Bilgisayar Bilimleri ve Mühendisliği Dergisi

Cilt-17
Sayı-2
2024

Makine Öğrenmesi Yöntemleri Kullanılarak Kötü Amaçlı Yazılım Sınıflandırması: CIC-MamMem-2022 Veri Kümesi Üzerinde bir Başarım Karşılaştırması

Malware Classification Using Machine Learning Methods: A Performance Benchmark on CIC-MamMem-2022 Dataset

Oğuzhan KIRLAR
Kocaeli Üniversitesi
Bilgisayar Mühendisliği Bölümü
Kocaeli, Türkiye
oguzhankirlar@gmail.com
ORCID: 0009-0006-2023-1457

Gamze PEKSÖZ AKIN
Kocaeli Üniversitesi
Bilgisayar Mühendisliği Bölümü
Kocaeli, Türkiye
gamzepeksoz@gmail.com
ORCID: 0009-0003-5239-1955

Meltem KURT PEHLİVANOĞLU
Kocaeli Üniversitesi
Bilgisayar Mühendisliği Bölümü
Kocaeli, Türkiye
meltem.kurt@kocaeli.edu.tr
ORCID:0000-0002-7581-9390

Öz

Zararlı yazılım veya kötü amaçlı yazılım; bilgisayar ve mobil cihazların işlevlerini bozmak, kritik bilgileri toplamak, özel bilgisayar sistemlerine erişim sağlamak ve istenmeyen reklamları göstermek amacı ile kullanılan yazılımdır. Kötü amaçlı yazılımların güvenlik ve antivirüs sistemlerinde tespit edilebilmesi ya da engellenmesi için makine öğrenmesi tabanlı saldırı tespit/önleme sistemleri kullanılmaktadır. Bu çalışmada CIC-MamMem-2022 veri kümesi üzerinde, makine öğrenmesi yöntemleriyle kötü amaçlı yazılımların sınıflandırılması amaçlanmıştır. Bu veri kümesi üzerinde zorlu bir problem olan on altı sınıf sınıflandırma için literatürde bilinen en iyi F1 ölçüsü, kesinlik, hassasiyet ve doğruluk değerleri sırasıyla %69,46, %70,94, %69,48 ve %69,48 iken; bu çalışmada özellikle on altı sınıf sınıflandırma problemi üzerine odaklanılmış ve literatürde bilinen en iyi sonuçlardan daha iyi sonuçlar elde edilmiştir. Yapılan deneysel çalışmalar sonucunda XGBoost ile F1 ölçüsü, tutturma, bulma ve doğruluk değerleri sırasıyla %75,53, %75,43, %75,65 ve %75,53 olarak elde edilmiştir.

Gönderme, düzeltme ve kabul tarihi: 24.06.2024 - 31.07.2024 – 09.08.2024
Makale türü: Araştırma

Anahtar sözcükler: *Zararlı Yazılım Sınıflandırma, Zararlı Yazılım Tespiti, Makine öğrenmesi, Saldırı Tespit Sistemi*

Abstract

Malware or malicious software is software used to disrupt the functioning of computers and mobile devices, collect critical information, gain access to private computer systems, and display unwanted advertisements. Machine learning-based intrusion detection/prevention systems are used to detect or block malware in security and antivirus systems. This study aims to classify malware using machine learning methods on the CIC-MamMem-2022 dataset. For the challenging problem of sixteen-class classification on this dataset, the best-known F1 score, precision, recall, and accuracy values in the literature are 69.46%, 70.94%, 69.48%, and 69.48%, respectively. In this study, a particular focus was placed on the sixteen-class classification problem, and better results than the best-known results in the literature were achieved. As a result of the experimental studies, the F1 score, precision, recall, and accuracy values obtained with XGBoost were 75.53%, 75.43%, 75.65%, and 75.53%, respectively.

Keywords: *Malware classification, Malware detection, Machine learning, Intrusion Detection System*

1. Giriş

Saldırı tespit ve saldırı önleme sistemleri, sistemlere yapılan saldırıları tespit etmeyi ve engellemeyi amaçlayan makine öğrenmesi tabanlı sistemlerdir. Özellikle pandemi döneminde uzaktan çalışma, uzaktan eğitim gibi internetin iş, eğitim ve sosyal amaçlarla daha yoğun kullanılmasıyla birlikte sistemlere ve cihazlara yapılan saldırılar da artış göstermiştir. Yapay zekanın ilerlemesi ve dijitalleşmenin her geçen gün her alanda artmasıyla saldırıların da artacağı öngörülmektedir. Bu nedenle zararlı yazılımların yüksek oranda tespit edilmesi ile ilgili çalışmaların yapılması büyük öneme sahiptir.

1.1 Motivasyon ve Katkı

Bu çalışmada, zararsız ve kötü amaçlı yazılımları içeren CIC-MalMem-2022 veri kümesi [1] üzerinde, yüksek başarımla kötü amaçlı yazılımları tespit eden makine öğrenmesi tabanlı yeni bir saldırı tespit sistemi geliştirilmesi amaçlanmıştır. Veri kümesi üzerinde farklı ön işleme ve özellik çıkarımı adımları gerçekleştirildikten sonra, Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), Gradient Boosting (GB), Deep Neural Network (DNN), K-Nearest Neighbors (KNN), Decision Tree (DT), XGBoost algoritmaları kullanılarak ikili ve çoklu sınıflandırma problemleri üzerinde çalışılmıştır.

İkili sınıflandırma en başarılı model DNN olup, bu model için doğruluk, F1-skor, kesinlik ve hassasiyet metrikleri sırasıyla; %100, %100, %100, %99,99 elde edilmiştir. Literatürde yer alan ikili sınıflandırma modelleri ile elde edilen sonuçlar değerlendirildiğinde bu değer literatürde yer alan çalışmalar gibi en yüksek başarıma sahiptir. Bu çalışmanın literatüre sağladığı önemli katkı on altı-sınıf sınıflandırma modelleri içindir. Özellikle alt kategori sınıflarına göre yapılan on altı-sınıf sınıflandırmada; en başarılı model XGBoost için doğruluk, F1-skor, kesinlik ve hassasiyet skorları sırasıyla; %75,53, %75,53, %75,43, %75,65 elde edilmiş olup, bu skorlar literatürde bilinen en iyi skorlardan (%69,46, %70,94, %69,48 ve %69,48) daha başarılıdır.

1.2 Organizasyon

Çalışmanın ikinci bölümünde literatürde CIC-MalMem-2022 veri kümesi üzerinde yapılan çalışmalar özetlenmiştir. Üçüncü bölümde ise bu çalışmada önerilen saldırı tespit modelleri detaylandırılmıştır. Dördüncü bölümde gerçekleştirilen deneyler kapsamlı olarak sunulmuştur. Son bölümde ise elde edilen bulgular tartışılarak ileriki çalışmalardan bahsedilmiştir.

2. İlgili Çalışmalar

Literatürde, CIC-MamMem-2022 veri kümesi üzerinde, makine öğrenmesi ve/veya derin öğrenme yöntemleriyle kötü amaçlı yazılım tespitini hedefleyen birçok çalışma yer almaktadır.

CIC-MalMem-2022 veri kümesi üzerinde yapılan bu çalışmalar kapsamlı olarak incelenmiş olup; Çizelge 1, Çizelge 2 ve Çizelge 3'de sırasıyla ikili, dört-sınıf, on altı-sınıf sınıflandırma yapan çalışmalarda kullanılan en başarılı model, elde edilen başarım oranları bilgileri ayrıntılı olarak sunulmuştur. Çizelge 1'de ikili sınıflandırma için verilen sonuçlar değerlendirildiğinde; en başarılı modellerin RobustCBL [3], Decision Tree [6] ve Random Forest [17] olduğu görülmektedir. Çizelge 2'de dört-sınıf sınıflandırma için verilen sonuçlar değerlendirildiğinde; XGBoost [4], modeli ile en yüksek F1-ölçüsü, tutturma, bulma değerleri sırasıyla %88,13, %88,12 ve %88,15 elde edilirken, en yüksek doğruluk değeri ise %89,74 değeriyle Decision Tree [8] algoritmasıyla elde edilmiştir. Çizelge 3'te on altı-sınıf sınıflandırma için verilen sonuçlar değerlendirildiğinde; ExtraTrees [21] modeli ile en yüksek F1-Ölçüsü, tutturma, bulma ve doğruluk değerlerinin sırasıyla %69,46, %70,94, %69,48, %69,48 olarak elde edildiği gözlemlenmiştir.

Çizelge-1: CIC-MalMem-2022 Veri kümesi üzerinde ikili sınıflandırma yapan ilgili çalışmalar

Yıl	Kaynak	Kullanılan En İyi Sınıflandırma Modeli	Başarım Ölçüleri			
			F-Ölçüsü (F1)	Tutturma	Bulma	Doğruluk
2024	[2]	KNN	%99,97	%99,97	%99,97	%99,97
2023	[3]	RobustCBL (Robust CNN-BiLSTM)	%**100,00**	**%100,00**	**%100,00**	%99,96
2023	[4]	XGBoost	%99,99	%99,99	%99,99	%99,99
2023	[5]	Logistic Regression	%99,99	%99,99	%99,99	%99,99
2023	[6]	Decision Tree	%**100,00**	**%100,00**	**%100,00**	%99,00
2023	[7]	Random Forest	%99,99	%99,99	%99,99	%99,98
2023	[8]	Decision Tree	%99,99	**%100,00**	%99,98	%99,99
2022	[9]	Logistic Regression	%99,97	%99,98	%99,97	%99,97
2022	[10]	Random Forest	%99,99	%99,99	%99,99	%99,98
2023	[11]	XGBoost	-	-	-	%99,99
2023	[12]	CNN	-	-	-	%99,80
2022	[13]	Logistic Regression	-	-	-	%99,97
2024	[14]	Random Forest	-	-	%99,99	%99,99
2023	[15]	OCC-PCA	%99,00	%99,00	%99,00	%99,40
2023	[16]	Recursive Feature Elimination (RFE)	-	-	-	%99,80
2022	[17]	Random Forest	%**100,00**	**%100,00**	%99,99	**%100,00**
2023	[18]	K-Means	-	-	-	%99,00
2024	[19]	EnsAdp_CIDS algorithm	-	-	-	%99,85
2024	[20]	CNN	%98,72	-	-	%98,82
2023	[21]	Extremely Randomized Trees Classifier (ExtraTrees)	%99,83	%99,78	%99,86	%99,82

2024	[22]	DNN	%99,70	%99,60	%99,99	%99,70
2022	[23]	Decision Tree	-	%99,97	-	%99,98
2023	[24]	SVM	-	-	-	%96,20
2023	[25]	LSTM	-	-	-	%97,69
2024	[26]	VolMemLyzer	%95,45	-	-	%91,25

Çizelge-2: CIC-MalMem-2022 Veri kümesi üzerinde dört-sınıf sınıflandırma yapan ilgili çalışmalar

Yıl	Kaynak	Kullanılan En İyi Sınıflandırma Modeli	Başarım Ölçüleri			
			F-Ölçüsü (F1)	Tutturma	Bulma	Doğruluk
2024	[2]	KNN	%82.24	%82,39	%82,21	%82,21
2023	[3]	RobustCBL (Robust CNN-BiLSTM)	%84,00	%85,00	%85,00	%84,56
2023	[4]	XGBoost	**%88,13**	**%88,12**	**%88,15**	%87,21
2023	[7]	Gradient Boosted Tree	-	-	-	%85,12
2023	[8]	Decision Tree	-	-	-	**%89,74**
2022	[10]	CNN	%75,13	%75,79	%75,18	%83,53
2023	[21]	Extremely Randomized Trees Classifier (ExtraTrees)	%83,02	%83,11	%83,05	%83,05

Çizelge-3: CIC-MalMem-2022 Veri kümesi üzerinde on altı-sınıf sınıflandırma yapan ilgili çalışmalar

Yıl	Kaynak	Kullanılan En İyi Sınıflandırma Modeli	Başarım Ölçüleri			
			F-Ölçüsü (F1)	Tutturma	Bulma	Doğruluk
2024	[2]	KNN	%67,32	%68,88	%66,93	%66,93
2023	[21]	Extremely Randomized Trees Classifier (ExtraTrees)	**%69,46**	**%70,94**	**%69,48**	**%69,48**

3. Önerilen Saldırı Tespit Modelleri

3.1 Veri Kümesi

Bu çalışmada kullanılan CIC-MamMem-2022 veri kümesi 29.298 zararsız, 29.298 kötü amaçlı olmak üzere toplam 58.596 farklı gözlem içermektedir. Her bir gözlem için 57 farklı öznitelik bilgisi verilmiştir. Veri kümesi Trojan Horse, Ransomware ve Spyware olmak üzere üç farklı kötü amaçlı yazılım kategorisi içermektedir. Her bir kategori kendi içerisinde farklı kötü amaçlı yazılım ailelerine ayrılmıştır. Trojan Horse kategorisi Zeus, Emotet, Refroso, scar, Reconyc alt yazılım ailelerinden oluşurken; Spyware kategorisi 180Solutions, Coolwebsearch, Gator, Transponder, TIBS ailelerinden oluşur. Ransomware kategorisi ise Conti, MAZE, Pysa, Ako, Shade alt ailelerini içerir. Veri kümesinde yer alan zararsız ve kötü amaçlı yazılım kategorisi ve ilgili ailelere ait gözlem sayıları ile öznitelik bilgileri sırasıyla Çizelge-4 ve Çizelge-5'de sunulmuştur.

Çizelge-4: CIC-MamMem-2022 veri kümesinde yer alan yazılım sınıf ve alt sınıf gözlem bilgileri

Yazılım Türü	Yazılım Kategorisi	Yazılım Ailesi	Gözlem Sayıları
Kötü Amaçlı Yazılım	Trojan Horse	Zeus	1950
		Emotet	1967
		Refroso	2000
		scar	2000
		Reconyc	1570
	Spyware	180Solutions	2000
		Coolwebseah	2000
		Gator	2200
		Transponder	2410
		TIBS	1410
	Ransomware	Conti	1988
		MAZE	1958
		Pysa	1717
		Ako	2000
		Shade	2128
Zararsız Yazılım	-	-	29298

Çizelge-5: CIC-MamMem-2022 Veri kümesinde yer alan öznitelikler

No	Öznitelik	No	Öznitelik
1	Category	30	malfind.protection
2	pslist.nproc	31	malfind.uniqueInjections
3	pslist.nppid	32	psxview.not_in_pslist
4	pslist.avg_threads	33	psxview.not_in_eprocess_pool
5	pslist.nprocs64bit	34	psxview.not_in_ethread_pool
6	pslist.avg_handlers	35	psxview.not_in_pspcid_list
7	dlllist.ndlls	36	psxview.not_in_csrss_handles
8	dlllist.avg_dlls_per_proc	37	psxview.not_in_session
9	handles.nhandles	38	psxview.not_in_deskthrd
10	handles.avg_handles_per_proc	39	psxview.not_in_pslist_false_avg
11	handles.nport	40	psxview.not_in_eprocess_pool_false_avg
12	handles.nfile	41	psxview.not_in_ethread_pool_false_avg
13	handles.nevent	42	psxview.not_in_pspcid_list_false_avg

14	handles.ndesktop	43	psxview.not_in_csrss_handles_false_avg
15	handles.nkey	44	psxview.not_in_session_false_avg
16	handles.nthread	45	psxview.not_in_deskthrd_false_avg
17	handles.ndirectory	46	modules.nmodules
18	handles.nsemaphore	47	svcscan.nservices
19	handles.ntimer	48	svcscan.kernel_drivers
20	handles.nsection	49	svcscan.fs_drivers
21	handles.nmutant	50	svcscan.process_services
22	ldrmodules.not_in_load	51	svcscan.shared_process_services
23	ldrmodules.not_in_init	52	svcscan.interactive_process_services
24	ldrmodules.not_in_mem	53	svcscan.nactive
25	ldrmodules.not_in_load_avg	54	callbacks.ncallbacks
26	ldrmodules.not_in_init_avg	55	callbacks.nanonymous
27	ldrmodules.not_in_mem_avg	56	callbacks.ngeneric
28	malfind.ninjections	57	Class
29	malfind.commitCharge		

3.2 Veri Önişleme

Çalışmada kullanılan CIC-MamMem-2022 veri kümesi toplamda 58.596 farklı gözlem ve 57 farklı öznitelikten oluşmaktadır. Veri kümesi üzerinde farklı ön işleme yöntemleri ile veri kümesini temsil eden öznitelikler 33'e, gözlem sayısı ise 58.062'ye indirilmiştir. Aşağıdaki alt başlıklarda önişleme adımları kapsamlı olarak verilmiştir.

3.2.1 Öznitelik Seçimi

Veri kümesi üzerinde öncelikle her bir öznitelik için gözlem değerleri aynı olan öznitelikler kontrol edilmiştir. *"pslist.nprocs64bit"*, *"handles.nport"*, *"svcscan.interactive_process_services"* özniteliklerinde yer alan gözlem değerlerinin tamamının aynı olduğu tespit edilmiştir. Bu üç farklı özniteliğin, model üzerinde bir etkisi olmayacağı düşünülerek veri kümesinden çıkarılmıştır.

Ayrıca veri kümesi üzerinde eksik değer kontrolü yapılmış, ancak herhangi bir eksik değer gözlemlenmemiştir.

Son olarak öznitelik değerleri birbiri ile tamamen aynı olan gözlem sayısı kontrolü yapılmış ve 534 gözlemin birbiriyle aynı olduğu tespit edilerek, bu gözlemler veri kümesinden çıkarılmıştır.

İkili sınıflandırma yapabilmek için "*Class*" özniteliğine "One hot encoding" işlemi uygulanmıştır. Bunun sonucunda da verinin zararlı ya da zararsız olma durumu boolean değerine indirgenmiştir.

Veri kümesi içerisinde anlam olarak birbirine benzeyen öznitelikler bulunduğu için bazılarının toplamı ya da ortalamasını almanın modellere daha fazla katkısı olduğu fark edilmiştir. Ayrı ayrı işlem sayısını almak yerine toplam işlem sayısını belirlemek için "*pslist.nproc*" ve "*pslist.nprocs64bit*" öznitelikleri toplanıp "*total_procs*" sayısı belirlendi. Ancak "*pslist.nprocs64bit*" değeri tüm verilerde aynı olduğu ve veri kümesinden kaldırıldığı için "*pslist.nproc*" özniteliğinin ismi "*total_procs*" olarak değerlendirilmiştir.

Ortalama iş parçacığı ve işleyici sayısının toplamını belirlemek için "*pslist.avg_threads*" ve "*pslist.avg_handlers*" öznitelikleri toplanarak "*avg_thread_handlers*" özniteliği oluşturulmuştur.

Toplam handle sayısını belirlemek için "*handles.nport*", "*handles.nfile*", "*handles.nevent*", "*handles.ndesktop*", "*handles.nkey*", "*handles.nthread*", *handles.ndirectory, "handles.nsemaphore", "handles.ntimer", "handles.nsection", "handles.nmutant"* öznitelikleri toplanarak ve "*total_handles*" özniteliği oluşturulmuştur. Ancak "*handles.nport*" özniteliği için veri kümesindeki tüm değerler aynı olduğundan bu öznitelik kaldırıldığından ve bu toplama bu öznitelik dahil edilmemiştir.

Ayrıca ortalama yüklenmeyen modül hesaplaması yapılarak, "*ldrmodules.not_in_load*", "*ldrmodules.not_in_init*", "*ldrmodules.not_in_mem*" özniteliklerinin ortalaması alınıp "*avg_not_loaded_modules*" özniteliği oluşturulmuştur.

Son olarak, toplam görünmeyen işlem sayısı hesaplanarak; *"psxview.not_in_pslist", "psxview.not_in_eprocess_pool", "psxview.not_in_ethread_pool", "psxview.not_in_pspcid_list", "psxview.not_in_csrss_handles", "psxview.not_in_session", "psxview.not_in_deskthrd"* öznitelikleri toplanarak "*total_hidden_processes*" özniteliği oluşturulmuştur.

Yukarıda verilen hesaplamalarda kullanılan öznitelikler veri kümesinden çıkarılarak, yeni oluşturulan öznitelikler veri kümesine eklenmiştir. Ayrıca çoklu sınıflandırmada kullanılmak üzere "*Category*" özniteliğinden "*category_name*" ve "*subcategory_name*" isimli iki yeni öznitelik oluşturulmuştur. Bu iki öznitelik dört sınıf ve onaltı sınıf sınıflandırmada kullanılmıştır, sınıf özniteliğini temsil ettiğinden model eğitim özniteliklerine dahil edilmemiştir.

Yapılan işlemler sonucunda veri kümesinin son hali Çizelge 6'da sunulmuştur.

Çizelge-6: Önişleme ve öznitelik mühendisliği uygulandıktan sonra veri kümesinde yer alan öznitelikler

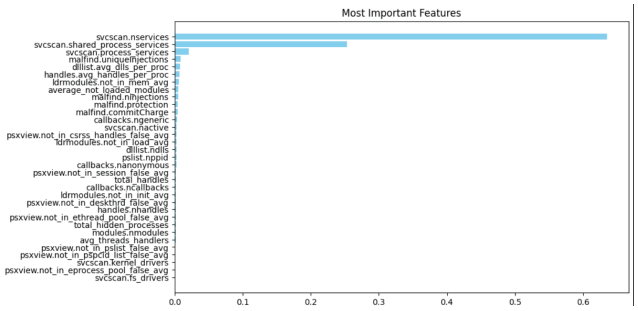
No	Öznitelik	No	Öznitelik
1	pslist.nppid	18	psxview.not_in_session_false_avg
2	dlllist.ndlls	19	psxview.not_in_deskthrd_false_avg
3	dlllist.avg_dlls_per_proc	20	modules.nmodules
4	handles.nhandles	21	svcscan.nservices
5	handles.avg_handles_per_proc	22	svcscan.kernel_drivers
6	ldrmodules.not_in_load_avg	23	svcscan.fs_drivers
7	ldrmodules.not_in_init_avg	24	svcscan.process_services
8	ldrmodules.not_in_mem_avg	25	svcscan.shared_process_services
9	malfind.ninjections	26	svcscan.nactive
10	malfind.commitCharge	27	callbacks.ncallbacks
11	malfind.protection	28	callbacks.nanonymous

12	malfind.uniqueInjections	29	callbacks.ngeneric
13	psxview.not_in_pslist_false_avg	30	avg_threads_handlers
14	psxview.not_in_eprocess_pool_false_avg	31	total_handles
15	psxview.not_in_ethread_pool_false_avg	32	average_not_loaded_modules
16	psxview.not_in_pspcid_list_false_avg	33	total_hidden_processes
17	psxview.not_in_csrss_handles_false_avg		

3.2.2. Özellik/Öznitelik Öneminin Uygulanması

Dengesiz ve dengeli veri kümeleri üzerinde en ayırt edici özniteliklerin belirlenmesi için öznitelik önemi kullanılmıştır. En önemli öznitelikler çıkarılırken XGBoost modeli üzerinde 'feature_importance_' yöntemi kullanılmış olup, önem grafiği Şekil 1'de sunulmuştur. Özniteliklerin önem oran grafiğine göre ilk 20 öznitelik çoklu sınıflandırma (dörtlü ve on altılı sınıflandırma) için modellerde kullanılmıştır. Dengesiz veri kümesi üzerinde kullanılan 20 öznitelik Çizelge 7'de sunulmuştur.

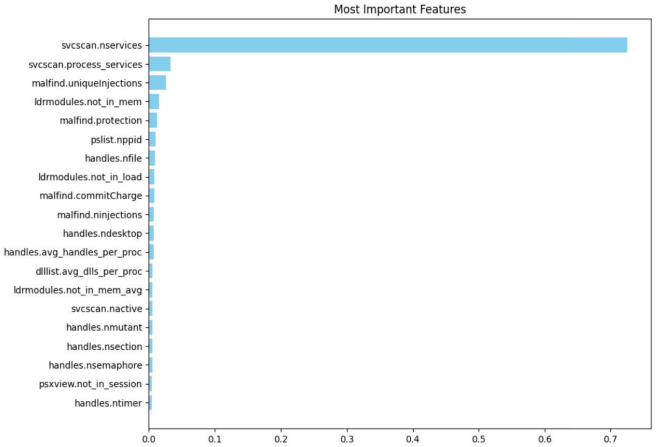
Dengeli veri kümesi için elde edilen önem grafiği ve özniteliklerin isimleri sırasıyla Şekil 2 ve Çizelge 8'de verilmiştir.



Şekil-1: Dengesiz veri kümesi üzerinde en önemli özniteliklerin belirlenmesi

Çizelge-7: Dengesiz veri kümesi üzerinde model eğitiminde kullanılan en önemli 20 öznitelik

No	Öznitelik	No	Öznitelik
1	svcscan.nservices	11	malfind.commitCharge
2	svcscan.shared_process_services	12	callbacks.ngeneric
3	svcscan.process_services	13	svcscan.nactive
4	malfind.uniqueInjections	14	psxview.not_in_csrss_handles_false_avg
5	dlllist.avg_dlls_per_proc	15	ldrmodules.not_in_load_avg
6	handles.avg_handles_per_proc	16	dlllist.ndlls
7	ldrmodules.not_in_mem_avg	17	pslist.nppid
8	average_not_loaded_modules	18	callbacks.nanonymous
9	malfind.ninjections	19	psxview.not_in_session_false_avg
10	malfind.protection	20	total_handles



Şekil-2: Dengeli veri kümesi için en önemli özniteliklerin belirlenmesi

Çizelge-8: Dengeli veri kümesi üzerinde model eğitiminde kullanılan en önemli 20 öznitelik

No	Öznitelik	No	Öznitelik
1	svcscan.nservices	11	handles.ndesktop
2	svcscan.process_services	12	handles.avg_handles_per_proc
3	malfind.uniqueInjections	13	dlllist.avg_dlls_per_proc
4	ldrmodules.not_in_mem	14	ldrmodules.not_in_mem_avg
5	malfind.protection	15	svcscan.nactive
6	pslist.nppid	16	handles.nmutant
7	handles.nfile	17	handles.nsection
8	ldrmodules.not_in_load	18	handles.nsemaphore
9	malfind.commitCharge	19	psxview.not_in_session
10	malfind.ninjections	20	handles_ntimer

3.2.3. Hiper Parametre Ayarlaması

Daha etkili ve başarılı sonuçlar elde etmek için hiper parametre ayarlaması yapılmıştır. Bu işlemde GridSearchCV fonksiyonundan yararlanılmıştır. Bu fonksiyon, kendisine verilen hiper parametre değerlerinin tüm olasılıklarını deneyerek, çapraz doğrulama ile en yüksek başarıyı sağlayan parametre kombinasyonlarını belirler.

4. Deneysel Sonuçlar

Çalışma kapsamında veri kümesi üzerinde ikili sınıflandırma, dört-sınıf sınıflandırma ve on altı-sınıf sınıflandırma için modeller eğitilmiş ve test edilmiştir. Tüm sınıflandırma problemlerinde önişleme adımlarından ve öznitelik seçimi sonucundan oluşan veriler kullanılmıştır. Modeller üzerinde hiper parametre ayarlaması yapılarak başarım oranları arttırılmıştır.

Sınıflandırma başarımları değerlendirilirken; 58,062 gözlemin %60'ı eğitim, %20'si test verisi, %20'si ise validasyon verisi olarak ayrılmıştır.

Her bir sınıflandırma için gerçeklenen modellerin performansını değerlendirmek ve genelleme yeteneğini ölçmek amaçlı 5-katmanlı (fold) çapraz doğrulama yöntemi kullanılmıştır.

Bunun yanı sıra özellikle dört-sınıf ve on altı-sınıf sınıflandırma probleminde oluşacak dengesiz veri problemini çözmek amaçlı "zararsız yazılım" sınıfından (Bknz. Çizelge 4) diğer kötü amaçlı yazılımların ortalaması olacak şekilde rastgele gözlem seçilerek dengeli veri kümeleri oluşturulmuştur. Bu sayede dengesiz veri problemi çözülmüştür. Dengeli veri kümesi kullanarak kurulan modellerde XGBoost modeli üzerinde 'feature_importance_' uygulanmış önem sırasına göre ilk 20 öznitelik modellerde kullanılmıştır.

Dört-sınıf sınıflandırmada için oluşturulan dengeli veri kümesi; Çizelge 4'de ayrıntılı verilen 3 sınıfa ait 28.831 adet kötü amaçlı yazılım ile 9610 adet zararsız yazılım sınıfı olmak üzere, toplam 38.441 gözlem içerir.

Onaltı-sınıf sınıflandırma için oluşturulan dengeli veri kümesi; Çizelge 4'de ayrıntılı verilen 15 sınıfa ait 28.831 adet kötü amaçlı yazılım ile 1.949 adet zararsız yazılım sınıfı olmak üzere, toplam 30.780 gözlem içerir.

İki-sınıf sınıflandırmada dengesiz veri problemi oluşmadığından bu deneyler için gözlem sayısında herhangi bir değişiklik yapılmamış olup Çizelge 4'de verilen sayıda gözlem kullanılmıştır.

Yukarıda verilen bilgiler doğrultusunda aşağıdaki alt başlıklarda ikili, dörtlü ve onaltılı sınıflandırma modelleri için sınıflandırma modelleri kurulmuş ve her bir sınıflandırma problemi için elde edilen sonuçlar detaylandırılmıştır.

4.1. İkili Sınıflandırma

İkili sınıflandırma için "one hot encoding" ile oluşturulan öznitelik ("Class" özniteliği kullanılarak boolean değerinde zararlı ve zararsız özelliğine temsilen) sınıflandırma için kullanılmıştır. Veri kümesinde yer alan gözlemler zararlı (toplam 28.831 gözlem) ve zararsız (toplam 29.298 gözlem) olmak üzere iki sınıfa ayrılmıştır. İkili sınıflandırma için LR, RF, SVM, GB, DNN algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur. Yapılan deneysel sonuçlar Çizelge 9'da sunulmuş olup en başarılı model DNN olmuştur. Literatürde yer alan ikili sınıflandırma ile ilgili çalışmalar (Bknz. Çizelge 1) ile elde edilen sonuçlar değerlendirildiğinde en başarılı metriklerin elde edildiği açıktır.

Çizelge-9: İkili Sınıflandırma İçin Modellerin Başarımı

Algoritma	Hiperparametre	Kullanılan Öznitelik Sayısı	Başarım Ölçüleri			
			F-Ölçüsü (F1)	Tutturma	Bulma	Doğruluk
LR	max_iter=1000	33	%99,62	%99,69	%99,69	%99,69
RF	n_estimators=100, random_state=42	33	%99,99	%99,98	**%100,00**	%99,99
SVM	kernel='linear', random_state=42	33	%99,82	%99,93	%99,72	%99,82
GB	n_estimators=100, learning_rate=1.0, max_depth=1, random_state=42	33	%99,97	%99,98	%99,96	%99,97
DNN	epochs=20, batch_size=32	33	**%100,00**	**%100,00**	%99,99	**%100,00**

4.2. Dört Sınıf Sınıflandırma

Zararsız (Benign), Trojan Horse, Ransomware ve Spyware sınıflandırılması için öznitelik önem sırası dikkate alınarak, en önemli 20 öznitelik kullanılmıştır. Yapılan deneyler sonucunda dört-sınıf sınıflandırma için DT, XGBoost, RF, KNN modelleri 5-katmanlı çapraz doğrulama kullanılarak elde edilen başarım metrikleri Çizelge 10'da verilmiştir. Çizelgeden de görüleceği gibi dört-sınıf sınıflandırma için en başarılı model XGBoost olup, literatürde dört-sınıf sınıflandırma içeren çalışmalar [4,8] ile karşılaştırıldığında, başarı oranı daha düşük kalmıştır.

Literatürde dört-sınıf sınıflandırma problemi üzerine çalışan çalışmalarda [2-4, 7, 8, 21] veri dengesizliği problemi ele alınmamıştır. Dengesiz veri problemini çözmek amaçlı oluşturulan dengeli veri kümesi üzerinde elde edilen dört-sınıf sınıflandırma sonuçları ise Çizelge 11'de verilmiştir. Çizelgeden de görüleceği gibi dengeli veri kümesi üzerinde modellerin sınıflandırma başarımları düşmüştür, ancak modeller arasında en başarılı algoritma XGBoost'tur.

Çizelge-10: Dört Sınıf Sınıflandırma İçin Modellerin Başarımı

Algoritma	Hiperparametre	Kullanılan Öznitelik Sayısı	Başarım Ölçüleri			
			F-Ölçüsü (F1)	Tutturma	Bulma	Doğruluk
DT	'criterion': ['gini', 'log_loss'], 'splitter': ['best', 'random'], 'min_samples_leaf': [100, 200, 300], 'max_depth': [3, 5, 8]	20	%77,83	%77,83	%77,83	%77,83
XGBoost	'objective': ['binary:logistic', 'multi:softmax'], 'n_estimators': [100, 200, 300], 'max_depth': [3, 5, 8], 'tree_method': ['hist']	20	**%85,64**	**%85,87**	**%85,42**	**%85,84**
RF	'n_estimators': [100, 200, 300], 'max_depth': [3, 5, 8, 10], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False]	20	%81,60	%81,58	%81,62	%81,75
KNN	'n_neighbors': [3, 5, 7, 9, 11], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan']	20	%80,72	%80,85	%80,60	%80,72

Çizelge-11: Dengeli Veri Kümesi Üzerinde Dört Sınıf Sınıflandırma İçin Modellerin Başarımı

Algoritma	Hiperparametre	Kullanılan Öznitelik Sayısı	Başarım Ölçüleri			
			F-Ölçüsü (F1)	Tutturma	Bulma	Doğruluk
DT	'criterion': ['gini', 'log_loss'], 'splitter': ['best', 'random'], 'min_samples_leaf': [100, 200, 300], 'max_depth': [3, 5, 8]	20	%72,61	%71,46	%73,81	%71,46
XGBoost	'objective': ['binary:logistic', 'multi:softmax'], 'n_estimators': [100, 200, 300], 'max_depth': [3, 5, 8], 'tree_method': ['hist']	20	**%78,00**	**%77,98**	**%78,04**	**%77,98**
RF	'n_estimators': [100, 200, 300], 'max_depth': [3, 5, 8, 10], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False]	20	%74,73	%74,66	%74,81	%74,91
KNN	'n_neighbors': [3, 5, 7, 9, 11], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan']	20	%73,50	%73,64	%73,38	%73,22

4.3. On altı Sınıf Sınıflandırma

Çalışmada ayrıca veri kümesinde zararlı yazılımlara ait alt kategoriler kullanılarak on altı-sınıf sınıflandırma yapılmıştır. Bu sınıflandırma için dört-sınıf sınıflandırmadaki gibi en önemli 20 öznitelik kullanılarak DT, XGBoost, RF, KNN modelleri test edilmiştir. Yapılan deneyler sonucunda onaltı-sınıf sınıflandırma ile elde edilen en iyi F1-skor, kesinlik, hassasiyet ve doğruluk skorları (sırasıyla %75,53, %75,43, %75,65, %75,53) Çizelge 12'de verilmiştir. Elde edilen deneysel sonuçlar değerlendirildiğinde en başarılı algoritma XGBoost olurken, elde edilen başarım değerleri literatürdeki en iyi sonuçlardır (Bknz. Çizelge 3).

Literatürde on altı-sınıf sınıflandırma problemi üzerine çalışan çalışmalarda [2,21] veri dengesizliği problemi ele alınmamıştır. Bu çalışmada dengesiz veri problemini çözmek amaçlı oluşturulan dengeli veri kümesi üzerinde elde edilen on altı-sınıf sınıflandırma sonuçları ise Çizelge 13'de verilmiştir. Çizelgede verilen sonuçlar değerlendirildiğinde modellerin başarımlarının önemli derecede düştüğü gözlemlenmiştir. Alt sınıflar birbirleri arasında benzerlik gösterdiğinden modellerin sınıfları ayırması daha da zorlaşmıştır.

Çizelge-12: On altı Sınıf Sınıflandırma için Modellerin Başarımı

Algoritma	Hiperparametre	Kullanılan Öznitelik Sayısı	Başarım Ölçüleri			
			F-Ölçüsü (F1)	Tutturma	Bulma	Doğruluk
DT	'criterion': ['gini', 'log_loss'], 'splitter': ['best', 'random'], 'min_samples_leaf': [100, 200, 300], 'max_depth': [3, 5, 8]	20	%62,28	%62,28	%62,28	%62,28
XGBoost	'objective': ['binary:logistic', 'multi:softmax'], 'n_estimators': [100, 200, 300], 'max_depth': [3, 5, 8], 'tree_method': ['hist']	20	**%75,53**	**%75,43**	**%75,65**	**%75,53**
RF	'n_estimators': [100, 200, 300], 'max_depth': [3, 5, 8, 10], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False]	20	%70,72	%72,12	%69,38	%69,42
KNN	'n_neighbors': [3, 5, 7, 9, 11], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan']	20	%67,49	%67,44	%67,70	%67,70

Çizelge-13: Dengeli Veri Kümesi Üzerinde On altı Sınıf Sınıflandırma için Modellerin Başarımı

Algoritma	Hiperparametre	Kullanılan Öznitelik Sayısı	Başarım Ölçüleri			
			F-Ölçüsü (F1)	Tutturma	Bulma	Doğruluk
DT	'criterion': ['gini', 'log_loss'], 'splitter': ['best', 'random'], 'min_samples_leaf': [100, 200, 300], 'max_depth': [3, 5, 8]	20	%55,64	%55,64	%55,64	%55,64
XGBoost	'objective': ['binary:logistic', 'multi:softmax'], 'n_estimators': [100, 200, 300], 'max_depth': [3, 5, 8], 'tree_method': ['hist']	20	**%67,60**	**%67,67**	**%67,52**	**%67,75**
RF	'n_estimators': [100, 200, 300], 'max_depth': [3, 5, 8, 10], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False]	20	%64,73	%64,62	%64,84	%64,80
KNN	'n_neighbors': [3, 5, 7, 9, 11], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan']	20	%60,38	%60,26	%60,50	%60,48

Bu çalışma kapsamında yapılan deneysel sonuçlar literatürdeki çalışmalar ile ikili ve çoklu sınıflandırma başarımları açısından karşılaştırıldığında, diğer çalışmalara kıyasla önemli başarılar elde edildiği açıktır. Özellikle İkili sınıflandırma için literatürdeki en yüksek başarımlar elde edilirken, on altı-sınıf sınıflandırma ise; literatürde en iyi sonuçları elde eden çalışmaya [21] oranla çok daha yüksek başarımlar sağlanmıştır. [21]'de verilen çalışmada F1-skor, kesinlik, hassasiyet ve doğruluk skorları sırasıyla %69,46, %70,94, %69,48, %69,48 elde edilirken, bu çalışmada aynı skorlar XGBoost algoritmasıyla sırasıyla %75,53, %75,43, %75,65, %75,53 olarak elde edilmiştir.

Bu çalışmada ayrıca literatürdeki diğer çalışmalardan farklı olarak dengesiz veri problemini çözmek için 'zararsız yazılım' örneklerinin sayısı diğer sınıfların ortalaması olacak şekilde azaltılmıştır. Bu da problemi daha zorlu hale getirmiştir. Çalışmada bu modellerin başarımlarının arttırılması ileriki çalışma olarak bırakılmıştır.

Ayrıca bu çalışmada yapılan tüm deneysel çalışmaları içeren kaynak kodları, GitHub üzerinde [27] erişime açık olarak paylaşılmıştır.

5. Sonuç ve İleriki Çalışmalar

Bu çalışmada CIC-MalMeM-2022 veri kümesi kullanılarak zararlı ve zararsız olarak ikili sınıflandırma, zararlı yazılımların kategorileri kullanılarak dört-sınıf sınıflandırma ve alt kategorileri kullanılarak on altı-sınıf sınıflandırma probleminin çözülmesi üzerine çalışılmıştır. Çalışma kapsamında LR, RF, SVM, DT, GB, XGBoost, DNN ve KNN modelleri kullanılarak modeller oluşturulmuştur.

Deneysel sonuçlar incelendiğinde en önemli özniteliklerin belirlenmesi ve hiper parametre ayarlaması yapılarak başarım oranının her modelde arttığı gözlemlenmiştir.

Literatürde yer alan diğer çalışmalar incelendiğinde; veri CIC-MalMeM-2022 kümesi özelinde, özellikle on altı-sınıf sınıflandırma için en yüksek başarımlar elde edilmiştir.

İleriki çalışmalarda özellikle dört-sınıf ve on altı-sınıf sınıflandırma için dengeli veri kümesi üzerinde modellerin başarımının arttırılması amaçlı farklı veri artırma tekniklerinin kullanılması, farklı modellerin denenmesi gibi çalışmalar hedeflenmektedir.

Kaynakça

[1] Carrier, T., Victor, P., Tekeoglu, A., & Lashkari, A. H. (2022, February). Detecting Obfuscated Malware using Memory Feature Engineering. In Icissp (pp. 177-188).

[2] Abualhaj, M., Abu-Shareha, A., Shambour, Q., Alsaaidah, A., Al-Khatib, S., & Anbar, M. (2024). Customized K-nearest neighbors' algorithm for malware detection. International Journal of Data and Network Science, 8(1), 431-438.

[3] Shafin, S. S., Karmakar, G., & Mareels, I. (2023). Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications. Sensors, 23(11), 5348.

[4] Hasan, S. R., & Dhakal, A. (2023, December). Obfuscated Malware Detection: Investigating Real-World Scenarios Through Memory Analysis. In 2023 IEEE International Conference on Telecommunications and Photonics (ICTP) (pp. 01-05). IEEE.

[5] Jiang, Q., Zhao, X., & Huang, K. (2011, June). A feature selection method for malware detection. In 2011 IEEE International Conference on Information and Automation (pp. 890-895). IEEE.

[6] Smith, D., Khorsandroo, S., & Roy, K. (2023, February). Supervised and unsupervised learning techniques utilizing malware datasets. In 2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC) (pp. 1-7). IEEE.

[7] Benkerroum, S., & Chougdali, K. (2023, December). Enhancing Forensic Analysis Using a Machine Learning-based Approach. In 2023 6th International Conference on Advanced Communication Technologies and Networking (CommNet) (pp. 1-6). IEEE.

[8] Balasubramanian, K. M., Vasudevan, S. V., Thangavel, S. K., Kumar, G., Srinivasan, K., Tibrewal, A., & Vajipayajula, S. (2023, July). Obfuscated Malware detection using Machine Learning models. In 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT) (pp. 1-8). IEEE.

[9] Dener, M., Ok, G., & Orman, A. (2022). Malware detection using memory analysis data in big data environment. Applied Sciences, 12(17), 8604.

[10] Mezina, A., & Burget, R. (2022, October). Obfuscated malware detection using dilated convolutional network. In 2022 14th international congress on ultra modern telecommunications and control systems and workshops (ICUMT) (pp. 110-115). IEEE

[11] Talukder, M. A., Hasan, K. F., Islam, M. M., Uddin, M. A., Akhter, A., Yousuf, M. A., ... & Moni, M. A. (2023). A dependable hybrid machine learning model for network intrusion detection. Journal of Information Security and Applications, 72, 103405

[12] Naeem, H., Dong, S., Falana, O. J., & Ullah, F. (2023). Development of a deep stacked ensemble with process based volatile memory forensics for platform independent malware detection and classification. Expert Systems with Applications, 223, 119952.

[13] Dener, M., Ok, G., & Orman, A. (2022). Malware detection using memory analysis data in big data environment. Applied Sciences, 12(17), 8604.

[14] Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2024). Android Malware Detection and Identification Frameworks by Leveraging the Machine and Deep Learning Techniques: A Comprehensive Review. Telematics and Informatics Reports, 100130.

[15] Al-Qudah, M., Ashi, Z., Alnabhan, M., & Abu Al-Haija, Q. (2023). Effective one-class classifier model for memory dump malware detection. Journal of Sensor and Actuator Networks, 12(1), 5.

[16] Alani, M. M., Mashatan, A., & Miri, A. (2023). XMal: A lightweight memory-based explainable obfuscated-malware detector. Computers & Security, 133, 103409.

[17] Louk, M. H. L., & Tama, B. A. (2022). Tree-based classifier ensembles for PE malware analysis: A performance revisit. Algorithms, 15(9), 332.

[18] Smith, D., Khorsandroo, S., & Roy, K. (2023, February). Supervised and unsupervised learning techniques utilizing malware datasets. In 2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC) (pp. 1-7). IEEE.

[19] Roshan, K., & Zafar, A. (2024). Ensemble adaptive online machine learning in data stream: a case study in cyber intrusion detection system. International Journal of Information Technology, 1-14.

[20] Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2024). MeMalDet: A memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations. Computers & Security, 142, 103864.

[21] Roy, K. S., Ahmed, T., Udas, P. B., Karim, M. E., & Majumdar, S. (2023). Malhystack: A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. Intelligent Systems with Applications, 20, 200283.

[22] Cevallos-Salas, D., Grijalva, F., Estrada-Jiménez, J., Benítez, D., & Andrade, R. (2024). Obfuscated Privacy Malware Classifiers based on Memory Dumping Analysis. IEEE Access.

[23] Nugraha, A., & Zeniarja, J. (2022). Malware Detection Using Decision Tree Algorithm Based on Memory Features Engineering. Journal of Applied Intelligent System, 7(3), 206-210b

[24] Noor, B., & Qadir, S. (2023). Machine Learning and Deep Learning Based Model for the Detection of Rootkits Using Memory Analysis. Applied Sciences, 13(19), 10730.

[25] Özkam, Y. (2023). Malware Detection in Forensic Memory Dumps: The Use of Deep Meta-Learning Models. Acta Infologica, 7(1), 165-172

[26] Yogesh, K. M., Arpitha, S., Stephan, T., Praksha, M., & Raghu, V. (2023, December). Unravelling Obfuscated Malware Through Memory Feature Engineering and Ensemble Learning. In International Conference on Information and Communication Technology for Competitive Strategies (pp. 323-332). Singapore: Springer Nature Singapore.

[27] MalMem-Classification, https://github.com/oguzhankirlar/MalMem-Classification, Erişim Tarihi: 24.06.2024.