
GAZİ ÜNİVERSİTESİ
Gazi
1926

SQL ENJEKSİYONU SALDIRILARININ MAKİNE ÖĞRENMESİ İLE TESPİTİ VE KORUNMA YÖNTEMLERİ

Emre POLAT

YÜKSEK LİSANS TEZİ
BİLGİ GÜVENLİĞİ MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AĞUSTOS 2023

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Emre POLAT

08/08/2023

SQL ENJEKSİYONU SALDIRILARININ MAKİNE ÖĞRENMESİ İLE TESPİTİ VE KORUNMA YÖNTEMLERİ

(Yüksek Lisans Tezi)

Emre POLAT

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Ağustos 2023

ÖZET

Veri tabanı sistemleri, birbirleriyle ilişkili bilgilerin ya da verilerin tablolar halinde yapılandırılmak suretiyle depolandığı elektronik sistemlerdir. Günümüzde veri tabanı sistemlerinden, yemek siparişinden bankacılık işlemlerine, otel rezervasyon işlemlerinden e-devlet işlemlerine, sağlık işlemlerinden sigortacılığa kadar uzanan çok geniş bir yelpaze içerisinde istifade edilmektedir. Ayrıca bu sistemler üzerinde devlete ait gizli ve hizmete özel gizlilik dereceli veriler, kurum / kuruluşlara ait veriler ve kişilere ait nüfus bilgileri, fotoğraf ve parmak izi gibi özel nitelikli veriler muhafaza edilmektedir. Dolayısıyla veri tabanı sistemlerinin başta SQL enjeksiyonu saldırıları olmak üzere muhtemel tehditlere karşı korunması çok önemli bir konudur. Bu çalışmada; SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespit edilmesi kapsamında geleneksel makine öğrenmesi yöntemleri ile artırma algoritmaları karşılaştırılmıştır. Uygulama neticesinde artırma algoritmalarından XGBoost algoritması kullanılarak 0,993 doğruluk değeri elde edilmiş ve literatürde aynı veri seti ile yapılan farklı bir çalışmaya oranla 0,043 değerinde bir iyileştirme sağlanmıştır. Yapılan çalışma ile SQL enjeksiyonu saldırılarının makine öğrenmesiyle tespit edilmesinde, artırma algoritmalarının kullanılması önerilmektedir.

Bilim Kodu	:	92403
Anahtar Kelimeler	:	SQL Enjeksiyonu Saldırısı, Siber Saldırı, Makine Öğrenmesi
Sayfa Adedi	:	84
Danışman	:	Prof.Dr. Halil İbrahim BÜLBÜL

DETECTION OF SQL INJECTION ATTACKS WITH MACHINE LEARNING AND PREVENTION METHODS

(M. Sc. Thesis)

Emre POLAT

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

August 2023

ABSTRACT

Database systems are electronic systems in which information or data related to each other is stored by structuring in tables. Today, database systems are used in a wide range from food ordering to banking transactions, from hotel reservations to e-government transactions, from health transactions to insurance. Furthermore, confidential and service-specific confidential data belonging to the state, data belonging to institutions / organizations and personal data such as population information, photographs and fingerprints are stored on these systems. Therefore, it is very important to protect database systems against possible threats, especially SQL injection attacks. In this study; within the scope of detecting SQL injection attacks with machine learning, traditional machine learning methods and boosting algorithms were compared. According to the application, an accuracy rate of 0.993 was obtained by using the XGBoost algorithm, one of the boosting algorithms, and an improvement of approximately 0.043 was achieved compared to a different study in the literature with the same data set. In this study, it is recommended to use boosting algorithms to detect SQL injection attacks with machine learning.

Science Code	:	92403
Key Words	:	SQL Injection Attack, Cyber Attack, Machine Learning
Page Number	:	84
Supervisor	:	Prof.Dr. Halil İbrahim BÜLBÜL

TEŞEKKÜR

Ciddi bir çalışma ve fedakârlık ile hazırladığım yüksek lisans tezimi tamamlamanın heyecanı ve gururunu yaşıyorum. Bu heyecanı yaşarken her daim yanımda olan değerli eşim Özge KULCANAY POLAT ve oğlum Umut POLAT'a, beni yetiştirerek bugünlere gelmemde pay sahibi olan aileme, Şehit Ütğm. Emre KARGIN başta olmak üzere vatanın en ücra köşelerinde gözlerini dahi kırpmadan canlarını feda eden aziz şehitlerimize, Cumhuriyetimizin kurucusu Gazi Mustafa Kemal ATATÜRK ve silah arkadaşlarına, çok kıymetli mesai arkadaşlarıma ve fikirleri ile her daim yolumu aydınlatan tez danışmanım sayın Prof.Dr. Halil İbrahim BÜLBÜL'e saygı ve şükranlarımı sunarım.

Emre POLAT

İÇİNDEKİLER

Sayfa

ÖZET iv

TEŞEKKÜR vi

ÇİZELGELERİN LİSTESİ x

ŞEKİLLERİN LİSTESİ xi

SİMGELER VE KISALTMALAR xii

1. GİRİŞ 1

1.1. Çalışmanın Amacı ve Kapsamı 2

2. LİTERATÜR TARAMASI 5

3. KAVRAMSAL ÇERÇEVE 9

3.1. Siber Tehditler ve Siber Tehdit Araçları 9

3.1.1. Hizmet engelleme ve dağıtık hizmet engelleme saldırıları 9

3.1.2. Ortadaki adam saldırısı 10

3.1.3. Kötücül yazılım saldırısı 10

3.1.4. Kaba kuvvet saldırısı 11

3.1.5. Oltalama saldırısı 12

3.2. SQL Enjeksiyonu Saldırıları 12

3.2.1. Enjeksiyon mekanizmaları 13

3.2.2. SQL enjeksiyonu saldırılarının amaçları 14

3.2.3. SQL enjeksiyonu türleri 16

3.2.4. SQL enjeksiyonu saldırılarında ve zafiyet tespitinde kullanılan araçlar 19

3.3. SQL Enjeksiyonu Saldırılarına Karşı Korunma Yöntemleri 21

3.3.1. Kodlama pratikleri 22

3.3.2. Tespit ve önleme sistemleri 23

Sayfa

4. YÖNTEM VE MATERYAL..... 27
4.1. Makine Öğrenmesi Türleri..... 27
4.1.1. Denetimli öğrenme 27
4.1.2. Denetimsiz öğrenme 28
4.1.3. Yarı denetimsiz öğrenme 28
4.1.4. Takviyeli öğrenme 28
4.2. Değerlendirmede Kullanılan Ölçüler 28
4.3. Uygulamada Kullanılan Sınıflandırma Algoritmaları 30
4.3.1. Geleneksel makine öğrenmesi algoritmaları 32
4.3.2. Artırma algoritmaları..... 34
4.4. Veri Setinin Elde Edilmesi 36
4.5. Veri Setlerinin Düzenlenmesi 37
5. BULGULAR VE TARTIŞMA 39
5.1.Deneysel Çalışma 39
5.2.Deney Sonuçlarının Literatür ile Karşılaştırılması..... 47
6. SONUÇ VE ÖNERİLER..... 51
KAYNAKLAR..... 53
EKLER 59
EK-1 K-En yakın komşu algoritmasına ait kod parçası..... 60
EK-2. Naive bayes algoritmasına ait kod parçası 62
EK-3. Karar ağacı algoritmasına ait kod parçası 64
EK-4. Destek vektör makinesi algoritmasına ait kod parçası 66
EK-5. Gaussian naive bayes algoritmasına ait kod parçası 68
EK-6. Rastgele ağaçlar algoritmasına ait kod parçası 70

Sayfa

EK-7. Lojistik regresyon algoritmasına ait kod parçası..... 72

EK-8. AdaBoost algoritmasına ait kod parçası..... 74

EK-9. CatBoost algoritmasına ait kod parçası..... 76

EK-10. XGBoost algoritmasına ait kod parçası..... 78

EK-11. LightGBM algoritmasına ait kod parçası..... 80

EK-12. Gradient tree boosting algoritmasına ait kod parçası..... 82

ÖZGEÇMİŞ..... 84

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 2.1. Literatürde yer alan çalışmalar 7

Çizelge 3.1. Saldırı tespit ve önleme sistemleri……..……..……..…….. 26

Çizelge 4.1. Veri seti-1..... 38

Çizelge 4.2. Veri seti-2..... 38

Çizelge 5.1. Doğruluk değerleri..... 40

Çizelge 5.2. Kesinlik değerleri..... 42

Çizelge 5.3. Duyarlılık değerleri..... 43

Çizelge 5.4. F1 puanı..... 45

Çizelge 5.5. Çapraz doğrulama değerleri..... 46

Çizelge 5.6. Karmaşıklık matrisi 47

Çizelge 5.7. Literatürde yer alan benzer çalışmalar 48

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 1.1. Venturebeat araştırması 2

Şekil 3.1. Totoloji 16

Şekil 3.2. Birleşik sorgular 17

Şekil 3.3. Saklı yordamlar 17

Şekil 3.4. Alternatif kodlama 18

Şekil 3.5. Geçersiz/mantık hatalı sorgu 18

Şekil 3.6. BBQSQL 19

Şekil 3.7. Leviathan 20

Şekil 3.8. NoSQLMAP 20

Şekil 3.9. Tyrant 21

Şekil 3.10. WhiteWidow 21

Şekil 4.1. Makine öğrenmesi türleri 27

Şekil 4.2. Doğruluk değeri hesaplaması 29

Şekil 4.3. Kesinlik değeri hesaplaması 29

Şekil 4.4. Duyarlılık değeri hesaplaması 29

Şekil 4.5. F1 puanı 29

Şekil 4.6. Karmaşıklık matrisi 30

Şekil 5.1. Doğruluk değerlerinin grafik ile gösterimi 41

Şekil 5.2. Kesinlik değerlerinin grafik ile gösterimi 42

Şekil 5.3. Duyarlılık değerlerinin grafik ile gösterimi 44

Şekil 5.4. F1 puanının grafik ile gösterimi 45

Şekil 5.5. Çapraz doğrulama değerlerinin grafik ile gösterimi 46

Şekil 5.6. Karşılaştırma grafiği 49

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklamalar
ABD	Amerika Birleşik Devletleri
API	Application Programming Interface
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
DT	Decision Tree
FN	False Negative
FP	False Positive
GNB	Gaussian Naive Bayes
GTB	Gradient Tree Boosting
JDBC	Java Database Connectivity
KNN	K-Nearest Neighbour
LR	Logistic Regression
MITM	Man in the Middle
NATO	North Atlantic Treaty Organization
NB	Naive Baiyes
RF	Random Forest
SQL	Structured Query Language
SVM	Support Vector Machine
TP	True Positive
TN	True Negative
XSS	Cross Site Scripting

1. GİRİŞ

Medeniyet ve bilim, ilk insandan itibaren nesilden nesile aktarılan bilgi üzerine inşa edilmiştir. Medeniyet ve bilimin gelişmesinde temel unsur olan bilgi, ilk başlarda sözlü olarak aktarılmakta olan bir olgu iken daha sonraları mağara duvarlarına resmedilmiş, yazının bulunması ile birlikte belirli bir kalıba evrilmiş ve teknolojinin hızlı gelişimi sayesinde disketlerde, kompakt disklerde, harici belleklerde ve veri tabanı sunucularında muhafaza edilmiş ve aktarılmıştır.

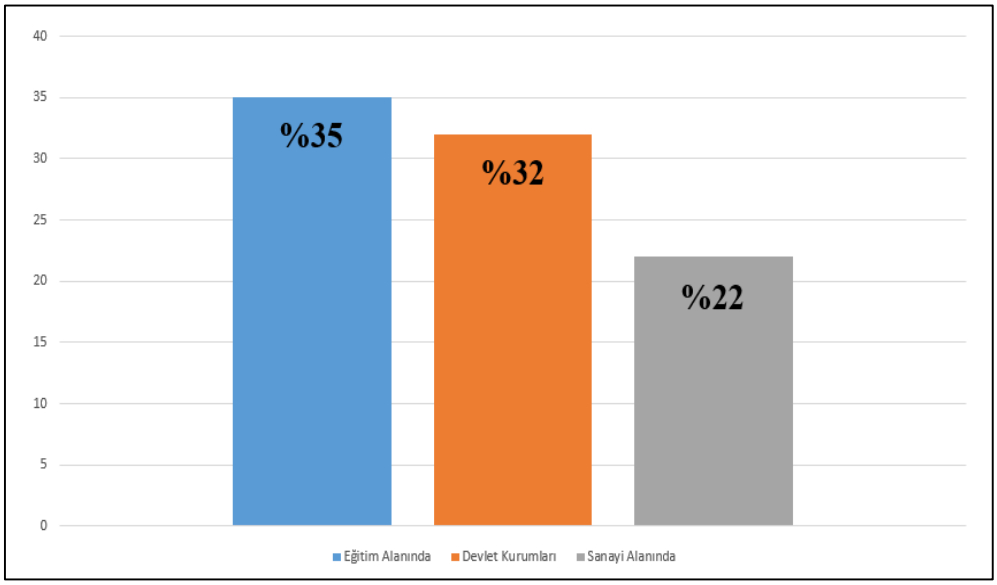
Veri tabanı sistemleri, birbirleriyle ilişkili bilgilerin ya da verilerin tablolar halinde yapılandırılmak suretiyle depolandığı elektronik sistemlerdir (Elmasri ve Navathe, 2010). Basılı olarak tutulması halinde belki binlerce cilde sığmayacak büyüklükte veri, veri tabanı sistemleri sayesinde, üstelik fiziki ortamda meydana gelebilecek bozulma ve yıpranmalara karşı bir tedbir olarak da, rahatlıkla elektronik ortamda muhafaza edilmektedir.

Günümüzde veri tabanı sistemlerinden, yemek siparişinden bankacılık işlemlerine, otel rezervasyon işlemlerinden e-devlet işlemlerine, sağlık işlemlerinden sigortacılığa kadar uzanan çok geniş bir yelpaze içerisinde istifade edilmektedir. Ayrıca bu sistemler üzerinde devlete ait gizli ve hizmete özel gizlilik dereceli veriler, kurum / kuruluşlara ait veriler ve kişilere ait nüfus bilgileri, ikametgâh bilgileri, iletişim bilgileri, fotoğraf ve parmak izi gibi özel nitelikli veriler muhafaza edilmektedir. Dolayısıyla insanlığa hizmet maksadıyla geliştirilen teknolojinin aynı zamanda kötü çıkarlar besleyenlerce istismar edildiği bir dünyada söz konusu bilgi ve verilerin korunması çok önemli bir mesele halini almıştır.

Teknolojiyi istismar ederek kötü niyetli çıkarlara hizmet edenler tarafından söz konusu bilgi ve verileri elde etmek maksadıyla birçok farklı teknik ve yöntem kullanılmaktadır. Bu çalışmaya konu olan SQL enjeksiyonu saldırıları, bu saldırı yöntem ve tekniklerinin en önemlileri arasında yer almaktadır.

Yazılım alanında siber güvenliğin iyileştirilmesi maksadıyla faaliyet gösteren bir kuruluş olan OWASP'ın 2021 yılında yayınladığı raporda SQL enjeksiyonu saldırıları, bilgisayar korsanları tarafından en çok tercih edilen saldırı yöntemlerinden birisi olarak gösterilmiştir (OWASP, 2022). Yine OWASP tarafından test edilen uygulamalarda 2021 yılı içerisinde

yaklaşık 274.000 SQL enjeksiyonu saldırı girişimi tespit edilmiştir (Crowdstrike, 2022). Yapılan başka bir araştırmada (Venturebeat, 2022) ise, eğitim alanındaki web uygulamalarının %35'inin, devlet kurumlarına ait web uygulamalarının %32'sinin ve sanayi alanında hizmet sunan web uygulamalarının %22'sinin SQL enjeksiyonuna karşı zafiyet barındırdığından bahsedilmektedir (Şekil 1.1.). Dolayısıyla yapılan araştırmalarla görüldüğü üzere SQL enjeksiyonu saldırılarına karşı tedbir almamanın hem kurumsal düzeyde hem de kişisel düzeyde ciddi sonuçlarının olabileceği söylenebilir.



Şekil 1.1. Venturebeat araştırması

1.1. Çalışmanın Amacı ve Kapsamı

SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespit edilmesi kapsamında literatürde birçok çalışma bulunmaktadır. Bunların arasında özellikle makine öğrenmesi uygulamalarının başladığı 1950'li yıllardan itibaren kullanılagelen geleneksel makine öğrenmesi metotları ile yapılan çalışmalara sıkça rastlanmaktadır. Literatürde ayrıca geleneksel makine öğrenmesi algoritmalarıyla yapılan çalışmaların yanında, 2000'li yılların başlarından itibaren geliştirilen topluluk (ensembl) öğrenme metotlarından olan artırma (boosting) algoritmaları ile SQL enjeksiyonu saldırılarının tespitine yönelik çalışmalar da mevcuttur.

Çalışma kapsamında SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespit edilmesinde, geleneksel makine öğrenmesi algoritmaları ile topluluk öğrenme yöntemlerinin

bir türü olan artırma yöntemleri karşılaştırılacaktır. Çalışmanın amacı, bu iki türden algoritmaların SQL enjeksiyonu saldırılarının tespit edilmesi konusunda birbirlerine karşı iyi olduğu yönleri ya da zayıf taraflarını ortaya koymaktır. Bu amaca yönelik olarak;

- SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespit edilmesinde kullanılmak üzere iki adet veri seti elde edilmiş,
- Makine öğrenmesi uygulamasında kullanılacak geleneksel makine öğrenmesi algoritmaları ve artırma algoritmaları belirlenmiş ve çalışma ortamında Pyhton programlama dili ile bu algoritmaların kodları hazırlanmış,
- Veri setleri hazırlanan algoritmalar ile denenmiş ve başarı sonuçları elde edilmiş,
- Elde edilen sonuçlar birbiri ile mukayese edilmiş,
- Sonuçlar literatürde bulunan benzer çalışma sonuçları ile karşılaştırılmıştır.

Çalışma kapsamında, ayrıca SQL enjeksiyonu saldırılarından korunmak maksadıyla kullanılan yöntemlere ilişkin literatürde yer alan çalışmalar, kodlama pratikleri ile tespit ve önleme sistemleri başlıkları altında ele alınmış ve SQL enjeksiyonu saldırılarının tespit edilmesi ile ilgili çalışmalarda kullanılmak üzere yeni bir veri seti önerilmiştir.

Bu çalışmanın bugüne kadar yapılan çalışmalardan farkı SQL enjeksiyonu saldırılarının makine öğrenmesi tespit edilmesi kapsamında geleneksel makine öğrenmesi yöntemleri ile artırma algoritmaları arasında bir karşılaştırma sunmasıdır.

Uygulamada kullanılmak maksadıyla veri seti elde etme imkânları oldukça sınırlıdır. Açık kaynaklarda SQL deyimi çok fazla bulunmasına rağmen birkaç açık kaynak haricinde SQL enjeksiyonu saldırısına yönelik SQL deyimi bulmak oldukça zordur. Bu durum yapılan çalışmanın en önemli sınırlılığıdır. Kurum / kuruluşların veri setlerini gizlilik ve güvenlik ile ilgili kaygılar taşımaları dolayısıyla paylaşmak istememelerinin yanında gerçek ortamda böyle bir veri setini elde etmenin oldukça zor olması bu durumun sebebi olarak gösterilebilir.

Çalışmanın ikinci bölümünde, SQL enjeksiyonu saldırılarına ilişkin literatürde yer alan çalışmalar incelenmiştir. Üçüncü bölümde siber tehditler ve siber tehdit araçlarına değinilmiş, SQL enjeksiyonu saldırıları açıklanmış ve SQL enjeksiyonu saldırılarından korunma yöntemleri belirtilmiştir. Dördüncü bölümde çalışmada izlenen yöntem ve

kullanılan materyaller ortaya konulmuştur. Bu bölüm içerisinde makine öğrenmesi türlerinden bahsedilmiş, makine öğrenmesi uygulamasında başarı ölçümünde kullanılan metrikler hakkında ve uygulamada kullanılan veri seti hakkında bilgi verilmiştir. Beşinci bölümde, SQL enjeksiyonu saldırısının makine öğrenmesi ile tespit edilmesine yönelik yapılan uygulamaya ilişkin sonuçlara yer verilmiş ve yapılan çalışma literatürde bulunan diğer çalışmalarla karşılaştırılmıştır. Altıncı bölümde, çalışmaya ait sonuçlar ve öneriler paylaşılmıştır.

2. LİTERATÜR TARAMASI

Literatürde SQL enjeksiyonu saldırıları, SQL enjeksiyonu saldırılarının tespiti ve önlenmesi konusunda bu çalışmayı da yakından ilgilendiren çalışmaların bazıları aşağıda sunulmuştur.

Avcı ve diğerleri (2021) tarafından gerçekleştirilen çalışmada ise bir önce bahsedilen çalışmadan farklı olarak SQL enjeksiyonu saldırısı bir uygulama ile canlandırılmış ve saldırı sonuçları analiz edilmiştir. Yapılan uygulama sonucunda SQL enjeksiyonu saldırısına karşı alınacak tedbirler; web formunda parametrik SQL sorgularının kullanılması, web formlarını içeren sayfalara girişlerde gerekli kontrollerin yapılması, veri tabanına gönderilen sorguların filtrelenmesi, veri tabanı yönetiminden sorumlu kişilerce kullanıcıları uygun yetkiler verilmesi ve veri silme / güncelleme / ekleme işlemlerinin sadece yetkili kullanıcılar tarafından yapılması şeklinde açıklanmıştır.

Hasan ve diğerleri (2019) tarafından yapılan çalışmada hem statik hem de dinamik analizin avantajlarından yararlanarak SQL enjeksiyonu saldırılarının tespitinde yukarıda bahsedilen çalışmalardan çok daha farklı bir yöntem ortaya konmuştur. Bu yöntemi geliştirmek için MATLAB uygulamasını kullanılmış, çok iyi analiz edilmiş SQL deyimlerinden faydalanılarak hazırlanan tek bir veri seti ile 23 farklı makine öğrenmesi sınıflandırıcısı test edilmiştir. 23 farklı makine öğrenmesi sınıflandırıcısının kullanılması hem makine öğrenmesi uygulamasının güvenirliliğinin artırılması bakımından hem de sınıflandırıcıların kıyaslanması bakımından çok önemlidir. Çalışmaları kapsamında ayrıca SQL enjeksiyonu saldırılarını tespit etmede kullanılan bir grafik kullanıcı arayüzü tasarlamışlardır. Önerdikleri sistem SQL enjeksiyonu saldırılarını %93.8 doğruluk oranı ile tespit etmiştir. Ayrıca veri tabanı sunucusu ile veri tabanı uygulaması arasına konumlandırmak suretiyle tasarladıkları grafik kullanıcı arayüzü vasıtasıyla SQL deyimlerinin veri tabanı erişimine izin verilip verilmeme durumu denetlenebilmektedir.

Krishnan ve diğerleri (2021) web uygulamalarının SQL enjeksiyonu saldırılarına karşı hassas olduğunu göz önünde bulundurarak makine öğrenmesi yöntemiyle SQL enjeksiyonu saldırılarını tespit eden bir sistem önermişlerdir. Sistemlerini sunucu tarafında veri tabanını saldırılardan korumak yerine kullanıcı tarafında bir filtre olarak tasarlamışlar ve 5 farklı makine öğrenmesi algoritması kullanmışlardır. Çalışma sonucunda evrişimli sinir ağı

algoritmaları (convolutional neural network algorithm) ile yapılan makine öğrenmesinin, %97 oranında doğrulukla saldırıları tespit ettiği görülmüştür.

Muhammad ve diğerleri (2021) yaptıkları çalışmada makine öğrenmesini yöntemini kullanarak web sitesi erişimi log kayıtları ile daha önceden bilinen kötü niyetli girişimleri karşılaştırmak suretiyle SQL enjeksiyonu saldırılarını tespit etmeye çalışmışlardır. Bu kapsamda önerdikleri çalışmayı çıkarım, eğitim ve tespit olmak üzere 3 safhaya ayırmışlardır. İlk safhada log erişimlerini incelemişler ve elde ettikleri veriyi test ve eğitim seti olarak sınıflandırmışlardır. Müteakiben makine öğrenmesi yoluyla bilgi tabanını (knowledge base) oluşturmuşlar ve son adımda test verilerini bilgi tabanı ile oluşturulan tespit sistemini kullanarak analiz etmişlerdir. Toplanan veri kümeleri aralığında ilk test için %93 diğer 4 test için %100 olmak üzere oldukça yüksek doğruluk değerleri elde etmişlerdir.

Al-Maliki ve Jasim (2022) SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespiti konusunda literatürde var olan çalışmaları kullanılan metot, kullanılan algoritmalar ve bu algoritmaların doğruluk oranları açısından karşılaştırarak literatüre katkı sağlamışlardır.

Halfond ve diğerleri (2006) tarafından SQL enjeksiyonu ile ilgili bugüne kadar yapılan çalışmalar çok daha kapsamlı bir şekilde gözden geçirilmiş, saldırı türleri belirlenmiş, her bir saldırı türü için tanımlamalar yapılmış ve saldırıların nasıl gerçekleştirildiğine dair örnekler sunulmuştur. Yapılan çalışmada, SQL enjeksiyonu saldırıları ilk defa bu kadar detaylı olarak ele alınmış ve enjeksiyon mekanizmaları kullanıcı girdisi ile yapılan enjeksiyonlar, çerezler (cookies) vasıtasıyla yapılan enjeksiyonlar, sunucu değişkenleri vasıtasıyla enjeksiyonlar ve ikinci dereceden enjeksiyonlar (second-order injection) olarak sınıflandırılmıştır. SQL enjeksiyonu saldırı türleri ise totolojiler (tautologies), illegal/mantık hatalı sorgular (illegal/logically incorrect queries), birleşim sorguları (union queries), çıkarımlar (inferences), saklı yordamlar (stored procedures) ve alternatif kodlama (alternate encodings) başlıkları altında sınıflandırılmışlardır. SQL enjeksiyonuna karşı alınacak önlemler ise savunmaya yönelik kodlama pratikleri ve tespit / önleme teknikleri olarak ikiye ayrılmış, sonuç bölümünde bütün bu teknikleri karşılaştırmalı olarak incelenmiştir. Sonuç bölümünde yaptıkları değerlendirmede birden fazla SQL enjeksiyonu tekniğinin iyi olan yönlerini kullanan bilgisayar korsanlarınca gerçekleştirilen SQL enjeksiyonu saldırılarının başarıya ulaşma ihtimalinin yüksek olduğunu vurgulamışlardır. Halfond ve arkadaşlarının yaptığı çalışma, SQL enjeksiyonu mekanizmaları ile SQL enjeksiyonu saldırılarını

sınıflandırması ve SQL enjeksiyonu saldırılarını tespit / önleme mekanizmalarını karşılaştırması açısından oldukça önemlidir.

Literatürde yer alan çalışmalara ilişkin bilgiler aşağıda sunulmuştur (Çizelge 2.1).

Çizelge 2.1. Literatürde yer alan çalışmalar

Çalışma	Kullanılan Yöntem	Açıklamalar
Avcı ve diğerleri (2021)	SQL enjeksiyonu saldırısı bir uygulama ile canlandırılmış ve saldırı sonuçları analiz edilmiştir	Sonuç bölümünde SQL enjeksiyonu saldırılarına karşı alınacak önlemler sıralanmıştır.
Hasan ve diğerleri (2019)	Çok iyi analiz edilmiş SQL deyimlerinden faydalanılarak hazırlanan tek bir veri seti ile 23 farklı makine öğrenmesi sınıflandırıcısı test edilmiştir	Önerdikleri sistem SQL enjeksiyonu saldırılarını %93.8 doğruluk oranı ile tespit etmiştir.
Krishnan ve diğerleri (2021)	Kullanıcı tarafında bir filtre olarak tasarlamışlar ve 5 farklı makine öğrenmesi algoritması kullanmışlardır.	Evrimli sinir ağı algoritmaları ile yapılan makine öğrenmesinin, %97 oranında doğrulukla saldırıları tespit ettiği görülmüştür.
Muhammad ve diğerleri (2021)	Makine öğrenmesini yöntemini kullanarak web sitesi erişimi log kayıtları ile daha önceden bilinen kötü niyetli girişimleri karşılaştırmak suretiyle SQL enjeksiyonu saldırılarını tespit etmeye çalışmışlardır.	Uygulama sonucunda ilk test için %93 diğer 4 test için %100 doğruluk oranı yakalamışlardır.
Al-Maliki ve Jasim, (2022)	Literatürde yer alan çalışmaları değerlendirmişlerdir.	Makine öğrenmesi uygulaması ile yapılan çalışmaların doğruluk oranı ve diğer değerlere göre karşılaştırmasını sunmuşlardır.
Halfond ve diğerleri (2006)	SQL enjeksiyonu ile ilgili bugüne kadar yapılan çalışmalar çok kapsamlı bir şekilde gözden geçirilmiştir.	Birden fazla SQL enjeksiyonu tekniğinin iyi olan yönlerini kullanan bilgisayar korsanlarınca gerçekleştirilen SQL enjeksiyonu saldırılarının başarıya ulaşma ihtimalinin yüksek olduğunu vurgulamışlardır.

3. KAVRAMSAL ÇERÇEVE

Bu bölümde siber tehditler ve siber tehdit araçlarına ilişkin tanımlamalar yapılmış, veri tabanı sistemleri ve yapılandırılmış sorgulama dili (SQL) hakkında bilgi verilmiş, literatürde yer alan SQL enjeksiyonu saldırılarına karşı alınacak tedbirler pasif tedbirleri içeren kodlama pratikleri ve aktif tedbirleri içeren önleme ve korunma yöntemleri başlıkları altında derlenmiştir.

3.1. Siber Tehditler ve Siber Tehdit Araçları

Bilgisayar korsanları tarafından siber saldırı gerçekleştirmek maksadıyla birçok farklı teknik ve yöntem kullanılmaktadır. Bunlar arasında hizmet engelleme ve dağıtık hizmet engelleme saldırısı, ortadaki adam saldırısı, kötücül yazılım (malware) saldırısı, oltalama (phishing) saldırısı, kaba kuvvet (brute force) saldırısı, siteler arası kod çalıştırma (cross site scripting), sıfırıncı gün (zero-day) saldırısı ve daha birçok yöntem yer almaktadır. Bilgisayar korsanları çok ciddi bir hazırlık sürecinin ardından kendi hedefleri doğrultusunda uygulama tasarımındaki zafiyetleri, sistemdeki donanımsal / yazılımsal zafiyetleri ve kritik sistemlere çevrimiçi erişim imkânlarını incelemeyi müteakip uygulayacakları teknik ve yönteme karar vermektedirler. Bu bölümde söz konusu saldırı yöntemlerinden bazıları hakkında genel bir bilgilendirme yapılmıştır.

3.1.1. Hizmet engelleme ve dağıtık hizmet engelleme saldırıları

Hizmet engelleme (DoS) ve dağıtık hizmet engelleme (DDoS) saldırıları, web uygulamalarının ve servislerinin hizmet dışı bırakılmasına yönelik gerçekleştirilen saldırılardır (KAREL Elektronik, 2023). Bu tür saldırılar, kurum ve kuruluşların sağladıkları hizmete ve uygulamalara erişimin engellenmesi nedeniyle genellikle en çok korktukları ve buna rağmen en sık karşılaştıkları saldırılar arasında yer almaktadır.

Bu tür saldırılar bilgisayar korsanları tarafından geçmiş yıllarda küçük çapta gerçekleştirilmekteyken geniş bant ağ teknolojisinin ilerlemesi ve internetin gündelik hayatımızda önemli bir yer edinmesinin yanında ağa bağlı cihaz sayısında meydana gelen artış ile birlikte çok daha büyük boyutlara ulaşmıştır (Atasever, Özçelik ve Sağıroğlu, 2019).

1999 yılında Kosova'da bulunan NATO kuvvetlerine ve 2007 yılında Estonya'da gerçekleştirilen saldırılar hizmet engelleme ve dağıtık hizmet engelleme saldırılarının birer örneğini teşkil etmektedir. Dağılma sürecine giren Yugoslavya'nın Kosova'da gerçekleştirdiği insanlık dışı katliama NATO Kuvvetleri tarafından 1999 yılında müdahale edilmiştir. Ancak bu müdahale karşısında NATO kuvvetlerine ait sistemler siber saldırılara maruz kalmıştır. NATO'ya karşı yapılan saldırılarda bilgisayar korsanları tarafından dağıtık hizmet engelleme saldırıları kullanılmış ve bunun yanında binlerce zararlı bilgisayar virüsünü içeren e-posta hizmetleri vasıtası ile oltalama saldırıları düzenlenmiştir (Kara, 2013). Gerçekleştirilen siber saldırılar ile NATO'nun resmi web sitesi bir müddet hizmet dışı bırakılmış ve NATO'da görevli personelin e-posta hesapları günlerce kapalı kalmıştır. Bu durum NATO'nun muhabere sistemini ve iletişim teknolojilerini sekteye uğratmıştır.

2007 yılında ise siber saldırıların hedefinde yer alan ülke Estonya olmuştur. Ülkenin en büyük bankasına ait web uygulamasının yanı sıra, Estonya Meclisi'ne ait web sitesi, bakanlıklara ait sayfalar, siyasi parti web sayfaları, önde gelen haber kuruluşlarının web siteleri, ekonomide öncü kuruluşların web tabanlı uygulamaları saldırıların odağında yer almıştır (Kara, 2013). Saldırılar neticesinde ülkenin bilgi sistemleri ve internet alt yapısı çok büyük maddi zarara uğramış ve bunun yanında bankacılık sektörü, büyük finans şirketleri, devlet kurumları ve borsanın çalışamaması neticesinde başta ekonomi olmak üzere bütün faaliyetler durma noktasına gelmiştir.

3.1.2. Ortadaki adam saldırısı

Ortadaki adam (Man In The Middle-MITM) saldırı, bir bilgisayar ağı üzerinden birbirleriyle iletişim halinde olan tarafların gönderdiği paketlerin araya giren bir saldırgan tarafından izlenmesi, manipüle edilmesi, değiştirilmesi gibi amaçlarla yapılan bir saldırı yöntemidir (Güner, 2021). Ortadaki adam saldırılarının en yaygın bilinen türleri arasında ARP zehirlemesi, DNS yanıltması, MAC adresi saldırısı, Wi-Fi dinleme ve oturum ele geçirme yer almaktadır.

3.1.3. Kötücül yazılım saldırısı

Kötücül yazılım (malware) saldırıları, sızdığı bilgisayar sisteminde veya bilgisayar ağında bulunan cihazlarda zarara yol açmak ve çalışmaları sekteye uğratmak, engellemek

maksadıyla hazırlanan bir yazılım çeşidi ve saldırı yöntemidir (Canbek, 2005). Truva atları (Trojan horse), virüsler, telefon çeviriciler (dialers), solucanlar (worms), arka kapılar (back door), klavye dinleme sistemleri (keyloggers), casus yazılımlar (spyware), tarayıcı soyma (browser hijacking) ve fidye yazılım (ransomware) gibi çok sık karşılaşılan ve ismi sıkça duyulan saldırılar en çok bilinen kötücül yazılım saldırılarıdır (Canbek, 2005).

Yapılan bir çalışmada; kötücül yazılım saldırılarına karşı verilen mücadele bir teknoloji savaşına benzetilmiş, teknolojiyi kendi çıkarları için kullanan ülkeler, kurumlar ve kötü niyetli kullanıcıların teknolojiyi kötücül yazılımlar geliştirmek için kullanırken, diğer bir grubun ise kötücül yazılımlara karşı benzer teknolojileri kullanarak karşılık verme gayreti içerisinde olduğu belirtilmiştir (Grimes, 2001).

Kötücül yazılım saldırılarının en iyi bilinen örneği 2009 yılında gerçekleştirilen Stuxnet saldırısıdır. 2009 yılında Amerika Birleşik Devletleri tarafından İran'ın Natanz nükleer tesislerine yönelik olarak gerçekleştirildiğine inanılmaktadır. Kötücül yazılım ile gerçekleştirilen saldırı ancak bir yıl sonra fark edilebilmiştir. Zararlı yazılımın o zamana kadar tespit edilen yazılımlardan daha karmaşık bir teknoloji ile oluşturulduğunun anlaşılması üzerine saldırının kaynağını bulmak üzere birçok yazılım ve siber güvenlik şirketinin katılımıyla geniş çaplı bir çalışma yürütülmüştür (Çelik, 2013).

Teknolojide meydana gelen gelişmeleri yakinen takip etmesi ve bu gelişmelere çok çabuk uyum göstermesi sebebiyle kötücül yazılım saldırıları en tehlikeli saldırı türleri arasında yer almaktadır.

3.1.4. Kaba kuvvet saldırısı

Kaba kuvvet (brute force) saldırılarının hedefi, sisteme sızarak yönetici (administrator) yetkisine erişim sağlamaktır. Bu saldırılar basit olarak şifre kırma saldırıları olarak da tanımlanabilir. Bu saldırılar, basit deneme yanılma yöntemlerini içerebileceği gibi çok daha gelişmiş uygulama ve araçların kullanıldığı karmaşık yöntemleri de ihtiva edebilir. Son yıllarda en etkili siber saldırı yöntemleri arasında bulunan kaba kuvvet saldırıları alınan önlemlere rağmen, yazılımda bulunan hatalar, bağlanılan ağlar, güvenlik zafiyetleri ve dikkatsizlik sebebiyle tehdit olmaya devam etmektedir (Kara İ. , 2019).

3.1.5. Oltalama saldırısı

Oltalama (phishing) saldırısı, bilgi teknolojilerinin sağladığı imkânlardan yararlanarak hedef kişilerin ya da kişiler üzerinden kurum ve kuruluşların tuzağa düşürülmek suretiyle bir takım bilgilerin elde edilmesi ve bu bilgilerin daha sonra bir takım menfaatler doğrultusunda kötü amaçlarla kullanılması olarak tanımlanmaktadır (Ünver ve Mirzaoğlu, 2011). Oltalama saldırıları bilgisayar korsanları ve kötü niyetli kişilerce kişisel bilgileri elde ederek karşılığında para ya da menfaat temin etmek maksadıyla son yıllarda en çok tercih edilen saldırı yöntemlerindendir. Özellikle basında ve medyada oltalama saldırılarına ilişkin haberler sıklıkla yer almaktadır.

Siber saldırı türlerinden birisi de aşağıda detaylı olarak anlatılacağı üzere SQL enjeksiyonu saldırılarıdır.

3.2. SQL Enjeksiyonu Saldırıları

Yazılım alanında siber güvenliğin iyileştirilmesi maksadıyla faaliyet gösteren bir kuruluş olan OWASP'ın 2021 yılında yayınladığı raporda SQL enjeksiyonu saldırıları, bilgisayar korsanları tarafından en çok tercih edilen saldırı yöntemlerinden birisi olarak gösterilmiştir (OWASP, 2022). Yine OWASP tarafından test edilen uygulamalarda 2021 yılı içerisinde yaklaşık 274.000 SQL enjeksiyonu saldırı girişimi tespit edilmiştir (Crowdstrike, 2022). Ayrıca yapılan araştırmalarla eğitim alanındaki web uygulamalarının %35'inin, devlet kurumlarına ait web uygulamalarının %32'sinin ve sanayi alanında hizmet sunan web uygulamalarının %22'sinin SQL enjeksiyonuna karşı zafiyet barındırdığı görülmüştür (Venturebeat, 2022).

SQL enjeksiyonu saldırısı, veri tabanına dayalı çalışan web uygulamalarının zafiyetlerini SQL deyimlerinin içerisine yerleştirilen birtakım ifadelerle istismar etmek suretiyle gerçekleştirilen bir çeşit siber saldırı türüdür (Laval, Sultan ve Shakiru, 2016). Yapılan bir başka tanıma göre ise SQL enjeksiyonu saldırıları, zararlı ifadeler içermeyen SQL deyimlerinin içerisine bir takım semboller, deyimler ve ifadeler yerleştirerek sistemin manipülasyon edilmesi olarak görülmektedir (Avcı, Koca ve Atasoy, 2021).

SQL enjeksiyonu saldırıları ilk olarak 1998 yılında web güvenliği ile ilgili yayın yapan bir dergi sayesinde duyulmuştur (Forristal, 1998). Günümüzde ise SQL enjeksiyonu saldırıları web uygulamalarına yönelik en tehlikeli saldırılar arasında kendisine yer edinmiştir. Çünkü hassas ve kişisel verilerin tutulduğu veri tabanları, SQL enjeksiyonu saldırılarına karşı zafiyeti bulunan web uygulamaları vasıtasıyla bilgisayar korsanlarının tam yetki ile erişimine açılabilmektedir. Kimlik hırsızlığı, gizli bilgilerin çalınması ve sahtekârlık böyle bir durumun olası sonuçlarıdır.

SQL enjeksiyonu saldırıları, veri tabanına dayalı çalışan web uygulamalarının zafiyetlerini SQL deyimlerinin içerisine yerleştirilen birtakım ifadelerle istismar etmek suretiyle gerçekleştirilen bir siber saldırı türü olması dolayısıyla kullanıcı tarafından kimlik bilgisi, parola vb. şeklinde girdi alan her türden web uygulaması için SQL enjeksiyonu saldırıları bir tehdit teşkil etmektedir (Halfond, Viegas ve Orso, 2006).

SQL enjeksiyonu saldırılarının temel nedeni, kullanıcı tarafından yapılan girdinin tam olarak doğrulanmamasıdır (Alwan ve Younis, 2017). Halbuki kod yazım sürecinde geliştiricilere veri doğrulamanın uygun bir şekilde ve tam olarak yapılması maksadıyla birçok kaynak sunulmuştur (Howard ve David, 2003). Bu kaynaklarda yer alan tekniklerin titiz ve dikkatli bir şekilde uygulanması ile bir dereceye kadar SQL enjeksiyonu saldırılarının önleneceği düşünülebilir. Ancak uygulama geliştiricilerin bir insan olduğu ve her zaman hata yapmaya meyilli olabileceği göz ardı edilmemelidir. Ayrıca çok farklı türde ve çok fazla sayıda SQL enjeksiyonu saldırısı tekniği olduğu düşünüldüğünde her bir saldırı tekniği için farklı bir tespit ve önleme metoduna ihtiyaç duyulmaktadır (Halfond, Viegas ve Orso, 2006). Şimdi bu bölümde SQL enjeksiyonu saldırı türleri ve teknikleri açıklanacaktır. SQL enjeksiyonu saldırılarının tespit ve önlenmesi ile ilgili bilgilere ilerleyen bölümlerde yer verilecektir.

3.2.1. Enjeksiyon mekanizmaları

Kötü niyetli SQL deyimleri uygulamaya birçok farklı yöntem ile enjekte edilebilir. Bu yöntemler aşağıda açıklanmıştır.

Bu yöntemlerden ilki SQL enjeksiyonunun kullanıcı girdisi vasıtasıyla yapılmasıdır (Jemal, Omar, Habib ve Mahfoudhi, 2020). Bir web uygulaması kullanıcı girdisini genellikle bir web formu şeklinde alır. Bu form içerisinde, bilgisayar korsanlarının istismar etmesine

imkân sağlayan, bir metin alanı barındırır. Web uygulamalarını hedef alan SQL enjeksiyonu saldırılarında bu form web uygulamasına HTTP GET ya da HTTP POST metotları vasıtası ile gönderilir.

Bir diğer yöntem SQL enjeksiyonu saldırılarının çerezler (cookies) vasıtasıyla enjekte edilmesidir (Jemal, Omar, Habib ve Mahfoudhi, 2020). Çerezler, kullanıcı tarafında web uygulamaları ile ilgili durum bilgilerini içeren klasörlerdir. Çerezlerin içeriği, kullanıcı tarafında tutulması dolayısıyla kötü niyetli kullanıcılar tarafından bir SQL enjeksiyonu saldırısında kullanılmak üzere değiştirilebilir.

Başka bir yöntem de SQL enjeksiyonu deyiminin sunucu değişkenleri ile enjekte edilmesidir (Halfond, Viegas ve Orso, 2006). Sunucu değişkenleri HTTP ve çevre değişkenleri gibi bir dizi değişkeni içermekte olup bu değişkenler kullanım istatistiklerinin log kaydı alınması ve tarama eğilimlerinin belirlenmesi gibi çeşitli yollarla kullanılmaktadır. Eğer bu değişkenler herhangi bir denetimden geçirilmezlerse SQL enjeksiyonu saldırıları için kaynak teşkil edebilirler.

Son olarak kullanılan diğer yöntem ikinci dereceden enjeksiyonlardır (Jemal, Omar, Habib ve Mahfoudhi, 2020). Bilgisayar korsanları tarafından önceden belirlenen bir girdinin SQL enjeksiyonu saldırısını tetiklemesi suretiyle gerçekleştirilir.

3.2.2. SQL enjeksiyonu saldırılarının amaçları

SQL enjeksiyonu saldırıları, bilgisayar korsanları tarafından genellikle aşağıdaki amaçları gerçekleştirmek maksadıyla yapılır.

Enjekte edilebilir parametreleri belirlemek (Halfond, Viegas ve Orso, 2006):

Bilgisayar korsanları daha sonra yapmayı planladıkları kapsamlı bir SQL enjeksiyonu saldırısına hazırlık yapmak amacıyla önceden düzenleyecekleri bir SQL enjeksiyonu saldırısı ile enjekte edilebilir parametreleri belirlemeye çalışabilirler. Bu parametreler örneğin bir kütüphane otomasyon sistemi için “KitapAdi”, “KitapYazari” vb. şeklinde olabilir.

Veri tabanı hakkında ipuçları elde etmek (Halfond, Viegas ve Orso, 2006):

Bilgisayar korsanları daha sonra yapmayı planladıkları kapsamlı bir SQL enjeksiyonu saldırısına hazırlık yapmak amacıyla önceden düzenleyecekleri bir SQL enjeksiyonu saldırısı ile veri tabanı hakkında ipuçları elde etmeye çalışırlar. Örneğin Microsoft SQL Server T-SQL kullanırken Oracle SQL Server tarafından PL/SQL kullanılmaktadır (Alakoç Burma, 2009). Dolayısıyla böyle bir farklılıktan yola çıkarak ilk olarak veri tabanının türü ve versiyonunu belirlemek öncelikli hedef olacaktır.

Veri tabanı şemasını belirlemek (Jemal, Omar, Habib ve Mahfoudhi, 2020):

Veri tabanında başarılı bir şekilde veri elde edebilmek için öncelikle veri tabanının yapısı belirlenmelidir. Tablo isimleri, tablolarda bulunan sütunlar, veri tipleri bilgisayar korsanlarının önem atfettiği kıymetli bilgilerdir. Dolayısıyla bilgisayar korsanları bir SQL enjeksiyonu saldırısı düzenleyerek bu bilgileri elde etmek isteyebilirler.

Verileri elde etmek (Nathan, Kumar ve Vaidyanathan, 2012):

Birçok SQL enjeksiyonu saldırısı veri tabanında tutulan kredi kartı bilgisi, nüfus bilgileri vb. türden kişisel bilgileri ya da çeşitli büyüklüklerdeki şirketlere / kurumlara ait bilgileri ele geçirerek bu yolla bilgisayar korsanları tarafından ekonomik çıkar ya da başka türden bir menfaat sağlamak maksadıyla düzenlenmektedir (Avcı, Koca ve Atasoy, 2021).

Kimlik doğrulamasını atlatmak (Halfond, Viegas ve Orso, 2006):

SQL enjeksiyonu saldırıları bilgisayar korsanları tarafından hedef web uygulamasında bulunan kimlik doğrulaması adımını geçmeyi ve böylelikle başka bir kullanıcıya ait yetkileri ve ayrıcalıkları elde etmek maksadıyla gerçekleştirilebilir.

Uzak komutları yürütmek (Jemal, Omar, Habib ve Mahfoudhi, 2020):

Bilgisayar korsanları düzenleyecekleri bir SQL enjeksiyonu saldırısı ile saklı yordamları ya da bir takım fonksiyonları gelişigüzel bir şekilde tetikleyerek veri tabanını hizmet dışı bırakmayı amaçlayabilirler.

Veri tabanında ayrıcalık elde etmek (Jemal, Omar, Habib ve Mahfoudhi, 2020):

SQL enjeksiyonu saldırıları, bilgisayar korsanları tarafından daha yüksek yetki ve ayrıcalıklara sahip bir kullanıcıya ait yetkileri ve ayrıcalıkları elde etmek maksadıyla gerçekleştirilebilir. Kök kullanıcı ya da yöneticiye ait ayrıcalıkların ve yetkilerin bilgisayar korsanlarınca elde edilmesi çok ciddi sonuçlar doğurabilmektedir.

Veri tabanında değişiklik yapmak (Nathan, Kumar ve Vaidyanathan, 2012):

Bilgisayar korsanları kurum, şirket ya da kişilere zarar vermek maksadıyla düzenleyecekleri bir SQL enjeksiyonu saldırısı ile veri tabanında bulunan bilgileri değiştirebilir ya da manipüle edebilirler.

3.2.3. SQL enjeksiyonu türleri

Bu bölümde bilgisayar korsanları tarafından çoğunlukla kullanılan SQL enjeksiyonu saldırı türleri üzerinde durulacaktır.

Totoloji:

Veri tabanına gönderilen SQL sorgusunun veri tabanı tarafından her zaman doğru şartı sağlayan bir sorgu (1=1 ya da "--") şeklinde değerlendirilmesi esasına dayanır (Alwan ve Younis, 2017). Bu tür sorgular bilgisayar korsanları tarafından genellikle kimlik doğrulamasını atlamak ya da veri elde etmek istenildiğinde kullanılır. Bu tür sorgular WHERE şartının istismar edilmesi suretiyle gerçekleştirilir (Farooq, 2021). Her zaman doğru sonuç döndürmesini istediğimiz şart bu kısmın içerisine yazılır. Aşağıda bu tür bir sorgu için örnek görülmektedir (Şekil 3.1.).

```
SELECT KullaniciID FROM Kullanici WHERE Parola = ' ' OR 1=1
```

Şekil 3.1. Totoloji

Kör SQL enjeksiyonu:

Kör SQL enjeksiyonu, bilgisayar korsanları tarafından veri tabanına gönderilen çeşitli sorgulara dönen yanıtları gözlemleyerek sonraki basamaklarda saldırının gidişatına karar verildiği bir enjeksiyon türüdür (Singh, 2016). Veri tabanı hakkında bilgisayar korsanlarının neredeyse hiçbir bilgiye sahip olmamasından ötürü en zor saldırı türü olarak değerlendirilir. Bu saldırı türü bilgisayar korsanları tarafından genellikle veri tabanından sözdizimi hatası döndüğü zaman kullanılır (Mishra, 2019).

Birleşik sorgular:

Bu enjeksiyon türü, bilgisayar korsanlarının sorguyu istedikleri gibi yönlendirmelerine imkan tanıyan bir enjeksiyon türüdür. Enjekte edilmek istenen ifade UNION SELECT ile birleştirilen sorgunun ikinci kısmında yer alır (Alwan ve Younis, 2017). Birleşik sorguya ait bir örnek aşağıda görülmektedir (Şekil 3.2.).

```
SELECT KullaniciID FROM Kullanici WHERE KullaniciAdi = ‘ ’ UNION
SELECT kartNo FROM KartBilgisi WHERE KullaniciNo=26267 – AND
Parola = ‘ ’
```

Şekil 3.2. Birleşik sorgular

Saklı yordamlar:

Bu tür SQL enjeksiyonu saldırıları, veri tabanını hizmet dışı bırakmak ya da veri tabanında ayrıcalık elde etmek maksadıyla gerçekleştirilir. Bu tür saldırılarda saklı yordam olarak kullanılabilecek “EXEC”, “SHUTDOWN” gibi anahtar kelimeler bulunarak tetiklenmeye çalışılır (Farooq, 2021). Aşağıda bu türden sorgulara bir örnek görülmektedir (Şekil 3.3.).

```
SELECT * FROM Kullanici WHERE KullaniciAdi = ‘Emre ’ AND Parola =
‘1234abcd’ ; EXEC ;
```

Şekil 3.3. Saklı yordamlar

Çıkarım:

Bu enjeksiyon türünde ilk olarak bilgisayar korsanları tarafından enjeksiyona maruz bırakılabilecek değişkenler belirlenmeye çalışılır. Bunun için veri tabanına doğru/yanlış şeklinde veri döndürmesi için bir dizi sorgu gönderilir. Sorgulara dönen cevaplardan bir çıkarım yapılır. Sonraki adımda yapılan çıkarımlarla değişkenlerin belirlenmesinin ardından farklı türde bir SQL enjeksiyonu saldırısı uygulanır. Bu tür saldırılar, genellikle kör SQL enjeksiyonu saldırısı ile gerçekleştirilir (Alwan ve Younis, 2017).

Alternatif kodlama:

Bu saldırı türünde bilgisayar korsanları sorgularını ASCII ve Unicode benzeri alternatif kodlama teknikleri kullanarak değiştirmek suretiyle saldırının tespit edilmesinden sakınmaya çalışırlar. Bu saldırı türü diğer saldırı türlerinde birisi ile birleştirildiğinde daha güçlü bir saldırı düzenlenmesi olasıdır (Alwan ve Younis, 2017). Aşağıda bu tür saldırılar için kullanılabilecek örnek bir SQL sorgusu görülmektedir (Şekil 3.4.).

```
SELECT KullaniciID FROM Kullanici WHERE KullaniciAdi = ‘  ’ AND Parola
= ‘0’ ; EXEC(char(0x736875746446f776e)) ;
```

Şekil 3.4. Alternatif kodlama

Geçersiz ya da mantık olarak hatalı sorgular:

Bu türde veri tabanına mantık olarak hatalı ya da geçersiz bir dizi sorgu gönderilir ve bu sorguların sonucunda veri tabanın bir hata mesajı göndermesi beklenir (Farooq, 2021). Bu hata mesajları genellikle bilgisayar korsanları tarafından kullanılabilecek çok değerli bilgiler içerebilmektedir. Aşağıda bu türden sorgular için bir örnek görülmektedir (Şekil 3.5.).

```
SELECT KullaniciID FROM Kullanici WHERE KullaniciAdi = ‘123124’ AND
Parola = ‘145132’ AND CONVERT (CHAR,NO)
```

Şekil 3.5. Geçersiz / mantık hatalı sorgu

3.2.4. SQL enjeksiyonu saldırılarında ve zafiyet tespitinde kullanılan araçlar

Bilgisayar korsanları tarafından SQL enjeksiyonu saldırılarını gerçekleştirmek ya da sızma testi uzmanları tarafından bir uygulamanın SQL enjeksiyonuna karşı zafiyetlerini belirlemek maksadıyla kullanılan araçlardan bir kısmı aşağıda sunulmuştur.

BBQSQL:

Kör SQL enjeksiyon saldırılarında kullanılan ve Pyhton programlama dili ile yazılmış bir araçtır (Mozumder, Mahi ve Whaiduzzaman, 2017). BBQSQL uygulamasına ait bir görsel aşağıda görülmektedir (Şekil 3.6.).



Şekil 3.6. BBQSQL (KitPloit, 2023)

Leviathan:

Tarama, kaba kuvvet saldırısı, SQL enjeksiyon saldırısı tespiti ve bir takım istismar yeteneklerini gerçekleştirmeye imkan tanıyan Python tabanlı bir denetim aracıdır (Chica-Manjarrez, Marin-Lopez, Diaz-Sanchez ve Almenares-Mendoza, 2020). Leviathan aracı sayesinde aynı anda birkaç sistemde taramalar yapılabilmektedir. Leviathana ait bir görsel aşağıda görülmektedir (Şekil 3.7.).



Şekil 3.7. Leviathan (Kali Linux Tutorials, 2023)

NoSQLMap:

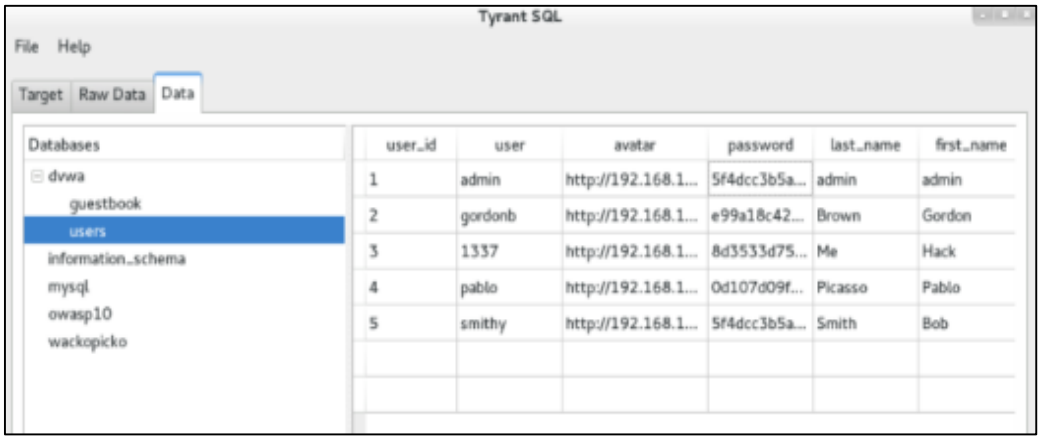
NoSQL veri tabanlarını hedef alan SQL enjeksiyonu saldırılarında kullanılabilen açık kaynak kodlu bir araçtır (Holik ve Neradova, 2017). NoSQLMap uygulamasına ait bir görsel aşağıda görülmektedir (Şekil 3.8.).



Şekil 3.8. NoSQLMap (Kali Linux Tutorials, 2023)

Tyrant SQL:

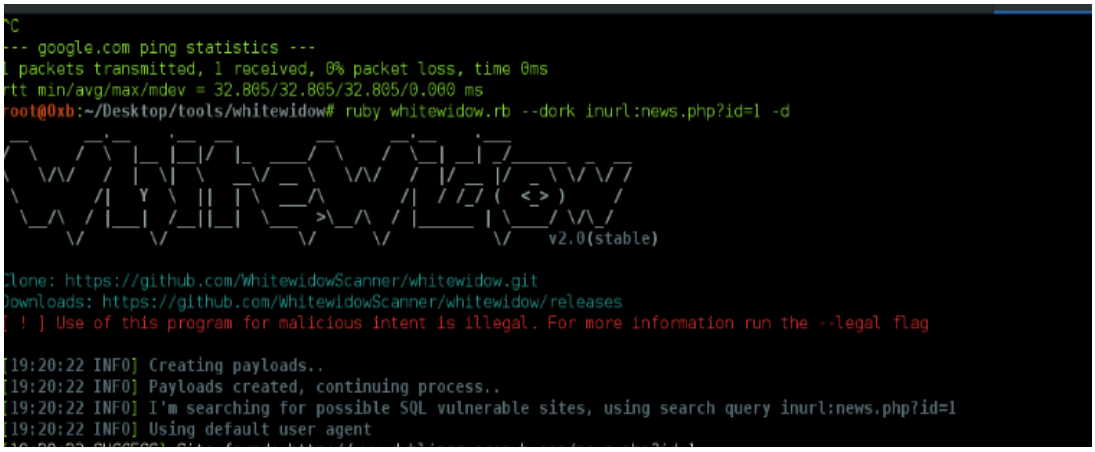
Tyrant SQL, SQLMap'e benzer bir Python tabanlı SQL enjeksiyon aracıdır. Grafik arayüzü oldukça basittir. Bu durum savunmasız bağlantıların analizinde ve zayıflıkların nerede olduğunun belirlenmesinde kolaylık sağlamaktadır (Serverwatch, 2022). Tyrante ait bir görsel aşağıdadır (Şekil 3.9.).



Şekil 3.9. Tyrant (Serverwatch, 2022)

WhiteWidow:

Açık kaynak kodlu bir SQL enjeksiyonu güvenlik açığı tarama aracıdır. Otomasyona dayalı olduğundan dolayı WhiteWidow sayesinde uzun bir dosya listesinden hızla geçilebilir veya potansiyel olarak savunmasız web siteleri için Google arama motoru üzerinde araştırma yapılabilir (Serverwatch, 2022). WhiteWidowa ait bir görsel aşağıdadır (Şekil 3.10.).



Şekil 3.10. WhiteWidow (KitPloit, 2023)

3.3. SQL Enjeksiyonu Saldırılarına Karşı Korunma Yöntemleri

Literatürde araştırmacılar tarafından SQL enjeksiyonu saldırılarının tespiti ve önlenmesi maksadıyla birçok teknik ileri sürülmüştür. Bu teknikler, yazılım geliştirme esnasında uygulanan kodlama pratiklerinden tamamen otomasyona dayanan tespit ve önleme

sistemlerine kadar geniş bir yelpaze içerisinde yer almaktadır. Bu bölümde söz konusu teknikleri özetlemeye çalışacağım.

3.3.1. Kodlama pratikleri

SQL enjeksiyonu saldırılarının özünde yazılım sürecinde girdi kontrolünün hiç yapılmaması ya da tam olarak yapılmaması bulunmaktadır. Dolayısıyla yazılım geliştirme sürecinde girdi kontrolünün yapılması yönünde tedbirler alınması SQL enjeksiyonu saldırılarının önlenmesinde çok önemlidir.

Girdi türünün kontrolü:

SQL enjeksiyonu saldırıları bir string ya da sayısal veri türünde parametrenin içerisine komut ya da bir takım ifadeler enjekte etmek suretiyle gerçekleştirilmektedir. Dolayısıyla bu parametrelerin basit bir kontrolü bile SQL enjeksiyonu saldırılarını birçok durumda engelleyebilmektedir. Örneğin sayısal ifade şeklinde belirlenen veri türüne farklı türden bir karakter ya da ifade yerleştirilmesine yazılım sürecinde müsaade edilmemelidir (Halfond, Viegas ve Orso, 2006).

Girdilerin kodlanması:

String veri türü içerisine yerleştirilen enjeksiyon deyimleri genellikle özel karakterler barındırmaktadır. Dolayısıyla bu türden özel karakterlerin kullanımının engellenmesi SQL enjeksiyonu saldırılarını engelleyecektir. Bu durumda en iyi çözüm için string değişkenlerin içerisindeki özel karakterlerin kodlandığı ve bu karakterlerin veri tabanı tarafından normal karakterler gibi algılanmasını sağlayan fonksiyonlar kullanılmalıdır (Halfond, Viegas ve Orso, 2006).

Girdi kaynaklarının kontrolü:

Yazılımcılar SQL enjeksiyonu saldırılarından korunmak maksadıyla uygulamaya yapılan bütün girdileri kontrol etmek zorundadırlar. Kısım 3.2.4.'te açıklandığı üzere SQL enjeksiyonu saldırısı gerçekleştirmek maksadıyla birçok mekanizma kullanılabilmektedir. Dolayısıyla söz konusu kaynakların her birinin yazılım geliştiriciler tarafından kontrol

edilmesi ve bu kaynaklara yönelik tedbirler alınması önemlidir (Jemal, Omar, Habib ve Mahfoudhi, 2020).

Yukarıda bahsedilen hususlara ilave olarak;

- Veri tabanında kullanılan tablo ve sütun isimlerinin kolaylıkla tahmin edilemeyecek şekilde belirlenmesi (Demirol, Daş ve Baykara, 2013),
- Parametreli sorgu deyimlerinin kullanılması (Clarke, 2009),
- SQL deyimleri taranmak suretiyle istismara yol açabileceği değerlendirilen karakterlerin özel fonksiyonlarla zararsız karakterlere dönüştürülmesi (Demirol, Daş ve Baykara, 2013),
- Uygulama kodlarının yazılımı test eden kişiler tarafından çok dikkatli bir şekilde kontrol edilmesi (Vural ve Sağıroğlu, 2010),
- Veri tabanı yöneticisi tarafından veri tabanına çeşitli araçlarla (MS Excel, MS SQL Server Management Studio vb.) erişim imkânı bulunan kişilere yetki dağılımı yapılırken "Bilmesi Gereken" ve "En Az Yetki" prensibine uyulması (Vural ve Sağıroğlu, 2010),
- Uygulamanın bulunduğu sunucuda siber saldırılara karşı güvenlik önlemlerinin artırılması ya da güvenlik duvarı kurulması (Daş, Kara ve Gündüz, 2012) şeklindeki tedbirlerin SQL enjeksiyonu saldırılarının önlenmesinde etkili olabileceği değerlendirilmektedir. Ancak bu bölümde bahsedilen kodlama pratiklerinin bir insan ürünü olduğu ve insanın bulunduğu her yerde hata olabileceği hususu hiçbir zaman gözardı edilmemelidir.

3.3.2. Tespit ve önleme sistemleri

Araştırmacılar tarafından yukarıda bahsedilen kodlama pratiklerinin yetersiz kaldığı alanlar görülmüş ve SQL enjeksiyonu saldırılarını daha etkili araçlarla engellemek maksadıyla literatüre birçok tespit ve önleme sistemi sunulmuştur. Bu sistemlerin birçoğu makine öğrenmesi yöntemlerini barındırmaktadır. Bunlardan bazıları bu bölümde incelenecektir.

Huang ve diğerleri (2003) tarafından WAVES isimli, kara kutu tekniği ile web uygulamalarının SQL enjeksiyonu saldırılarına karşı açıklıklarını tespit eden bir sistem geliştirilmiştir (Huang, Huang, Lin ve Tsai, 2003). Bu teknikte web uygulamasının SQL enjeksiyonu saldırı maksadıyla kullanılabilecek bütün noktalarını tarayan bir çeşit arama

robotu kullanılmıştır. Daha sonra belirli bir listeye göre bu noktaları hedef alan saldırılar oluşturulmuştur. Bu esnada WAVES tarafından makine öğrenmesi teknikleri uygulanarak web uygulaması tarafından saldırılara verilen tepkiler izlenmiş ve sonuçlar elde edilmiştir. Ancak diğer kara kutu sistemlerinde olduğu gibi bu sistem de tam bir doğruluk garanti etmemektedir.

Gould ve diğerleri (2004) tarafından dinamik olarak oluşturulan SQL sorgularının tür doğruluğu bakımından kontrolünü yapmak maksadıyla JDBC-Checker statik analiz aracı geliştirilmiştir (Gould, Su ve Devanbu, 2004). JDBC-Checker vasıtasıyla dinamik olarak geliştirilen SQL sorgularındaki tür uyuşmazlıklarından faydalanarak SQL enjeksiyonu saldırıları tespit edilmektedir.

Fu ve diğerleri (2007) yaptıkları çalışma ile SAFELI adlı tespit ve önleme sistemini literatüre sunmuşlardır (Fu, Lu, Peltsverger ve Chen, 2007). SAFELI vasıtasıyla SQL enjeksiyonu saldırısına yönelik çok hassas güvenlik açıklıkları yazılımın kaynak kodu bilgisinden faydalanarak tespit edilmektedir. SAFELI, SQL enjeksiyonu saldırılarını uygulamanın derleme zamanında tespit etmeyi amaçlayan bir araçtır.

Halfond ve Orso (2005) tarafından statik analiz ile yürütme zamanını gözlemlemeye dayalı ve model tabanlı AMNESIA tespit ve önleme sistemi geliştirilmiştir (Halfond ve Orso, 2005). İlk aşamanda AMNESIA tarafından değişik türde sorgu modelleri üretmek için statik analiz kullanılır. Dinamik aşama olarak adlandırılan bir sonraki aşamada ise bütün sorgular veri tabanına erişimden önce daha önceden statik analiz ile oluşturulan modeller ile kıyaslanır. Statik modellere uyum sağlamayan sorgular SQL enjeksiyonu olarak değerlendirilir ve veri tabanına erişi engellenir.

Alattar ve Medhane (2013) tarafından SQL enjeksiyonu saldırılarını etki bir şekilde gerçek zamanlı ortamda tespit eden ve önleyen R-WASP isimli bir araç önerilmiştir (Alattar ve Medhane, 2013). Ancak bahse konu aracın kısıtlaması saklı yordamlar vasıtasıyla gerçekleştirilen SQL enjeksiyonu saldırılarının tespit edilmesinde çok fazla pratiğe ihtiyaç duyulmasıdır.

Huang ve diğerleri (2004) tarafından bilgi akış analizini kullanarak girdi doğrulamasına bağlı hataları tespit eden WebSSARI tespit ve önleme sistemi geliştirilmiştir. Bu

yaklaşımda, statik analiz hassas fonksiyonlar için ön koşulları kontrol maksadıyla kullanılmaktadır (Huang ve diğerleri, 2004). Bu analiz ile sisteme ön koşulların karşılamadığı ve bu ön koşullar karşısında otomatik olarak eklenebilecek filtreler önerilmektedir. Bu sistemde girdiler, daha önceden belirlenmiş filtrelerden geçerek sterilize edilir. Böylece sterilize edilen girdiler, web uygulaması için SQL enjeksiyonu saldırısı bakımından herhangi bir sorun teşkil etmeyecektir.

Thomas ve Williams (2007) tarafından yapılan çalışmada, SQL enjeksiyonu saldırılarını engellemeye yönelik önceden hazırlanan SQL ifadelerinin önemi vurgulanmıştır (Thomas ve Williams, 2007). Veri tabanı bağlantısının JDBC (Java Database Connectivity) vasıtasıyla yapıldığı durumlarda, önceden hazırlanan ifadelerin yer aldığı arayüz tarafından özellik arz eden karakterlerin kullanımından kaçınılmaktadır.

Manmadhan ve Thankappan (2007) tarafından yapılan çalışmada, sorgu semantiği kontrol edilmek suretiyle dinamik sorgu yapısı doğrulaması temeli üzerine dayalı bir sistem önerilmiştir (Manmadhan ve Thankappan, 2012). Bu tekniğin asıl amacı SQL enjeksiyonu saldırılarına karşı genel bir koruma sağlamanın yanında saklı yordamlarla geliştirilen SQL enjeksiyonu saldırılarına karşı tedbir geliştirilmesidir.

Bisht ve diğerleri (2010) yaptıkları çalışmada, dinamik aday değerlendirmesi metodu uygulanarak SQL enjeksiyonu saldırılarını engellemeyi sağlayan CANDID isimli bir sistem önerilmiştir (Bisht, Madhusudan ve Venkatakrishnan, 2010). Bu teknik, bilgisayar korsanları tarafından uygulamaya gönderilen klasik SQL deyiminin yapısının değiştirilmesi durumundan çıkarım yapmaya dayandırılmıştır.

Grazie (2008) tarafından yapılmış olan tez çalışmasında HTTP talep kesici içeren SQL Prevent adında bir teknik önerilmiştir (Grazie, 2008). Söz konusu teknikte ilk olarak HTTP talepleri yerel hafızaya alınır. Daha sonra SQL engelleyici tarafından uygulamaya gelen SQL sorguları, SQL enjeksiyonu saldırılarını tespit eden modüle aktarılmaktadır. Son adımda ise yerel hafızada tutulan HTTP talebi geri çağırılmakta ve SQL enjeksiyonu içeren herhangi bir sorgu bulunup bulunmadığı kontrol edilmektedir.

Ali ve diğerleri (2009) tarafından kullanıcı kimlik doğrulamasını daha da iyileştirmek için hash değeri yaklaşımını kullanan SQLIPA isimli bir teknik geliştirilmiştir (Ali, Rauf ve

Javed, 2009). Bu teknikte kullanıcı adı ve parola için birer hash değeri kullanılır. Sisteme girişte bir kullanıcı hesabı oluşturulduğunda yürütme zamanında kullanıcı adı ve parolayı temsilen bir hash değeri oluşturulur ve böylelikle SQL enjeksiyonu saldırılarına karşı koruma sağlanır. Ancak bu tekniğin sadece totoloji türünde SQL enjeksiyonu saldırılarına karşı kullanılabiliyor oluşu bir eksiklik olarak görülmektedir.

Bu bölümde bahsedilen SQL enjeksiyonu saldırılarını tespit ve önlemede kullanılan araç ve uygulamaların özellikleri ile güçlü / zayıf yönlerini gösteren çizelge aşağıda sunulmuştur (Çizelge 3.1.).

Çizelge 3.1. Saldırı tespit ve önleme sistemleri

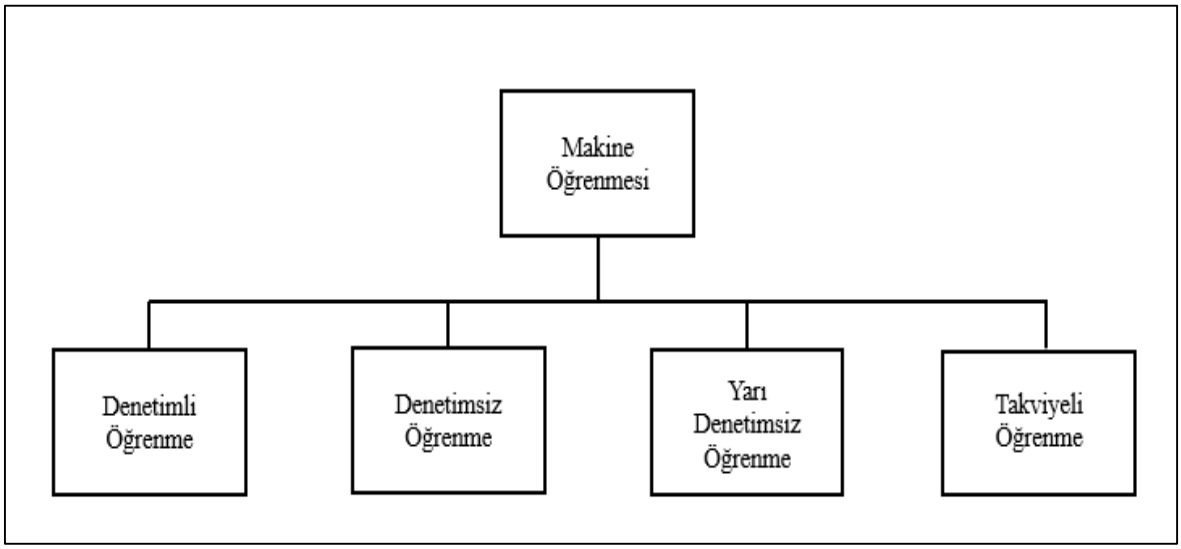
Araç / Uygulama	Özellikleri	Güçlü / Zayıf Yönleri
WAWES	Web uygulamalarının SQL enjeksiyonu saldırılarına karşı açıklıklarını tespit eden bir sistem	Tam bir doğruluk garanti etmemekte
JDBC-Checker	SQL sorgularının tür doğruluğu bakımından kontrolünü yapmak maksadıyla	Bütün saldırı türlerine karşı etkili
SAFELI	SQL enjeksiyonu saldırılarını uygulamanın derleme zamanında tespit etmeyi amaçlamakta	Totoloji türünden saldırıları tespit etmede başarısız
AMNESIA	Statik analiz ile yürütme zamanını gözlemlemeye dayalı ve model tabanlı	Başarısı statik modelleri doğruluğu ile orantılı
R-WASP	SQL enjeksiyonu saldırılarını etki bir şekilde gerçek zamanlı ortamda tespit eden ve önleyen bir araç	Saklı yordamlar ile geliştirilen saldırılara karşı hassas
WebSSARI	Bilgi akış analizini kullanarak girdi doğrulamasına bağlı hataları tespit eden bir sistem	Bütün saldırı türlerine karşı etkili
CANDID	Dinamik aday değerlendirmesi metodu kullanarak	Bütün saldırı türlerine karşı etkili
SQLIPA	Kullanıcı kimlik doğrulamasını iyileştirmek için hash değeri yaklaşımını kullanmakta	Sadece totoloji türünden saldırılara karşı etkili
SQL Prevent	HTTP talebi geri çağırılmakta ve SQL enjeksiyonu içeren herhangi bir sorgu olup olmadığı kontrol edilmekte	Bütün saldırı türlerine karşı etkili

4. YÖNTEM VE MATERYAL

Bu bölümde, SQL enjeksiyonu saldırılarının tespit edilmesi için kullanılmış olan makine öğrenmesi yöntemlerine ve makine öğrenmesi uygulamasında kullanılmış olan veri seti hakkında ayrıntılı bir açıklamaya yer verilmiştir.

4.1. Makine Öğrenmesi Türleri

Makine öğrenmesi literatürde, bir probleme ait verilerin kullanılması suretiyle problemin modellenmesi şeklinde tanımlanmıştır (Atalay ve Çelik, 2017). Söz konusu modellemeler en yüksek verimi elde etmek maksadıyla kurgulanmaktadır. Makine öğrenmesi türleri aşağıdaki şekilde sınıflandırılmaktadır (Şekil 4.1.).



Şekil 4.1. Makine Öğrenmesi Türleri (Aytekin, 2021)

4.1.1. Denetimli öğrenme

Denetimli öğrenme ifadesi ile makine öğrenmesi modelinin eğitim maksadıyla kullanılan bir veri setine dayandığı süreç belirtilmektedir (Akay, 2018). Denetimli öğrenme ile kastedilen aslında eğitim verisinin bir test verisi ile denetlenmesi olup bu denetim tahmin ve kestirmeye odaklanmaktadır. Denetimli makine öğrenmesi regresyon ve sınıflandırma şeklinde ele alınmaktadır. Nicel değişkenlerin tahmin edilmesi ve modellerinin oluşturulması maksadıyla

regresyon kullanılırken nitel kategorilerle gözlemlerin ilişkilendirilmesi için sınıflandırma kullanılmaktadır (Akay, 2018).

4.1.2. Denetimsiz öğrenme

Denetimsiz öğrenme ile kullanıcının sisteme herhangi bir müdahalenin bulunmadığı bir süreç ifade edilmektedir (Kızılkaya ve Oğuzlar, 2018). Kullanıcı tarafından yalnızca girdiler sisteme aktarılır, fakat bu işlemden sonra başka bir işlem yapılmamakta ve sistem kendiliğinden gözlemler yaparak verilerin arasındaki ilişkileri ortaya koymaya çalışmaktadır (Alpaydın, 2010).

4.1.3. Yarı denetimsiz öğrenme

Hem denetimli hem de denetimsiz öğrenmenin yetersiz kaldığı bazı durumlarda yarı denetimsiz öğrenme bir alternatif olarak değerlendirilebilir. Yarı denetimsiz öğrenme kavramı ile miktar olarak yetersiz ancak işlenmiş veriden nispeten daha fazla miktardaki işlenmemiş veriyi tahmin etme ifade edilmektedir (Kara ve Şamlı, 2021). Böylelikle sisteme deneme ve yanılma öğretilir.

4.1.4. Takviyeli öğrenme

Bu öğrenme şeklinde girdilerin çevreleriyle etkileşim kurarak geri bildirimi maksimum seviyeye çıkarması ve sistemin optimum davranışı bulması amaçlanır (Arslankaya ve Toprak, 2021). Dolayısıyla diğer öğrenme türlerinden daha farklı bir yaklaşım içermektedir.

4.2. Değerlendirmede Kullanılan Ölçüler

Doğruluk değeri bir modelin seçiminde en çok kullanılan değer olmakla birlikte tek başına kullanılması çoğu zaman yeterli değildir. Doğruluk değeri, oluşturduğumuz modelde doğru tahmin ettiğimiz verilerin toplam veri sayısına oranı ile hesaplanmaktadır. Doğruluk değerinin hesaplanmasına ilişkin formül aşağıda sunulmuştur (Şekil 4.2.).

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Şekil 4.2. Doğruluk değeri hesaplaması

Kesinlik değeri, tahmin değeri bilimsel açıdan müspet olan verilerin aslında kaç tanesinin müspet olduğunu görmemize yarayan bir orandır. Kesinlik değerinin hesaplanmasına ilişkin formül aşağıda sunulmuştur (Şekil 4.3.).

$$\frac{TP}{TP + FP}$$

Şekil 4.3. Kesinlik değeri hesaplaması

Duyarlılık değeri, bilimsel açıdan müspet olarak tahminlememiz gereken değerlerin aslında kaç tanesinin müspet tahminlendiğini belirten orandır. Duyarlılık değerinin hesaplanmasına ilişkin formül aşağıda sunulmuştur (Şekil 4.4.).

$$\frac{TP}{TP + FN}$$

Şekil 4.4. Duyarlılık değeri hesaplaması

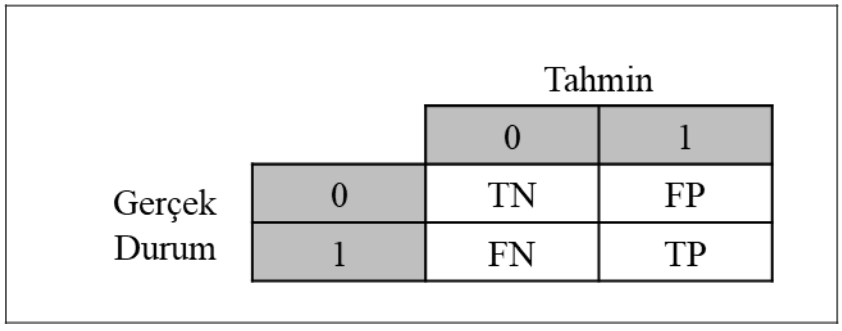
F_1 puanı ise doğruluk ve kesinlik değerlerinin ortalaması hesaplanarak elde edilen bir değerdir. F_1 puanının kullanılmasının en önemli nedeni heterojen yapıdaki verilerden kaynaklı yanlış model seçimi yapmanın önüne geçmektir. F_1 puanının hesaplanmasına ilişkin formül aşağıda sunulmuştur (Şekil 4.5.).

$$2 * \frac{Kesinlik * Duyarlılık}{Kesinlik + Duyarlılık}$$

Şekil 4.5. F_1 puanı hesaplaması

Adını aynı veri setindeki örneklerin farklı şekillerde gruplanmasından alan çapraz doğrulama metodu ile eğitim veri setinden örnek gözlem kümeleri oluşturularak her bir test kümesinin performansının ölçülmesiyle modelin genel başarısı hesaplanır (Tektaş ve Korkmaz, 2023). Örneğin alt küme sayısının 7 olarak seçildiği bir makine öğrenmesi uygulamasında bu yedi parça ayrı ayrı bir kez test kümesi olacak şekilde eğitim gerçekleştirilir ve test sonuçlarının ortalaması alınarak sistemin genel başarısı ölçülür (Bilgin, 2017). Çapraz doğrulama yöntemi, özellikle birçok sınıflandırıcı içerisinden en iyi performansı gösteren sınıflandırıcının bulunması gerektiği durumlarda kullanılan önemli bir metriktir.

Performans ölçüm yöntemlerinden biri olan karmaşıklık matrisini gösteren şekle aşağıda yer verilmiştir (Şekil 4.6.). Matris içerisinde yer alan True-Negative (TN) değeri ile doğru tahmin edilen SQL enjeksiyonu saldırısına yönelik ifade içermeyen deyimler belirtilmiştir. True-Positive (TP) değeri ile doğru tahmin edilen SQL enjeksiyonu saldırısına yönelik ifade içeren deyimler belirtilmiştir. False-Positive (FP) değeri ile gerçekte SQL enjeksiyonu saldırısına yönelik herhangi bir ifade içermeyen ancak model tarafından yanlış tahmin edilen veriler gösterilmiştir. False-Negative (FP) değeri ile de gerçekte SQL enjeksiyonu saldırısına yönelik bir ifade içeren ancak model tarafından yanlış tahmin edilen veriler gösterilmiştir. Karmaşıklık matrisi TN, FN, FP ve TP değerleri hakkında bize bilgi sunan bir matristir (Alpaydın, 2010).



Şekil 4.6. Karmaşıklık matrisi

4.3. Uygulamada Kullanılan Sınıflandırma Algoritmaları

Bu çalışma kapsamında, SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespit edilmesinde 1950'li yıllardan itibaren kullanılagelen geleneksel makine öğrenmesi metotları

ile özellikle 2000'lerden sonra gelişim gösteren topluluk öğrenme algoritmalarının bir türü olan artırma algoritmaları karşılaştırılacaktır.

Geleneksel makine öğrenmesi algoritmaları tarafından kümeleme, regresyon ve sınıflandırma görevleri yerine getirilmektedir (Oracle, 2023). Literatürde birçok sınıflandırma probleminin çözümünde bu algoritmalardan faydalanılmıştır. Geleneksel makine öğrenmesi algoritmalarının diğer algoritmalara göre avantajı ise birçok algoritmaya oranla daha hızlı sonuç vermesidir (Kapernikov, 2023). Destek Vektör Makinesi (Support Vector Machine-SVM), Karar Ağacı (Decision Tree-DT), K-En Yakın Komşu (K-Nearest Neighbour-KNN), Naive Bayes (NB), Gaussian Naive Bayes (GNB), Lojistik Regresyon (LR) ve Rastgele Orman (Random Forest-RF) gibi algoritmalar geleneksel makine öğrenmesi algoritmaları arasında gösterilmektedir (Kamath, Bukhari ve Dengel, 2018).

Topluluk (ensembl) öğrenme algoritmaları, birden fazla temel öğrenme algoritmasını bir araya getirerek daha güçlü ve daha genelleyici bir model oluşturmayı amaçlayan makine öğrenmesi algoritmalarıdır (Kokoç ve Kokoç, 2021). Artırma (boosting) algoritmaları da, topluluk öğrenme algoritmalarından biridir. Artırma metotları, zayıf modelleri daha yüksek doğruluk oranları elde etmek maksadıyla birleştirmek suretiyle uygulanan bir yöntemdir (Uyanık ve Kasapbaşı, 2021). İlk adımda oluşturulan bu zayıf model her seferinde çağrılarak yeni bir zayıf tahmin kuralı oluşturulur ve birçok turdan sonra, artırma algoritması tarafından bu zayıf kurallar bir araya getirilerek doğruluk oranı çok daha yüksek tek bir tahmin kuralı elde edilir (Schapire, 2003). Literatüre giren ilk artırma algoritması olan AdaBoost 2001 yılından itibaren makine öğrenmesi uygulamalarında kullanılmaktadır (Uyanık ve Kasapbaşı, 2021). Artırma algoritmalarının en iyi bilinen örnekleri Catboost, LightGBM, XGBoost, AdaBoost ve Gradient Boosting Tree (GBT) algoritmalarıdır.

Uygulamada kullanılan algoritmalar aşağıda sunulmuştur. Seçilen algoritmalar, metin biçimindeki verileri sınıflandırabilmesi dolayısıyla çalışma kapsamında tercih edilmiştir.

4.3.1. Geleneksel makine öğrenmesi algoritmaları

K-en yakın komşu algoritması:

Bir tür denetimli makine öğrenmesi algoritması olmakla birlikte literatürde yer alan birçok çalışmada bu algoritmadan istifade edilmiştir. Algoritmanın temelinde birbirine en çok benzeyen verilerin bulunması ve bu verilere göre bir sınıflandırılma yapılması mantığı bulunmaktadır (Manning ve Schütze, 1999). "k" değeri aranacak komşu sayısını bildirmekle beraber bu değer en az iki olmalıdır (Manning ve Schütze, 1999). Eğitim aşamasında algoritma verileri etiketleriyle birlikte öğrenir. Test aşamasında ise algoritmaya tahmin edilmesi gereken yeni bir veri noktası verilir. Bu yeni veri noktası, eğitim veri kümesindeki diğer veri noktalarıyla benzerlik ölçüsüne göre karşılaştırılır. Algoritma bu noktaya en yakın k komşuyu (yani k en yakın veri noktasını) seçer. Bu komşuların etiketleri arasında en sık görülen sınıfı, yeni veri noktasının tahmin edilen sınıfı olarak kabul eder ve algoritma tahminini gerçekleştirir (Akyel ve Seçkin, 2012). Makine öğrenmesi uygulamasında kullanılan k-en yakın komşu algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-1'de sunulmuştur.

Naive Bayes algoritması:

Naive Bayes algoritması, belirli bir veri kümesindeki değerlerin frekansının ve kombinasyonlarının sayılarak olasılık kümesinin hesaplandığı basit bir olasılık sınıflandırıcısıdır (Sarıtaş ve Yaşar, 2019). Algoritma tarafından bayes teoremini kullanılır ve sınıf değişkeninin değerini dikkate alarak tüm değişkenlerin bağımsız olduğu varsayımına dayanır, ancak bu varsayım gerçek uygulamalarda nadiren geçerlidir ve dolayısıyla naive olarak nitelendirilir (Sarıtaş ve Yaşar, 2019). Bu algoritma tarafından eğitim aşamasında veri setindeki her bir sınıfın olasılığı hesaplanır ve daha sonra test aşamasında yeni bir veri örneği geldiğinde bu verinin olasılığı hesaplanarak hangi sınıfa dahil olduğu tahmin edilir. Makine öğrenmesi uygulamasında kullanılan naive bayes algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-2'de sunulmuştur.

Karar ağacı algoritması:

Karar ağacı, bir tür denetimli makine öğrenmesi algoritmasıdır. Model, eğitim setlerini algoritmaya öğretirken tıpkı bir ağaçta olduğu gibi kökler ve yapraklar oluşturulur ve bu ağaç veri kümesindeki niteliklerle birlikte dallanır (Özbalcı, 2020). İlk olarak, veri kümesindeki öznitelikler ayrılır. Müteakiben bu ayrılmış nitelikler her bir düğümü oluşturur ve ağaç bu düğümlerden gelen cevaplarla öğrenmeye başlar (Özbalcı, 2020). Makine öğrenmesi uygulamasında kullanılan karar ağacı algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-3'te sunulmuştur.

Destek vektör makinesi algoritması:

Bu algoritma, lineer olmayan sınıfların, lineer olarak ayrıldığı bir boyut ile ilişkilendirilmek suretiyle örnekler arasındaki maksimum sınırın bulunması temeline dayanan bir algoritmadır (Demirci, 2007). Sınıflar arasında seçilecek iki çizgi birbirine paralel olmalı ve her biri bir sınıfın en az bir veri noktasına değecek şekilde çizilmelidir (Özbalcı, 2020). Bu veri noktalarına destek vektörü denilmektedir. Destek vektör makinesi algoritmaları, sınıflandırma problemleri için özellikle küçük ve orta ölçekli veri kümelerinde etkili sonuçlar verirken büyük veri kümeleriyle çalışırken hesaplama maliyetini artırabilir. Makine öğrenmesi uygulamasında kullanılan destek vektör makinesi algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-4'te sunulmuştur.

Gaussian Naive Bayes algoritması:

Naive bayes algoritması ile aynı mantık ile çalışmaktadır. Ancak Gaussian naive bayes algoritması veri özelliklerinin Gaussian prensibine, normal dağılıma, uyma olasılığını varsayarak sınıflandırmayı uygular (Lou ve diğerleri, 2014). Bu algoritma nümerik veriler ile kullanıldığında daha başarılı sonuçlar vermektedir. Makine öğrenmesi uygulamasında kullanılan Gaussian naive bayes algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-5'te sunulmuştur.

Rastgele orman algoritması:

Rastgele orman algoritmaları, birkaç karar ağacının rastgele büyütülerek birleştirilmesine dayanan bir sınıflandırma yöntemidir (Nguyen, Wang ve Nguyen, 2013). Rastgele orman algoritmalarında her bir karar ağacı, verileri sınıflandırır ve müteakiben her ağacın tahmin ettiği sınıfların oyları alınır ve en çok oy alan sınıf rastgele orman algoritmasının tahmin ettiği sınıf olarak kabul edilir (Akar ve Güngör, 2012). Makine öğrenmesi uygulamasında kullanılan rastgele orman algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-6'da sunulmuştur.

Lojistik regresyon:

Lojistik regresyon algoritması veri katsayılarını, belirli bir olasılık dağılımına (örneğin normal dağılım, binom dağılım, Poisson dağılım gibi) göre modelleyerek, bu dağılımın parametrelerini tahmin etmeye çalışan en çok olabilirlik yöntemi ile yapan bir sınıflandırma algoritmasıdır (Yavuz ve Çilengiroğlu, 2020). Katsayıların bulunası ile modelin eğitimi tamamlanmış olur ve daha sonra test aşamasında test verisinin tahmini algoritma tarafından yapılır. Makine öğrenmesi uygulamasında kullanılan lojistik regresyon algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-7'de sunulmuştur.

4.3.2. Artırma algoritmaları

AdaBoost algoritması:

AdaBoost algoritması artırma algoritmalarının ilk örneklerindendir. İlk olarak 1995 yılında kullanılmaya başlayan ve 1998 yılında gelişim gösteren AdaBoost algoritması esas olarak sınıflandırma problemlerine odaklansa bile ikili sınıf problemleri, çok sınıflı tek etiket problemleri, çoklu sınıflandırma ve çoklu etiket problemleri, tek etiket problemleri ve regresyon problemleri gibi birçok problemin çözümüne olanak sunmaktadır (Chengsheng, Huacheng ve Bing, 2017). Yukarıdaki bölümde açıklanan artırma algoritmaları mantığı üzerine kurulmuştur. AdaBoost algoritması birçok algoritmaya göre oldukça hızlı olmasının yanında basit ve kolay programlanabilirdir (Chengsheng, Huacheng ve Bing, 2017). Makine öğrenmesi uygulamasında kullanılan AdaBoost algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-8'de sunulmuştur.

CatBoost algoritması:

CatBoost algoritması, gradyan artırma (gradient boosting) tekniğini temel alan, zayıf öğrenicileri (genellikle karar ağaçları) ardışık olarak eğiten ve her ağacın bir önceki ağacın hatalarını düzeltmeye odaklandığı bir artırma algoritmasıdır (Zhou ve diğerleri, 2021). Algoritma ile bu şekilde bir araya getirilen zayıf öğreniciler, daha güçlü bir öğrenici modeli oluştururlar. 2017 yılında Yandex firması tarafından geliştirilen CatBoost algoritması, diğer artırma algoritmalarına göre daha hızlıdır ve işlenecek olan verileri otomatik olarak sayısal verilere dönüştürerek işlem yapar (Sanjeetha ve diğerleri, 2021). Makine öğrenmesi uygulamasında kullanılan CatBoost algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-9'da sunulmuştur.

XGBoost algoritması:

XGBoost algoritması, sınıflandırma ve regresyon problemlerinde kullanılan gradyan artırma tabanlı bir makine öğrenmesi algoritmasıdır (Mitchell ve Frank, 2017). Diğer artırma algoritmaları ile aynı mantıkla çalışır. XGBoost algoritmaları verileri yüksek doğruluk oranı ile sınıflandırsalar bile veri işleme hızları oldukça düşüktür (Mitchell ve Frank, 2017). Gelişmiş düzenlileştirme (regularization) teknikleri, ağaç düzenlemesi ve paralel hesaplama gibi özellikleri ile bilinen XGBoost algoritmaları 2011 yılında yapılan bir çalışma ile literatüre tanıtılmıştır (Li, Yin, Quan ve Zhang, 2019). XGBoost algoritması tahmin modellerinin yanı sıra çok büyük veri kümelerinin analiz ve sınıflandırılmasında kullanılabilir (Çoşkun ve Çetin, 2022). Makine öğrenmesi uygulamasında kullanılan XGBoost algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-10'da sunulmuştur.

LightGBM algoritması:

LightGBM algoritması verilerle ilgili tahmin problemlerinin daha etkin bir şekilde çözülebilmesini amaçlayan bir artırma algoritmasıdır. LightGBM algoritması göstergeleri seçmek ve bölmek için ön sıralama yaklaşımını kullanan gradyan artırma karar ağacı (Gradient Boosting Decision Tree-GBDT) algoritması çerçevesi kullanılarak tasarlanmıştır (Liang, Luo, Zhao ve Wu, 2020). Bu yöntem, sınıfları ayırma noktasını tam olarak belirleyebilse de, daha fazla zaman ve bellek gerektirdiğinden LightGBM algoritmasında

eğitim hızını artırmak ve bellek tüketimini azaltmak için histogram tabanlı algoritma ve maksimum derinlik sınırına sahip yaprak bazında büyüme stratejisi benimsenmiştir (Liang, Luo, Zhao ve Wu, 2020). Bellek kullanımı, işlem süresi ve aritmetik hız dikkate alındığında, LightGBM algoritması optimum çözümler sunabilir (Çoşkun ve Çetin, 2022). Makine öğrenmesi uygulamasında kullanılan LightGBM algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-11'de sunulmuştur.

Gradient tree boosting algoritması:

Gradyan artırma ağaç algoritması, bir önceki öğrenicinin hatalarını düzeltmek ve yeni öğrenicilerin daha iyi sonuçlar elde etmesi için ekleme yapmak suretiyle güçlü bir öğrenici inşa eden bir artırma yaklaşımıdır (Adler ve Painsky, 2022). Makine öğrenmesi uygulamasında kullanılan gradient boosting tree algoritmasına ait Pyhton programlama dili ile hazırlanmış olan kod parçası Ek-12'de sunulmuştur.

4.4. Veri Setinin Elde Edilmesi

SQL enjeksiyonu saldırıları üzerine yapılan çalışmalarda kullanılan veriler, genel olarak, balküpü (honeypot) veya bir web sitesi üzerindeki gerçek zamanlı web trafiğinin izlenmesi yoluyla elde edilen veriler ve bir web trafiğinin simüle edilmesi yoluyla elde edilen veriler olmak üzere iki şekilde elde edilmektedir (Ross, 2018).

Gerçek zamanlı web trafiğinin izlenmesi ile elde edilen veriler, bilgisayar korsanlarının güncel metotlarını göstermesi ve hepsinden öte gerçekçi olması bakımından değerli olmasına rağmen bu tür verilerin elde edilmesi firmaların ve organizasyonların güvenlik ve gizliliğe verdiği önem bakımından oldukça zordur. Diğer yandan, oluşturulacak sanal bir ortamda SQLMap ve benzeri uygulamalar ile başka bir sanal makine üzerinde bulunan veri tabanının zafiyetlerini istismar etmek suretiyle bir SQL enjeksiyonu saldırısı simüle edilerek elde edilen veriler veya açık kaynaktan elde edilen veri setleri literatürdeki çalışmalarda sıklıkla kullanılmaktadır.

Bu çalışmada kullanılan veri seti-1, açık kaynak web sitesinde yer alan ve gerçek web trafiğinin izlenmesi ile elde edilen bir veri setidir. Veri seti-1'in elde edildiği url adresi aşağıda sunulmuştur: *https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-*

dataset. Veri seti-2 ise açık kaynaktan elde edilen ve SQL enjeksiyonu saldırılarına yönelik ifadeler içeren deyimlerle birlikte bu çalışma kapsamında hazırlanan SQL deyimlerinin birlikte kullanılması ile elde edilen yeni bir veri setidir. SQL enjeksiyonu saldırılarına yönelik veri seti elde etmenin zor olması dolayısıyla farklı çalışmalarda da istifade edilmesi maksadıyla açık kaynak olarak sunulmuştur. Veri seti-2'de SQL enjeksiyonu saldırısına yönelik ifadeler içeren deyimler *https://github.com/payloadbox/sql-injection-payload-list* adresinden elde edilmiştir.

4.5. Veri Setlerinin Düzenlenmesi

Veri seti-1 içerisinde yaklaşık 26.700 satırlık düz metin ifadeleri, SQL deyimleri ve SQL enjeksiyonuna yönelik ifade içeren deyimler bir arada bulunmaktadır. Veri seti-2 içerisinde ise yaklaşık 1.200 satırlık SQL deyimleri ve SQL enjeksiyonuna yönelik ifade içeren deyimler yer almaktadır.

Her iki veri seti içerisinde "sentence" değişkeninin yer aldığı ilk sütunda SQL enjeksiyonu saldırısında kullanılan çeşitli semboller ve ifadeler içeren deyimlerle birlikte SQL enjeksiyonuna yönelik sembol ve ifade içermeyen temiz diye ifade edilebilecek SQL deyimleri yer almıştır. "label" değişkenin yer aldığı ikinci sütunda ise makine öğrenmesi uygulamasında sınıflandırma verisi olarak kullanılacak değişkenler yer almıştır. "label" değişkeninde yer alan veriler SQL enjeksiyonu saldırısına yönelik bir ifade, rakam ya da sembol içerip içermediğinin incelenmesi neticesinde; SQL enjeksiyonu saldırısına yönelik ifade içeren deyimler "1" ve SQL enjeksiyonu saldırısına yönelik ifade içermeyen deyimler "0" şeklinde integer değerler ile ifade edilmiştir. Çalışma kapsamında veri setlerinin bir bölümü eğitim verisi, bir kısmı test verisi olarak kullanılmıştır. Bu oran her iki veri setinde de %20 olarak uygulanmıştır.

Çalışmanın tekrar edilebilirliğin sağlanması maksadıyla veri setleri açık kaynak olarak aşağıda belirtilen adreste erişime sunulmuştur: *https://github.com/emrept?tab=repositories*. Veri seti-1'e ait bir bölüm Çizelge 4.1'de ve veri seti-2'ye ait bir bölüm Çizelge 4.2'de sunulmuştur.

Çizelge 4.1. Veri seti-1

Sentence	Label
select * from users where id = 1 or " ( ]" or 1 = 1 -- 1	1
or 1 = 1 --	1
select * from users where id = 1 or "& (" or 1 = 1 -- 1	1
select * from users where id = 1 or "%?" or 1 = 1 -- 1	1
select * from users where id = 1 or "?#" or 1 = 1 -- 1	1
or 1 = 1 or "" =	1
/**/or/**/1/**/ = /**/1	1
SELECT * FROM though WHERE explain NOT LIKE 'ought%'	0
SELECT * FROM younger WHERE could LIKE '%chart%'	0
SELECT * FROM combine WHERE happen BETWEEN '1996-07-01' AND '1996-07-31'	0
SELECT AVG (tired) FROM worker SELECT SUM (rocket)	0
c/ honduras 139 7-a	0
calle dinamarca 14	0
c/ concepcion alexandre 82	0
chorrito bajo11	0
c/ el breton187	0
cuesta sagrado corazon 155 5?a	0
calle juan sebastian elcano 157 11?b	0
sierra bermeja s/n	0
calle joan salvat papasseit s/n	0

Çizelge 4.2. Veri seti-2

Sentence	Label
UNION ALL SELECT 'INJ'||'ECT'||'XXX'23456789101112131415161718 19#	1
UNION ALL SELECT 'INJ'||'ECT'||'XXX'234567891011121314151617181920#	1
UNION ALL SELECT 'INJ'||'ECT'||'XXX'2345678910111213141516171819202 1#	1
UNION ALL SELECT 'INJ'||'ECT'||'XXX'23456789101112131415161718192021 22#	1
UNION ALL SELECT 'INJ'||'ECT'||'XXX'234567891011121314151617181920212 223#	1
UNION ALL SELECT 'INJ'||'ECT'||'XXX'2345678910111213141516171819202122 2324#	1
UNION ALL SELECT 'INJ'||'ECT'||'XXX'234567891011121314151617181920212 2232425#	1
SELECT * FROM ruler WHERE master NOT LIKE '[before]%'	0
SELECT AVG (cookies) FROM branch SELECT SUM (property)	0
SELECT AVG (act) FROM service SELECT SUM (program)	0
SELECT AVG (mud) FROM learn SELECT SUM (cannot)	0
SELECT COUNT (ourselves) FROM saw	0
SELECT COUNT (fall) FROM gift	0
SELECT AVG (noon) FROM finger	0
SELECT * FROM before WHERE monkey NOT LIKE 'screen%'	0
SELECT * FROM instead WHERE solar LIKE '%create%'	0
SELECT * FROM fix WHERE noon NOT LIKE '[cent]%'	0

5. BULGULAR VE TARTIŞMA

Bu bölümde SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespiti ile ilgili yapılan çalışamaya yer verilmiş ve müteakiben elde edilen sonuçlar literatürde yer alan diğer çalışmalarla kıyaslanmıştır.

5.1.Deneysel Çalışma

Makine öğrenmesi uygulaması Pyhton ile gerçekleştirilmiştir. Python, geniş bir kullanıcı tabanına sahip, açık kaynaklı, yüksek seviyeli bir programlama dilidir. Veri bilimi ve makine öğrenmesi alanında yaygın olarak kullanılmaktadır.

Python, veri bilimi ve makine öğrenmesi için veri işleme, model eğitimi, model değerlendirmesi gibi birçok alanda kapsamlı kütüphaneler ve araçlar sunmaktadır (Erden, 2021). Bunlardan bazıları NumPy, Pandas, Scikit-learn, TensorFlow, Keras ve PyTorch gibi yaygın olarak kullanılan kütüphanelerdir (İnik ve Ülker, 2017). Python, SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespit edilmesinde veri manipülasyonu ve temizleme işlemleri için elverişli bir uygulamadır. Örneğin Pandas gibi kütüphaneler, verileri okuma, filtreleme, dönüştürme ve birleştirme gibi işlemleri kolaylaştırır. Ayrıca Python, birçok farklı makine öğrenmesi algoritmasının uygulanması için gerekli kütüphanelere sahiptir (Erden, 2021). Yukarıda bahsedilen nedenlerden dolayı Python programlama dili bu çalışma kapsamında tercih edilmiştir.

Makine öğrenmesi uygulaması, JetBrains tarafından geliştirilen ve Python programlama dili için tasarlanmış Pycharm ortamında gerçekleştirilmiştir (Jetbrains, 2023). PyCharm, Python projelerinin verimli bir şekilde geliştirilmesini sağlamak için kullanıcı dostu bir arayüz ve birçok özellik sunmaktadır.

Makine öğrenmesi uygulaması kapsamında artırma algoritmalarından Catboost, LightGBM, XGBoost, AdaBoost ve Gradient Boosting Tree(GBT) ve geleneksel makine öğrenmesi algoritmalarından Destek Vektör Makinesi (Support Vector Machine-SVM), Karar Ağacı (Decision Tree-DT), K-En Yakın Komşu (K-Nearest Neighbour-KNN), Naive Bayes (NB), Gaussian Naive Bayes (GNB), Lojistik Regresyon (LR) ve Rastgele Orman (Random

Forest-RF) algoritmaları olmak üzere 2 farklı türden toplam 12 farklı algoritma kullanılmıştır. Her bir algoritma iki farklı veri seti ile denenmiş ve elde edilen sonuçlar doğruluk, kesinlik, duyarlılık, F_1 puanı, çapraz doğrulama değeri ve karmaşıklık matrisi değerleri üzerinden değerlendirilmiştir. Çapraz doğrulama değerleri hesaplanırken her iki veri setinde yer alan veriler 5 parçaya bölünerek basit çapraz doğrulama (cross validation) yöntemi uygulanmış ve ortalama doğruluk değerleri elde edilmiştir. Grafik gösterimlerinde veri seti-1 için mavi, veri seti-2 için turuncu renk kullanılmıştır.

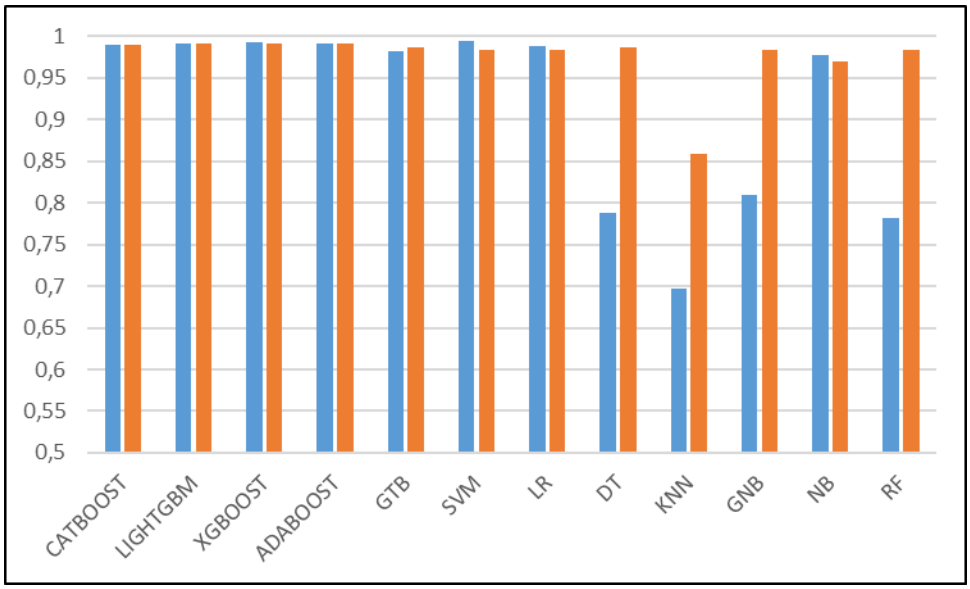
Yapılan uygulama neticesinde veri seti-1 için en yüksek doğruluk değeri 0,994 ile SVM algoritması ile elde edilmiştir. SVM algoritmasının ardından 0,993 ile XGBoost, 0,991 ile AdaBoost ve LightGBM, 0,989 ile Catboost gibi diğer artırma algoritmalarının da yüksek doğruluk değerlerine sahip olduğu görülmektedir. LR, GTB ve NB algoritmaları sırasıyla 0,988, 0,982 ve 0,977 değerleri ile bu algoritmalara en yakın değerleri elde etmişlerdir. Ancak diğer geleneksel makine öğrenmesi algoritmaları GNB, DT, RF ve KNN modellerinin doğruluk değerleri diğer algoritmaların çok daha gerisinde kalmıştır.

Veri seti-2 için algoritmalar doğruluk değerleri açısından değerlendirildiğinde; artırma algoritmaları olan LightGBM, XGBoost, Adaboost, Catboost ve GTB sınıflandırıcıları sırayla dizilmişlerdir. Geleneksel makine öğrenmesi algoritmaları, artırma algoritmalarına oranla küçük farklarla nispeten daha düşük performans sergilemişlerdir. Doğruluk değerlerine ilişkin değerler aşağıda sunulmuştur (Çizelge 5.1.).

Çizelge 5.1. Doğruluk değerleri

Sınıflandırıcı Türü	Sınıflandırıcı	Doğruluk	
		Veri Seti-1	Veri Seti-2
Artırma Algoritmaları	CATBOOST	0,989	0,989
	LIGHTGBM	0,991	0,991
	XGBOOST	0,993	0,991
	ADABOOST	0,991	0,991
	GTB	0,982	0,987
Geleneksel Algoritmalar	SVM	0,994	0,983
	LR	0,988	0,983
	DT	0,788	0,987
	KNN	0,697	0,858
	GNB	0,809	0,983
	NB	0,977	0,97
	RF	0,782	0,984

Doğruluk değerlerine ilişkin grafik aşağıda sunulmuştur (Şekil 5.1.).



Şekil 5.1. Doğruluk değerlerinin grafik ile gösterimi

Verilen grafikler bir sınıflandırma modelinin ne kadar iyi bir performans gösterdiğini belirten doğruluk değeri açısından incelendiğinde; veri seti-1 için geleneksel makine öğrenmesi algoritmalarından olan SVM ve veri seti-2 için artırma algoritmalarından olan LightGBM, XGBoost ve AdaBoost algoritmalarının SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespit edilmesi konusunda doğruluk değerleri açısından daha iyi sonuçlar verdiği söylenebilir.

Uygulama neticesinde kesinlik değerleri açısından; veri seti-1 için artırma tabanlı bir makine öğrenmesi modelleri olan Catboost ve LightGBM algoritmalarının 0,996 ile en başarılı algoritmalar olduğu söylenebilir. Bu iki algoritmanın hemen ardından bir geleneksel öğrenme modeli olan SVM algoritması yer almıştır. Diğer artırma tabanlı makine öğrenmesi modelleri olan XGBoost 0,993 ve AdaBoost 0,991 değerleri ile yüksek kesinlik değerlerine sahiptir. LR, GTB ve NB algoritmaları yüksek kesinlik değerleri ile hemen bu algoritmaların ardında yer almıştır. Geleneksel makine öğrenmesi yöntemleri olan GNB, DT, RF kesinlik değeri açısından en başarısız algoritmalar olmuştur.

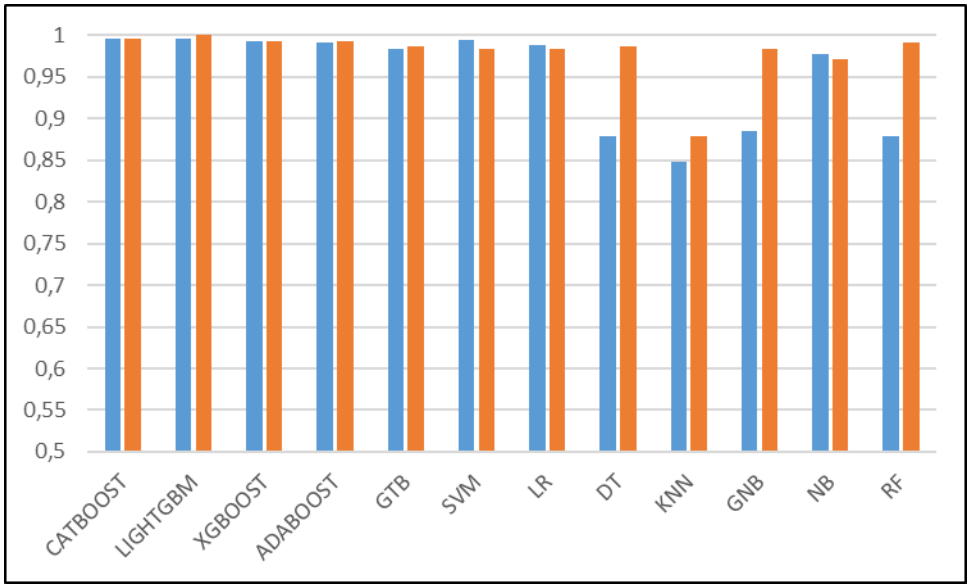
Veri seti-1'e göre daha az veri barındıran veri seti-2 ile yapılan uygulamadan elde edilen sonuçlarda artırma algoritmalarının daha iyi sonuçlar elde ettiği göze çarpmaktadır. Artırma algoritmalarından olan LightGBM 1,0 kesinlik değeri ile en yüksek değere sahipken

geleneksel makine öğrenmesi algoritmaları listenin altında yer almıştır. Algoritmaların kesinlik değerleri aşağıdaki tabloda sunulmuştur (Çizelge 5.2.).

Çizelge 5.2. Kesinlik değerleri

Sınıflandırıcı Türü	Sınıflandırıcı	Kesinlik	
		Veri Seti-1	Veri Seti-2
Artırma Algoritmaları	CATBOOST	0,996	0,996
	LIGHTGBM	0,996	1
	XGBOOST	0,993	0,992
	ADABOOST	0,991	0,992
	GTB	0,983	0,987
Geleneksel Algoritmalar	SVM	0,994	0,984
	LR	0,988	0,984
	DT	0,879	0,987
	KNN	0,848	0,879
	GNB	0,885	0,983
	NB	0,977	0,971
	RF	0,878	0,991

Kesinlik değerlerine ilişkin grafik aşağıda sunulmuştur (Şekil 5.2.).



Şekil 5.2. Kesinlik değerlerinin grafik ile gösterimi

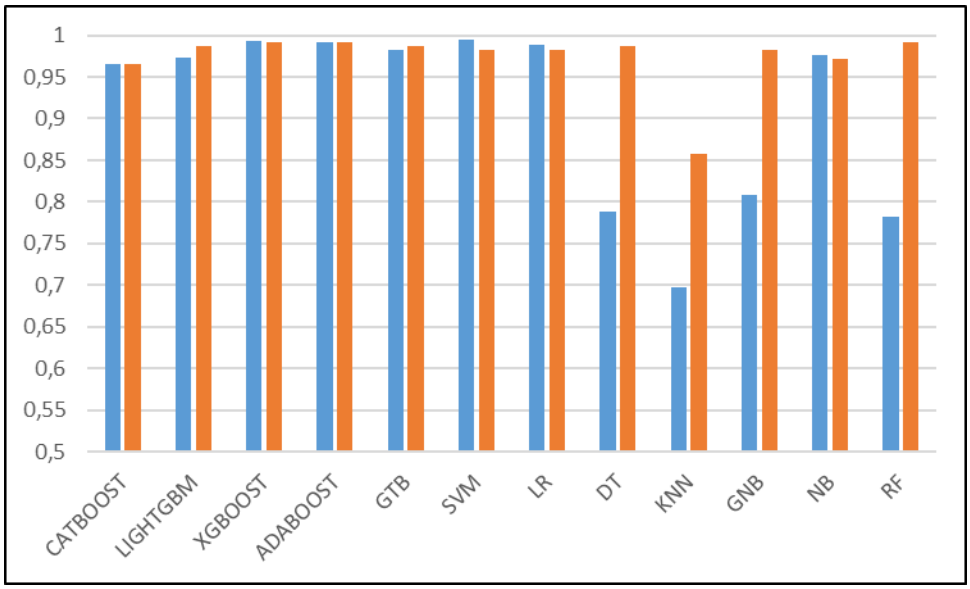
Bir sınıflandırma modelinin pozitif olarak tahmin ettiği örneklerin gerçekten pozitif olma oranını ifade eden kesinlik değerleri açısından algoritmalar değerlendirildiğinde; LightGBM algoritmasının veri seti-2 için bütün pozitif değerleri doğru tahmin ettiğini ve veri seti-2 içinde yüksek bir değere sahip olduğunu söylememiz mümkündür. Geleneksel makine öğrenmesi algoritmaları ise her iki veri seti içinde en düşük performansa sahip algoritmalar olmuştur.

Duyarlılık değeri bir sınıflandırma modelinin gerçek pozitifleri ne kadar başarılı bir şekilde tespit ettiğini belirten göstergedir. Yapılan uygulama neticesinde elde edilen duyarlılık değerleri incelendiğinde; veri seti-1 için SVM algoritmasının 0,994 ile en yüksek duyarlılık değerini elde ettiğini söylememiz mümkündür. Ancak SVM algoritmasının veri sayısı daha az olan veri seti-2 ile aynı başarıyı yakaladığını söyleyemeyiz. Her iki veri setinde de XGBoost ve AdaBoost algoritmaları duyarlılık açısından daha istikrarlı sonuçlar elde etmişlerdir. Tabloların genelinde artırma algoritmalarının duyarlılık değeri açısından daha başarılı sonuçlar elde ettiğini, geleneksel algoritmaların ise daha geride kaldığını söyleyebiliriz. Algoritmaların duyarlılık değerleri aşağıdaki tabloda sunulmuştur (Çizelge 5.3.).

Çizelge 5.3. Duyarlılık değerleri

Sınıflandırıcı Türü	Sınıflandırıcı	Duyarlılık	
		Veri Seti-1	Veri Seti-2
Artırma Algoritmaları	CATBOOST	0,966	0,966
	LIGHTGBM	0,973	0,987
	XGBOOST	0,993	0,991
	ADABOOST	0,991	0,991
	GTB	0,982	0,987
Geleneksel Algoritmalar	SVM	0,994	0,983
	LR	0,988	0,983
	DT	0,788	0,987
	KNN	0,697	0,858
	GNB	0,809	0,983
	NB	0,977	0,971
	RF	0,782	0,991

Duyarlılık değerlerine ilişkin grafik aşağıda sunulmuştur (Şekil 5.3.).



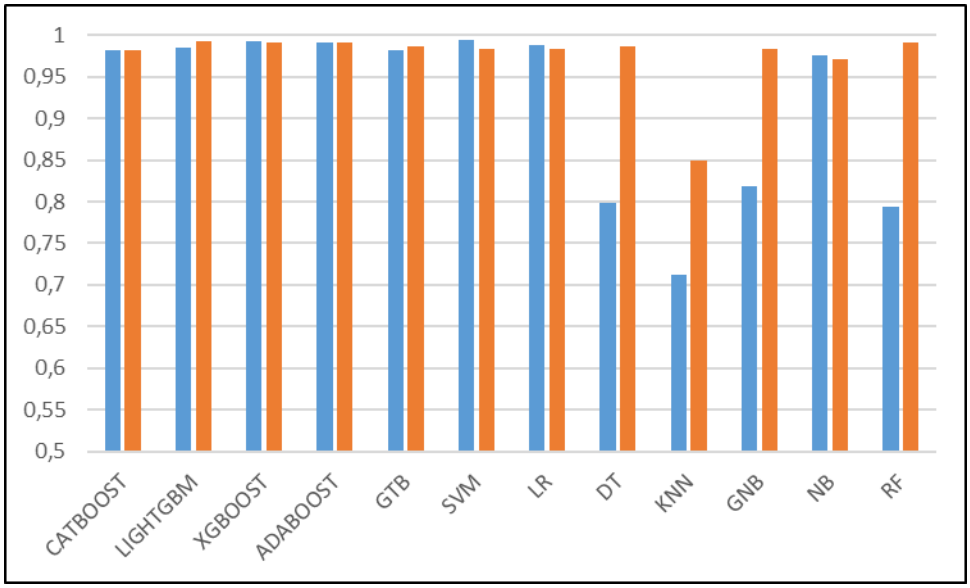
Şekil 5.3. Duyarlılık değerlerinin grafik ile gösterimi

Bir sınıflandırma modelinin hem duyarlılık hem de kesinlik performansını birleştiren bir ölçüt olmanın yanında, dengeli bir değerlendirme imkânı sağlayan ve yanlış pozitiflerle yanlış negatifler arasında bir denge kurmayı hedefleyen F_1 Puanı açısından algoritmalar değerlendirildiğinde; her iki veri setinde de XGBoost ve AdaBoost artırma algoritmalarının F_1 puanı açısından diğer algoritmalara oranla daha istikrarlı olduğunu söylememiz mümkündür. Ancak SVM algoritmasının veri seti-1 için küçük bir farkla da olsa en yüksek değeri elde ettiği göz ardı edilmemelidir. Her ne kadar veri sayının daha az olduğu veri seti-2 için geleneksel makine öğrenmesi algoritmalarının performansı F_1 puanı açısından bir iyileşme gösterse bile artırma algoritmalarına kıyasla daha düşük performans gösterdikleri söylenebilir. Veri seti-1 için DT, RF ve KNN algoritmaları 0,80 değerinin altında bir performans sergilemiş ve sırayla 0,799, 0,794 ve 0,712 F_1 puanı değerlerini elde etmişlerdir. KNN algoritması veri seti-2 ile 0,849 değeri ile diğer algoritmalara kıyasla F_1 puanı açısından oldukça düşük bir performans sergilemiştir. Algoritmaların F_1 puanı aşağıdaki tabloda sunulmuştur (Çizelge 5.4.).

Çizelge 5.4. F_1 puanı

Sınıflandırıcı Türü	Sınıflandırıcı	F_1 Puanı	
		Veri Seti-1	Veri Seti-2
Artırma Algoritmaları	CATBOOST	0,981	0,981
	LIGHTGBM	0,985	0,993
	XGBOOST	0,993	0,991
	ADABOOST	0,991	0,991
	GTB	0,982	0,987
Geleneksel Algoritmalar	SVM	0,994	0,983
	LR	0,988	0,983
	DT	0,799	0,987
	KNN	0,712	0,849
	GNB	0,819	0,983
	NB	0,976	0,971
	RF	0,794	0,991

F_1 Puanına ilişkin grafik aşağıda sunulmuştur (Şekil 5.4.).



Şekil 5.4. F_1 puanının grafik ile gösterimi

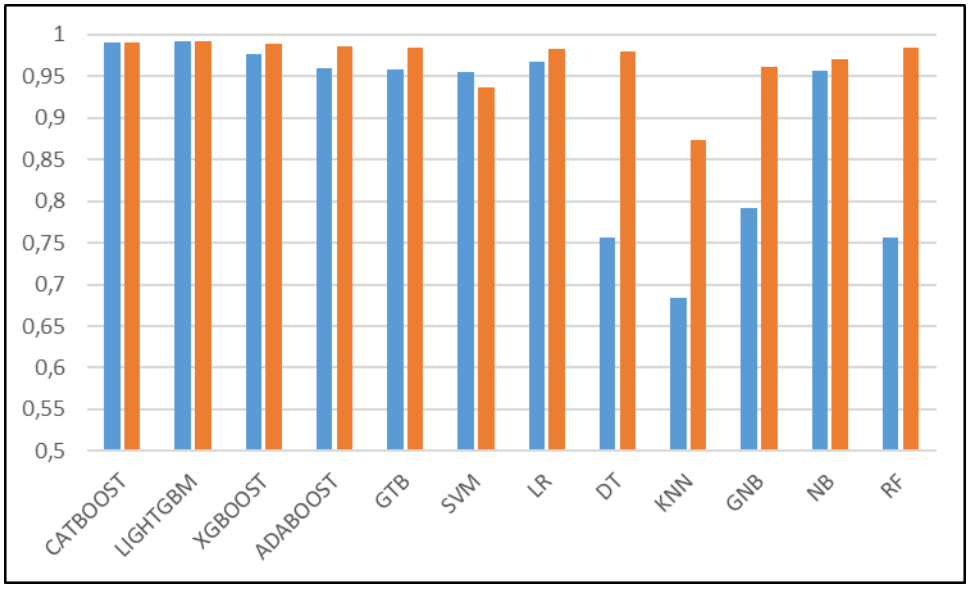
Çapraz doğrulama değerleri açısından algoritmalar değerlendirildiğinde; her iki veri seti için LightGBM 0,992 değeri ile en başarılı algoritma olarak öne çıkmıştır. Müteakiben diğer artırma algoritmaları olan CatBoost ve XGBoost algoritmaları göze çarpmaktadır. Geleneksel öğrenme algoritmalarından olan LR ise veri seti-1 için 0,968 değeri artırma algoritmalarına yakın değer elde etmiştir. Ancak veri seti-2 için bütün artırma

algoritmalarının gerisinde kalmıştır. Her iki veri seti için değerlere göz atıldığında geleneksel makine öğrenmesi algoritmalarının geride yer aldığı görülmektedir. Algoritmaların çapraz doğrulama ile elde edilen ortalama doğruluk değerleri aşağıdaki tabloda sunulmuştur (Çizelge 5.5.).

Çizelge 5.5. Çapraz doğrulama değerleri

Sınıflandırıcı Türü	Sınıflandırıcı	Çapraz Doğrulama (Ortalama Doğruluk)	
		Veri Seti-1	Veri Seti-2
Artırma Algoritmaları	CATBOOST	0,99	0,99
	LIGHTGBM	0,992	0,992
	XGBOOST	0,976	0,989
	ADABOOST	0,959	0,986
	GTB	0,958	0,984
Geleneksel Algoritmalar	SVM	0,955	0,936
	LR	0,968	0,983
	DT	0,756	0,979
	KNN	0,684	0,873
	GNB	0,791	0,961
	NB	0,957	0,97
	RF	0,756	0,984

Çapraz doğrulama değerlerine ilişkin grafik aşağıda sunulmuştur (Şekil 5.5.).



Şekil 5.5. Çapraz doğrulama değerlerinin grafik ile gösterimi

Karmaşıklık matrisine ilişkin değerler incelendiğinde; SQL enjeksiyonuna yönelik ifade içeren deyimleri, TN değerlerini, en fazla sayıda tespit eden algoritmalar XGBoost ve SVM algoritmaları olmuştur. Hem veri seti-1 hem de veri seti-2 için incelendiğinde ise SQL enjeksiyonuna yönelik ifade içeren deyimleri tespit etmede artırma algoritmalarının daha iyi sonuçlar verdiğin söylemek mümkündür. Ancak artırma algoritmaları SQL enjeksiyonuna yönelik ifade içermeyen temiz deyimleri, TP değerlerini, tespit etme konusunda geleneksel algoritmaların gerisinde kalmıştır. Geleneksel makine öğrenmesi algoritmaları, SVM, NB ve LR algoritmaları haricinde, SQL enjeksiyonuna yönelik ifade içeren çok sayıda deyimi temiz olarak değerlendirmiş ve FP olarak sınıflandırmıştır. Diğer yandan artırma algoritmaları ise temiz olan ifadelerin bazılarını SQL enjeksiyonuna yönelik ifade içeriyor şeklinde yani FN olarak tahminlenmiştir. Ancak bu sayı önceki durumda yer alan geleneksel makine öğrenmesi algoritmalarına kıyasla önemsenmeyecek bir sayı olarak değerlendirilebilir. Algoritmalara ait karmaşıklık matrisi ile elde edilen değerler aşağıdaki tabloda sunulmuştur (Çizelge 5.6.).

Çizelge 5.6. Karmaşıklık matrisi

Sınıflandırıcı Türü	Sınıflandırıcı	Veri Seti-1				Veri Seti-2			
		TN	TP	FN	FP	TN	TP	FN	FP
Artırma Algoritmaları	CATBOOST	3432	1267	44	5	89	155	2	1
	LIGHTGBM	3435	1285	26	2	90	155	2	0
	XGBOOST	3436	1282	29	1	90	155	2	0
	ADABOOST	3425	1266	45	2	90	155	2	0
	GTB	3432	1235	76	5	89	155	2	1
Geleneksel Algoritmalar	SVM	3436	1285	26	1	89	154	1	3
	LR	3413	1281	30	24	89	154	3	1
	DT	2436	1309	2	1001	89	155	2	1
	KNN	2024	1290	21	1413	56	156	1	34
	NB	3420	1219	92	17	87	153	4	3
	GNB	2588	1308	7	909	87	156	1	3
	RF	2407	1310	1	1030	90	155	2	0

5.2.Deney Sonuçlarının Literatür ile Karşılaştırılması

Yapılmış olan makine öğrenmesi uygulamasının literatürde yer alan benzer çalışmalarla kıyaslanması maksadıyla aşağıdaki çizelge oluşturulmuştur (Çizelge 5.7.).

Çizelge 5.7. Literatürde yer alan benzer çalışmalar

Çalışma	Kullanılan Veri Seti	Kullanılan Algoritmalar ve Doğruluk Değerleri	Doğruluk Değerine Göre Başarı Durumu
Farooq (2021)	SQL deyimlerinin birleştirilmesi ile elde edilen veri seti	GBM (0,991), AdaBoost (0,991), XGBoost (0,992), LightGBM (0,993).	LightGBM (0,993)
Muslihi ve Alghazzawi (2020)	616 satırlık açık kaynaktan elde edilen veri seti	GBT (0,938), SVM (0,935).	GBT (0,938)
Uwagbole ve diğerleri (2017)	Açık kaynaktan elde edilen veri seti	RF (0,998), GBT (0,996), AdaBoost (0,995), DT (0,995).	RF (0,998)
Hasan, ve diğerleri (2019)	Açık kaynaktan elde edilen veri seti	KNN, LR, SVM, GBT.	GBT (0,938)
Mishra (2019)	Açık kaynaktan elde edilen veri seti	NB (0,928), GBT (0,974).	GBT (0,974)
Krishnan, ve diğerleri (2021)	Yapılan çalışma ile aynı kaynaktan elde edilen veri seti	NB (0,950), LR (0,920), SVM (0,790), CNN (0,970).	CNN (0,970)

Farooq (2021) tarafından yapılan çalışma farklı kaynaklardan SQL deyimlerinin birleştirilmesi ile elde edilen veri seti ile gerçekleştirilmiş ve yapılmış olan çalışmanın artırma algoritmaları ile yapılan uygulamayla benzer doğruluk oranlarını yakaladığı görülmüştür (Farooq, 2021).

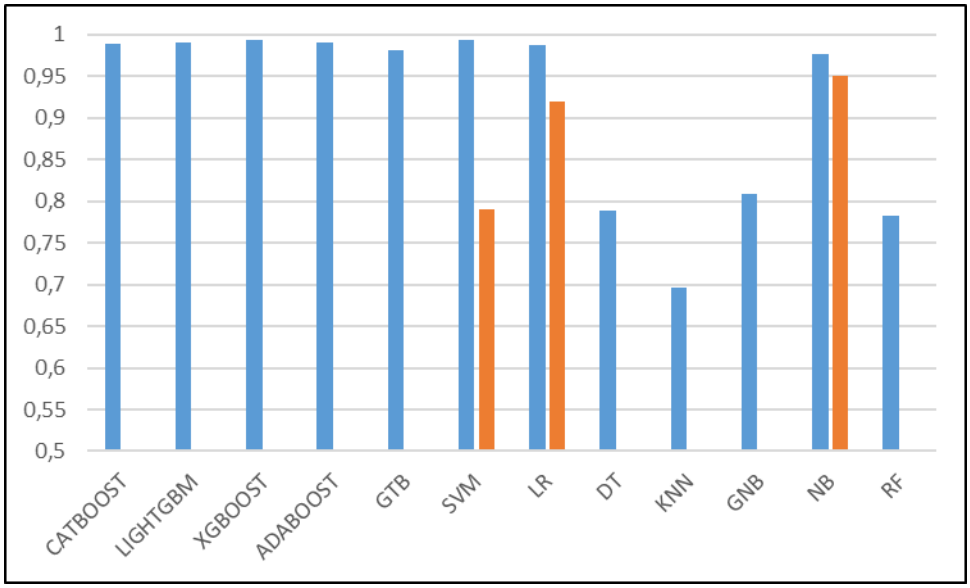
Muslihi ve Alghazzawi (2020) tarafından yapılan çalışma ile benzer şekilde yapılan bir diğer çalışmada (Mishra, 2019) bir geleneksel makine öğrenmesi algoritması ve bir artırma algoritması kullanılmıştır. Her iki çalışmada da, bu çalışmada olduğu gibi, bir artırma algoritması olan GBT algoritmasının daha iyi performans sergilediği görülmüştür.

Uwagbole ve diğerleri (2017) tarafından yapılan geleneksel makine öğrenmesi algoritmasından RF ile DT ve artırma algoritmalarından AdaBoost ve GBT kullanılmıştır çalışmada (Uwagbole ve William J. Buchanan, 2017). Çalışmada açık kaynaklardan elde edilen veri seti kullanılmıştır. Aralarında küçük farklar olsa da geleneksel makine öğrenmesi

yöntemlerinden olan RF algoritması, SQL enjeksiyonu saldırılarını tespit etmede başarılı bir performans sergilemiştir.

Hasan ve diğerleri (2019) tarafından yapılan çalışma açık kaynaktan elde edilen veri seti ile gerçekleştirilmiş ve artırma algoritması olan GBT 0,938 doğruluk değeri ile SQL enjeksiyonu saldırılarını tespit etmede başarılı olmuştur (Hasan, Balbahaith ve Tarique, 2019).

Krishnan ve diğerleri (2021) tarafından yapılan çalışmada (Krishnan, Sabu, Sajan ve Sreedeep, 2021) bu çalışmada veri seti-1 olarak belirtilen veri seti kullanılmış ve NB, LR, SVM, CNN algoritmaları ile denenmiştir. Krishnan ve arkadaşları tarafından NB, LR, SVM ve CNN algoritmaları için sırasıyla 0,950, 0,920, 0,790 ve 0,970 değerleri elde edilmiştir. Yapılan çalışma kapsamında ise NB, LR ve SVM algoritmaları için sırasıyla 0,970, 0,980 ve 0,994 değerleri elde edilmiştir. Her iki çalışmada elde edilen değerler arasındaki farklılıkların veri setinin düzenlenmesi neticesinde oluşan farklılıklardan kaynaklandığı değerlendirilmektedir. Ayrıca bu çalışma kapsamında Krishnan vd. tarafından yapılan çalışmadan farklı olarak DT, KNN, GNB, RF geleneksel makine öğrenmesi algoritmaları ve CatBoost, XGBoost, LightGBM, AdaBoost ve GBT algoritmaları kullanılmış ve daha yüksek doğruluk değerlerinin elde edildiği görülmüştür. Krishnan ve arkadaşları tarafından yapılan çalışma ile bu çalışma arasındaki algoritmaların doğruluk değerlerinin kıyaslandığı grafik aşağıda sunulmuştur (Şekil 5.6.).



Şekil 5.6. Karşılaştırma grafiği

6. SONUÇ VE ÖNERİLER

SQL enjeksiyonu saldırılarının günümüzde bilgisayar korsanlarınca en çok başvurulan saldırı türlerinden birisi olması, bu saldırılardan korunma yollarının aranmasına sebep olmuştur. Bu yollardan birisi de makine öğrenmesi alt yapısına dayanan sistem ve araçların kullanılmasıdır.

2000'li yıllardan sonra gelişim gösteren artırma algoritmaları ile yapılan bu çalışma, literatürde aynı veri seti kullanılarak NB, LR, SVM gibi geleneksel makine öğrenmesi algoritmaları ile gerçekleştirilen başka bir çalışma ile kıyaslandığında; artırma algoritmalarından XGBoost algoritmasının 0,993 doğruluk değeri elde ettiği ve literatürdeki söz konusu çalışmaya oranla XGBoost algoritması kullanılarak 0,043 değerinde bir iyileşme sağlandığı görülmektedir. Yine yapılan makine öğrenmesi uygulaması ile her iki veri seti için LightGBM, XGBoost ve AdaBoost gibi artırma algoritmalarının 0,980 doğruluk değerinin üzerinde oranlarla geleneksel algoritmalara göre daha başarılı sonuçlar elde ettiği göze çarpmaktadır. Her ne kadar geleneksel algoritmalardan SVM ve LR algoritmalarının SQL enjeksiyonu saldırılarının tespitinde başarılı olduğu göze çarpsa bile NB, GNB, DT ve KNN gibi geleneksel makine öğrenmesi algoritmalarının nispeten daha düşük sonuçlar yakaladığı görülmektedir. Dolayısıyla yapılan çalışma ile SQL enjeksiyonu saldırılarının makine öğrenmesiyle tespit edilmesinde, geleneksel makine öğrenmesi algoritmalarına göre daha yeni ve güncel metotları barındıran artırma algoritmalarının kullanılması önerilmektedir.

Literatürde SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespit edilmesi konusunda yapılan çalışmalar incelendiğinde; çalışmaların birçoğunda tekrar edilebilirliğin sağlanması adına makine öğrenmesi algoritmalarına ait kodların ve çalışmalarda kullanılan veri setlerinin paylaşılmadığı görülmektedir. Yapılan çalışmada ise hem kullanılan veri setleri açık kaynak olarak paylaşılmış hem de uygulamada kullanılan kodlar Ek bölümünde sunulmuştur. Böylelikle diğer birçok çalışmadan farklı olarak çalışmanın tekrar edilebilirliği sağlanmıştır.

SQL enjeksiyonu saldırılarının makine öğrenmesi ile tespit edilmesi konusunda yapılan çalışmalarda en çok karşılaşılan olumsuzluklardan birisi çalışmalarda kullanılmak üzere açık

kaynak olarak sunulan veri seti sayısının az olmasıdır. Bu çalışmada kullanılan veri seti-2, açık kaynaktan elde edilen ve SQL enjeksiyonu saldırılarına yönelik ifadeler içeren deyimlerle birlikte temiz içerikli SQL deyimlerinin birlikte kullanılması ile elde edilen yeni bir veri setidir. Hazırlanan veri seti farklı çalışmalarda istifade edilmesi maksadıyla açık kaynak olarak sunulmuştur. Ayrıca açık kaynak olarak sunulması itibariyle yeni SQL deyimleri eklenerek geliştirilmeye müsaittir.

SQL enjeksiyonu saldırılarının tespit edilmesinde artırma algoritmalarının derin öğrenme algoritmaları ya da farklı türden algoritmalarla kıyaslanacağı çalışmalar gelecek çalışmalar kapsamında değerlendirilebilir.

KAYNAKLAR

Adler, A. I. ve Painsky, A. (2022). Feature importance in gradient boosting trees with cross-validation feature selection. *Entropy* 24(5), 687.

Akar, Ö. ve Güngör, O. (2012). Rastgele orman algoritması kullanılarak çok bantlı görüntülerin sınıflandırılması. *Jeodezi ve Jeoinformasyon Dergisi,* 1(2), 139-146.

Akyel, N. ve Seçkin, K. (2012). K-en yakın komşuluk algoritmasının hile denetiminde kullanımı. *Muhasebe ve Vergi Uygulamaları Dergisi,* 5(1), 21-40.

Alakoç Burma, Z. (2009). *Veri tabanı yönetim sistemleri.* Ankara: Seçkin Yayıncılık, 5-11.

Alattar, M. ve Medhane, S. P. (2013). R-wasp: real time-web application sql injection detector and preventer. *International Journal of Innovative Technology and Exploring Engineering,* 2(5), 327-330.

Ali, S., Rauf, A. ve Javed, H. (2009). Sqlipa: an authentication mechanism against sql injection. *European Journal of Scientific Research,* 38(4), 604-611.

Al-Maliki, M. H. ve Jasim, M. N. (2022). Review of sql injection attacks: detection, to enhance the security of the web site from client side attacks. *International Journal Nonlinear Analysis And Applications,* 13(1), 3773-3782.

Alpaydın, E. (2010). *Introduction to machine learning,* (2. Baskı) Cambridge: MIT Press, 489-490.

Alwan, Z. ve Younis, M. (2017). Detection and prevention of sql injection attack:a survey. *International Journal of Computer Science and Mobile Computing,* 6(8), 5-17.

Atalay, M. ve Çelik, E. (2017). Büyük veri analizinde yapay zekâ ve makine öğrenmesi Uygulamaları. *Mehmet Akif Ersoy Üniversitesi Sosyal Bilimler Enstitüsü Dergisi,* 9(22), 155-172.

Atasever, S., Özçelik, İ.ve Sağıroğlu, Ş. (2019). Siber terör ve ddoS. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi,* 23(1), 238-244.

Avcı, İ., Koca, M.ve Atasoy, M. (2021). Windows tabanlı uygulamalarda sql enjeksiyon siber saldırı senaryosu ve güvenlik önlemleri. *Avrupa Bilim ve Teknoloji Dergisi*, 28, 213-219.

Aytekin, H. T. (2021). *Makine öğreniminin araştırmacıların veri analizi bağlamında potansiyel önemi.* Ufuk Üniversitesi, 1. Uluslararası Sosyal Bilimler Kongresi, 85-106.

Azman, M. A., Marhusin, M. F.ve Sulaiman, R. (2021). Machine learning-based technique to detect sql ınjection attack. *Journal of Computer Science,* 17(3), 296-303.

Bilgin, M. (2017). Gerçek veri setlerinde klasik makine öğrenmesi yöntemlerinin performans analizi. *Breast,* 2(9), 683-688.

Bisht, P., Madhusudan, P.ve Venkatakrishnan, V. (2010). CANDID: dynamic candidate evaluations for automatic prevention of sql ınjection attacks. *ACM Transaction on Information System Security,* 13(2), 1-39.

Canbek, G. (2005). *Klavye dinleme ve önleme sistemleri analiz, tasarım ve geliştirme*. Yüksek Lisans Tezi Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 32-41

Chengsheng, T., Huacheng, L.ve Bing, X. (2017). A*daboost typical algorithm and its application research.* MATEC Web of Conferences, ICTIMITE, 139.

Chica-Manjarrez, S., Marin-Lopez, A., Diaz-Sanchez, D. ve Almenares-Mendoza, F. (2020). On the automation of auditing in power grid companies. *Intelligent Environments,* 28, 331-340.

Clarke, J. (2009). *Sql injection attacks and defence*. (1. Baskı) Waltham, Syngress, 409-441.

Çelik, Ş. (2013). Stuxnet saldırısı ve ABD'nin siber savaş stratejisi: uluslararası hukukta kuvvet kullanmaktan kaçınma ilkesi çerçevesinde bir değerlendirme. *Dokuz Eylül Üniversitesi Hukuk Fakültesi Dergisi,*15(1), 137-175.

Çoşkun, K.ve Çetin, G. (2022). A comparative evaluation of the boosting algorithms for network attack classification. *International Journal of 3D Printing Technologies and Digital Industry,* 6(1), 102-112.

Daş, R., Kara, Ş.ve Gündüz, M. Z. (2012). *Casus yazılımların bilgisayar sistemlerine bulaşma belirtileri ve çözüm önerileri*. 5. Uluslararası Bilgi Güvenliği ve Kriptoloji Konferansı.Orta Doğu Teknik Üniversitesi, ANKARA.

Demirol, D., Daş, R.ve Baykara, M. (2013). *Sql enjeksiyon saldırılarına karşı güvenlik önlemleri.* 1st International Symposium on Dijital Forensics and Security (ISDFS'13). Elazığ.

Elmasri, R.ve Navathe, S. B. (2010). *Fundamentals of database systems* (6. Baskı) Georgia, Pearson, 3-28.

Erden, C. (2021). *Pyhton ile veri madenciliği.* (3. Baskı) İstanbul: Kodlab, 105-185.

Farooq, U. (2021). Ensemble machine learning approaches for detection of sql injection attack. *Technical Journal,* 15(1), 112-120.

Forristal, J. (1998). NT web teknolojisi güvenlik açıkları. *Phrack,* 54.

Fu, X., Lu, X., Peltsverger, B.ve Chen, S. (2007). *A static analysis framework for detecting sql ınjection vulnerabilities.* 1st Annual International Computer Software and Applications Conference, 1-8.

Gould, C., Su, Z.ve Devanbu, P. T. (2004). *JDBC checker: a static analysis tool for sql/jdbc applications.* 26th International Conference on Software Engineering, 697-698.

Grazie, P. (2008). *Sql prevent thesis*. Yüksek Lisans Tezi, University of British Columbia, Vancouver, Canada.

Grimes, R. A. (2001). *Malicious mobile code.* (1. Baskı), Sebastopol, O'Reilly Media, 1-21.

Güner, M. A. (2021). Yaygın ağ saldırıları ve genel hatlarıyla ağ savunması.

Halfond, W. G.ve Orso, A. (2005). *Amnesia: analysis and monitoring for neutralizing sql-ınjection attacks*. IEEE and ACM International Conference on Automated Software Engineering, 174-183.

Halfond, W. G., Viegas, J.ve Orso, A. (2006). *A classification of sql ınjection attacks and countermeasures.* In Proceedings of the IEEE International Symposium on Secure Software Engineering, IEEE, 13-15

Hasan, M., Balbahaith, Z.ve Tarique, M. (2019). *Detection of sql ınjection attacks: a machine learning approach.* International Conference on Electrical and Computing Technologies and Applications (ICECTA), IEEE , 1-6.

Holik, F.ve Neradova, S. (2017). *Vulnerabilities of modern web applications.* 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, 1256-1261.

Howard, M.ve David, L. (2003). *Writing secure code.* (2. Baskı), Washington: Microsoft Press, 455-535.

Huang, Y.-W., Huang, S.-K., Lin, T.-P.ve Tsai, C.-H. (2003). *Web application security assessment by fault injection and behavior monitoring.* Conference: Proceedings of the 12th international conference on World Wide Web, 148-159.

Huang, Y.-W., Yu, F., Hang, C., Tsai, C.-H., Lee, D.-T.ve Kuo, S.-Y. (2004). *Securing web application code by static analysis and runtime protection.* WWW '04: Proceedings of the 13th International Conference on World Wide Web, 40-52.

İnik, Ö.ve Ülker, E. (2017). *Derin öğrenmede kullanılan veri setleri ve yazılım kütüphaneleri.* 1st International Symposium on Multidisciplinary Studies and Innovative Technologies Proceedings Book, Tokat, 72-77.

İnternet: Jetbrains. (2023). URL: https://www.jetbrains.com/, Son Erişim Tarihi:23.02.2023

İnternet: Kali Linux Tutorials. (2023). URL:https://kalilinuxtutorials.com/leviathan/, Son Erişim Tarihi: 03.04.2023

İnternet: Kapernikov. (2023). URL: https://kapernikov.com/traditional-machine-learning-algorithms-for-machine-vision/, Son Erişim Tarihi: 28.02.2023

İnternet: Crowdstrike. (2022). URL: https://www.crowdstrike.com/cybersecurity-101/sql-injection/, Son Erişim Tarihi: 22.11.2022

İnternet: KAREL Elektronik. (2023). URL:https://www.karel.com.tr/blog/siber-saldiri-nedir-turleri-nelerdir, Son Erişim Tarihi: 26.04.2023

İnternet: KitPloit. (2023). URL: https://www.kitploit.com/2016/10/bbqsql-blind-sql-injection-exploitation.html, Son Erişim Tarihi: 05.05.2023

İnternet: Oracle. (2023). URL: https://blogs.oracle.com/ai-and-datascience/post/overview-of-traditional-machine-learning-techniques, Son Erişim Tarihi: 23.07.2023

İnternet: OWASP. (2022). URL: https://owasp.org/www-project-top-ten/, Son Erişim Tarihi: 15.08.2022

İnternet: Venturebeat. (2022). URL: https://venturebeat.com/security/report-35-of-educational-institutions-have-a-sqli-vulnerability/, Son Erişim Tarihi: 07.11.2022

İnternet: Serverwatch. (2022). URL: https://www.serverwatch.com/reviews/sql-injection-detection-tools/, Son Erişim Tarihi: 07.12.2022

Jemal, I., Omar, C., Habib, H.ve Mahfoudhi, A. (2020). SQL injection attack detection and prevention techniques using machine learning. *International Journal of Applied Engineering Research,* 15(6), 569-580.

Kamath, C. N., Bukhari, S. S.ve Dengel, A. (2018). *Comparative study between traditional machine learning and deep learning approaches for text classification.* Proceedings of the ACM Symposium on Document Engineering 2018, 1-11.

Kara, İ. (2019). Detection, technical analysis of brute force attack. *Sakarya University Journal of Computer and Information Sciences*, 2(2), 61-69.

Kara, M. (2013). *Siber saldırılar - siber savaşlar ve etkileri.* Yüksek Lisans Tezi, İstanbul Bilgi Üniversitesi Sosyal Bilimler Enstitüsü, İstanbul, 43-45.

Kokoç, M.ve Kokoç, F. (2021). Tahmin performansını arttırmak için kolektif öğrenme yaklaşımı: engelli bireylere yönelik deneysel çalışma. *Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi,* 10(1), 100-114.

Krishnan, A., Sabu, A., Sajan, P.ve Sreedeep, A. (2021). Sql injection detection using machine learning. *Gestao Inovaçao e Technologias,*11(3), 300-310.

Laval, M., Sultan, A. B.ve Shakiru, A. O. (2016). Systematic literature review on sql injection attack. *International Journal of Soft Computing,* 11(1), 26-35.

Li, W., Yin, Y., Quan, X.ve Zhang, H. (2019). Gene expression value prediction based on xgboost algorithm. *Frontiers in Genetics,* 10, 1077.

Liang, W., Luo, S., Zhao, G.ve Wu, H. (2020). Predicting hard rock pillar stability using gbdt, xgboost, and lightgbm algorithms. *Mathematics,* 8(5), 765.

Lou, W., Wang, X., Chen, F., Chen, Y., Jiang, B.ve Zhang, H. (2014). Sequence based prediction of dna-binding proteins based on hybrid feature selection using random forest and gaussian naive bayes. *PLoS ONE*, 9(1), e86703.

Manmadhan, S.ve Thankappan, M. (2012). A method of detecting sql ınjection attack to secure web applications. *International Journal of Distributed and Parallel Systems* 3(6), 1-8.

Mishra, S. (2019). *Sql ınjection detection using machine learning*. Yüksek Lisans Projesi, San Jose State University The Faculty of the Department of Computer Science, San Jose State, 14-41.

Mitchell, R.ve Frank, E. (2017). Accelerating the xgboost algorithm using gpu computing. *PeerJ Compututer Sciences*, 3(127).

Mozumder, D. P., Mahi, M. J.ve Whaiduzzaman, M. (2017). Cloud computing security breaches and threats analysis. *International Journal of Scientific and Engineering Research*, 8(1), 1287-1297.

Muslihi, M. T.ve Alghazzawi, D. (2020). *Detecting sql ınjection on web application using deep learning techniques: a systematic literature review.* In Proceedings of the 2020 Third International Conference on Vocational Education and Electrical Engineering, IEEE, 1-6.

Nathan, A., Kumar, V.ve Vaidyanathan, G. (2012). Preventing sql ınjection attacks. *International Journal of Computer Applications*, 52(13), 28-32.

Nguyen, C., Wang, Y.ve Nguyen, H. N. (2013). Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic. *Journal of Biomedical Science and Engineering*, 6(5), 551-560.

Ross, K. (2018). *Sql ınjection detection using machine learning techniques and multiple data sources*. Yüksek Lisans Tezi, San Jose State University Department of Computer Sciences, San Jose State, 2-25 .

Sanjeetha, R., Raj, A., Saivenu, K., Ahmed, M. I., Sathvik, B.ve Kanavall, A. (2021). Detection and mitigation of botnet based ddos attacks using catboost machine learning algorithm in sdn environment. *International Journal of Advanced Technology and Engineering Exploration*, 8(76), 445-461.

Schapire, R. E. (2003). The boosting approach to machine learning: an overview. *MSRI Workshop on Nonlinear Estimation and Classification*, 149-171.

Singh, J. P. (2016). Analysis of sql ınjection detection techniques. *Theorotical and Applied Informatics*, 28(1-2), 37-55.

Tektaş, N.ve Korkmaz, E. (2023). *Makine öğrenmesi algoritması ile çanakkale ilinin güneş ışınım şiddetinin analizi.* 1st International Conference on Contemporary Academic Research, 48-52.

Thomas, S.ve Williams, L. (2007). *Using automated fix generation to secure sql statements.* Proceedings of the Third International Workshop on Software Engineering for Secure Systems, 9-9.

Uwagbole, S. O.ve William J. Buchanan, L. F. (2017). *An applied pattern-driven corpus to predictive analytics in mitigating sql ınjection attack.* 2017 Seventh International Conference on Emerging Security Technologies (EST), Canterbury, UK, 12-27.

Uyanık, F.ve Kasapbaşı, M. C. (2021). Telekomünikasyon sektörü için veri madenciliği ve makine öğrenmesi teknikleri ile ayrılan müşteri analizi. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, 9, 172-191.

Ünver, M.ve Mirzaoğlu, A. G. (2011). *Yemleme ("phishing").* Bilgi Teknolojileri ve Koordinasyon Dairesi Başkanlığı, Ankara, 2-18.

Vural, Y.ve Sağıroğlu, Ş. (2010). Veritabanı Yönetim sistemleri güvenliği: tehditler ve korunma yöntemleri. *Politeknik Dergisi,* 13(2), 71-81.

Yavuz, A.ve Çilengiroğlu, Ö. V. (2020). Lojistik regresyon ve cart yöntemlerinin tahmin edici performanslarının yaşam memnuniyeti verileri için karşılaştırılması. *Avrupa Bilim ve Teknoloji Dergisi*, 18, 719-727.

Zhou, F., Pan, H., Gao, Z., Huang, X., Qian, G., Zhu, Y.ve Xiao, F. (2021). Fire prediction based on catboost algorithm. *Mathematical Problems in Engineering*, 1-9.

EKLER

EK-1. K-en yakın komşu algoritmasına ait kod parçası

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# Veri çerçevesini oku
data = pd.read_csv(r"C:\Users\HP\Desktop\dataset.csv")
# NaN değerlere sahip örnekleri çıkar
data = data.dropna()
# "sentence" ve "label" sütunlarını içeren veri çerçevesini oluştur
df = data[["sentence", "label"]]
# Bağımsız değişkenler (X) ve hedef değişken (y) olarak ayır
X = df["sentence"]
y = df["label"]
# Metin verilerini sayısal özelliklere dönüştürmek için CountVectorizer kullanalım
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# Eğitim ve test veri setlerini ayır
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
# K-En Yakın Komşu modelini oluştur
model = KNeighborsClassifier()
# Çapraz doğrulama ile ortalama doğruluk değerini hesapla
cv_scores = cross_val_score(model, X_vectorized, y, cv=5, scoring='accuracy')
mean_accuracy = cv_scores.mean()
# Modeli eğit
model.fit(X_train, y_train)
# Tahmin yap
y_pred = model.predict(X_test)
# Doğruluk (Accuracy) hesapla
accuracy = accuracy_score(y_test, y_pred)
```

EK-1. (devam) K-En Yakın Komşu algoritmasına ait kod parçası

```
# Kesinlik (Precision) hesapla
precision = precision_score(y_test, y_pred, average='weighted')
# Duyarlılık (Recall) hesapla
recall = recall_score(y_test, y_pred, average='weighted')
# F1 puanı hesapla
f1 = f1_score(y_test, y_pred, average='weighted')
# Karmaşıklık matrisi oluştur
cm = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Çapraz Doğrulama ortalama Doğruluk (Accuracy):", mean_accuracy)
print("Doğruluk (Accuracy):", accuracy)
print("Kesinlik (Precision):", precision)
print("Duyarlılık (Recall):", recall)
print("F1 Puanı:", f1)
print("Karmaşıklık Matrisi:\n", cm)
```

EK-2. Naive bayes algoritmasına ait kod parçası

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# Veri çerçevesini oku
data = pd.read_csv(r"C:\Users\HP\Desktop\dataset.csv")
# NaN değerlere sahip örnekleri çıkar
data = data.dropna()
# "sentence" ve "label" sütunlarını içeren veri çerçevesini oluştur
df = data[["sentence", "label"]]
# Bağımsız değişkenler (X) ve hedef değişken (y) olarak ayır
X = df["sentence"]
y = df["label"]
# Metin verilerini sayısal özelliklere dönüştürmek için CountVectorizer kullanalım
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# Naive Bayes modelini oluştur
model = MultinomialNB()
# Çapraz doğrulama yap
scores = cross_val_score(model, X_vectorized, y, cv=5)
# Ortalama doğruluk değerini hesapla
mean_accuracy = scores.mean()
# Eğitim ve test veri setlerini ayır
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
# Modeli eğit
model.fit(X_train, y_train)
# Tahmin yap
y_pred = model.predict(X_test)
# Doğruluk (Accuracy) hesapla
```

EK-2. (devam) Naive bayes algoritmasına ait kod parçası

```
accuracy = accuracy_score(y_test, y_pred)
# Kesinlik (Precision) hesapla
precision = precision_score(y_test, y_pred, average='weighted')
# Duyarlılık (Recall) hesapla
recall = recall_score(y_test, y_pred, average='weighted')
# F1 puanı hesapla
f1 = f1_score(y_test, y_pred, average='weighted')
# Karmaşıklık matrisi oluştur
cm = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Çapraz Doğrulama Ortalama Doğruluk (Accuracy):", mean_accuracy)
print("Doğruluk (Accuracy):", accuracy)
print("Kesinlik (Precision):", precision)
print("Duyarlılık (Recall):", recall)
print("F1 Puanı:", f1)
print("Karmaşıklık Matrisi:\n", cm)
```

EK-3. Karar ağacı algoritmasına ait kod parçası

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# Veri çerçevesini oku
data = pd.read_csv(r"C:\Users\HP\Desktop\dataset.csv")
# NaN değerlere sahip örnekleri çıkar
data = data.dropna()
# "sentence" ve "label" sütunlarını içeren veri çerçevesini oluştur
df = data[["sentence", "label"]]
# Bağımsız değişkenler (X) ve hedef değişken (y) olarak ayır
X = df["sentence"]
y = df["label"]
# Metin verilerini sayısal özelliklere dönüştürmek için CountVectorizer kullanalım
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# Eğitim ve test veri setlerini ayır
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
# Karar Ağacı modelini oluştur
model = DecisionTreeClassifier()
# Çapraz doğrulama ile ortalama doğruluk değerini hesapla
cv_scores = cross_val_score(model, X_vectorized, y, cv=5, scoring='accuracy')
mean_accuracy = cv_scores.mean()
# Modeli eğit
model.fit(X_train, y_train)
# Tahmin yap
y_pred = model.predict(X_test)
# Doğruluk (Accuracy) hesapla
accuracy = accuracy_score(y_test, y_pred)
```

EK-3. (devam) Karar ağacı algoritmasına ait kod parçası

```
# Kesinlik (Precision) hesapla
precision = precision_score(y_test, y_pred, average='weighted')
# Duyarlılık (Recall) hesapla
recall = recall_score(y_test, y_pred, average='weighted')
# F1 puanı hesapla
f1 = f1_score(y_test, y_pred, average='weighted')
# Karmaşıklık matrisi oluştur
cm = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Çapraz Doğrulama ortalama Doğruluk (Accuracy):", mean_accuracy)
print("Doğruluk (Accuracy):", accuracy)
print("Kesinlik (Precision):", precision)
print("Duyarlılık (Recall):", recall)
print("F1 Puanı:", f1)
print("Karmaşıklık Matrisi:\n", cm)
```

EK-4. Destek vektör makinesi algoritmasına ait kod parçası

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# Veri çerçevesini oku
data = pd.read_csv(r"C:\Users\HP\Desktop\dataset.csv")
# NaN değerlere sahip örnekleri çıkar
data = data.dropna()
# "sentence" ve "label" sütunlarını içeren veri çerçevesini oluştur
df = data[["sentence", "label"]]
# Bağımsız değişkenler (X) ve hedef değişken (y) olarak ayır
X = df["sentence"]
y = df["label"]
# Metin verilerini sayısal özelliklere dönüştürmek için CountVectorizer kullanalım
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# Eğitim ve test veri setlerini ayır
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
# Destek Vektör Makinesi modelini oluştur
model = SVC()
# Çapraz doğrulama ile ortalama doğruluk değerini hesapla
cv_scores = cross_val_score(model, X_vectorized, y, cv=5, scoring='accuracy')
mean_accuracy = cv_scores.mean()
# Modeli eğit
model.fit(X_train, y_train)
# Tahmin yap
y_pred = model.predict(X_test)
# Doğruluk (Accuracy) hesapla
accuracy = accuracy_score(y_test, y_pred)
```

EK-4. (devam) Destek vektör makinesi algoritmasına ait kod parçası

```
# Kesinlik (Precision) hesapla
precision = precision_score(y_test, y_pred, average='weighted')
# Duyarlılık (Recall) hesapla
recall = recall_score(y_test, y_pred, average='weighted')
# F1 puanı hesapla
f1 = f1_score(y_test, y_pred, average='weighted')
# Karmaşıklık matrisi oluştur
cm = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Çapraz Doğrulama ortalama Doğruluk (Accuracy):", mean_accuracy)
print("Doğruluk (Accuracy):", accuracy)
print("Kesinlik (Precision):", precision)
print("Duyarlılık (Recall):", recall)
print("F1 Puanı:", f1)
print("Karmaşıklık Matrisi:\n", cm)
```

EK-5. Gaussian naive bayes algoritmasına ait kod parçası

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
from sklearn.naive_bayes import GaussianNB
# Veri çerçevesini oku
data = pd.read_csv(r"C:\Users\HP\Desktop\dataset.csv")
# NaN değerlere sahip örnekleri çıkar
data = data.dropna()
# "sentence" ve "label" sütunlarını içeren veri çerçevesini oluştur
df = data[["sentence", "label"]]
# Bağımsız değişkenler (X) ve hedef değişken (y) olarak ayır
X = df["sentence"]
y = df["label"]
# Metin verilerini sayısal özelliklere dönüştürmek için CountVectorizer kullanalım
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# Eğitim ve test veri setlerini ayır
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
# Yapay Sinir Ağı modelini oluştur
model = GaussianNB()
# Çapraz doğrulama ile ortalama doğruluk değerini hesapla
cv_scores = cross_val_score(model, X_vectorized.toarray(), y, cv=5, scoring='accuracy')
mean_accuracy = cv_scores.mean()
# Modeli eğit
model.fit(X_train.toarray(), y_train)
# Tahmin yap
y_pred = model.predict(X_test.toarray())
# Doğruluk (Accuracy) hesapla
```

EK-5. (devam) Gaussian naive bayes algoritmasına ait kod parçası

```
accuracy = accuracy_score(y_test, y_pred)
# Kesinlik (Precision) hesapla
precision = precision_score(y_test, y_pred, average='weighted')
# Duyarlılık (Recall) hesapla
recall = recall_score(y_test, y_pred, average='weighted')
# F1 puanı hesapla
f1 = f1_score(y_test, y_pred, average='weighted')
# Karmaşıklık matrisi oluştur
cm = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Çapraz Doğrulama ortalama Doğruluk (Accuracy):", mean_accuracy)
print("Doğruluk (Accuracy):", accuracy)
print("Kesinlik (Precision):", precision)
print("Duyarlılık (Recall):", recall)
print("F1 Puanı:", f1)
print("Karmaşıklık Matrisi:\n", cm)
```

EK-6. Rastgele ağaçlar algoritmasına ait kod parçası

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# Veri çerçevesini oku
data = pd.read_csv(r"C:\Users\HP\Desktop\dataset.csv")
# NaN değerlere sahip örnekleri çıkar
data = data.dropna()
# "sentence" ve "label" sütunlarını içeren veri çerçevesini oluştur
df = data[["sentence", "label"]]
# Bağımsız değişkenler (X) ve hedef değişken (y) olarak ayır
X = df["sentence"]
y = df["label"]
# Metin verilerini sayısal özelliklere dönüştürmek için CountVectorizer kullanalım
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# RandomForestClassifier modelini oluştur
model = RandomForestClassifier(n_estimators=100, random_state=42)
# Çapraz doğrulama yap
scores = cross_val_score(model, X_vectorized, y, cv=5)
# Ortalama doğruluk değerini hesapla
mean_accuracy = scores.mean()
# Eğitim ve test veri setlerini ayır
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
# Modeli eğit
model.fit(X_train, y_train)
# Tahmin yap
y_pred = model.predict(X_test)
# Doğruluk (Accuracy) hesapla
```

EK-6. (devam) Rastgele ağaçlar algoritmasına ait kod parçası

```
accuracy = accuracy_score(y_test, y_pred)
# Kesinlik (Precision) hesapla
precision = precision_score(y_test, y_pred, average='weighted')
# Duyarlılık (Recall) hesapla
recall = recall_score(y_test, y_pred, average='weighted')
# F1 puanı hesapla
f1 = f1_score(y_test, y_pred, average='weighted')
# Karmaşıklık matrisi oluştur
cm = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Çapraz Doğrulama Ortalama Doğruluk (Accuracy):", mean_accuracy)
print("Doğruluk (Accuracy):", accuracy)
print("Kesinlik (Precision):", precision)
print("Duyarlılık (Recall):", recall)
print("F1 Puanı:", f1)
print("Karmaşıklık Matrisi:\n", cm)
```

EK-7. Lojistik regresyon algoritmasına ait kod parçası

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# Veri çerçevesini oku
data = pd.read_csv(r"C:\Users\HP\Desktop\dataset.csv")
# NaN değerlere sahip örnekleri çıkar
data = data.dropna()
# "sentence" ve "label" sütunlarını içeren veri çerçevesini oluştur
df = data[["sentence", "label"]]
# Bağımsız değişkenler (X) ve hedef değişken (y) olarak ayır
X = df["sentence"]
y = df["label"]
# Metin verilerini sayısal özelliklere dönüştürmek için CountVectorizer kullanalım
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# Lojistik Regresyon modelini oluştur
model = LogisticRegression()
# Çapraz doğrulama yap
scores = cross_val_score(model, X_vectorized, y, cv=5)
# Ortalama doğruluk değerini hesapla
mean_accuracy = scores.mean()
# Eğitim ve test veri setlerini ayır
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
# Modeli eğit
model.fit(X_train, y_train)
# Tahmin yap
y_pred = model.predict(X_test)
# Doğruluk (Accuracy) hesapla
```

EK-7. (devam) Lojistik regresyon algoritmasına ait kod parçası

```
accuracy = accuracy_score(y_test, y_pred)
# Kesinlik (Precision) hesapla
precision = precision_score(y_test, y_pred, average='weighted')
# Duyarlılık (Recall) hesapla
recall = recall_score(y_test, y_pred, average='weighted')
# F1 puanı hesapla
f1 = f1_score(y_test, y_pred, average='weighted')
# Karmaşıklık matrisi oluştur
cm = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Çapraz Doğrulama Ortalama Doğruluk (Accuracy):", mean_accuracy)
print("Doğruluk (Accuracy):", accuracy)
print("Kesinlik (Precision):", precision)
print("Duyarlılık (Recall):", recall)
print("F1 Puanı:", f1)
print("Karmaşıklık Matrisi:\n", cm)
```

EK-8. Adaboost algoritmasına ait kod parçası

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# Veri çerçevesini oku
data = pd.read_csv(r"C:\Users\HP\Desktop\dataset.csv")
# NaN değerlere sahip örnekleri çıkar
data = data.dropna()
# "sentence" ve "label" sütunlarını içeren veri çerçevesini oluştur
df = data[["sentence", "label"]]
# Bağımsız değişkenler (X) ve hedef değişken (y) olarak ayır
X = df["sentence"]
y = df["label"]
# Metin verilerini sayısal özelliklere dönüştürmek için CountVectorizer kullanalım
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# Eğitim ve test veri setlerini ayır
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
# Adaboost modelini oluştur
model = AdaBoostClassifier()
# Çapraz doğrulama ile ortalama doğruluk değerini hesapla
cv_scores = cross_val_score(model, X_vectorized, y, cv=5, scoring='accuracy')
mean_accuracy = cv_scores.mean()
# Modeli eğit
model.fit(X_train, y_train)
# Tahmin yap
y_pred = model.predict(X_test)
# Doğruluk (Accuracy) hesapla
accuracy = accuracy_score(y_test, y_pred)
```

EK-8. (devam) Adaboost algoritmasına ait kod parçası

```
# Kesinlik (Precision) hesapla
precision = precision_score(y_test, y_pred, average='weighted')
# Duyarlılık (Recall) hesapla
recall = recall_score(y_test, y_pred, average='weighted')
# F1 puanı hesapla
f1 = f1_score(y_test, y_pred, average='weighted')
# Karmaşıklık matrisi oluştur
cm = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Çapraz Doğrulama ortalama Doğruluk (Accuracy):", mean_accuracy)
print("Doğruluk (Accuracy):", accuracy)
print("Kesinlik (Precision):", precision)
print("Duyarlılık (Recall):", recall)
print("F1 Puanı:", f1)
print("Karmaşıklık Matrisi:\n", cm)
```

EK-9. Catboost algoritmasına ait kod parçası

```
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_extraction.text import TfidfVectorizer
from catboost import CatBoostClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# Veri setini yükle
data_path = "C:/Users/HP/Desktop/dataset.csv"
df = pd.read_csv(data_path)
# NaN değerlere sahip örnekleri çıkar
df = df.dropna()
# Veri setini özellikler (X) ve etiketler (y) olarak ayır
X = df['sentence']
y = df['label']
# Veri setini eğitim ve test olarak böle
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Metin özelliklerini vektörlere dönüştür
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
# CATBOOST sınıflandırıcıyı tanımla
model = CatBoostClassifier()
# Çapraz doğrulama ile ortalama doğruluk değerini hesapla
cv_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy')
mean_accuracy = cv_scores.mean()
# Modeli eğit
model.fit(X_train, y_train)
# Test veri seti üzerinde tahminler yap
y_pred = model.predict(X_test)
# Değerlendirme metriklerini hesapla
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
```

EK-9. (devam) Catboost algoritmasına ait kod parçası

```
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Çapraz Doğrulama ortalama Accuracy:", mean_accuracy)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:")
print(confusion_mat)
```

EK-10. Xgboost algoritmasına ait kod parçası

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import cross_val_score, train_test_split
import xgboost as xgb
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# Veri çerçevesini oku
data = pd.read_csv(r"C:\Users\HP\Desktop\dataset.csv")
# NaN değerlere sahip örnekleri çıkar
data = data.dropna()
# "sentence" ve "label" sütunlarını içeren veri çerçevesini oluştur
df = data[["sentence", "label"]]
# Bağımsız değişkenler (X) ve hedef değişken (y) olarak ayır
X = df["sentence"]
y = df["label"]
# Metin verilerini sayısal özelliklere dönüştürmek için CountVectorizer kullanalım
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# XGBoost modelini oluştur
model = xgb.XGBClassifier()
# Çapraz doğrulama yap
scores = cross_val_score(model, X_vectorized, y, cv=5)
# Ortalama doğruluk değerini hesapla
mean_accuracy = scores.mean()
# Eğitim ve test veri setlerini ayır
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
# Modeli eğit
model.fit(X_train, y_train)
# Tahmin yap
y_pred = model.predict(X_test)
# Doğruluk (Accuracy) hesapla
```

EK-10. (devam) Xgboost algoritmasına ait kod parçası

```
accuracy = accuracy_score(y_test, y_pred)
# Kesinlik (Precision) hesapla
precision = precision_score(y_test, y_pred, average='weighted')
# Duyarlılık (Recall) hesapla
recall = recall_score(y_test, y_pred, average='weighted')
# F1 puanı hesapla
f1 = f1_score(y_test, y_pred, average='weighted')
# Karmaşıklık matrisi oluştur
cm = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Ortalama Doğruluk (Accuracy):", mean_accuracy)
print("Doğruluk (Accuracy):", accuracy)
print("Kesinlik (Precision):", precision)
print("Duyarlılık (Recall):", recall)
print("F1 Puanı:", f1)
print("Karmaşıklık Matrisi:\n", cm)
```

EK-11. Lightgbm algoritmasına ait kod parçası

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, KFold
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
import lightgbm as lgb
# Veri setini yükle
data = pd.read_csv('C:/Users/HP/Desktop/dataset.csv')
# NaN değerlere sahip satırları filtrele
data = data.dropna()
# Veri setini özellikler (X) ve hedef (y) olarak ayır
X = data['sentence'].values
y = data['label'].values
# Veri setini eğitim ve test kümelerine ayır
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# CountVectorizer ile metin verilerini vektörlere dönüştürme
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
# LightGBM veri setlerini oluşturma
train_data = lgb.Dataset(X_train_vectorized.astype(float), label=y_train.astype(int))
test_data = lgb.Dataset(X_test_vectorized.astype(float), label=y_test.astype(int))
# Model parametrelerini tanımlama
params = {
    'objective': 'binary',
    'metric': ['binary_logloss']}
# Modeli eğitme
model = lgb.train(params, train_data)
# Test veri kümesi üzerinde tahmin yapma
y_pred = model.predict(X_test_vectorized.astype(float))
# Tahminleri ikili sınıflara dönüştürme
```

EK-11. (devam) Lightgbm algoritmasına ait kod parçası

```
y_pred_binary = np.where(y_pred >= 0.5, 1, 0)
# Performans metriklerini hesaplama
accuracy = accuracy_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
recall = recall_score(y_test, y_pred_binary)
f1 = f1_score(y_test, y_pred_binary)
confusion_mat = confusion_matrix(y_test, y_pred_binary)
# Sonuçları yazdırma
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:\n", confusion_mat)
```

EK-12. Gradient tree boosting algoritmasına ait kod parçası

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
# Veri çerçevesini oku
data = pd.read_csv(r"C:\Users\HP\Desktop\dataset.csv")
# NaN değerlere sahip örnekleri çıkar
data = data.dropna()
# "sentence" ve "label" sütunlarını içeren veri çerçevesini oluştur
df = data[["sentence", "label"]]
# Bağımsız değişkenler (X) ve hedef değişken (y) olarak ayır
X = df["sentence"]
y = df["label"]
# Metin verilerini sayısal özelliklere dönüştürmek için CountVectorizer kullanalım
vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)
# Eğitim ve test veri setlerini ayır
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
# GradientBoostingClassifier modelini oluştur
model = GradientBoostingClassifier()
# Çapraz doğrulama ile ortalama doğruluk değerini hesapla
cv_scores = cross_val_score(model, X_vectorized, y, cv=5, scoring='accuracy')
mean_accuracy = cv_scores.mean()
# Modeli eğit
model.fit(X_train, y_train)
# Tahmin yap
y_pred = model.predict(X_test)
# Doğruluk (Accuracy) hesapla
accuracy = accuracy_score(y_test, y_pred)
```

EK-12. (devam) Gradient tree boosting algoritmasına ait kod parçası

```
# Kesinlik (Precision) hesapla
precision = precision_score(y_test, y_pred, average='weighted')
# Duyarlılık (Recall) hesapla
recall = recall_score(y_test, y_pred, average='weighted')
# F1 puanı hesapla
f1 = f1_score(y_test, y_pred, average='weighted')
# Karmaşıklık matrisi oluştur
cm = confusion_matrix(y_test, y_pred)
# Sonuçları yazdır
print("Çapraz Doğrulama ortalama Doğruluk (Accuracy):", mean_accuracy)
print("Doğruluk (Accuracy):", accuracy)
print("Kesinlik (Precision):", precision)
print("Duyarlılık (Recall):", recall)
print("F1 Puanı:", f1)
print("Karmaşıklık Matrisi:\n", cm)
```



Gazili olmak ayrıcalıktır…