akışlara ait desenlerin tespitinin literatürdeki manuel tekniklerin aksine daha otonom hale dönüştürülmesini sağlayan yeni bir özniteliğin geliştirilmesi şeklindedir.

## 2. KAYNAK ARAŞTIRMASI

Ağ trafiği sınıflandırma problemi, değişen ve büyüyen ağ trafiğinin etkisiyle literatürdeki akademik ve bilişim sektörünün güncel bir araştırma konusu olmaya devam etmektedir. Khalife ve diğerleri (2014), ağ trafiği sınıflandırması konusunda detaylı bir literatür taraması yapmışlardır. Yaptıkları çalışma sonucunda makine öğrenmesi tekniklerinin ağ trafiği sınıflandırması için yüksek doğruluk ve hızlı tepki vermesi ile öne çıktığını belirtmişlerdir. Ayrıca, derin paket inceleme yaklaşımlarının, veri gizliliğini ihlal etmesi durumundan dolayı uygun olmadığını tespit etmişlerdir. Bu sebeple, ağ trafiği sınıflandırma problemi için literatürdeki istatiksel tabanlı yaklaşımın (veya makine öğrenmesi tabanlı yaklaşım) uygulandığı çalışmalar incelenmektedir.

Ağ trafiği sınıflandırma problemi için derin öğrenme ve makine öğrenmesi algoritmaları sıklıkla kullanılmaktadır. Literatürde derin öğrenme algoritmalarından Evrişimli Sinir Ağları (Convolutional Neural Network, CNN) algoritması daha çok tercih edilmektedir. CNN algoritması literatürdeki çalışmalarda (Zhou vd., 2017; Tong vd., 2018; Zhang, W. vd., 2019), genellikle 2 boyutlu CNN mimarisi ile oluşturulmaktadır. Bazı çalışmalarda (Wang vd., 2017; Ran vd., 2018) ise 1 ve 3 boyutlu tasarlanan CNN mimarileri de kullanılmaktadır. CNN algoritmasını, performans artışı sağlamak amacıyla kullanan çalışmalar (Lopez-Martin vd. 2017; Lotfollahi vd., 2019; Chen vd. 2020), Yinelemeli Sinir Ağı (Recurrent Neural Network, RNN) ile Otomatik Kodlayıcı (Autoencoder) algoritmaları ve metrik öğrenme (metric learning) yöntemleri ile geliştirilmektedirler. CNN algoritması dışında bazı çalışmalar (Vu vd., 2018; Lim vd., 2019), Uzun-Kısa Süreli Bellek (Long-Short Term Memory, LSTM) derin öğrenme algoritmasını uygulamaktadır.

Literatürde uygulanan makine öğrenmesi algoritmalarının derin öğrenme algoritmalarına karşılık tercih edildiği birçok çalışma bulunmaktadır. Bu tez çalışmasında, ağ trafiği sınıflandırmasında makine öğrenmesi algoritmaları tercih edilmektedir. Bölüm 3.2.4'te detaylandırılan makine öğrenmesi yönteminin tercih edilme sebeplerinden bahsedilmektedir. Literatürdeki makine öğrenmesi algoritmalarının uygulandığı çalışmalar, bu bölümde yer alan alt başlıklarında detaylı incelenmektedir.

## 2.1. Uygulama Sınıflandırma

Ağ trafiği uygulamalarının sınıflandırılması bazı çalışmalarda uygulama tanımlama olarak da adlandırılmaktadır. Bu sınıflandırmanın hedefi, şifreli veya şifresiz gerçekleştirilen ağ trafiği uygulamalarının tespit edilmesidir.

Zhang, J. ve diğerleri (2013), ağ akışı özelliklerinin ve korelasyon tabanlı analiz bilgilerinin sınıflandırma probleminde öznitelik olarak kullanılmasını önermektedir. Korelasyon analizi, bir akış torbası (bag of flows) ile modelleme işlemi sonucunda yapılmaktadır. Çıkarılan tüm öznitelikler, k en yakın komşu (k-Nearest Neighbors, kNN) algoritmasının farklı formlarda oluşturulması ile sınıflandırma işlemi gerçekleştirilmektedir. Sınıflandırma performansları deney kapsamında kullanılan WIDE ve ISP veri setlerinde sırasıyla %90 ve %85 doğruluğa ulaştığı belirtilmektedir. Önerilen yöntemin az sayıda eğitim verisiyle bu performansa ulaşıldığı gösterilmektedir.

Datta ve diğerleri (2015), eşler arası iletişim (peer-to-peer, P2P) gerçekleştiren ağ trafiği uygulamalarının davranışsal özelliklerinin tespit edilerek sınıflandırılmasını önermektedir. Davranışsal özelliklerin analizi ile eşler arası iletişim sağlayan uygulamalardaki tünelleme ve benzer hizmetleri sağlayan uygulamaların sınıflandırılma zorluklarının aşılabileceği ifade edilmektedir. Davranışsal özellikler, uygulamalara ait ağ trafiklerinin ayrıntılı incelenmesi sonucunda her uygulamaya özgün özellikler ile belirlenmektedir. Bu yöntem, yapılan deney kapsamında toplanan Google Hangout uygulamasının ağ paketleri üzerinde uygulanması ile davranışsal özellikler tanımlanmaktadır. Belirlenen davranışsal özellikler ile Hangout, Google Plus, Gmail ve Hangout olmayan ağ paketleri üzerinde Adaptif Artırma (Adaptive Boosting, AdaBoost) ve Karar Ağacı (Decision Tree, DT) algoritmalarında %100 duyarlılık skoruna ulaşıldığı belirtilmektedir.

Zhang J. ve diğerleri (2015), sıfır gün (zero-day) uygulamaları sınıflandırması için Güçlü Trafik Sınıflandırma (Robust Traffic Classification, RTC) olarak adlandırdıkları çerçeveyi önermektedirler. Eğitim veri kümesinde bulunmayan uygulamalara ait ağ trafiği verileri, önerilen çerçevenin test aşamasında gözetimsiz öğrenme teknikleri ile

tespit edilmesi sonucunda RTC çerçevesi güncellenmektedir. RTC sisteminde, akış tabanlı istatiksel özellikler ile gözetimsiz ve gözetimli öğrenme algoritmaları olarak sırasıyla K-ortalama (K-means) ve Rastgele Orman (Random Forest, RF) algoritmaları kullanılmaktadır. Bu şekilde RTC çerçevesi ile sıfır gün uygulamalarının tespit edilebildiği belirtilmektedir.

Yamansavascılar ve diğerleri (2017), akış tabanlı istatiksel özellikler ile ağ trafiğini uygulama türlerine göre sınıflandırmaktadırlar. Çalışmalarında kendi hazırladıkları veri kümesi ile ISCX VPN-nonVPN veri kümesini kullanmaktadırlar. Veri kümeleri üzerinde öznitelik seçimi sonrasında DT, RF, kNN ve Bayes Net makine öğrenmesi algoritmalarının sınıflandırma başarımları incelenmektedir. Deney sonuçlarında kendi veri kümelerinde RF algoritmasının yaklaşık %91 ve VPN-nonVPN veri kümesinde kNN algoritmasının yaklaşık %94 oranlarında doğruluk skorlarına ulaştıkları belirtilmektedir.

### 2.2. Uygulama Türü Sınıflandırma

Ağ trafiği uygulamalarının, ağ trafiği türlerine göre sınıflandırılma problemi uygulama türü sınıflandırma olarak adlandırılmaktadır. Uygulama türlerini VoIP, Mail, Streaming gibi hedefler oluşturmaktadır. Bu bölümde, uygulama türlerine göre sınıflandırma problemi üzerine yapılan literatürdeki çalışmalara yer verilmiştir.

Draper-Gil ve diğerleri (2016), Sanal Özel Ağ (Virtual Private Network, VPN) kullanımında ağ trafiği sınıflandırma performansını incelemektedirler. Sınıflandırma sürecinde, ağ akışı süresindeki değişikliğin kNN ve DT algoritmaları üzerindeki etkileri de analiz edilmektedir. Ağ trafiğinin akış tabanlı sınıflandırılmasında ağ akışı sürelerindeki değişimin etkileri incelenmektedir. Ağ akışları 15, 30, 60 ve 120 saniyelik ağ akışı sürelerinden oluşmaktadır. Sınıflandırma yaklaşımında, ISCXFlowMeter aracı ile çıkarılan akış tabanlı istatiksel özellikler kullanılmaktadır. Yapılan deney kapsamında ağ akışı süresinin 15 saniye olmasıyla, VPN ve VPN olmayan ağ trafikleri sınıflandırılmasında DT ve kNN algoritmalarının en iyi sonuca ulaştığı belirtilmektedir.

Shafiq ve diğerleri (2016), ağ akışı istatiksel özelliklerinin makine öğrenmesi algoritmaları ile sınıflandırma başarımlarını incelemektedirler. Tcpdump aracı ile ağ trafikleri toplanmaktadır. Netmate aracı ile ağ akışlarının istatiksel özellikleri çıkarılmaktadır. Ağ trafikleri çıkarılan öznitelikler ile Destek Vektör Makinesi (Support Vector Machine, SVM), Naif Bayes, Bayes ağı ve DT algoritmaları ile sınıflandırılmaktadır. Sınıflandırma sonucunda DT algoritmasının %78,91 ile en iyi sonuca ulaştığı belirtilmektedir.

Gómez ve diğerleri (2017), ağ trafiği sınıflandırmasına geliştirdikleri Uyarlanmış Karar Ağacı Zinciri (Tailored Decision Tree Chain, T-DTC) algoritmasını önermektedirler. T-DTC algoritması, DT algoritmalarından oluşan topluluk öğrenme yönteminin uygulandığı bir algoritmadır. T-DTC algoritmasında oluşturulan her DT algoritması bir ağ trafiği türünü sınıflandırma görevini üstlenmektedir. DT algoritmaları sıralı olarak karar vermektedir. T-DTC algoritmasının diğer topluluk öğrenme algoritmalarından daha iyi doğruluk sağladığı belirtilmektedir.

Nazari ve diğerleri (2019), ağ akışlarının sınıflandırılması için geliştirilen DPI-tabanlı Akış Sınıflandırıcı (DPI-Based Stream Classifier, DSCA) olarak adlandırılan sistemi önermektedirler. DSCA, öncelikle derin paket inceleme (deep packet inspection, DPI) teknikleri ile ağ trafiği uygulamasını tanımlanmakta daha sonrasında ağ trafiği türleri makine öğrenmesi algoritmaları ile sınıflandırılmaktadır. DSCA sisteminin başarısı VPN-nonVPN ve Tor-nonTor veri kümeleri üzerinde yapılan deney ile gösterilmektedir. Yapılan deneyde veri kümeleri için öznitelik seçimi işlemi sonrasında makine öğrenmesi algoritmalarından Adaptif Rastgele Orman (Adaptive Random Forest, ARF) ve olasılıksal uyarlanabilir pencereleme (probabilistic adaptive windowing, PAW) tekniğiyle birlikte uygulanan kNN algoritmaları ile en iyi sonuçlara ulaştıkları belirtilmektedir. ARF algoritması yaklaşık %97, PAW tekniğiyle birlikte kNN algoritması yaklaşık %87 oranlarında doğruluk skorlarına ulaşıldığı yapılan deney ile gösterilmektedir.

### 2.3. Zararlı Yazılım Analizi

Anderson ve McGrew (2016), zararlı yazılımların ağ trafiklerini sınıflandırabilmek ve modelin performansını arttırabilmek için yeni öznitelikler önermektedirler. Önerilen öznitelikler sıralı dizi içerisinde ağ paketi uzunlukları, Taşıma Katmanı Güvenliği (Transport Layer Security, TLS) bilgileri, Alan Adı Sistemi (Domain Name System, DNS) ve Hiper-Metin Transfer Protokolü (Hyper-Text Transfer Protocol, HTTP) verilerinden oluşmaktadır. Bu özniteliklerin farklı kombinasyonlarda kullanımları ile Lojistik Regresyon (Logistic Regression, LR) algoritmasının sınıflandırma performansları incelenmektedir. Yapılan deney kapsamında önerilen tüm özniteliklerin birlikte kullanıldığı durumda en iyi sonuca ulaşıldığı belirtilmektedir.

# 3. MATERYAL ve YÖNTEM

Bu bölümde, yapılan çalışma ile ilgili kullanılan materyal ve uygulanan yönteme yer verildi. Materyal ve yöntem sırasıyla, Bölüm 3.1'de Materyal ve Bölüm 3.2'de Ağ Trafiği Sınıflandırması başlıkları altında detaylı olarak açıklandı.

## 3.1. Materyal

Önerilen sistemin hazırlanmasında ve yapılan deney kapsamında kullanılan materyaller bu bölümde detaylandırıldı. İlgili detaylara veri kümesi, yazılım teknolojileri ve deney ortamı alt bölümlerinde yer verildi.

### 3.1.1. Veri Kümesi

Bu çalışmada, açık erişime sahip UNB ISCX VPN-nonVPN 2016 veri kümesi kullanıldı (Draper-Gil vd., 2016). Veri kümesi, ağ trafiği verilerini barındıran "pcap" uzantılı dosya formatında sunulmaktadır. Ağ trafiği verilerini, laboratuvar ortamında "Alice" ve "Bob" isminde tanımlanan iki kullanıcı arasındaki ağ iletişiminden toplanan ağ trafikleri oluşturmaktadır. Veri kümesinde, toplamda yaklaşık olarak 27 gigabayt (gigabyte, GB) büyüklüğüne ulaşan ağ trafiği verisi bulunmaktadır.

Veri kümesi, ağ trafiği uygulamaları üzerinden gerçekleştirilen ağ iletişimi ile hazırlanmıştır. Ağ trafiği uygulamalarını YouTube, Hangout, Spotify, Facebook, Gmail gibi popüler uygulamalar oluşturmaktadır. Ağ trafiği verileri, hem Sanal Özel Ağ (Virtual Private Network, VPN) hem de temel ağ yapısı üzerinden oluşturulan ağ trafiklerini içermektedir. Ayrıca ağ trafiği verileri, ağ trafiği türlerine göre gruplanmış olarak veri kümesinde bulunmaktadır. Ağ trafiği türlerini; dosya transferi (File Transfer, FT), mesajlaşma (Chat), internet protokolü üzerinden ses iletimi (Voice over Internet Protocol, VoIP), eşler arası iletişim (Peer-to-Peer, P2P), elektronik posta iletişimi (Mail) ve gerçek zamanlı veri akışı (Streaming) oluşturmaktadır.

Veri kümesinde, VPN ve VPN olmayan ağ trafiği verileri ayrı olarak değerlendirilmesi ile 28 farklı ağ trafiği uygulaması yer almaktadır. Veri kümesi, ağ trafiği türlerine göre gruplandığında 6 farklı ağ trafiği türü içermektedir.

### 3.1.2. Yazılım Teknolojileri

Bu bölümde, yapılan çalışma ile önerilen sistemin gerçekleştirilme süreci için tercih edilen yazılım teknolojileri sunuldu. Yazılım teknolojilerini Python, Apache Spark, NFStream, MLflow ve PostgreSQL oluşturmaktadır.

Python, nesne yönelik programlama yaklaşımına sahip, kolay tasarlanabilir ve öğrenilebilir açık kaynaklı bir programlama dilidir (Python Software Foundation, 2020). Python, bu çalışmada uygulanan makine öğrenmesi algoritmalarını ve daha fazlasını barındırmaktadır. Bunun yanında, önerilen sistemde kullanılan diğer yazılım teknolojilerinin uygulanmasını desteklemektedir. Önerilen sistemin hazırlanmasında, gerekli tüm işlemlerin gerçekleştirilmesine uygunluğu ve avantajlarından dolayı programlama dili olarak Python tercih edildi.

Apache Spark, büyük verilerin performanslı bir şekilde işlenebilmesi için tek veya çoklu makineler üzerinde paralel işlem yeteneği olan açık kaynaklı bir çerçeve sunmaktadır (Zaharia vd., 2016). Apache Spark, yapılan çalışmada ağ trafiği sınıflandırma problemi için büyük bir veri hacmine sahip olan ağ trafiği verilerinin performans odaklı işlenebilmesi amacıyla kullanılmıştır. Apache Spark çerçevesi, yapılan çalışmada Bölüm 3.2.3'te detaylandırılan veri hazırlama sürecinde uygulandı.

Ağ trafiği paketlerinden, ağ akışı verilerinin çıkarımı (flow exporter) ve öznitelik hesaplama işlemleri NFStream aracı ile gerçekleştirilmiştir. NFStream, çevrimiçi veya çevrimdışı ağ trafiği analizlerinde, akış tabanlı öznitelik çıkarmayı kolaylaştıran, performans odaklı ve esnek veri yapısı ile açık kaynaklı bir Python kütüphanesidir (Aouini ve Pekar, 2022). Literatürde ağ akışı verilerinin çıkarımı için geliştirilmiş birçok yazılımı aracı bulunmaktadır. Hofstede ve diğerleri (2014), ağ akışı verilerinin çıkarımında literatürde bulunan yazılım araçlarını raporlamaktadırlar. Ancak, bu araçların

kullanılabilmesi için operasyonel bir ağ ortamı hazırlama ve yönetme gibi karmaşıklığı ve zorlukları bulunmaktadır (Aouini ve Pekar, 2022).

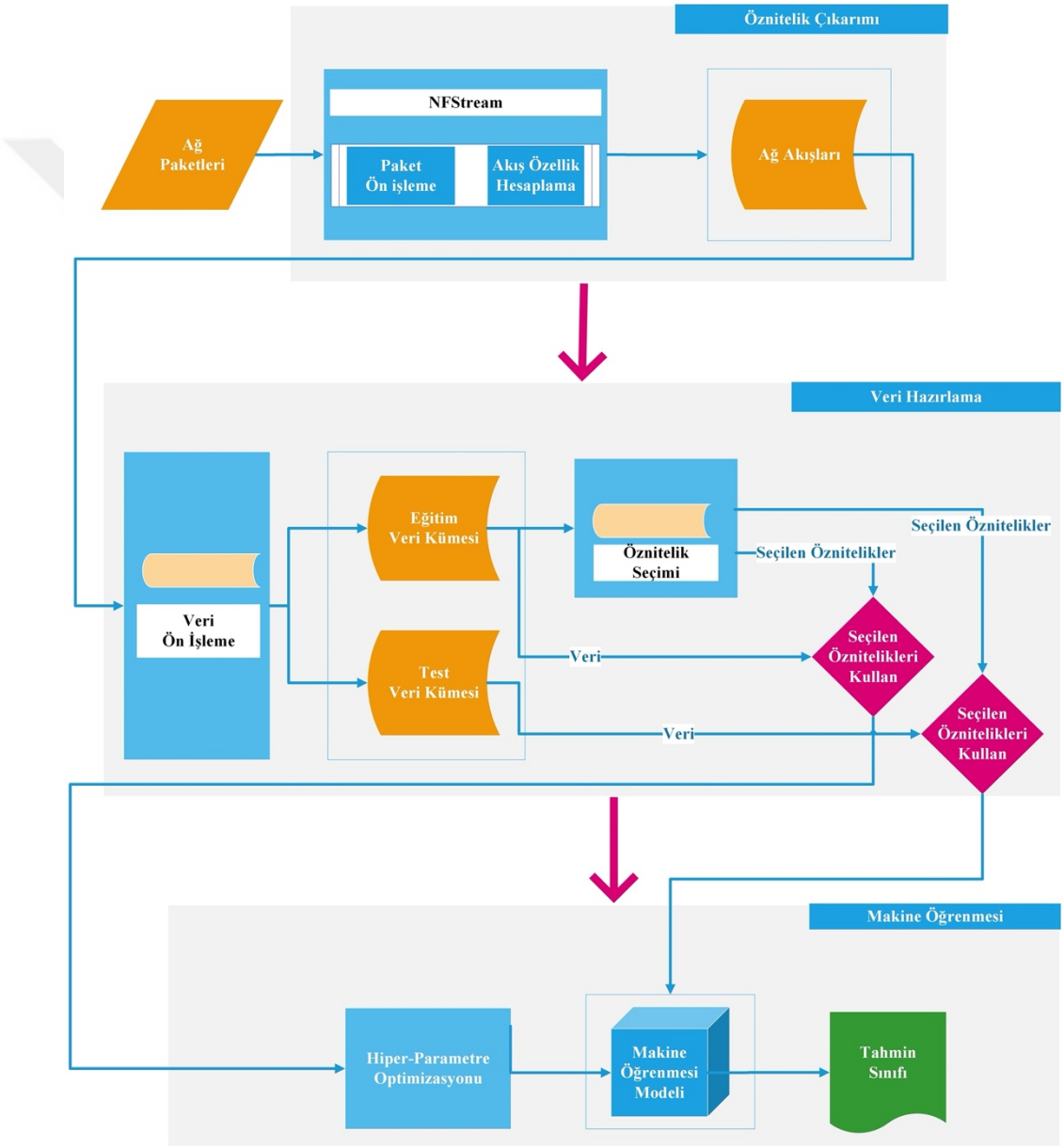
Yazılım geliştirme süreci normalden farklı olarak, deneyimlemenin yoğun olduğu makine öğrenmesi süreçlerinin uygulandığı bu çalışma kapsamında, yazılım geliştirme yaşam döngüsünün yönetimi için MLflow yapısı kullanıldı. MLflow (Zaharia vd., 2018), popüler birçok makine öğrenmesi kütüphanesinin algoritma ve programlama dili ile uyumlu bir şekilde çalışan Uygulama Programlama Arayüzleri (Application Programming Interface, API) sayesinde deneyimlemeyi, yeniden üretilebilirliği ve model dağıtımını kolaylaştıran bir çerçeve sunmaktadır. MLflow, gerçekleştirilen deneylere ilişkin parametrelerin, hesaplanan çıktıların, model bilgilerinin, kullanılan araçların versiyonları gibi bilgilerin depolanabilmesini ve web tabanlı bir arayüz ile süreç yönetimini sağlamaktadır. Deney bilgilerinin depolanmasında MLflow yapısı tarafından desteklenen PostgreSQL veri tabanı kullanılmıştır. PostgreSQL (The PostgreSQL Global Development Group, 2021), güvenirliği kanıtlanmış, veri bütünlüğü ve ölçeklenebilirliği sağlayan açık kaynaklı ilişkisel bir veri tabanıdır.

### 3.1.3. Deney Ortamı

Deneyin gerçekleştirimi, Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz ve 32 GB DDR4 2400 MHz RAM özelliklerine sahip bir sunucu üzerinde yapıldı. Bu çalışmada, kullanılan yazılım teknolojileri doğrultusunda 18.04 sürümüne sahip Ubuntu işletim sistemi tercih edilmiştir. Tercih edilen işletim sistemi ve sürümünün belirlenmesinde en önemli faktör Apache Spark teknolojisi oldu. Yapılan çalışmada veri hazırlama süreçlerinde gerçekleştirilen veri ayıklama, dönüştürme ve yükleme (Extract, Transform and Load, ETL) işlemleri için Apache Spark 3 sürüme ve sonraki sürümleri ile ilk defa Grafik İşlemi Birimi (Graphics Processing Unit, GPU) altyapısı üzerinde gerçekleştirilmektedir. Apache Spark 3 sürümü ilk olarak minimum sistem gereksinimini Ubuntu 18.04 işletim sistemi ve sürümü ile desteklemektedir. Apache Spark 3 ve sonraki sürümlerinde, ETL işlemlerinin Grafik İşlem Birimine sahip bir sistem üzerinde “no code change” (kaynak kod değişikliği olmadan) yaklaşımı ile yapılandırma ayarlarında kullanılacak alt yapı sisteminin değiştirilmesi sonucu sağlanabilmektedir.

### 3.2. Ağ Trafiği Sınıflandırması

Bu tez çalışmasında, ağ trafiğinin uygulama ve uygulama türlerinin sınıflandırma problemleri ele alınmaktadır. Bu problemlerle birlikte diğer ağ trafiği sınıflandırma problemlerine de uygulanabilecek, geliştirilen performans odaklı çözüm yönteminin genel yapısı Şekil 3.1'de gösterilmektedir.



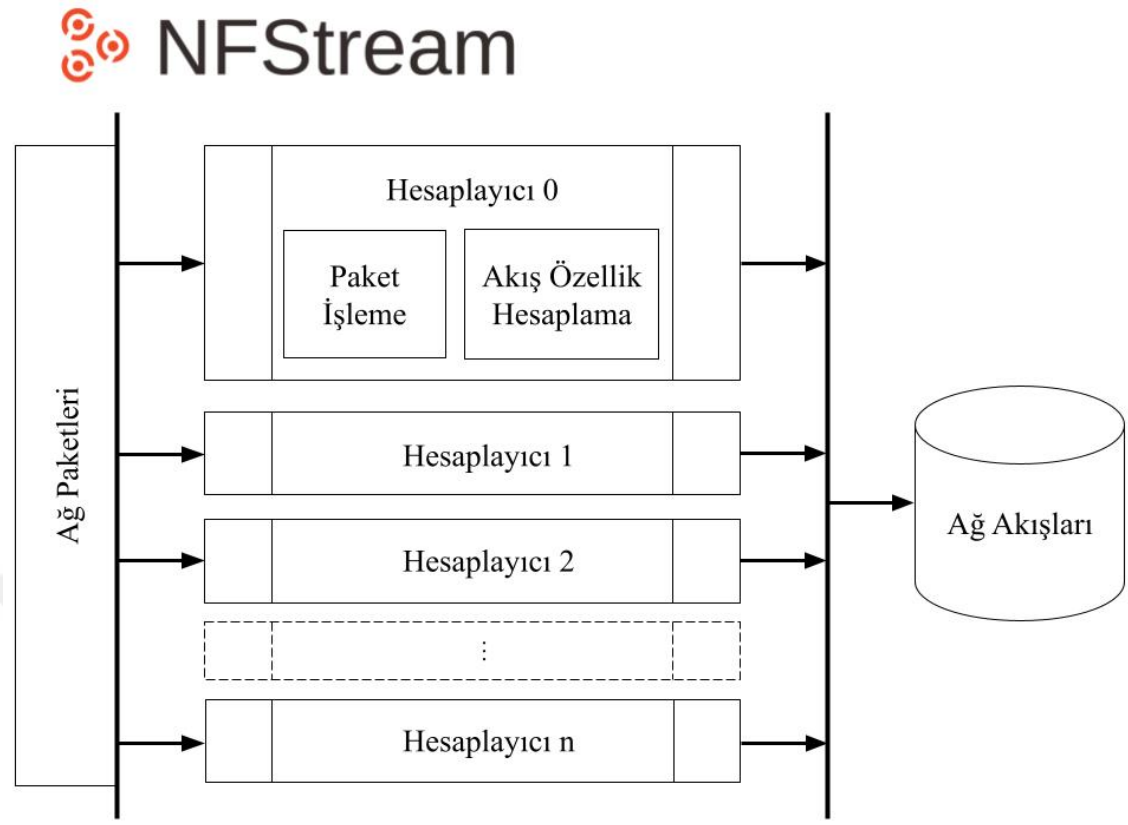

**Şekil 3.1.** Önerilen sistemin genel yapısı

Önerilen sistem üç temel bölüm olan öznitelik çıkarımı, veri hazırlama ve makine öğrenmesi bölümlerinden oluşmaktadır. Öznitelik çıkarımında, ağ trafiği paketlerinin akış tabanlı özniteliklerin çıkarılması ile ağ akışları oluşturulur. Veri hazırlama aşaması, ağ akışı verilerinin makine öğrenmesi algoritmaları tarafından kullanılabilir formlara dönüştürülmesi ve öznitelik seçimi işlemlerini içermektedir. Makine öğrenmesi aşamasında ise makine öğrenmesi algoritmalarının eğitim, hiper-parametrelerinin belirlenme ve değerlendirme süreçleri gerçekleştirilmektedir. Önerilen sistem, bu bölümde bulunan alt başlık içerisinde detaylandırılmaktadır.

### 3.2.1. Öznitelik çıkarımı

Veri kümesinde ağ paketlerini içeren "pcap" uzantılı dosyaların NFStream aracı ile işlenmesi sonucu ağ akışı verileri oluşturulmuştur. NFStream aracı ağ akışı verilerini oluşturulmasında öznitelik hesaplamalarına ilişkin hesaplama süreçleri Şekil 3.2'de gösterilmektedir. NFStream, ağ paketlerini paralel işlem yeteneğine sahip hesaplayıcılar (meter) ile işlemektedir. Hesaplayıcılar, paket işleme ve akış özelliklerine ilişkin hesaplamaların yapıldığı birimdir. Her bir hesaplayıcı, belirtilen akış süresinde ağ paketlerini işlemesi sonucunda bir ağ akışı verisini oluşturmaktadır.

Ağ akışı verilerinin oluşturulmasında, hesaplanan öznitelikler ağ paketlerinin sayısı ile doğrudan ilişkilidir. Çünkü hesaplanılan öznitelikler, ağ paketlerindeki ilgili bölümleri istatiksel olarak ifade etmektedir. Örneğin; "src2dst_packets" özniteliği ağ akışı içerisinde kaynaktan hedefe gönderilen paket sayısını ifade eder. Burada ağ akış süresindeki değişime göre özniteliklerin değerleri doğrudan etkilenmektedir. Özniteliklerdeki değişiklik ile makine öğrenmesi algoritmalarının başarımları da değişmektedir. Bu sebeple ağ akışı süresi, UNB ISCX VPN-nonVPN veri kümesinde gerçekleştirilen akış süresinin makine öğrenmesi tekniklerine olan etkisinin analizi sonuçlarında, 15 saniye olarak belirlenmesinin uygun olduğu görülmektedir (Bozkır vd., 2022; Draper-Gil vd., 2016).




**Şekil 3.2.** NFStream aracının öznitelik hesaplama süreci

### 3.2.2. Yeni bir öznitelik

Yapılan çalışmada, ağ trafiğinin akış tabanlı sınıflandırması için NFStream alt yapısında öznitelik çıkarımı (feature extraction) tekniği ile yeni bir öznitelik geliştirilmiştir. Şifreli ağ trafiği için, yük verileri içerisinde tespit edilen bayt kalıpları ile bir öznitelik tanımlanabilir (Dimou vd., 2020). Yapılan bu öneride geliştirilen özniteliğin uygulanabilirliğini destekler niteliktedir. Geliştirilen yeni özniteliğin, sınıflandırma sürecinde uygulanan makine öğrenmesi tekniklerinin performansını iyileştirmeye destek olacağı düşünülmüştür. Yeni öznitelik "pattern_byte" olarak isimlendirilmiştir.

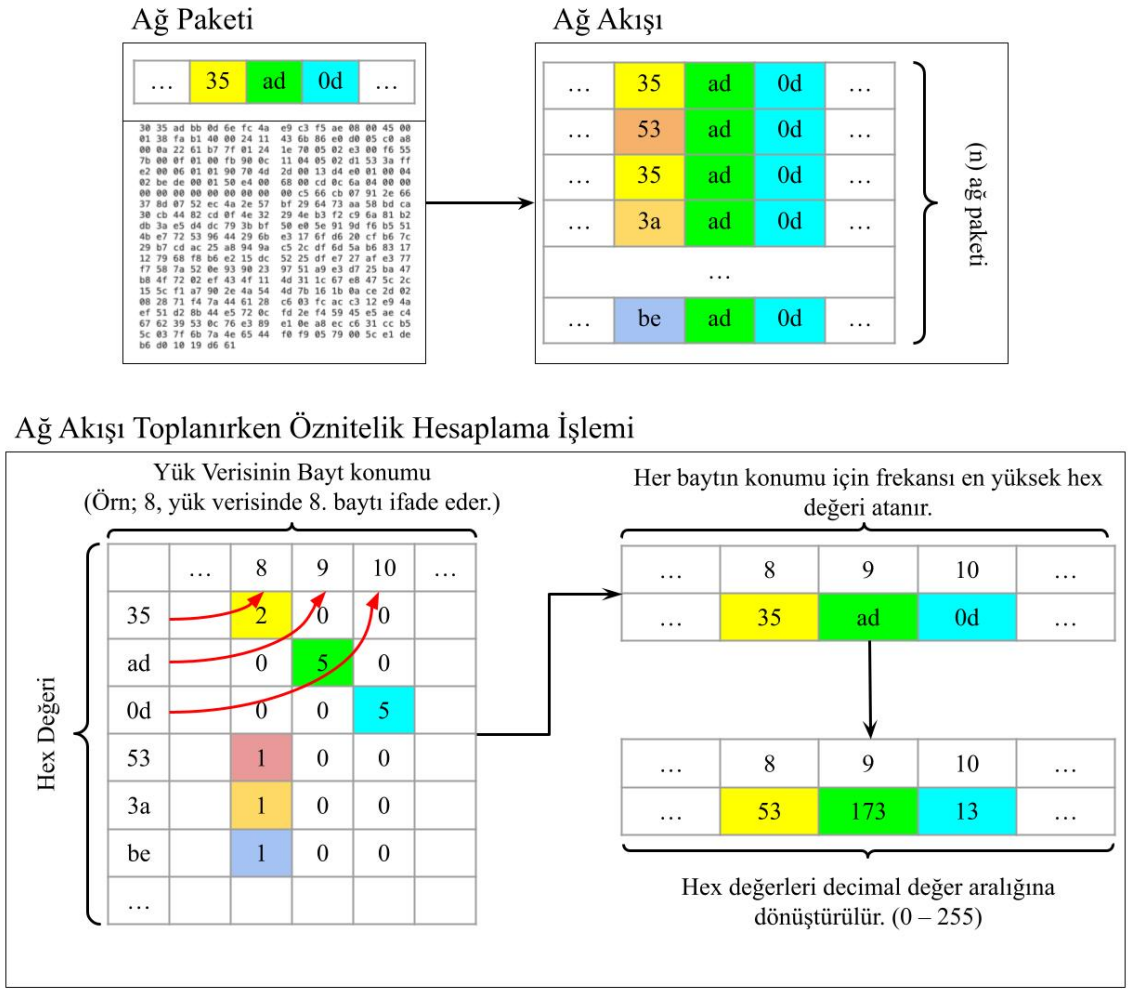
Öznitelik çıkarımı yapılan "pattern_byte" özniteliğinin sözde kodu Çizelge 3.1'de gösterildi. Bu öznitelik için algoritmanın gerçekleştiriminde, ağ akışı ve tarama bölgesinin başlangıç ve bitiş konumları girdi olarak alınmaktadır. Algoritmanın ilk adımında $T$ ve $A$ değişkenleri oluşturulur. $T$ değişkeni yük verisinde tarama aralığının uzunluğunu, tarama bölgesinin bitiş ile başlangıç konumlarının farkı ile ifade etmektedir.

$A$ değişkeni, 256 satır ve $T$ sütundan oluşan bir sıfır matrisidir. İkinci adımda, ağ akışı içerisindeki her bir ağ paketinin ilgili bölümleri matrise eklenmektedir. Burada ağ akışı $F$, ağ paketi $P$ ve ağ paketinin yük verisi $P_{yük\ verisi}$ olarak ifade edilmektedir. Bu adımda bir ağ akışı içerisindeki her bir ağ paketinin ilgili bayt konumları için $A$ matrisine karşılık gelen satır ve sütundaki değer 1 sayı artırılmaktadır. Bu ekleme işleminden önce, ağ paketi yük verisinin ilgili konumunda olan on altılık sayı sistemindeki değer, onluk sayı sistemine dönüştürülmesiyle bir $x$ değişkenine atanır. $A$ matrisinde $x$ değişkeni satırı, $i$ değişkeni ise sütunu ifade etmektedir. Algoritmanın üçüncü adımında ise tamamlanan $A$ matrisindeki ilgili sütundaki maksimum değerin indeksi (yani satırı), bir $Y$ dizisine eklenir. Algoritma sonucunda çıktı olarak elde edilen $Y$ dizisi, ilgili ağ akışından ağ paketi yük verilerindeki taranan bölge için en çok tekrarlanan baytları ifade etmektedir. Yani ağ akışları için ilgili bayt konumlarındaki frekansı en yüksek olan bayt değerleri tespit edilmektedir.

**Çizelge 3.1.** "pattern_byte" öznitelik çıkarımına ilişkin sözde kodu

| | |
|---|---|
| Girdi: | Ağ akışı ($\boldsymbol{F}$), Tarama aralığı $\{\boldsymbol{R_{başlangıç}}, \boldsymbol{R_{bitiş}}\}$. |
| Algoritma: | 1. Bayt konumlarını tanımlayan $T$ boyutunda boş bir dizi ve bir sıfır matrisi oluşturulur.<br>$T = R_{bitiş} - R_{başlangıç}$<br>$A = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}_{256\ x\ T}$<br>2. $F$ içerisindeki her $P$ için:<br>$i = 0$'dan $T$'ye kadar:<br>$x = onluk_sayı_sistemi(P_{yük\ verisi}[R_{başlangıç} + i])$<br>$A_{x,i} = A_{x,i} + 1$<br>3. $A$ matrisinin her sütunundaki en yüksek değerin indeks değeri (satır numarasını) $Y$ dizisine eklenir.<br>$i = 0$'dan $T$'ye kadar:<br>$Y[i] = indeks(maksimum(A_{0\ldots255,i}))$ |
| Çıktı: | Y dizisi |

Geliştirilen “pattern_byte” özniteliğinin sözde koduna ek olarak hesaplama süreci Şekil 3.3’te gösterilen adımlar ile görsel örneği verildi. Hesaplama sürecinde yük verilerinin, belirlenen konumlarındaki onaltılık sayı sisteminde (hex) ifade edilen değerleri alınır. Hex değerleri iki boyutlu bir matrise aktarılır. Matris içerisinde sütunları yük verisinin bayt konumları, satırların indeksini ise hex değerleri ifade eder. Matrise her bir ağ paketinin, yük verisi için bayt konumlarının aldığı hex değerleri eklenir. Örneğin; yük verisinin 8. bayt konumundaki “3a” hex değeri matrise eklendiğinde, matrisin 8. sütunun “3a” indeksli satırında yer alan değeri bir puan artırılır. Ağ akışı tamamlandığında, ağ akışı içerisindeki tüm ağ paketleri matrise eklenmiş olur. Tamamlanan ağ akışı için oluşan matris içerisinde her bir sütun için frekansı en yüksek hex değerleri, yük verisindeki ilgili bayt konumlarının değerleri olarak belirlenir. Örneğin; matrisin 8. sütunda en yüksek puanın alındığı “35” indeks değeri (yani, hex değeri), 8. sütunu ifade eden yük verisinin 8. bayt konumu için en yüksek frekansa sahip hex değeri “35” olarak belirlenir. Yük verisindeki bayt konumları için belirlenen en yüksek frekansa sahip hex değerleri onluk sayı (decimal) sistemindeki değer aralığına dönüştürülmesi ile “pattern_byte” özniteliklerinin hesaplama işlemleri tamamlanır.




**Şekil 3.3.** Yeni bir öznitelik olan “pattern_byte” için hesaplama süreci

Yeni öznitelik hesaplama işlemleri, NFStream aracının esnek yapısı içerisinde hesaplayıcı biriminde tanımlandı. Hesaplayıcı ile ağ paketlerini ağ akışı verilerine dönüştürme aşamasında öznitelik hesaplamaları yapılmaktadır. Yeni tanımlanan “pattern_byte” özniteliği de hesaplayıcı içerisinde performanslı bir şekilde hesaplanmaktadır. Hesaplayıcı birimi içerisinde, ağ akışını oluşturan her bir ağ paketi için yük verisi (payload data) bilgileri ile “pattern_byte” hesaplandı. Özniteliğin hesaplanmasında, yük verisinin istenilen bayt aralığı için hesaplama işlemi yapan bir yapı tanımlandı. Örneğin; yük verisinin ilk 16 baytlık bir bölümde bulunan desenlerin çıkarılması isteniyor ise, başlangıç bayt konumu 0 bitiş bayt konumu 16 olarak belirlenir. Yük verisinde belirlenen bayt konumlarında “pattern_byte” özniteliği hesaplandığında,

her bir bayt konumu için bir “pattern_byte” özniteliği elde edilir. Örneğin; yük verisinin 0. bayt konumu için “pattern_byte_0”, 8. bayt konumu için “pattern_byte_8” gibi isimlendirilen öznitelikler çıkarılır.

### 3.2.3. Veri hazırlama

Veri Hazırlama sürecinde, oluşturulan ağ akışı verileri makine öğrenmesi algoritmaları tarafından işlenebilir hale getirilmektedir. Bu süreç Apache Spark çerçevesinde gerçekleştirilmiştir. Apache Spark ile ağ akışı verileri, makine öğrenmesi sürecinde kullanıma uygun hale getirildi. Bu süreçte, veri ön işleme ve öznitelik seçimi teknikleri uygulandı. Apache Spark yapısında hazırlanan tüneller (pipeline) ile bu teknikler performanslı ve yeniden kullanılabilirliği sağlamaktadır. Hazırlanan tünellerde tanımlanan işlemler sırasıyla gerçekleştirilmektedir. Apache Spark ile hazırlanan veri ön işleme ve öznitelik seçimi tünelleri, yeniden uygulanabilirliği sağlanması amacıyla, MLFlow bünyesinde model olarak depolanmaktadır.

Veri ön işleme tünelinde, veri temizleme ve veri dönüştürme işlemleri uygulanmıştır. Veri temizleme işlemi ile, sınıflandırma sürecinde verinin anlam ifade etmeyen öznitelikleri (örneğin; id, ip adresi, mac adresi, gibi.) ağ akışı verilerinden kaldırılmıştır. Veri dönüştürme ile öznitelik değerleri metin olarak tanımlanan ifadeler, frekansı en yüksek kategoriye sahip metinlerin birer sayı ile indekslenmesi sonucunda sayısal ifadelere dönüştürülmüştür. Örneğin; hedef sınıfı ifade eden öznitelik değerleri olan “Facebook” ve “Hangout” gibi metin ifadeleri 0 ve 1 ile indekslenmesi sonucunda sayısal olarak ifade edildi. Sonuç olarak, veri ön işleme tünelinin girişi bölümünü ağ akışı verilerinin tüm öznitelikleri, çıkış bölümünü ise makine öğrenmesi algoritmaları tarafından işlenebilir değerleri ifade eden öznitelikler ve hedef sınıf etiketleri barındıran veri kümesi oluşturur. Veri kümesi, veri ön işleme tünelinden geçirilmesi sonucunda elde edilen veriler makine öğrenmesi sürecinde kullanılmak üzere %70 oranında eğitim ve %30 oranında test veri kümeleri olarak ayrılır. Eğitim ve test veri kümeleri Apache Spark yapısında önerilen “Parquet” dosya formatında depolandı.

Öznitelik seçimi tünelinde, veri ön işleme tünelinin sonucunda eğitim olarak ayrılan veri kümesinde öznitelik seçimi yapıldı. Öznitelik seçimi ile veri kümesinde veriyi tanımlayan öznitelikler arasından hedef sınıflar için ayırt edici olan öznitelikler seçilmektedir. Öznitelik seçiminin, makine öğrenmesi eğitim yükünü azaltması ve model doğruluğunu artırması gibi avantajları bulunmaktadır (Shen vd., 2020; Shi vd., 2017). Öznitelik seçimi tünelinde, kategorik ve sürekli öznitelikler için farklı teknikler kullanıldı. Kategorik öznitelikler, öznitelik değeri belirli bir durumu, kategoriyi veya indeksi ifade eden sabit değerler ile tanımlanır. Örneğin; veri kümesinde "protocol" özniteliğinin aldığı 6 değeri "TCP", 17 değeri "UDP" gibi kategorileri ifade eden öznitelikler kategorik özniteliklerdir. Sürekli öznitelikler ise özniteliğin aldığı değerlerin değişkenlik gösterdiği bir sınır veya kategori ile ifade edilmeyen öznitelikler olarak tanımlanır. Örneğin; veri kümesinde "src2dst_bytes" özniteliği ağ akışı içerisinde kaynaktan hedefe gönderilen ağ paketlerinin toplam bayt değerini ifade etmesi gibi aldığı değerler değişken olmasından dolayı sürekli bir özniteliktir. Veri kümesinde, hedef sınıf (Örneğin; uygulama adı veya uygulama türü) kategorik olarak ifade edilmektedir. Bu sebeple, veri kümesinde öznitelik seçiminde, kategorik öznitelikler için Ki-Kare (Chi-Square), sürekli öznitelikler için ise Varyans Analizi (Analysis of Variance, ANOVA) teknikleri kullanıldı.

Ki-Kare, kategorik özniteliklerin frekans dağılımlarının hedef sınıflar ile ilişkisinin çıkarılmasıyla hesaplanır. Varyans Analizi, her bir sürekli özniteliğin varyans değerleri ile hedef sınıfların arasındaki değişkenliğin sınıf içi değişkenliğe oranları ile hesaplanmaktadır. Öznitelik seçiminde kullanılan tekniklerin sonucunda öznitelikler ayırt edicilik düzeylerine göre sıralanır. Hem Varyans Analizi hem de Ki-Kare testi sonucundaki sıralanmış özniteliklerin %50'si öznitelik olarak seçildi.

Veri hazırlama sürecinde, oluşturulan veri ön işleme ve öznitelik seçimi tünelleri, MLFlow bünyesinde ilgili yöntemlerin modelleri olarak belirlendi.

### 3.2.4. Makine öğrenmesi

Ağ trafiği sınıflandırmasında, derin öğrenme (deep learning, DL) ve makine öğrenmesi (machine learning, ML) teknikleri uygulanabilmektedir. Derin Öğrenme algoritmaları,

makine öğrenmesi algoritmalarına kıyasla hesaplama maliyetleri oldukça yüksek olması sebebiyle çok güçlü donanım kaynaklarına ihtiyaç duyarlar (Arfeen vd., 2022). Bununla beraber, derin öğrenme algoritmaları için hiper-parametre optimizasyonunun yapılabilmesini, hesaplama maliyetinin yüksek olması sebebiyle zorlaştırmaktadır. Halbuki ağ trafiği sınıflandırmasında, veri kümesinin sürekli olarak güncellenme ihtiyacı bulunmaktadır. Çünkü sınıflandırılan uygulamaların değişen ağ trafiği yapılarına uyarlanması gerekmektedir. Bu sebeple, tercih edilecek sınıflandırma yönteminin, hızlı ve etkili bir şekilde değişiklere uyum sağlaması gerekir. Bu ihtiyaçların karşılanabilmesi için derin öğrenme algoritmaları yerine makine öğrenmesi algoritmaları tercih edildi.

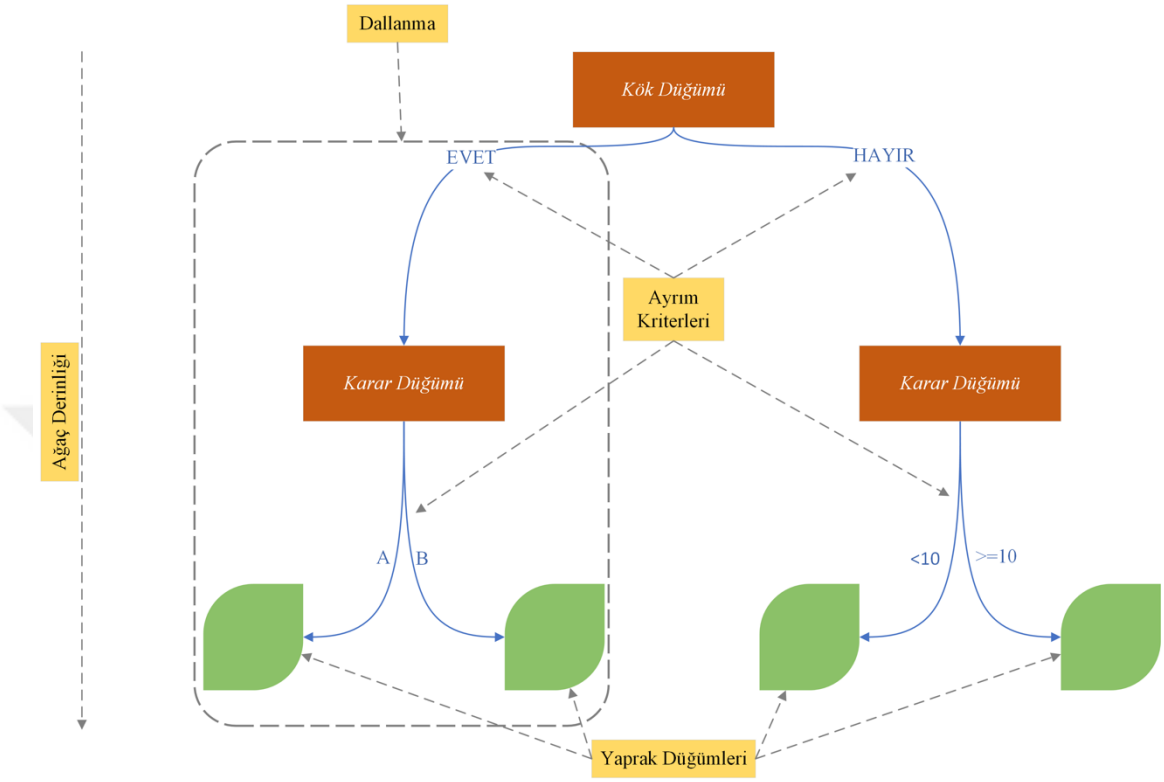
Bu aşamaya kadar gerçekleştirilen işlemler, ağ trafiği verilerini barındıran veri kümesinin, makine öğrenmesi algoritmaları tarafından uygun ve etkili bir şekilde işlenebilir hale getirilmesini kapsamaktadır. Hazırlanan eğitim ve test veri kümeleri, makine öğrenmesi algoritmalarının eğitim ve test aşamalarında kullanıldı.

Bu çalışmada, makine öğrenmesinde topluluk öğrenme algoritmalarından Artırma (Boosting) tekniğinin uygulandığı algoritmalar ile ağ trafiği sınıflandırıldı. Sınıflandırma algoritmaları olarak Gradyan Artırma Ağacı (Gradient Boosting Tree, GBTree), Hafif Gradyan Artırma Makinesi (Light Gradient Boosting Machine, LightGBM) ve Aşırı Gradyan Artırma (Extreme Gradient Boosting, XGBoost) algoritmaları tercih edildi.

GBTree algoritması, temelde zayıf karar ağacı algoritmalarından oluşan topluluk öğrenmesi yöntemini uygulamaktadır. Topluluk öğrenmesini oluşturan birden fazla karar ağaçları bulunmaktadır. Topluluk öğrenmesi, GBTree algoritmasında artırma (boosting) tekniği ile uygulanmaktadır. GBTree algoritması, zayıf karar ağaçlarında hesaplanan kayıp fonksiyonundaki (loss function) hata değeri ile yeni karar ağaçlarını oluşturmaktadır.

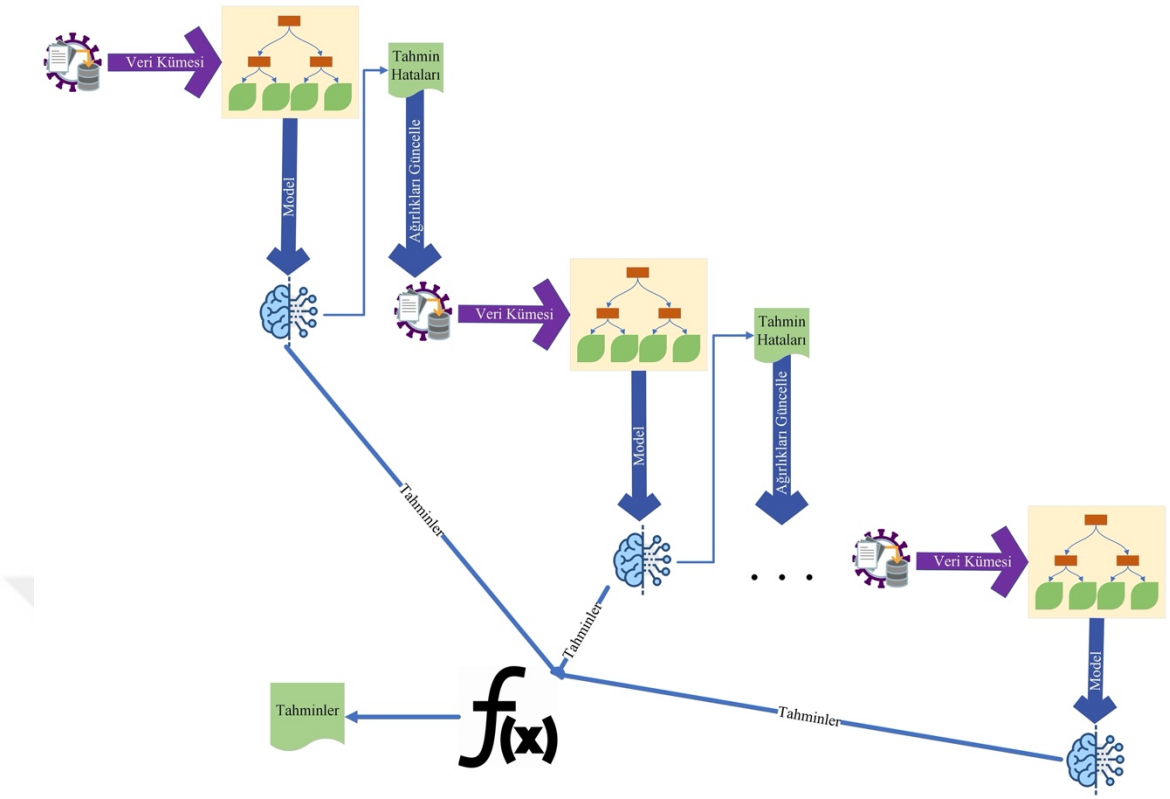
Karar ağaçları kök, karar ve yaprak düğümlerinden oluşmaktadır. Kök ve karar düğümleri sırasıyla veriyi tanımlayan özniteliklerden ve yaprak düğümler ise tahmin sınıflarından oluşmaktadır. Kök düğüm karar ağacının başlangıç düğümünü tanımlamaktadır. Düğümlerde öznitelikler için belirlenen kriterlerde ayrım sağlanır. Ayrım sonucunda,

yeni bir düğüm ile ağacın derinliği artabilir veya bir yaprak düğümü ile sonlanabilir. Karar ağacının genel yapısı Şekil 3.4'te gösterilmiştir.




**Şekil 3.4.** Karar ağacının genel yapısı

Artırma tekniği, topluluk öğrenmesini oluşturan karar ağaçlarının sıralı olarak eğitilmesini kapsamaktadır. Artırma tekniğinin uygulandığı topluluk öğrenmesinin yapısı Şekil 3.5'te gösterildi. Artırma tekniğinde, ilk oluşturulan karar ağacı için veriler eşit ağırlıklı olarak ele alınır. Daha sonra verilerin ağırlıkları, gerçek ve tahmin değerleri ile hataların hesaplanması sonucunda güncellenir. Verilerin güncel ağırlıkları ile yeni bir karar ağacı oluşturulur. Bu şekilde belirtilen karar ağacı sayısı kadar, güncellenen veri ağırlıkları ile yeni karar ağaçları oluşturularak devam eder. Tüm karar ağaçları tamamlandıktan sonra yapılan tahminlerin ağırlıklı ortalamaları alınması ile tahmin sonucu belirlenir.




**Şekil 3.5.** Artırma tekniği ile topluluk öğrenme yönteminin genel yapısı

Gradyan artırma, temelde artırma tekniğinin uygulandığı bir algoritmadır. Gradyan artırma algoritmasının genel yapısına ilişkin sözde kodu Çizelge 3.2'de gösterildi. Algoritmanın girdisini eğitim veri kümesi, tur sayısı ($M$) olarak belirtilen karar ağacı sayısı ve probleme uygun tanımlanan kayıp fonksiyonu ($L$) oluşturulmaktadır. Algoritma ilk olarak zayıf bir karar ağacı oluşturur. İlk karar ağacında tahmin sınıfı ($\gamma$) ile kayıp fonksiyonu hesaplanmaktadır. Bu aşamadaki işlem içerisinde gradyan azaltma (gradient descent) algoritması ile kayıp fonksiyonunun türevlerinin alınması sonucunda global minimumları bulunmaktadır. İkinci aşamada karar ağaçlarının her birinde uygulanan adımlar yer almaktadır. Birinci adımda, karar ağacında her bir veri için tahmin değerleri, gerçek değer ve bir önceki karar ağacının tahmin değerleriyle birlikte hesaplanmaktadır. Burada, kayıp fonksiyonu içerisinde kullanılan $F(x)$ fonksiyonunda bir önceki karar ağacının tahminlerinin kullanılması $F_{m-1}(x)$ ile ifade edilmektedir. İkinci adım, yeni karar ağacının tahmin hatalarına göre oluşturulmasıdır. Bu adımda, $j$ yaprak sayısını, $I_{R_{jm}}$ karar ağacının yaprak düğümünde bulunan bölgenin (yani $R_{jm}$) alt kümesini ve $b_{jm}$ değeri de ilgili bölgedeki yaprak düğümünde bulunan tahmin değerini ifade etmektedir. Bu adım

sonucunda, $h_m(x)$ ile karar ağacındaki tahminler hesaplanır. Üçüncü adımda, karar ağacının tahmin sınıfları için $\gamma_m$, karar ağacındaki tahminlerin kayıp fonksiyonunun hesaplanması sonucunda global minimumun bulunması ile hesaplanır. Dördüncü adımda ise öğrenme modeli, bulunan tahminler $\upsilon$ olarak belirtilen öğrenme oranı (learning rate) ile güncellenir. Algoritmanın sonucunda eğitilen bir model elde edilmektedir.

**Çizelge 3.2.** Gradyan artırma algoritmasının sözde kodu (Friedman, 2001)

| | |
|---|---|
| Girdi: | Eğitim kümesi $\{(\boldsymbol{x_i}, \boldsymbol{y_i})\}_{\boldsymbol{i=1}}^{\boldsymbol{n}}$ , Tur sayısı $\boldsymbol{M}$ , Kayıp fonksiyonu $(\boldsymbol{L}(\boldsymbol{y}, \boldsymbol{F}(\boldsymbol{x}))$. |
| Algoritma: | 1. Başlangıçta $f_0$ hesaplanır.<br>$$f_0 = arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$<br>2. $m = 1$' den $M$'e kadar:<br>a. Hatalar hesaplanır.<br>$i = 1$'den $n$'e kadar:<br>$$\gamma_{im} = -\left[\frac{\partial L\left(y_i, F(x_i)\right)}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$<br>b. Eğitilen bir sınıflandırıcının eğitim kümesi $\{(x_i, \gamma_{im})\}_{i=1}^{n}$ şeklinde kullanılması ile $h_m(x)$ hesaplanır.<br>$$h_m(x) = \sum_{j=1}^{J_m} b_{jm} \mathrm{I}_{R_{jm}}(x)$$<br>c. $\gamma_m$ hesaplanır.<br>$$\gamma_m = arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$<br>d. Model güncellenir.<br>$$F_m(x) = F_{m-1}(x) + \nu * \gamma_m h_m(x), \quad 0 < \nu \leq 1$$ |
| Çıktı: | Eğitilen model $F_M(x)$ |

Kayıp fonksiyonu, sınıflandırma ve regresyon problemlerine göre farklılık göstermektedir. Genellikle kayıp fonksiyonları, regresyon problemleri için Denklem 3.1, sınıflandırma problemleri için ise Denklem 3.2'de bulunan formüller kullanılmaktadır. Denklem 3.2'deki p değeri tahmin olasılığını belirtir. Kayıp fonksiyonundaki amaç,

formüller ile de belirtildiği üzere tahmin değerinin gerçek değerden farkının ifade edilmesidir.

$$L\big(y, F(x)\big) = \frac{1}{n}\sum_{i=1}^{n}(y_i - F(x_i))^2 \tag{3.1}$$

$$L\big(y, F(x)\big) = -[\sum_{i=1}^{n}(y_i \log(p) + (1 - y_i)\log(1 - p))] \tag{3.2}$$

GBTree algoritması, eğitim süresinin uzun olması, öğrenme modelinin yorumlanabilir olmaması ve öğrenme parametreleri ile budama (pruning) işlemlerinin etkisinin zayıf olması gibi dezavantajları bulunmaktadır. XGBoost ve LightGBM algoritmaları, GBTree algoritmasının teknik açıdan zayıf yönlerinin geliştirilmesi sonucunda ortaya çıkmıştır.

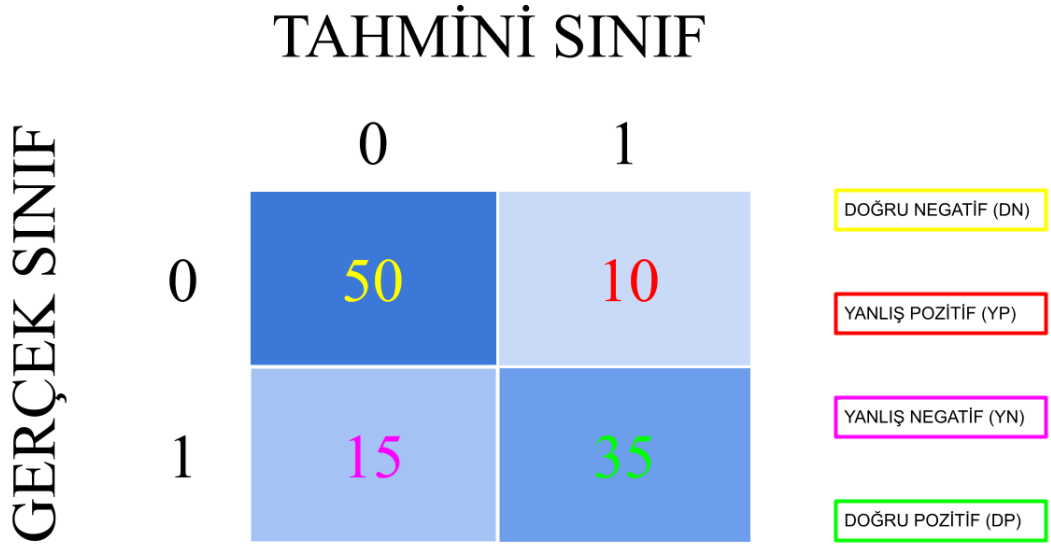
XGBoost algoritmasında gradyan artırma ağaçlarının ölçeklenebilmesi, çekirdek veya dağınık sistemlerde paralel çalışabilmesi ve önbelleğe duyarlı işlemlerin yapılabilmesi gibi yenilikler bulunmaktadır. Algoritmadaki iyileştirmeler karar ağaçlarındaki bölünme noktalarının, özniteliklerin ve zayıf ayrımların belirlenmesi şeklinde ifade edilebilir. Bunun yanında öğrenme parametrelerin belirlenmesinde genel parametreler (örneğin; çekirdek sayısı, grafik işlemci birimi gibi.), öğrenme parametreleri (örneğin; tur sayısı, budama, karar ağacı parametreleri gibi.) ve görev parametreleri (örneğin; değerlendirme ölçütü, erken durdurma gibi.) kullanılabilmektedir.

LightGBM algoritmasında paralel öğrenebilme, hız ve bellek kullanımı optimizasyonu, doğruluk optimizasyonu özellikleri bulunmaktadır. Paralel öğrenebilme özelliği, öznitelikler ve verilerde en iyi ayrımın belirlenmesi ile karar ağaçlarının eğitimin sağlanmasını ve tahminlerin oylanması kapsamaktadır. Hız ve bellek kullanımı optimizasyonu verilerin histogram içerisinde kullanımı ve histogram işlemlerinin paralel işlenmesi ile gerçekleştirilir. Doğruluk optimizasyonu, karar ağaçlarının yaprak bazlı olarak büyümesi ile sağlanmasıdır. Karar ağaçlarının seviye bazlı büyümeye göre yaprak bazlı büyüme sağlaması ile doğruluğu artmaktadır. Ancak, daha fazla veriye ihtiyaç duymaktadır. Algoritma iyileştirmelerinde gradyan tabanlı tek taraflı örnekleme (gradient-based one-side sampling) ve özel öznitelik paketleme (exclusive feature bundling) özellikleri bulunmaktadır. Gradyan tabanlı tek taraflı örneklemede amaç,

küçük gradyanlara sahip verilerin hariç tutulması sonucunda veri sayısının azaltılması ile öznitelikler için bilgi kazancının hesaplanmasıdır. Özel öznitelik paketleme ile özniteliklerin birleştirilmesi sonucunda daha az öznitelik ile model verimliliğinin artırılması hedeflenir. LightGBM algoritması parametrelerin belirlenmesinde, XGBoost algoritmasındaki yeniliklere benzer özellikleri içermektedir.

### 3.2.5. Makine öğrenmesi değerlendirme metrikleri

Sınıflandırma algoritmaları için başarımların ölçülmesi bu bölümde belirtilen hesaplamalar ile gerçekleştirildi. Eğitilen sınıflandırma algoritmaları, test veri kümeleri ile değerlendirilmektedir. Sınıflandırma algoritmalarının, test veri kümesindeki veriler için yaptıkları tahminler tahmini sınıf olarak adlandırılmaktadır. Bir algoritmanın gerçek sınıfları ve tahmin edilen sınıfları ilişkisi Şekil 3.6'da verilen temsili grafikte olduğu üzere, karmaşıklık matrisi (confusion matrix) ile ifade edilmektedir. Temsili gösterimde 0 ve 1 hedef sınıfları ifade etmektedir. Karmaşıklık matrisi gösteriminde dört temel durum bulunmaktadır. Bu durumlar; Doğru Negatif (DN), Yanlış Negatif (YN), Doğru Pozitif (DP) ve Yanlış Pozitif (YP) şeklindedir. DN, gerçekte hedef sınıfı negatif olan verilerin, negatif olarak doğru tahmin edilmesidir. YN, gerçekte hedef sınıfı pozitif olan verilerin, negatif olarak yanlış tahmin edilmesidir. DP, gerçekte hedef sınıfı pozitif olan verilerin, pozitif olarak doğru tahmin edilmesidir. YP, gerçekte hedef sınıfı negatif olan verilerin, pozitif olarak yanlış tahmin edilmesidir.




**Şekil 3.6.** Karmaşıklık matrisi gösterimi

Karmaşıklık matrisinde olan durumlar ile sınıflandırma algoritmaları için skorları ifade edebilen metrikler hesaplanmaktadır. Bu çalışmada hesaplanan metrikler doğruluk (accuracy), kesinlik (precision), duyarlılık (recall) ve F1 skorlarından oluşmaktadır. Doğruluk skoru, test verileri için doğru tahminlerin yüzdesini ifade eder (Denklem 3.3). Kesinlik skoru, belirli bir hedef sınıf için doğru tahminlerin tüm tahmin sınıfına oranı ile ifade edilir (Denklem 3.4). Duyarlılık skoru, belirli bir hedef sınıf için doğru tahminlerin tüm gerçek sınıfa oranı ile ifade edilir (Denklem 3.5). F1 skoru ise duyarlılık ve kesinlik skorlarının harmonik ortalaması ile ifade edilir (Denklem 3.6).

$$Doğruluk\ Skoru = \frac{DN+DP}{DN+YN+DP+YP} \tag{3.3}$$

$$Kesinlik\ Skoru = \frac{DP}{DP+YP} \tag{3.4}$$

$$Duyarlılık\ Skoru = \frac{DP}{YN+DP} \tag{3.5}$$

$$F1\ Skoru = 2 \times \frac{(Kesinlik\ Skoru \times Duyarlılık\ Skoru)}{(Kesinlik\ Skoru + Duyarlılık\ Skoru)} \quad (3.6)$$

Algoritmaların değerlendirilmesinde duyarlılık, kesinlik ve F1 skorlarının hem mikro ve hem de makro değerleri hesaplandı. Mikro değerler, hedef sınıflara veri dağılımının dengesiz olduğu durumlarda sınıflandırma algoritmalarının değerlendirilmesi için yeterli bir bilgi sağlamamaktadır. Çünkü çok fazla veriye sahip bir hedef sınıftaki sınıflandırma başarısı, diğer hedef sınıfların sınıflandırma başarılarına ait olumsuz etkileri gizlemektedir. Makro değerler ise, veri dağılımı eşit kabul edilmesi ile hesaplanmaktadır. Bu sebeple makro değerler, veri dağılımının dengesizliğinden etkilenmeyen skorları ifade etmesiyle algoritmaların değerlendirilmesine fayda sağlamaktadır. Bu çalışmada yapılan deneyden elde edilen sonuçlar, F1 skorunun makro sonuçları dikkate alınmasıyla değerlendirilmektedir.

### 3.2.6. Hiper-parametre optimizasyonu

Hiper-parametre optimizasyonu, makine öğrenmesi algoritmalarının eğitim aşamasında, algoritmanın yapısını oluşturulan parametrelerin en iyi değerlerinin belirlenmesidir. Ağ trafiği sınıflandırmasında, sürekli olarak veri kümesinin güncellenmesi ve hedef sınıf sayısının artması durumlarında, algoritmalar için en iyi parametrelerin belirlenme ihtiyacı oldukça önemlidir. Bu sebeple önerilen sistemde, algoritmaların yapısında etkili parametrelerin istenilen değer aralığı ve artış miktarlarının tanımlanması ile hiper-parametre optimizasyonu gerçekleştirilir.

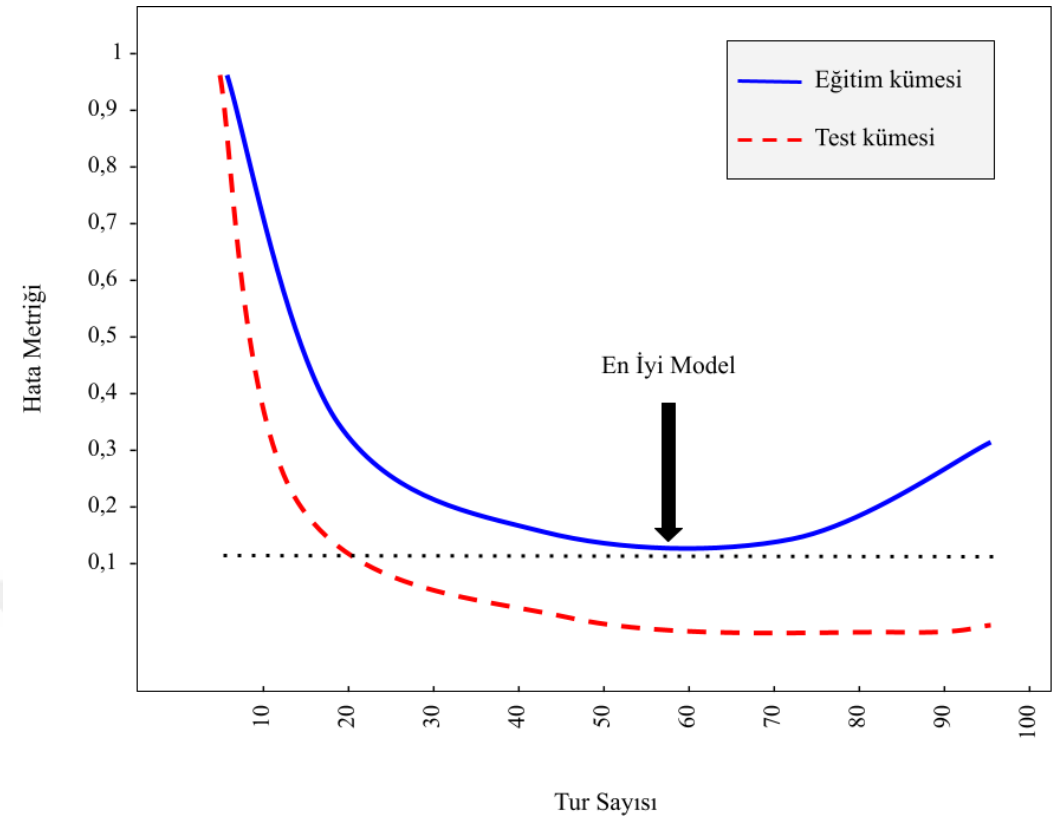
Hiper-parametre optimizasyonunun uygulanmasında, k-katmanlı çapraz-doğrulama (cross-validation, CV) tekniği kullanıldı. Çapraz doğrulamada, belirlenen k değerine göre veri kümesi bölünmektedir. Örneğin; 5 olarak belirlenen k değerinde, veri kümesinde 1000 veri var ise, her katmanda 200 veri olacak şekilde 5 katmanlı veri kümesi oluşturulur. Çapraz doğrulamada, k-katmanlı veri kümesinin bir katmanı test diğer katmanları eğitim verileri olarak değerlendirilir. Makine öğrenmesi algoritması k-katmanı kadar veri kümesi ile eğitilir ve test edilir. Sonuç olarak, algoritmanın

performansı ölçülür. Bu yöntem, algoritmanın parametreleri için belirlenen değer kümesindeki her bir parametre değerinde uygulanır. Örneğin; XGBoost algoritmasının "max_depth" parametresi için belirlenen değer kümesi 3, 5 ve 7 değerlerinden oluşursa, çapraz doğrulama tekniği "max_depth" parametresinin aldığı her bir değeri için uygulanmaktadır. Bu şekilde algoritmanın parametreleri için belirlenen değer aralığındaki tüm parametreler için çapraz doğrulama sonucundan en iyi performansa ulaşan değerler belirlenir. Hiper-parametre optimizasyonunda, algoritmalar için kullanılan parametreler ve açıklamaları Çizelge 3.3'te gösterilmektedir.

**Çizelge 3.3.** Hiper-parametre optimizasyonunda kullanılan parametreler ve açıklamaları

| Parametre | Değer Aralığı | | Açıklama |
|---|---|---|---|
| | min. | maks. | |
| "num_iterations" | 1 | - | Karar ağacı sayısı (tur sayısı). |
| "learning_rate" | 0.0 | 1.0 | Karar ağacı tahminlerini öğrenme oranı. |
| "max_depth" | 0 | - | Karar ağacı maksimum derinliği. |
| "max_leaf_nodes" | 0 | - | Karar ağacında maksimum yaprak düğümü sayısı. |
| "min_samples_leaf" | 0 | - | Bir yaprak düğümü için minimum örnek sayısı. |
| "min_child_weight" | 0 | - | Bir yaprak düğümünde ayrım yapılabilmesi için minimum örnek sayısı. |
| "min_sum_hessian_in_leaf" | 0 | - | "min_child_weight" ile aynı ifadedir. |
| "subsample" | 0.0 > | 1.0 | Karar ağaçlarında kullanılacak veri kümesinin alt örnekleme oranı. |
| "colsample_bytree" | 0.0 > | 1.0 | Her karar ağacı için kullanılacak alt örnekleme oranı. |
| "lambda_L1" | 0 | - | Ağırlıklar üzerinde L1 düzenlileştirme parametresi. |
| "lambda_L2" | 1 | - | Ağırlıklar üzerinde L2 düzenlileştirme parametresi. |
| "lambda" | 1 | - | "lambda_L2" ile aynı ifadedir. |
| "alpha" | 0 | - | "lambda_L1" ile aynı ifadedir. |
| "gamma" | 0 | - | Yaprak düğümde büyüme için gereken minimum kayıp azaltma değeri. |
| "bagging_freq" | 0 | 1 | Karar ağacı için alt örnekleme oluşturma sıklığı. |

Hiper-parametrelerin belirlenmesi maliyetli bir işlem olması sebebiyle algoritmalarda erken durdurma "early stopping" tekniği kullanıldı (Şekil 3.7). Erken durdurma tekniği, eğitim sırasında her turda değerlendirilen algoritmanın başarısında bir ilerleme olmadığında, eğitim işleminin sonlandırılmasıdır. Erken durdurma ile eğitim süresi ve model karmaşıklığı azaltılmaktadır. Ayrıca, algoritmaların parametreleri için belirlenen değer kümesinin genişletilebilmesine olanak sağlamaktadır.




**Şekil 3.7.** Erken durdurma tekniğinin temsili gösterimi

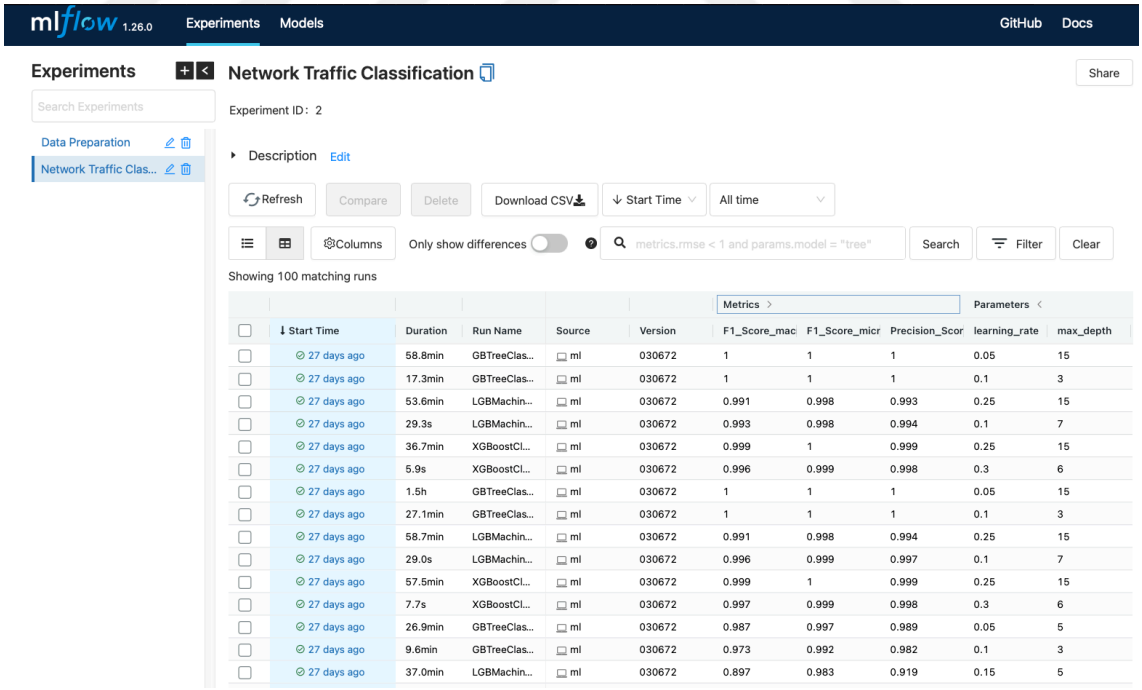
### 3.2.7. Süreç yönetimi

Makine öğrenmesi tekniklerinin uygulandığı bir proje, standart yazılım geliştirme yaşam döngülerinin uygulanamadığı, deneysel olan bir sürece sahiptir (Zaharia vd., 2018). Makine öğrenmesinde, deneysel süreçlerin etkili bir şekilde yönetilme ihtiyacı, yeni bir süreç yönetimi yapısını gerektirmektedir. Bu sebeple, makine öğrenmesinde uygulanan süreçlerin sağlıklı ve yapılan deneylerin tekrarlanabilir olmasını sağlamak amacıyla MLFlow yapısı kullanıldı.

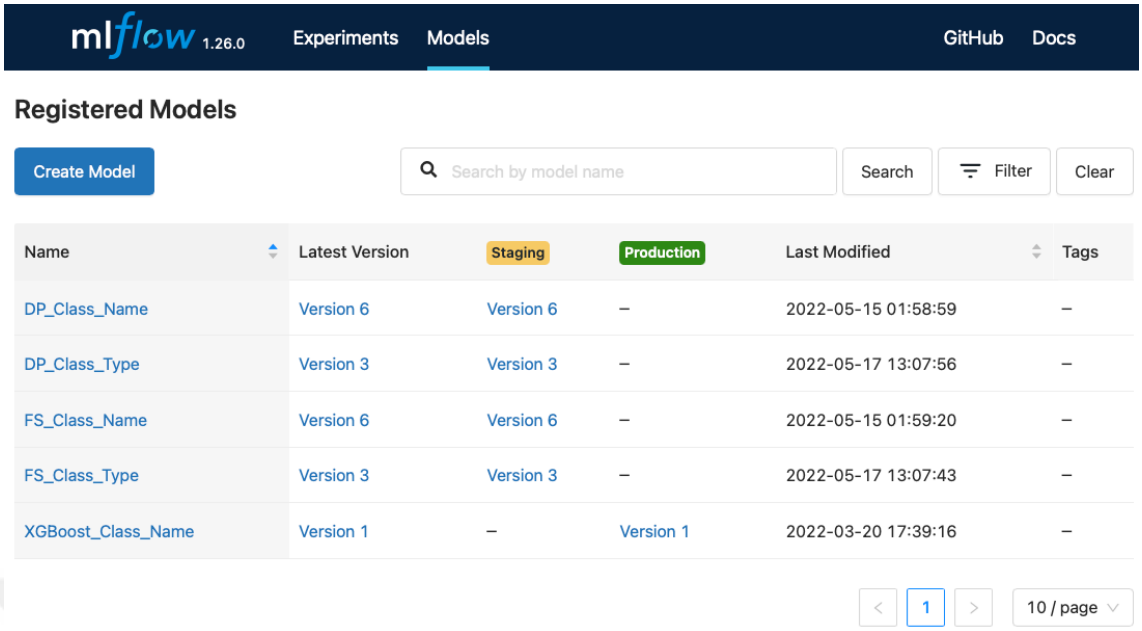
MLFlow, makine öğrenmesi süreçlerinde versiyon kontrolünü, geliştirme ve ürün yönetimini, deneylerin izlenmesini ve tekrarlanabilmesini kolaylaştıran bir çerçeve sunmaktadır (Zaharia vd., 2018). MLFlow izleme, proje ve model bileşenlerinden oluşmaktadır. İzleme bileşeni, gerçekleştirilen deney ile ilgili parametreleri, değerlendirme sonuçlarını, kaynak kodları, girdi verilerini ve diğer tüm çıktıların

kayıtlarını tutar. Bu kayıtlara bir uygulama programlama arayüzü (application programing interface, API) veya web tabanlı kullanıcı arayüzü (user interface, UI) aracılığıyla erişilebilmesini sağlar. Proje bileşeni, gerçekleştirilen proje ile ilgili gerekli yazılım kütüphanelerini, kullanılan versiyonlarını ve bağlantılı diğer verilere ilişkin bilgileri sunmaktadır. Model bileşeni, eğitilen makine öğrenmesi modellerinin ürün olarak dağıtımının gerçek zamanlı yapılmasına olanak sağlar.

Tez çalışmamızda kullanılan MLFlow yapısındaki deneyler ve modellerin listelendikleri kullanıcı arayüzleri sırasıyla Şekil 3.8 ve Şekil 3.9'da gösterilmektedir. Kullanıcı arayüzlerinde listelenen deneyler ve modellerin detaylı olarak incelenebilmektedir. Deney çıktıları ayrıntılı olarak incelenebilmekte ve öğrenme modelleri model olarak kaydedilebilmektedir. Kayıtlı modellerin, modellerin listelendiği kullanıcı arayüzünde ilgili model grubunun detaylarının gösterildiği sayfa içerisinde versiyon kontrolleri yapılabilmektedir. Deney içerisinde kullanılan modellerin değiştirilmesi, MLFlow arayüzünde ilgili modellerin kullanım durumunun güncellenmesi ile sağlanabilmektedir.

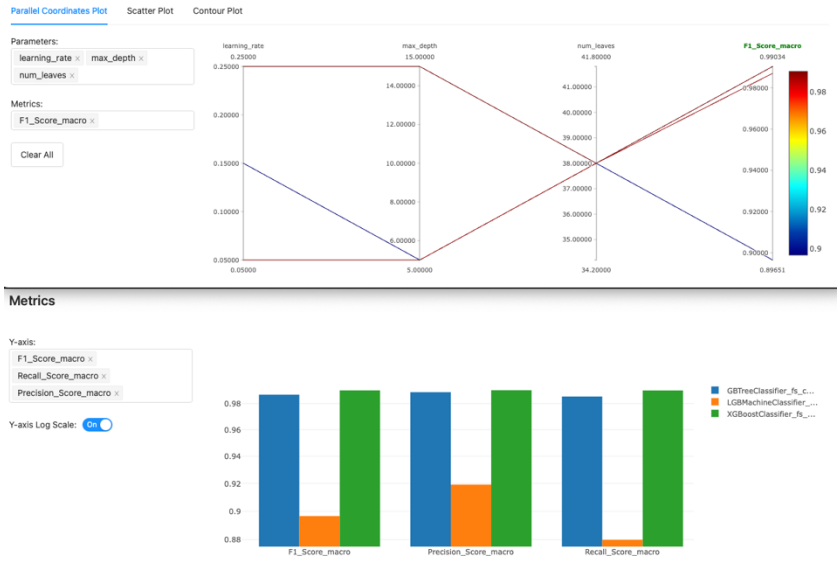



**Şekil 3.8.** MLFlow kullanıcı arayüzünde deneylerin listelendiği sayfanın ekran görüntüsü




**Şekil 3.9.** MLFlow kullanıcı arayüzünde modellerin listelendiği sayfanın ekran görüntüsü

Ayrıca deneyler, kullanılan parametrelere ve değerlendirme ölçümlerine göre ayrıntılı olarak kıyaslanabilmekte ve interaktif olarak görselleştirilebilmektedir. İnteraktif görselleştirme sonuçlarına örnekler Şekil 3.10'da gösterilmektedir.




**Şekil 3.10.** MLFlow kullanıcı arayüzünde örnek interaktif görselleştirmelerin ekran görüntüleri

Bu tez çalışmasında, bu aşamaya kadar şifreli ağ trafiğinin sınıflandırılması için geliştirilen platform içerisinde uygulanan yöntemler detaylandırıldı. Önerilen platform öznitelik çıkarımı, veri hazırlama ve makine öğrenmesi aşamalarından oluşan uçtan-uca şifreli ağ trafiğini sınıflandırmayı sağlamaktadır. Önerilen platform üzerinde her aşamaya uygun, etkili ve performansı kanıtlanan yazılım teknolojileri kullanılmıştır. Öznitelik çıkarımı için kullanılan NFStream çerçevesi güncel ve performanslı yönleri ile gerçek-zamanlı ağ trafiklerinde uygulanabilir olması açısından çalışmamızda kullanılmıştır. Ayrıca öznitelik çıkarımı aşamasında şifreli ağ trafiği verilerindeki gözlemlerimiz ışığında "pattern byte" olarak isimlendirdiğimiz yeni bir öznitelik çıkarılmıştır. Veri hazırlama aşamasında büyük veri hacmine sahip verilerin performans odaklı işlenebilmesini sağlayan Apache Spark çerçevesi kullanılmıştır. Ayrıca bu aşama içerisinde Ki-Kare ve ANOVA teknikleri ile öznitelik seçimi gerçekleştirilmiştir. Veri hazırlama aşamasının tamamlanması sonucunda makine öğrenmesi algoritmaları tarafından işlenebilir eğitim ve test veri kümeleri elde edilmektedir. Makine öğrenmesi aşamasında GBTree, XGBoost ve LightGBM algoritmaları kullanılmakta ve performansları ölçülmektedir.

Bu bölümde detaylandırılan yöntem ile şifreli ağ trafiği sınıflandırma hedefleri için gerçekleştirilen deneylere ilişkin bulgular bir sonraki bölümde sunulmuştur.

# 4. BULGULAR

Bu bölümde, ağ trafiği sınıflandırılmasında önerilen sistem ile gerçekleştirilen deneylere ilişkin sonuçlar gösterildi. Ağ trafiği sınıflandırma problemlerinden, ağ trafiğinin uygulama ve uygulama türlerine yönelik iki farklı sınıflandırma hedefleri üzerine deney gerçekleştirildi.

Hedeflenen sınıflandırma problemleri gerçekleştirilen deneyde görev olarak tanımlaması ile isimlendirilmesi şu şekildedir:

- Görev A: Uygulama sınıflandırma
- Görev B: Uygulama türü sınıflandırma

Deney kapsamında, Bölüm 3.1.1'de detaylandırılan UNB ISCX VPN-nonVPN veri kümesi kullanıldı. Veri kümesinde, "pcap" uzantılı dosyalarda bulunan ağ paketi verileri Bölüm 3.2.1' detaylandırılan öznitelik çıkarımı yönteminin uygulanması ile ağ akışı verileri oluşturuldu. Ağ akışı verilerinin oluşturulmasında, ağ akışı süresi 15 saniye olarak belirlendi. Ağ akışını, şifreli veya şifresiz olan ağ trafiğinin 15 saniye boyunca gelen ve giden ağ paketlerinden oluşur. Deneyde bir ağ akışı verisi, ağ akışı oluştururken çıkarılan öznitelikler ile ifade edildi.

Bir ağ akışında çıkarılan öznitelikler, Çizelge 4.1'de verilen öznitelik tablosunda gösterildi. Öznitelik isimlendirmesi, öznitelik tablosunda bulunan öznitelik sütunu ile belirtilen akış yönüne göre yapıldı. Öznitelik tablosunda, öznitelikler belirtilen akış yönleri için ayrı ayrı hesaplandı. Akış yönleri kaynaktan hedefe (K-H), hedeften kaynağa (H-K) ve çift yönlü olarak değerlendirildi. Öznitelik isimlendirmesinde kaynaktan hedefe, hedeften kaynağa ve çift yönlü olan akış yönleri sırasıyla "src2dst", "dst2src" ve "bidirectional" olarak ifade edildi. Öznitelik hesaplamasında, akış yönü sadece çift yönlü olarak belirtilen öznitelikler için "bidirectional" ifadesi isimlendirmede kullanılmadı. Öznitelik isimlendirmelerine örnekler şu şekildedir:
(öznitelik + akış yönü = öznitelik adı)

- "protocol" + "bidirectional" = "protocol"
- "packets" + "bidirectional" = "bidirectional_packets"

- “packets” + “src2dst” = “src2dst_packets”
- “bytes” + “dst2src” = “dst2src_bytes”

**Çizelge 4.1.** Öznitelik tablosu

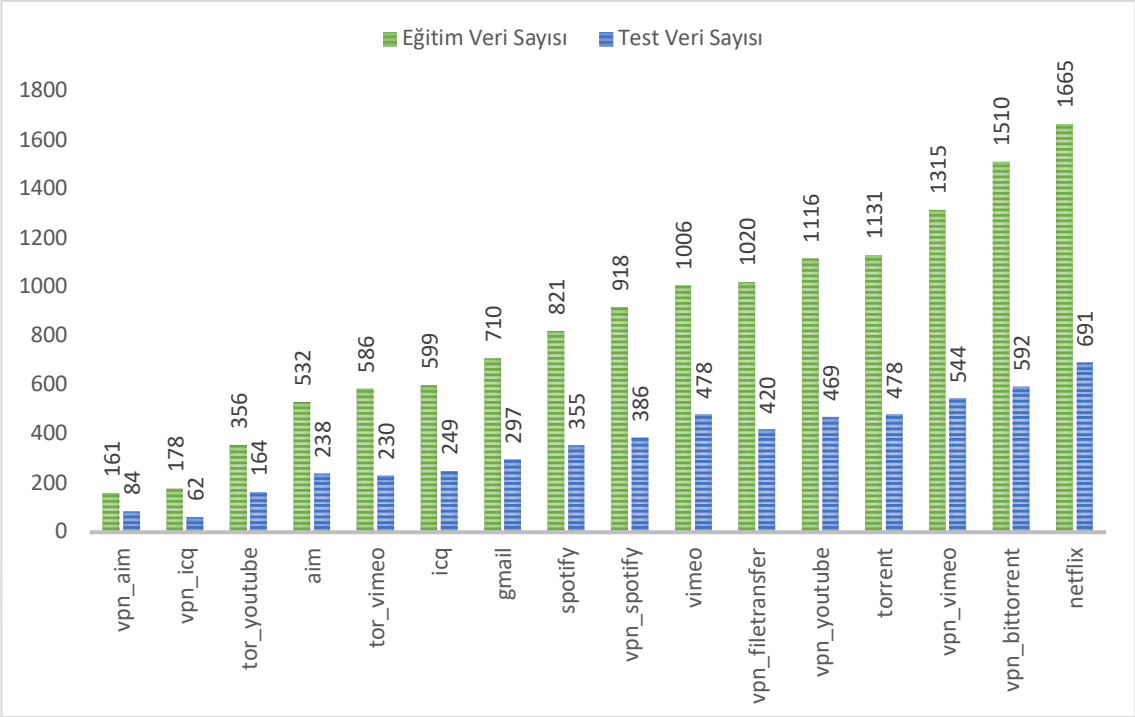
| Öznitelik | Akış Yönü | | | Açıklama |
|---|---|---|---|---|
| | K-H | H-K | Çift Yönlü | |
| **id** | | | ✓ | Akış tanımlayıcı. |
| **expiration_id** | | | ✓ | Akış sonlandırma tanımlayıcısı. |
| **src_ip** | | | ✓ | Akış kaynak IP adresi metinsel gösterimi. |
| **src_mac** | | | ✓ | Akış kaynak MAC adresi metinsel gösterimi. |
| **src_oui** | | | ✓ | Akış kaynak OUI adresi metinsel gösterimi. |
| **src_port** | | | ✓ | Akış taşıma katmanı kaynak port. |
| **dst_ip** | | | ✓ | Akış hedef IP adresi metinsel gösterimi. |
| **dst_mac** | | | ✓ | Akış hedef MAC adresi metinsel gösterimi. |
| **dst_oui** | | | ✓ | Akış hedef OUI adresi metinsel gösterimi. |
| **dst_port** | | | ✓ | Akış taşıma katmanı hedef port. |
| **protocol** | | | ✓ | Akış taşıma katmanı protokol tanımlayıcı. |
| **ip_version** | | | ✓ | Akış IP versiyonu. |
| **vlan_id** | | | ✓ | Akış Sanal LAN tanımlayıcı. |
| **first_seen_ms** | ✓ | ✓ | ✓ | Akıştaki ilk paketin zaman damgası (milisaniye). |
| **last_seen_ms** | ✓ | ✓ | ✓ | Akıştaki son paketin zaman damgası (milisaniye). |
| **duration_ms** | ✓ | ✓ | ✓ | Akış süresi (milisaniye). |
| **packets** | ✓ | ✓ | ✓ | Akıştaki toplam paket boyutu. |
| **bytes** | ✓ | ✓ | ✓ | Akıştaki toplam bayt boyutu. |
| **tunnel_id** | | | ✓ | Tünel tanımlayıcı |
| **min_ps** | ✓ | ✓ | ✓ | Akışta minimum paket boyutu. |
| **mean_ps** | ✓ | ✓ | ✓ | Akışta ortalama paket boyutu. |
| **stdev_ps** | ✓ | ✓ | ✓ | Akışta paket boyutu örnek standart sapma. |
| **max_ps** | ✓ | ✓ | ✓ | Akışta maksimum paket boyutu. |
| **min_piat_ms** | ✓ | ✓ | ✓ | Akışta minimum paket varış süresi (milisaniye). |
| **mean_piat_ms** | ✓ | ✓ | ✓ | Akışta ortalama paket varış süresi (milisaniye). |
| **stdev_piat_ms** | ✓ | ✓ | ✓ | Akışta paket varış süresi örnek standart sapması (milisaniye). |
| **max_piat_ms** | ✓ | ✓ | ✓ | Akışta maksimum paket varış süresi (milisaniye). |
| **syn_packets** | ✓ | ✓ | ✓ | Akış paketlerindeki TCP SYN bayraklarının toplam sayısı. |
| **cwr_packets** | ✓ | ✓ | ✓ | Akış paketlerindeki TCP CWR bayraklarının toplam sayısı. |
| **ece_packets** | ✓ | ✓ | ✓ | Akış paketlerindeki TCP ECE bayraklarının toplam sayısı. |
| **urg_packets** | ✓ | ✓ | ✓ | Akış paketlerindeki TCP URG bayraklarının toplam sayısı. |
| **ack_packets** | ✓ | ✓ | ✓ | Akış paketlerindeki TCP ACK bayraklarının toplam sayısı. |
| **psh_packets** | ✓ | ✓ | ✓ | Akış paketlerindeki TCP PSH bayraklarının toplam sayısı. |
| **rst_packets** | ✓ | ✓ | ✓ | Akış paketlerindeki TCP RST bayraklarının toplam sayısı. |
| **fin_packets** | ✓ | ✓ | ✓ | Akış paketlerindeki TCP FIN bayraklarının toplam sayısı. |
| **pattern_byte** | | | ✓ | Akış içerisindeki paketlerde belirlenen konumların bayt kalıpları. |

Öznitelik çıkarımında, Bölüm 3.2.2'de detaylandırılan yeni tanımlanan öznitelik olan "pattern_byte" özniteliğinin yapısı gereği isimlendirilmesi diğer özniteliklerinden farklıdır. "pattern_byte" özniteliği isimlendirmede sonuna desenin ifade ettiği yük verisindeki baytın konumunun eklenmesiyle gerçekleştirildi. Örneğin; yük verisinde 8. bayt konumu için çıkarılan "pattern_byte" özniteliği, "pattern_byte_8" olarak isimlendirilmektedir. Bu deneyde, "pattern_byte" özniteliği yük verisinin ilk 16 baytlık alanı için hesaplandı. Bu şekilde "pattern_byte_0" ile "pattern_byte_16" arasında toplam 16 öznitelik çıkarıldı.

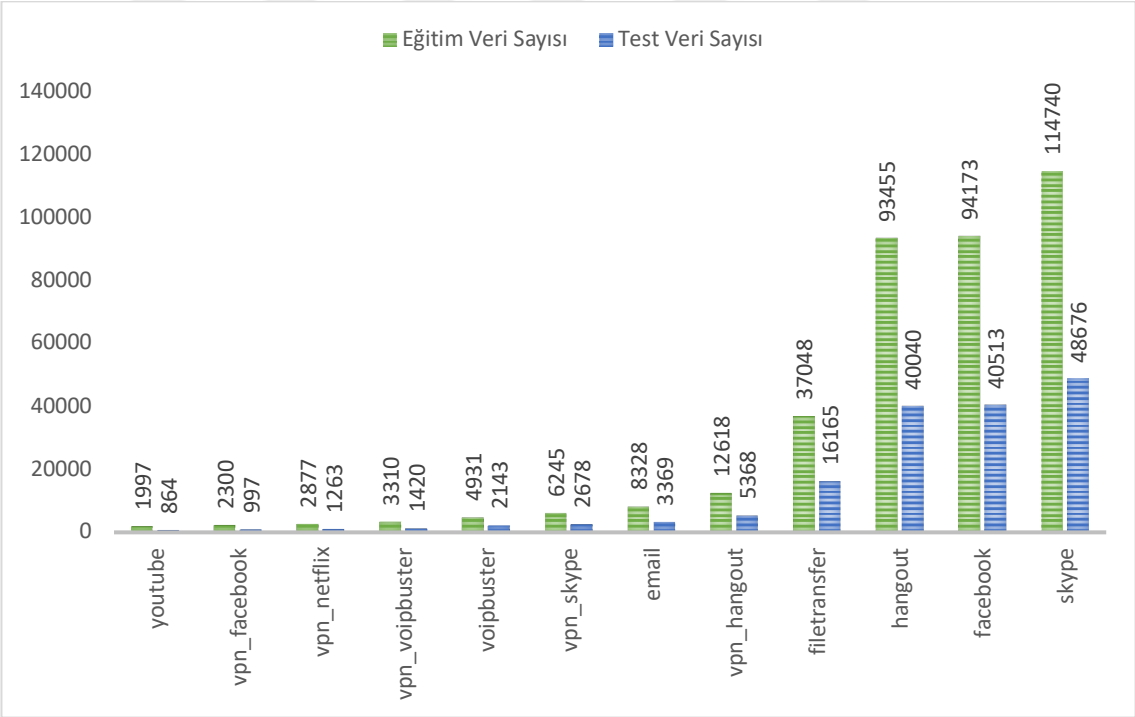
Öznitelik çıkarımı sonrasında bir ağ akışı verisi için toplamda 93 adet öznitelik çıkarıldı. Ağ akışı verileri, Bölüm 3.2.3 detaylandırılan veri hazırlama yönteminde veri ön işleme tünelinden geçirildi. Ön işleme tünelinde "id", "expiration_id", "src_ip", "src_mac", "src_oui", "dst_ip", "dst_mac", "dst_oui", "vlan_id" öznitelikleri, makine öğrenmesi sürecinde sınıflandırma için bir anlam ifade etmediğinden dolayı temizlendi. Ön işlemi tüneli sonucundan öznitelik sayısı 93'ten 84'e düşürüldü.

Ön işleme tünelinin tamamlanmasının ardından, ağ akışı verileri eğitim ve test veri kümesi olarak ayrıldı. Veri kümesini ayırma işleminde, verilerin %70 eğitim geriye kalanı ise test veri kümesi olarak ayrıldı. Eğitim ve test veri kümelerinin ayrılmasında hedef sınıflar için ayırma oranları eşit olarak tanımlandı. Her hedef sınıfın verileri için eğitim verisinin %70, test verisinin ise %30 oranlarında ayrım yapıldı. Bu süreç A ve B görevleri için uygulandı.

Görev A'da hedef sınıflar için eğitim ve test verilerinin dağılımı Şekil 4.1 ve Şekil 4.2'de verilen grafikler ile gösterildi. Görev A için toplamda 28 adet hedef sınıf bulunur. Hedef sınıflarını, ağ trafiği uygulamalarının normal ve VPN ağ trafikleri oluşturur. Örneğin; YouTube uygulamasının hedef sınıf olarak isimlendirilmesi, normal ağ trafiğinde "youtube" ve VPN kullanımı içeren ağ trafiğinde "vpn_youtube" şeklindedir. Görev A'da hedef sınıf sayısının fazla olması sebebiyle veri dağılım iki grafik üzerinde gösterildi. Hedef sınıfın toplam veri sayısı, 2500'den az olanlar Şekil 4.1'de, 2500'den büyük ve eşit olanlar Şekil 4.2'de gösterildi.



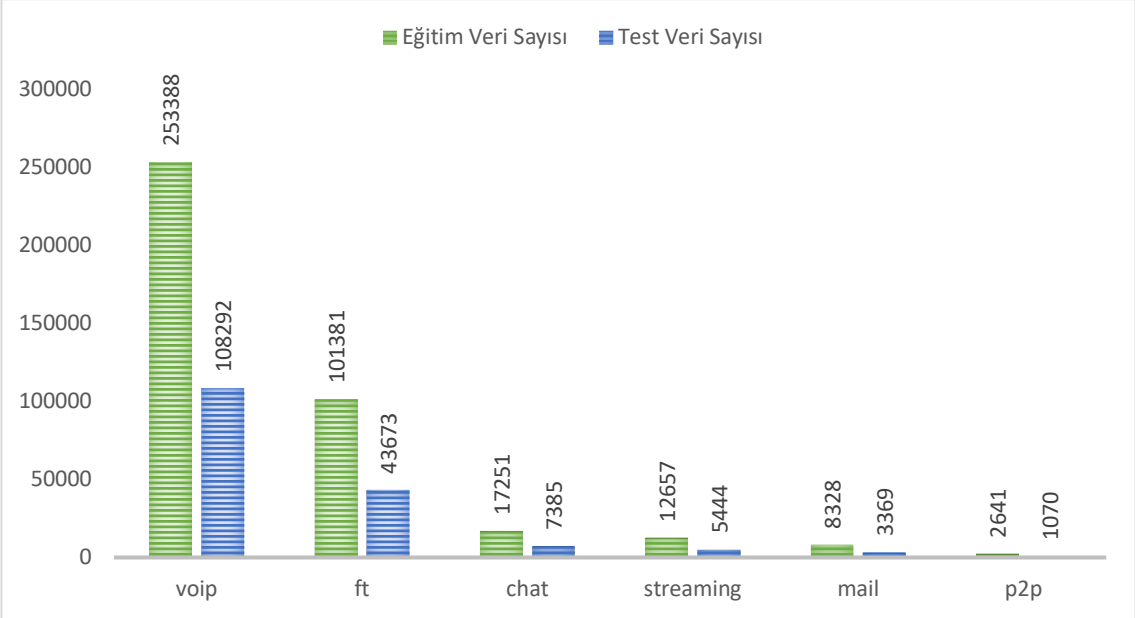

**Şekil 4.1.** Görev A'da hedef sınıflar için veri dağılımı ($sınıf\ verisi < 2500$)




**Şekil 4.2.** Görev A'da hedef sınıflar için veri dağılımı ($sınıf\ verisi \geq 2500$)

Görev A'da hedef sınıflarından eğitim ve test verilerinin toplam veri sayısı, en fazla "skype" sınıfında 163416 veri, en az "vpn_aim" sınıfında 245 veri bulunmaktadır. Görev A için eğitim ve test veri sayılarının standart sapmaları sırasıyla 31555,1221 ve 13488,1927 olduğu görülmektedir. Veri dağılımının toplam standart sapması 45042,9616'dır.

Görev B için veri kümesindeki uygulamaların ağ trafiği türlerine göre gruplandırılmasıyla hedef sınıflar tanımlandı. Gruplama işlemi, veri kümesinde "pcap" dosyalarında yapılan isimlendirmelerde belirtilen ağ trafiği türlerine göre yapıldı. Görev B'de bulunan hedef sınıfların eğitim ve test veri sayılarının dağılımı Şekil 4.3'te gösterildi.




**Şekil 4.3.** Görev B'de hedef sınıflar için veri dağılımı

Görev B'nin hedef sınıfları arasında toplam veri sayısı, en fazla "voip" sınıfında 361680 veri ve en az "p2p" sınıfında 3711 veri bulunmaktadır. Görev B için eğitim ve test verilerinin standart sapmaları sırasıyla 98922,6162 ve 42326,9589 olduğu görülmektedir. Veri dağılımının toplam standart sapması ise 141249,336'dır.

Görev A ve B için veri kümelerinin veri ön işleme sonucunda eğitim ve test veri kümeleri elde edildi. Sadece veri ön işlemesi sonucunda elde edilen veri kümeleri, yapılan deneyde

“varsayılan” durum olarak ifade edildi. Varsayılan durum, öznitelik çıkarımı ve veri temizleme işlemlerinin yapıldığını gösterir. Veri kümesi, varsayılan durum sonrasında 84 adet özniteliğe sahiptir.

Veri ön işlemede elde edilen eğitim veri kümeleri için veri hazırlama yönteminde bulunan öznitelik seçimi tünelindeki işlemler uygulandı. Öznitelik seçimi tünelinin uygulanması ile eğitim veri kümesinde bulunan öznitelikler arasından ayırt edici etkisi yüksek olan öznitelikler seçilmektedir. Kategorik ve sürekli özniteliklerin, ayırt edici etkilerine göre sıralanması sonucunda ilk %50’yi oluşturan öznitelikler seçildi. Seçilen özniteliklerin, eğitim ve test veri kümelerinde kullanılması “öznitelik seçimi (feature selection, FS)” durumu olarak ifade edilmektedir. Öznitelik seçimi durumu, öznitelik çıkarımı ve veri hazırlama yöntemindeki veri ön işleme ve öznitelik seçimi tünellerinin uygulandığını gösterir. Veri kümesi, öznitelik seçimi durumunun uygulanması sonucunda 42 adet öznitelik sayısı içerir.

Varsayılan ve öznitelik seçimi durumları sonrasında veri kümeleri, Bölüm 3.2.4’te detaylı olarak ifade edilen makine öğrenmesi yönteminde kullanıldı. Deney, makine öğrenmesi algoritmalarından GBTree, LightGBM ve XGBoost algoritmaları ile gerçekleştirildi. Deney, makine öğrenmesi algoritmalarının eğitim ve test aşamalarını içermektedir. Eğitim aşamasında, algoritmalar eğitim veri kümesi ile eğitilmesi sonucunda makine öğrenmesi modeli oluşturuldu. Test aşamasında, eğitilen makine öğrenmesi modeli ile test verilerinin hedef sınıfları tahmin edildi. Tahmin ve gerçek hedef sınıfları için sonuçların değerlendirilmesinde doğruluk, kesinlik, duyarlılık ve F1 skorları hesaplandı.

Makine öğrenmesi algoritmaları için Bölüm 3.2.5’te detaylandırılan hiper-parametre optimizasyonu uygulanması ile algoritmalar için önemli etkiye sahip parametrelerin en iyi değerleri belirlendi. Hiper-parametre optimizasyonun uygulandığı durum “çapraz-doğrulama (cross-validation, CV)” olarak ifade edilmektedir. Hiper-parametre optimizasyonu, varsayılan durum için uygulanırsa “çapraz-doğrulama”, öznitelik seçimi durumundan sonra uygulanırsa “öznitelik seçimi ve çapraz-doğrulama (FS ve CV)” durumları olarak ifade edilir.

Makine öğrenmesi algoritmaları için çapraz-doğrulama durumunun uygulanmasında kullanılan değer aralıkları ve parametreler Çizelge 4.2'de gösterilmektedir. Parametrelerin değer kümelerinde GBTree, LightGBM ve XGBoost algoritmalarındaki aynı parametreler için eşit değerler verildi. Farklı parametrelerde ise algoritmalar için değer aralıkları farklılık göstermektedir. Algoritmalar için aynı parametreler farklı isimlendirmeler içermektedir. Bu sebeple bazı parametreler tek bir isim ile ifade edildi. Örneğin; GBTree ve LightGBM algoritmalarında "learning_rate" olarak isimlendirilen parametre, XGBoost algoritmasında "eta" olarak isimlendirilmektedir.

**Çizelge 4.2.** Çapraz-doğrulama işleminde algoritmalar için kullanılan parametreler ve değer aralıkları

| Parametre | Algoritma | | | Değer Aralığı ve Artış Miktarı | | | Değer Kümesi |
|---|---|---|---|---|---|---|---|
| | GBTree | LightGBM | XGBoost | Min. | Maks. | Artış | |
| "num_iterations" | ✓ | ✓ | ✓ | 50 | 250 | 100 | [50, 150, 250] |
| "learning_rate" | ✓ | ✓ | ✓ | .05 | .25 | .10 | [.05, .15, .25] |
| "max_depth" | ✓ | ✓ | ✓ | 5 | 15 | 5 | [5, 10, 15] |
| "max_leaf_nodes" | ✓ | ✓ | ✓ | 38 | 307 | - | [38, 307] |
| "min_samples_leaf" | ✓ | ✓ | | 20 | 40 | - | [20, 40] |
| "min_child_weight" | | | ✓ | 3 | 12 | 3 | [3, 6, 9, 12] |
| "min_sum_hessian_in_leaf" | | ✓ | | .001 | .003 | - | [.001, .003] |
| "subsample" | | ✓ | ✓ | .6 | 1.0 | .2 | [.6, .8, 1.0] |
| "colsample_bytree" | | ✓ | ✓ | .6 | 1.0 | .2 | [.6, .8, 1.0] |
| "lambda_L1" | | ✓ | | 0 | 32 | 16 | [0, 16, 32] |
| "lambda_L2" | | ✓ | | 1 | 29 | 14 | [1, 15, 29] |
| "lambda" | | | ✓ | 1 | 19 | 3 | [1, 4, 7, 10, 13, 16, 19] |
| "alpha" | | | ✓ | 0 | 18 | 3 | [0, 3, 6, 9, 12, 15, 18] |
| "gamma" | | | ✓ | 0 | 40 | 10 | [0, 10, 20, 30, 40] |
| "bagging_freq" | | ✓ | | 10 | 20 | - | [10, 20] |

Bu aşamaya kadar gerçekleştirilen dört durum bulunmaktadır. Bu durumlar varsayılan, öznitelik seçimi (FS), çapraz-doğrulama (CV), öznitelik seçimi ve çapraz-doğrulama (FS ve CV) durumlarından oluşmaktadır. Bu dört durumun Görev A ve B için GBTree, LightGBM ve XGBoost sınıflandırma algoritmalarının performanslarına ilişkin sonuçların doğruluk, kesinlik, duyarlılık ve F1 skorları hesaplandı. Değerlendirme sonuçlarına ilişkin şekillerde kesinlik, duyarlılık ve F1 skorlarının makro sonuçları verildi. Görev A ve B için veri kümesinde, veri dağılımının dengesiz olmasından dolayı

değerlendirme skorlarından makro sonuçlar dikkate alındı. F1 skoru, kesinlik ve duyarlılık skorlarının harmonik ortalamasını ifade ettiğinden dolayı skorların kıyaslanabilmesi için F1 skorları incelendi.

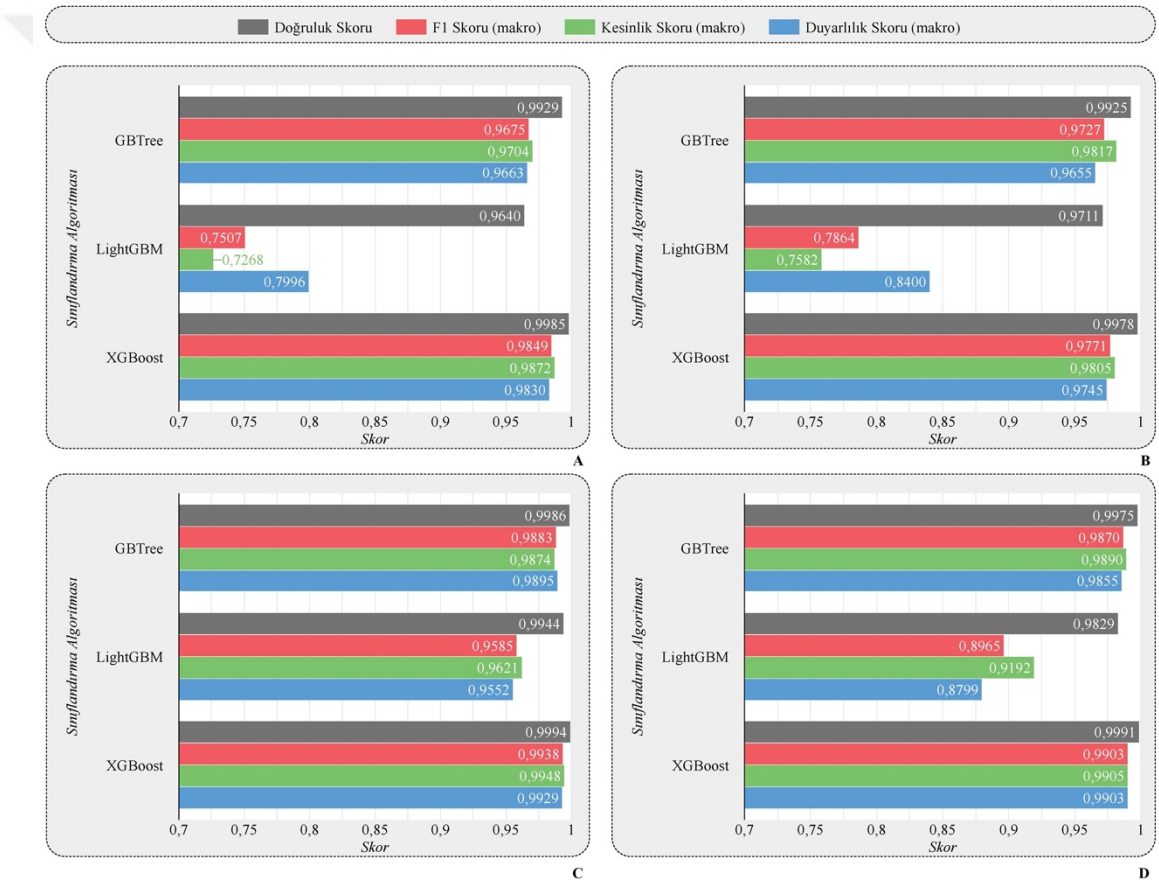
Görev A'da makine öğrenmesi algoritmaları için kullanılan parametrelerin varsayılan, çapraz-doğrulama ve öznitelik ve çapraz-doğrulama durumları sonrasında belirlenen değerleri Çizelge 4.3'te gösterildi. İlgili çizelgede varsayılan olarak belirtilen sütun varsayılan durum ile öznitelik seçiminde aynı değerler olduğundan sadece varsayılan durum olarak gösterildi.

**Çizelge 4.3.** Görev A'nın sınıflandırma algoritmalarında kullanılan parametrelerin, varsayılan ve çapraz-doğrulama sonrasında belirlenen değerleri

| Parametre | GBTree | | | LightGBM | | | XGBoost | | |
|---|---|---|---|---|---|---|---|---|---|
| | Varsayılan | CV | FS ve CV | Varsayılan | CV | FS ve CV | Varsayılan | CV | FS ve CV |
| "num_iterations" | 100 | 50 | 50 | 100 | 50 | 50 | 100 | 50 | 50 |
| "learning_rate" | 0.1 | 0.05 | 0.05 | 0.1 | 0.15 | 0.15 | 0.3 | 0.25 | 0.25 |
| "max_depth" | 3 | 10 | 5 | 7 | 10 | 5 | 6 | 15 | 15 |
| "max_leaf_nodes" | 75 | 307 | 38 | 75 | 38 | 38 | 75 | 38 | 38 |
| "min_samples_leaf" | 20 | 20 | 20 | 20 | 20 | 20 | - | - | - |
| "min_child_weight" | - | - | - | - | - | - | 1 | 3 | 3 |
| "min_sum_hessian_in_leaf" | - | - | - | 0.001 | 0.001 | 0.001 | - | - | - |
| "subsample" | - | - | - | 1.0 | 0.6 | 0.6 | 1.0 | 1.0 | 1.0 |
| "colsample_bytree" | - | - | - | 1.0 | 0.8 | 1.0 | 1.0 | 0.8 | 0.6 |
| "lambda_L1" | - | - | - | 0 | 0 | 16 | - | - | - |
| "lambda_L2" | - | - | - | 0 | 1 | 15 | - | - | - |
| "lambda" | - | - | - | - | - | - | 1 | 1 | 1 |
| "alpha" | - | - | - | - | - | - | 0 | 0 | 0 |
| "gamma" | - | - | - | - | - | - | 0 | 0 | 0 |
| "bagging_freq" | - | - | - | 0 | 10 | 10 | - | - | - |

Görev A için sınıflandırma algoritmalarının dört durumdaki değerlendirme skorları Şekil 4.4'te gösterildi. Varsayılan durumda en iyi F1 skoruna XGBoost %98,49 ve en kötü F1 skoruna LightGBM %75,07 ile ulaştığı görülmektedir. GBTree, varsayılan durumda XGBoost algoritmasına kıyasla %2,34 fark ile daha düşük bir F1 skoruna sahiptir. Öznitelik seçimi durumunda varsayılan duruma kıyasla, F1 skorları için XGBoost algoritmasında %0,78 oranında bir düşüş gözlenirken, GBTree algoritmasında %0,52 ve LightGBM algoritmasında ise %3,57 oranlarında bir artış görülmektedir. Çapraz

doğrulama durumunda varsayılan duruma kıyasla F1 skorunda en fazla artışın %20,78 ile LightGBM algoritmasında olduğu görülmektedir. Çapraz doğrulama durumunda en iyi F1 skoru XGBoost algoritmasında %99,38 ile ulaşıldığı görülmektedir. Öznitelik seçimi ve çapraz doğrulama durumunun öznitelik seçimi durumuna kıyasla XGBoost algoritmasında %1,32 LightGBM algoritmasında %11,01 GBTree algoritmasında ise %1,52 oranlarında F1 skorlarında artış görülmektedir. Ancak, bu durumun çapraz doğrulamaya kıyasla XGBoost algoritmasında %0,30 LightGBM algoritmasında %6,20 GBTree algoritmasında ise %0,13 oranlarında F1 skorları için düşüş görülmektedir. XGBoost algoritmasının tüm durumlarda daha iyi F1 skoruna ulaştığı görülmektedir.




**Şekil 4.4.** Görev A'da sınıflandırma algoritmalarının değerlendirme skorları **A)** varsayılan **B)** öznitelik seçimi **C)** çapraz-doğrulama **D)** öznitelik seçimi ve çapraz-doğrulama

Görev B'de makine öğrenmesi algoritmalarında kullanılan parametreler için Çizelge 4.4'te varsayılan, öznitelik seçimi ve öznitelik seçimi ve çapraz doğrulama durumlarında

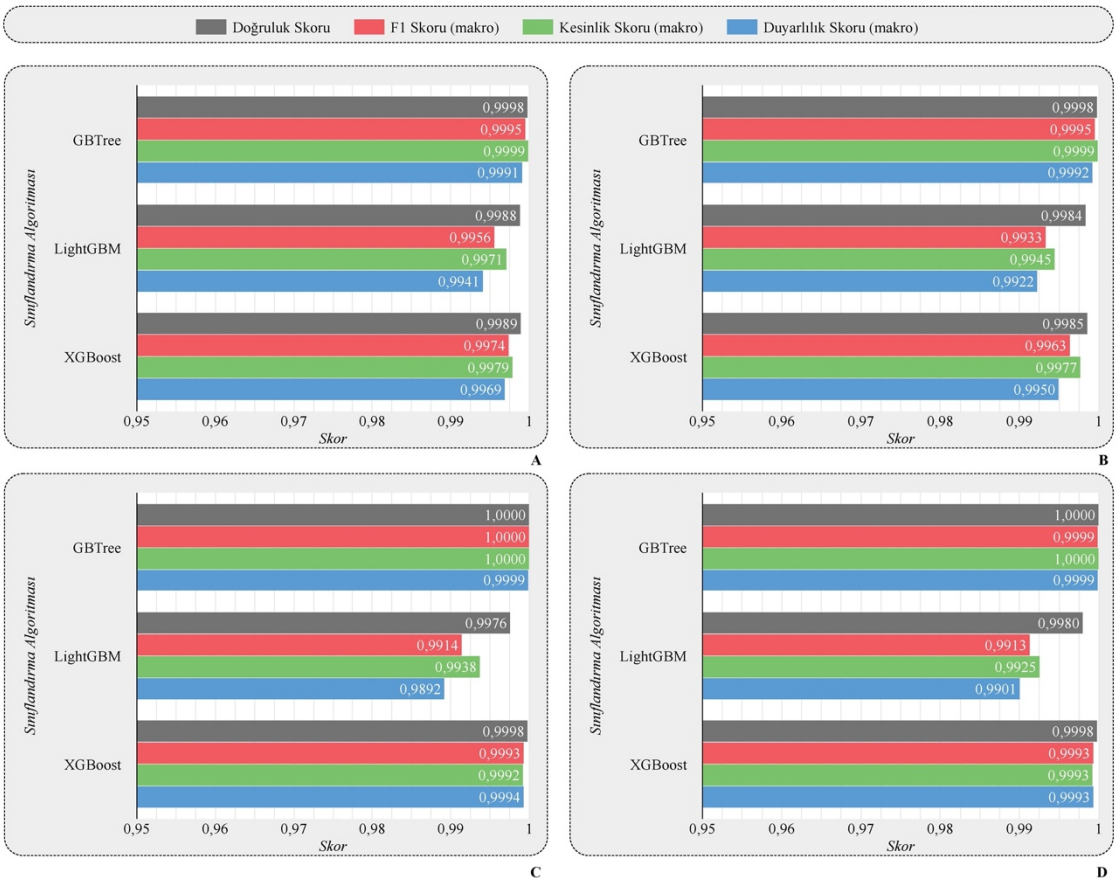
aldıkları değerler gösterildi. Çizelge 4.3'te olduğu gibi Çizelge 4.4'te de varsayılan ve öznitelik seçimi durumlarında parametreler aynı değerleri almaktadır. Bu sebeple ilgili çizelgede de her iki durum, varsayılan durumdaki değerler ile ifade edildi.

**Çizelge 4.4.** Görev B'nin sınıflandırma algoritmalarında kullanılan parametrelerin, varsayılan ve çapraz-doğrulama sonrasında belirlenen değerleri

| Parametre | GBTree | | | LightGBM | | | XGBoost | | |
|---|---|---|---|---|---|---|---|---|---|
| | Varsayılan | CV | FS ve CV | Varsayılan | CV | FS ve CV | Varsayılan | CV | FS ve CV |
| "num_iterations" | 100 | 50 | 50 | 100 | 50 | 50 | 100 | 50 | 50 |
| "learning_rate" | 0.1 | 0.05 | 0.05 | 0.1 | 0.15 | 0.15 | 0.3 | 0.25 | 0.25 |
| "max_depth" | 3 | 15 | 15 | 7 | 15 | 15 | 6 | 15 | 15 |
| "max_leaf_nodes" | 75 | 38 | 38 | 75 | 38 | 38 | 75 | 38 | 38 |
| "min_samples_leaf" | 20 | 40 | 40 | 20 | 20 | 20 | - | - | - |
| "min_child_weight" | - | - | - | - | - | - | 1 | 9 | 3 |
| "min_sum_hessian_in_leaf" | - | - | - | 0.001 | 0.001 | 0.001 | - | - | - |
| "subsample" | - | - | - | 1.0 | 0.6 | 0.6 | 1.0 | 1.0 | 1.0 |
| "colsample_bytree" | - | - | - | 1.0 | 1.0 | 1.0 | 1.0 | 0.6 | 0.6 |
| "lambda_L1" | - | - | - | 0 | 0 | 0 | - | - | - |
| "lambda_L2" | - | - | - | 0 | 15 | 1 | - | - | - |
| "lambda" | - | - | - | - | - | - | 1 | 1 | 1 |
| "alpha" | - | - | - | - | - | - | 0 | 0 | 0 |
| "gamma" | - | - | - | - | - | - | 0 | 0 | 0 |
| "bagging_freq" | - | - | - | 0 | 10 | 10 | - | - | - |

Görev B için sınıflandırma algoritmalarının dört durumdaki değerlendirme skorları Şekil 4.5'te gösterildi. Varsayılan durum için F1 skoru %99,95 ile en yüksek skora GBTree algoritması ulaştığı görülmektedir. GBTree algoritmasının F1 skorundan %0,21 fark ile XGBoost ve %0,39 fark ile de LightGBM algoritmaları takip etmektedir. Öznitelik seçimi durumu varsayılan durumu kıyasla, F1 skoru GBTree algoritmasında değişmediği ve en yüksek sonuca ulaştığı görülmektedir. Ancak, F1 skoru LightGBM algoritmasında %0,23 XGBoost algoritmasında ise %0,11 oranında düşüş olduğu gözlenmektedir. Çapraz-doğrulama durumu varsayılan duruma kıyasla, F1 skorunda GBTree algoritmasında %0,05 XGBoost algoritmasında %0,19 oranlarında artış gözlenirken, LightGBM algoritmasında ise %0,42 oranında düşüş görülmektedir. Öznitelik seçimi ve çapraz-doğrulama durumunun öznitelik seçimi durumuna kıyaslanmasıyla, F1 skoru XGBoost ve GBTree algoritmalarında sırasıyla %0,30 ve %0,04 oranında artış gözlenirken, LightGBM algoritmasında ise %0,20 oranında düşüş görülmektedir. Bu son durumun

çapraz-doğrulama durumuna kıyasla, F1 skoru XGBoost algoritmasında aynı kalmaktayken, GBTree ve LightGBM algoritmalarında %0,01 oranında düşüş görülmektedir.




**Şekil 4.5.** Görev B'de sınıflandırma algoritmalarının değerlendirme skorları **A)** varsayılan **B)** öznitelik seçimi **C)** çapraz-doğrulama **D)** öznitelik seçimi ve çapraz-doğrulama

Görev A ve B için deney kapsamında uygulanan dört durumun tüm değerlendirme skorlarının makro ve mikro sonuçlarının tamamı sırasıyla Çizelge 4.5 ve Çizelge 4.6'da ayrıntılı olarak gösterilmektedir. Çizelgelerde kalın punto ile gösterimi yapılan değerler, algoritmaların dört durum içerisinden ilgili değerlendirme skorunun en yüksek olduğu sonucu belirtmektedir.

Görev A için Çizelge 4.5'te tüm algoritmalar için çapraz-doğrulama durumunda genel olarak tüm değerlendirme skorlarında en iyi sonuca ulaşıldığı görülmektedir. Algoritmalar içerisinden iyi sonuç XGBoost algoritması ile elde edilmektedir.

**Çizelge 4.5.** Görev A'da sınıflandırma algoritmalarının tüm değerlendirme skorları

| Algoritma | Yöntem | | | Doğruluk Skoru | F1-Skoru | | Kesinlik Skoru | | Duyarlılık Skoru | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Varsayılan | CV | FS | Genel | Makro | Mikro | Makro | Mikro | Makro | Mikro |
| GBTree | ✓ | | | 0,9929 | 0,9675 | 0,9929 | 0,9704 | 0,9929 | 0,9663 | 0,9929 |
| | | ✓ | | **0,9986** | **0,9883** | **0,9986** | 0,9874 | **0,9986** | **0,9895** | **0,9986** |
| | | | ✓ | 0,9925 | 0,9727 | 0,9925 | 0,9817 | 0,9925 | 0,9655 | 0,9925 |
| | | ✓ | ✓ | 0,9975 | 0,9870 | 0,9975 | **0,9890** | 0,9975 | 0,9855 | 0,9975 |
| LightGBM | ✓ | | | 0,9640 | 0,7507 | 0,9640 | 0,7268 | 0,9640 | 0,7996 | 0,9640 |
| | | ✓ | | **0,9944** | **0,9585** | **0,9944** | **0,9621** | **0,9944** | **0,9552** | **0,9944** |
| | | | ✓ | 0,9711 | 0,7864 | 0,9711 | 0,7582 | 0,9711 | 0,8400 | 0,9711 |
| | | ✓ | ✓ | 0,9829 | 0,8965 | 0,9829 | 0,9192 | 0,9829 | 0,8799 | 0,9829 |
| XGBoost | ✓ | | | 0,9985 | 0,9849 | 0,9985 | 0,9872 | 0,9985 | 0,9830 | 0,9985 |
| | | ✓ | | **0,9994** | **0,9938** | **0,9994** | **0,9948** | **0,9994** | **0,9929** | **0,9994** |
| | | | ✓ | 0,9978 | 0,9771 | 0,9978 | 0,9805 | 0,9978 | 0,9745 | 0,9978 |
| | | ✓ | ✓ | 0,9991 | 0,9903 | 0,9991 | 0,9905 | 0,9991 | 0,9903 | 0,9991 |

Görev B'de en iyi skorlara GBTree, LightGBM ve XGBoost algoritmaları sırasıyla çapraz-doğrulama, varsayılan ve öznitelik seçimi ve çapraz doğrulama durumlarında %99'ın üzerinde en iyi skorlara ulaştıkları görülmektedir.

**Çizelge 4.6.** Görev B'de sınıflandırma algoritmalarının tüm değerlendirme skorları

| Algoritma | Yöntem | | | Doğruluk Skoru | F1-Skoru | | Kesinlik Skoru | | Duyarlılık Skoru | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Varsayılan | CV | FS | Genel | Makro | Mikro | Makro | Mikro | Makro | Mikro |
| GBTree | ✓ | | | 0,9998 | 0,9995 | 0,9998 | 0,9999 | 0,9998 | 0,9991 | 0,9998 |
| | | ✓ | | **1,0000** | **1,0000** | **1,0000** | **1,0000** | **1,0000** | **0,9999** | **1,0000** |
| | | | ✓ | 0,9998 | 0,9995 | 0,9998 | 0,9999 | 0,9998 | 0,9992 | 0,9998 |
| | | ✓ | ✓ | 1,0000 | 0,9999 | 1,0000 | 1,0000 | 1,0000 | 0,9999 | 1,0000 |
| LightGBM | ✓ | | | **0,9988** | **0,9956** | **0,9988** | **0,9971** | **0,9988** | **0,9941** | **0,9988** |
| | | ✓ | | 0,9976 | 0,9914 | 0,9976 | 0,9938 | 0,9976 | 0,9892 | 0,9976 |
| | | | ✓ | 0,9984 | 0,9933 | 0,9984 | 0,9945 | 0,9984 | 0,9922 | 0,9984 |
| | | ✓ | ✓ | 0,9980 | 0,9913 | 0,9980 | 0,9925 | 0,9980 | 0,9901 | 0,9980 |
| XGBoost | ✓ | | | 0,9989 | 0,9974 | 0,9989 | 0,9979 | 0,9989 | 0,9969 | 0,9989 |
| | | ✓ | | 0,9998 | 0,9993 | 0,9998 | 0,9992 | 0,9998 | **0,9994** | 0,9998 |
| | | | ✓ | 0,9985 | 0,9963 | 0,9985 | 0,9977 | 0,9985 | 0,9950 | 0,9985 |
| | | ✓ | ✓ | **0,9998** | **0,9993** | **0,9998** | **0,9993** | **0,9998** | 0,9993 | **0,9998** |

Görev A ve B için sınıflandırma algoritmalarının öznitelik seçimi ve çapraz-doğrulama durumunda öznitelik önemi en yüksek olan ilk 10 öznitelik, sırasıyla Çizelge 4.7 ve Çizelge 4.8'de gösterildi. İlgili çizelgelerde verilen öznitelik önemleri, bu tez çalışması kapsamında literatüre kazandırılan "pattern byte" özniteliğinin sınıflandırma aşamasındaki önemini göstermek amacı ile sunulmuştur. İlgili durumda toplamda 42 adet öznitelik bulunmaktadır. Öznitelikler içerisinden, bu çalışmada çıkarılan "pattern byte" özniteliğinin Görev A ve B'de tüm sınıflandırma algoritmalarında etkilerinin yüksek olduğu görülmektedir. Sınıflandırma algoritmalarında "pattern byte" özniteliğinin, öznitelik önem sırası değişiklik göstermektedir.

**Çizelge 4.7.** Görev A için sınıflandırma algoritmalarının, öznitelik seçimi ve çapraz-doğrulama durumunda öznitelik önemi en yüksek ilk 10 öznitelik

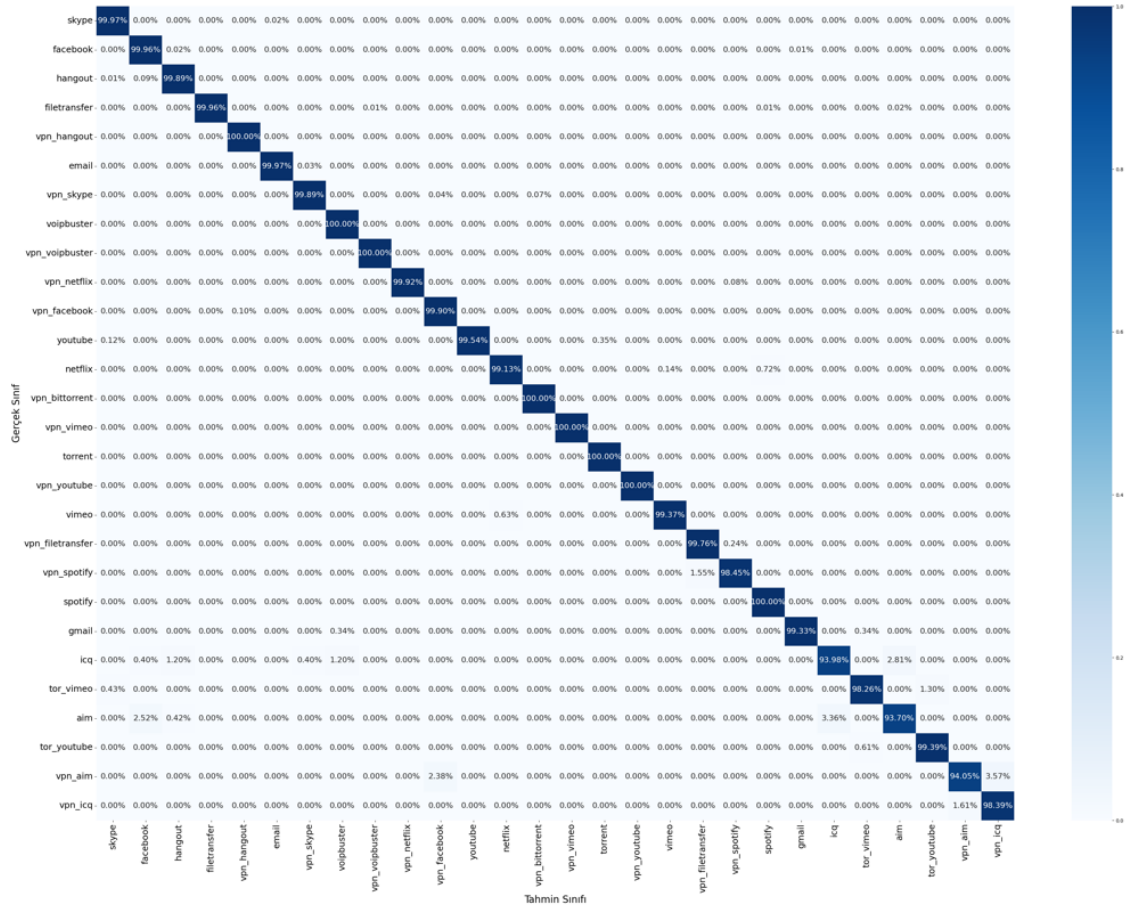
| GBTree | LightGBM | XGBoost |
|---|---|---|
| "src2dst_first_seen_ms" | "pattern_byte_5" | "pattern_byte_1" |
| "src2dst_last_seen_ms" | "bidirectional_first_seen_ms" | "pattern_byte_2" |
| "bidirectional_first_seen_ms" | "bidirectional_last_seen_ms" | "bidirectional_first_seen_ms" |
| "bidirectional_last_seen_ms" | "pattern_byte_1" | "src_port" |
| "bidirectional_max_ps" | "pattern_byte_7" | "pattern_byte_3" |
| "src2dst_packets" | "src2dst_first_seen_ms" | "bidirectional_last_seen_ms" |
| "dst2src_first_seen_ms" | "pattern_byte_2" | "pattern_byte_4" |
| "dst2src_last_seen_ms" | "src2dst_last_seen_ms" | "dst2src_first_seen_ms" |
| "pattern_byte_3" | "pattern_byte_3" | "pattern_byte_7" |
| "pattern_byte_6" | "pattern_byte_4" | "pattern_byte_5" |

**Çizelge 4.8.** Görev B için sınıflandırma algoritmalarının, öznitelik seçimi ve çapraz-doğrulama durumunda öznitelik önemi en yüksek ilk 10 öznitelik

| GBTree | LightGBM | XGBoost |
|---|---|---|
| "bidirectional_first_seen_ms" | "pattern_byte_1" | "pattern_byte_1" |
| "src2dst_first_seen_ms" | "pattern_byte_2" | "pattern_byte_2" |
| "bidirectional_last_seen_ms" | "pattern_byte_7" | "bidirectional_first_seen_ms" |
| "src2dst_last_seen_ms" | "pattern_byte_4" | "src_port" |
| "pattern_byte_5" | "src_port" | "pattern_byte_3" |
| "pattern_byte_3" | "pattern_byte_3" | "bidirectional_last_seen_ms" |
| "bidirectional_stdev_ps" | "dst2src_first_seen_ms" | "pattern_byte_4" |
| "pattern_byte_6" | "pattern_byte_5" | "dst2src_first_seen_ms" |
| "protocol" | "dst_port" | "pattern_byte_7" |
| "dst2src_first_seen_ms" | "bidirectional_mean_piat_ms" | "pattern_byte_5" |

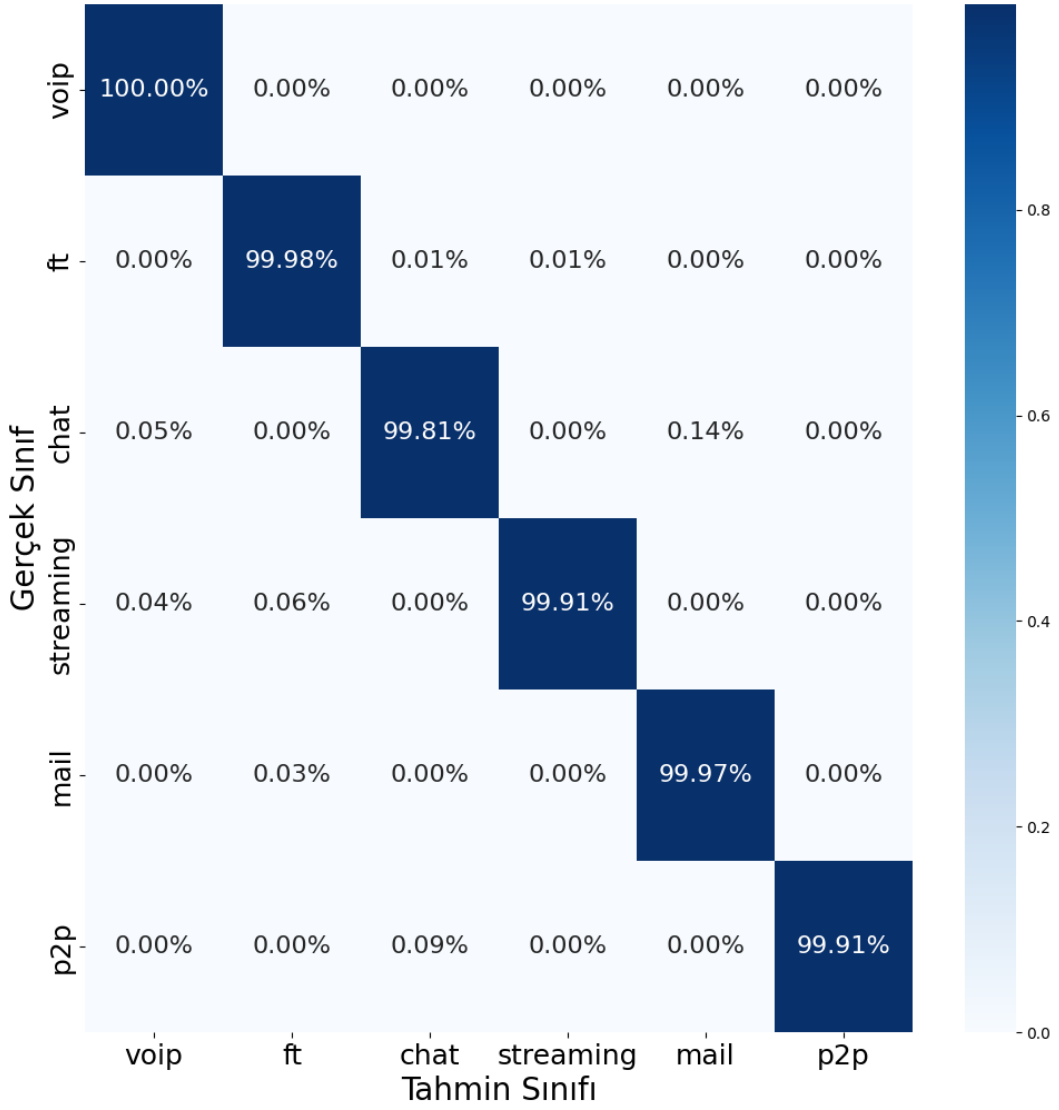
Görev A ve B'de LightGBM ve XGBoost algoritmalarında "pattern byte" özniteliğinin yük verisinin 1, 2, 3, 4, 5 ve 7 bayt konumları için öznitelik önemlerinin etkili olduğu görülmektedir. GBTree algoritmasında ise genellikle zaman damgalarına ilişkin özniteliklerin önem derecelerinin yüksek olduğu görülmektedir.

Görev A ve B için XGBoost algoritmasının öznitelik seçimi ve çapraz-doğrulama durumundaki karmaşıklık matrisleri sırasıyla Şekil 4.6 ve Şekil 4.7'de gösterildi. Karmaşıklık matrislerinde hedef sınıfların tahmin edilen sınıfları ile ilişkileri gösterilmektedir. Görev A'nın karmaşıklık matrisinde doğru pozitif oranı %99 küçük %93'ten büyük olan hedef sınıfların "vpn_spotify", "icq", "vpn_icq", "aim" ve "vpn_aim" sınıfları oldukları görülmektedir. Diğer hedef sınıflarının ise doğru pozitif oranlarının %99'dan büyük oldukları görülmektedir.




**Şekil 4.6.** Görev A'da XGBoost algoritmasının öznitelik seçimi ve çapraz-doğrulama durumundaki karmaşıklık matrisi

Görev B'nin karmaşıklık matrisinde "voip" hedef sınıfının doğru pozitif oranı %100 yanlış pozitif oranının ise toplamda yaklaşık %1 olduğu görülmektedir. En yüksek yanlış pozitif oranı yaklaşık %0,14 ile "mail" ve en düşük yanlış pozitif oranının yaklaşık %0 ile "p2p" hedef sınıfları olduğu görülmektedir.




**Şekil 4.7.** Görev B'de XGBoost algoritmasının öznitelik seçimi ve çapraz-doğrulama durumundaki karmaşıklık matrisi

Bu bölümde, Görev A ve B olarak tanımlanan sınıflandırma problemleri için önerilen sistemin uygulanması sonucunda elde edilen sonuç verildi ve önemli sonuçlar belirtildi.

# 5. TARTIŞMA ve SONUÇ

Bu çalışma ile şifreli ağ trafiğinin sınıflandırılması için uçtan-uca tüm sınıflandırma süreçlerini kapsayan bir çözüm önerilmiştir. Önerilen çözüm yöntemi ile şifreli ağ trafiği problemlerinden, uygulama ve uygulama türlerini sınıflandırma problemleri için bir deney gerçekleştirilmiştir. Yapılan deneyin değerlendirilmesi sonucunda performans ile sınıflandırma başarıları dikkate alındığında en kararlı ve en iyi sonuca XGBoost algoritması ile ulaşılmıştır. XGBoost algoritması, her iki sınıflandırma probleminde yaklaşık %99 oranında F1 skoru ile en iyi sonuca ulaşmıştır. LightGBM algoritmasının, hiper-parametre optimizasyonu sonrasında sınıflandırma başarısının %20'nin üzerinde bir performans artışı göstermesi ile parametre değerlerine oldukça hassas olduğu görülmüştür.

Draper-Gil ve diğerleri (2016), çalışmalarında hazırladıkları veri kümesinin, yapılan deneyde hedef sınıflar için veri dağılımı standart sapmasının oldukça yüksek olması ile dengesiz olduğu gösterilmiştir. Ancak ilgili veri kümesi, çalışmamızın hedefleri doğrultusunda popüler uygulamaları ve VPN ağ trafiklerini içermesi açısından en uygun veri kümesidir. Bu sebeple ağ trafiği sınıflandırmasında yapılacak çalışmalar için daha iyi ve uygun veri kümelerinin oluşturulmasına ihtiyaç olduğu düşünülmektedir.

Ağ akışı tabanlı ağ trafiği sınıflandırmalarında akış süresinin, makine öğrenmesi tekniklerinde sınıflandırma performansına etkisinin yüksek olduğu belirtilmektedir (Bozkır vd., 2022; Draper-Gil vd., 2016). Ancak, bu yöntemin uygulandığı birçok çalışma (Örneğin; Bu vd. 2020; Cherif vd. 2019; Dias vd. 2019), ağ akışı süresini belirtmediğinden, gerçekleştirildikleri deneylerin %100 tekrarlanabilir olmasını sağlayamamaktadır. Bu çalışmada ise deneyin tekrarlanabilir olmasına özen gösterilerek, önerilen sistem içerisinde MLFlow yapısı ile deneye ilişkin bütün bilgiler sunulmaktadır.

De Donato ve diğerleri (2014) tarafından geliştirilen ağ trafiği sınıflandırma platformunun ağ akışı toplama ve öznitelik çıkarımı yöntemlerinin doğruluk ve performans açılarından zayıf yönleri bulunmaktadır (Aouini ve Pekar, 2022). Önerilen

sistemde, literatürdeki zayıf yönleri bulunan yöntemlerden daha iyi olan NFStream aracının kullanılması ile güncel ve etkili bir deney gerçekleştirilmiştir.

Bu çalışma ile literatüre kazandırılan "pattern_byte" özniteliğinin, sınıflandırma algoritmaları üzerindeki öznitelik önemi derecelerinin yüksek olması sonucunda ayırt edici özelliğe sahip olduğu deney sonuçları ile gösterilmiştir. Öznitelik hesaplama sürecinde "pattern_byte" özniteliği de dahil olmak üzere, yüksek performansa sahip NFStream aracı ile gerçekleştirilmesi gerçek-zamanlı uygulamalarda da kullanılabileceğini göstermektedir.

Önerilen sistemin gerçek-zamanlı bir sınıflandırma senaryosunda, literatürde incelenen çalışmalardan farklı olarak veri hazırlama sürecinin performansı kanıtlanan Apache Spark çerçevesinde gerçekleştirilmesi ile uygulanabilir olduğu düşünülmektedir.

Sonuç olarak, şifreli ağ trafiği sınıflandırma problemlerine uygulanabilecek uçtan-uca tüm süreçleri barından bir sistem önerilmektedir. Önerilen sistemin başarısı açık erişime sahip literatürde sıklıkla kullanılan veri kümesi üzerinde yapılan deneyler ile gösterilmiştir. Veri kümesinde şifreli ağ trafiklerinin ve VPN kullanımını barındırması ile bu tür sınıflandırmayı zorlaştıran durumlardaki başarısı da gösterilmiştir. Önerilen sistemin performanslı ve uçtan-uca tüm süreçleri kapsaması ile gerçek dünya veri kümelerinde sıklıkla karşılaşılan değişiklik ve güncellemelere kolay uyum sağlayabileceği düşünülmektedir.

## KAYNAKLAR


Anderson, B., & McGrew, D. (2016). Identifying encrypted malware traffic with contextual flow data. *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*. https://doi.org/10.1145/2996758.2996768

Aouini, Z., & Pekar, A. (2022). NFStream A flexible network data analysis framework. *Computer Networks*, *204*, 108719. https://doi.org/10.1016/j.comnet.2021.108719

Arfeen, A., Ul Haq, K., & Yasir, S. M. (2020). Application layer classification of internet traffic using ensemble learning models. *International Journal of Network Management*. https://doi.org/10.1002/nem.2147

Arivudainambi, D., Varun Kumar, K. A., Sibi Chakkaravarthy, S., & Visu, P. (2019). Malware traffic classification using principal component analysis and artificial neural network for extreme surveillance. *Computer Communications*, *147*, 50-57. https://doi.org/10.1016/j.comcom.2019.08.003

Bozkır, R., Cicioğlu, M., Toğay, C., & Çalhan, A. (2022). Ağ Trafiğinin Akış Tabanlı Sınıflandırılmasında Akış Sürelerinin Makine Öğrenimi Algoritmalarına Etkisi. *European Journal of Science and Technology*. *https://doi.org/10.31590/ejosat.1112866*

Bu, Z., Zhou, B., Cheng, P., Zhang, K., & Ling, Z. (2020). Encrypted network traffic classification using deep and parallel network-in-network models. *IEEE Access*, *8*, 132950-132959. https://doi.org/10.1109/access.2020.3010637

Chen, M., Wang, X., He, M., Jin, L., Javeed, K., & Wang, X. (2020). A network traffic classification model based on metric learning. *CMC-computers Materials & Continua*, *64*(2), 941-959. https://doi.org/10.32604/cmc.2020.09802

Cherif, I. L., & Kortebi, A. (2019). On using extreme gradient boosting (XGBoost) machine learning algorithm for home network traffic classification. *2019 Wireless Days (WD)*. https://doi.org/10.1109/wd.2019.8734193

Cisco Systems. (2021). *Global 2021 Forecast Highlights*. https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf

Datta, J., Kataria, N., & Hubballi, N. (2015). Network traffic classification in encrypted environment: A case study of Google hangout. *2015 Twenty First National Conference on Communications (NCC)*. https://doi.org/10.1109/ncc.2015.7084879

De Donato, W., Pescape, A., & Dainotti, A. (2014). Traffic identification engine: An open platform for traffic classification. *IEEE Network*, *28*(2), 56-64. https://doi.org/10.1109/mnet.2014.6786614

Dias, K. L., Pongelupe, M. A., Caminhas, W. M., & De Errico, L. (2019). An innovative approach for real-time network traffic classification. *Computer Networks*, *158*, 143-157. https://doi.org/10.1016/j.comnet.2019.04.004

Dimou, P., Fajfer, J., Müller, N., Papadogiannaki, E., Rekleitis, E., & Střasák, F. (2020, April 23). ENCRYPTED TRAFFIC ANALYSIS. *European Union Agency for Cybersecurity (ENISA)*.

Draper-Gil, G., Lashkari, A. H., Mamun, M. S., & A. Ghorbani, A. (2016). Characterization of encrypted and VPN traffic using time-related features. *Proceedings of the 2nd International Conference on Information Systems Security and Privacy*. https://doi.org/10.5220/0005740704070414

Fernandes, G., Rodrigues, J. J., Carvalho, L. F., Al-Muhtadi, J. F., & Proença, M. L. (2018). A comprehensive survey on network anomaly detection. *Telecommunication Systems*, *70*(3), 447-489. https://doi.org/10.1007/s11235-018-0475-8

Finsterbusch, M., Richter, C., Rocha, E., Muller, J., & Hanssgen, K. (2014). A survey of payload-based traffic classification approaches. *IEEE Communications Surveys & Tutorials*, *16*(2), 1135-1156. https://doi.org/10.1109/surv.2013.100613.00161

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5). https://doi.org/10.1214/aos/1013203451

Google (2022). Web'de HTTPS şifrelemesi. Google Transparency Report. Retrieved May 2022, from https://transparencyreport.google.com/https

Gómez, S. E., Martínez, B. C., Sánchez-Esguevillas, A. J., & Hernández Callejo, L. (2017). Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal. *Computer Networks*, *127*, 68-80. https://doi.org/10.1016/j.comnet.2017.07.018

Hofstede, R., Celeda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., & Pras, A. (2014). Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX. *IEEE Communications Surveys & Tutorials*, *16*(4), 2037-2064. https://doi.org/10.1109/comst.2014.2321898

International Telecommunication Union. (2021). *Measuring digital development. Facts and figures 2021* (978-92-61-35401-5). https://www.itu.int/en/ITU-D/Statistics/Documents/facts/FactsFigures2021.pdf

Khalife, J., Hajjar, A., & Diaz-Verdejo, J. (2014). A multilevel taxonomy and requirements for an optimal traffic-classification model. *International Journal of Network Management*, *24*(2), 101-120. https://doi.org/10.1002/nem.1855

Lim, H. K., Kim, J. B., Kim, K., Hong, Y. G., & Han, Y. H. (2019). Payload-based traffic classification using multi-layer LSTM in software defined networks. *Applied Sciences*, *9*(12), 2550. https://doi.org/10.3390/app9122550

Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., & Lloret, J. (2017). Network traffic classifier with Convolutional and recurrent neural networks for Internet of things. *IEEE Access*, *5*, 18042-18050. https://doi.org/10.1109/access.2017.2747560

Lotfollahi, M., Jafari Siavoshani, M., Shirali Hossein Zade, R., & Saberian, M. (2019). Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, *24*(3), 1999-2012. https://doi.org/10.1007/s00500-019-04030-2

Nazari, Z., Noferesti, M., & Jalili, R. (2019). DSCA: An Inline and Adaptive Application Identification Approach in Encrypted Network Traffic. *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy - ICCSP '19*. https://doi.org/10.1145/3309074.3309102

Nguyen, T. T., & Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials*, *10*(4), 56-76. https://doi.org/10.1109/surv.2008.080406

Python Software Foundation. (2020). *Python Language Reference, version 3.9*. *https://www.python.org*

Ran, J., Chen, Y., & Li, S. (2018). Three-dimensional convolutional neural network based traffic classification for wireless communications. *2018 IEEE Global*

*Conference on Signal and Information Processing (GlobalSIP)*. https://doi.org/10.1109/globalsip.2018.8646659

Shafiq, M., Yu, X., Laghari, A. A., Yao, L., Karn, N. K., & Abdessamia, F. (2016). Network traffic classification techniques and comparative analysis using machine learning algorithms. *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. https://doi.org/10.1109/compcomm.2016.7925139

Shen, M., Liu, Y., Zhu, L., Xu, K., Du, X., & Guizani, N. (2020). Optimizing feature selection for efficient encrypted traffic classification: A systematic approach. *IEEE Network*, *34*(4), 20-27. https://doi.org/10.1109/mnet.011.1900366

Shi, H., Li, H., Zhang, D., Cheng, C., & Wu, W. (2017). Efficient and robust feature extraction and selection for traffic classification. *Computer Networks*, *119*, 1-16. https://doi.org/10.1016/j.comnet.2017.03.011

The PostgreSQL Global Development Group. (2021). *PostgreSQL Database Reference, version 14. https://www.postgresql.org*

Tong, V., Tran, H. A., Souihi, S., & Mellouk, A. (2018). A novel QUIC traffic classifier based on Convolutional neural networks. *2018 IEEE Global Communications Conference (GLOBECOM)*. https://doi.org/10.1109/glocom.2018.8647128

Vu, L., Thuy, H. V., Nguyen, Q. U., Ngoc, T. N., Nguyen, D. N., Hoang, D. T., & Dutkiewicz, E. (2018). Time series analysis for encrypted traffic classification: A deep learning approach. *2018 18th International Symposium on Communications and Information Technologies (ISCIT)*. https://doi.org/10.1109/iscit.2018.8587975

Wang, W., Zhu, M., Wang, J., Zeng, X., & Yang, Z. (2017). End-to-end encrypted traffic classification with one-dimensional convolution neural networks. *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. https://doi.org/10.1109/isi.2017.8004872

Yamansavascilar, B., Guvensan, M. A., Yavuz, A. G., & Karsligil, M. E. (2017). Application identification via network traffic classification. *2017 International Conference on Computing, Networking and Communications (ICNC)*. https://doi.org/10.1109/iccnc.2017.7876241

Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Zumar, C. (2018). Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Engineering Bulletin*, *41*(4), 39-45. http://sites.computer.org/debull/A18dec/p39.pdf

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I. (2016). Apache spark. *Communications of the ACM*, *59*(11), 56-65. https://doi.org/10.1145/2934664

Zeng, Y., Gu, H., Wei, W., & Guo, Y. (2019). Deep-Full-Range : A deep learning based network encrypted traffic classification and intrusion detection framework. *IEEE Access*, *7*, 45182-45190. https://doi.org/10.1109/access.2019.2908225

Zhang, J., Chen, X., Xiang, Y., Zhou, W., & Wu, J. (2015). Robust network traffic classification. *IEEE/ACM Transactions on Networking*, *23*(4), 1257-1270. https://doi.org/10.1109/tnet.2014.2320577

Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y., & Guan, Y. (2013). Network traffic classification using correlation information. *IEEE Transactions on Parallel and Distributed Systems*, *24*(1), 104-117. https://doi.org/10.1109/tpds.2012.98

Zhang, W., Wang, J., Chen, S., Qi, H., & Li, K. (2019). A framework for resource-aware online traffic classification using CNN. *Proceedings of the 14th International Conference on Future Internet Technologies*. https://doi.org/10.1145/3341188.3341195

Zhou, H., Wang, Y., Lei, X., & Liu, Y. (2017). A method of improved CNN traffic classification. *2017 13th International Conference on Computational Intelligence and Security (CIS)*. https://doi.org/10.1109/cis.2017.00046