T.C
DÜZCE ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

LİSANS BİTİRME TEZİ

UNSW-NB15 VERİ SETİ İLE SINIFLANDIRMA ALGORİTMALARI

MELİH CAN AKBULUT

DANIŞMAN
DR. ÖĞR. ÜYESİ ESRA ŞATIR

HAZİRAN 2021
DÜZCE

ÖNSÖZ

Bu çalışma boyunca gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Sayın Esra ŞATIR'a en içten dileklerimle teşekkür ediyorum.

Ayrıca tüm eğitim hayatım boyunca bana yol gösteren, destek olan, en önemlisi bana eğitimin ne kadar önemli olduğu bilincini ve çalışma disiplinimi kazandıran , her zaman yanımda olan aileme en içten dileklerimle teşekkürlerimi sunuyorum.

Melih Can Akbulut 151002016

İÇİNDEKİLER

ŞEKİL LİSTESİ

ÖZET

UNSW-NB15 VERİ SETİ İLE SINIFLANDIRMA ALGORİTMALARI

Her geçen yıl siber saldırılar küresel ekonomi için bir numaralı tehdit olma yolunda ilerliyor. Devletler, büyük şirketler, küçük diyebileceğimiz şirketler dahil olmak üzere çoğunun siber güvenlik departmanı bulunmaktadır. Bulunmayan şirketler ise siber güvenlik firmalarından danışmanlık almaktadır. Açıklanan verilere göre 2017 yılında siber suç, küresel ekonomiye 600 milyar dolara 2018 yılında ise mali hasar yıllık %50 artışla 1 trilyon doları aştı [1]. Bunun yanı sıra siber zorbalık her geçen yıl artmakta ve önüne geçilmesi gereken bir sorun haline gelmektedir. Bu çalışmada ise Siber Güvenlik de Yapay Zeka Uygulamaları konu başlığı altında Ağ Trafik Analizi ile anomali tabanlı Saldırı Tespit / Engelleme Sistemleri (IDS/IPS) geliştirmiştir. Çalışmada kullanılan veri kümesi, 2015 yılında New South Wales Üniversitesi tarafından Avusturalya Siber Güvenlik Merkezi'nin laboratuvarlarında 49 özellik oluşturmak için on iki algoritma geliştirerek oluşturduğu veri seti olan UNSW-NB15 [2] veri kümesidir. Bu veri seti üzerinde farklı sınıflandırma algoritmaları kullanılmış ve sonuçlar gözlemlenmiştir.

Anahtar sözcükler : Siber Güvenlik, Makine Öğrenmesi, UNSW-NB15, Yapay Sinir Ağları, En Yakın Komşu, Navie Bayes

SUMMARY

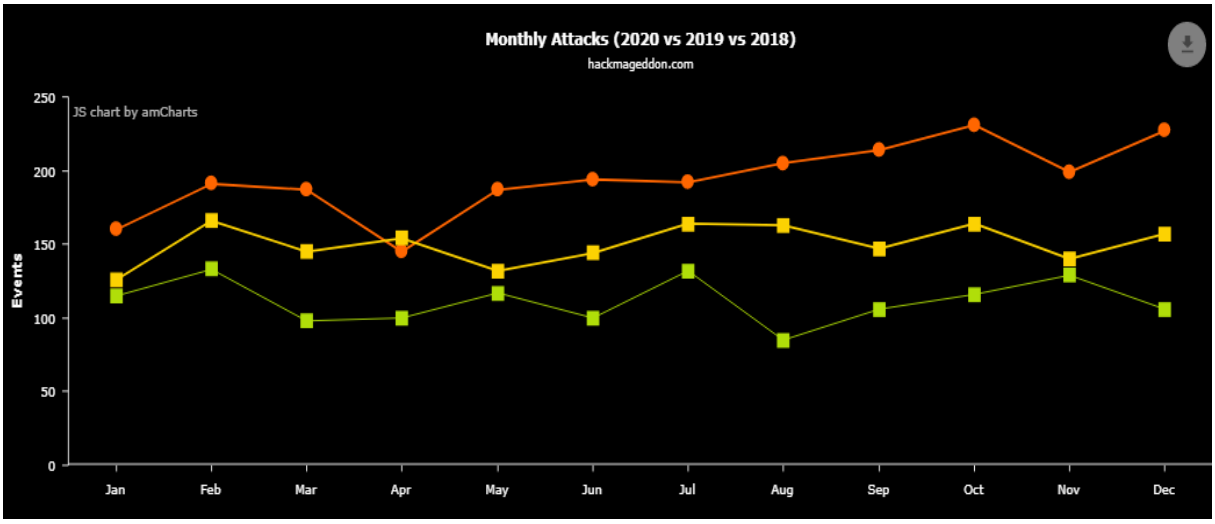
CLASSIFICATION ALGORITHMS WİTH UNSW-NB15 DATASET

With each passing year, cyber attacks are on track to become the number one threat to the global economy. Most of them, including states, large companies, small companies, have cybersecurity departments. Companies that do not exist receive advice from cybersecurity firms. According to the data announced in 2017, cybercrime cost the global economy $ 600 billion in 2018, while financial damage exceeded $ 1 trillion, an annual increase of 50% [1]. In addition, cyberbullying is increasing every year and becoming a problem that needs to be avoided. In this study, Cyber Security also developed anomaly-based intrusion detection / blocking systems (IDS/IPS) with Network Traffic Analysis under the topic of artificial intelligence applications. The dataset used in the study is the UNSW-NB15 [2] dataset, which was created by the University of New South Wales in 2015 by developing twelve algorithms to create 49 features in the laboratories of the Australian Cybersecurity Centre. Different classification algorithms were used on this data set and the results were observed.

Keywords : Cyber Security, Machine Learning, UNSW-NB15, Artificial Neural Networks, Nearest Neighbor, Navie Bayes

1. GİRİŞ

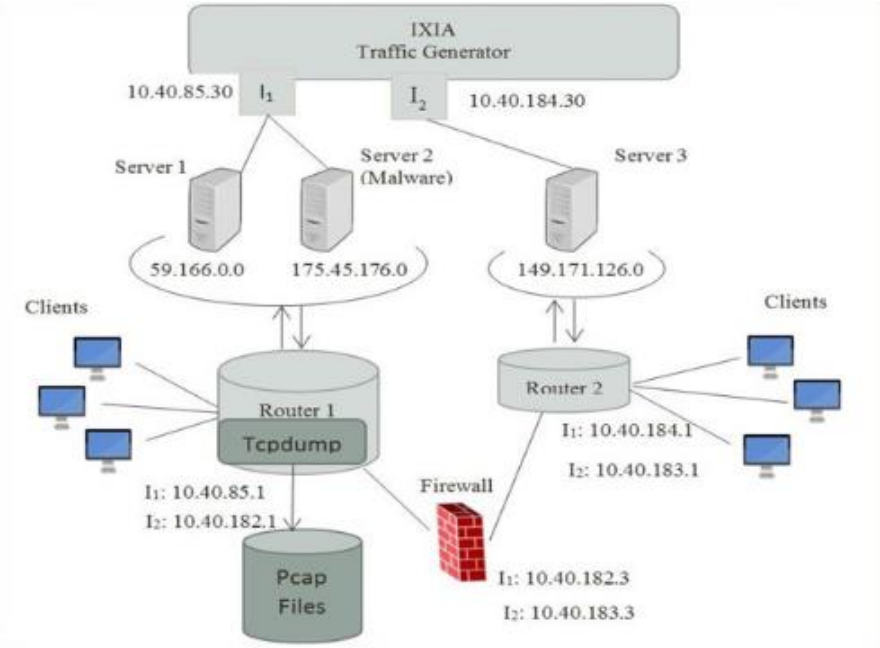
Siber güvenlik kavramı ve bilgisayarların güvenliğe ihtiyacı 1986 yılında kamu ve kişisel fark etmeksizin kişilerin bilgisayarlarına yoğun bir şekilde saldırı yapılması Amerikan hükümetini harekete geçirdi. IBM 1989 yılında ilk anti virüs programını yazdı. 1994 yılında web tarayıcıları üretildi. Bunların peşine güvenlik duvarları ve belirli işletim sistemlerinin bazı arka kapıları ve birçok güvenlik açığı kapatıldı. Hızlı bir tarihçenin ardından aslında siber güvenlik her zaman kullandığımız bir alt yapı idi. Bunlara örnek olarak virüs programları, güvenlik duvarları, e-posta kutumuzda ki spam postalara kadar daha detay verecek olursak bir yerel ağ üzerinde port yönetimlerine kadar aslında siber güvenlik tamamen hayatımıza girmiş durumda. Fakat siber zorbaların mottosu olan 'Her sistemin bir arka kapısı vardır, hiçbir sistem kusursuz değildir' ile her geçen gün başka arka kapılar başka sızma yöntemleri geliştirilmektedir. Bununla beraber bu hızlı gelişime ayak uydurmaya çalışan siber güvenlik kavramıdır. Şekil 1'de [3] açıklanan son verilere ve yıllara göre siber saldırı oranları yer almaktadır.



Şekil 1.1 : Yıllara Göre Siber Saldırı

Günümüzde Yapay Zeka ve Siber Güvenlik üzerine çokça bilim insanının çalışmakta olduğu ve her gün gelişmekte olan çalışma dallarıdır. Yapay zeka; sağlık bilimleri, e-ticaret gibi çoğu alanda da kullanıldığı gibi siber güvenlik alanında da hali hazırda kullanılmakta. Siber güvenlik virüs korumaları, güvenlik duvarları, ağ üzerindeki veya bilgisayarların gerekli zafiyet açıkları, e-postaların incelenmesi ve gerekli eğitimlerin sağlanması gibi birçok alt başlıkta incelenebilir ve tedbir alınmalıdır ki bunlar sadece üstün körü bahsedebileceğimiz güvenliklerdir. Bu çalışmada kullanılan veri setinde Avusturalya Siber Güvenlik Merkezinin bir ağ modeli

üzerinde belirli saldırıların gerçekleştiği ağ trafiğini izleyerek oluşturulan UNSW-NB15 veri setidir. Aşağıda bu veri seti için hazırlanan ağ modeli görseli verilmiştir



Şekil 1.2 : UNSW-NB15 İçin Oluşturulan Ağ Modeli

Projemizin problemi her ne kadar ağ üzerinde bazı saldırılar log alma işlemleri ile veya gerekli incelemeler ve izlemeler ile fark edilebilir olsa da insanoğlunun gözden kaçırabileceği, ağı 7/24 izlemek için firmaların vardiyalı şekilde çalışanlarına daha fazla bütçe ayırması gerekeceği kısacası daha çok insan gücünü kullanmak vb. durumlar ile karşı karşıyadır. Bu iş gücü, hata toleransı ve maliyetinden kurtulmak için makine öğrenmesinin birden çok sınıflandırma algoritması kullanılacak ve en iyi doğruluk ile çalışan model UNSW-NB15 veri seti ile eğitilen ve 9 farklı saldırı türü için anormallikleri tespit ederek gelebilecek herhangi bir tehdit veya saldırıların önüne geçilebilecektir.

Bu çalışmada, Python ve ilgili kütüphaneleri kullanılarak Jupyter Notebook üzerinde veri seti, sınıflandırma algoritmaları üzerinde farklı modeller ile eğitim ve test işlemleri gözlemlenmiştir. Sonuçlara, Tartışma ve Sonuç kısmında yer verilmiştir.

2. GENEL KISIMLAR

KDDcup99[4] veri seti ile geçmiş yıllarda yapılan bir Yapay Sinir Ağları ile Saldırı Tespit Sistemi geliştirilmiş. KDDcup99 veri setinin %10'luk kısmı kullanılmış ve Matlab r2012b ile kodlanmış bir çalışma mevcut. Kısaca KDDcup99 veri setinden de bahsetmek gerekirse askeri ağ ortamında simüle edilen ve çok çeşitli izinsiz girişleri içeren ve Üçüncü Uluslararası Bilgi Keşfi ve Veri Madenciliği Araçları Yarışması için hazırlanan ve kullanılan veri setidir. Yapılan bu çalışmada Yapay Sinir Ağları kullanılarak gerekli nöron ve iterasyon sayılarının değiştirilerek denenmesi sonucunda %99'un üzerinde bir başarı sağlanmıştır. [5]

İnternetin yaygınlaşması ve ağa bağlı cihazların artmasından kaynaklı gelişen sorunlarımızın en önemlisi siber güvenliktir. Bir bilgisayarın ağına saldırı olup olmadığını yüksek başarı oranı ile tespit edilebilir bunun yanı sıra makineye öğrettiğimiz saldırı türlerinin sisteme zarar vermeye çalışıp çalışmadığını da ayırt edebileceğinden makine öğrenme algoritmaları hala geliştirilmeye çalışmakta ve saldırı tespit sistemleri de makine öğrenmesi algoritmaları ile geliştirilmektedir. Bu çalışmada ise CSE-CIC-IDS2018 veri seti kümesi kullanılmıştır. Benim için burada yapılan en önemli çalışma veri seti üzerinde bütün 79 özniteliğin tespit edilmek istenilen saldırı türüne göre öznitelik azaltma çalışmayı yapılmış ve 5 farklı veri seti ile ağı eğiterek sadece 79 özniteliğin sahip olduğu veri setinden daha yüksek bir doğruluk oranı elde edilmiştir. Yapılan çalışmaya ek olarak kdd-99 veri seti üzerinde destek vektör makineleri ve yapay sinir ağlarının doğruluk oranı test edilmiş ve yapay sinir ağlarının saldırı tespit doğruluğu destek vekör makinelerinden daha iyi bir performans sağladığı görülmüştür. Bu çalışmanın proje konumla hemen hemen çok benzemekte fakat makaledeki çalışmada DDOS, Bot, BruteForce, DOS şeklinde 4 farklı tehdit sınıflandırıcısı oluşturulmuştur. Kullanacağım veri seti ve planladığım çalışma, amaç olarak bu saldırı türlerinden daha fazlası için kontrol sağlanması amaçlanmaktadır. [6]

Bu çalışmada, anomali tabanlı saldırı tespit sistemlerinin makine öğrenmesi metodolojileri ile daha önce öğrenilmemiş saldırılar üzerinde daha az yanlış alarm vermekte olduğu ve daha iyi performans ile çalıştığı gözlemlenmiştir. Anomali tespitini en üst düzeyde doğrulukla ve daha hassas çalışması için denetimli, yarı denetimli, denetimsiz makine öğrenmesi yaklaşımları kullanılmıştır. Makalede daha önce saldırı tespit sistemleri üzerine çalışılmış birden fazla araştırmacının hangi algoritma ve hangi yaklaşımlar ile çalıştığını anlatmıştır. Yine bu makalede de destek vektör ile yapay sinir ağı da karşılaştırılmıştır. Destek vektörü yine geride kalmıştır. Ayrıca, kendi önerdikleri binom sınıflandırma kullanılarak 40 öznitelik üç gizli

katman ile daha az sayıda öznitelik kullanılarak performansı karşılaştırılmış ve daha öz nitelik ile daha kesin sonuçlar elde edilmiş bir çalışma incelenmiştir. Yine bu makalede de NSL-KDD, KDD99, ADFA, UNSW-NB15 veri setleri incelenmiştir. Bu makale diğer makalelere göre daha çok ilgimi çekme sebebi benim kullandığım veri seti kullanılmıştır. Ayrıca her bir katmanda 10 nöron olacak şekilde 10 gizli katman kullanılmış ve 10 epoch ile eğitilmiştir. Önerdikleri metod %99.5 doğruluk ile çalışırken yapay sinir ağları %81.5 doğruluk ile çalıştığı gözlemlenmiştir. [7]

Bu çalışmada, çok katmanlı perceptron kullanılmış ve geri yayılım algoritması ile geliştirilmiştir. İmzalara dayalı 'daha önceden öğretilmiş bir saldırı şeklini' anomali tespit yöntemleri, düşük yanlış alarm oranları ile çalışırken bilinmeyen saldırılar için ise ağımıza sürekli güncel veriler ile destek sağlamalıyız. Makalede NewFlow protokolü ile alınan veriler analiz edilmiştir. Bunun yanı sıra yapay sinir ağında aktivasyon fonksiyonu olarak sigmoid fonksiyon kullanılmıştır. Eğitim sırasında ağa ileri doğru yayılan girişler sunularak YSA'nın ürettiği çıkış ile istenen çıkış karşılaştırıldıktan sonra çıkış farklı ise geri yayılım ile çalışılmıştır. [8]

Parameter	KDD99 dataset	UNSW-NB15 dataset
Number of sub-networks	2	3
Number of distinct IPs	11	45
Simulation	Yes	Yes
Duration of simulation	5 weeks	31 hours
Data formats	Tcpdump, BSM and data dump files	Pcap, BRO, Argus and CSV files
Attack types	Dos, Probe, U2R and R2L	Fuzzers for malicious activities, Analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode and Worm
Feature extraction tools	Bro-IDS	Argus, Bro-IDS and new scripts
Number of features extracted	41	47

Şekil 2.1 : KDD99 ile UNSW-NB15 Veri Setlerinin Karşılaştırılması

3. KULLANILAN ARAÇ VE YÖNTEM

3.1 KULLANILAN VERİTABANI

Uygulamada kullanılan veri tabanı New South Wales Üniversitesi üzerinde paylaşılan UNSW-NB15 veri kümesidir. Bu veri seti Avusturalya Siber Güvenlik Merkezinde (ACCS) laboratuvar ortamında 9 farklı saldırı türü ve 100GB ham veri olan normal ağ paketlerinin yakalanarak 12 farklı algoritma ile oluşturulmuştur. Ağ üzerinde gerçek ve sentetik saldırı melezleri oluşturulmuştur. Bu veri setinde depolanan 2 milyon 540 bin veri bulunmaktadır. Bu verileri işlemesi ve bir ağa eğitimi kolay olmayacağından ayrı ayrı hem eğitim için ayrılmış 175.341 kayıt test için ise ayrılmış 82.332 kaydın csv formatındaki dosyaları paylaşıma sunulmuştur. Veri setinde bir ağ üzerinde tehdit yaratabilecek 49 farklı özniteliğe yer verilmiştir. Veri kümesinde Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode ve Worms olmak üzere dokuz saldırı türü vardır. Veri setinde 1 sınıf bulunmaktadır '1' için saldırı var '0' için saldırı yok şeklinde hazırlanmıştır. Öznitelikler; integer, nominal, float, ve binary türünden öznitelikler bulunmaktadır. Veri setinde herhangi bir boş veri bulunmamaktadır fakat bazı bulunmayan değerler (-) ile doldurulmuştur. Kullanılacak algoritmalara ve denemelere göre veri seti üzerinde normalizasyon işlemleri yapılacaktır bir sonra ki başlık altında incelenecektir. Aşağıdaki şekilde UNSW-NB15 için öznitelikler ve açıklamaları yer almaktadır.

Attribute Number	Feature	Description
42	ct_srv_dst	No. of connections that contain the same service (#14) and destination address (#3) in 100 connections according to the last time (#26).
43	ct_dst_ltm	No. of connections of the same destination address (#3) in 100 connections according to the last time (#26).
44	ct_src_ ltm	No. of connections of the same source address (#1) in 100 connections according to the last time (#26).
45	ct_src_dport_ltm	No of connections of the same source address (#1) and the destination port (#4) in 100 connections according to the last time (#26).
46	ct_dst_sport_ltm	No of connections of the same destination address (#3) and the source port (#2) in 100 connections according to the last time (#26).
47	ct_dst_src_ltm	No of connections of the same source (#1) and the destination (#3) address in in 100 connections according to the last time (#26).
48	attack_cat	The name of each attack category.
49	Label	0 for normal and 1 for attack records

Attribute Number	Feature	Description
1	srcip	Source IP address
2	sport	Source port number
3	dstip	Destination IP address
4	dsport	Destination port number
5	proto	Transaction protocol
6	state	Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state)
7	dur	Record total duration
8	sbytes	Source to destination transaction bytes
9	dbytes	Destination to source transaction bytes
10	sttl	Source to destination time to live value
11	dttl	Destination to source time to live value
12	sloss	Source packets retransmitted or dropped
13	dloss	Destination packets retransmitted or dropped
14	service	http, ftp, smtp, ssh, dns, ftp-data ,irc and () if not much used service
15	Sload	Source bits per second
16	Dload	Destination bits per second
17	Spkts	Source to destination packet count
18	Dpkts	Destination to source packet count
19	swin	Source TCP window advertisement value
20	dwin	Destination TCP window advertisement value
21	stcpb	Source TCP base sequence number
22	dtcpb	Destination TCP base sequence number
23	smeansz	Mean of the packet size transmitted by the source
24	dmeansz	Mean of the packet size transmitted by the destination
25	trans_depth	Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_len	Actual uncompressed content size of the data transferred from the server's http service
27	Sjit	Source jitter (mSec)
28	Djit	Destination jitter (mSec)
29	Stime	record start time
30	Ltime	record last time
31	Sintpkt	Source interpacket arrival time (mSec)
32	Dintpkt	Destination interpacket arrival time (mSec)
33	tcprtt	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'.
34	synack	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
35	ackdat	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
36	is_sm_ips_ports	If source (#1) and destination (#3) IP addresses equal and port numbers (#2)(#4) equal then, this variable takes value 1 else 0
37	ct_state_ttl	No. for each state (#6) according to specific range of values for source/destination time to live (#10) (#11).
38	ct_flw_http_mthd	No. of flows that has methods such as Get and Post in http service.
39	is_ftp_login	If the ftp session is accessed by user and password then 1 else 0.
40	ct_ftp_cmd	No of flows that has a command in ftp session.
41	ct_srv_src	No. of connections that contain the same service (#14) and source address (#1) in 100 connections according to the last time (#26).

Şekil 3.1 : UNSW-NB15 Veri Seti Öznitelikleri ve Açıklamaları

3.2 VERİ ÖN İŞLEME VE NORMALİZASYON

Veri tabanında belirtilen 49 öznitelik için hazırlanan eğitim ve test veri setinde hazırlanan kurum tarafından bazı bilgi içermeyen öznitelikler (“srcip” gibi kaynak ip adresi) çıkartılarak öznitelik sayısı 44’e düşürülmüştür. Modellerimiz oluşturulmadan önce ID kısmı bilgi içermediğinden işlenecek veriden çıkartılmıştır. Ayrıca veri setinin Label kısmı da sınıf niteliği taşıdığı için giriş olarak yer alamayacağından veri setinden çıkartılmıştır. Bu işlemler sonucunda modellerimiz için gerekli öznitelik sayısı 43 olarak belirlenmiştir ve 1 de sınıf bilgisi içeren özniteliğimiz yer almaktadır. Normalizasyon işlemi için ;

```
“from sklearn.preprocessing import MinMaxScaler”
```

Kütüphanesi kullanılmıştır. Bu işlemle beraber veri setinde bulunan nominal değerleri rastgele sayısal değerlere çevirmek için ve oluşturulan modellerde sayısal olarak kullanılması için ;

```
“from sklearn.preprocessing import LabelEncoder ”
```

Kütüphanesi kullanılmıştır.

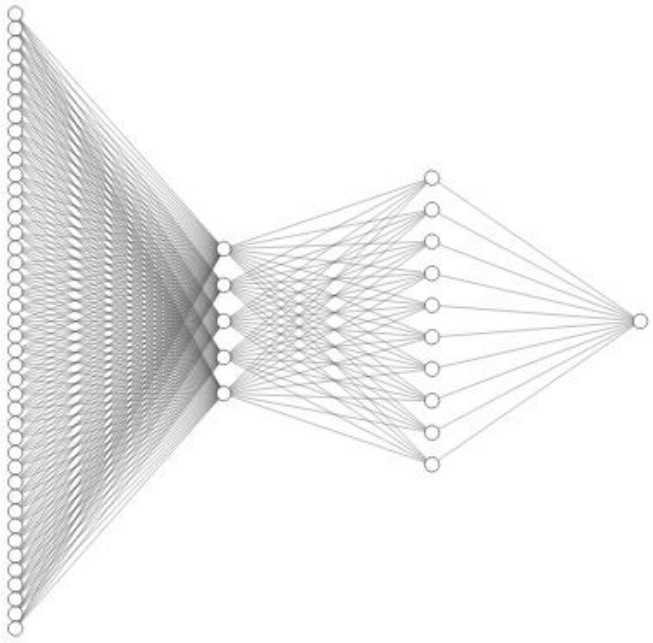
3.3 YAPAY SİNİR AĞI (ANN)

Normalizasyon işlemlerinde gördüğümüz gibi yapay sinir ağı modellerimiz için 43 öznitelik için 43 giriş nöronu kullanmak gerekiyor ve 1 de çıkış nöronu kullanılması gerekmektedir. Oluşturulan bütün yapay sinir ağı modellerimizde 1. gizli katmanda ve 2 gizli katmanda ‘relu’ aktivasyon fonksiyonu, çıkış katmanı için ise ‘sigmoid’ aktivasyon fonksiyonu kullanılmıştır. Durdurma kriterimiz olarak 10 epoch seçilmiştir. Ağırlık güncellemeleri için mini-batch gradient descent kullanılmış ve bu değer ilgili kodumuzda default olarak 32 verilmiştir. Öğrenme katsayısı ise yine default olarak 0.001 olarak kullanılmıştır. Ayrıca sınıflandırma işlemi yaptığımızdan loss fonksiyonu olarak binary crossentropy kullanılmıştır.

Veri ön işleme ve normalizasyon başlığı altında veri setine uygulanan işlemlerin tamamı burada kullanılmıştır. 2 farklı yapay sinir ağı modeli üzerinde normalize edilmiş veri kullanılarak eğitim ve test işlemleri gerçekleştirilmiş ve sonuç gözlemlenmiş ve alt başlıklarda incelenmiştir.

3.3.1 MODEL 43, 5, 10, 1

Uygulanan bu modelde birinci ara katmanda 5 ikinci ara katmanda 10 nöron bulunmaktadır. Test sonucu doğruluk oranı 0.946 olarak gözlemlenmiştir. Ağın eğitimi, test sonuçları ve modelin şekli aşağıda verilmiştir.



Şekil 3.2 : Kullanılan Yapay Sinir Ağ Modeli

```
Epoch 1/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.2785 - accuracy: 0.8780
Epoch 2/10
3836/3836 [==============================] - 4s 1ms/step - loss: 0.1199 - accuracy: 0.9389
Epoch 3/10
3836/3836 [==============================] - 4s 1ms/step - loss: 0.1182 - accuracy: 0.9407
Epoch 4/10
3836/3836 [==============================] - 4s 1ms/step - loss: 0.1175 - accuracy: 0.9427
Epoch 5/10
3836/3836 [==============================] - 4s 1ms/step - loss: 0.1153 - accuracy: 0.9443
Epoch 6/10
3836/3836 [==============================] - 4s 1ms/step - loss: 0.1143 - accuracy: 0.9440
Epoch 7/10
3836/3836 [==============================] - 4s 1ms/step - loss: 0.1148 - accuracy: 0.9436
Epoch 8/10
3836/3836 [==============================] - 4s 1ms/step - loss: 0.1166 - accuracy: 0.9430
Epoch 9/10
3836/3836 [==============================] - 4s 1ms/step - loss: 0.1154 - accuracy: 0.9447
Epoch 10/10
3836/3836 [==============================] - 4s 1ms/step - loss: 0.1161 - accuracy: 0.9437
<tensorflow.python.keras.callbacks.History at 0x7f64403e2510>
```

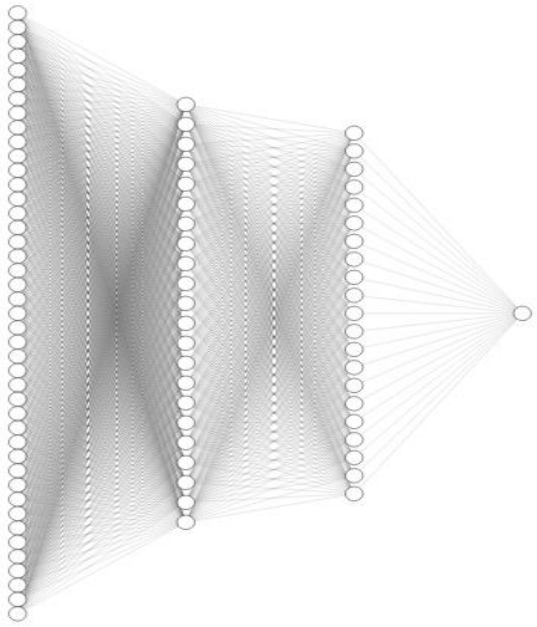
Şekil 3.3 : Ağın Eğitimi

```
1644/1644 [==============================] - 1s 818us/step - loss: 0.1141 - accuracy: 0.9460
[0.11411053687334406, 0.9459726810455322]
```

Şekil 3.4 : Test Sonucu Doğruluk Oranı

3.3.2. MODEL 43, 22, 22, 1

Bu modelde birinci ara katmanda 5 ikinci ara katmanda 10 nöron bulunmaktadır ayrıca normalizasyonun modele katkısını gözlemlemek amacıyla hem normalize edilmemiş veri ile hem de normalize edilmiş veri ile ağın eğitim ve test işlemleri gerçekleştirilmiştir. Normalize edilmemiş veri ile eğitilen ağda test veri seti üzerinde ki doğruluk 0.744 olarak gözlemlenmiştir. Normalize edilmiş veri ile ise ağ 0.9997 olarak gözlemlenmiştir. Normalize edilmiş veri için aşağıda ekran alıntıları paylaşılmıştır.



Şekil 3.5 : Kullanılan Yapay Sinir Ağ Modeli

```
Epoch 1/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.2142 - accuracy: 0.9022
Epoch 2/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.1032 - accuracy: 0.9546
Epoch 3/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.0805 - accuracy: 0.9666
Epoch 4/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.0421 - accuracy: 0.9843
Epoch 5/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.0153 - accuracy: 0.9961
Epoch 6/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.0051 - accuracy: 0.9990
Epoch 7/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.0038 - accuracy: 0.9992
Epoch 8/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.0026 - accuracy: 0.9994
Epoch 9/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.0022 - accuracy: 0.9995
Epoch 10/10
3836/3836 [==============================] - 5s 1ms/step - loss: 0.0022 - accuracy: 0.9995
<tensorflow.python.keras.callbacks.History at 0x7f643f25dc10>
```

Şekil 3.6 : Ağın Eğitimi

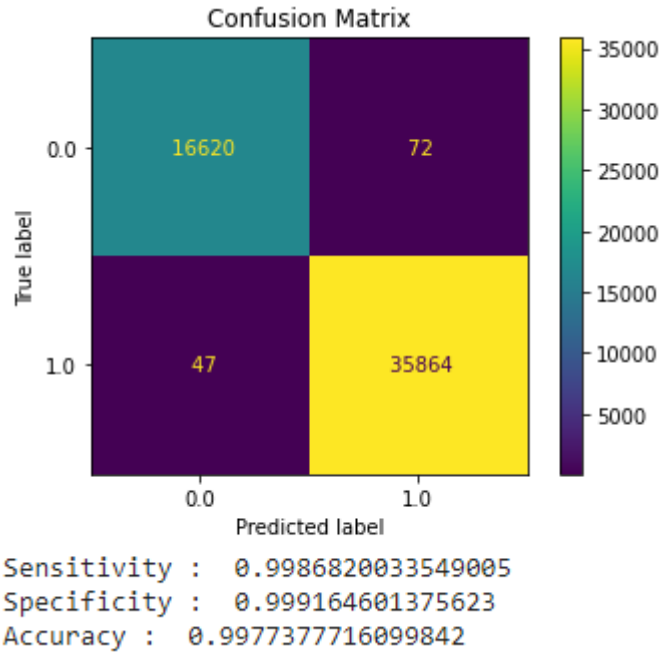
```
1644/1644 [==============================] - 1s 830us/step - loss: 0.0012 - accuracy: 0.9997
[0.0011957898968830705, 0.9996768236160278]
```

Şekil 3.7 : Test Sonucu Doğruluk Oranı

3.4. EN YAKIN KOMŞU(KNN)

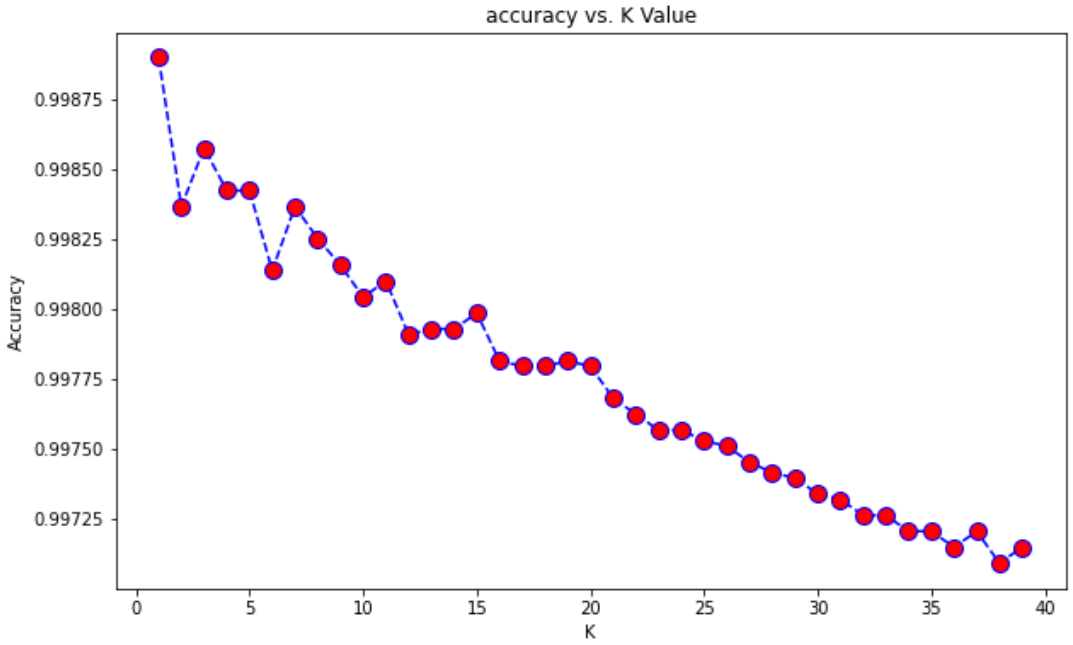
Bu algoritmada daha önce yapay sinir ağlarında kullanılan normalizasyon işlemlerinin aynısı kullanılmıştır. Bu çalışmada başlangıçta veri setine en uygun mesafe metriğini görebilmek adına komşu sayısı default olarak 5 iken euclidian, manhattan ve minkovski metrikleri ile ayrı ayrı eğitim ve test aşamaları gerçekleştirilmiş ve manhattan metriğinin performansı diğer metriklere göre doğruluk oranının daha yüksek olduğu gözlemlenmiştir. Bu işlemden sonra yine deneme yanılma yöntemi ile en iyi K değerini bulmak adına denemeler yapılmıştır. Alt başlıklarda ilgili K değeri ve performans metrikleri yer almaktadır.

Kullanılan KNN sınıflandırma algoritması için K değerleri 1, 3, 5, 7 komşu değerleri ayrı ayrı denenmiş ve test sonucu en iyi performans K=3 değerinde gözlemlenmiştir. Bu algoritmada mesafe metriği olarak minkowski metriği kullanılmıştır.

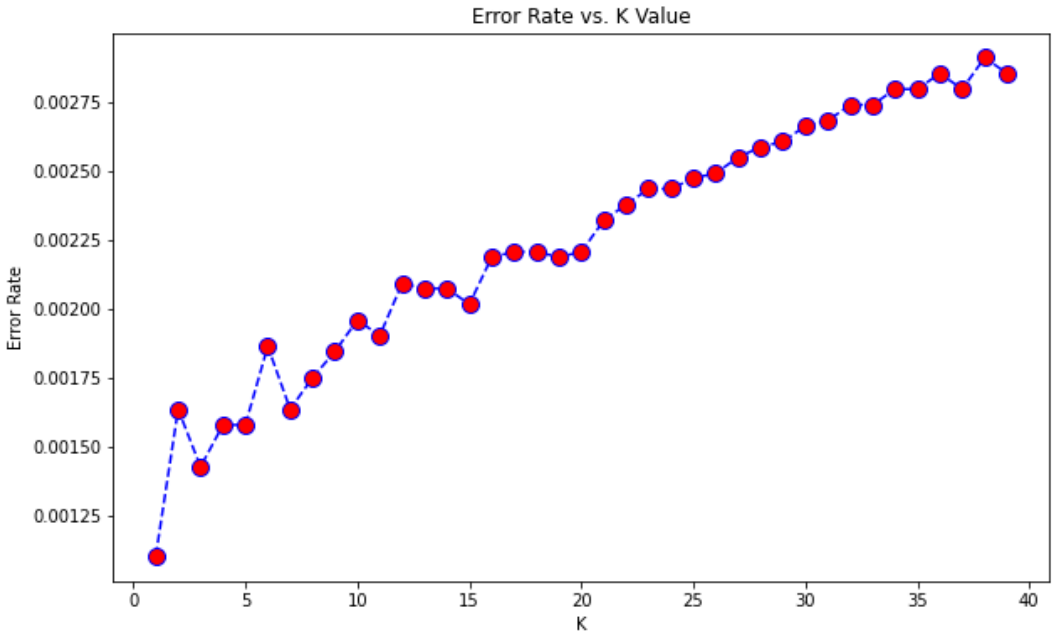


Şekil 3.8 : K=3 İçin Performans Metrikleri

Aşağıda en iyi K değerini belirlemek için döngü içerisinde her K değeri için oluşturulan modelde eğitim ve test sonucu doğruluk, hata oranı, K değeri hesaplanmış ve grafiğe dökülmüştür.



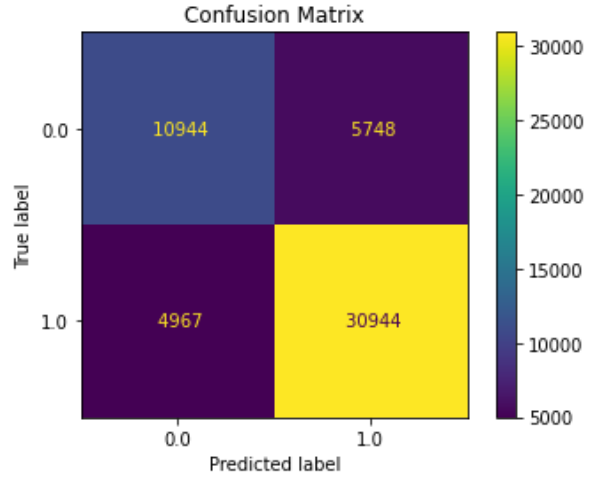
Şekil 3.9 : Doğruluk Oranının K Değerine Göre Değişimi



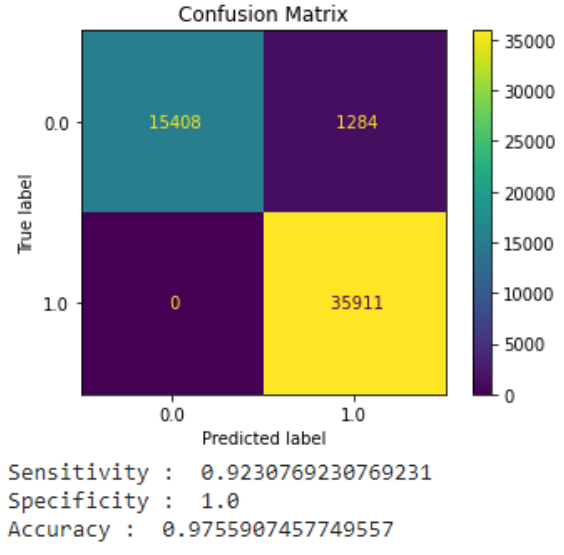
Şekil 3.10 : Hata Oranının K Değerine Göre Değişimi

3.5. NAVİE BAYES (NB)

Bu algoritmada da yine daha önce ki normalizasyon işlemleri ile hazırlanan veri seti kullanılmıştır. Bu modelde de önce normalizasyon edilmemiş veri ve normalizasyon uygulanmış verilerin performans metrikleri karşılaştırılmıştır. Aşağıdaki şekillerde performans metrikleri sonucu yer almaktadır.



Şekil 3.11 : Normalizasyon Uygulanmamış Veri Performans Metrikleri



Şekil 3.12 : Normalizasyon Uygulanmış Veri Performans Metrikleri

4. TARTIŞMA VE SONUÇ

Bu çalışmada, kullanılan algoritmalar ve normalizasyon işlemleri sonrasında ki gözlemlere göre veri setine en uygun ve en yüksek doğruluk oranı, yapay sinir ağlarında ve 43 giriş 22 birinci gizli katman 22 ikinci gizli katman 1 çıkış şeklinde kullanılan modelde 0.9997 olarak gözlemlenmiştir.

KAYNAKLAR

[1] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

[2] https://www.technology.org/2019/07/17/biggest-cyber-attacks-and-their-cost-for-the-global-economy/

[3] https://www.hackmageddon.com/category/security/cyber-attacks-statistics/

[4] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[5] Mehmet Y., Abdullah Ç., Baha Ş., İdris B., 2014, "Yapay Sinir Ağları ile Ağ Üzerinde Saldırı Tespiti ve Paralel Optimizasyonu"

[6] Mehmet S. K., Metin T., Muhammed A. A., 2020, "Yapay Sinir Ağı Kullanılarak Anomali Tabanlı Saldırı Tespiti Modeli Uygulaması"

[7] Liu Z., Ghulam M., Li B., Luo J., Zhu Y., Lin Z., 2019, " Modeling Network Intrusion Detection System Using Feed-Forward Neural Network Using UNSW-NB15 Dataset "

[8] Andropov S., Budko M., Budko Mi., Guirik A., 2017, "Network Anomaly Detection using Artificial Neural Networks "

EKLER

EK-ANN

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
df = pd.read_csv('unsw_egitim.csv')
df.drop("service", axis='columns')
df.drop("proto", axis='columns')
df.drop("state", axis='columns')
df.drop("attack_cat", axis='columns')
l2 = LabelEncoder()
label1 = l2.fit_transform(df['service'])
df["service"] = label1
label2 = l2.fit_transform(df['proto'])
df["proto"] = label2
label3 = l2.fit_transform(df['state'])
df["state"] = label3
label4 = l2.fit_transform(df['attack_cat'])
df["attack_cat"] = label4
Egitim = df
Egitim.pop("id")
print(Egitim)
veri = Egitim.values
X = veri[:,0:43]
Y = veri[:,43]
x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size=0.3,
random_state = 0)
minMaxScaler = MinMaxScaler()
minMaxScaler.fit(x_train)
minMaxScaler.fit(x_train)
x_train_norm = minMaxScaler.transform(x_train)
x_test_norm = minMaxScaler.transform(x_test)
print(x_train_norm)
#learning_rate=0.001,
classifier = Sequential()
classifier.add(Dense(22,kernel_initializer="normal", activation = 'relu
', input_dim=43))
classifier.add(Dense(22,kernel_initializer="normal", activation = 'relu
'))
classifier.add(Dense(1,kernel_initializer="normal", activation = 'sigmo
id'))
classifier.compile(optimizer= 'adam', loss = 'binary_crossentropy', met
rics = ['accuracy',])
classifier.fit(x_train, y_train ,epochs = 10)
classifier.evaluate(x_test,y_test)
classifier = Sequential()
```

```
classifier.add(Dense(10,kernel_initializer="normal", activation = 'relu
', input_dim=43))
classifier.add(Dense(5,kernel_initializer="normal", activation = 'relu'
))
classifier.add(Dense(1,kernel_initializer="normal", activation = 'sigmo
id'))
classifier.compile(optimizer= 'adam', loss = 'binary_crossentropy', met
rics = ['accuracy'])
classifier.fit(x_train_norm, y_train ,epochs = 10)
classifier.evaluate(x_test_norm,y_test)
classifier = Sequential()
classifier.add(Dense(22,kernel_initializer="normal", activation = 'relu
', input_dim=43))
classifier.add(Dense(22,kernel_initializer="normal", activation = 'relu
'))
classifier.add(Dense(1,kernel_initializer="normal", activation = 'sigmo
id'))
classifier.compile(optimizer= 'adam', loss = 'binary_crossentropy', met
rics = ['accuracy'])
classifier.fit(x_train_norm, y_train ,epochs = 10)
classifier.evaluate(x_test_norm,y_test)
```

EK-KNN

```
classifier = KNeighborsClassifier(n_neighbors=3)
classifier.fit(x_train_norm, y_train)
y_pred = classifier.predict(x_test_norm)
matrix = plot_confusion_matrix(classifier, x_test_norm, y_test,values_f
ormat=" ")
matrix.ax_.set_title("Confusion Matrix", color="black")
plt.show()
sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ',sensitivity1)
specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)
acc = accuracy_score(y_test, y_pred)
print('Accuracy : ',acc)
classifier = KNeighborsClassifier(n_neighbors=1)
classifier.fit(x_train_norm, y_train)
y_pred = classifier.predict(x_test_norm)
matrix = plot_confusion_matrix(classifier, x_test_norm, y_test,values_f
ormat=" ")
matrix.ax_.set_title("Confusion Matrix", color="black")
plt.show()
sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ',sensitivity1)
specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)
acc = accuracy_score(y_test, y_pred)
```

```
print('Accuracy : ',acc)
classifier = KNeighborsClassifier(n_neighbors=7)
classifier.fit(x_train_norm, y_train)
y_pred = classifier.predict(x_test_norm)
matrix = plot_confusion_matrix(classifier, x_test_norm, y_test,values_f
ormat=" ")
matrix.ax_.set_title("Confusion Matrix", color="black")
plt.show()
sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ',sensitivity1)
specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)
acc = accuracy_score(y_test, y_pred)
print('Accuracy : ',acc)
```

EK-NB

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
y_pred = model.fit(x_train_norm, y_train).predict(x_test_norm)
matrix = plot_confusion_matrix(model, x_test_norm, y_test,values_format
=" ")
matrix.ax_.set_title("Confusion Matrix", color="black")
plt.show()
cm = confusion_matrix(y_test, y_pred)
sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ',sensitivity1)
specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)
acc = accuracy_score(y_test, y_pred)
print('Accuracy : ',acc)
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Ad Soyad	Melih Can AKBULUT
Doğum Tarihi	16.04.1997
E-Mail	melihcanakbulutt61@gmail.com

İş Deneyimi

Ağustos 2019 – Eylül 2019	TÜBİTAK - Stajyer
Mayıs 2018 – Nisan 2019	MakiSharp – Kurucu Ortak Web Yazılım ve Tasarım

Eğitim

2015 – Haziran 2021	Düzce Üniversitesi Bilgisayar Mühendisliği

Sertifikalar ve Kurslar

11 Mart 2021	R Programala ile Modern Veri Bilimi - Udemy
21 Mayıs 2019 - 28 Mayıs 2019	Siber Güvenliğe Giriş – Cisco Networking Academy
14 Mayıs 2019 - 21 Mayıs 2019	WİFİ ve Kablosuz Ağlar - Cisco Networking Academy
2-3-4 Mart 2018	Univercity4Society ile Girişimcilik Programı Düzce Üniversitesi
16 Şubat 2017 – 24 Şubat 2017	Youthpass Genç Beyin,Genç Bakış, Genç Bilişim : Bilişim Zirvesi – Uludağ Üniversitesi
2016	Python Kursu - Düzce Üniversitesi Teknopark