T.C.
VAN YÜZÜNCÜ YIL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
İSTATİSTİK ANABİLİM DALI

# PHISHING SALDIRISINDA KULLANILAN WEB SİTELERİNİN MAKİNE ÖĞRENMESİ ALGORİTMALARI YARDIMIYLA TESPİTİ VE UYGULAMASI

YÜKSEK LİSANS TEZİ

HAZIRLAYAN: Fırat KAPAR
DANIŞMAN: Prof. Dr. H. Eray ÇELİK

VAN-2018

T.C.
VAN YÜZÜNCÜ YIL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
İSTATİSTİK ANABİLİM DALI

# PHISHING SALDIRISINDA KULLANILAN WEB SİTELERİNİN MAKİNE ÖĞRENMESİ ALGORİTMALARI YARDIMIYLA TESPİTİ VE UYGULAMASI

YÜKSEK LİSANS TEZİ

HAZIRLAYAN: Fırat KAPAR

VAN-2018

## KABUL VE ONAY SAYFASI

İstatistik Anabilim Dalı'nda Prof. Dr. H.Eray ÇELİK danışmanlığında, Fırat KAPAR tarafından sunulan "**Phishing Saldırısında Kullanılan Web Sitelerinin Makine Öğrenmesi Algoritmaları Yardımıyla Tespiti Ve Uygulaması**" isimli bu çalışma Lisansüstü Eğitim ve Öğretim Yönetmeliği'nin ilgili hükümleri gereğince ----/----/ 2018 tarihinde aşağıdaki jüri tarafından oy birliği ile başarılı bulunmuş ve Yüksek Lisans tezi olarak kabul edilmiştir.

Başkan: İmza:

Prof. Dr. H. Eray ÇELİK

Üye: İmza:

Doç. Dr. Hamit MİRTAGİOĞLU

Üye: İmza:

Dr. Öğr. Üyesi Recep ÖZADAĞ

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …./..../…... tarih ve ..... sayılı kararı ile onaylanmıştır.

Enstitü Müdürü

## TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Fırat KAPAR

# ÖZET

## PHISHING SALDIRISINDA KULLANILAN WEB SİTELERİNİN MAKİNE ÖĞRENMESİ ALGORİTMALARI YARDIMIYLA TESPİTİ VE UYGULAMASI

KAPAR, Fırat
Yüksek Lisans Tezi, İstatistik Anabilim Dalı
Tez Danışmanı: Prof. Dr. H. Eray ÇELİK
Ağustos 2018, 67 sayfa

Kimlik Avı (Phishing), son zamanlarda internet kullanıcıları açısından güvenlik riski yaratan en popüler saldırı türlerinden biridir. Saldırıda kullanılan bu yöntem için kullanılan web siteleri kolayca çoğaltılabilmekte ve saldırı sırasında kullanılan sosyal mühendislik yöntemlerinin tespit edilmesini zorlaştırmaktadır. Sınıflandırma algoritmaları kullanılarak bu amaçla hazırlanan web sitelerinin saldırıları anında tespit edilebilmektedir. Bu çalışmada Lojistik Regresyon, KNN ve Naive Bayes algoritmaları kullanılarak, PhishTank ve Google arama sonuçlarında elde edilen URL setleri üzerinde modeller eğitilmiştir. Modellerin başarısı; Doğruluk, Hassasiyet, Geri Çağırma ve F-skoru ile değerlendirilmiş ve algoritmaların başarıları karşılaştırılmıştır. Lojistik Regresyon algoritmasının diğer algoritmalara göre daha başarılı olduğu tespit edilmiştir.

**Anahtar kelimeler:** Makine öğrenmesi, Makine öğrenmesi algoritmaları, Phishing.

## ABSTRACT

### DETECTION OF PHISHING WEBSITES VIA MACHINE LEARNING ALGORITHMS AND APPLICATION

KAPAR, Fırat
M.Sc. Thesis, Department of Statistics
Supervisor: Prof. Dr. H. Eray ÇELİK
August 2018, 67 pages

Phishing is one of the most popular types of attacks that have recently created a security risk for internet users. The web sites used for this offensive method can easily be multiplied and make it difficult to determine the social engineering methods used during the attack. Using the classification algorithms, the attacks of the web sites prepared for this purpose can be detected instantaneously. In this study, models were trained on URL sets obtained from PhishTank and Google search results using Logistic Regression, KNN and Naive Bayes algorithms. The success of the models; Accuracy, Sensitivity, Recall, and F-score were evaluated and the successes of the algorithms were compared. The Logistic Regression algorithm is found to be that more successful than the other algorithms.



**Keywords**: Machine learning, Machine learning algorithms, Phishing.

# ÖN SÖZ

Bu tez çalışmasında verdiği desteklerden dolayı Yüzüncü Yıl Üniversitesi Bilgisayar Bilimleri Araştırma ve Uygulama Merkezi'nde çalışan bütün arkadaşlarıma, her türlü ilgi ve yardımlarını esirgemeyen danışmanım Sayın Prof. Dr. H. Eray ÇELİK'e, değerli katkılarından dolayı arkadaşım Arş. Gör. Fatih ULUDAĞ'a, ayrıca çalışmalarımda bana her zaman destek olan aileme ve değerli dostlarıma en içten teşekkürlerimi sunarım.

2018

Fırat KAPAR

# İÇİNDEKİLER

**Sayfa**

**Sayfa**

**Sayfa**

# ÇİZELGELER LİSTESİ

**Çizelge** **Sayfa**

# ŞEKİLLER LİSTESİ

**Şekil** **Sayfa**

## SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

| Kısaltmalar | Açıklama |
|---|---|
| **TF-IDF** | Term Frequency — Inverse Document Frequency |
| **HTML** | Hypertext Markup Language |
| **HTTP** | Hypertext Transfer Protocol |
| **HTTPS** | Secure Hypertext Transfer Protocol |
| **PHP** | Hypertext Preprocessor |
| **CART** | Classification and Regression Trees |
| **BART** | Bayesian Additive Regression Trees |
| **SVM** | Support Vector Machine |
| **VOIP** | Voice over Internet Protocol |
| **IP** | Internet Protocol |
| **URL** | Uniform Resource Locator |
| **FTP** | File Transfer Protocol |
| **SSH** | Secure Shell |
| **TELNET** | Telecommunication Network |
| **RDP** | Remote Desctop Protocol |
| **DNS** | Domain Name Server |
| **XML** | eXtensible Markup Language |

# EKLER DİZİNİ

**Sayfa**

# 1. GİRİŞ

Phishing, internet dolandırıcılarının kullandığı ve orijinal bir web sayfasının kopyasını oluşturarak kişilerin bilgilerini çalmaya yönelik kullanılan bir aldatma tekniğidir (Milletary ve Center, 2005). Phishing, son zamanlarda internet kullanıcılarının güvenliğini risk altına sokan en popüler saldırı türlerinden biridir. Bu tür bir saldırıda kullanılan web siteleri kolayca çoğaltılır ve saldırı sırasında kullanılan sosyal mühendislik yöntemleri tespit edilmesini zorlaştırır.

Phishing saldırıları, nispeten yeni bir web tehdidi olmasına rağmen, bu saldırıların e-ticaret endüstrisi üzerinde büyük etkileri bulunmaktadır. Phishing siteleri yasal orijinallerine çok benzeyip insanları aldatmaya yönelik amaçlarla oluşturulmaktadır.

Phishing, internet kullanıcılarının dikkatsizliklerinden faydalanarak kişisel bilgilerin ele geçirilmesini amaçlayan yeni nesil bir siber suç türüdür. Bir phishing saldırganı bir web sitesinin HTML kodlarını kopyalayarak oluşturduğu sahte bir web sitesi yardımıyla sosyal mühendislik tekniklerini de kullanarak hedeflediği kişinin kredi kartı, ikamet bilgisi, web sitesi giriş bilgisi, sosyal medya hesapları gibi kişisel bilgilerini ele geçirmeye çalışır. Kullanıcılara gönderilen "hesabınıza biri giriş yapmaya çalışıyor" , "güvenliğiniz için acilen şifrenizi değiştirmeniz gerekmekte" vb. sistem yöneticilerinden gelmiş gibi görünen mesajları kullanarak onların bir anlık zaafından faydalanılmaya çalışılmaktadır. Phishing saldırıları sadece web üzerinden gönderilen metin mesajları veya e-mail şeklinde olmayıp telefon araması, VoIP phishing veya vhishing, mobil telefonlara gönderilen SMS olarak, smishing, şeklinde de gerçekleştirilmektedir. İnternet kullanıcılarının dikkatsizlikleri, teknoloji okur-yazarlık seviyelerinin düşük olması veya bu tip saldırılardan haberdar olmamaları sebebiyle bu saldırılara gün geçtikçe daha fazla maruz kalmalarına sebep olmaktadır (Hong, 2012).

Internet, hayatımızda giderek önemli ve gerekli bir araç haline gelmiştir. Bununla birlikte, internet kullanıcılarının, web üzerinden gelen farklı tehditlere karşı kişisel güvenlikleri oldukça zayıftır. Bilindiği gibi web üzerinden yapılan alışveriş ve bankacılılık işlemleri gibi doğrudan maddi riskler taşıyan işlemler de hayatımızın vazgeçilmez bir parçası haline gelmiştir. Bu durum internet dolandırıcılarını da bu alana gün geçtikçe daha fazla çekmektedir. Dolayısıyla internet kullanıcıları için doğabilecek

web tehditleri, daha riskli durumlar ortaya çıkarabilmektedir. Phishing, güvenli bir web sitesi görünümündeki bazı sitelerin, kişinin kullanıcı adı, şifre, sosyal güvenlik numarası ve kredi kartı numarası gibi özel bilgilerini elde etmeyi amaçlayan bir yöntem olarak açıklanmaktadır (Van der Merwe ve ark., 2005; Ramzan, 2010).

Literatürde Phishing web sayfalarına yönelik birçok tanım verilmiştir. Phishing bir saldırganın sahte e-mail ve taklit web sitesi kullanarak dikkatsiz müşterilerin banka hesap bilgisi ve web sitesi giriş bilgileri gibi gizli bilgilerini elde etmeye yönelik çevrimiçi bir saldırı türüdür (Topkara ve ark. 2005).

Martin ve ark. (2011), Phishing, saldırganların özel bilgileri yasal olmayan bir şekilde bir e-mail veya web sitesi kullanımıyla elde etmeye çalıştıkları bir internet aldatmaca türü olarak tanımlamışlardır.

Phishing virüs ve hacking gibi diğer internet tehdit türleri ile karşılaştırıldığında çok daha hızlı büyüyen bir internet suçudur. İletişimde ana form olarak internetin geniş kullanımından dolayı phishing saldırıları aşağıdaki gibi farklı şekillerde uygulanabilir (Alnajim ve Munro, 2009);

1. Email den email'e,
2. Email den web sitesine,
3. Web sitesinden web sitesine,
4. Tarayıcıdan web sitesine.

Bir phishing web sayfası alt yapısının çok kısa bir sürede hazırlanabilmesi ve bunların önlenmesi için geliştirilen tekniklerin yetersizliği bu saldırıların yıldan yıla artış göstermesine sebep olmuştur. Anti Phishing Working Group raporlarına göre 2004 yılının Mart ayında 402 phishing saldırısı belirlenmişken, 2016 yılının Kasım ayında bu sayı 118,928 olarak saptanmıştır. 2016 yılında toplam saldırı sayısı ise 1,220,523 olarak belirlenmiş olup, 2016 yılında saldırılar bir önceki yıla göre %65 düzeyinde artış göstermiştir (APWG, 2016).

Phishing, son kullanıcıları sahte web sitelerini ziyaret etmeye ve kişisel bilgileri (ör. Kullanıcı kimliği, şifre) vermek üzere kullanan web tabanlı bir saldırı olarak açıklanmaktadır. Çalınan bilgiler, çevrim içi para aklama gibi birçok gayri yasal etkinliğin başlangıç noktasıdır. Phishing saldırıları, ticari kuruluşlara ve son

kullanıcılara milyarlarca dolara mal olmakta ve e-ticaret endüstrisini tehlikeye atmaktadır. Bu nedenle, kimlik avı saldırılarına karşı gerekli önelmeleri almak oldukça önemlidir (Shahriar ve Zulkernine, 2012).

Phishing Saldırılarını başlatmak için sosyal mühendislik ve teknik bilgiler genellikle birleştirilerek kullanılmaktadır. Genellikle, bir phishing saldırısı, potansiyel kurbanlara e-posta içindeki bir URL bağlantısını gönderen, bilgilerini güncellemelerini veya doğrulamalarını isteyen gerçek bir e-postanın gönderilmesiyle başlamaktadır (Rami ve ark., 2013).

Phishing saldırılarının önsel olarak tespiti, internet kullanıcılarının güvenli internet kullanımı konusunda bilinçlendirilmesi ve farkındalık düzeylerinin arttırılması ve bu konuda gerekli yasal düzenlemelerin yapılması oldukça önem arzetmektedir.

Bu tez çalışmasında yukarıda değinilen üç paramatreden biri olan Phishing saldırılarının önsel olarak tespit edilmesinde sınıflandırma algoritmaları kullanılarak, Phishing için kullanılan web sitelerinin tespit edilmesine çalışılmıştır.

## 2. KAYNAK BİLDİRİŞLERİ

Bu bölümde Phishing Web sayfalarının belirlenmesinde kullanılan bazı teknik ve yapılan çalışmalar hakkında bilgiler derlenip sunulmuştur.

Phishing sitelerinin belirlenmesinde literatürde birçok metot ve tanımlama bulunmaktadır.

Kan ve Thi (2005), Kötü niyetli web sitelerinin belirlenmesinde makine öğrenmesi algoritmalarını kullanan ilk araştırmacılardır.

Fu ve ark. (2006), Earth Mover uzaklığı kullanarak şüpheli web sitesi ile yasal web siteleri arasında ki görsel benzerliği hesaplanarak, phishing sitelerini belirlemeye çalışmışlardır. Yasal web sitelerinin kolaylıkla kopyalanıp görsel olarak tamamen aynı phishing sitelerinin oluşturulabilmesi bu yöntemi etkisiz kılmaktadır.

Pan ve Ding (2006), phishing sitesi belirlemede seçilen DOM özelliklerinden (site başlığı, meta bilgisi, yayın hakları metni vb. ) ve HTTP işlemlerinden (Server Form Handler vb.) elde edilen web sitesi niteliklerinin kullanıldığı bir yöntem önermişlerdir.

Zhang ve ark. (2007), Metin tabanlı phishing algılama tekniği olan CANTINA' da, Term Frequency—Inverse Document Frequency (TF-IDF) algoritması kullanarak web sitelerinden anahtar kelimeleri çıkarılmakta ve elde ettiği bu anahtar kelimeleri Google arama motorunda taranıp web sitesinin sonuçlarda gösterilip gösterilmediğini incelemişlerdir.

Garera ve ark. (2007), 18 nitelik üzerinde lojistik regresyon analizi kullanarak phishing sitelerini sınıflandırmıştır.

Abu-Nimeh ve ark. (2007), Phishing sitelerini belirlemede Lojistik Regresyon, Sınıflandırma ve Regresyon Ağaçları (CART), Bayesyen Katkılı Regresyon Ağaçları (BART), Destek Vektör Makinesi (SVM), Random Forests ve Yapay Sinir Ağları'nı içeren 6 tane makine öğrenmesi algoritmasının tahmin doğruluğunu karşılaştıran bir çalışma yapmıştır. Bu çalışmada Yapay Sinir Ağları ve Random Forests algoritmlaları diğer metotlara göre daha iyi performans gösterdiği belirlenmiştir. Fakat her iki algoritmada yüksek hata oranı ve yanlış pozitif oranına sahip olduğu vurgulanmıştır.

Genkina (2007), Net Trust adını verdiği bir tarayıcı eklentisi geliştirmiştir. Bu tarayıcı eklentisi saldırganın kontrolü altında olmayan nitelikleri kullanmaktadır.

Bu yaklaşımda kötü niyetli web siteleri kullanıcıların lokal geçmiş verilerini ve sosyal ağlardaki girdileri bilgi olarak toplanarak belirlenmektedir. Fakat bu yaklaşım kullanıcı kötü niyetli web sitesini ilk defa ziyaret ettiğinde başarılı olmamaktadır.

Aburrous ve ark. (2010), elektronik bankacılıkta karşılaşılan phishing sitelerinin belirlenmesi için zeki bir sistem önermişlerdir. Bu çalışmada önerilen model elektronik bankacılık phishing sitelerinin elemanlarını karakterize etmek için veri madenciliği yöntemleriyle birleştirilen bulanık mantığa dayanmaktadır.

Xiang ve ark. (2011), Gelişmiş HTML ayrıştırma ve metin işleme tekniklerini kullanılarak CANTINA+'ı geliştirmişlerdir. Ancak her iki çalışmada da ingilizce olmayan dillerde başarılı sonuçlar elde edememişler.

Le ve ark. (2011), URL uzunluğu, domain ismi, sembol sayısı gibi nitelikleri kullanarak phishing olasılığının hesaplandığı bir sınıflandırma algoritması kullanmışlardır.

Liu ve ark. (2012), Phishing sitelerinin belirlenmesi için web sitesinin kimlik analizine dönük Semantic Link Network (SLN) yöntemini önermişlerdir. SLN sorgulanan web sitesi ile ilişkili web sitesi kümesine bağlı ağırlıklı linklerden oluşmaktadır. SLN için bir kaç metrik, hedef kimliğini bulmak ve sorgulanan web sitesinin yasal olup olmadığını belirlemek için kullanılmıştır. Liu ve ark.(2012), SLN modülü yerine parazit topluluk modülü yerleştirilerek yeni bir sistemi geliştirmişlerdir.

Ramesh ve ark. (2014), Hedef domaini belirlemek için SLN'ye benzer bir strateji uygulamıştır. Ancak bu strateji hedeflenen yasal web sitesine dair herhangi bir URL içermeyen phishing sitelerini belirlemede etkisiz kalmaktadır.

Bazı araştırmacılar bir web sitesinin URL niteliklerinin o web sitesinin phishing sitesi olup olmadığını belirlemede kullanılabileceğini göstermiştir.

Hodzic ve ark. (2016), Phishing web sitesi sınıflandırmalarında hangi algoritmanın daha iyi sonuç verdiğini belirlemek amacıyla Rasgele Orman, C4.5, REP Ağacı, Decision Stump, Hoeffding Ağacı, Rotation Forest ve MLP algoritmaları ile çalışmışlardır. Tüm örnekler; Legal(-1), Şüpheli(0) ve Phishing(1) olarak kategorileştirilmiştir. Araştırmanın sonuçları göstermiştir ki Phishing web sayfalarının sınıflandırılmasında kullandıkları veri setlerinde en iyi performansları Rasgele Orman ve REP Ağacı algoritmaları vermiştir.

Subasi va ark. (2017), Farklı veri madenciliği tekniklerini kullanarak web sayfalarını Meşru ve Phishing olarak kategorileştirmeye çalışmışlardır. Web syafalarını kategorileştirmek için farklı sınıflandırıcılar kullanmışlardır. Sınıflandırıcıların performansının ölçülmesinde *receiver operating characteristic* (ROC) ve F-Ölçüsü kullanılmış, kullanılan sınıflandırma algoritmaları arasında Rastgele Orman sınıflandırma algoritması %97.36 oranında bir doğrulu oranıyla en iyi performansı gösteren algortima olmuştur.

Rao ve Pais (2018), Anti-Phishing tekniklerinin sahip olduğu olumsuzluklara alternatif olarak URL, kaynak kod ve üçüncü taraf hizmetlerden elde edilen sezgisel algoritmalara dayanan yeni bir sınıflandırma modeli önermişlerdir. Modelde sekiz farklı makine öğrenme algoritması kullanılmış ve Random Forest algortiması en iyi sonucu vermiştir.

Jain ve Gupta (2018), Bir web sitesinin URL adresine ait 14 özellik kullanarak PHISH-SAFE adını verdikleri bir anti Phishing sistemi önermişlerdir. Önerdikleri bu sistemde 33,000'den fazla Meşru ve Phishing web sitesini kullanarak SVM ve Naive Bayes sınıflandırma algoritmalarını eğitmişler ve neticede SVM algoritmasının Phishing web sitelerini %90'dan fazla bir doğruluk ile tahmin edebildiğini gözlemlemişlerdir.

# 3. MATERYAL VE YÖNTEM

Bu tez kapsamında yapılan çalışmada sonuca ulaşmak birçok yöntem ve araçtan yararlanılmıştır. Google arama motorundan elde edilen URL adreslerinin toplanmasında, Google Chrome tarayıcısıyla çalışan gInfinity adlı eklenti kullanılarak java script dili ile yazılmış scriptlerden yararlanılmıştır. URL adreslerinine ait niteliklerin toplanmasında PHP Programlama dilinden yararlanılmış ve Makine Öğrenmesi algoritmaları python dili ile birlikte gelen scikit learn kütüphaneleri kullanılmıştır. URL adreslerine ait niteliklerin toplanması Windows İşletim Sistemi ortamında yapılmış ancak python dili ile yazılan makine öğrenmesi algoritmalarının çalıştırılması UBUNTU işletim sisteminde gerçekleştirilmiştir.

## 3.1. Makine Öğrenmesi

Makine öğrenmesi bilgisayarların öğrenmesini sağlayan algoritmaların tasarımı ve analizi ile ilgilenen hızla gelişen bir alandır. Makine öğrenmesi resimlerden nesnelerin otomatik tespiti, ses tanısı, veri tabanlarından bilgi keşfi ve tahminleyici analitik gibi bir çok alanda kullanılmaktadır (Alpaydın, 2016).

Aileler çocuklarına kedi ve köpek arasındaki farkı öğretmek için onlara asla kedi ve köpeğin bilimsel tanımlarını vermezler. Bunun yerine çocuklar bu canlıların gördüklerinde ailelerinden aldıkları bu köpektir veya bu kedidir bilgileriyle eğitilirler. Yeni bir kedi veya köpekle karşılaştıklarında bunları doğru tanımlamalarını doğru bir şekilde yapabiliyorlarsa çocuk öğrenmiş demektir. Tıpkı insanlar gibi bazı görevleri gerçekleştirmek bilgisayarlarda öğrenebilmektedir (Mitchell,1997).

Makine öğrenmesi veriden bilgi çıkarmayla ilgilenen istatistik, yapayzeka ve bilgisayar bilimlerinin kesişiminde bulunan bir alandır ve *tahminleyici analitik* veya *istatistiksel öğrenme* olarak ta isimlendirilmektedir. Makine öğrenmesi metotlarının uygulamaları artık günlük hayatın hemen her köşesinde yer almaya başlamıştır. Facebook, Amazon veya Netflix gibi karmaşık web siteleri başarılarını kullandıkları makine öğrenmesi algoritmalarına borçludur. Ticari uygulamaların dışında makine öğrenmesi veriye dayanan araştırmalarda giderek artan bir etkiye sahiptir.

Mitchell, makine öğrenmeyi bir bilgisayar programını da kapsayacak biçimde bir işteki performansın deneyimle artması olarak tanımlamıştır. Daha açık bir tanımla; bir bilgisayar programı eğer *P* ile ölçülen *T*' de bulunan işlerdeki performansı *E* deneyimine göre artıyorsa *P* performans ölçüsü ve *T* işlerinin bazı sınıfları konusunda *E* deneyiminden öğrenir (Mitchell,1997).

Tanımda sözü geçen;

*T* görevi: Herhangi bir dilden başka bir dile çeviri,

*P* performans ölçüsü: Yapılan çevirilerin doğruluk yüzdesi,

*E* deneyimi: Sistem üzerinde yapılan tüm çeviriler

olarak düşünülebilir.

Makine öğrenmesi, performans optimizasyonu için örnek veri ya da geçmiş deneyimleri kullanarak bilgisayar programlamaktır (Alpaydın, 2016).

Makine öğrenmesi algoritmaları

- Metin veya belge sınıflandırma
- Doğal dil işleme
- Ses tanısı
- Optik karakter tanımlama
- Hesaplamalı biyoloji
- Sahtekarlık tespiti
- Oyun
- Sürücüsüz araç kontrolü
- Tıbbi tanı
- Öneri sistemleri

gibi alanlarda bir çok başarılı uygulamaya sahiptir (Mohri ve ark., 2012).

Makine öğrenmesi problemleri genel olarak 5 ana başlıkta toplanabilir;

1. Sınıflandırma
2. Regresyon
3. Ranking
4. Kümeleme
5. Boyut azaltma veya manifold öğrenmesi. (Mitchell, 1997)

### 3.1.1. Bazı tanım ve terminolojiler

Bu kısımda tez boyunca kullanılacak bazı terimlerin tanımları verilecektir.

**Örnek**: Makine öğrenmesi için kullanılacak veri setindeki gözlemler ve nesneler,

**Nitelik**: Örneği en iyi açıkladığı düşünülen özellikler,

**Etiket**: Örneklere atanan değer veya sınıflar,

**Eğitim verisi:** Öğrenme algoritmasının eğitiminde kullanılan örneklerin oluşturduğu veri seti,

**Test verisi:** Öğrenme algoritmasının performansını değerlendirmek için kullanılan veri seti,

**Hata fonksiyonu:** Tahmin edilen etiket ile gerçek etiket arasındaki farkı veya kaybı ölçen fonksiyon,

**Hipotez kümesi:** Nitelikleri etiketlere dönüştüren fonksiyon ailesi (Alpaydın, 2016).

### 3.1.2. Makine öğrenmesi yaşam döngüsü

Makine öğrenmesi algoritmaları eğitim verisinin türü, eğitim verisinin aldığı metot ve bunun sırası ile öğrenme algoritmasının değerlendirilmesinde kullanılan test verisine göre farklı senaryolara ayrılabilir (Mohri ve ark., 2012).

Bu senaryoları şu şekilde belirlemiştir;

1. Denetimli Öğrenme: Öğrenme algoritması etikete sahip bir eğitim verisi alır ve etiketi bilinmeyen veriler için tahmin yapar. Bu tip öğrenme makine öğrenmesinde en çok kullanılan öğrenme senaryosudur. Sınıflandırma, regresyon ve ranking problemleri denetimli öğrenme türündeki problemlere dahildir.
2. Denetimsiz Öğrenme: Öğrenme algoritması etiketsiz bir eğitim verisine sahiptir ve bilinmeyen noktalar için tahmin yapar. Bu tip öğrenmede etiketsiz verilere sahip olunduğundan öğrenme algoritmasının performans değerlendirmesini niceliksel olarak yapmak zordur. Kümeleme ve boyut azaltma algoritmaları denetimsiz öğrenme algoritmalarıdır.

3. Yarı-denetimli Öğrenme: Öğrenme algoritması hem etiketli eğitim verisini hem de etiketsiz eğitim verisi ile çalışır ve bilinmeyen noktalar için tahmin yapmaya çalışır. Yarı-denetimli öğrenme genel olarak verilerin kolay etiketlerin zor elde edildiği problemlerde kullanılır.
4. Transdüktif Çıkarsama: Yarı- denetimli öğrenmede olduğu gibi öğrenme algoritması etiketsiz eğitim verisi ile birlikte etiketli eğitim verisini alır. Fakat transdüktif çıkarsamanın amacı belli test örneklerinin etiketlerini tahmin etmektir.
5. Çevrimiçi Öğrenme: Daha önceki senaryoların tersine çevrimiçi öğrenme eğitim ve test aşamalarının karıştırıldığı bir öğrenme türüdür. Her bir aşamada öğrenme algoritması bir etiketsiz nokta alır tahmin yapar bu tahmini gerçek etiket ile karşılaştırır ve bir hata hesaplar. Çevrimiçi öğrenmede amaç kümülatif hatayı minimize etmektir.
6. Takviyeli Öğrenme: Takviyeli öğrenmede de test ve eğitim aşamaları karıştırılır. Bilgi toplamak için öğrenme algoritması aktif bir şekilde çevreyle etkileşim içindedir ve bazı durumlarda çevreyi de etkiler ve her bir eylem için bir karşılık alır.
7. Aktif Öğrenme: Öğrenme algoritması uyarlamalı veya interaktif şekilde eğitim örneklerini toplar (Mohri ve ark., 2012).

Bu çalışma kapsamında denetimli öğrenme algoritmaları ele alınacağından, aşağıdaki örnek üzerinde bir denetimli öğrenme algoritması açıklanmıştır.

Bir bölgedeki dairelere ait büyüklük ($m^2$) ve fiyatların (₺) bilgisinin kaydedildiği bir veri setine sahip olunduğu düşünülsün.

Çizelge 3.1. Makine öğrenmesi örnek veri (Ng, 2000)

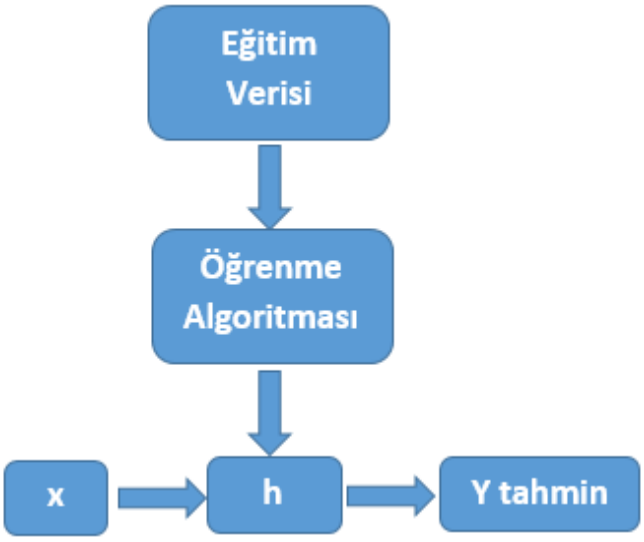
| Büyüklük ($m^2$) | Fiyat (Bin ₺) |
|---|---|
| 120 | 170 |
| 130 | 185 |
| 145 | 225 |
| 180 | 255 |
| 150 | 200 |
| . | . |
| . | . |
| . | . |

Bu veri setini baz; alarak aynı bölgedeki dairelerin fiyatları büyüklüklerinin bir fonksiyonu olarak ifade edilebilir mi? problemi klasik bir denetimli öğrenme problemidir. Her bir örnek $x^{(i)}$ ile gösterilir. Bu örnek için $x^{(i)}$ $i$. dairenin büyüklüğüne karşılık gelmektedir. Etiketler $y^{(i)}$ ile gösterilecektir. $\left(x^{(i)}, y^{(i)}\right)$ çiftine eğitim örneği, $m$ tane eğitim örneğinin oluşturduğu $\left\{\left(x^{(i)}, y^{(i)}\right) \middle| i = 1,2, \dots, m\right\}$ kümesine de eğitim verisi denilmektedir. Bu tez kapsamında örneklerin uzayı $X$ ile, etiket değerlerinin oluşturduğu uzay $Y$ ile gösterilecektir.

Denetimli öğrenme problemi matematiksel olarak;

$$h: X \longrightarrow Y \tag{3.1}$$

şeklinde gösterilebilir. Burada $h$ fonksiyonu hipotez olarak isimlendirilir. $h(x)$ değeri de $x$ örneğine karşılık gelen tahmin değeridir. Denetimli öğrenmede amaç $h(x)$ değerleri ile $x$'in gerçek etiketleri arasındaki farkların toplamını minimum yapacak $h$ hipotezinin belirlenmesidir (Şekil 3.1).




Şekil 3.1. Makine öğrenmesi yaşam döngüsü (Mitchell, 1997).

Eğer etiket değişkeni sürekli ise bu tip bir probleme regresyon, kategorik ise problem sınıflandırma problemi olarak isimlendirilir.

Bir çok problemde örnekler birden çok nitelik ile gösterilir. Daire örneğinde veri seti;

Çizelge 3.2. Makine öğrenmesi çok değişkenli örnek veri çizelgesi (Ng, 2000)

| Büyüklük (m2) | Oda Sayısı | Fiyat (TL) |
|---|---|---|
| 120 | 3 | 170b |
| 130 | 3 | 185b |
| 145 | 4 | 225b |
| 180 | 4 | 255b |
| 150 | 2 | 200b |
| . | . | . |
| . | . | . |
| . | . | . |

Olarak ele alınırsa bu durumda bir örnek $\left(x_1^{(i)}, x_2^{(i)}\right)$ çifti ile gösterilecektir. Bu tip bir problem için denetimli öğrenme algoritması uygulandığı zaman ilk yapılması gereken bir $h$ hipotezinin nasıl tanımlanacağıdır. Bu problem için bir dairenin fiyatı büyüklük ve oda sayısının lineer bir fonksiyonu olarak ifade edilsin.

Bu durumda $h$;

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \tag{3.2}$$

olarak ifade edilebilir. Burada $\theta_i$ ler eğitim verisinden elde edilecek parametrelerdir. Eğitim verisindeki örnekler kullanılarak tahmin edilen $h(x)$ ve $y$ değerleri arasındaki hata minimize edilmeye çalışılır.

Formel bir şekilde hata fonksiyonu

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta\left(x^{(i)}\right) - y^{(i)})^2 \tag{3.3}$$

şeklinde tanımlanır. (Ng, 2000)

Öğrenme algoritmasında amaç $J(\theta)$ yı minimum yapacak $\theta$ nın bulunmasıdır. En dik iniş algoritması bu amaç için kullanılabilir. En dik iniş algoritması bir $\theta$ başlangıç değeriyle başlar ve $j = 0,1,2,\ldots,n$ için;

$$\theta_j \coloneqq \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{3.4}$$

döngüsünü tekrarlar.

Burada $\alpha$ öğrenme hızı olarak isimlendirilmektedir. Bu algoritmanın uygulanması için yukarıdaki denklemin sağ tarafındaki kısmi türevin belirlenmesine ihtiyaç vardır. Kolaylık olması adına elde bir adet örnek olduğu düşünülsün dolayısıyla $J$ deki toplam sembolü göz ardı edilebilir (Ng, 2000).

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{\partial}{\partial\theta_j}\frac{1}{2}(h_\theta(x)-y)^2$$
$$= 2.\frac{1}{2}(h_\theta(x)-y)\frac{\partial}{\partial\theta_j}(h_\theta(x)-y)$$
$$= (h_\theta(x)-y)\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i - y\right)$$
$$= (h_\theta(x)-y)x_j$$

Bir eğitim örneği için yukarıdaki sonuç şu güncelleme kuralını vermektedir.

$$\theta_j \coloneqq \theta_j + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x_j{}^{(i)} \qquad (3.5)$$

Bu kurala ortalama kare güncelleme kuralı denilmektedir (Ng, 2000). Bu sonuç tek bir eğitim örneği için geçerlidir. Birden fazla örnek için bu kuralı kullanmak için iki tane yol bulunmaktadır. Bunlardan birincisi

$$\theta_j \coloneqq \theta_j + \alpha\sum_{i=1}^{m}\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x_j{}^{(i)} \qquad her\ j\ için$$

dir. Toplamın içindeki büyüklük $\frac{\partial}{\partial\theta_j}J(\theta)$ türevine karşılık gelmektedir. Bu güncelleme kuralı $J$ hata fonksiyonunda en dik iniş algoritması olarak bilinmektedir. Bu metot tüm eğitim verisindeki her bir örneği tek tek incelediğinden yığın en dik iniş algoritması (batch gradient descent) olarak ta isimlendirilir.

*tekrarla {*

$1\ den\ m'e\ kadar\ \{$

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x_j{}^{(i)} \qquad (3.6)$$

$her\ j\ için$

}

}

Bu algoritmada veri seti üzerinden tekrarlı bir şekilde geçilir ve her eğitim örneği ile karşılaştığımızda bu eğitim verisinin hatasının gradiyentine göre parametreler güncellenir. Buna stokastik en dik iniş algoritması (stochastic gradient descent) denir. Eğitim verisinin büyük olduğu durumlarda stokastik en dik iniş algoritması yığın en dik iniş algoritmasına tercih edilir (Ng, 2000).

Olasılıksal yorum etiketlerin ve niteliklerin;

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)} \qquad (3.7)$$

modeli ile ilişkili olduğu varsayılsın. Burada $\varepsilon^{(i)}$ gözlenmeyen niteliklere karşılık gelen hata terimini ifade etmektedir. $\varepsilon^{(i)}$ lerin 0 ortalamalı ve $\sigma^2$ varyanslı Normal dağılıma sahip oldukları ve bağımsız ve özdeşçe dağılmış oldukları varsayılır. Bu varsayım $\varepsilon^{(i)} \sim N(0, \sigma^2)$ ile gösterilir (Ng, 2000). Yani $\varepsilon^{(i)}$ lerin yoğunluğu

$p\left(\varepsilon^{(i)}\right) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}\right)$ ile verilir.

Buradan

$$p\left(y^{(i)}\middle|x^{(i)};\theta\right) = \frac{1}{\sqrt{2\pi}\sigma}exp\left(-\frac{\left(y^{(i)}-\theta^T x^{(i)}\right)^2}{2\sigma^2}\right) \qquad (3.8)$$

eşitliği elde edilebilir. $p\left(y^{(i)}\middle|x^{(i)};\theta\right)$ gösterimi $x^{(i)}$ verildiğinde $y^{(i)}$ nin dağılımı $\theta$ parametresine bağlıdır anlamına gelmektedir. $\theta$ bir rasgele değişken olmadığından bir koşul belirtmez. $p\left(y^{(i)}\middle|x^{(i)};\theta\right)$ büyüklüğü $x$ ve $y$ ye bağlı bir fonksiyondur. Bu

fonksiyonu $\theta$ nın bir fonksiyonu olarak ifade etmek için $\varepsilon^{(i)}$ lerin bağımsız ve özdeş oldukları varsayımı da kullanılarak;

$$L(\theta) = \prod_{i=1}^{m} p\left(y^{(i)} \middle| x^{(i)}; \theta\right)$$
$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma}} exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right) \quad (3.9)$$

şeklinde $L$ fonksiyonu tanımlanabilir. Bu fonksiyon olabilirlik fonksiyonu olarak isimlendirilir. $L(\theta)$ yı maksimum yapan $\theta$ değerinin bulunması maksimum olabilirlik metodu olarak bilinmektedir. $L(\theta)$ yı maksimum yapan $\theta$ değerinin yerine $L(\theta)$ nın sürekli artan bir fonksiyonun maksimum değeri de bulunabilir. Bunun için log olabilirlik fonksiyonu;

$$l(\theta) = logL(\theta)$$
$$= log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma}} exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right)$$
$$= \sum_{i=1}^{m} log \frac{1}{\sqrt{2\pi\sigma}} exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right)$$
$$= mlog \frac{1}{\sqrt{2\pi\sigma}} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} \left(y^{(i)} - \theta^T x^{(i)}\right)^2 \quad (3.10)$$

şeklinde tanımlanır. Yani $l(\theta)$ yı maksimum yapan değer $\sum_{i=1}^{m} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$ değerini minimum yapacaktır (Ng, 2000).

Bir problemin çözümüne makine öğrenmesi ile ulaşılmaya çalışılırken belli adımların uygulanması gerekmektedir. Literatürde izlenmesi gereken adımlar için bir çok standart belirlenmesine rağmen bu aşamalar genel olarak birbirine oldukça benzemektedir.

Veri madenciliği için birçok uzmanın bir araya gelip bu adımları standartlaştırmaya çalıştığı Veri Madenciliği için Çarpraz Endüstri Standard Süreç Modeli (Cross-Industry Standard Process for Data Mining - CRISP) makine öğrenmesi algoritmaları için de kullanılabilir. (Shearer, 2000)

Bu modelde problemin çözümüne ulaşmak için gerekli adımlar

1. Problemin tanımlanması
2. Veriyi anlama
3. Veri ön işleme
4. Modelleme
5. Model Değerlendirmesi ve Seçimi
6. Uygulamaya Geçilmesi

olarak belirlenmiştir (Kartal, 2015; Brownlee, 2014).

#### 3.1.2.1. Problemin tanımlanması

Problem tanımlanması aşamasında makine öğrenmesi algoritmaları yardımıyla çözülmesi amaçlanan problemin açıkça ifade edilebilmesi gerekir. Problem Mitchell tarafından verilen tanıma tam olarak uydurulmalıdır.

#### 3.1.2.2. Veriyi anlama

Makinenin öğrenmek için kullanacağı veri, ele alınan probleme uygun bir biçimde oluşturulmalıdır. Toplanan veri düzenli (veritabanı formatı gibi), ya da düzensiz (Twitter'dan elde edilen twitler gibi) olabilir. Veri toplama aşamasında farklı kaynaklardan yararlanılabilmektedir. Bunlardan ilki internette yer alan hazır veri setleridir. Bu veri setleri, erişim ve kullanım kolaylığı bakımından makine öğrenmesi çalışmalarına zaman kazanma açısından önemli avantaj sağlamaktadır. Nitekim verinin analizler için hazırlanması belki de makine öğrenmesi sürecinin en zor ve zaman alıcı aşaması olarak kabul edilebilir. Bazı çalışmalarda da gerekli olan veri seti araştırmacılar tarafından internetten özel bazı araçlarla elde edilirken, kimi çalışmalarda da internetteki hazır veri setlerinden birini kullanmak yerine, ele alınan probleme uygun özgün veri setleri problem için özellikle oluşturulmaktadır. Veriye uygun modelin kurulmasından önce mevcut verinin iyi anlaşılması ve iyi analiz edilmesi gerekmektedir. Veri seti elde edildikten sonra veri hakkında ön fikir edinilmesi için

verinin normalleştirilmesi gibi bazı basit istatistiksel metotlardan yararlanılabilir (Kartal, 2015; Maurizio, 2011).

#### 3.1.2.3. Veri ön işleme

Veri analizine geçmeden önce verinin analizler için hazır hale getirilmesi gerekmektedir. Nitekim pratik hayatta veriler çoğu zaman eksik, tutarsız veya hatalı olabilmektedir. Literatürde veri ön-işleme (data preprocessing) olarak adlandırılan veri hazırlama süreci Kartal (2015) ve Han ve Kamber (2006) tarafından veri temizleme, veri bütünleştirme, veri seçme, veri dönüştürme, veri madenciliği, desen değerlendirmesi ve bilgi sunumu başlıkları altında ele almıştır. Veri hazırlama sürecinde gerçekleştirilecek işlemlerin herhangi bir standardı olmayıp, kullanılan veri setine göre değişmektedir (Kartal, 2015; Maurizio, 2011).

#### 3.1.2.4. Modelleme

Bir makine öğrenmesi algoritmasında model, girdiyi çıktıya dönüştüren matematiksel bir bağıntı veya veri içerisinde saklı bulunan örüntüdür (Flach, 2012). Makine öğrenmesi algoritması, oluşturulabilecek modeller arasından en iyisini seçmeye çalışacaktır.

Bu tez kapsamında ele alınacak modelleri belirlemek için kullanılacalak algoritmalar sırasıyla Lojistik Regresyon, *k-en* yakın komşu, Naive Bayes algoritmalarıdır.

##### 3.1.2.4.1. Lojistik regresyon

Sınıflandırma problemlerine regresyonda olduğu gibi yaklaşılabilir ancak burada farklı olarak etiket verileri kesiklidir. Etiket verisinin sadece iki değer aldığı sınıflandırma problemlerine ikili sınıflandırma problemi denilmektedir. Örneğin e-postaların sınıflandırıldığı bir problem göz önüne alındığında; spam olanlar 0 spam olmayanlar 1 olarak etiketlenebilir. 0 negatif sınıf ve 1 pozitif sınıf olarak isimlendirilir.

Bir sınıflandırma problemine y'lerin kesikli oldukları göz ardı edilerek $x$ verildiğinde y 'yi tahmin etmek için regresyon kullanılabilirdi. Fakat sezgisel olarak 1 den büyük veya 0 dan küçük değerler alan bir hipotez kümesini kullanmak mantıklı olmamaktadır. Bunun üstesinden gelmek için hipotez kümesi olarak;

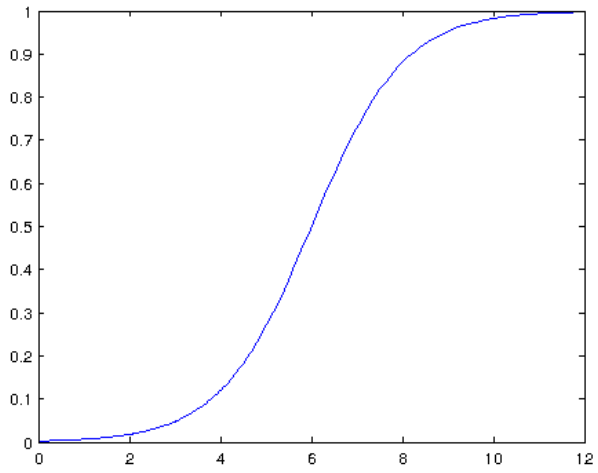
$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \tag{3.11}$$

seçilebilir.

Burada;

$$g(z) = \frac{1}{1+e^{-z}} \tag{3.12}$$

fonksiyonuna lojistik fonksiyon ya da sigmoid fonksiyonu denir (Şekil 3.2).




Şekil 3.2. Lojistik regresyon grafiği.

Dikkat edilirse $z \longrightarrow \infty$ iken $g(z)$ 1 e ve $z \longrightarrow -\infty$ iken $g(z)$ -1 e yaklaşmaktadır. $g(z)$ Yani $h(x)$ hipotez kümemiz 0 ve 1 ile sınırlıdır. 0 ile 1 arasında başka fonksiyonlarda kullanılabilir ancak sigmoid fonksiyonunun sahip olduğu matematiksel özellikler sebebiyle kullanımı oldukça yaygındır (Ng, 2000).

$$g'(z) = \frac{d}{dz}\frac{1}{1+e^{-z}}$$
$$= \frac{1}{(1+e^{-z})^2}(e^{-z})$$
$$= \frac{1}{1+e^{-z}}\left(1-\frac{1}{1+e^{-z}}\right) \quad (3.13)$$
$$= g(z)(1-g(z))$$

.

Bir öğrenme problemi için hipotez kümesi;

$$P(y=1|x;\theta) = h_\theta(x)$$
$$P(y=0|x;\theta) = 1-h_\theta(x)$$

olarak belirlensin. Bu daha kompakt bir halde,

$$p(y|x;\theta) = \left(h_\theta(x)\right)^y\left(1-h_\theta(x)\right)^{1-y}$$

şeklinde yazılabilir. $m$ eğitim örneğinin birbirinden bağımsız olduğu varsayımıyla olabilirlik fonksiyonu;

$$L(\theta) = p(\vec{y}|X;\theta)$$
$$= \prod_{i=1}^{m} p\left(y^{(i)}\middle|x^{(i)};\theta\right)$$
$$= \prod_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)\right)^{y^{(i)}}\left(1-h_\theta\left(x^{(i)}\right)\right)^{1-y^{(i)}} \quad (3.14)$$

olarak yazılabilir.

log olabilirlik fonksiyonu;

$$l(\theta) = logL(\theta)$$
$$= \sum_{i=1}^{m} y^{(i)} logh(x^{(i)}) + (1-y^{(i)})\log(1-h(x^{(i)}))$$

En dik iniş algoritması kullanılarak;

$$\frac{\partial}{\partial\theta_j}l(\theta) = \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x)$$
$$= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{g(\theta^T x)}\right)g(\theta^T x)(1-g(\theta^T x))\frac{\partial}{\partial\theta_j}\theta^T x$$
$$= \left(y\left(1-g(\theta^T x)\right) - (1-y)g(\theta^T x)\right)x_j$$
$$= (y - h_\theta(x))x_j$$
$$g'(z) = g(z)(1-g(z))$$

olduğu kullanılarak;

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x_j{}^{(i)} \tag{3.15}$$

bulunur.

### 3.1.2.4.2. Naive Bayes sınıflandırıcı

Naive Bayes Sınıflandırıcı, güçlü bağımsızlık varsayımıyla Bayes teoreminin uygulanmasına dayanan basit olasılıksal bir sınıflandırıcıdır. Daha basit bir ifadeyle Naive Bayes Sınıflandırıcı bir sınıfın belli bir niteliğinin varlığı başka bir niteliğin varlığı ya da yokluğu ile ilişkili olmadığını varsayar. Örneğin bir meyve eğer kırmızı, yuvarlak ve 3 cm yarıçapında ise elma olarak değerlendirilebilir. Bu nitelikler birbirleri ile ilişkili olsa bile Naive Bayes sınıflandırıcı tüm bu niteliklerin bu meyvenin elma olma olasılığına bağımsız katkı yaptığını göz önünde bulundurur.

Daha soyut bir ifade ile bir sınıflandırıcı için olasılıksal model bir koşullu olasılık modelidir (Ng, 2000).

$$P(C|F_1, \dots, F_n) \tag{3.16}$$

Burada C, $F_1, \dots, F_n$ niteliklerine bağlı sınıfları veya çıktıları belirtmektedir. Bayes teoremi yardımıyla bu olasılık;

$$P(C|F_1, \dots, F_n) = \frac{P(C)p(F_1,\dots,F_n|C)}{p(F_1,\dots,F_n)} \quad (3.17)$$

şeklinde yazılabilir. Pratikte bu formülün sadece paydası ile ilgilenilir. Çünkü payda C ye ve F niteliklerinin değerlerine bağlı değildir.

$$p(C, F_1, \dots, F_n) = p(C)p(F_1, \dots, F_n|C)$$
$$= p(C)p(F_1|C)p(F_2, \dots, F_n|C, F_1)$$
$$= p(C)p(F_1|C)p(F_2|C, F_1)p(F_3, \dots, F_n|C, F_1, F_2)$$
$$= p(C)p(F_1|C)p(F_2|C, F_1) \dots p(F_n|C, F_1, F_2, F_3, \dots, F_{n-1})$$

Her bir $F_i$ niteliğinin $i \neq j$ için $F_j$ niteliğinden bağımsız olduğu varsayıldığında $i \neq j$ için;

$p(F_i|C, F_j) = p(F_i|C)$ dir.

Dolayısıyla

$$p(C, F_1, \dots, F_n) = p(C)(F_1|C)p(F_2|C) \dots P(Fn|C)$$
$$= p(C) \prod_{i=1}^{n} p(F_i|C)$$

olarak yazılabilir.

Bağımsızlık koşulu altında $C$ sınıf değişkeni üzerindeki koşullu olasılık dağılımı

$$p(C|F_1, F_2, \dots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(F_i|C) \quad (3.18)$$

şeklinde yazılabilir (Ng, 2000).

### 3.1.2.4.3. k- en yakın komşu algoritması

*k*- en yakın komşu algoritması 1951 yılında Fix & Hodges tarafından bir örüntü tanımlama probleminin çözümü için önerilmiştir (Fix ve Hodges, 1951). k en yakın

komşu algoritması örüntü tanımlaması için çözüm olarak önerilmesine rağmen bir çok sınıflandırma probleminde de başarılı bir şekilde kullanılmıştır.

*k*-en yakın komşu sınıflandırıcısı etiketi tahmin edilmek istenen örneğin sınıfını o örneğe en yakın *k* adet örneğin sınıfları arasında en çok tekrar eden sınıfı kullanarak belirlemeye çalışır. Yakınlık ölçüsü olarak kullanılan metrikler arasında en çok kullanılanı Öklid metriğidir.

$(X_i, C_i) \quad i = 1,2, \dots, n$ örnekleri göz önünde bulundurulsun. Burada her bir $i$ için $X_i$ nitelikleri $C_i$ sınıfları göstermektedir. Sınıf sayısı $c$ olarak ele alınsın. Her bir $i$ için $c_i \in \{1,2,3, \dots, c\}$

$x$ noktası sınıfı bilinmeyen ve *k*-en yakın komşu algoritması kullanılarak sınıfı tahmin edilmeye çalışılan bir örnek olsun. Bu durumda *k*-en yakın komşu algoritmasının adımları şu şekildedir

Her bir $i$ için $d(x, x_i)$ uzaklıkları hesaplanır. Burada $d$ metriği Öklid uzaklığı olarak alınmıştır.

Hesaplanan bu $n$ tane değer azalmayan bir şekilde sıralanır;
*k* bir tamsayı değeri olmak üzere, sıralanan bu değerlerden ilk k tanesi seçilir,
bu k tane değere karşılık gelen örneklerin dahil oldukları sınıflar belirlenir,
belirlenen sınıflar arasında en çok tekrar eden sınıf $x$ örneğinin sınıfı olarak tahmin edilir (Ng, 2000).

### 3.1.3. Model performansı değerlendirme ölçütleri

Makine öğrenmesi algoritmalarının performansları değerlendirilirken literatürde çeşitli ölçütler belirlenmiştir. Bu ölçütler regresyon ve sınıflandırma problemlerinde farklılık göstermektedir. Regresyon problemlerinde ortalama kare hata, ortalama mutlak hata vb. model değerlendirme ölçüleri kullanılırken, sınıflandırma problemlerinde kullanılan ölçüler kontenjans tablosundan elde edilen doğruluk, hata oranı, duyarlılık, belirleyicilik, yanlış pozitif oranı, yanlış negatif oranı, kesinlik ve F-ölçüsüdür (Japkowicz, 2011). Bu çalışma kapsamında sınıflandırma algoritmalarının performans ölçümleri için Doğruluk, Duyarlılık, Kesinlik ve F-ölçüsü kullanılmıştır.

Çizelge 3.3. Kontenjans çizelgesi

| | | **Gerçek** | | |
|---|---|---|---|---|
| | | **Pozitif** | **Negatif** | **Toplam** |
| **Tahmin** | **Pozitif** | Doğru Pozitif(dp) | Yanlış Pozitif(yp) | tPoz |
| | **Negatif** | Yanlış negatif(yn) | Doğru Negatif(dn) | tNeg |
| | **Toplam** | Pozitif | Negatif | m |

Tabloya göre

$$Doğruluk(ACC) = \frac{dp + dn}{m}$$

$$Duyarlılık\ (Recall) = \frac{dp}{pozitif} = \frac{dp}{dp + yn}$$

$$Kesinlik = \frac{dp}{tPoz}$$

$$F - \text{ölçüsü} = \frac{2 * Kesinlik * Duyarlılık}{Kesinlik + Duyarlılık} \quad (3.19)$$

# 4. BULGULAR

## 4.1. Phishing Web Sayfalarının Özellikleri

Günümüzde Makine Öğrenmesi tekniklerini kullanarak Phishing web sitelerini öngörme konusunda birçok makale yayımlanmış olmasına rağmen, güvenilir bir eğitim veri seti yayınlanmamıştır. Phishing web sitelerini karakterize eden kesin özellikler konusunda literatürde kesin bir anlaşma yapılmadığı için tüm olası özellikleri kapsamaya çalışan bir veri kümesi bu tez kapsamında şekillendirmeye çalışılmıştır.

Mohammad ve ark. (2015), Phishing Websites Features adlı çalışmalarında bir web sayfasının belirlenmesinde adres tabanlı anormal davranışlar, HTML ve JavaScript tabanlı ve Domain tabanlı olmak üzere 4 ana başlıkta topladıkları 30 tane niteliği belirlemişlerdir. Lee ve ark. (2015), çalışmalarında Mohammad ve Ark.(2015), çalışmasından farklı olarak *Similarity of primary domain and Google Suggestion and Google Suggestion, Similarity of subdomain and Google Suggestion, Similarity of path domain and Google Suggestion, Safety of Google Suggestion for domain, subdomain and path domain, Google Page Rank, Number of TLD and out of TLD position, Phishing words in URL, primary domain spelling mistake, country matching, Value of TTL, PTR Record* niteliklerini kullanarak phishing websitelerini makine öğrenmesi algoritmaları ile belirlemeye çalışmışlardır.

Phishing amaçlı web sitelerini öngörmede etkili olduğu kanıtlanan, önemli niteliklere ışık tutmak amacıyla farklı nitelikler göz önünde bulundurulmuştur. Buna ek olarak önerilen yeni özelliklerle birlikte, bazı tanınmış özelliklere yeni kurallar atanması ve güncellenmesi bu çalışmada sağlanmıştır.

## 4.2. Çalışmada Kullanılan Veri Setinin PHP Dili ile Analize Hazırlanması

Bu çalışma kapsamında kullanılan veri setinin toplanmasında PD-F (Phishing Detection Fırat ) adını verdiğimiz, PHP programlama dili ile yazılan bir uygulamadan yararlanılmıştır.

PHP (Hypertext Preprocessor) geniş bir kitle tarafından kullanılan, özellikle web tabanlı uygulamalarını geliştirmek için tasarlanmış, web sayfalarında dinamik içerik sağlamak ve kullanıcıyla iletişim kurmak amacıyla kullanılan, sunucu tarafında çalışan ve HTML içine gömülebilen bir betik dilidir.

Çalışma kapsamında oluşturulan veri seti PhishTank'den alınan 6864 tane phishing sitesi, belli anahtar kelimeleriyle yapılan Google arama sonuçlarından elde edilen 12635 geçerli web site URL'leri alınarak elde edilmiştir. Phishing websayfalarının kısa ömürlü olması nedeniyle Google arama motoru tarafından indekslenmemektedirler. Dolayısıyla Phishing web sayfalarının Google arama sonuçlarında ilk sıralarda yer almamaması beklenmektedir. URL adreslerinden elde edilen ve özellikle URL tabanlı niteliklerin toplanmasında PHP dilinde bulunan string ve Regular Rxpression( Düzenli İfadeler) fonksiyonlarından yararlanılmıştır. Örneğin URL içerisinde bir IP adresinin tespit edilmesinde,

*if( preg_match( '/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/', $urlListesi[ $f ] ) ) $parametreler[] = 1; else $parametreler[] = -1;* şeklinde regular expression yapısı kullanılmıştır.

URL'ye ait sitenin içeriği ile ilgili niteliklerin( dış URL sayısı, Favicon vb.) toplanmasında ise CURL(Client URL) kütüphanesinden yararlanılmıştır.

CURL bir web adresinin linkine tıklamadan kodlama yolu ile ilgili web adresinin içeriğini indirmeye yarayan bir kütüphanedir. CURL kütüphanesi kullanılarak ilgili URL'ye ait sitenin HTML içeriği indirilmiş ve indirilen HTML içeriğinden niteliklerin elde edilmesi için xpath gibi yapılardan yararlanılmıştır. XPath, XML etiketleri içerisinde yer alan verilere ulaşmak için geliştirilen bir standarttır. Pavicon niteliğini elde etmek için kullanılan xpath kullanınımına bir örnek aşağıda verilmiştir.

```
$url = $siteninURLAdresi;
$doc = new DOMDocument();
$doc->loadHTML( $icerik );
$xml = simplexml_import_dom( $doc );
$arr = $xml->xpath( '//link[@rel="shortcut icon"]' );
```

Bu şekilde favicon özelliğinini kullanılıp kullanılmadığı kontrol edilmiş ve sitenin URL adresi ile favicon'un yüklendiği adres aynı ise 1(bir) değeri üretilmiştir. Aksi durumda ise -1(eksi bir) değeri elde edilmiştir.

URL adrsine ait doamin yaşı, DNS kaydı gibi niteliklerin elde edilmesinde PHP dilininde kullanılan fsockopen gibi socket fonksiyonlarından yararlanılmıştır. Socket, iki farklı bilgisayarda çalışan uygulamanın birbiri ile belirli bir port üzerinden haberleşmesi (veri alışverişi yapmak) için tasarlanmış bir protokoldür. Günümüzde kullanılan gerçek zamanlı yazışma ve konuşma uygulamaları (WhatsApp vb.) bu protokol çerçevesinde geliştirilmektedir. Socket protokolü ile uygulamaların izin verdiği ölçüde iki bilgisayar arasında veri alışverişi yapılabilmektedir. Bu çalışmada aşağıdaki WHOIS sunucularından incelenen URL adresindeki alan adının yaşı socket protokolü kullanılarak öğrenilmektedir.

*whois.verisign-grs.com*
*whois.verisign-grs.com*
*whois.pir.org*
*whois.afilias.info*
*whois.neulevel.biz*
*whois.nic.us*
*whois.nic.uk*
*whois.cira.ca*
*whois.nic.tel*
*whois.iedr.ie*
*whois.nic.it*
*whois.nic.cc*
*whois.nic.ws*
*whois2.afilias-grs.net*
*whois.dotmobiregistry.net*
*whois.registrypro.pro*
*whois.educause.net*
*whois.nic.tv*
*whois.nic.travel*
*whois.inregistry.net*

*whois.nic.me*

*whois.cnnic.cn*

*whois.nic.asia*

*whois.rotld.ro*

*whois.aero*

*whois.nic.nu*

Yukarıdaki sunuculardan uygulamaya gelen Creation Date:(.*) niteliğindeki oluşturulma tarihi ve niteliklerin toplanması esnasındaki tarihler karşılaştırılarak domainin yaşı hesaplanmıştır.

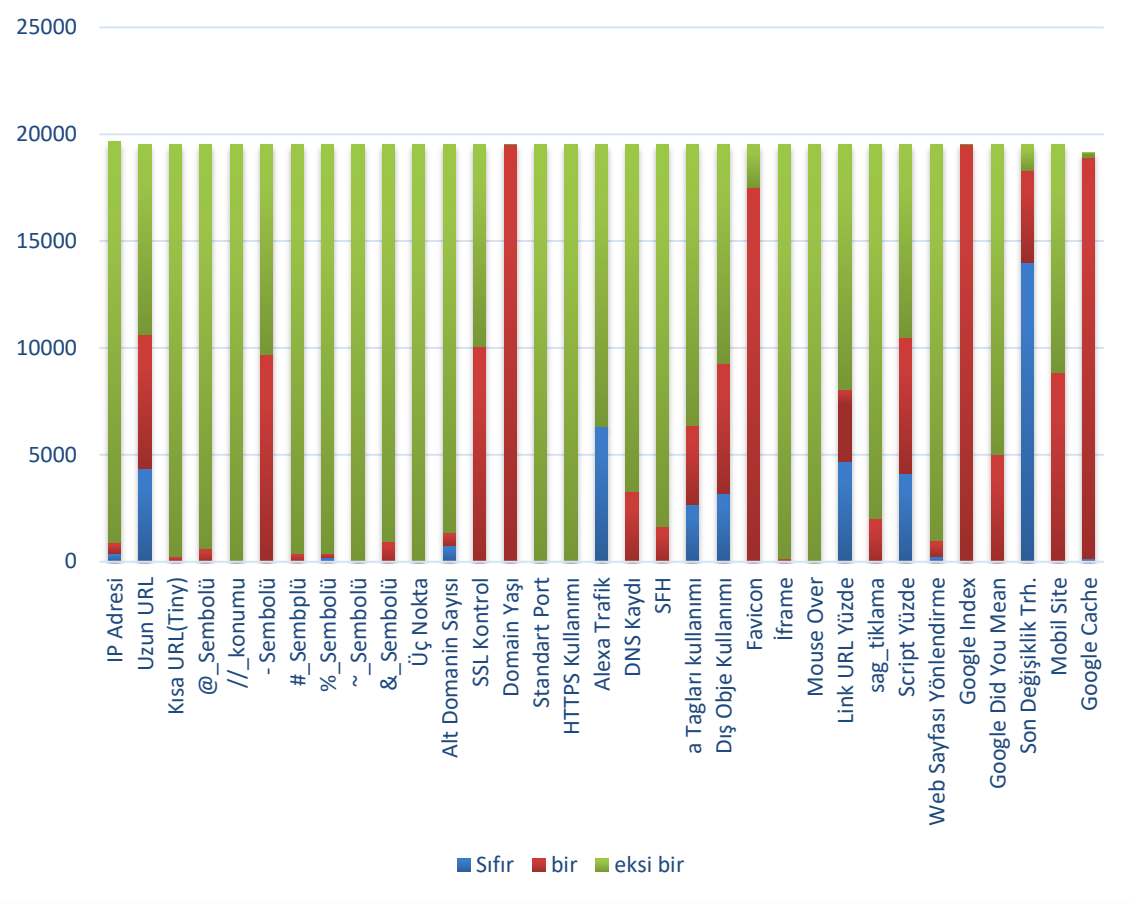
URL adresine ait web sayfasının trafik bilgileri ise, bu bilgiyi XML formatında sunan ALEXA gibi web sayfalarından PHP'de kullanılan XML okuma fonksiyonlarından yararlanarak elde edilmiştir. Aşağıda ALEXA sitesinin sunduğu XML dosyasını okuma örneği verilmiştir.

```
$xml    = simplexml_load_file(
'http://data.alexa.com/data?cli=10&dat=snbamz&url=' . $url );
$rank   = isset( $xml->SD[ 1 ]->POPULARITY ) ? $xml->SD[ 1 ]-
>POPULARITY->attributes()->TEXT:0;
```

Yukarıda anlatılan teknikler ile elde edilen veri seti (Şekil 4.1) 'de gösterildiği gibi vektörlere dönüştürülmüştür.

```
-1,-1,-1,-1,-1,1,-1,-1,-1,-1,1,-1,-1,1,-1,-1,0,-1,-1,-1,-1,1,-1,-1,1,-1,-1,1,-1,1,1,0,1,1
-1,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,0,-1,-1,-1,-1,1,-1,-1,1,-1,-1,1,-1,1,-1,0,1,1
-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,0,1,-1
-1,0,-1,-1,-1,1,-1,-1,-1,-1,1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,1,-1,-1,1,-1,1,1,0,1,1
-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,1,-1,-1,1,-1,1,1,0,1,1
-1,0,-1,-1,-1,1,-1,-1,-1,-1,1,0,-1,1,-1,-1,0,-1,-1,-1,-1,1,-1,-1,1,-1,-1,1,-1,1,-1,0,1,1
-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,0,-1,-1,0,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,0,-1,-1
-1,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,1,1,-1,-1,0,-1,-1,-1,-1,1,-1,-1,1,-1,-1,1,-1,1,1,0,1,1
-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1,1,-1,-1,-1,1,-1,-1,-1,1,-1,-1,1,-1,-1,1,-1,1,1,0,1,1
-1,1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,0,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,1,-1,-1
-1,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,1,1,-1,1,-1,-1,-1,-1,-1,1,1,1,-1,-1,-1,-1,-1,1,-1,1,-1,1,1,1
-1,0,-1,-1,-1,1,-1,-1,-1,-1,1,0,1,1,-1,-1,0,-1,-1,-1,0,1,-1,-1,-1,-1,-1,0,-1,1,-1,0,-1,1
-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,1,-1,-1,1,-1,1,-1,0,1,1
-1,1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,-1,0,-1,-1
-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,0,-1,-1,-1,-1,1,-1,-1,1,-1,-1,1,-1,1,1,0,1,1
```

Şekil 4.1. Örnek veri seti.




Şekil 4.2. Veri seti nitelikleri için gözlem sayıları.

### 4.3. Verilerin Analizinde Kullanılan Nitelikler

Bu kısımda çalışmada kullanılan niteliklerin neler olduğu tanıtılmıştır. Çalışmada kullanılan nitelikler 4 ana başlıkta toplanmıştır;

1. Adres Tabanlı Nitelikler
2. Anormal URL Tabanlı Nitelikler
3. HTML ve Javascript Tabanlı Nitelikler
4. Domain Tabanlı Nitelikler

Nitelikleri incelerken kullanılan “Phishing”, “Şüpheli” ve “Güvenilir” kavramları yerine sırası ile 1,0 ve -1 kullanılmıştır.

### 4.3.1. Adres tabanlı nitelikler

#### 4.3.1.1. IP adresi kullanımı

URL adresinde alan adının bir alternatifi olarak bir IP adresinin kullanılması durumudur.
Örnek URL: "http://125.98.3.124/ornek.html"

*Kural* : *Eğer alan adı IP adresi içeriyorsa => 1*
*Diğer durumda => -1*

#### 4.3.1.2. Uzun URL adresi kullanımı

Normalden uzun URL adreslerinin okunması zor olması nedeniyle Phishing saldırganları uzun URL kullanmayı tercih edilmektedir.
Örnek URL:
http://www.perseus.tufts.edu/cgi-bin/ptext?doc=Perseus:text:1999.04.0006&query=head=#315

*Kural : Eğer URL Uzunluğu < 54 => -1*
*Değilse ve URL Uzunluğu >= 54 ve <= 75 ise => 0*
*Diğer durumda => 1*

#### 4.3.1.3. Kısa (Tiny) URL kullanımı

TinyURL, URL adreslerinin kısaltılması ve uzun URL adreslerinin yeniden yönlendirilmesi için kısa takma adlar sağlayan bir web hizmetidir.
Örnek URL:
https://tinyurl.com/ydfgj2vj

*Kural : Eğer URL TinyURL ise => 1*
*Diğer Durumda = > -1*

#### 4.3.1.4. URL adresi içerisinde “@” sembolünün kullanımı

URL adreslerinde "@" sembolünün kullanılması tarayıcının "@" sembolünün önündeki her şeyi yok saymasına ve gerçek adres genellikle "@" sembolünü takip etmesine neden olmaktadır.

Örnek URL:

http://www.rambalbuilders.in/admin/status/login/index.php?email=abuse@...

*Kural : Eğer URL adresi @ sombolü içeriyorsa => 1*

*Diğer durumda => -1*

#### 4.3.1.5. URL adresi içerisinde “//” sembolünün kullanılması

URL adresi içinde "//" varlığı, kullanıcının başka bir web sitesine yönlendirileceği anlamına gelir. Bu tür URL adreslerinin bir örneği:

*http://70-40-204-111.unifiedlayer.com//sites/all/themes/zen/translatio…*

"//" ifadesinin bulunduğu yer incelenmekte ve URL'nin "HTTP" ile başlaması durumunda "//" nin altıncı sırada olması gerektiği anlamına gelmektedir. Bununla birlikte, URL "HTTPS" kullanıyorsa "//" yedinci sırada görünmelidir.

Örnek URL:

http://www.terciarioscapuchinosecuador.org/media/k2/1//index-dados.php...

*Kural : Eğer URL içindeki “//” ifadesinin konumı > 7 => 1*

*Diğer Durumda = > -1*

#### 4.3.1.6. URL adresi İçerisinde “-” sembolünün kullanılması

“-” simgesi, yasal URL adreslerinde nadiren kullanılmaktadır. Phishing saldırganları, alan adlarına “-” ile ayrılmış ön ekler veya son ekler ekleme eğilimindedir.

Örnek URL :

*http://info-recovery.hol.es/facebook*

*Kural : Eğer alan adı "-" ifadesinin içeriyorsa => 1*

*Diğer Durumda = > -1*

#### 4.3.1.7. URL adresi içerisinde "%" sembolünün kullanılması

"%" simgesi, URL' lerdeki özel karakterleri belirtmek için kullanılan escape(kaçış) kodlarını tanımlamaktadır. Phishing saldırganları, alan adlarına "%" simgesini ekleyerek url adreslerini okunması zor duruma getirmeyi amaçlamaktadır.
Örnek URL:
http://octopus.ua/fml/dropbox%20sign/dropbox/view/...

*Kural : Eğer alan adı "%" ifadesinin içeriyorsa => 1*

*Diğer Durumda = > -1*

#### 4.3.1.8. URL adresi içerisinde "#" sembolünün kullanılması

"#" simgesi, sayfanın herhangi bir noktasına odaklanmak için kullanılır. Phishing saldırganları, bu sembolü kullanarak normal bir URL görünümü vermeyi amaçlamaktadır. Öte yandan phishing sayfalarının istedikleri konumun kullanıcının önüne gelmesi sağlanmaktadır.
Örnek URL:
http://ip6.si#IzyZb0

*Kural : Eğer alan adı "%" ifadesinin içeriyorsa => 1*

*Diğer Durumda = > -1*

#### 4.3.1.9. URL adresi İçerisinde "~" sembolünün kullanılması

"~" simgesi, kök klasörünü temsil eder. Phishing saldırganları, bu sembolü kullanarak kök klasör yönlendirmesi yapabilmektedirler.
Örnek URL:
http://78.142.63.250/~redcross/ug/domain/?email=abuse@tradinghouse.ca...

*Kural : Eğer alan adı "~" ifadesinin içeriyorsa => 1*

*Diğer Durumda = > -1*

#### 4.3.1.10. URL adresi içerisinde "&" sembolünün kullanılması

"&" simgesi, URL ile birlikte istenilen sayfaya istenilen bilgileri parametre olarak göndermek için kullanılan bir parametre ayıracıdır. Phishing saldırganları, bu sembolü kullanarak normal bir URL görünümü vermeyi amaçlamaktadır. Ancak "&" sembolü alan adı kısmında değil URL' nin parametre kısmında kullanılmaktadır.
Örnek URL:
https://docs.google.com/uc?id=0BxGrJcAMovW4RXdoc2dvQjhQMG8&export=...

*Kural : Eğer alan adı "&" ifadesinin içeriyorsa => 1*

*Diğer Durumda = > -1*

#### 4.3.1.11. HTTPS kullanımı

HTTPS varlığı, web sitese güvenilir izlenimi vermek açısından çok önemlidir, ancak bir web sitesinin güvenilir olması için sadece bu kriter yeterli değildir. HTTPS varlığı ile birlikte sertifikanın asgari yaşının en az iki yıl olması gerekmektedir.

*Kural : Eğer HTTPS varsa => -1*

*Diğer durumda => 1*

#### 4.3.1.12. Alt alan adı ve çoklu alt alan adı kullanımı

Bir alan adı, örneğimizde "tr" olan ülke kodu (ccTLD) içerebilir. "edu" kısmı "education" kısaltmasıdır, birleşik "edu.tr" ikinci düzey etki alanı (SLD), "yyu" ise etki alanının gerçek adıdır. Bu özelliğin çıkartılması için öncelikle (www.) kısmının URL'den çıkarılması gerekir. Daha sonra, varsa (ccTLD) kaldırılır. Son olarak, kalan noktalar sayılır. Nokta sayısı birden fazla ise, URL, bir alt alana sahip olduğu için "Şüpheli" olarak sınıflandırılmaktadır. Bununla birlikte, noktalar ikiden fazla ise, birden fazla alt alana sahip olacağından "Phishing" olarak sınıflandırılmaktadır. Aksi takdirde

"Güvenilir" olarak belirlenmekte. Bu tür URL adreslerinin bir örneği: https://www.yyu.edu.tr/AkademikBirimler/index.php?s=edebiyat

*Kural : Eğer alan adı içerisindeki nokta sayısı = 1 ise => -1*

*Değilse ve alan adı içerisindeki nokta sayısı = 2 => 0*

*Diğer durumda => 1*

#### 4.3.1.13. Favicon kullanımı

Favicon, belirli bir web sayfasıyla ilişkili bir grafik resimdir (simge). Bir çok kullanıcı favicon'u adres çubuğundaki web sitesi kimliğinin görsel hatırlatıcısı olarak göstermektedir. Favicon adres çubuğunda gösterilenden farklı bir adresten yüklendiyse, web sayfası "Phishing" olarak nitelendirilmektedir.

Kural *: Eğer favicon farklı bir URL'den yükleniyorsa => 1*

*Diğer durumda => -1*

#### 4.3.1.14. Standart port kullanımı

Bu özellik, belirli bir sunucudaki belirli bir hizmetin (ör. HTTP) etkin olup olmadığını doğrulamak için kullanışlıdır. Saldırıları kontrol altında tutmak için, yalnızca ihtiyaç duyulan portların açılması tavsiye edilmektedir. Sistemdeki tüm portların açık bırakılması ve gerekli olmayan portların açılması ciddi saldırılara maruz kalmaya neden olmaktadır. Açık port açık kapı anlamına gelemekte ve saldırganların bir sisteme sızmak için kullanılan yaygın açıklardan bir tanesi olabilmektedir. Varsayılan olarak, birçok güvenlik duvarı, Proxy ve Ağ Adresi Çevirisi (NAT) sunucuları, bağlantı noktalarının tümünü veya çoğunu engellemekte ve yalnızca seçilenleri açmaktadır. Tüm bağlantı noktaları açıksa, phishing saldırganları neredeyse istediği herhangi bir hizmeti çalıştırabilmekte ve bu nedenle kullanıcı bilgileri tehdit altında olabilmektedir. Bu çalışmada HTTP ve HTTPS portları kontrol edilmiştir.

*Kural : Eğer kullanılan # portu önerilen port ise => -1*

*Diğer durumda => 1*

#### 4.3.1.15. Domain içinde HTTPS ifadesinin kullanımı

Phishing saldırganları, kullanıcıları kandırmak için "HTTPS" belirtecini bir URL'nin etki alanına ekleyebilir.
Örnek URL : http://vhcirurgiaplastica.com.br/https-applaypal.com-redirect-to-limit…

*Kural : Eğer domain içinde HTTPS kullanılmışsa => 1*
*Diğer durumda => -1*

### 4.3.2. Anormal URL tabanlı nitelikler

#### 4.3.2.1. Resim, video ve ses dosyaları gibi materyallerin adreslerinin kullanımı

Resim, video ve ses gibi harici nesnelerin başka bir alan adından yüklenip yüklenmediğini inceler. Güvenilir web sayfalarında, web sayfası adresi ve web sayfasında gömülü nesnelerin çoğu aynı alan adını taşır.

*Kural : Eğer nesnelerin alan adları < 22% ise => 1*
*Değilse ve nesnelerin alan adları >= 22% ve <= 61% ise => 0*
*Diğer durumda => 1*

#### 4.3.2.2. <a> etiketlerinin kullanımı

Bağlantı, <a> etiketi tarafından tanımlanan bir öğedir. Bu özellik, tam olarak "İstek URL'si" olarak değerlendirilir. Web sayfası içerisindeki bağlantıların yüzde olarak ne kadarı web sayfalarının alan adıyla aynı olduğu incelenmiştir.

*Kural : Eğer bağlantıların sayısı < 31% => -1*
*Değilse ve bağlantıların sayısı >=31% ve <= 67% ise => 0*
*Diğer durumda = > 1*

#### 4.3.2.3. <Meta>, <Script> ve <Link> etiketlerinin kullanımı

Bir web sayfası kaynak kodunda kullanılabilecek tüm <Meta>, <Script> ve <Link> etiketlerini göz önüne aldığımızda bu etiketlerin referans gösterdiği URL adreslerinin çoğunluğunun aynı alan adını taşıması gerekmektedir.

*Kural : Eğer* <Meta>,<Script> ve <Link> etiketlerinin Dış URL sayısı < *17% => -1*

*Değilse ve* <Meta>, <Script> ve <Link> etiketlerinin dış URL sayısı *>= 17% ve <= 81 => 0*

*Diğer durumda = > 1*

#### 4.3.2.4. Form nesnesinin referans gösterdiği URL adresi(SFH)

Çoğu web sayfalarında formlar bulunur ve kullanıcılardan çeşitli bilgileri örneğin iletişim bilgilerini almaktadırlar. Bu bilgiler genellikle aynı alan adına sahip bir dosyaya gönderilerek veritabanına kaydedilmektedir. Bu durumda legal bir websayfasındaki form nesnesinin bilgileri yolladığı URL nin içindeki alan adı ile sitenin alan adının aynı olması beklenmektedir.

*Kural : Eğer formdaki bilgiler farklı bir alan adına yollanıyorsa => 1*

*Diğer durumda = > -1*

### 4.3.3. HTML ve javascript tabanlı nitelikler

#### 4.3.3.1. Sayfa yönlendirmelerinin kullanımı

Phishing web sayfalarını güvenilir web sitelerinden ayıran niteliklerden biri de, web sitesinin kaç defa yönlendirildiğidir.

*Kural : Eğer sayfa yönlendirmelerini sayısı <=1 ise => -1*

*Değilse ve sayfa yönlendirmelerini sayısı >= 2 ve < 4 ise => 0*

*Diğer durumda = >1*

#### 4.3.3.2. Durum çubuğunun değişmesi durumunun kullanımı

Web sayfalarında fare ile bir bağlantının üzerine gidildiğinde o bağlantının kullanıcıyı hangi adrese yönlendirdiği görülebilir. Phishing saldırganları, kullanıcılara durum çubuğunda sahte bir URL göstermek için JavaScript'i kullanabilir. Özellikle "onMouseOver" olayı incelemeli ve hangi adrese yönlendirildiği öğrenilmelidir.

*Kural : Eğer onMouseOver sahte bir adresi gösteriyorsa => 1*

*Diğer durumda = >-1*

#### 4.3.3.3. Fare ile sağ tıklama olayının devre dışı bırakılmasının kullanımı

Phishing saldırganları sağ tıklama işlevini devre dışı bırakmak için Java Script kullanır, böylece kullanıcılar web sayfası kaynak kodunu görüntüleyemez ve kaydedemez.

*Kural : Eğer fare sağ tıklama olayı kapalı ise => 1*

*Diğer durumda = >-1*

#### 4.3.3.4. Pop-Up pencerelerinin kullanımı

Kullanıcılardan kişisel bilgilerini bir açılır pencere aracılığıyla göndermelerini isteyen güvenilir bir web sitesi bulmak alışılmadık bir durumdur. Öte yandan, bu özellik bazı güvenilir web sitelerinde kullanılmaktadır ancak Phishing sayfalarının aksine sadece “hoş geldiniz” gibi bir karşılama mesajı için kullanılır. Bu kriterde Pop-up penceresinde kullanıcıdan bilgi alıp almadığı bu çalışmada ele alınmıştır.

*Kural : Eğer Pop-up penceresinde kullanıcıdan bilgi alınıyorsa => 1*

*Diğer durumda = >-1*

#### 4.3.3.5. IFrame kullanımı

IFrame, bir web sayfasının içerisinde ek bir web sayfası görüntülemek için kullanılan bir HTML etiketidir. Phishing saldırganları iframe'in frameBorder özelliğini kapatarak görünmez olabilirler. Bu şekilde web sayfalarını bir anlamda gizleyebilirler.

*Kural : Eğer IFrame kullanılıyorsa => 1*

*Diğer durumda = >-1*

### 4.3.4. Domain tabanlı nitelikler

#### 4.3.4.1. Domain yaşı

Bu özellik WHOIS veritabanından çıkarılabilir. WHOIS domainlere ait bilgilerin tutulduğu ve domain sağlayıcı firmaların barındırdığı bir veri tabanıdır. Çoğu Phishing web sayfasının domain yaşı 6 aydan kısadır.

*Kural : Eğer Domain yaşı(Ay) > 6 => -1*

*Diğer durumda = > 1*

#### 4.3.4.2. DNS kaydı

Phishing web sayfasının belirttiği alan adı WHOIS veritabanında bulunmaz. Bu kriterin kontrol edilmesinde WHOIS veritabanına başvurularak ilgili alan adının DNS kaydının olup olmadığına bakılmıştır.

*Kural : Eğer DNS kaydı varsa => -1*

*Diğer durumda = > 1*

#### 4.3.4.3 Alan adı kayıt zamanı

Bir Phishing web sayfasının kısa bir süre yaşadığı gerçeğine dayanarak, birkaç yıl içinde ödeme yaptıklarını, dolayısıyla alan adının kayıt zamanının yakın tarihli olması gerektiği sonucuna ulaşılmaktadır.

*Kural : Eğer alan adı kayıt zamanı (Yıl) <= 1 => 1*

*Diğer durumda => -1*

#### 4.3.4.4. Alexa trafik rank'ı

Alexa, web siteleri hakkında bilgileri toolbarı yoluyla toplayarak, ziyaretçi sayıları (trafik) üzerinden sıralama yapan ve web sitesi hakkında bir takım bilgiler sunan bir servistir. Trafik rank'ı ise bir web sayfasının ziyaret edilme sıralamasıdır.

Bu özellik ziyaretçilerin sayısını ve ziyaret ettikleri sayfa sayısını belirleyerek web sitesinin popülaritesini ölçer. Bununla birlikte, Phishing web siteleri kısa bir süre canlı kaldıkları için Alexa veritabanı (AIWC, 1996) tarafından tanınmayabilir.

*Kural : Eğer Alexa trafik rank'ı varsa ve < 100,000 => -1*

*Diğer durumda = > 1*

#### 4.3.4.5. Google index

Bu özellik bir web sitesinin Google dizininde olup olmadığını inceler. Bir site Google tarafından dizine eklendiğinde, arama sonuçları üzerinde görüntülenir (Web Yöneticisi kaynakları, 2014). Genellikle, Phishing web sayfalarına yalnızca kısa bir süre erişebilir ve sonuç olarak, birçok Phishing web sayfası Google dizininde bulunmayabilir.

*Kural : Eğer web sayfası google dizinine eklenmişse => -1*

*Diğer durumda = > 1*

#### 4.3.4.6. Google did you mean

Bir Web sayfasını anahtar kelime ile aratarak bulma yönteminde Google arama motoru çoğu kez yazdığımız kelimeyi düzeltme önerisinde bulunmaktadır. Dolayısıyla *facbook* gibi bir kelimeyi aradığımızda bunu mu demek istediniz "facebook" şeklinde bir düzeltme önerisi varsa; phishing domain adreslerinin taklit ettikleri web sayfalarının domain adreslerine genellikle bir karakter farklılık göstererek kullanıldığı

unutulmamlıdır. Dolayısyla *facbook* kelimesinin geçtiği domaninin phishing olma ihtimalini üzerinde durulabilir.

*Kural : Eğer google aranan kelimeyi düzeltmiyorsa => -1*
*Diğer durumda = > 1*

#### 4.3.4.7. Mobil websitesi

Günümüzde mobil cihazarın yaygınlaşmasıyla birlikte web sayfalarının büyük bir çoğunluğunun mobil cihazlara uygun web sayfaları tasarlanmaktadır. Phishing bir web sayfasının kısa ömürlü olması nedeniyle çoğunlukla bir mobil versiyona sahip olmamaktadırlar. Eğer bir web sayfasının mobil versiyonu yoksa güvenilirlik konusunda şüphe edilebilinmektedir.

*Kural : Mobil Websayfası varsa => -1*
*Diğer durumda = > 1*

#### 4.3.4.8. Son değişiklik tarihi

Phishing web sayfaları kullanıcıları aldatmaya yönelik web sayfaları oldukları için genellikle oluşturuldukları andan yaklaşık bir gün sonra yayından kaldırılırlar. Bu bilgiler ışığında bir Phishing web sayfasının ömrünün çok kısa olması beklenmektedir. Bu çalışmada incelenen web sayfalarının HTTP Header bilgilerinden son değişiklik tarihlerine bakılmıştır.

*Kural : Son değiliklik tarihi > 48 saat => -1*
*Diğer durumda = > 1*

#### 4.3.4.9. Google cache

Günümüz legal web sayfalarının Google tarafından indekslendiği ve dolayısıyla bir kopyasını önbelleği(Cache)'ne aldığı bilinmektedir. Dolayısıyla kısa ömürlü olan Phishing Web sayfalarının Google'ın önbelleğinde olma ihtimali çok düşük bir

ihtimaldir. Bu noktadan yola çıkarak Google önbelleğinde bir kopyası bulunmayan web sayfalar'nın Phishing olma ihtimali kuvvet kazanmaktadır.

*Kural : Google cash varsa => -1*

*Diğer durumda = > 1*

### 4.4. Makine öğrenme algotrimaları ile elde edilen sonuçlar

Web sayfalarının URL' lerinden elde edilen veri seti, 4 başlık altında yer alan 33 niteliğe göre analizlenmiştir,(Çizelge 4.1,4.2,4.3 ve 4.4).

Pythonda yazılmış Lojistik Regresyon, Naive Bayes ve kNN makine öğrenmesi algoritmaları kullanılarak analizler gerçekleştirilmiş ve bu algotitmaların performans değerlendirmeleri doğruluk, kesinlik, duyarlılık ve F-ölçüsü kullanılılarak elde edilmiştir.

Sınıflandırma algoritmalarının uygulaması Python Kütüphanesi sci-kit learn ile yapılmıştır. Veri seti sınıflandırma algoritmalarının başarısının ölçümünü yapabilmek amacıyla veri seti, eğitim seti ve veri seti olmak üzere iki parçaya ayrılmıştır. Eğitim seti, veri setinin %66'sından oluşmaktadır. Geri kalan kısım test verisi olarak belirlenmiştir.

Çizelge 4.1. Adres tabanlı nitelikler kullanılarak algoritma performanslarının karşılaştırılması

| Algoritma | Doğruluk | Duyarlılık | Kesinlik | F-Ölçüsü |
|---|---|---|---|---|
| Lojistik Regresyon | %76,83 | 0,81 | 0,84 | 0,82 |
| Naive Bayes | %76,63 | 0,81 | 0,84 | 0,82 |
| kNN | %76,57 | 0,81 | 0,82 | 0,82 |

Çizelge 4.2. Anormal URL tabanlı nitelikler kullanılarak algoritma performanslarının karşılaştırılması

| Algoritma | Doğruluk | Duyarlılık | Kesinlik | F-Ölçüsü |
|---|---|---|---|---|
| Lojistik Regresyon | %83,89 | 0,86 | 0,90 | 0,88 |
| Naive Bayes | %83,19 | 0,85 | 0,89 | 0,87 |
| kNN | %76,83 | 0,84 | 0,95 | 0,89 |

Çizelge 4.3. HTML ve Javascript tabanlı nitelikler kullanılarak algoritma performanslarının karşılaştırılması

| Algoritma | Doğruluk | Duyarlılık | Kesinlik | F-Ölçüsü |
|---|---|---|---|---|
| Lojistik Regresyon | %64,23 | 0,64 | 1,0 | 0,78 |
| Naive Bayes | %64,23 | 0,64 | 1,0 | 0,78 |
| kNN | %64,18 | 0,64 | 1,0 | 0,78 |

Çizelge 4.4. Domain tabanlı nitelikler kullanılarak algoritma performanslarının karşılaştırılması

| Algoritma | Doğruluk | Duyarlılık | Kesinlik | F-Ölçüsü |
|---|---|---|---|---|
| Lojistik Regresyon | %64,13 | 0,64 | 1,0 | 0,78 |
| Naive Bayes | %64,13 | 0,64 | 1,0 | 0,78 |
| kNN | %64,35 | 0,65 | 0,97 | 0,78 |

Çizelge 4.5. Tüm nitelikler kullanılarak algoritma performanslarının karşılaştırılması

| Algoritma | Doğruluk | Duyarlılık | Kesinlik | F-Ölçüsü |
|---|---|---|---|---|
| Lojistik Regresyon | %98.60 | 0,96 | 0,95 | 0,95 |
| Naive Bayes | %97.32 | 0,93 | 0,98 | 0,95 |
| kNN | %98.37 | 0,93 | 0,98 | 0,95 |

Çizelgede değerlendirme metriklerinin sonuçlarına bakıldığında doğruluk metriğine göre en başarılı algoritma Lojistik Regresyon olmuştur. Ancak diğer algoritmalar da oldukça başarılı sonuçlar göstermiştir.

# 5. TARTIŞMA VE SONUÇ

Bu tez çalışmasında üç farklı makine öğrenme algoritması Lojistik Regresyon, Naive Bayes ve kNN ve bu öğrenme algoritmalarının performanslarını değerlendirmek için doğruluk, duyarlılık, kesinlik ve F-ölçüsü kullanılmıştır.

Yapılan analziler sonucunda öncelikle HTML ve javascript tabanlı Nitelikler kullanılarak algoritma performanslarının karşılaştırılması yapılmıştır, (Çizelge 4.3). Bu karşılaştırma sonucunda performans değerlendirme ölçütlerine göre Lojistik Regresyon ve Naive Bayes makine öğrenme algoritmalarının benzer sonuçlar verdiği görülmüştür.

Domain tabanlı nitelikler kullanılarak algoritma performanslarının karşılaştırılmasında doğruluk, duyarlılık, kesinlik ve F-ölçüsü performans ölçülerine göre Lojistik Regresyon ve Naive Bayes aynı sonuçları vermiştir,(Çizelge 4.4). kNN algoritması diğer iki algoritma ile karşılaştırıldığında F- ölçüsünde aynı sonucu, kesinlik ölçüsüne göre daha düşük (0,97), diğer iki performans ölçüsünde ise daha yüksek performans göstermiştir.

Son olarak tüm nitelikler kullanılarak algoritma performanslarının karşılaştırılması sonucunda elde edilen bilgiler ışığında; F- ölçüsü performansının tüm makine öğrenme algoritmalarında eşit, doğruluk performans ölçüsü ise sırasıyla lojistik regresyon (%98,60), kNN (%98,37) ve Naive Bayes (%97,32) şeklinde sonuçlar vermiştir. Duyarlılık performans ölçüsüne göre en iyi sonuçları veren makine öğrenme algoritması Lojistik Regresyon algoritmasıdır (0,96). Kesinlik performans ölçüsüne göre Naive Bayes ve KNN algoritmaları (0,98) benzer sonuçlar verirken, Lojistik Regresyon 0,95 ile daha düşük bir performans göstermiştir. Benzer veri setlerinin kullanıldığı çalışmalar ile elde edilen sonuçlar karşılaştırıldığında benzer sonuçlar elde edildiği görülmüştür, (Çizelge 4.5).

Kuşkusuz Phisihing saldırılarına karşı korunmanın en önemli yolu internet kullanıcılarının teknoloji okur - yazarlığını artırmak ve bu alanda nitelikli eğitim olanakları sağlamaktır. Kullanıcıların güvenilir URL adresleri hakkında bilinçlendirmesi ve buna ilişkin eğitim olanaklarının artırılması bu tarz saldırıların azalmasında oldukça önemli bir rol oynayacaktır. Bir web sayfasını ziyaret ederken URL'yi kontrol etmek Phising saldırılarının azalmasını sağlayacaktır. Aynı zamanda

internet kullanıcılarının e-posta içeriklerini kontrol edebilecek düzeyde bir farkındalığa sahip olmaları ek olarak bu alandaki saldırıların doğrusal olarak azalmasında da etkili olacaktır.

Makine öğrenme algoritmaları son yıllarda siber güvenlik alanında oldukça yaygın bir biçimde kullanılarak; kötü amaçlı URL algılama ve önlem alma konusunda pek çok kolaylık sağlamaktadır. Mevcut durumda bu alanda yapılan çalışmalar elde edilen verilerin makine öğrenme algoritmaları yardımıyla sınıflandırılması ve sahte URL adreslerinin tespit edilmesini sağlamaktadır. Bu alandaki teknolojinin gelişmesi ile birlikte web tarayıcılarında yer alacak otomatik URL algılama eklentileri sayesinde kötü amaçlı URL adreslerinin ve bu adresler üzerinden gerçekleştirilen siber saldırıların önlemi alınacaktır.

Bu çalışmada kullanılan üç makine öğrenme algoritması dışında; lineer regresyon, karar ağaçları, SVm, Random Forest vd. kullanılarak elde edilen URL bilgi tabanının analizlenmesi sonucunda performans ölçülerine göre makine öğrenme algoritmalarının karşılaştırmaları yapılabilir.

## KAYNAKLAR


Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S. 2007**.** A comparison of machine learning techniques for phishing detection. ***In Proceedings Of The Anti-Phishing Working Groups 2nd Annual Ecrime Researchers Summit*** (pp. 60-69). ACM.

Aburrous, M., Hossain, M. A., Dahal, K., & Thabtah, F. 2010. Intelligent phishing detection system for e-banking using fuzzy data mining. ***Expert Systems With Applications***, 37(12), 7913-7921.

Alnajim, A., Munro, M. 2009. ***An Approach to the Implementation of the Anti-Phishing Tool for Phishing Websites Detection***. In Intelligent Networking and Collaborative Systems, 2009. INCOS'09. International Conference on (pp. 105-112). IEEE.

Alpaydin, E., 2016. ***Yapay Öğrenme***, Boğaziçi Üniversitesi Yayınevi, İstanbul. 37-259.

AWIC, 2016. Anti Phishing Working Group (APWG) Phishing activity trends report, 2nd quarter 2016 ***http://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf*****.** Erişim tarihi:10.04.2018

Basnet, R., Mukkamala, S., Sung, A. H. 2008. ***Detection Of Phishing Attacks: A machine learning approach***. In Soft Computing Applications in Industry (pp. 373-383). Springer Berlin Heidelberg.

Brownlee, J. 2014. Machine learning mastery. ***http://machinelearningmastery.com/discover-feature-engineering-howtoengineer-features-and-how-to-getgood-at-it.*** Erişim Tarihi : 16.02.2018

Choo, X. M., Chiew, K. L., Ibrahim, D. H. A., Musa, N., Tiong, W. K. 2016. Feature-based phishing detection technique. ***Journal of Theoretical and Applied Information Technology***, **91**(1): 101.

Dhanalakshmi, R., Prabhu, C., Chellapan, C. 2011. ***Detection of phishing websites and secure transactions*****. IJCNS, 1**(11): 15-21.

Fix, E., Hodges Jr, J. L. 1951. ***Discriminatory analysis-nonparametric discrimination: consistency properties***. California Univ Berkeley.

Flach, P. 2012. ***Machine Learning: The Art and Science of Slgorithms That Make Sense of Data***. Cambridge University Press.

Fu, A. Y., Wenyin, L., Deng, X. 2006. ***Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD).*** IEEE Transactions on Dependable and Decure Computing, 3(4), 301-311.

Garera, S., Provos, N., Chew, M., Rubin, A. D. 2007. A framework for detection and measurement of phishing attacks. ***In Proceedings of the 2007 ACM workshop on Recurring malcode*** (pp. 1-8). ACM.

Genkina, A., L. J. Camp 2007. Phishing and Countermeasures: ***Understanding the Increasing Problem of Electronic Identity Theft, chapter Case Study***: "Net Trust". John Wiley & Sons.

Gowtham, R., Krishnamurthi, I. 2014. ***A comprehensive and efficacious architecture for detecting phishing webpages***. Computers & Security, 40, 23-37.

Hamid, I. R. A., Abawajy, J. H. 2014. ***An approach for profiling phishing activities***. Computers & Security, 45, 27-41.

Hassan, D. 2015. On determining the most effective subset of features for detecting phishing websites**.** ***International Journal of Computer Applications*** (0975-8887), 122(20).

Han, J., Kamber, M., Pei, J. 2006. ***Data mining: concepts and techniques***, (the morgan kaufmann series in data management systems). pp-230-240.

Hodzic, A., Kevric, J., Karadag, A. 2016. ***Comparison of Machine Learning Techniques in Phishing Website Classification***.

Hong, J. 2012. ***The state of phishing attacks. Communications of the ACM***, 55(1), 74-81.

Hu, C., Xu, Z., Liu, Y., Mei, L., Chen, L., Luo, X. 2014. ***Semantic link network-based model for organizing multimedia big data***. IEEE Transactions on Emerging Topics in Computing, 2(3), 376-387.

Japkowicz and Shah, 2011 N. Japkowicz, M. Shah **Evaluating Learning Algorithms:A classification perspective** Cambridge University Press, Cambridge; New York (2011)

Jain, A. K., Gupta, B. B. 2018. PHISH-SAFE: ***URL Features-Based Phishing Detection System Using Machine Learning.*** In Cyber Security: Proceedings of CSI 2015 (pp. 467-474). Springer Singapore.

Kan, M. Y., Thi, H. O. N. 2005**.** ***Fast webpage classification using URL features***. In Proceedings of the 14th ACM international conference on Information and knowledge management (pp. 325-326). ACM.

Kartal, E., 2015 ***Sınıflandırmaya dayalı makine öğrenmesi teknikleri ve kardiyolojik risk değerlendirmesine ilişkin bir uygulama***. İstanbul Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul

Kaytan, M., Hanbay, D. 2017. ***Effective classification of phishing web pages based on new rules by using extreme learning machines***. Anatolian Journal of Computer Sciences, 2(1), 15-36.

Kazemian, H. B., Ahmed, S. 2015. ***Comparisons of machine learning techniques for detecting malicious webpages***. Expert Systems with Applications, 42(3), 1166-1177.

Lakshmi, V. S., Vijaya, M. S. 2012. ***Efficient prediction of phishing websites using supervised learning algorithms***. Procedia Engineering, 30, 798-805.

Le, A., Markopoulou, A., & Faloutsos, M. 2011. ***Phishdef: Url names say it all. In 2011 Proceedings IEEE INFOCOM*** (pp. 191-195). IEEE.

Lee, J. L., Kim, D. H., Chang-Hoon, 2015**.** ***Heuristic-based Approach for Phishing Site Detection Using URL Features***. Proc. of the Third Intl. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2015

Li, Y., Xiao, R., Feng, J., Zhao, L. 2013. ***A semi-supervised learning approach for detection of phishing webpages***. Optik-International Journal for Light and Electron Optics, 124(23), 6027-6033.

Li, Y., Yang, L., Ding, J. 2016. A minimum enclosing ball-based support vector machine approach for detection of phishing websites. Optik-International Journal for Light and Electron Optics, 127(1), 345-351.

Marchal, S., Saari, K., Singh, N., Asokan, N. 2016. ***Know your phish: Novel techniques for detecting phishing sites and their targets***. In Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on (pp. 323-333). IEEE.

Martin, A., Anutthamaa, N., Sathyavathy, M., Francois, M. M. S., Venkatesan, D. V. P. 2011. ***A framework for predicting phishing websites using neural networks***. arXiv preprint arXiv:1109.1074.

Maurizio, M. 2011. Data Mining Concepts And Techniques. ***https://s3.amazonaws.com/academia.edu.documents/35676056/06_-_datamining.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1533631476&Signature=0lvS1wiX7LSEfuwTuy7e6XckL%2Fo%3D&response-content-disposition=inline%3B%20filename%3DBai_dc_tham_kho_Phan_tich_va_k_thut.pdf.*** Erişim tarihi: 10.05.2018

Milletary, J., Center, C. C. 2005. ***Technical trends in phishing attacks***. Retrieved December, 1(2007), 3-3.

Mitchell, T. M. 1997. ***Machine learning*** (mcgraw-hill international editions computer science series), *18*(3), 11.

Moghimi, M., Varjani, A. Y. 2016. ***New rule-based phishing detection method***. Expert systems with applications, 53, 231-242.

Mohammad, R. M., Thabtah, F., McCluskey, L. 2013. ***Predicting phishing websites using neural network trained with back-propagation***. In Proceedings on the International Conference on Artificial Intelligence (ICAI) (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Mohammad, R. M., Thabtah, F., McCluskey, L. 2015. ***Tutorial and critical analysis of phishing websites methods***. Computer Science Review, 17, 1-24.7

Mohri, M., Rostamizadeh, A., Talwalkar, A. 2012. ***Foundations Of Machine Learning***. MIT press.

Ng, A., 2000. CS229 Lecture notes. ***http://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf.*** Erişim tarihi:11.02.2018

Pan, Y., Ding, X. 2006. ***Anomaly based web phishing page detection***. In Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual (pp. 381-392). IEEE.

Pattekari, S. A., Parveen, A. 2012. Prediction system for heart disease using Naïve Bayes. International ***Journal of Advanced Computer and Mathematical Sciences***, 3(3), 290-294.

Ramesh, G., Krishnamurthi, I., Kumar, K. S. S. 2014. ***An efficacious method for detecting phishing webpages through target domain identification***. Decision Support Systems, 61, 12-22.

Ramzan, Zulfikar 2010. ***Phishing attacks and countermeasures. In Stamp***, Mark & Stavroulakis, Peter. Handbook of Information and Communication Security. Springer. ISBN 978-3-642-04117-4.

Rao, R. S., Pais, A. R. 2018. ***Detection of phishing websites using an efficient feature-based machine learning framework***. Neural Computing and Applications, 1-23.

Shearer, C. 2000. The CRISP-DM model: the new blueprint for data mining. Journal of data warehousing, 5(4), 13-22.

Shahriar, H., Zulkernine, M. 2012. ***Trustworthiness testing of phishing websites: A behavior model-based approach***. Future Generation Computer Systems, 28(8), 1258-1271.

Subasi, A., Molah, E., Almkallawi, F., Chaudhery, T. J. 2017. ***Intelligent phishing website detection using random forest classifier***. In Electrical and Computing Technologies and Applications (ICECTA), 2017 International Conference on (pp. 1-5). IEEE.

Tan, C. L., Chiew, K. L., Wong, K. 2016. ***PhishWHO: Phishing webpage detection via identity keywords extraction and target domain name finder***. Decision Support Systems, 88, 18-27.

Topkara, M., Kamra, A., Atallah, M. J., Nita-Rotaru, C. 2005. ***Viwid: Visible watermarking based defense against phishing***. In International Workshop on Digital Watermarking (pp. 470-483). Springer, Berlin, Heidelberg.

Van der Merwe, A J, Loock, M, Dabrowski, M. 2005. ***Characteristics and Responsibilities involved in a Phishing Attack***, Winter International Symposium on Information and Communication Technologies, Cape Town, January 2005.

Wenyin, L., Liu, G., Qiu, B., Quan, X. 2012. ***Antiphishing through phishing target discovery***. IEEE Internet Computing, 16(2), 52-61.

Xiang, G., Hong, J., Rose, C. P., Cranor, L. 2011. ***Cantina+: A feature-rich machine learning framework for detecting phishing web sites***. ACM Transactions on Information and System Security (TISSEC), 14(2), 21.

Zhang, Y., Hong, J. I., Cranor, L. F. 2007. ***Cantina: a content-based approach to detecting phishing web sites.*** In Proceedings of the 16th international conference on World Wide Web (pp. 639-648). ACM.

**EKLER**

Ek-1.1. Lojistik Regresyon

```
# Dataset'i dosyadan yükle( .csv veya .txt )
def dosya_yukle( dosyaAdi):
	dataset = list()
	with open( dosyaAdi, 'r' ) as file:
		csv_okuyucu = reader( file )
		for satir in csv_okuyucu:
			if not satir:
				continue
			dataset.append( satir )
	return dataset

# Alanları float tipine çevir
def alanlari_floata_cevir( dataset, alan):
	for satir in dataset:
		satir[ alan ] = float( satir[ alan ].strip() )

# Dataseti k adet bölmeye ayır
def cross_validation_bolme( dataset, n_bolum ):
	dataset_bolum = list()
	dataset_kopya = list( dataset )
	bolum_sayisi = int( len( dataset ) / n_bolum )
	for i in range( n_bolum ):
		bolum = list()
		while len( bolum ) < bolum_sayisi:
			index = randrange( len( dataset_kopya ) )
			bolum.append(dataset_kopya.pop( index ) )
		dataset_bolum.append( bolum )
	return dataset_bolum
```

```
# Doğruluk metriği
def dogruluk_metrigi( gercek, tahmin ):
        dogru_sayisi = 0
        for i in range( len( gercek ) ):
                if gercek[ i ] == tahmin[ i ]:
                        dogru_sayisi += 1
        return dogru_sayisi / float( len( gercek ) ) * 100.0

# cross_validation_bolme ile değerlendirme algoritması
def degerlendirme_algoritmasi( dataset, algoritma, n_bolum, *args ):
        bolumler = cross_validation_bolme( dataset, n_bolum )
        degerlendirme = list()
        for bolum in bolumler:
                ogrenme_set = list( bolumler )
                ogrenme_set.remove( bolum )
                ogrenme_set = sum( ogrenme_set, [] )
                test_set = list()
                for satir in bolum:
                        satir_kopya = list( satir )
                        test_set.append( satir_kopya )
                        satir_kopya[ -1 ] = None
                tahmin = algoritma( ogrenme_set, test_set, *args )
                gercek = [ satir[ -1 ] for satir in bolum ]
                dogruluk = dogruluk_metrigi( gercek, tahmin )
                degerlendirme.append( dogruluk )
        return degerlendirme

# Tahmin fonksiyonu
def tahmin( satir, katsayilar ):
        ytahmin = katsayilar[ 0 ]
        for i in range( len( satir ) -1 ):
                ytahmin += katsayilar[ i + 1 ] * satir[i]
```

```
        return 1.0 / ( 1.0 + exp( -ytahmin ) )

# stochastic gradient descent algoritması ile katsayıları tahin et
def katsayilar_sgd( ogrenme, adim_buyuklugu, iterasyon ):
        katsayi = [ 0.0 for i in range( len( ogrenme[ 0 ] ) ) ]
        for epoch in range( iterasyon ):
                for satir in ogrenme:
                        ytahmin = tahmin( satir, katsayi )
                        error = satir[ -1 ] - ytahmin
                        katsayi[ 0 ] = katsayi[ 0 ] + adim_buyuklugu * error * ytahmin * (
1.0 - ytahmin )
                        for i in range( len( satir ) -1 ):
                                katsayi[ i + 1 ] = katsayi[ i + 1 ] + adim_buyuklugu * error
* ytahmin * ( 1.0 - ytahmin ) * satir[ i ]
        return katsayi

# Linear Regression Algorithm With Stochastic Gradient Descent
def lojistik_regresyon( ogrenme, test, adim_buyuklugu, iterasyon ):
        predictions = list()
        katsayi = katsayilar_sgd( ogrenme, adim_buyuklugu, iterasyon )
        for satir in test:
                ytahmin = tahmin( satir, katsayi )
                ytahmin = round( ytahmin )
                predictions.append( ytahmin )
        return( predictions )




# Test the logistic regression algorithm on the diabetes dataset
seed( 1 )
# Veriyi yükle ve hazırla
```

```
dosyaAdi = '/home/frt/Desktop/a.txt'
dataset = dosya_yukle( dosyaAdi )
for i in range( len( dataset[ 0 ] ) ):
    alanlari_floata_cevir( dataset, i )

# Değerlendirme
n_bolum = 5
adim_buyuklugu = 0.1
iterasyon = 100
degerlendirme = degerlendirme_algoritmasi( dataset, lojistik_regresyon, n_bolum, adim_buyuklugu, iterasyon )
print( 'degerlendirme: %s' % degerlendirme )
print( 'Mean Accuracy: %.3f%%' % ( sum( degerlendirme ) / float( len( degerlendirme ) ) ) )
```

Ek-1.2. kNN

```
from random import seed
from random import randrange
from csv import reader
from math import sqrt

# Dataset'i dosyadan yükle( .csv veya .txt )
def dosya_yukle( dosyaAdi):
	dataset = list()
	with open( dosyaAdi, 'r' ) as file:
		csv_okuyucu = reader( file )
		for satir in csv_okuyucu:
			if not satir:
				continue
			dataset.append( satir )
	return dataset

# Alanları float tipine çevir
def alanlari_floata_cevir( dataset, alan):
	for satir in dataset:
		satir[ alan ] = float( satir[ alan ].strip() )

# Parametreleri sayısal tipine çevir
def metin_kolonlarini_sayisala_cevir( dataset, kolon ):
	sinif_degerleri = [ satir[ kolon ] for satir in dataset ]
	unique = set( sinif_degerleri )
	sozluk_ara = dict()
	for i, value in enumerate( unique ):
		sozluk_ara[ value ] = i
	for satir in dataset:
		satir[ kolon ] = sozluk_ara[ satir[ kolon ] ]
```

```
        return sozluk_ara

# Dataseti k adet bölmeye ayır
def cross_validation_bolme( dataset, n_bolum ):
        dataset_bolum = list()
        dataset_kopya = list( dataset )
        bolum_sayisi = int( len( dataset ) / n_bolum )
        for i in range( n_bolum ):
                bolum = list()
                while len( bolum ) < bolum_sayisi:
                        index = randrange( len( dataset_kopya ) )
                        bolum.append(dataset_kopya.pop( index ) )
                dataset_bolum.append( bolum )
        return dataset_bolum

# Doğruluk metriği
def dogruluk_metrigi( gercek, tahmin ):
        dogru_sayisi = 0
        for i in range( len( gercek ) ):
                if gercek[ i ] == tahmin[ i ]:
                        dogru_sayisi += 1
        return dogru_sayisi / float( len( gercek ) ) * 100.0

# cross_validation_bolme ile değerlendirme algoritması
def degerlendirme_algoritmasi( dataset, algoritma, n_bolum, *args ):
        bolumler = cross_validation_bolme( dataset, n_bolum )
        degerlendirme = list()
        for bolum in bolumler:
                ogrenme_set = list( bolumler )
                ogrenme_set.remove( bolum )
                ogrenme_set = sum( ogrenme_set, [] )
                test_set = list()
```

```
                for satir in bolum:
                        satir_kopya = list( satir )
                        test_set.append( satir_kopya )
                        satir_kopya[ -1 ] = None
                tahmin = algoritma( ogrenme_set, test_set, *args )
                gercek = [ satir[ -1 ] for satir in bolum ]
                dogruluk = dogruluk_metrigi( gercek, tahmin )
                degerlendirme.append( dogruluk )
        return degerlendirme

# Öklid uzaklıklarını hesapla
def oklid_uzaligi( satir1, satir2 ):
        uzaklik = 0.0
        for i in range( len( satir1 ) -1 ):
                uzaklik += ( satir1[ i ] - satir2[ i ] )**2
        return sqrt( uzaklik )

# Benzer komşuları bul
def komsulari_bul( egitim, test_satiri, komsu_sayisi):
        uzakliklar = list()
        for train_row in egitim:
                uzaklik = oklid_uzaligi( test_satiri, train_row )
                uzakliklar.append( ( train_row, uzaklik ) )
        uzakliklar.sort( key = lambda tup: tup[ 1 ] )
        komsular = list()
        for i in range( komsu_sayisi ):
                komsular.append( uzakliklar[ i ][ 0 ] )
        return komsular

# Komşuları için tahminde bulun
def siniflandirma_tahmini(egitim, test_satiri, komsu_sayisi):
        komsular = komsulari_bul( egitim, test_satiri, komsu_sayisi )
```

```
        cikti_degerleri = [ row[ -1 ] for row in komsular ]
        tahmin = max( set( cikti_degerleri ), key = cikti_degerleri.count )
        return tahmin

# kNN Algoritması
def k_yakin_komsu( egitim, test, komsu_sayisi ):
        tahminler = list()
        for row in test:
                cikti = siniflandirma_tahmini( egitim, row, komsu_sayisi )
                tahminler.append( cikti )
        return( tahminler )

seed(1)

dosyaAdi = '/home/frt/Desktop/a.txt'
dataset = dosya_yukle( dosyaAdi )
for i in range( 1, len( dataset[ 0 ] ) ):
        str_column_to_float( dataset, i )

metin_kolonlarini_sayisala_cevir( dataset, 0 )
n_bolum = 5
komsu_sayisi = 5

degerlendirme = degerlendirme_algoritmasi( dataset, k_yakin_komsu, n_bolum, komsu_sayisi )
print( 'degerlendirme: %s' % degerlendirme )
print( 'Mean Accuracy: %.3f%%' % ( sum( degerlendirme ) / float( len( degerlendirme ) ) ) )
```

Ek-1.3. Naive Bayes

```
from csv import reader
from random import randrange
from math import sqrt
from math import exp
from math import pi

# Dataset'i dosyadan yükle( .csv veya .txt )
def dosya_yukle( dosyaAdi):
    dataset = list()
    with open( dosyaAdi, 'r' ) as file:
        csv_okuyucu = reader( file )
        for satir in csv_okuyucu:
            if not satir:
                continue
            dataset.append( satir )
    return dataset

# Alanları float tipine çevir
def alanlari_floata_cevir( dataset, alan ):
    for satir in dataset:
        satir[ alan ] = float( satir[ alan ].strip() )

# Parametreleri sayısal tipine çevir
def metin_kolonlarini_sayisala_cevir( dataset, kolon ):
    sinif_degerleri = [ satir[ kolon ] for satir in dataset ]
    unique = set( sinif_degerleri )
    sozluk_ara = dict()
    for i, value in enumerate( unique ):
        sozluk_ara[ value ] = i
    for satir in dataset:
```

```
                satir[ kolon ] = sozluk_ara[ satir[ kolon ] ]
        return sozluk_ara

# Dataseti k adet bölmeye ayır
def cross_validation_bolme( dataset, n_bolum ):
        dataset_bolum = list()
        dataset_kopya = list( dataset )
        bolum_sayisi = int( len( dataset ) / n_bolum )
        for i in range( n_bolum ):
                bolum = list()
                while len( bolum ) < bolum_sayisi:
                        index = randrange( len( dataset_kopya ) )
                        bolum.append(dataset_kopya.pop( index ) )
                dataset_bolum.append( bolum )
        return dataset_bolum

# Doğruluk metriği
def dogruluk_metrigi( gercek, tahmin ):
        dogru_sayisi = 0
        for i in range( len( gercek ) ):
                if gercek[ i ] == tahmin[ i ]:
                        dogru_sayisi += 1
        return dogru_sayisi / float( len( gercek ) ) * 100.0

# cross_validation_bolme ile değerlendirme algoritması
def degerlendirme_algoritmasi( dataset, algoritma, n_bolum, *args ):
        bolumler = cross_validation_bolme( dataset, n_bolum )
        degerlendirme = list()
        for bolum in bolumler:
                ogrenme_set = list( bolumler )
                ogrenme_set.remove( bolum )
                ogrenme_set = sum( ogrenme_set, [] )
```

```
                test_set = list()
                for satir in bolum:
                        satir_kopya = list( satir )
                        test_set.append( satir_kopya )
                        satir_kopya[ -1 ] = None
                tahmin = algoritma( ogrenme_set, test_set, *args )
                gercek = [ satir[ -1 ] for satir in bolum ]
                dogruluk = dogruluk_metrigi( gercek, tahmin )
                degerlendirme.append( dogruluk )
        return degerlendirme

# Dataseti sınıflara göre ayır, ve ayrımları geri yolla
def sinifa_gore_ayir( dataset ):
        ayrimlar = dict()
        for i in range( len( dataset ) ):
                vektor = dataset[ i ]
                sinif_degeri = vektor[ -1 ]
                if (sinif_degeri not in ayrimlar):
                        ayrimlar[ sinif_degeri ] = list()
                ayrimlar[ sinif_degeri ].append( vektor )
        return ayrimlar

# Ortalama hesapla
def ortalama( degerler ):
        return sum( degerler ) / float( len( degerler ) )

# Standart sapmayı hesapla
def standart_sapma( degerler ):
        ort = ortalama( degerler )
        varyans = sum( [ ( x-ort )**2 for x in degerler ] ) / float( len( degerler ) -1 )
        return sqrt( varyans )
```

```
# Her bir colon için standart sapma ve ortalmayı bul
def dataset_ozeti( dataset ):
    deger = [ ( ortalama( kolon ), standart_sapma( kolon ), len( kolon ) ) for kolon in zip( *dataset ) ]
    del( deger[ -1 ] )
    return deger

# Dataseti sınıflara ayır ve her satır için istatistikleri hesapla
def dataseti_siniflara_gore_ozetle( dataset ):
    ayrim = sinifa_gore_ayir( dataset )
    degerler = dict()
    for sinif_degeri, satirlar in ayrim.iteritems():
        degerler[sinif_degeri] = dataset_ozeti( satirlar )
    return degerler

# x için Gaus olasılık dağılımını hesapla
def olasilik_hesapla( x, ortalama, standart_sapma ):
    kuvvet = exp( - ( ( x-ortalama )**2 / ( 2 * standart_sapma**2 ) ) )
    return ( 1 / ( sqrt( 2 * pi ) * standart_sapma ) ) * kuvvet

# Her sınıf için olasılıkları hesapla
def sinif_olasiliklarini_hesapla( degerler, satir ):
    toplam_satir = sum([degerler[label][0][2] for label in degerler])
    olasiliklar = dict()
    for sinif_degeri, sinif_ozeti in degerler.iteritems():
        olasiliklar[ sinif_degeri ] = degerler[ sinif_degeri ][ 0 ][ 2 ] / float( toplam_satir )
        for i in range( len( sinif_ozeti ) ):
            ortalama, standart_sapma, count = sinif_ozeti[ i ]
            olasiliklar[ sinif_degeri ] *= olasilik_hesapla( satir[ i ], ortalama, standart_sapma )
    return olasiliklar
```

```
# Her satır için tahmin bul
def tahmin( degerler, satir ):
	olasiliklar = sinif_olasiliklarini_hesapla( degerler, satir )
	en_iyi_etiket, en_iyi_olasilik = None, -1
	for sinif_degeri, olasilik in olasiliklar.iteritems():
		if en_iyi_etiket is None or olasilik > en_iyi_olasilik:
			en_iyi_olasilik = olasilik
			en_iyi_etiket = sinif_degeri
	return en_iyi_etiket

# Naive Bayes Algoritması
def naive_bayes( egitim, test ):
	summarize = dataseti_siniflara_gore_ozetle( egitim )
	tahminler = list()
	for satir in test:
		output = tahmin(summarize, satir)
		tahminler.append(output)
	return( tahminler )
# Test Naive Bayes on Iris Dataset
dosyaAdi = '/home/frt/Desktop/a.txt'
dataset = dosya_yukle( dosyaAdi )
for i in range( len( dataset[ 0 ] ) -1 ):
	alanlari_floata_cevir( dataset, i )
metin_kolonlarini_sayisala_cevir( dataset, len( dataset[ 0 ] ) -1 )
n_bolum = 5
degerlendirme = degerlendirme_algoritmasi( dataset, naive_bayes, n_bolum )
print( 'degerlendirme: %s' % degerlendirme )
print( 'Mean Accuracy: %.3f%%' % ( sum( degerlendirme ) / float( len( degerlendirme )
) ) )
```

# ÖZGEÇMİŞ

1986 yılında Uludere'de doğdu. İlk, orta ve lise öğrenimini Cizre'de tamamladı. 2010 yılında Mersin Üniversitesi, Bilgisayar Mühendisliği Bölümü'nden mezun oldu. 2015 yılında Van Yüzüncü Yıl Üniversitesi, Özalp Meslek Yüksekokulu'nda, Öğretim Görevlisi olarak çalışmaya başladı. 2015 yılında Van Yüzüncü Yıl Üniversitesi Fen Bilimleri Enstitüsü, İstatistik Anabilim Dalı'nda Yüksek Lisans eğitimine başladı.

**T.C**
**VAN YÜZÜNCÜ YIL ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**
**LİSANSÜSTÜ TEZ ORİJİNALLİK RAPORU**

**Tarih:** 27/09/2018

Tez Başlığı / Konusu: Phishing Saldırısında Kullanılan Web Sitelerinin Makine Öğrenmesi Algoritmaları Yardımıyla Tespiti ve Uygulaması

Yukarıda başlığı/konusu belirlenen tez çalışmamın Kapak sayfası, Giriş, Ana bölümler ve Sonuç bölümlerinden oluşan toplam 67 sayfalık kısmına ilişkin, 27/09/2018. tarihinde şahsım/tez danışmanım tarafından Turnitin intihal tespit programından aşağıda belirtilen filtreleme uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı %5 (Beş) dır.

Uygulanan filtreler aşağıda verilmiştir:

- Kabul ve onay sayfası hariç,
- Teşekkür hariç,
- İçindekiler hariç,
- Simge ve kısaltmalar hariç,
- Gereç ve yöntemler hariç,
- Kaynakça hariç,
- Alıntılar hariç,
- Tezden çıkan yayınlar hariç,
- 7 kelimeden daha az örtüşme içeren metin kısımları hariç (Limit inatch size to 7 words)

Van Yüzüncü Yıl Üniversitesi Lisansüstü Tez Orijinallik Raporu Alınması ve Kullanılmasına İlişkin Yönergeyi inceledim ve bu yönergede belirtilen azami benzerlik oranlarına göre tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Gereğini bilgilerinize arz ederim.

27/09/2018

Adı Soyadı: Fırat KAPAR

Öğrenci No:149102194

Anabilim Dalı: İstatistik

Programı: Tezli Yüksek Lisans

Statüsü: Y. Lisans ☒ Doktora ☐

**DANIŞMAN ONAYI**
UYGUNDUR

Prof. Dr. H. Eray ÇELİK

**ENSTİTÜ ONAYI**
UYGUNDUR

Prof. Dr. Fuat ŞENSOY
Enstitü Müdürü