T.C. İSTANBUL TİCARET ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

SİBER GÜVENLİKTE KLAVYE DAVRANIŞ ANALİZİ

Nurgül AKŞİT

Danışman
Prof. Dr. Abdül Halim ZAİM

İkinci Tez Danışmanı
Doç. Dr. Muhammed Ali AYDIN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
İSTANBUL-2022

KABUL VE ONAY SAYFASI

Nurgül AKŞİT tarafından hazırlanan **"Siber Güvenlikte Klavye Davranış Analizi"** adlı tez çalışması 24/01/2022 tarihinde aşağıdaki jüri üyeleri önünde başarı ile savunularak, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman	**Prof. Dr. Abdül Halim ZAİM** İstanbul Ticaret Üniversitesi	.....
Jüri Üyesi	**Doç. Dr. Mustafa Cem KASAPBAŞI** İstanbul Ticaret Üniversitesi	.....
Jüri Üyesi	**Dr. Öğr. Üyesi Özgür Can TURNA** İstanbul Üniversitesi-Cerrahpaşa	.....

Onay Tarihi : 10.02.2022

İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsünün 10.02.2022 tarih ve 2022/339 numaralı Yönetim Kurulu Kararının 2. maddesi gereğince, ders yüklerini ve tez yükümlülüğünü yerine getirdiği belirlenen "Nurgül Akşit" adlı öğrencinin mezun olmasına oy birliği ile karar verilmiştir.

Prof. Dr. Necip ŞİMŞEK
Enstitü Müdürü

AKADEMİK VE ETİK KURALLARA UYGUNLUK BEYANI

İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

Nurgül AKŞİT

İÇİNDEKİLER

ÖZET

Yüksek Lisans Tezi

SİBER GÜVENLİKTE KLAVYE DAVRANIŞ ANALİZİ

Nurgül AKŞİT

İstanbul Ticaret Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Abdül Halim ZAİM

İkinci Tez Danışmanı: Doç. Dr. Muhammed Ali AYDIN
2022, 58 sayfa

Bu çalışmada,
Kullanıcıların klavye kullanım alışkanlıklarına ve bu alışkanlıkların bilgi güvenliğine katkılarına odaklanıyoruz. Her biri birbirinden farklı alışkanlıklara sahip kullanıcıların verilerini geliştirdiğimiz Klavye Davranış Analizi (KDA) programı ile alıyor, belirlenmiş örneklemleri makine öğrenmesi ile yapay zekada kullanılması için analizler yapıyor ve sonuçlarını paylaşıyoruz.

Geliştirme kodları ile paylaşımını yaptığımız çalışmamızın; kullanıcıların bilgisayarını ele geçiren kötü niyetli saldırganlar tarafından kullanıldığında sistemi korumak adına aksiyonlar aldıran ve alarmlar üreten bir yapıya dönüşmesi veya kimlik doğrulama sistemi olarak kullanılabileceği yönünde fikriler sunuyor, bu konuda çalışmalar yapmak istenlere temel oluşturuyoruz.

Geliştirilen KDA isimli programımızın yüklendiği on kullanıcıdan alınan gerçek verileri analiz ederken sonuçların siber güvenlikte çoklu doğrulama sistemlerinde kullanılabilecek ve dinamik doğrulama yöntemi ile klavye kullanım alışkanlığının kullanıcıya ait olup olmadığını sürekli test edebilecek derin öğrenme modelinde paylaşımlar yapıyoruz.

2019 yılında Çin'de ortaya çıkan ve tüm dünyayı etkisi altına alan Covid-19 salgını ile birlikte değişen çalışma şartlarını bilgi sistemleri üzerinde daha güvenli bir ortam haline getirmek için uzaktan çalışma risklerine değiniyoruz. Çalışmanın sonuçları siber güvenlik projelerine fayda sağlayacaktır.

Anahtar Kelimeler: Derin öğrenme, klavye kullanım alışkanlığı, kullanıcı davranış analizi, makine öğrenmesi, siber güvenlik önlemleri, siber savunma yöntemleri.

ABSTRACT

M.Sc. Thesis

KEYBOARD BEHAVIOR ANALYSIS IN CYBERSECURITY

Nurgul AKSIT

Istanbul Commerce University
Graduate School of Applied and Natural Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. Abdul Halim ZAIM

Co-Supervisor: Assoc. Prof. Dr. Muhammed Ali AYDIN
2022, 58 pages

In this study,
We focus on the keyboard usage habits of users and the contribution of these habits to information security. We collect the data of users, each with different habits, with the Keyboard Behavior Analysis program we developed, analyze the determined samples for use in machine learning and artificial intelligence and share the results.

Our work that we share with development codes; We offer ideas that it can turn into a structure that takes actions and generates alarms to protect the system when used by malicious attackers who take over users' computers, or that it can be used as an authentication system, and we lay the groundwork for those who want to work on this issue.

While analyzing the real data received from ten users on whom our developed program called KDA was installed, we share the results in a deep learning model that can be used in multiple verification systems in cyber security and can constantly test whether the user's keyboard usage habits belong to the dynamic verification method.

We are talking about the risks of working remotely in order to make the changing working conditions a safer environment on information systems with the Covid-19 epidemic that emerged in China in 2019 and affected the whole world. The results of the study will benefit cyber security projects.

Keywords: Deep learning, keyboard usage habits, user behavior analysis, machine learning, cyber security measures, cyber defense methods.

TEŞEKKÜR

Bu araştırma için beni yönlendiren, yüksek lisans eğitimim boyunca karşılaştığım zorlukları bilgi ve tecrübeleri ile aşmamda yardımcı olan değerli danışman hocalarım Prof. Dr. Abdül Halim Zaim'e ve Doç. Dr. Muhammed Ali Aydın'a teşekkürlerimi sunarım.

Bugün ve her zaman beni destekleyen, maddi ve manevi hiçbir fedakarlıktan kaçınmayan, tüm eğitim hayatım boyunca yanımda olan değerli annem Fahriye Akşit başta olmak üzere tüm aileme sonsuz sevgi ve saygılarımı sunarım.

Nurgül AKŞİT
İSTANBUL, 2022

ŞEKİLLER

ÇİZELGELER

KISALTMALAR

API	Application Programming Interface
CA	Continuous Authentication
CNN	Convolutional Neural Network
KDA	Klavye Davranış Analizi
KVKK	Kişisel Verilerin Korunması Kanunu
LSTM	Long-Short Term Memory
MFA	Multi Factor Authentication
PC	Personal Computer
RNN	Recurrent Neural Network
SA	Static Authentication
SMS	Short Message Service
SOC	Security Operation Center
Vlan	Virtual Local Area Network

1. GİRİŞ

Gelişen ve büyüyen teknoloji ile birlikte covid-19 gibi kamu sağlığını tehdit eden hastalıkların bulaşıcılığını azaltmak adına insan temasını sınırlayan, ticari alışveriş işlemleri, eğitim-öğretim faaliyetleri, fiziki olarak şirkette bulunma şartı taşımayan birçok sektörün uzaktan çalışması ile birlikte hızla artan dijital işlemler, faaliyet gösteren tüm kurum ve kuruluşların siber güvenliğe verdiği önemi daha da artırmıştır. Çalışanlar kurum içerisinde iken kurumun bilgi güvenliği kurallarını daha fazla benimsiyor ve mahremiyet sınırlarını daha rahat çizebiliyordu. Uzaktan çalışmayla kişilerin evde yaşayan diğer bireyler ile birlikte çalışma ortamlarını paylaşması, kafeterya gibi dışarda çalışma ortamı yaratan mekanlarda bulunma durumlarının artması, tüm işlemlerin sanal ortam üzerinden yapılmaya dönüştüğü bu dönemde kurum ve şahsi işlemlerin tek bir bilgisayar üzerinden yapılmaya başlanması bilgi güvenlikçilerini uzaktan çalışma risklerini değerlendirmeye ve pandeminin siber güvenliğe etkisi üzerinde çalışmalar yapmaya itmiştir.

Kurum ve kuruluşlar siber güvenlikte teknik önlemler için ciddi yatırımlar yapmakta, yüksek yetkinliğe sahip çalışanlar istihdam etmekte, dışarıdan bilgi sistemlerinin ne kadar güvenli olduğunu test etmesi için dış kaynaklarla çalışmaktadır. Tüm bu önlemlerin kurumu korumada büyük öneme sahip olduğu yadsınamaz bir gerçek fakat siber tehditlere karşı teknik güvenlik önlemleri ne kadar sağlansa da zincirin en zayıf halkası olan insan faktörü üzerinde daha fazla durulması gerekmektedir. Tabi ki çalışanlar kuruma doğrudan zarar verme niyetinde değillerdir. Kurumlar, düşük farkındalığa sahip çalışanlara eğitim vermek gibi farkındalık artırıcı faaliyetler yapsa bile teknik önlemler ile birlikte bu kanaldan gelen tehditleri bertaraf etmek gerekmektedir. İnsanlar tarafından yapılan bilgi güvenliğini tehdit eden davranışlar; bilgisayarlarını kilitlemeden masalarından ayrılmaları, uzaktan çalışmanın artması ile kamuya açık alanlarda arkadaşları ile sohbet ederken bilgisayarlarının ekranını açık unutmaları, parolalarını not defterleri gibi herkesin erişebileceği yerlere yazarak temiz masa temiz ekran politikalarına uymamaları ve benzeri durumlar bilgilerin ifşasına

sebep olabilecek siber saldırılara zemin hazırlamaktadır. Bu sebeplerden dolayı uzaktan çalışma risklerinin değerlendirilmesi elzem hale gelmiştir.

Bu çalışma ile kullanıcıların klavye kullanım alışkanlıklarının derin öğrenme ile analiz edilerek kimlik tespitlerinin yapılması amaçlanmıştır. Bunun için ek bir yazılım ve donanım masrafı olmadan kişisel verilerle kullanıcı kimliğini doğrulayan bir yöntem modellenmiştir. Daha önce literatürde yer alan bilimsel çalışmalarda mouse(fare) takibi ve klavye kullanımı ile kimlik doğrulama çalışması yapıldığı görülmüş fakat derin öğrenme algoritmaları ile konuyu inceleyip detaylıca ele alan bir çalışma görülememiştir. Bu çalışmada kullanıcı alışkanlıkları modellenmiş ve deneysel testler ile analizlerinin sayısal sonuçları gösterilerek bu eksikliğin giderilmesi hedeflenmiştir.

Parolalar çok uzun zamandır bilgisayar sistemlerine erişimin birincil yöntemi olarak kullanılmakta olup bilgi güvenliğinde önemli bir yere sahiptir. Fakat kullanıcıların kimliklerini doğrulamak için oldukça zayıf bir mekanizmadır. Özellikle zayıf belirlenen parolalar basit parola kırıcı programlar ile kısa sürede çözülebilmektedir. Yapılan araştırmalar kullanıcıların parolalarını unutmamak üzere hayatlarında karşılığı olan kavramlar içinden basit parolalar seçmeye eğilimli olduğunu göstermektedir (Monrose ve Rubin, 2000). Sıklıkla kullanılan parola sıkılaştırma politikaları hemen hemen tüm sistemlerde aynıdır (Monrose vd, 2002). Büyük-küçük harf, en az bir sayı ve özel karakter ile 8-10 karakter uzunluktan oluşmaktadır. Olabilecek tüm parolaların küçük bir alt kümesi ile kolaylıkla tahmin edilebilir niteliktedirler. Bu sebepler bizi erişim sistemlerine girişte ikincil hatta üçüncül doğrulama yöntemlerini araştırmaya sürüklemektedir.

Çalışmamız ile dikkateleri çekmek istediğimiz nokta; çoklu kimlik doğrulama yöntemlerinin ele geçirilme ihtimalleri yüksek olan SMS ve e-posta kanallardan kurtarmak ve aynı zamanda parmak izi gibi Kişisel Verilerin Korunması Kanunu (KVKK) tarafından ayrı teknik ve idari önlemler ile değerlendirilen biyometrik verilerin de alınmamasını sağlamaktır. Bu sayede hem gerçek verilerle doğrulama yapılır hem de hassas kişisel verilerin alınması gibi yüksek

sorumluluklara girilmez. Bu doğrultuda siber saldırgan tarafından yakından veya uzaktan Personel Computer (PC)'yi ele geçirmek için klavyeyi kullandığında (sanal veya fiziksel klavye) geliştirmiş olduğumuz program PC kullanıcısının davranışını öğrenmiş olduğundan farklı davranışı (siber saldırganın hareketlerini) tespit edecektir.

Çalışmanın bir sonraki aşamasında farklı davranışı tespit eden geliştirdiğimiz program, ürettiği log ile birlikte merkezi log sistemine alarm oluşturacak ve saldırı teşebbüsünden Security Operation Center (SOC) ekiplerini haberdar edecektir. Siber güvenlik alt yapısının proaktif olarak yönetildiği kurumlarda alarm ile birlikte ilgili PC'nin karantina Virtual Local Area Network (Sanal Yerel Alan Ağı-VLAN)'a alınması, şirket ağından çıkarılması, internet erişimlerinin kesilmesi ve kullanıcı hesaplarının pasife alınması gibi birçok aksiyonu insan müdahalesi gerektirmeden otomasyonla yapılması sağlanabilecektir.

2. LİTERATÜR ÖZETİ

Monrose vd. (2002) biyometrik bilgilerin şifreye dahil edilmesiyle şifre tabanlı uygulamaların güvenliğinin daha da sıkılaşacağını belirtmiştir. Tuş vuruş dinamiklerine dayalı parola güçlendirmesini hedefleyen çalışmada, parola güvenliğini artırmak için tuş vuruş dinamikleri kullanılmıştır. Parolanın tekrar eden karakterini tespit etmek için süre geçişleri referans alınarak bir şema parola oluşturulmuş ve tuş vuruşları ile kişiye özgü bir giriş yöntemi belirlenmiştir. Kullanıcı parolasını güçlendirmek için ek bir önlem olarak yazma kalıplarının kullanılabileceğini ifade etmişlerdir. Geliştirdikleri programda dosya şifrelemede veya uzun vadeli gizli anahtar ihtiyaçı olan diğer uygulamalarda kullanmak için sıkılaştırılmış aynı parolayı bu sayede korurken, aynı zamanda kullanıcının değişebilecek tuş vuruşlarına da otomatik olarak uyumlanır. Yapılan prototip uygulama ile güvenlik ve performas açısından pratikte uygulanabileceğine dair araştırmalar sunulmuştur.

Juola vd. (2013) klavye, fare ve web sitesi ziyaret sıklığına bakarak davranış analizi çıkaran çalışmalarında, kimlik doğrulama konusunda incelemeler yapmışlardır. 79 katılımcıya kadar belirli gruplarda 10 dakikalık sürelerle yaptıkları çalışmada dahi fare hareketlerinin farklı davranışsal veriler sunduğu belirtilmiştir. Çalışmanın veri setlerini toplamak için basit bir çalışma ortamı kurulup her seferinde 1 haftalık geçici kullanıcılara bir dizi yazma görevi ve standart bir bilgisayar verilmiştir. Bilgisayar üzerinde fare tıklama harekletlerini alan, klavye basma hızını hesaplayan paket program izleme yazılımları kurulmuştur. 12 hafta boyunca blog makaleleri araştırıp yazmaları için 80 geçici kullanıcı tutulmuştur. Ayrıntılı analizlerde fare eğrilik açısı ve mesafesi, tuş vuruş sayısı, aralığı ve bekleme süreleri, web sitesi ziyaret sıklığı gibi stili ortaya çıkacak özelliklere odaklanılmıştır. Kullanıcılardan sadece klavye ve fare verisi alınmamış aynı zamanda psikometrik sınıflarını da ölçmek için bir takım demografik anketler yapılmıştır. Bu anketler ile cinsiyet, eğitim durumu, ana dili, yaş ve baskın el gibi bilgiler alınmıştır. Ek olarak bazı veri setlerini toplamak için klavye ve fare üzerine sensörler yerleştirilmiştir. Çalışmada yapılan analizlerin doğruluğu Şekil 2.1. ile gösterilmiştir.

Task	Classes	Features	No. of subjects	Analysis	Data size	Accuracy (%)
Identification	Subject ID	Writeprints feature set	79	One vs. one sequential-minimal-optimization support vector machine 10-fold cross validation	500 characters	Exact: 55.30 In top 3: 70.30
					1,000 characters	Exact: 63.98 In top 3: 79.30
		Character bigrams	60	One vs. all, L_1 distance Leave-one-out cross validation	One day	Exact: 19.60 In top 3: 85.30
Myers-Briggs Personality Inventory (MBTI)	Extrovert/ Introvert	Character bigrams	60	One vs. all, Jaccard distance Leave-one-out cross validation	One day	78.4
	Sensing/ Intuition					73.0
	Thinking/ Feeling					76.5
	Judging/ Perceiving					80.1
Gender	Male/ female	100 most frequent words	60	One vs. all, Jaccard distance Leave-one-out cross validation	One day	76.6
Dominant hand	Right/left	Character bigrams (no case unification)	60	One vs. all, Jaccard distance Leave-one-out cross validation	One day	99.2
	Right/left/ ambidextrous					96.2

Şekil 2.1. Aktif stilometrik analizlerin doğruluğu

Mondal ve Bours (2017) yazarlarının, tuş vuruşu ve mouse biyometrisinin bir kombinasyonunu kullanarak sürekli kimlik doğrulama üzerine yaptıkları "A study on continuous authentication using a combination of keystroke and mouse biometrics" isimli çalışma hem geliştirdiğimiz projeye yakın olduğu için hem de diğer çalışmalara göre daha kapsamlı olduğu için referans alınmış, yazarları ile iletişime geçilmiş ve model oluşturulmuştur. Bu çalışmada diğer çalışmalardan farklı olarak sürekli kimlik doğrulama sistemi için araştırma yapılmıştır. Sağlam bir dinamik güven modeli algoritması sundukları, yeni bir performans raporlama tekniği ile katkı da bulundukları belirtilmiştir.

Makale ilk başta doğrulama yöntemlerinden bahsetmekte ve bu yöntemleri iki tipe ayırmaktadır.

Static Authentication (Statik Doğrulama-SA): Bu tip yöntemlerde kullanıcı başarılı bir şekilde oturumu açtıysa kullanıcı sistemin içine alınmakta ve oturum süresi dolana kadar sistemlere erişimi sağlanmaktadır.

Continuous Authentication (Devamlı Doğrulama-CA): Bu tip yöntemlerde kullanıcı sisteme başarılı bir şekilde giriş yaptıktan sonra ya belli zaman

aralığında ya da dinamik olarak kullanıcının kimliğini kontrol etmektedir. Kullanıcı tespit edilmediği hallerde oturum sonlandırılmaktadır. Bunlara dayanarak hem daha güvenilir olması hem de tuş verilerini kullanıcıyı rahatsız etmeden kontrol etmek kolay olduğu için devamlı doğrulama tipi uygulanmaya karar verilmiştir

Ülkemizdeki uzun çalışma saatleri çalışanlarda bir süre sonra yorgunluk yaratabildiğinden, bu da dolaylı olarak klavye kullanımını etkileyebileceğinden false positive (hatalı onaylanmış) alarm oluşmaması için çalışmamızda 1. yöntem amaçlanmış fakat başarılı sonuçlarından dolayı 2. yöntem teknikleri üzerinde durulmuştur.

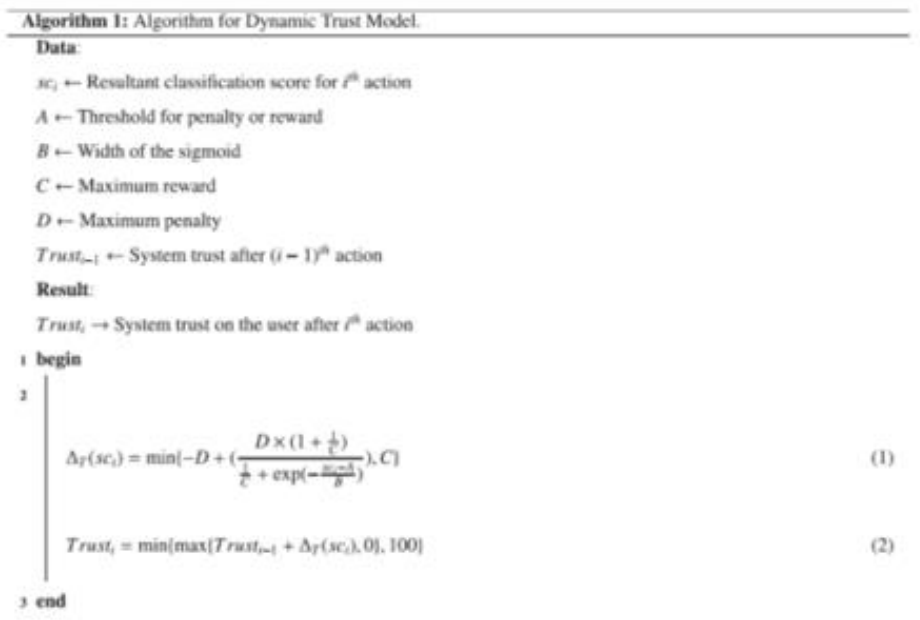
CA tip yöntemler periyodik zamanlarda kullanıcıyı kontrol etmektedir fakat ilgili periyodik zaman dilimi içerisinde başka bir kullanıcının (veya saldırgan) girip çıktığı tespit edilememektedir. Bu yüzden makalede (Periodic Authentication - PA) değil Dinamik CA kullanılmaya karar verilmiştir.

PA tip yöntemlerinin başarısını ölçmek için kullanılan parametreler (Raul vd, 2020):
- FMR: False Match Rate (Yanlış eşleşme oranı)
- FAR: False Acceptance Rate (Yanlış kabul oranı)
- FNMR: False Non Match Rate (Yanlış eşleşmeme oranı)
- FRR: False Rejection Rate (Yanlış reddetme oranı)
- EER: Equal Error Rate (Eşit hata oranı)

Dinamik CA tip yöntemlerinin başarısını ölçmek için makalede önerilen parameterler:
- ANIA: Average Number of Imposter Actıons (Ortalama taklit işlem sayısı).
- ANGA: Average Number of Genuine Actions (Ortalama gerçek işlem sayısı).

Keystroke Dynamics Sistemin Çalışma Şekli:
Güven Modeli (Trust Modeli): Dinamik CA gerçekleştirmek için dinamik ve esnek bir kontrol sistemine ihtiyaç vardır. Makalede önerilen sistem bu şekilde

çalışmaktadır. Kullanıcı verilerini alıp öğrendikten sonra bir kullanıcı profili oluşturulmakta ve doğrulama sisteme devreye girmektedir. Her bir tuş verisi (Action) geldikten sonra gelen sayı profildeki sayılarla karşılaştırılmaktadır. Sayılar benzediği durumda artı puan verilmekte (Reward) ve genel kullanıcı puanına eklenmektedir. Sayılar benzemediği durumda eksi puan vermekte (Penalty) ve genel kullanıcı puanından çıkartılmaktadır. Şekil 2.2'de dinamik güven modeli formülü makaleden alıntılanarak gösterilmiştir.



Algorithm 1: Algorithm for Dynamic Trust Model.

Data:

$sc_i \leftarrow$ Resultant classification score for i^{th} action

$A \leftarrow$ Threshold for penalty or reward

$B \leftarrow$ Width of the sigmoid

$C \leftarrow$ Maximum reward

$D \leftarrow$ Maximum penalty

$Trust_{i-1} \leftarrow$ System trust after $(i-1)^{th}$ action

Result:

$Trust_i \rightarrow$ System trust on the user after i^{th} action

1 **begin**

2

$$\Delta_T(sc_i) = \min[-D + (\frac{D \times (1 + \frac{1}{C})}{\frac{1}{C} + \exp(-\frac{sc_i - A}{B})}), C] \quad (1)$$

$$Trust_i = \min[\max[Trust_{i-1} + \Delta_T(sc_i), 0], 100] \quad (2)$$

3 **end**

Şekil 2.2. DTM (Dynamic Trust Model) formülü(Mondal ve Bours, 2017)

Tek bir formül ile hem Reward hem de Penalty puanları hesaplanabilmektedir. Bu modeli kullanarak 49 kullanıcı verisinden alınan ön sonuçlar Çizelge 2.1.'de gösterilmiştir.

Çizelge 2.1. CA sistemi için performans raporlama örneği (Mondal ve Bours, 2017)

Category(Kategori)	Users (Kullanıcılar)	ANGA	ANIA
+/+	41		99
+/-	4		807
-/+	3	4630	164

Keystroke Dynamics (Tuş Dinamikleri-KD) üzerinde toplanan veriler: Hold Time ve Seek Time

1- (Bup - Aup)

2- (Bup - Adown)

3- (Bdown - Aup)

4- (Bdown – Adowna)

Referans çalışmada modeli eğitmek için 4 yöntem önerilmiştir (Mondal ve Bours, 2017);

VP-1 internal (iç): Model, her kullanıcı için kurumdaki bütün kullanıcıların verilerini imposter (taklitçi) olarak eğitmektedir.

VP-2 mixed (karışık): Model, her kullanıcı için kurumdaki bazı kullanıcıların verilerine benzer olarak eğitmektedir. Doğrulama aşamasında ise hem daha önce karşılaştığı hem de hiç karşılaşmadığı veriler ile test etmektedir.

VP-3 external (dış): Model, her kullanıcı için kurumdan bağımsız bir kullanıcı datasetinin (veri seti) verilerini taklitçi olarak eğitmektedir. Doğrulama aşamasında daha önce hiç karşılaşmadığı veriler ile test etmektedir.

VP-4: KD için VP-3 ve MD VP-1 kullanılmaktadır.

Modeli oluşturmak için önerilen algoritmalar:

ANN (DL)

CPANN (DL)

SVM (ML)

Sonuçları değerlendirmek için önerilen yöntem:

Model, her kullanıcı için bütün kullanıcı (Imposter olarak) verileri ile eğitilmekte ve test sonuçlarına göre performansı 4 grup üzerinden değerlendirmektedir.

+/+) FNMR = 0, FMR = 0

+/-) FNMR = 0, FMR > 0

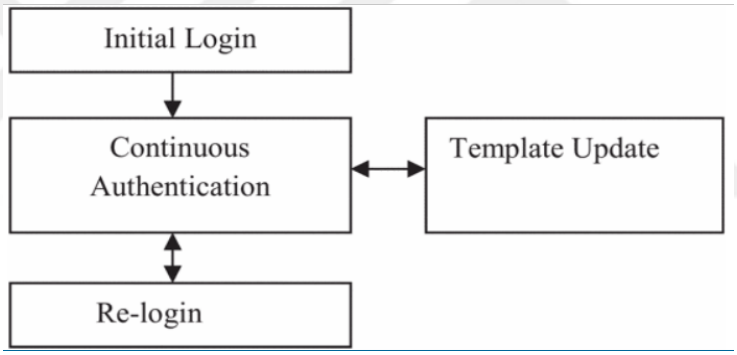
-/+) FNMR > 0, FMR = 0

-/-) FNMR > 0, FMR > 0

Ve gruplardaki kullanıcı sayısına göre de modelin başarısı ölçülmektedir.

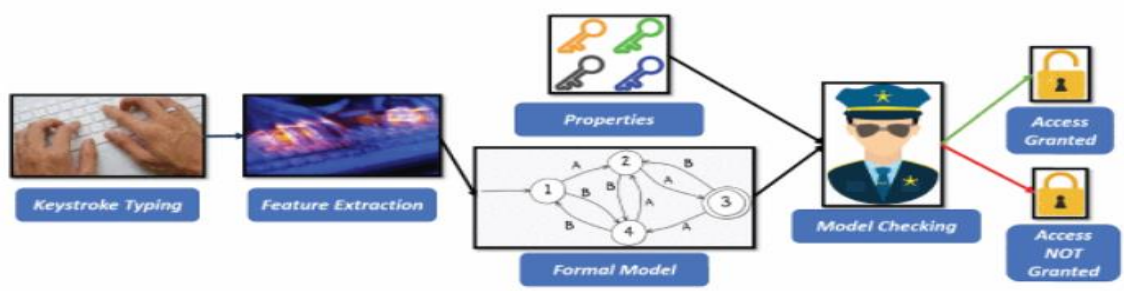
Singh (2018) oturum sırasında kullanıcıyı sürekli izleyen tuş vuruşuna dayalı bir kimlik doğrulama yöntemi üzerine çalışma yapmıştır. Bu çalışma ile oturumu başlatan ve sonlandıran kişinin aynı kişi olup olmadığı kontrol edilmektedir.

Oturum sırasında kullanıcı verilerini toplamak ve aynı kullanıcı olduğunu doğrulamak için bir matris kullanılmaktadır. Yapılan çalışmanın perfomansını ölçmek için Strokes to False Reject ve Stroke to False Accept isminde kendi geliştirdiği metrikleri kullanmaktadır. Çalışmanın diğer çalışmalardan farklı olduğunu öne sürdüğü nokta ise bir ön kayıt gerektirmemesidir. Kimlik doğrulama süreci eğitim ve test aşamalarını içerir. Eğitim aşamasında ise taslağı oluşturmak için kayıt yapılır. Doğrulama döngüsü ise kullanıcı gerçekliğini kanıtlar. Sürekli doğrulama için sayıma dayalı ve dinamik bir performans ölçümü kullanılmaktadır. Doğrulama şablonu olarak kullanılan 26'lık matriste kullanıcının gireceği ve matriste karşılığı olan anahtar çift harfler (örneğin aa, zz, zy gibi) ortama tuş basma hızını tutar. Kullanıcıyı rahatsız etmeden sürekli doğrulama yapan çalışma algoritması makaleden alıntılanarak Şekil 2.3'de gösterilmiştir.



Şekil 2.3. Soft biyometrik sürekli kimlik doğrulama

Di Tommaso vd. (2019) bir dizi tuş vuruşu özelliği kullanarak farklı kullanıcılar arasında ayrım yapmak için mantıksal zamansal özelliklerle model kontrolünü kullanan çalışma yapmışlardır. Destek vektör makineleri algoritması, tanımlama ve doğrulama görevlerinde iyi sonuçlar göstermiştir. Çalışmada izlenen sistemlere yazı yazan kullanıcılar takip edilmiştir. Kullanıcı tanımlamasını etkili bir şekilde sağlamak için özellik vektörü yaratılmıştır. Bu özellik vektörü ile tuş vuruş dinamiklerini kullanarak kullanıcılar arasındaki farkı göstermeye dayalı bir çalışma yapılmıştır. Makalede önerilen yaklaşım Şekil 2.4'de gösterilmiştir.



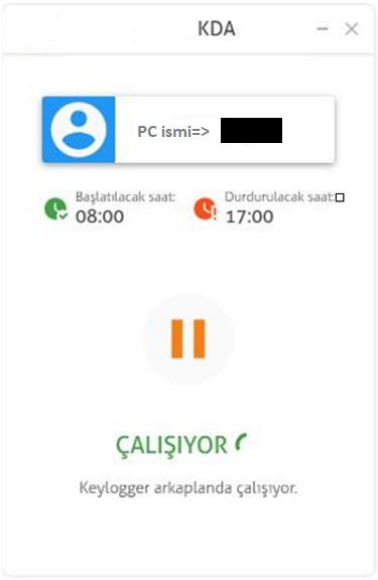
Şekil 2.4. Kullanıcı kimlik doğrulaması için önerilen yaklaşım

Raul vd. (2020) Tuş vuruş dinamiği tabanlı kimlik doğrulama mekanizması inceleyen bir çalışma yapmışlardır. Çalışmada, bir grup kullanıcının yazma stiline dayalı biyometrik verileri olduğu ve daktilo yazarlarının, kullanıcının gerçekliğini doğrulamak için analiz edilebilecek benzersiz yazma kalıpları olduğu görülmüş ve bu sonuçlardan kullanıcının gerçekliğini doğrulamak için yararlanılabileceğini öne sürülmüştür. Yazarlar, tuş vuruşu dinamiklerinin, çoğunlukla bir kullanıcının gerçekliğinin son derece güvenle tespit edilmesi gereken durumlarda uygulanabileceğine dair çalışmalarını paylaşmışlardır. Parola ihlali olsa dahi gerçek kullanıcının tuş vuruş dinamikleri bilinmeden herhangi bir işlem gerçekleştirilemeyeceği belirtilmiştir. İstatiksel bilgiler ile çalışmada elde edilen verilerin analizleri paylaşılmıştır. Farklı klavye türleri ve uzaktan erişimlerin çalışmada sorun olduğu paylaşılmıştır. Ek olarak, yapılan çalışma neticesinde tüm temel özelliklere sahip vuruş dinamiği veri setlerinin daha güçlü olması gerektiği bunun yine çalışmada sorun yarattığı ifade edilmiştir.

3. ÇALIŞMADA KULLANILAN YÖNTEM VE METOTLAR

Literatür taramalarından sonra aşağıdaki aşamalar kullanılarak çalışmanın genel çerçevesi belirlenmiştir.

Veri Toplama: Makine öğrenmesinin yakıtı veridir. Bu sebeple ilk aşamada çalışmanın ana amacına ulaşmak için ilgili araştırma grubundan veri toplanması ve alışkanlıklarının tespit edilebilmesi için veri setlerinin oluşturulması gerekmektedir. Daha sonra toplanan veriler temizlenip modele uygun hale getirilmelidir. Özellikle bilimsel araştırmalarda birden fazla gruptan veri toplanması gerektiği için bu veriler farklı kategorilerde ve organize olmayan biçimlerde alınabilir. Farklı zaman periyotlarında alınmamış, düzenlenmemiş ve temizlenmemiş veri grupları makine öğrenmesi için oluşturulacak modelde hata verecektir. Çalışmadaki ana hedefimiz klavye kullanım durumundan gerçek kişiyi tespit etmek olduğundan kullanıcıların nasıl klavye kullandığını analiz edeceğimiz verilere ihtiyaçımız bulunmaktadır. Hangi veriden hangi sonucun alınması gerektiği literatür araştırmaları sırasında ortaya çıkmıştır. Bu veri setlerini oluşturmak için C# tabanlı bir keylogger (tuş kaydedici) program (KDA agent) geliştirilmiştir. Geliştirilen programın çalışma akış diyagramı Şekil 3.16'da verilmiştir. Daha sonra geliştirilen bu program kullanıcılara çalışma amacı anlatılarak, bilgileri dahilinde bilgisayarlarına kurulmuştur. Program bilgisayara yerleştirildikten sonra modelde kullanılması plananan farklı tip veriler (Tuş basma zamanı ve sıklığı, tuşlar arasında ki geçiş zamanı ve sıklığı, kullanılan programlar ve kullanma süreleri) toplanmıştır. Program topladığı verileri bilgisayarda saatlik olarak gün sonunda gün olarak dosyalamaktadır. Kullanıcılarda 3 ay kalan programdan alınan ham verilerin modelde kullanılacak veri setlerine dönüştürülmesi amacıyla ana veri tabanına (bu çalışma kapsamında geliştirici PC) bir Application Programming Interface (Uygulama Programlama Arayüzü-API) üzerinden kaydedilmiştir. Kullanıcılarda kurulu olan KDA agent isimli keylogger programın arayüz görüntüsü Şekil 3.1'de verilmiştir.



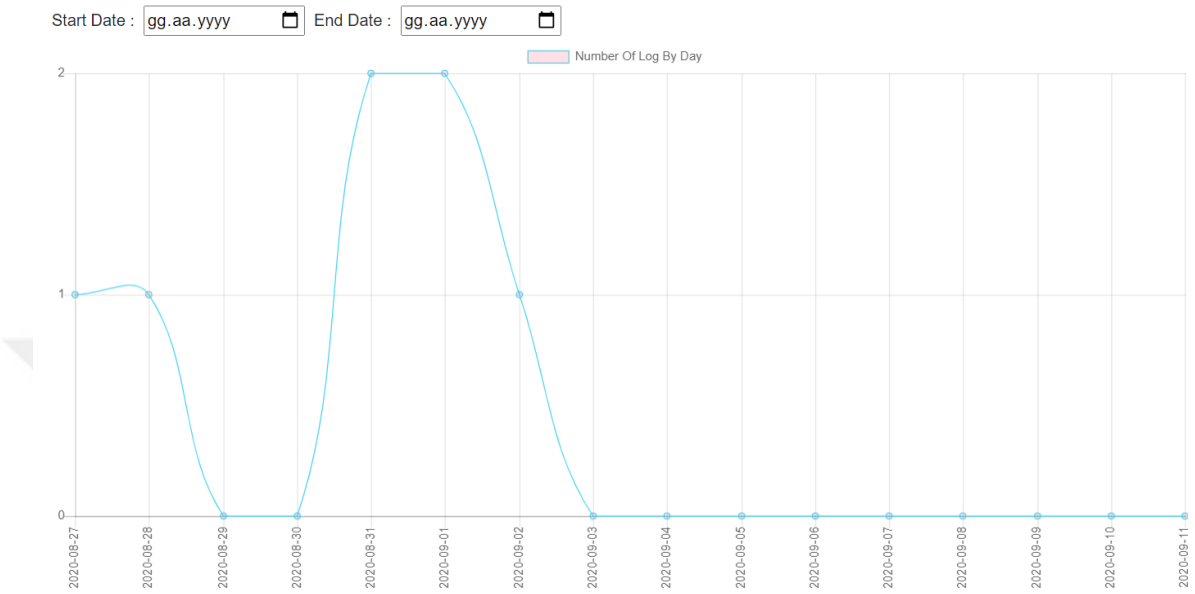
Şekil 3.1. Kullanıcıda kurulu KDA agent program.

Toplanan ham verilerin kullanılabilir veri setlerine dönüştürülmesi sürecini takip ve kontrol etmek amacıyla bir web uygulaması geliştirilmiştir. Bu uygulama sayesinde kullanıcılar, program tarafından toplanan verilerin saatlik olarak oluşturulup kaydedildiği dosyaları, program logları vb. tek bir arayüzden okunmasını kolay bir şekilde görüntüleyebileceklerdir. Aynı zamanda veri toplama istatistikleri ve oluşacak hatalar izlenebilir hale gelmiştir. Çalışmanın ileride ürünleşmesi durumunda bu web uygulama yöneticilere büyük kolaylık sağlayacaktır. Web uygulamasına ait anasayfa sekmeleri Şekil 3.2'de gösterilmiştir.

Şekil 3.2.WEB anasayfa sekmeler

Şekil 3.3' de bulunan grafik günlük log sayılarını göstermektedir. Uygulama ilk açıldığında başlangıç tarihi olarak o günün tarihini ve bitiş olarak da 15 gün öncesini alarak grafiği oluşturur. Tarih düzenlemelerin yapıldığı ekran Şekil

3.4'te gösterilmiştir. İstenilen tarih aralıklarına göre grafik görüntülenebilir. Loglar sayesinde programın herhangi bir hatası olup olmadığının analizi yapılır. Çalışmanın başlangıcında programın kendi kendini durduğu loglar sayesinde görülmüş ve hata giderimi yapılmıştır.



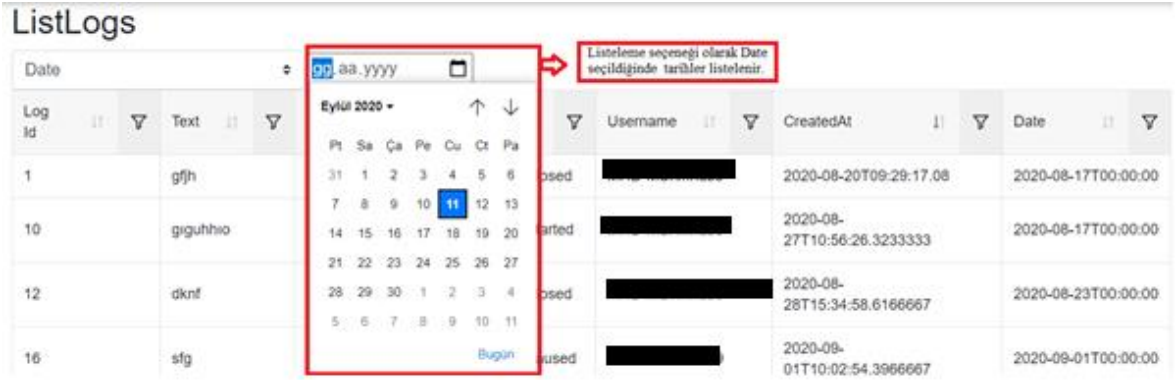
Şekil 3.3. Log sayısı grafiği

Şekil 3.4. Başlangıç/Bitiş tarihi

Yüklenen loglar ile ilgili bilgilerin listelendiği sayfa Şekil 3.5'te gösterilmiştir.

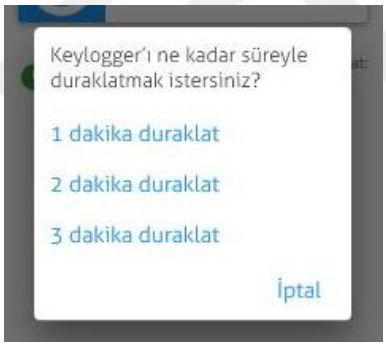
Şekil 3.5 Log list sayfası

Şekil 3.6' da listeleme seçeneği tarih seçildiğinde diğer kutucukta tarihler listelenir. Listede bulunan tarihlerden biri seçildiğinde o zaman ait loglar listelenir.



Şekil 3.6. Tarihe göre listeleme

Kullanıcılara güven vermek ve mahremiyet alanı yaratmak amacıyla Şekil 3.7.'de gösterildiği gibi programı durdurma hakkı verilmiştir. Fakat bu durumun verilerin sağlıklı bir şekilde toplanmasına engel olmaması için ne kadar sıklıkla durdurulduğunun izlenmesi, fazla sayıda durdurma yapıldığında ise sebebinin öğrenilmesi ve buna uygun çözümün kullanıcıya sunulması gerekmektedir.

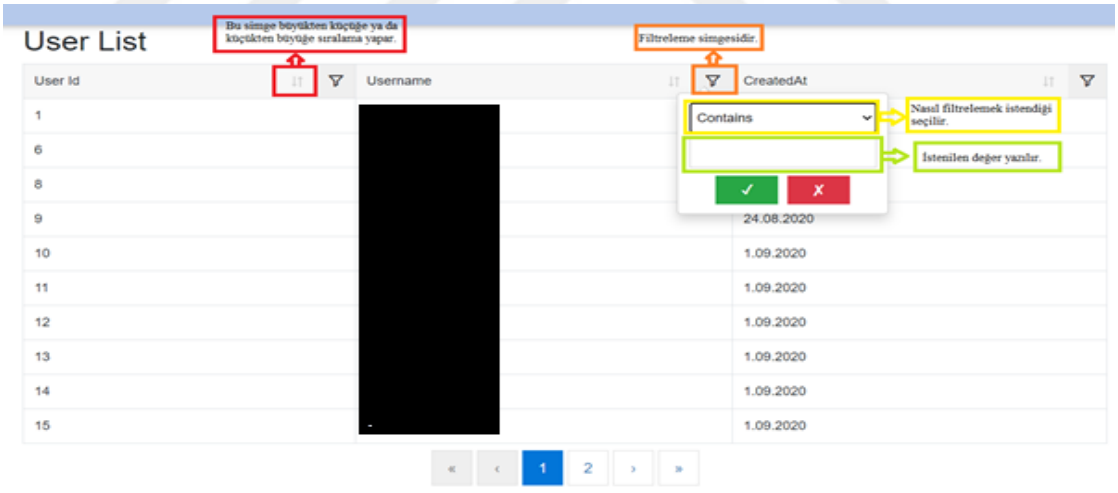


Şekil 3.7. KDA agent programını durdurma

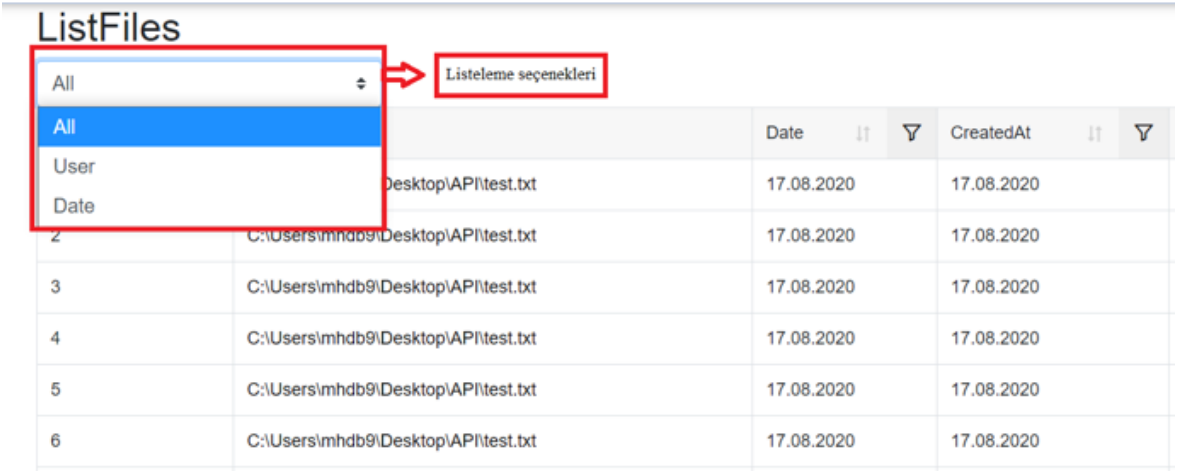
Şekil 3.8' te bulunan grafik uygulamayı en çok durduran (pause yapan) ilk 6 kullanıcıyı göstermektedir. Uygulama ilk açıldığında başlangıç tarihi olarak o günün tarihini ve bitiş olarak da 15 gün öncesini alarak grafiği gösterir. İstenilen tarih aralıklarına göre grafik görüntülenebilir.

Şekil 3.8. En çok puase yapan kullanıcılar grafiği

Programın kurulu olduğu kullanıcıların listesi Şekil 3.9'da, yüklenen dosyalar ile ilgili bilgilerin listelendiği sayfa Şekil 3.10'da gösterilmektedir. Listede bulunan dosya yolunun üstüne tıklayınca ilgili dosyayı bilgisayara indirir.

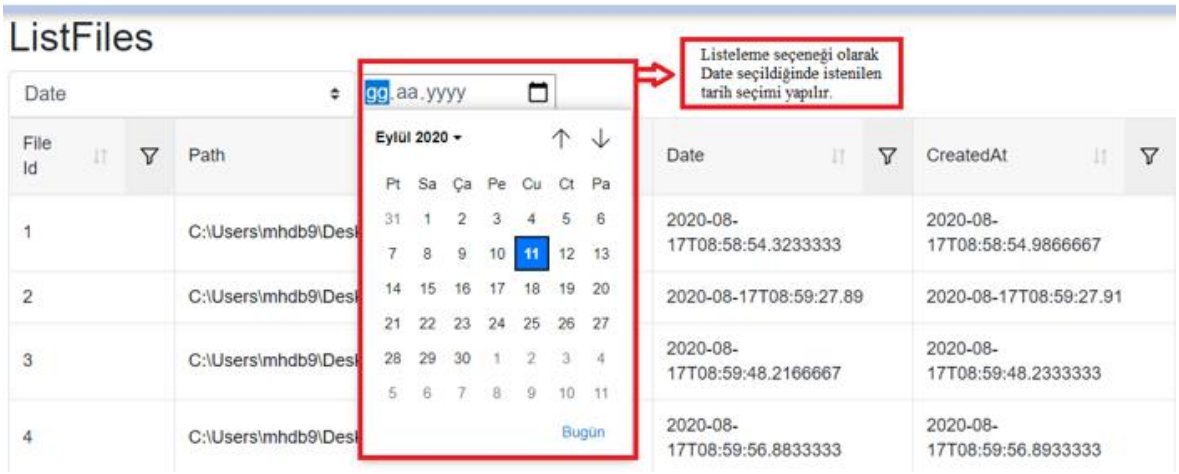


Şekil 3.9. Kullanıcı listesi



Şekil 3.10. Dosya listeleme sayfası

Şekil 3.10'da kullanıcıdan alınan dosyaların Şekil 3.11'de tarih bazlı olarak sıralanmasını sağlayan sayfaların görüntüsü gösterilmektedir.



Şekil 1.11. Tarihe göre dosya listeleme sayfası

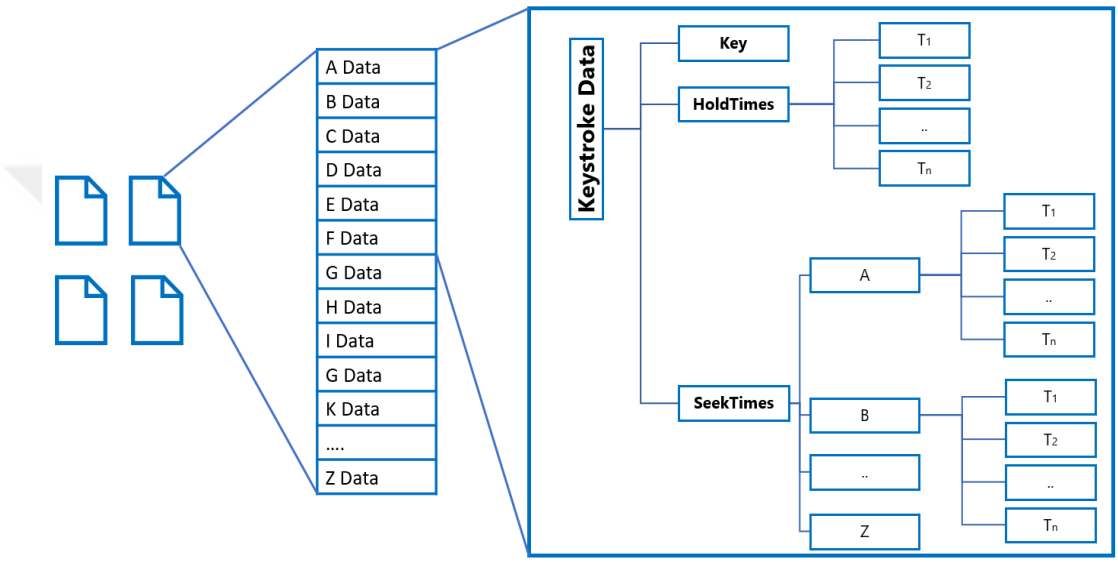
Veri Analizi ve Model Oluşturma: İlk aşamada ham veriler toplandıktan sonra makine öğrenmesi modellerini kurma aşamasında ki olmaz olmaz adımlardan biri de veriyi temizlemektir. Ham veride bulunan hataları giderme, tekrarları silme işlemi veriyi analize hazırlar. Kirli veri, yanlış veri analitiğine götürür, yanlış analiz de yanlış kararlar aldırır. Bu sebeple çalışmamızın en büyük zamanı, veri toplama, veri doğrulama, veri temizleme ve analiz görselleştirme işlemlerine harcanmıştır. Sistematik olmayan dağınık verilerde, verilerin yanlış yazımı, bozulması, çoğaltılması gibi yanlış değerlendirme sorunları vardır. Hatalı veriye örnek verecek olursak cinsiyeti erkek olan birinin daha önce doğum yapmış olma

bilgisine evet yazılması vb. gibidir. Daha tehlikeli olansa verdiğimiz örnek gibi bariz hataların ortaya çıkmaması ve ilk bakışta fark edilmeyecek hatalar sebebiyle analiz sonucunda önemli kararların verilecek olmasıdır. Veri temizleme işlemi yaparken bazı basit ama etkili yöntemler bulunmaktadır. Elinizdeki çalışmaya benzer daha önce eğitilmiş bir model var ise bu modeli kullanarak eksik değerler doldurulabilir. Toplanan verilerin normalini tanımlamak ve anormal verilerin değerini belirlemek için özet istatikler kullanılabilir. Aynı değerde olan veri gruplarının belirlenip kaldırılması, kaldırılması veri bütününü bozuyor ise bu değerlerin varyansının 0 olarak yazılması iyi bir yöntem olabilir. Tekrar eden veriler modelini bozduğu kadar birde aykırı veya uç gözlem verileri modeli olumsuz etkileyebilir. Normalin dışında olan ve farkılılığı ile bütün analizi etkileyen veriler ciddi problemler oluşturabilir. Uç değerlerin olmasının sebebi giriş hataları, veri bozulması, gerçek uç gözlemler olabilir. Aykırı verileri birkaç farklı yolla tespit edebiliriz. Çalıştığımız veri grubunun genel özelliklerini bilerek tespit yapmak basit ama etkili bir yöntemdir. Örneğin 11 haneli TCKN'de 10 haneli olarak gözlenen bir veri var ise aykırı veridir diyerek modelden çıkarılabilir. Amacımız genellenebilir özellikleri temsil eden güçlü yapılı ve yansız modeller oluşturmaktır. Diğer bir yöntem standart sapmayı hesaplayarak kontrol mekanizması oluşturmaktır. Verilerden alınan ortalama değerin üzerinde standart sapma eklenerek sınır değer ortaya çıkarılır. Bu sınırın altında veya üstünde kalan değerler uç veya aykırı değer olarak düşünülebilir. En çok kullanılan yöntem ise Boxplot yönetimidir. Değişken veri grubu küçükten büyüğe doğru sıralanır. 4 bölüme ayrılır. 1. yüzdelik dilim ve 3. yüzdelik dilim değerlerine karşılık gelen bir değer hesaplanır ve bu sınır değere göre uç veri tanımlanır.

Algoritmalar teorikte ne kadar iyi anlaşılırsa anlaşılsın pratikte hangi soruna hangi algoritmanın kullanılacağı tam olarak planlanamaz. Genellikle tahmine dayalı modelleme projesi planlanır ve veriler üzerinde algoritmaların denenmesi ve sonucunda en iyi başarıyı sunan seçilir.

Yukarıda anlatılanlar ışığında çalışmada Şekil 3.12'de gösterilen veri yapısında veri setleri oluşturulmuştur. Veri setleri oluştururulmadan önce toplanan ham

verilerden çıkan kirli veriler temizlenip (hata giderme, tekrarları silme vb.) analiz edilmeye başlanmıştır. Analiz ederken bazı teknikleri kullanıp (veri görselleştirme, raporlar çıkarma) öngörülemeyen hatalar veya sonuçların tespiti yapılmıştır. Ardından çıkan sonuçlara göre datasetler tekrar düzenlenmiş ve derin makine öğrenmesi kullanılarak bir model oluşturulmuştur. Birkaç derin öğrenme algoritmalarında denemeler yapılmıştır. Geliştirilen model ile yeni veriler test edilmiş ve algoritmaların başarı oranları ölçülmüştür.



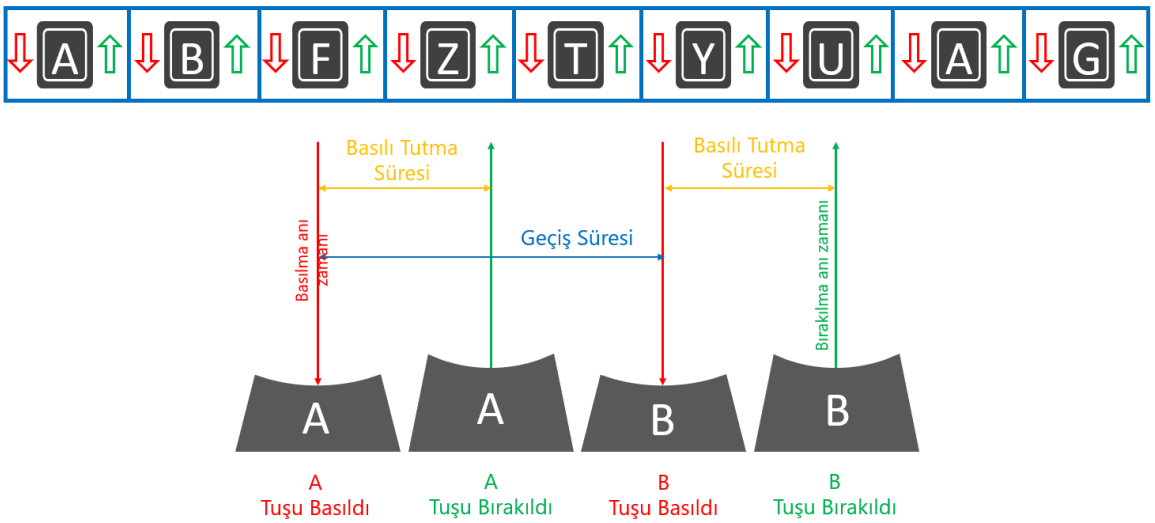
Şekil 3.12. Veri seti yapısı.

Modeli Kullanacak Yazılımı Geliştirme: Model ortaya çıktıktan sonra kullanıcılarda çalışan yazılıma entegre edilmesi, yazılımın kullanıcı alışkanlarını öğrenip ikinci aşamada kullanıcı tespiti yapmaya başlaması ve sürekli bir şekilde log ve data kaydetmesi gelecek çalışma için planlanmıştır.

3.1. KDA agent (Keylogger) çalışma mantığı

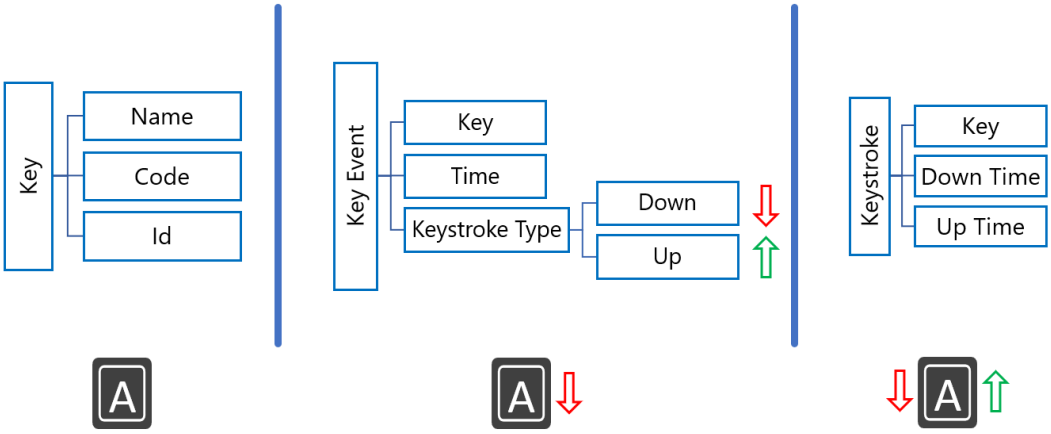
Tuş kaydedici görevi gören bu program kullanıcı bilgisayarlarına yerleştirildikten sonra; Şekil 3.13'de belirtilen biçimde tuş basma zamanı ile sıklığı, tuşlar arasında ki geçiş sıklığı ve geçiş süreleri toplanmıştır. Tuş vuruş dinamiklerinden yararlanarak kimlik doğrulamada;

- Key Up: Tuşa basma olayını,
- Key Down: Tuşu bırakma olayını ifade eder (Di Tommaso vd, 2019).



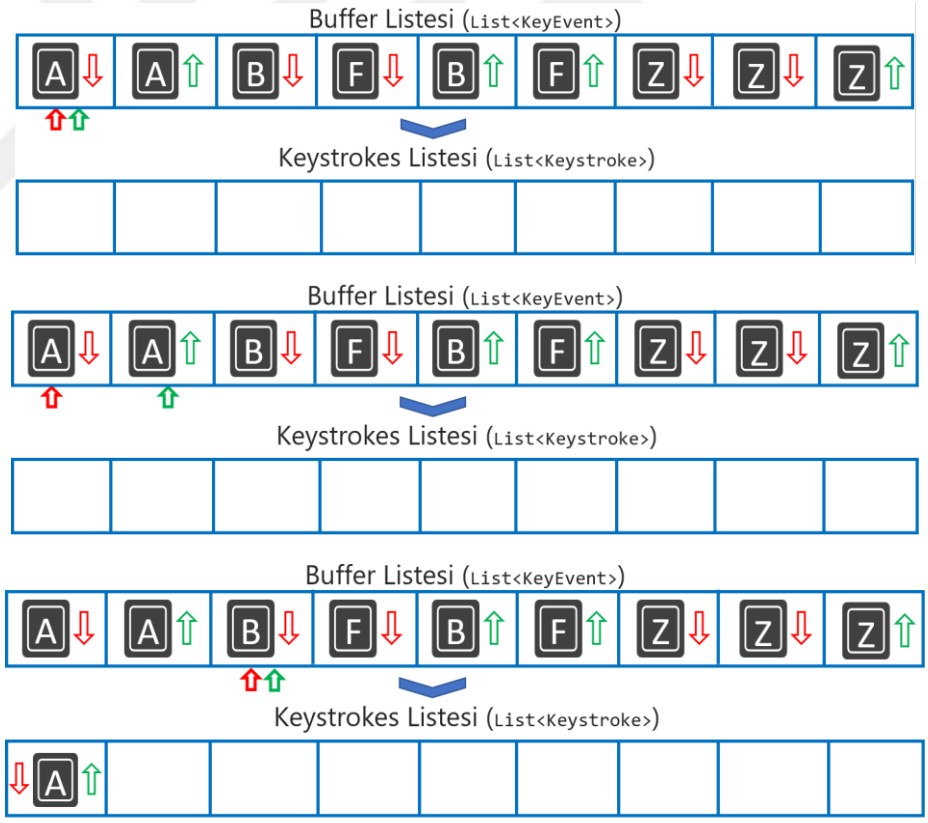
Şekil 3.13. KDA Analyzer topladığı veriler

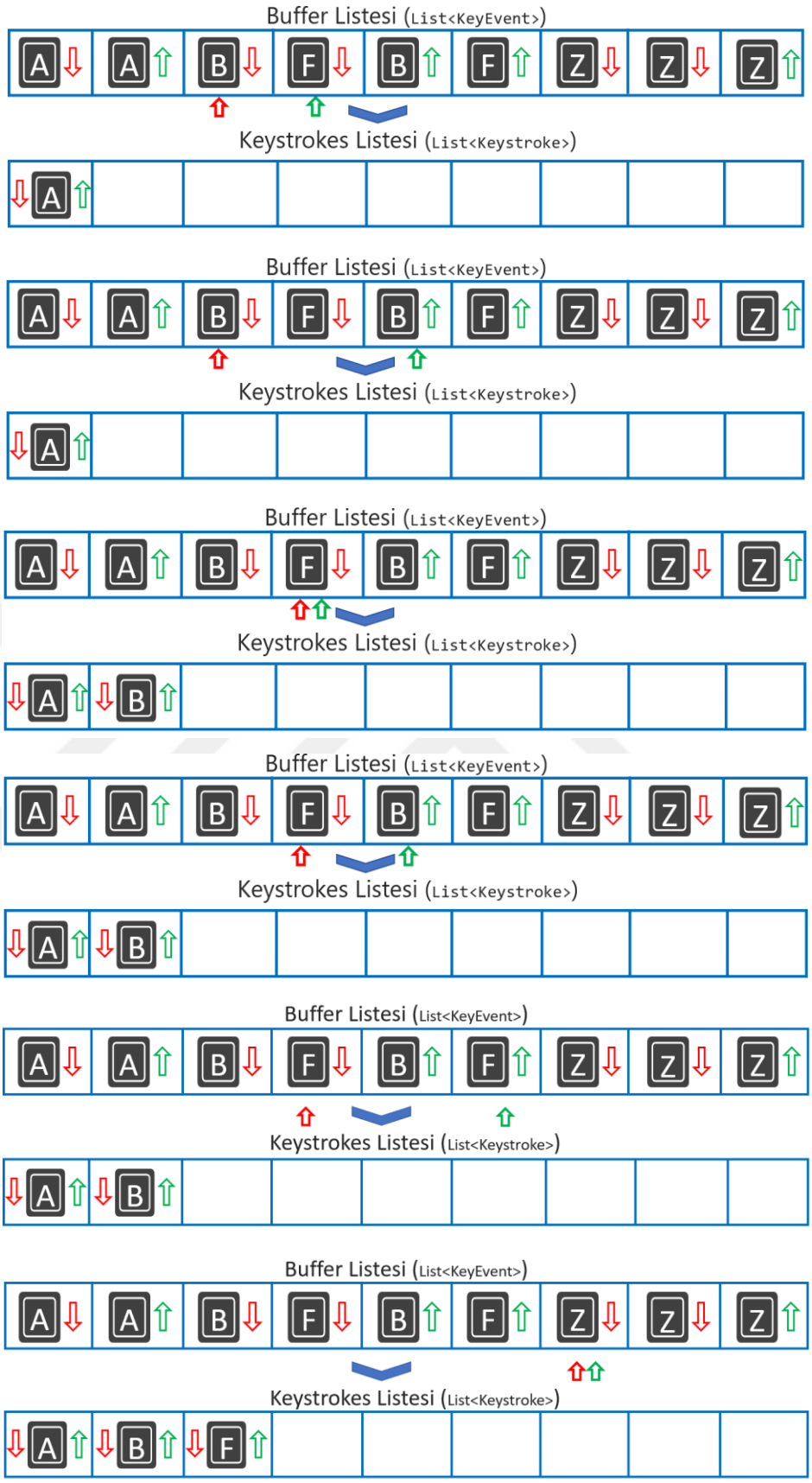
Çalışmanın ham verisi bu program sayesinde alınmaktadır. Program kullanıcılara kurulduktan sonra 3 ay süreyle izlenmiştir. Hatalar tespit edilmeye çalışılmış kullanıcı geribildirimlerine göre düzeltmeler sağlanmıştır. Program, kullanıcı bilgisayarında günlük dosyalar oluşturarak Şekil.3.14'de gösterilen verileri daha sonra veri tabanına aktarmak üzere toplamıştır. Günlük dosyalar içerisinde saatlik oturumları gösteren 23 ayrı dosya bulunmaktadır. Her saat başı yeni dosya yaratan bu uygulama sayesinde hatalar daha rahat tespit edilmiştir. Ham verilerin kontrolü ve verilerin kaydedilmesine engel bir durum olup olmadığı daha doğru bir şekilde görülmüştür. Keyloger mantığından esinlenmiş ve kodlanmış olan agent programın ana bileşenleri Şekil 3.14' de gösterilmiştir.

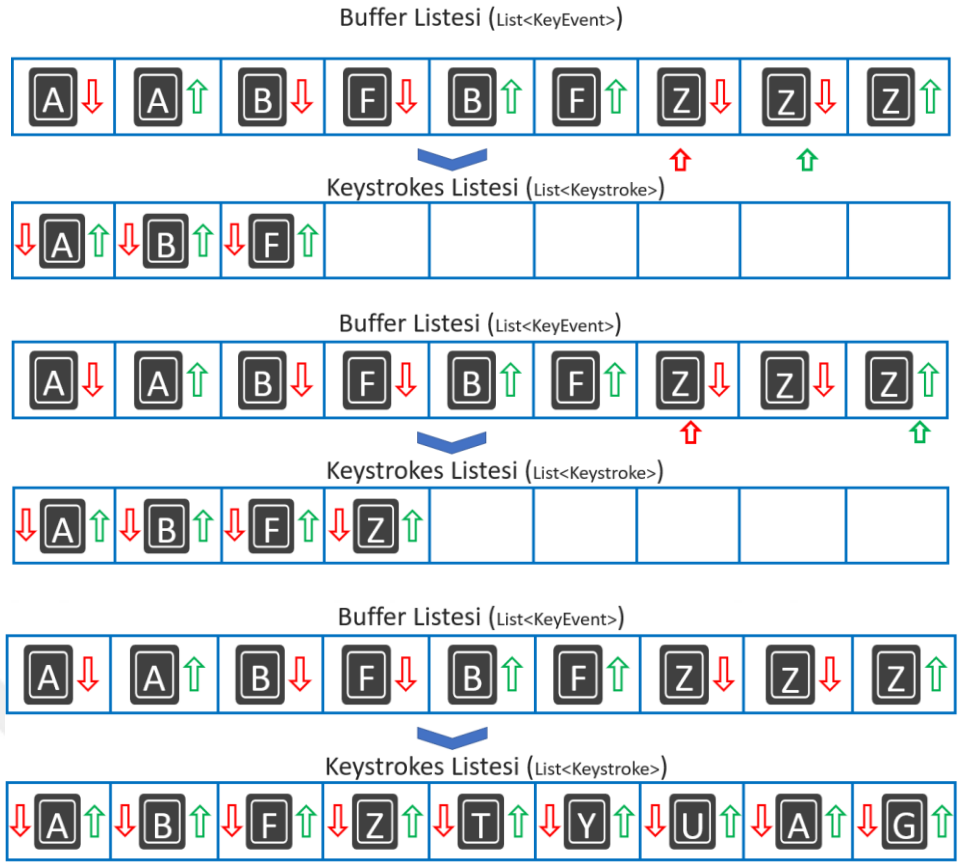


Şekil 3.14. KDA Agent ana bileşenleri

Programın çalışma mantığı Şekil 3.15' de görseller üzerinden adım adım anlatılmıştır. Kullanıcının Şekil 3.15'de gösterilen tuşlara sırasıyla bastığını ve buffer listesine kaydedildiğini varsayalım. 'A' harfine tuş vuruşu yapıp (⇩) sonra tuş vuruşundan parmağının çekildiği (⇧) anı tüm kombinasyonlar içinde sırasıyla arar, tekrar A harfini bulduğunda arada geçen zamanı vuruş hızı olarak alır. 'A' harfinden sonra basılan harf iki karakter arasında geçen süre olarak kaydedilir. A harfini Keystrokes listesine kaydettikten sonra ikinci tuş vuruşu yapılan B harfini bulur. Down olarak aldığı harfin Up olarak ne zaman bırakıldığını öğrenmek için Buffer Listte bulunan tuşlara sırası ile bakar. B harfinden sonra gelen F harfi yeni bir tıklanmadır. B harfinden henüz parmak çekilmediğinden aramaya devam eder. Bulduktan sonra arada geçen zamanı kaydeder ve Keystrokes listesine B harfini yazar. Listedeki tüm tuşları aynı mantık ile yeni listeye (Keystrokes) dizer.

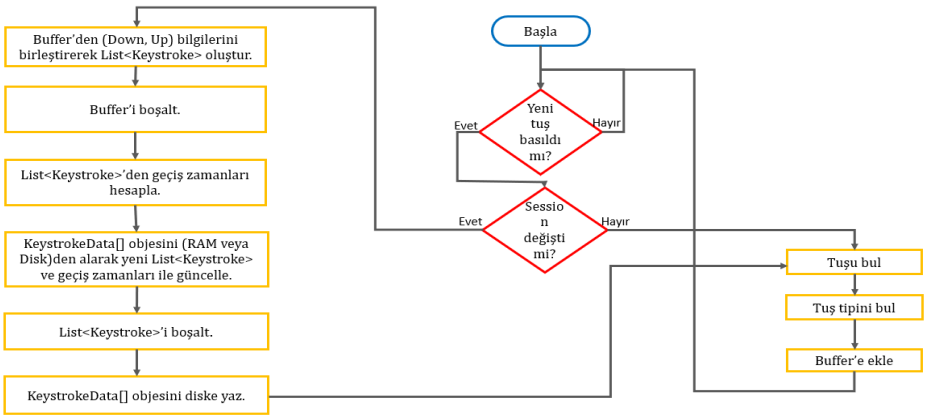



Buffer Listesi (List<KeyEvent>)
A A B F B F Z Z Z
Keystrokes Listesi (List<Keystroke>)
A
Buffer Listesi (List<KeyEvent>)
A A B F B F Z Z Z
Keystrokes Listesi (List<Keystroke>)
A
Buffer Listesi (List<KeyEvent>)
A A B F B F Z Z Z
Keystrokes Listesi (List<Keystroke>)
A B
Buffer Listesi (List<KeyEvent>)
A A B F B F Z Z Z
Keystrokes Listesi (List<Keystroke>)
A B
Buffer Listesi (List<KeyEvent>)
A A B F B F Z Z Z
Keystrokes Listesi (List<Keystroke>)
A B
Buffer Listesi (List<KeyEvent>)
A A B F B F Z Z Z
Keystrokes Listesi (List<Keystroke>)
A B F

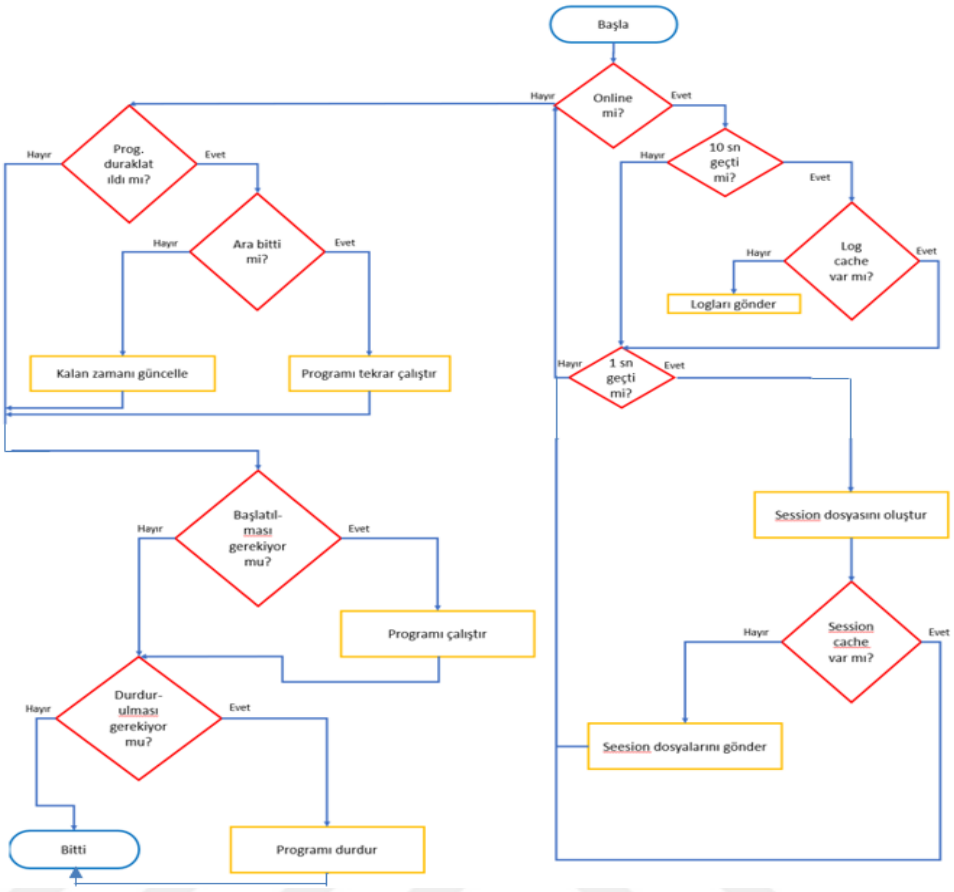


Şekil 3.15. KDA Agent çalışma mantığı

Şekil 3.15'de anlatılan adımlara ait program çalışma akış diyagramı Şekil 3.16'da, kullanıcı ile programın entegrasyonunu ve çalışma otomasyonun nasıl olduğunu gösteren analyzer otomasyonu çalışma akış diyagramı Şekil 3.17'de gösterilmiştir.



Şekil 3.16. KDA Agent (Keylogger) çalışma akış diyagramı



Şekil 3.17. KDA Analyzer otomasyonu çalışma akış diyagramı

3.2. Çalışmada kullanılan yazılım ve programlama dilleri

Çalışmayı geliştirirken kullanılan yazılım ve programlama dilleri, okunabilirliğin kolaylaşması açısından aşağıda sıralı olarak verilmektedir.

- Veri toplayan yazılım WPF Windows uygulaması C# diliyle yazılmıştır.
- Sunucudaki yazılım için ASP.net kullanılarak hem API hem de WEB uygulaması hazırlanmıştır.
- Veri tabanı teknolojisi olarak MySQL kullanılmıştır.
- Veri görselleştirmede C#, Python ve excel kullanılmıştır.
- Derin öğrenme ile modeli oluşturmak için pyton kütüphaneleri kullanılmıştır.

3.3. Veri Tabanı Tasarımı

İlk olarak verileri depolamak için bir dosya sistemi ihtiyacı ortaya konulmuştur. Geliştirilecek sistemde verilerin hem hızlı bir şekilde yazılması hem de ne zaman yazıldığı bilgisinin gizli olması için binary dosya kullanımı tercih edilmiştir. Veriler birden fazla kullanıcıdan alındığı için incelenen verileri dosyadan çıkartmak, karşılaştırmak ve sorgulamak çok zor olduğundan yapının veri tabanında etkili bir şekilde saklanması için tasarım yapılmıştır. Kağıt üzerinde oluşturulan taslak tasarım incelendikten sonra dijital ortamda oluşturmak için MS-SQL server kullanımı tercih edilmiştir.

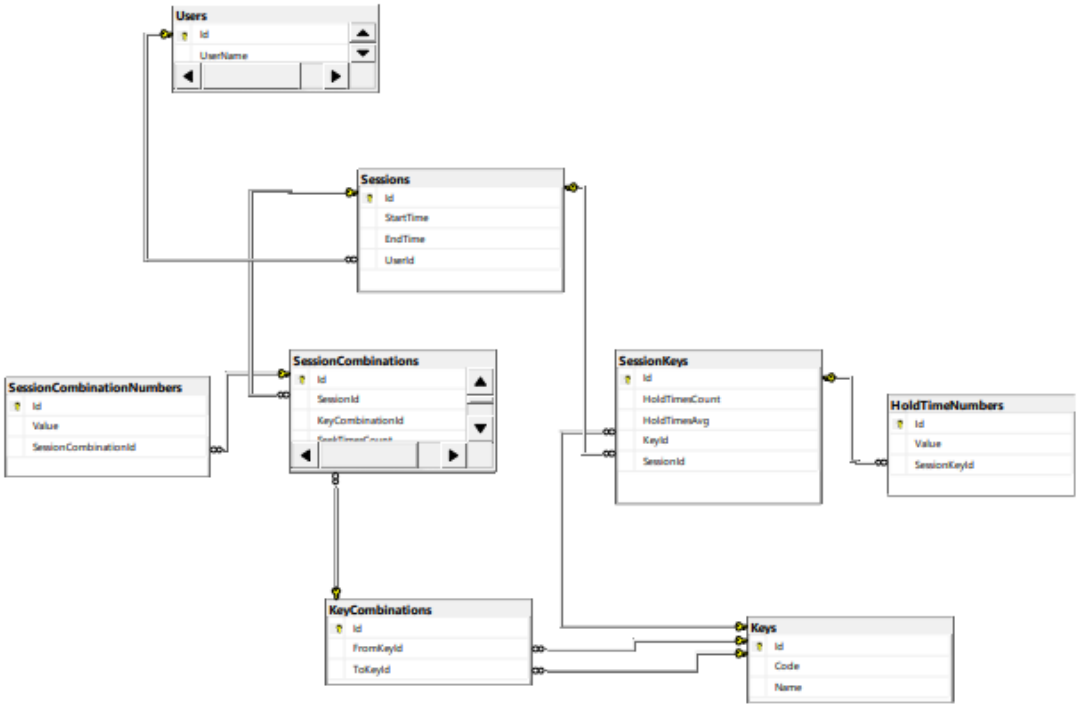
Kullanıcı bilgisayarına yüklenen program ile birlikte toplanacak farklı tipte klavye verileri Şekil 3.18'de şeması verilen veri tabanına eklenmiştir. Veriler şu başlıklar altında toplanmıştır;

Users: Araştırmaya dahil olan kişilerin atanan ID ve isim bilgileri tutulur.

Sessions: Verilerin saatlik olarak saklanması için oluşturulan session bilgileri tutulur.

Keys: Araştırmada verisi tutulan tuşların isimleri ve ID'leri tutulur.

KeyCombinations: İki tuş arasında olabilecek tüm geçiş kombinasyonları Şekil 3.19' da gösterildiği gibi From-To şeklinde tutulur.



Şekil 3.18. Veri tabanı yapısı

To

From	A	B	C	D	E	F	G	H	...	...
A	A→A	A→B	A→C	A→D	A→E	A→F	A→G	A→H	...	...
B	B→A	B→B	B→C	B→D	B→E	B→F	B→G	B→H	...	...
C	C→A	C→B	C→C	C→D	C→E	C→F	C→G	C→H	...	...
D	D→A	D→B	D→C	D→D	D→E	D→F	D→G	D→H	...	...
E	E→A	E→B	E→C	E→D	E→E	E→F	E→G	E→H	...	...
F	F→A	F→B	F→C	F→D	F→E	F→F	F→G	F→H	...	...
G	G→A	G→B	G→C	G→D	G→E	G→F	G→G	G→H	...	...
H	H→A	H→B	H→C	H→D	H→E	H→F	H→G	H→H	...	...
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...

Şekil 3.19. Tuşlar arası geçiş matrisi

SessionKeys: Bir sessionda basılan tüm tuşların verileri (ilgili tuşa basıldı veya basılmadı bilgisi) tutulur.

HoldTimesNumbers: Her SessionKey için tuşa basılan süreden, tuşu bırakma süresine kadar geçen süre bilgisi tutulur.

SessionCombinations: Bir sessionda yapılan tüm tuşlar arasındaki geçiş bilgisi tutulur.

SessionCombinationNumbers: Her SessionCombination için tüm geçiş zamanları tutulur.

3.4. Verileri veri tabanına kaydeden kodları geliştirmek

Kullanıcı bilgisayarından agent ile toplanan verilerin binary dosyalarından veri tabanına aktaran kodları ve SQL sorguları oluşturulmuştur

Birinci aşama, verileri veri tabanına kaydeden ve veri tabanından çekilmesini sağlayan verilerin saklanacağı veri yapısı modeli oluşturulmuştur.

İkinci aşama, verileri modellere aktardıktan sonra modelleri veri tabanına kaydetmek için bir C# class'ı oluşturmaktır. Bu class'ta verileri SQL veri tabanına göndermek için kullanılan dapper kütüphanesine yazılan fonksiyonların kodları yazılmıştır.

Üçüncü aşamada verileri veri tabanına girmek için SQL sorguları stored procedure olarak yazılmıştır.

3.5. Toplanan Verileri Veri Tabanına Kaydetmek

Kullanıcılardan toplanan verilerin binary dosyasından alıp veri tabanına kaydetmek için çalışma yapılmıştır. İlk başta sessionların saatlik dosyalar halinde saklanması gereken kodda bir sıralama hatası yüzünden dosyalar bir önceki session üzerine yazılmaktaydı. Bu sebeple önce sessionları saatlere göre ayıran bir fonksiyon geliştirilmiştir. Sessionlar ayrıldıktan sonra onları doğru bir şekilde

yükleyen kodlar yazılmıştır. Veriler eklendikten sonra bazı SQL sorguları yazılarak ön analizlere başlanmıştır.

3.6. Derin Öğrenme Kullanarak Kullanıcı Tespiti

Makine öğrenmesi, insanların öğrenme stillerini taklit ederek veri algoritmalarının kullanımına odaklanıp doğruluğunu artıran bir bilgisayar bilim dalıdır. Yapay zekanın ayrılmaz bir parçası olarak doğru bir öğrenme sağlar.

3.6.1. Derin öğrenmenin tarihi

İlk olarak 1943 yılında sinir ağlarının incelenmeye başlanması ile makine öğrenmesinin temeli atıldı denilebilir. Bir mantıkçı olan Walter Pitts ve nörobilimci Warren McCulloch ile birlikte yaptıkları çalışma ile bize insan beynindeki sinir ağlarının matematiksel modelini vermiş oldular.

İngiliz matematikçi Turing 1950 yılında yazmış olduğu "makinelerin işleyişi ve zeka" makelesi ile öğrenen bir makine tarif etmekte ve genetik algoritmalar hakkında bilgi vermektedirler. Aynı makele içinde Turing testi olarak tarihe kaydedilecek olan bir bilgisayarın düşünüp düşünemediğininin testini de ortaya koymaktadır.

1952 yılında makine öğrenmesinin babası olarak bilinen Arthur Samuel IBM laboratuvarında ilk makine öğrenimi programını oluşturdu. Program bilgisayarın dama oynaması için kodlanmıştır.

Frank Rosenblatt isimli psikolog 1957 yılında "Algılayan ve Tanıyan bir Otomasyon" yazısısı ile elektriksel veya ses dayalı kalıpları arasındaki benzerlikleri ve kimlikleri tanımayı öğrenecek elektromekanik bir sistem tasarlayacağını belirtti.

Henry J. Kelley, 1960 yılında "Optimal Uçus Rotaları için İrtifa Kuramı" isimli makalesi ile kontrol kuramı konusundaki birçok fikir sundu. Bu fikirler sürekli geri yayılım modellerinin gelişimindeki temelleri oluşturdu.

Çalışan ilk derin öğrenme modeli 1965 yılında Alexey Ivakhnenko ve V.G. Lapa tarafından yaratılmıştır. Öğrenme algoritmalarının en iyi özellikleri bulup bunları sistemin içinde öne koymak için her katmanda istatiksel seçenekler kullanan ileri derin öğrenme beslemeli çok katmanlı algılayıcılar kullandılar.

Sinir ağları üzerinde yaptığı çalışmalar ile bilinen Fukushima, görsel kalıpları nasıl tanıyacağını öğrenen Neocognitron olarak isimlenen öğrenen yapay sinir ağı ile duyulmuştur. 1979-1980 yıllarında yaptığı bu çalışmalar, örüntü tanıma görevlerinde, öneri sistemlerinde ve doğal dil işlemede kullanıldı.

1982'de Hopfield, kendi adını taşıyan sinir ağları ile ilişkilendirilebilir bellek sistemleri olarak çalışan tekrarlayan sinir ağlarını yarattı.

1985'te Terry Sejnowski, öğrenme sürecini ingilizce kelimeleri telaffuz eden bir programa yansıttı. Program en doğru telaffuzu yapana kadar öğrenmeye sürekli devam etti.

David Rumelhart ve arkadaşlarının 1986 yılında yazdığı "Geri-yayılımlı Hatalarla Öğrenme Temsilleri" makalesinde yayılım sürecini ayrıntılı olarak anlattılar.

1989 yılında Christopher Wathinks Q-öğrenme algoritması ile geçiş olasılıklarını veya beklenen ödülleri modellemeden, optimal kontrolü öğrenmenin yapılabileceğini sundu.

Alman bilim adamı Schmidhuber 1993'te tekrarlayan sinir ağında birçok öğrenme görevini başarıyla sonlandırdı.

Çalışmamızda da kullandığımız LSTM derin öğrenme algoritması ilk kez 1997 yılında Jurgen Schmidhuber ve Sepp Hochreiter tarafından ortaya konulmuştur.

Google yakın zamanda sürekli gelişen LSTM ağlarını DL halkalarında kullanarak akıllı telefonlarda konuşma-tanıma yazılımına entegre etti.

Yapay zeka profesörü Fei-Fei Li, 2009 yılında ImageNet ile çok büyük bir veritabanı oluşturmanın temelini attı. Görseller, ingilizce kelimelerin, isimlerin, yüklemlerin, zarfların ve sıfatların eş küme denen eş anlamlı gruplara göre sıralandığı sözcüksel bir veri tabanıdır. 2011 yılında Alex Krixhevshy, AlexNet ile birçok makine öğrenmesi yarışmasını kazandı. Yarattığı evrişimli sinir ağı, 8 katmanlı bir yapıda hızı ve düşünceyi oldukça güçlendirdi.

2012 yılında binlerce bilgisayara yayılmış sinir ağları ile youtube'dan alınmış 10.000.000 etiketlenmiş görselden program, kedileri belirlemeyi ve tanımayı öğrenmişti. Sunulan objelerden %15'ini tanımış olsa da bundan önceki öğrenmelerden %70 daha başarılı olmuştu.

2014 yılında Facebook, DeepFace ile yüzleri yaklaşık %97 oranında tanımayı başardı. Bu öğrenme sistemi ile bir önceki çalışmadan %27 daha başarılı oldu. Hiç kuşkusuz bu başarıda Facebook'un devasa veri platformuna sahip olmasının katkısı büyüktü.

2016 yılından sonra artık birçok şirket makine öğrenimi konusunda güçlü ürünler sunmaya başladı. Microsoft XC50 süperbilgisayarında 1000 tane Nvidia Tesla P100 grafik işleme ile öncekine kıyasla küçük bir zamanda derin öğrenme uygulayabilmektedir.

3.6.2. Derin öğrenme mimarileri

Makine öğrenmesi, makinenin büyük veri setlerini kullanarak öğrenmesini sağlar. Derin öğrenme ise bir makine öğrenme yöntemidir. Derin öğrenme sürecinde hedeflenen başarı oranı belirlenen seviyeye ulaşana dek sürekli öğrenme için tekrarlanır. Verilen veri kümesi ile sonuçları tahmin edebilecek yapay zekanın geliştirilmesine imkan sağlar.

Derin öğrenme algoritmaları içinde en yaygın kullanılanlar, convolutional neural network (konvolüsyonel sinir ağları-CNN), recurrent neural network (tekrarlayan sinir ağları-RNN), long-short term memory (uzun kısa süreli hafıza ağları-LSTM)'dir. Bu algoritmalar ile modellemeler güçlendirilip makine öğrenme adımları yapılmaktadır. Derin öğrenme mimarileri yapay zeka problemlerinin çözümü için pek çok yaklaşım sunmaktadır.

3.6.2.1. Konvolüsyonel sinir ağları (Convolutional Neural Network (CNN))

Makine öğrenmesinde CNN görüntülerin analiz edilmesi başarıyla uygulanmış derin ileri beslemeli yapay sinir ağıdır. Google'ın eklenen görseli tüm görseller içinde araması ve benzerlerini kullanıcılara göstermesini sağlayan görsel arama motoru bu yapay sinir ağı ile yapılmaktadır.

Beynimiz bir görselin ne olduğuna karar verirken aslında bir sıralama ve benzer ile kıyaslama yapmaktadır. Sokakta dört ayaklı, iki kulaklı ve kuyruklu bir hayvan gördüğünde bunun kedi veya köpek olabileceğine bir önceki bilgileri ile kıyaslama ve özellikleri sıralama yaparak karar vermektedir. CNN'de bir makinenin görseli algılama üzerindeki yeteneği beynin sinir yapısı yeteneğini kullanabilmesi ile olanaklı hale gelmiştir (Tüfekçi ve Karpat, 2019).

Birden fazla katmandan oluşan bu sinir ağı giriş ile çıkış verilerini birbirine bağlamaktadır. Genellikle görseller üzerinden kümeleştirme, tarama ve sıralayarak benzerlerini çıkarma gibi işlemler yapmaktadır. Sinir ağını sürekli aktif tutmak ve en başarılı sonuçlara ulaşmasını sağlamak için çok farklı parametrelerden ve bu yapıların oluşumuna katkı sağlayan yüksek hacimli verilere sahip kütüphanelerden yaralanmaktadır. Görsellerin girişleri yapılarak makineye aktarılması sonrası çalışmaya başlayan konvolüsyonel sinir ağları, bu verileri matris formatına çevirerek sisteme input etmektedir. Görselin derinliği matrise göre ifade edilmekte olup renklerine göre farklı ifadeler almaktadır.

Matrisler sayesinde resimler ve videolar üzerinde filtreler uygulanabilmektedir. Bu filtreler ile görselde aranan özellikler karşılaştırmalar ile çıkmaktadır.

Katmanlar üzerinde çeşitli değişkenler yardımı ile görsel incelenmektedir. Daha sonra farklı kategoriler içerisine konulmaktadır. Uygulanacak filtreler kullanıcılar tarafından seçilir veya kütüphaneden aktarılan veriler ile uygulanır. Bu filtreler yardımıyla resim hangi kategoride değerlendiriliyorsa o parametreler açısından analiz edilir ve çıktı verilir.

,

Kısacası CNN görüntüyü sınıflandırma, nesne tanımlama, görüntü segmentasyonu gibi işlemler başarılı bir şekilde yapılmaktadır. İnsanların görme sistemini örnek alan bu sinir ağları ile yapay sistemlerde, nesnelerin algılanması, tanımlanması ve sınıflandırılması amaçlanmıştır.

3.6.2.2. Tekrarlayan sinir ağları (Convolutional Neural Network (RNN))

Tekrarlayan sinir ağları (RNN), düğümler arasındaki bağlantıların yönlendirilmiş bir döngü oluşturduğu yapay sinir ağıdır. Bu ağda asıl amaç ardışık bilgileri kullanmaktır. Gizli katman çıktısını aynı katmana girdi olarak gönderebilen derin öğrenmedir (Doğan ve Türkoğlu, 2019). Oluşturulan katmanlar üzerine işlenen veriler üzerinde yapılan matematiksel işlemler ile çıktı veriyi ifade eder. Tekrarlayan sinir ağlarında veri ileriye yönelik işletilmektedir. İşlenen veriler ile bir tahmin elde edilmektedir. Veriler doğru sonuçlar ile karşılaştırılarak hata payı ortaya çıkmış olur. Sinir ağında hata payı oranında değişime gidebilecek gerçek sonuçlar gelmeye başlamaktadır.

Tekrarlayan sinir ağında hata oranının düşürülmesi temel hedeftir. Böylelikle girdi ile çıktı verileri birbirine bağlanmış olacaktır. Bu sinir ağında sadece o an giriş yapılan veri ile değil, o veriden önce sisteme yüklenen ve yüklenecek verilerle bağlantılı olarak çalışmaktadır. Önceki veride verilen karar yeni işlenmekte olan ve daha sonra işlenecek veriyi de etkilemektedir. Tıpki insan beyni nasıl daha önce yaşadıklarını şuan yaşadıkları ile birleştirip tüm bunlara göre bir karar veriyorsa RNN'de aynı mantıkta çalışmaktadır. Bu sepeble RNN belleğe sahip olan bir derin öğrenme algoritmasıdır.

RNN sayesinde zaman sorunlu problemler kısa sürede çözülebilmektedir. Tekrarı olan aktivitelerde bazı bilgiler ağ içerisinde tutulurken tekrar etmeyen bilgiler ağdan çıkarılabilir. Ayrıca faaliyet için gerekli bilgiler ağa geçmiş bir zaman içerisinde dahil edilmişse bu bilgiye tekrar ulaşmak sorun olabilir. Uzun zaman önce işlenmiş bilgiler hatırlanırsa algoritma daha sağlıklı tahminler verebilir.

Dil modelleme bu algoritma ile yapılmaktadır. Eğitim verileri open source kütüphaneler ile ağa yönlendirilir. RNN ile yapılan bu eğitim çalışmaları sonrası bir sonraki kelimenin tahmin edilebilme ihtimali yükselmiş olur.

Eğer CNN ve RNN aynı anda kullanılırsa konuşmayı tanıma özelliği oluşturulabilmektedir. Ses dalgalarının fonksiyonları ele alınarak bir tahmin sıralaması yapılır. Ses dalgalarını kelimeye çeviren programlar bu algoritmaları kullanarak çalışmaktadır.

3.6.2.3. Uzun Kısa Süreli Hafıza Ağları (Long Short Term Memory (LSTM))

LSTM, uzun vadeli bağımlılıkları öğrenebilen özel bir RNN türüdür. RNN'de tespit edilen bazı problemlerden dolayı ortaya çıkmıştır. RNN doğal dil işlemede başarılı sonuçlar vermiştir fakat bazı uzun cümleleri verdiğimizde geçmiş verileri hatırlamakta eksik kalmıştır. RNN, çıkış verisini giriş verisi olarak alır bu yüzden diğer öğrenme algoritmlarına göre sonraki kelimeyi tahmin etmede daha iyilerdir. Teorikte uzun dizinlerde de daha iyi sonuçlar vermesi beklenirken pratikte eksikleri olduğu görülmüş ve ihtiyaçtan dolayı LSTM gibi yeni sinir ağları tasarlanmıştır. LSTM öğrenmesini insan zekasından örnekleyecek olursak, uzun süreli bilgileri hatırlamak mücadele ile sürekli öğrendiğimiz değil varsayımsal bir davranışımızdır. LSTM, uzun veya kısa periyotları hatırlar. Çeşitli problemlerde çok iyi çalıştıkları için çalışmalarda yaygın bir şekilde kullanılmaktadır. Tekrarlayan sinir ağlarında öncü olarak kabul edilen LSTM'ler uzun vadeli bağlımlılık problemlerine çözüm olabilmesi için tasarlanmıştır. Uzun veya kısa periyotları hatırlama kabiliyetinin özelliği, tekrarlanan bileşenlerinde hiçbir etkinleştirmeyi kullanmamasıdır.

Çizelge 3.1'de derin öğrenme algoritmaları arasındaki farklar ve karşılaştırmalar verilmiştir.

Çizelge 3.1. Derin öğrenme karşılaştırma (Zhu vd, 2018).

Type(Tip)	**Variant (Varyant)**	**Network structure (Ağ Yapısı)**	**Applications (Uygulamalar)**
CNN	LeNet	Input Layer(Giriş Katmanı) Output Layer(Çıkış Katmanı) Hidden Layer(Gizli Katman)	Image processing(Görüntü İşleme) Speech signal(Konuşma Sinyali) Natural Language (Doğal Dil) Processing(İşleme)
RNN	LSTM	Input Layer(Giriş Katmanı) Output Layer(Çıkış Katmanı) Hidden Layer(Gizli Katman)	Time series analysis(Zaman serisi analizi) Emotion analysis(Duygu analizi) Natural Language Processing

Bir bilgisayarla etkileşim kurmanın en önemli yollarından biri klavye ve faredir. Klavye etkileşimi yalnızca davranışsal verileri (yazma hızı gibi) değil, aynı zamanda bilişsel ve dilsel verileri de içermektedir (Juola vd, 2013). Derin öğrenme algoritmaları ile klavye üzerindeki davranışsal verilerin analizinin çıkarılması için yeni modeller geliştirilmiştir.

4. ÇALIŞMANIN ANALİZLERİ

Veri analizi ham verinin toplanması ve bir takım analiz yöntemleri kullanılarak verinin doğruluğunun teyit edilmesi ve akabinde anlamlı ve sürece faydalı bilgiler haline getirilmesidir. Veri analizi, kurumların stratejilerinin çizilmesi ve önemli karar süreçlerinde hataların önüne geçilmesi için oldukça önemlidir. Veri analizi sürecinde bazı önemli aşamalar vardır. Yazının önceki bölümlerinde vurgulandığı üzere veri toplama ve veri temizleme en önemli ilk iki adımdır. Son adım ise en doğru sonuca ulaşana kadar sürekli tekrar etmektir. Ne kadar çok tekrarlanırsa hatalar o kadar ortaya çıkacak ve veri temizleme adımına sizi geri götürecektir. Kurulan modellerin doğruluğuna tekrar eden sonuçlarda benzer çıktıları alarak ulaşabiliriz. Analiz edilecek veri miktarı arttıkça bu konudaki uzmanlık ve doğru yorumlama mecburiyeti de artar. Analiz edilen verinin karşı tarafa aktarılması da çok önemlidir. Hitap edilen kesime bağlı olarak çalışma sonuçlarını görsel olarak paylaşmak en anlaşılır yol olacaktır.

Veri görselleştirme, verileri insan beyninin algılaması için daha kolay hale getirme uygulamasıdır. Görselleştirme, verilerdeki hataları ve Şekil 4.2'de gösterildiği gibi aykırı verileri daha rahat görmemezi sağlar. Veri görselleştirme araçları ve teknolojileri büyük miktarda veriyi analiz etmek ve stratejik kararlar almak için yardımcı olurlar. Görselleştirmeden kasıt grafikler, haritalar olduğu gibi Şekil 4.1'de gösterdiğimiz gibi çalışanın çervesine özelde yapılabilir. Karşı tarafa çalışmayı gözünde canlandırabilmesini ve çalışmayı yapanların istatiksel bilgileri daha rahat görmesini sağlar.

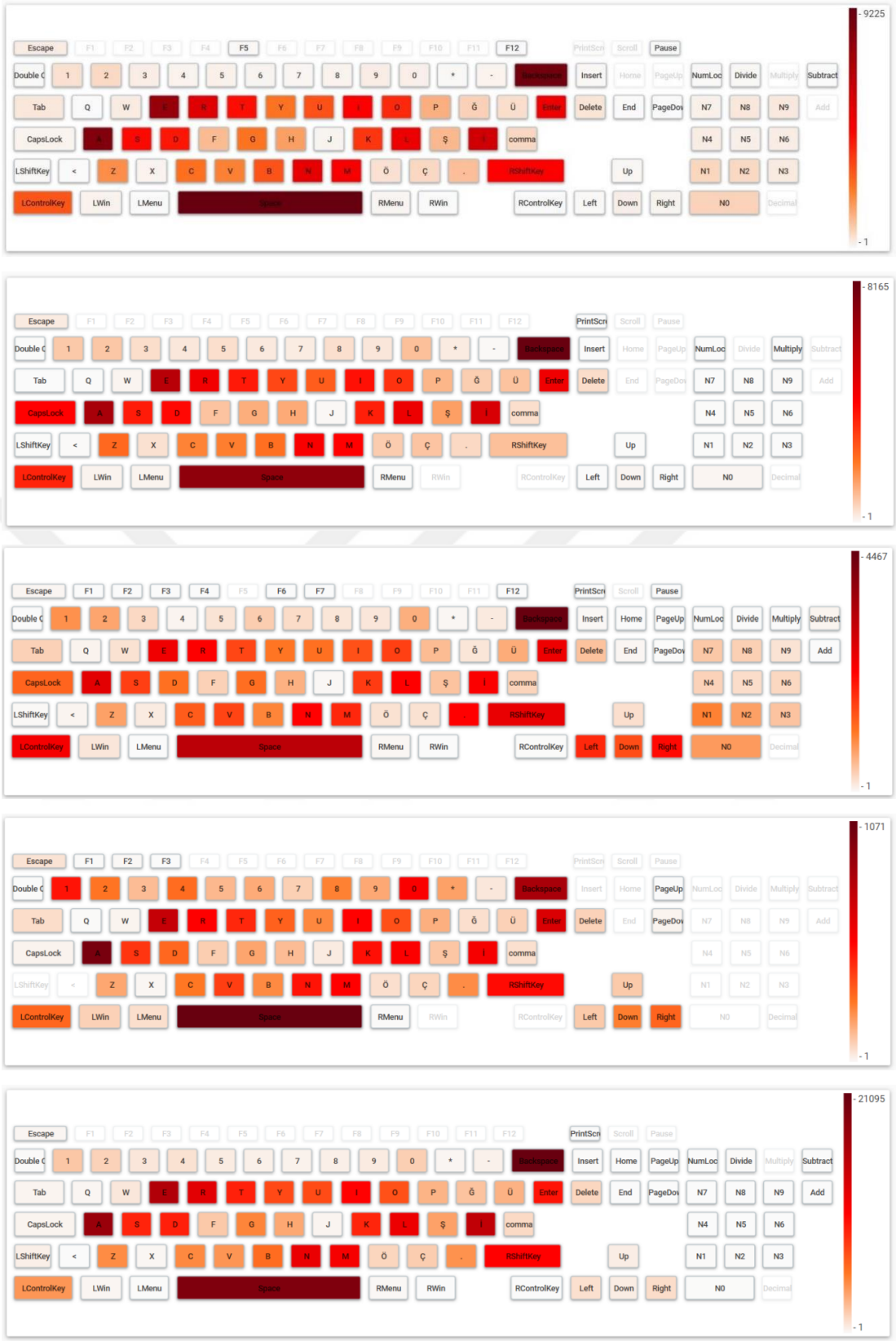
Çalışmada KDA agent programıyla toplanan veriler ile ilgili doğru sonuçların tahminini yapabilmek için bazı denemeler yapılmıştır. Bu deneme çalışmalarında amaç klavye davranışlarının kullanıcılarda gerçekten benzersiz olup olmadığını ve bu değerlerin bizi gerçek bir kişiye ulaştırıp ulaştıramayacağını görmektir.

Çalışmada tıklama sayısını görselleştirmek için kodlama yapılmış ve Keyboard Data Visualizer ile farklı çalışanlara ait tuş vuruş sayısının çıktısı Şekil 4.1'de gösterilmiştir. Aynı sürede verileri toplanmış 5 farklı kullanıcıya ait görsellerde

görünmektedir ki her kullanıcı farklı yoğunlukta yazma işlemi yapmakta ve farklı tuş takımlarını kullanmaktadır. Ortak özellikler olarak görünen hepsi en çok Backspace ve Space tuşunu kullanmaktadır. Rengin koyulaşması kullanım sıklığını göstermektedir. 4.görseldeki kullanıcı klavye sağındaki rakam tuş takımını hiç kullanmamaktadır. Bu kullanıcı için alışkanlığının klavye üstünde bulunan rakamları kullanmak olduğunu söyleyebiliriz. Fonksiyon tuşlarının bu kullanıcı grubu için çok sık kullanılan tuş takımı olmadığı görülmektedir.

Görseller, üzerinde makaleler yazılmış türkçe kelimelerde en çok kullanılan harflerin çalışma sonuçlarını da desteklemektedir. Yapılan çalışmaya göre Türkçe'de en sık kullanılan sesli harf "a" ve "e", sessiz harf ise "k" harfidir (Güneş ve Işık, 2018). Görsellerimizde görüldüğü üzere çalışmamızda da en sık kullanılan harfler bunlardır. Aynı çalışmaya göre ilk on sırada "a,e,k,i,l,m,r,n,t,ı" harfleri yer almakta olup Türkçe'de ki kelimelerin yarısından fazlası bu harfler kullanılarak yazılmaktadır. Yabancı paydaşı olmayan bu kuruma ait çalışanların W,X gibi Türkçe alfabede olmayan tuşları çok sık kullanmadıkları da görülmektedir.

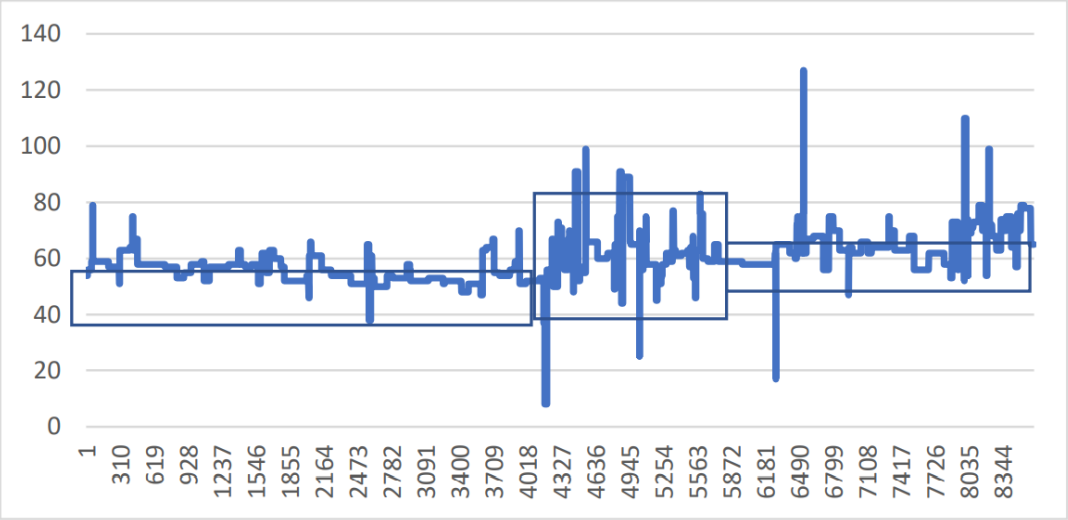
Tüm bu ortak ve farklılıklar ışığında söylememiz gerekir ki her kullanıcının hatta her milletin farklı klavye kullanım alışkanlığı vardır. Buradan alınacak veriler ile çeşitli çalışmalar yapılabileceği gösterilmiştir.



Şekil 4.1. Tıklama sayısını gösteren araç

Daha detaylı incelendiğinde veri toplanan zamanın belli bir periyodunda bir kullanıcının L tuşunun basılı tutma zamanları sabit bir şekilde gidiyorken birden daha uzun süre basılı tuttuğu verileri ortaya çıkmıştır. Bir süre sonra değerler daha yüksek bir değere çıkıp akabinde stabil haline dönmüştür. Bu olay verinin sahibine gösterildiğinde verilerin değiştiği zaman kullanıcının L tuşunun bozulduğu ve klavyenin vuruşu algılamamaya başladığını bu sebeple daha uzun ve sert bastığını iletmiştir. Kullanımdan bir süre sonra bu duruma artık elinin alıştığını daha sonra da tuşun değiştirildiği bilgisi alınmıştır.

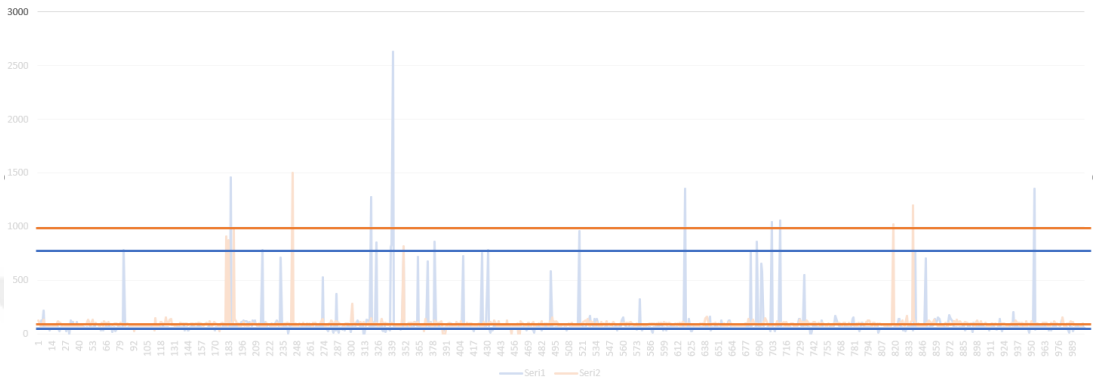
Çalışma her ne kadar siber güvenlik açısından değerlendirmiş olsa da Şekil 4.2'de gösterilen sonuç bir donanım arızasını da tespit etmiştir. Bu arızadan dolayı farklı tepkiler sergilemek durumunda kalan çalışanın çalışma şeklini de bozması insan kaynakları açısından değerlendirilebilecek sonuçlar elde edilebileceğini de göstermiştir.



Şekil 4.2. L tuş verileri değişimi

Bu görseller sayesinde her kullanıcının kendine özgü bir kullanım profili ve diğer kullanıcılardan farklı bir yazım tarzı olduğu açıkça görülmektedir. Çalışmanın tekil bir veri olan klavye kullanım alışkanlığı üzerinden yapılmasının doğru bir yol olduğu gösterilmiştir.

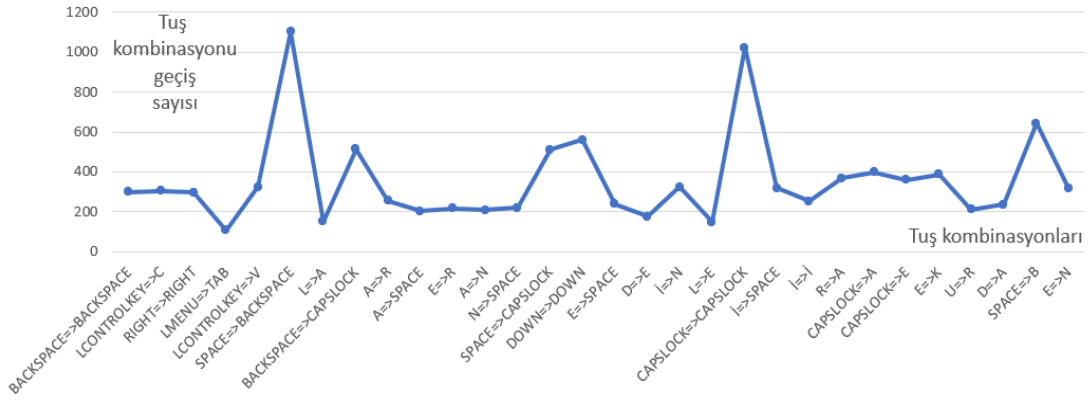
Tıklama sayısı verileri gözlemlendikten sonra tuşların basılı kalma süreleri arasındaki farklar incelenmiştir. İzleme yapan (monitoring) çalışan (series1) ve kod yazan yazılımcı çalışan (series2) arasındaki backspace tuşu basılı tutma süresi Şekil 4.3'te enter tuşu basılı tutma süresi karşılaştırmaları ise Şekil 4.4'te verilmiştir.



Şekil 4.3. İki kişinin backspace tuşu basılı tutma sürelerinin karşılaştırılması

Şekil 4.4. İki kişinin enter tuşu basılı tutma sürelerinin karşılaştırılması

İzleme yapan (series 1) çalışana ait klavye kombinasyonları ve kombinasyonları kullanma sayıları Şekil 4.5'te gösterilmiştir.



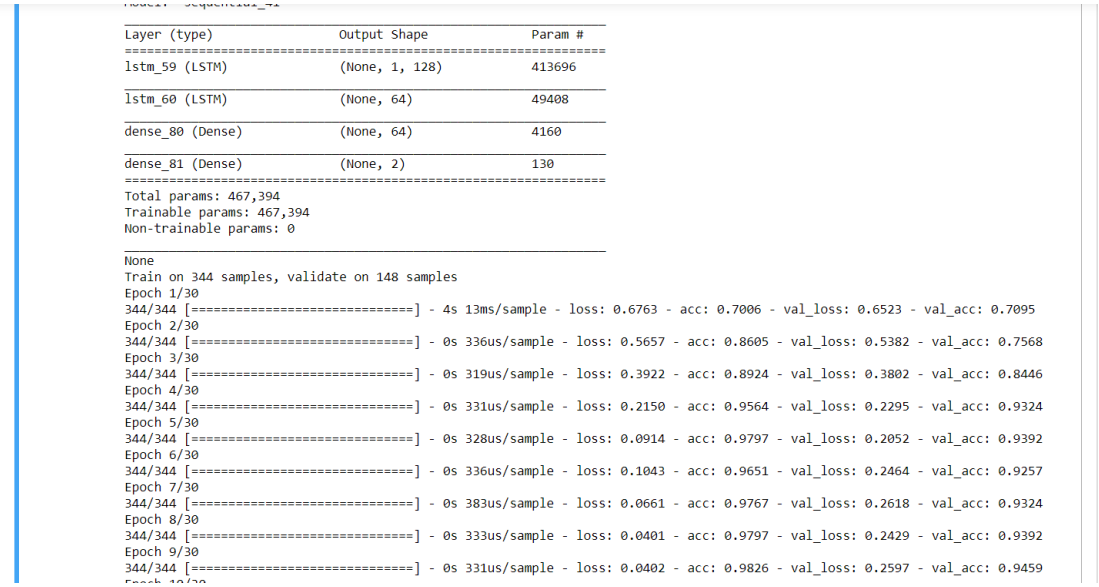
Şekil 4.5. Series 1 kullanıcı klavye kombinasyonları ve geçiş sayıları.

Çalışmanın odak noktasına uygun yeni bir derin öğrenme modeli oluşturulup farklı davranışlara sahip kullanıcıların klavye tuş vuruşları göz önüne alınmıştır. Bir kullanıcının davranışlarını öğrenen algoritma ile kimlik doğrulama yapmasını ve bir siber saldırganın sisteme giriş denemelerinin tespit edilmesi incelenmiştir. Oluşturulan algoritmaların performansı, araştırmada iki derin öğrenme modeli ile kıyaslanmış, RNN ve LSTM algoritmaları denenmiş ve Şekil 4.6 ve Şekil 4.7'de sonuçları paylaşılmıştır.

```
344/344 [==============================] - 0s 177us/sample - loss: 0.0316 - acc: 0.9884 - val_loss: 0.3073 - val_acc: 0.9257
Epoch 21/30
344/344 [==============================] - 0s 194us/sample - loss: 0.0394 - acc: 0.9826 - val_loss: 0.3192 - val_acc: 0.9189
Epoch 22/30
344/344 [==============================] - 0s 183us/sample - loss: 0.0317 - acc: 0.9826 - val_loss: 0.2975 - val_acc: 0.9257
Epoch 23/30
344/344 [==============================] - 0s 203us/sample - loss: 0.0515 - acc: 0.9738 - val_loss: 0.3081 - val_acc: 0.9189
Epoch 24/30
344/344 [==============================] - 0s 194us/sample - loss: 0.0343 - acc: 0.9884 - val_loss: 0.3025 - val_acc: 0.9324
Epoch 25/30
344/344 [==============================] - 0s 185us/sample - loss: 0.0337 - acc: 0.9855 - val_loss: 0.3366 - val_acc: 0.9122
Epoch 26/30
344/344 [==============================] - 0s 209us/sample - loss: 0.0344 - acc: 0.9826 - val_loss: 0.3072 - val_acc: 0.9257
Epoch 27/30
344/344 [==============================] - 0s 200us/sample - loss: 0.0317 - acc: 0.9855 - val_loss: 0.3182 - val_acc: 0.9257
Epoch 28/30
344/344 [==============================] - 0s 191us/sample - loss: 0.0322 - acc: 0.9738 - val_loss: 0.3147 - val_acc: 0.9324
Epoch 29/30
344/344 [==============================] - 0s 183us/sample - loss: 0.0400 - acc: 0.9826 - val_loss: 0.3154 - val_acc: 0.9324
Epoch 30/30
344/344 [==============================] - 0s 193us/sample - loss: 0.0381 - acc: 0.9767 - val_loss: 0.3154 - val_acc: 0.9324
Test loss: 0.31544431812457135
Test accuracy: 0.9324324
```

Şekil 4.6. RNN algoritma çıktısı

Şekil 4.7'de LSTM algoritmasının on kişilik veri havuzundan öğrendiği davranış ile iki kişinin verisini birbirinden yüzde 90 üzerinde ayırabildiğinin çıktısı gösterilmiştir.



Şekil 4.7. LSTM algoritma çıktısı

LTSM algoritması en başarılı accuracy (doğruluk) değerine 9. epoch (adım) denemede, RNN algoritması ise 30. denemede yaklaşmıştır.

Çizelge 4.1. Algoritma karşılaştırma.

Algorithm(Algoritma)	Epoch (Adım)	Users (Kullanıcılar)	Loss (Kayıp)	Accuracy (Doğruluk)
LSTM	9	10	0.0402	0.9826
RNN	30	10	0.0381	0.9767

Çizelge 4.1'deki değerlere göre iki algoritmanın performans karşılaştırması verilmiştir. LSTM algoritmasının daha başarılı sonuç verdiği doğruluk değerinin 1 sayısına yakınlaşması ile görülmüştür. LSTM ile 1. adımda kayıp değeri 0.6763, doğruluk değeri 0.7006 iken 3.adımdaki kayıp değeri 0.3922 olarak azalmış, doğruluk değeri ise 0.8924 olarak arttığı görülmüştür. Algoritmayı eğitmeye devam ettiğimizde 9.adımdan sonra kayıp değeri 0.0402, doğruluk değerimizde 0.9826 değerlerine ulaşmıştır. RNN algoritmasının 21.adımından itibaren incelediğinde kayıp değeri 0.0316, doğruluk değeri 0.9884 olarak görülmüştür. Bu sonuçlar, kayıp değeri 0'a, doğruluk değeri de 1'e yaklaştığında algoritmaların başarılı olarak öğrenme sürecini gerçekleştirdiğini göstermektedir. LSTM 9. adımında kayıp değerini 0'a, doğruluk değerini 1'e, RNN ise 30.adımında bu

değerlere yaklaşabilmiştir. Bu kıyas ışığında çabuk öğrenme gösteren LSTM algoritmasının daha başarılı olduğu görülmektedir.

5. TARTIŞMA

Siber güvenlikte parola güçlendirmesini hedefleyen çalışmada, parola güvenliğini artırmak için tuş vuruş dinamikleri kullanılmaktadır (Monrose vd, 2002). Buradaki eksiklik derin öğrenme teknikleri ile kimlik doğrulamanın kullanılmamış olmasıdır. Derin öğrenme algoritmaları ile modeller geliştirilerek literatürdeki bu boşluk tarafımızca doldurulmuştur.

Klavye ve mouse vuruşlarını biyometrik kimlik doğrulama yöntemi olarak kullanan çalışmanın veri seti oluşturma aşamasında veri toplama programı tuş verilerini sıralı bir şekilde aldığı için kullanıcıların verileri bilgilere dönüştürülebilir (Mondal ve Bours, 2017). Kullanıcının sistemlere girişi esnasında kullandığı hesap parolaları, alışveriş yaparken girdiği kredi kartı bilgileri ve e-devlet işlemleri sırasında vatandaşlık bilgileri sistem yöneticileri tarafından görülebilir ve bu durum istenmeyen güvenlik açıklıklarına sebep olabilir. Bu açığı kapatmak için çalışmamızda verilerin sırasız bir şekilde tutulabilmesi göz önüne alınarak yeni bir veri yapısı geliştirilmiştir

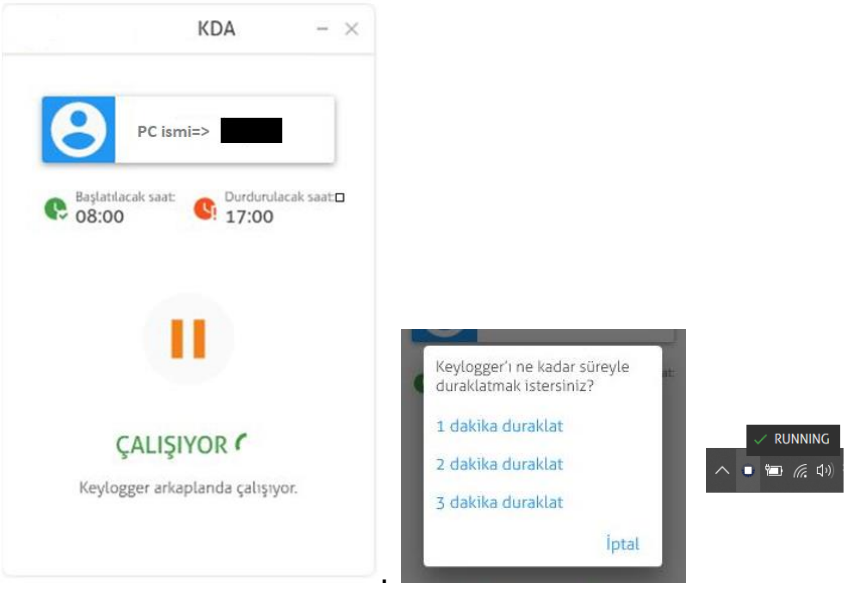
Tuş vuruşu dinamiği tabanlı kimlik doğrulama mekanizması inceleyen çalışmada, bir grup yazarın tuş vuruşları analiz edilerek benzersiz yazma kalıpları olduğu görülmüş ve bu sonuçlardan kullanıcının gerçekliğini doğrulamak için yararlanılabileceği öne sürülmüştür (Raul vd, 2020). Burada veri setlerin incelenmesi için örneklem alınan grup sadece aynı yetkinliklere sahip yazarlardan oluşmaktadır. Bunun farklı sektörlerde başarılı sonuçlar vermesi beklenilemez. Bu eksikliği gidermek için tarafımızca farklı tipte kullanıcılardan veri setleri toplanmış ve üzerinde analizler yapılmıştır.

Bir dizi tuş vuruşu özelliği kullanarak farklı kullanıcılar arasında ayrım yapmak için mantıksal zamansal özelliklerle model kontrolünü kullanan çalışma yapılmıştır (Di Tommaso vd, 2019). Destek vektör makineleri algoritması, tanımlama ve doğrulama görevlerinde iyi sonuçlar göstermiştir fakat değerlendirmeleri kendi veri kümelerinden çıkarılan bir dizi kısıtlı özelliklerdir.

Bu kısıtı aşmak için çalışmamızda kullanıcılardan tüm klavye tuş ve kombinasyonlarından alınan veri setleri üzerinde analizler yapılmıştır.

Klavye, fare ve web sitesi ziyaret sıklığına bakılarak davranış analizi çıkaran çalışmada, kimlik doğrulama konusunda incelemeler yapılmıştır (Juola vd, 2013). Bu çalışmada, veri setlerini toplamak için klavye ve fare üzerine sensörler yerleştirilmiştir. Bu sensörler ekstra donanım ve yazılım maliyeti oluşturduğundan tarafımızca KDA agent adındaki keylogger programı geliştirilmiş ve bu sayede klavye hareketleri masrafsız olarak toplanmıştır.

Yukarıda bahsedilen çalışmaların hiçbirinde veri toplanma sırasında kullanıcıya kontrol hakkı verilmemiştir. Çalışmadaki amacımız daha güvenli bir sistem yaratmakken çalışan motivasyonunu da düşürmemektir. Kullanıcının sürekli izleniyor olmasından dolayı farklı davranışlar (tedirginlik) sergilemesi riski öngörülmüş ve uygulamayı kontrol edebilecekleri Şekil 5.1.'de gösterilen bir arayüz geliştirilmiştir. Kullanıcının güvenini zedelememek adına istedikleri zaman programı durdurabilecekleri bir arayüz verilmiştir. Çalışmanın sağlıklı veri alabilmesini de engellememek adına kısa süreli tutulmuştur.



Şekil 5.1. KDA Agent kullanıcı arayüzü

Kişisel verilerin yasal olmayan yollar ile elde edilmesinin artması ve ülkemizde bu konudaki regülatif düzenlemelerle ifşaların daha çok gündeme gelmesi, kullanıcıların klavye hareketlerinin analiz edilmesini istememesine sebep olmuştur. Bu da çalışmanın daha büyük veri setlere ulaşmasında en büyük sorunu oluşturmuştur.

Kullanıcılardan alınacak verilerin saldırgan tarafından gelişmiş saldırı metotları ve sosyal mühendislik tekniklerini kullanarak çıkarılmasının az da olsa bir ihtimali vardır. Fakat bu riski azaltmak için veriler lokal bir sunucuya kaydedilmiştir. Ayrıca veri toplama aşaması bittikten sonra o verilerin kime ait olduğu bilgisi silinip veriler anonim hale getirilmiştir.

5.1. Yazılımın Güvenlik Açıklarının Araştırılması

Çalışma yapılırken aktif yazma işlemi yapmayan daha çok izleme (monitoring) yapan kullanıcının en sık kullandığı tuşları görselleştirdiğimiz de yoğun kullanılan tuşların kullanıcı hesap bilgileri içinde kullanılan karakterler olabileceğini ve oluşturulacak sözlükler ile sistemlere giriş parolalarının tespit edilebileceği riski görülmüş, kullanıcı rızası ile parola tespit edilmeye çalışılmıştır.

Çalışma için öncelikle yaklaşık 500 session incelenerek karakter sayısı 10'dan az sessionlar çekilmiş ve içindeki karakterler listelenmiştir. Ardından çıkan karakterler karşılaştırılarak en çok kullanılan 8 karakter çıkarılmıştır. Kullanıcıyla iletişime geçip programın çalıştığı zamanda kullandığı parola şifreli bir şekilde alınmıştır. Parola alındıktan sonra brute force algoritması yazılmış ve parola tespit edilmeye çalışılmıştır. Program iki gün çalıştıktan sonra parolayı eşleştiremediğine dair bilgi ve tahminler paylaşmıştır. Çıkarılan karakter listesi kullanıcıya gösterildiğinde parolasında kullandığı çoğu karakterin listede bulunmadığını iletilmiştir. Bu çalışma ile güvenlik açıklığı olup olmadığının teyiti yapılmıştır.

6. GELECEK ÇALIŞMA

Gelecek çalışma için daha geniş veri setleri elde edilmesi, algoritmanın bu veri setleri ile daha fazla kullanıcı üzerinde başarılı sonuç vermesi ve modelin yeniden geliştirilmesi hedeflenmektedir. Model geliştirildikten sonra kullanıcılarda çalışan yazılıma entegre edilmesi, yazılımın kullanıcı alışkanlarını öğrenip ikinci aşamada kullanıcı tespiti yapmaya başlaması ve sürekli bir şekilde log ve data kaydetmesi planlanmaktadır.

Çalışma, siber güvenlik perspektifinden değerlendirilip kişilerin sistemlere girişleri (MFA- Multi Factor Authentication) klavye davranışları ile birlikte birden fazla yöntem ile doğrulanacaktır. Farklı klavye davranışını tespit eden geliştirdiğimiz program, ürettiği log ile birlikte merkezi log sistemine alarm oluşturacak ve saldırı teşebbüsünden Security Operation Center (SOC) ekiplerini haberdar etmesi sağlanacaktır. Siber güvenlik alt yapısının proaktif olarak yönetildiği kurumlarda alarm ile birlikte ilgili PC'nin karantina Virtual Local Area Network (Sanal Yerel Alan Ağı-VLAN)'a alınması, şirket ağından çıkarılması, internet erişimlerinin kesilmesi ve kullanıcı hesaplarının pasife alınması gibi birçok aksiyonu insan müdahalesi gerektirmeden otomasyonla yapılması hedeflenmiştir. Bu çalışma, yukarıda bahsedilen alanlarda yapılacak çalışmara ön proje niteliğindedir.

7. SONUÇ VE ÖNERİLER

Yapılan çalışmada kullanıcılardan alınan klavye verileri ile makine öğrenmesi yaparak kişilerin tespit başarısı ölçülmüştür. Klavye verilerinin toplanması ve datasetlerinin oluşturulması için kullanıcı bilgisayarlarına yüklenmek üzere bir agent program geliştirilmiştir. Kullanıcılardan toplanan verilerin analiz sonuçları görsel araçlar ile paylaşılmıştır. Derin öğrenme algoritmaları incelenmiş ve LSTM ile analiz yapılmıştır. Gerçek 10 kişiden 3 aylık süre ile alınan klavye hareketleri öğretilerek farklı işlere sahip iki kişinin (yazılımcı, izlemeci) LSTM algoritması ile farklılığının tespiti yapılmıştır.

Siber güvenlik perspektifinden değerlendirildiğinde bu çalışmanın bakış açısı sayesinde kişilerin sistemlere girişleri (MFA- Multi Factor Authentication) birden fazla yöntem ile doğrulabilir. Diğer MFA yöntemlerine göre (SMS, e-posta) gerçek kişiden oluşan veriyi temsil ettiğinden ve taklidi neredeyse imkânsız olduğundan daha güvenilir bir yöntem sunması beklenmektedir.

MFA'ya ek olarak, dinamik analiz ile birlikte kullanıcı makinesine fiziken veya uzaktan erişen siber saldırgan farklı klavye davranışı sergileyeceği için programdan alarm üretilebilir ve merkezi log sistemine gönderilip ilgili güvenlik ekiplerine bilgi verebilir.

Kişisel verilerin yasal olmayan yollar ile elde edilmesinin artması ve bu konudaki regülatif düzenlemeler ile ifşaların daha çok gündeme gelmesi ile kullanıcıların klavye hareketlerinin analiz edilmesini istememesi çalışmanın daha büyük datasetlere ulaşmasında sorun oluşturmuştur.

Küçük bir örneklemde elde ettiğimiz sonuçlara bakarak çalışmanın daha büyük datasetler ile yapılarak siber güvenliği arttırmak için umut verici sonuçlar doğurabileceğini gördük. Bu çalışmanın yukarıda bahsedilen alanlarda yapılacak geniş çaplı güvenlik süreçlerine referans olacağı beklenmektedir.

KAYNAKLAR

Di Tommaso, F., Guerra, M., Martinelli, F., Mercaldo, F., Piedimonte, M., Rosa, G., Santone, A., 2019. User Authentication Through Keystroke Dynamics by Means of Model Checking: A Proposal. In 2019 IEEE International Conference on Big Data (Big Data) (PP. 6232-6234). IEEE.

Doğan, F., Türkoğlu, İ., 2019. Derin Öğrenme Modelleri ve Uygulama Alanlarına İlişkin Bir Derleme. Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi, 10(2), 409-445.

Güneş, F., Işık, A. D., 2018. Türkçede Sık Kullanılan Harfler ve Öğretilmesi . The Journal of Limitless Education and Research , 3 (1) , 1-26 . DOI: 10.29250/;Sead.402700.

Juola, P., Noecker, J. I., Stolerman, A., Ryan, M. V., Brennan, P., Greenstadt, R., 2013. Keyboard-Behavior-Based Authentication. IT Professional, 15(4), 8-11.

Mondal, S., Bours, P., 2017. A Study on Continuous Authentication Using a Combination of Keystroke and Mouse Biometrics. Neurocomputing, 230, 1-22.

Monrose, F., Rubin, A. D., 2000. Keystroke Dynamics as a Biometric for Authentication. Future Generation Computer Systems, 16(4), 351-359.

Monrose, F., Reiter, M. K., Wetzel, S., 2002. Password Hardening Based on Keystroke Dynamics. International Journal of Information Security, 1(2), 69-83.

Raul, N., Shankarmani, R., Joshi, P., 2020. A Comprehensive Review of Keystroke Dynamics-Based Authentication Mechanism. In International Conference on Innovative Computing and Communications (PP. 149-162). Springer, Singapore.

Singh, S., 2018. Keystroke Dynamics for Continuous Authentication. In 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (PP. 205-208). IEEE.

Tüfekçi, M., Karpat, F., 2019. Derin Öğrenme Mimarilerinden Konvolüsyonel Sinir Ağları (CNN) Üzerinde Görüntü İşleme-Sınıflandırma Kabiliyetininin Arttırılmasına Yönelik Yapılan Çalışmaların İncelenmesi.

Uzun / Kısa Süreli Bellek (Long / Short Term Memory). 2017. Veri Bilimcisi. Erişim Tarihi: 10.08.2021. https://veribilimcisi.com/2017/09/26/uzun-kisa-sureli-bellek-long-short-term-memory/

Zhu, N., Liu, X., Liu, Z., Hu, K., Wang, Y., Tan, J., ... Guo, Y., 2018. Deep Learning for Smart Agriculture: Concepts, Tools, Applications, and Opportunities.

International Journal of Agricultural and Biological Engineering, 11(4), 32-44.

EKLER

EK A. Kodlar

EK A. Kodlar

- User Modeli

```
public class UserModel
{
    public int Id { get; set; }
    public string UserName { get; set; }
    public List<SessionModel> Sessions { get; set; }
    public DateTime CreatedAt { get; set; }

}
```

- Session Modeli

```
public class SessionModel
{
    public int Id { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public int UserId { get; set; }
    public List<SessionKeyModel> SessionKeys { get; set; }
    public List<SessionCombinationModel> SessionCombinations { get; set; }
    public UserModel User { get; set; }
}
```

- Key Modeli

```
public class KeyModel
{
    public int Id { get; set; }
    public string Code { get; set; }
    public string Name { get; set; }
}
```

- Key Kombinasyon Modeli

```
public class KeyCombinationModel
{
    public int Id { get; set; }
    public int FromKeyId { get; set; }
    public int ToKeyId { get; set; }
    public string FromKey
    {
        get
        {
            return ((KeysList)FromKeyId).GetDescription();
        }
    }
    public string ToKey
    {
        get
        {
            return ((KeysList)ToKeyId).GetDescription();
        }
    }
}
```

- Session Key Modeli

```
public class SessionKeyModel
{
    public int Id { get; set; }
    public int HoldTimesCount { get; set; }
    public int HoldTimesAvg { get; set; }
    public int SessionId { get; set; }
    public int KeyId { get; set; }
    public List<HoldTimeNumberModel> HoldTimeNumbers { get; set; } = new
List<HoldTimeNumberModel>();
    public SessionModel Session { get; set; }
    public KeyModel Key { get; set; }
}
```

- Hold Time Number Modeli

```
public class HoldTimeNumberModel
{
    public int Id { get; set; }
    public int Value { get; set; }
    public int SessionKeyId { get; set; }
    public int KeystrokeDataId { get; set; }
    public SessionKeyModel SessionKey { get; set; }
}
```

- Session Kombinasyon Modeli
-

```
public class SessionCombinationModel
{
    public int Id { get; set; }
    public int SeekTimesCount { get; set; }
    public int SeekTimesAvg { get; set; }
    public int SessionId { get; set; }
    public int KeyCombinationId { get; set; }
    public List<SessionCombinationNumberModel> SessionCombinationNumbers { get; set; } =
new List<SessionCombinationNumberModel>();
    public KeyCombinationModel KeyCombination { get; set; } = new KeyCombinationModel();
    public SessionModel Session { get; set; }
}
```

- Session Combination Number Modeli
-

```
public class SessionCombinationNumberModel
{
    public int Id { get; set; }
    public int Value { get; set; }
    public int SessionCombinationId { get; set; }
    public SessionCombinationModel SessionCombination { get; set; }
}
```

- Tuş ekleme fonksiyonu

```
        public void Key_Insert(KeysList key)
        {
            using (IDbConnection connection = new
SqlConnection(GlobalConfig.CnnString(databaseName)))
            {
                var p = new DynamicParameters();
                p.Add("@Id", (int)key);
                p.Add("@Code", key.ToString());
                p.Add("@Name", key.GetDescription());
                connection.Execute("dbo.spKeys_Insert", p, commandType:
CommandType.StoredProcedure);
            }
        }
```

- User Ekleme Fonksiyonu

```
    public void User_Insert(UserModel model)
    {
        using (IDbConnection connection = new
SqlConnection(GlobalConfig.CnnString(databaseName)))
        {
            var p = new DynamicParameters();
            p.Add("@UserName", model.UserName);
            p.Add("@id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output);
            connection.Execute("dbo.spUsers_Insert", p, commandType:
CommandType.StoredProcedure);
            model.Id = p.Get<int>("@id");
        }
    }
```

- Session Tuşu Ekleme Fonksiyonu

```
    public void SessionKeys_Insert(SessionKeyModel model)
    {
        using (IDbConnection connection = new
SqlConnection(GlobalConfig.CnnString(databaseName)))
        {
            var p = new DynamicParameters();
            p.Add("@HoldTimesCount", model.HoldTimesCount);
            p.Add("@HoldTimesAvg", model.HoldTimesAvg);
            p.Add("@KeyId", model.KeyId);
            p.Add("@SessionId", model.Session.Id);
            p.Add("@id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output);
            connection.Execute("dbo.spSessionKeys_Insert", p, commandType:
CommandType.StoredProcedure);
            model.Id = p.Get<int>("@id");
        }
        foreach (var ht in model.HoldTimeNumbers)
        {
            HoldTimeNumbers_Insert(ht);
```

- Session Tuş Kombinasyonu Ekleme Fonksiyonu

```
        public void SessionCombinations_Insert(SessionCombinationModel model)
        {
            using (IDbConnection connection = new
SqlConnection(GlobalConfig.CnnString(databaseName)))
            {
                var p = new DynamicParameters();
                p.Add("@SeekTimesCount", model.SeekTimesCount);
                p.Add("@SeekTimesAvg", model.SeekTimesAvg);
                p.Add("@SessionId", model.Session.Id);
                p.Add("@FromKeyId", model.KeyCombination.FromKeyId);
                p.Add("@ToKeyId", model.KeyCombination.ToKeyId);
                //p.Add("@KeyCombinationId", model.KeyCombination.Id);
                p.Add("@id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output);
                connection.Execute("dbo.spSessionCombinations_Insert", p, commandType:
CommandType.StoredProcedure);
                model.Id = p.Get<int>("@id");
            }
            foreach (var st in model.SessionCombinationNumbers)
            {
                SeekTimeNumbers_Insert(st);
            }
        }
```

- Tuşlar arası geçiş zamanı ekleme fonksiyonu
-

```
        public void SeekTimeNumbers_Insert(SessionCombinationNumberModel model)
        {
            using (IDbConnection connection = new
SqlConnection(GlobalConfig.CnnString(databaseName)))
            {
                var p = new DynamicParameters();
                p.Add("@SessionCombinationId", model.SessionCombination.Id);
                p.Add("@Value", model.Value);
                p.Add("@id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output);
                connection.Execute("dbo.spSessionCombinationNumbers_Insert", p, commandType:
CommandType.StoredProcedure);
                model.Id = p.Get<int>("@id");
            }
        }
```

- Tuş kombinasyonları ekleme fonksiyonu
-

```
        public void KeyCombinations_Insert(int fromId, int toId)
        {
            using (IDbConnection connection = new
SqlConnection(GlobalConfig.CnnString(databaseName)))
            {
                var p = new DynamicParameters();
                p.Add("@FromKeyId", fromId);
                p.Add("@ToKeyId", toId);
                p.Add("@id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output);
                connection.Execute("dbo.spKeyCombinations_Insert", p, commandType:
CommandType.StoredProcedure);
                //model.Id = p.Get<int>("@id");
```

- Basılı Tutma Zamanı Ekleme Fonksiyonu
-

```
        public void HoldTimeNumbers_Insert(HoldTimeNumberModel model)
        {
            using (IDbConnection connection = new
SqlConnection(GlobalConfig.CnnString(databaseName)))
            {
                var p = new DynamicParameters();
                p.Add("@SessionKeyId", model.SessionKey.Id);
                p.Add("@Value", model.Value);
                p.Add("@id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output);
                connection.Execute("dbo.spHoldTimeNumbers_Insert", p, commandType:
CommandType.StoredProcedure);
                model.Id = p.Get<int>("@id");
            }
        }
```

- Session Ekleme Fonksiyonu
-

```
        public void Sessions_Insert(SessionModel model)
        {
            using (IDbConnection connection = new
SqlConnection(GlobalConfig.CnnString(databaseName)))
            {
                var p = new DynamicParameters();
                p.Add("@StartTime", model.StartTime);
                p.Add("@EndTime", model.EndTime);
                p.Add("@UserId", model.User.Id);
                p.Add("@id", 0, dbType: DbType.Int32, direction: ParameterDirection.Output);
                connection.Execute("dbo.spSession_Insert", p, commandType:
CommandType.StoredProcedure);
                model.Id = p.Get<int>("@id");
            }
            foreach (var sessionKey in model.SessionKeys)
            {
                SessionKeys_Insert(sessionKey);
            }
            Console.WriteLine("keys finishid");
            Console.WriteLine(DateTime.Now.ToString("h:mm:ss"));
            foreach (var sessionCombination in model.SessionCombinations)
            {
                SessionCombinations_Insert(sessionCombination);
            }
            Console.WriteLine("seek finishid");
            Console.WriteLine(DateTime.Now.ToString("h:mm:ss"));
```

- Session Tuşları Ekleme Sorgusu

```
ALTER PROCEDURE [dbo].[spSessionKeys_Insert]
        @HoldTimesCount int,
        @HoldTimesAvg int,
        @KeyId int,
       @SessionId int,
       @id int = 0 output
AS
BEGIN
                insert into dbo.SessionKeys(HoldTimesCount,HoldTimesAvg,
KeyId, SessionId)
                values (@HoldTimesCount,@HoldTimesAvg , @KeyId,
@SessionId);
                select @id = SCOPE_IDENTITY();
END
```

- Session Kombinasyonları Ekleme Sorgusu

```
ALTER PROCEDURE [dbo].[spSessionCombinations_Insert]
        @SeekTimesCount int,
        @SeekTimesAvg int,
        @SessionId int,
        @FromKeyId int,
        @ToKeyId int,
        --@KeyCombinationId int,
        @id int = 0 output
AS
BEGIN
                insert into dbo.SessionCombinations(SeekTimesCount,
SeekTimesAvg, SessionId, KeyCombinationId)
                values (@SeekTimesCount, @SeekTimesAvg, @SessionId,
(Select kc.Id from dbo.KeyCombinations kc where kc.FromKeyId = @FromKeyId
AND kc.ToKeyId = @ToKeyId));
                select @id = SCOPE_IDENTITY();
END
```

- Session kombinasyonları numaları ekleme sorgusu
-

```
ALTER PROCEDURE [dbo].[spSessionCombinationNumbers_Insert]
        @Value int,
        @SessionCombinationId int,
        @id int = 0 output
AS
BEGIN
                insert into dbo.SessionCombinationNumbers(Value,
SessionCombinationId)
                values (@Value, @SessionCombinationId);
                select @id = SCOPE_IDENTITY();
END
```

- Session Ekleme Sorgusu

```
ALTER PROCEDURE [dbo].[spSession_Insert]
        @StartTime datetime2(7),
        @EndTime datetime2(7),
        @UserId int,
        @id int = 0 output
AS
BEGIN
                insert into dbo.Sessions(StartTime, EndTime, UserId)
                values (@StartTime, @EndTime, @UserId);
                select @id = SCOPE_IDENTITY();
END
```

- Tuş kombinasyonları ekleme sorgusu
-

```
ALTER PROCEDURE [dbo].[spKeyCombinations_Insert]
        @FromKeyId int,
        @ToKeyId int,
        @id int = 0 output
AS
BEGIN
                insert into dbo.KeyCombinations(FromKeyId, ToKeyId)
                values (@FromKeyId, @ToKeyId);
                select @id = SCOPE_IDENTITY();
END
```

- Basılı tutma zamanı numarası ekleme sorgusu
-

```
ALTER PROCEDURE [dbo].[spHoldTimeNumbers_Insert]
        @Value int,
        @SessionKeyId int,
        @id int = 0 output
AS
BEGIN
                insert into dbo.HoldTimeNumbers(Value, SessionKeyId)
                values (@Value, @SessionKeyId);
                select @id = SCOPE_IDENTITY();
END

ALTER PROCEDURE [dbo].[spKeys_Insert]
        @Id int,
        @Code nvarchar(50),
        @Name nvarchar(50)
AS
BEGIN
                insert into dbo.Keys(Id,Code, Name)
                values (@Id,@Code, @Name);
END
```

- Tuş bilgileri ekleme SQL sorgusu
-

- Algoritmaya ait örnek kod parça
-

```
def rnn_model(X, y):
    labelencoder_X_1 = LabelEncoder()
    y = labelencoder_X_1.fit_transform(y)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = TEST_SIZE,
random_state = 1)
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)

    X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
    X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))

    model = Sequential()
    model.add(LSTM(128, input_shape=(X_train[1].shape), activation='relu',
return_sequences=True))
    model.add(LSTM(64,activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(2, activation='softmax'))

    # Compile model
    model.compile(
        loss='sparse_categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy']
    )
    print(model.summary())
```

- Parola tahmin algoritma kodu

```
static char[] chars = new char[] { 'E', 'I', 'A', 'L', 'G', 'R', 'S', 'M', 'N', 'T', 'e',
'i', 'a', 'l', 'g', 'r', 's', 'm', 'n', 't' ,' '};
        static char[] pass = new char[8];
        static void Try(string hash, int num, CancellationToken cancellationToken)
        {
            int a = num;
            for (int b = 0; b < chars.Length-1; b++)
            {
                for (int c = 0; c < chars.Length-1; c++)
                {
                    for (int d = 0; d < chars.Length-1; d++)
                    {
                        for (int e = 0; e < chars.Length-1; e++)
                        {
                            for (int f = 0; f < chars.Length-1; f++)
                            {
                                for (int g = 0; g < chars.Length-1; g++)
                                {
                                    for (int h = 0; h < chars.Length-1; h++)
                                    {
                                        pass[0] = chars[a];
                                        pass[1] = chars[b];
                                        pass[2] = chars[c];
                                        pass[3] = chars[d];
                                        pass[4] = chars[e];
                                        pass[5] = chars[f];
                                        pass[6] = chars[g];
                                        pass[7] = chars[h];
                                        byte[] data =
System.Text.Encoding.ASCII.GetBytes(pass);
                                        data = new
System.Security.Cryptography.SHA256Managed().ComputeHash(data);
```

ÖZGEÇMİŞ

Adı Soyadı : Nurgül Akşit

Eğitim Durumu

Lise : STFA Anadolu Teknik Lisesi, 2010

Lisans : Sakarya Üniversitesi, Bilgisayar ve Bilişim Bilimleri Fakültesi, Bilgisayar Mühendisliği Bölümü, 2016

Yayınlar

Aksit, N., Zaim A.H. 2022. Siber Güvenlikte Klavye Davranış Analizi, İstanbul Ticaret Üniversitesi Teknoloji ve Uygulamalı Bilimler Dergisi, Basımda.