# PRIMJENA BIHEVIORALNE BIOMETRIJE ZA SIGURAN PRISTUP INFORMACIJSKOM SUSTAVU

**Marenković, Nadan**

**Undergraduate thesis / Završni rad**

**2023**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Algebra University College / Visoko učilište Algebra**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:225:871340

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-12-02**

ALGEBRA SVEUČILIŠTE

*Repository / Repozitorij:*

Algebra Univerity - Repository of Algebra Univerity

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

**VISOKO UČILIŠTE ALGEBRA**

ZAVRŠNI RAD

# PRIMJENA BIHEVIORALNE BIOMETRIJE ZA SIGURAN PRISTUP INFORMACIJSKOM SUSTAVU

Nadan Marenković

Zagreb, veljača 2023.

# Predgovor

Ovim putem zahvaljujem se mentoru Zlatanu Moriću na smjernicama, znanju i savjetima pri izradi završnog rada i dodijeljenom vremenu za mentoriranje.

Također, zahvaljujem se svim svojim prijateljima i kolegama koji su mi pomogli u prikupljanju podataka za praktičnu izvedbu završnog rada.

Posebnu zahvalnost iskazujem svojoj užoj obitelji, svojoj zaručnici i njenoj obitelji na potpori.

# Sažetak

Jednostruka autentikacija putem korisničke zaporke se koristi kako bi identificirali korisnika prema onome što korisnik zna, a to je korisničko ime, zaporka ili PIN. Takvim načinom autentikacije korisnički račun i informacijski sustav na koji se korisnik prijavljuje je izložen ranjivošću i ugrozama. Ovaj rad je usmjeren na izradu koncepta proširenja jednostruke autentikacije korisničkom zaporkom i tehnikom bihevioralne biometrije dinamikom tipkanja na tipkovnici. Dinamikom tipkanja proširujemo autentikaciju putem zaporke na način da autentikacijski sustav pamti dinamiku tipkanja zaporke korisnika. Korištenjem klasifikacijskih algoritama strojnog učenja, višeklasnom klasifikacijom se može utvrditi da li korisnik koji je upisao zaporku zaista vlasnik korisničkog računa. U ovom radu opisati će se rad računalne aplikacije koja ima funkcionalnost zapisivanja korisničkog unosa u kojem se zahtjeva da korisnik upiše zadani tekst. Aplikacija mjeri koliko je korisniku trebalo da upiše specifičan tekst u sučelje aplikacije, koji je vremenski razmak između svake tipke i da li korisnik koristi specijalne znakove na tipkovnici. Dobiveni parametri se koriste za treniranje klasifikacijskih algoritama koji mogu izračunati vjerojatnost da li korisnik koji upisuje zadani tekst u sučelje vlasnik trenutnog korisničkog računa.

**Ključne riječi:** autentikacija, biometrija, strojno učenje, zaporka.

# Abstract

Single-factor Authentication using password is the simplest way to authenticate user by the pass phrase user knows which is username, password or PIN. Since single-factor authentication is simple, it may expose vulnerabilites to a user account or entire information system. This final paper is used to describe a proof of concept application for extending single-factor authentication with bihevoral biometric technique named keystroke dynamic. With keystroke dynamics we can extend authentication by password so that the authentication system saves and remembers keystroke dynamic for the given user. Using machine learning classification algorithms, multiclass classification can determine was the person who typed specific text account owner, and then authenticate the user. This final paper will describe functionallity of the computer application which has the capability of storing user input. Computer application measures the time user was typing to get the correct text, time of release between the keys, and the use of special characters. Used input parameters are sent to classification machine learning algorithms which can calculate the probabillity of user typing being the account owner.

**Key words:** authentication, biometrics, machine learning, password.

# Sadržaj

# 1. Uvod

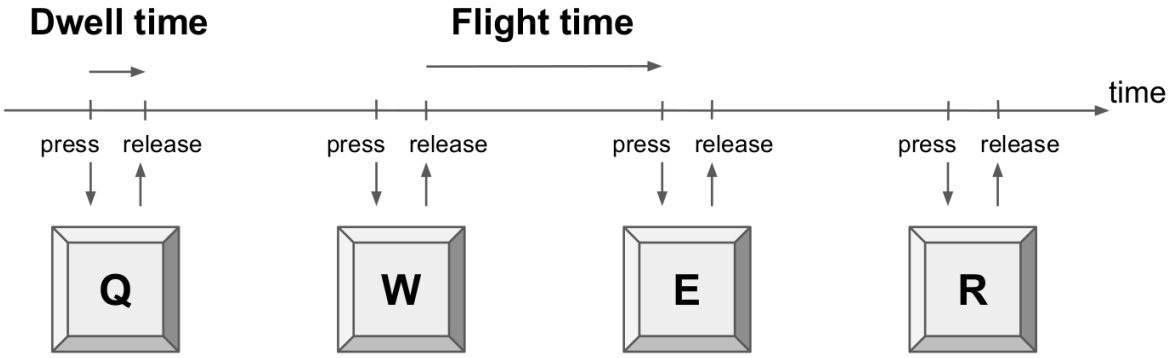
Bihevioralna biometrija je disciplina koja analizira fizičko i kognitivno ponašanje korisnika u informacijskom sustavu. Riječ biometrija dolazi od dvije riječi: „Bio“ što znači život i „Metrics“ što znači mjerenje. Korisnici se na različit način ponašaju u radu s računalom te svojim ponašanjem stvaraju svoj digitalni identitet. Koristeći algoritme strojnog učenja moguće je analizirati ponašanje korisnika u informacijskom sustavu. Uzorci korisničkog ponašanja se mogu pratiti pomoću perifernih uređaja spojenih na računalo. Primjeri ponašanja su: brzina pomicanja računalnog miša, korištenje određenih tipki na računalnom mišu, pritisak i kretanje na dodirniku. Kod korištenja tipkovnice, može se analizirati koje specijalne znakove korisnik koristi, vremenski raspon držanja tipke pritisnute, vremenski razmak između pritisnutih tipki. Klasifikacijskim algoritmima strojnog učenja moguće je analizirati podatke dinamike tipkanja i ustanoviti da li se informacijskim sustavom koristi korisnik koji je ujedno i vlasnik svojeg korisničkog računa, ili netko drugi. [1] Računalne aplikacije koje imaju korisničko sučelje u kojem korisnik može upisivati tekst ili koristiti računalni miš mogu programskim kodom pratiti podatke dobivene putem perifernih uređaja kao što su tipkovnica i računalni miš. Programski kod komunicira sa operacijskim sustavom kako bi dobio informacije o korištenju perifernih uređaja. Postoje računalni programi poput pretraživača interneta koji neovisno o vrsti informacijskog sustava mogu dohvatiti podatke korisnika pretraživača. Podaci dobiveni u računalnoj aplikaciji se koriste za treniranje i testiranje klasifikacijskog algoritma kako bi mogli izvoditi binarnu ili višeklasnu klasifikaciju.

# 2. Bihevioralna biometrija dinamikom tipkanja

Dinamikom tipkanja se može ustanoviti identitet pojedinca koji koristi računalo. Podaci koji se analiziraju dobiveni su korisničkim unosom putem tipkovnice. Informacije dobivene dinamikom tipkanja kategoriziraju se u dva dijela:

1. Vremenski raspon držanja tipke pritisnutom (engl. *Dwell time*)
2. Vremenski raspon između otpuštanja tipke i pritiskom tipke (engl. *Flight time*)

Dinamika tipkanja ovisi o pojedincu te se razlikuje između pojedinaca. Na primjer, osoba koja ima izražene računalne vještine će prije naći određenu tipku na tipkovnici i pritisnuti tipku i upisati traženi tekst u računalo. Dinamika tipkanja također ovisi o faktorima kao što su: starost, invalidnost, emocionalno stanje osobe, umor.



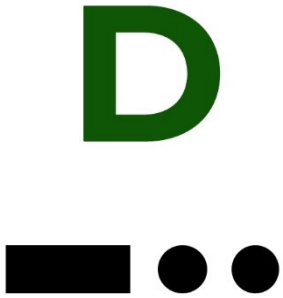

Slika 1. Prikaz vremenskih raspona između pritiskanja i otpuštanja tipki na tipkovnici[1]

## 2.1. Povijest bihevioralne biometrije dinamikom tipkanja

Sredinom 19. stoljeća, operateri telegrafskog stroja su mogli razmjenjivati poruke u obliku morseovog koda. Nakon određenog iskustva u korištenju telegrafa, operater bi stvorio određenu dinamiku tipkanja na telegrafu i s time stvorio unikatan potpis koji je dio njegovog identiteta. Koristeći svoju dinamiku tipkanja, operator bi se mogao identificirati sa drugim operatorom s kojim razmjenjuje poruke. Takvim načinom tipkanja osnovana je nova vrsta disciplina zvana bihevioralna biometrija. Tijekom drugog svjetskog rata se koristio telegraf

[1]https://blog.daftcode.pl/building-supervised-models-for-user-verification-part-1-of-the-tutorial-7496d5d394b9

i morseov kod kako bi zapovjednici i vojnici mogli razmjenjivati poruke. Znamenka koja je mogla biti slovo, broj ili znak interpunkcije su se mogli interpretirati u poslanoj poruci.




Slika 2. Slika predstavlja interpretirano slovo abecede morseovim kodom[2]

Na slici 2. crni krugovi prikazuju jedan kratak pritisak tipke na telegrafu, dok crni kvadrat prikazuje držanje tipke na telegrafu na engleskom jeziku zvan „dwell time“. Prostor koji se nalazi između crnih objekata na slici je vremenski period kada je tipka otpuštena „flight time“. Pomoću tih informacija se mogla ustanoviti dinamika tipkanja operatera. Operateri telegrafa bi mjerili razmak između poslanih poruka i razmak između signala poslanih porukom kako bi identificirali operatora telegrafa odnosno da li operater pripadnik savezničkih snaga. Identitet operatera je bio tajan i nepoznat operaterima i vojnom osoblju s kojima komunicira, a razlog tome je ukoliko operater bude uhićen od strane neprijateljske vojne sile, da ne odaje poruke koje su poslane. Naziv metode identifikacije pošiljatelja poruke u drugom svjetskom ratu nazvana je „Šaka pošiljatelja“ (engl. *The fist of the sender*) koja je među prvim metodama dinamike tipkana za identifikaciju osobe. [2]

## 2.2. Bihevioralna biometrija korištenjem računalnog miša

Nadziranjem kretanja računalnog miša po korisničkom sučelju moguće je stvoriti digitalni identitet korisnika koji se može koristiti za autentikaciju. [3] Parametri koji se mogu uzeti za mjerenje su:

[2] https://militarymachine.com/d-morse-code/

- Kretanje miša po podlozi
- Akcija povuci i ispusti (eng. *Drag and drop*)
- Akcija pokaži i klikni (eng. *Point and click*)
- Aktivnosti kotačića na mišu
- Mirnoća

Za svaku od ovih akcija moguće je stvoriti značajke poput prosječna brzina kretanja miša na način da se mjeri vrijeme i udaljenost po X i Y koordinatama pokazivača na ekranu. Identifikacija korisnika računalnim mišem ovisi o vrsti miša koji se koristi. Na identifikaciju utječu parametri vrsta podloge za miš i da li je miš analogni ili optički iz razloga što optički miš ne radi ispravno kada se nalazi nad površinom visokog sjaja.

## 2.3. Primjena bihevioralne biometrije

Danas se bihevioralna biometrija koristi u bankarskom sektoru i financijskim institucijama. Jedan od primjera je samostalno otvaranje računa u banci putem interneta gdje se na web stranici prati brzina tipkanja i klikanja miša koji predstavljaju indikatore dobrog ili lošeg ponašanja na stranici. Kod web i mobilnih aplikacija u sučelju za autenkaciju korisnika za prijavu na aplikaciju za pregled računa, nadzire se kompletan proces od prijave do odjave korisnika. [4] Najveći problem kod jednostruke autentikacije je socijalno inženjerstvo. Kriminalac koji želi doći do podataka za autentikaciju najčešće koriste metode poput telefonskih poziva ili slanja lažnih web stranica kako bi iz njih izvukao podatke pretvarajući se da je napadač osoba sa autoritetom. Najčešće se napadači predstavljaju kao bankarski službenici ili zaposlenici državnih agencija. Potom traže od žrtve da se prijave u bankarski sustav kako bi od njih izvukli novčana sredstva ili informacije. S obzirom da se cijeli proces odvija u žurbi, korisnik bankarskog sustava izvodi nepredvidiva ponašanja, te se u informacijskom sustavu pojavljuje indikator da je osoba prijavljena u sustav pod prisilom. Još jedan problem s kojim se suočavaju banke je prijenos velikih svota novca sa jednog računa na drugi putem digitalnog bankarstva kako bi se izvodile ilegalne aktivnosti novčanim sredstvima. Osobe koji prenose ogromne količine novca se nazivaju „mulama“ i pod utjecajem su kriminalaca koji im za prenesen novac isplate jedan dio sume novca. Tijekom pandemije zabilježen je visok promet prijenosa novaca putem digitalnog bankarstva s jednog računa na druge račune. Iz tog razloga pokrenuta je inicijativa razvoja i upotrebe

bihevioralne biometrije koja analizom aktivnosti računa može prepoznati mulu koja prenosi novac i zabilježiti sumnjivo ponašanje kod novčanih transakcija.

## 2.4. Druge značajne metode biometrije

Biometrija kao metoda autentikacije se dijeli na fizičku i bihevioralnu. Kod fizičke biometrije pojedinac se razlikuje prema njegovim fizičkim karakteristikama kao što su lice, otisak prsta, mrežnica oka i geometrija ruke dok kod bihevioralne biometrije karakteristike pojedinca su definirane na način kako se pojedinac ponaša a to su: glas pojedinca, hod, dinamika tipkanja i pokretanja miša. Kod fizičke biometrije teško je promijeniti svojstva pojedinca jer je ograničen vlastitom biologijom, dok je bihevioralna promjenjiva jer je definirana ljudskim navikama. [4]

| Vrsta biometrije | Tip | Prihvaćenost korisnika | Pouzdanost | Univerzalnost |
|---|---|---|---|---|
| Prepoznavanje lica | Fizička | Srednje | Visoka | Visoka |
| Prepoznavanje glasa | Bihevioralna | Visoka | Srednja | Srednja |
| Otisak prsta | Fizička | Srednje | Visoka | Srednja |
| Potpis osobe | Bihevioralna | Visoka | Srednja | Niska |
| Sken mrežnice | Fizička | Srednje | Visoka | Visoka |
| Hod osobe | Bihevioralna | Visoka | Visoka | Srednja |
| Dinamika tipkanja | Bihevioralna | Srednje | Srednje | Niska |
| Geometrija dlana | Fizička | Srednje | Srednje | Srednje |

Tablica 1. Korištene metode biometrije

U većini metoda biometrije identifikacija se sastoji od četiri glavna postupka. Prvi postupak je registracija koji služi kako bi se prikupili podaci korisnika ovisno o kojoj metodi se radi. Time se stvara šablona digitalnog identiteta osobe. Drugi postupak je obrada podataka u kojoj se brišu beskorisni podaci korisnika koji bi utjecali na degradaciju procesa identifikacije. Treći postupak je izdvajanje značajki u kojem je bitno precizno i efikasno izdvojiti značajke koje tvore digitalni identitet. Četvrti postupak je podudaranje koji se koristi pri autentikaciji korisnika u kojem podaci registriranog korisnika se procjenjuju sa korisnikom u trenutnom postupku autentikacije. Trenutno najkorištenije metode biometrije su prepoznavanje otiska prsta i lica osobe, međutim napredovanjem sigurnosnih sustava bihevioralne metode postaju popularnije iz razloga što ih je jeftinije za instalirati te imaju visoku prihvaćenost korisnika.

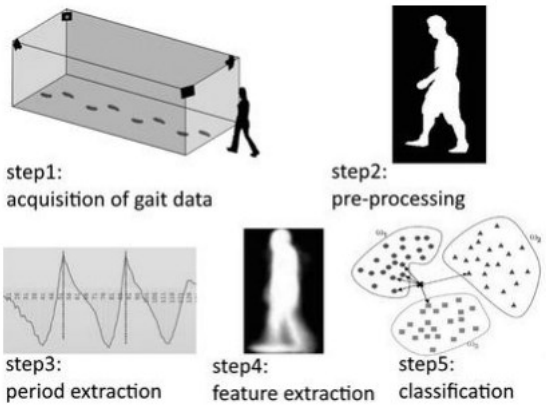
### 2.4.1. Metoda prepoznavanja glasa

Metoda prepoznavanjem glasa nastala je 1960-tih u Texasu. Od tada ova metoda je prikupila pažnju znanstvenika iz razloga što ljudski glas sadrži puno informacija koje se mogu analizirati. Unatoč što se ova metoda smatra bihevioralnom metodom, postoje biološki faktori koji nisu lako promjenjivi a odnose se na poziciju organa kao što su jezik, grlo, glasnice, nepce i usne koji se smatraju da su jedinstveni kod ljudi. Autentikacija prepoznavanjem glasa započinje kada korisnik ispusti glas usmjeren prema analogno digitalnom pretvaraču, a nakon toga proces mjerenja započinje na način da se ekstrakcijom dobivene frekvencije izvodi segmentacija valova između intervala. Nakon toga započinje se proces podudaranja pomoću dobivenih zvučnih valova i valova koji su dobiveni pri registraciji korisnika. Prednost ovakve metode je pouzdanost i prihvaćenost od strane korisnika. Metoda je implementirana u nekoliko sektora poput bankarstva putem telefonskog poziva, industrija video igara, industrija uređaja za kućanstvo poput televizora i mobilnih uređaja. U području forenzike i nadzora se također koristi navedena metoda. Nedostatak ove tehnologije snižena pouzdanost pri autentikaciji korisnika ukoliko korisnik ima problema sa izgovaranjem glasa što može nastati infekcijom ili starenjem.

### 2.4.2. Metoda prepoznavanja hoda

Prepoznavanje ljudskog hoda je moderna metoda identifikacije koja je visoko pouzdana i prihvaćena metoda. Metoda se izvodi na analizi stila hodanja i strukturi tijela. Kako bi

stvorili jedinstveni identitet, potrebno je promatrati balansirano kretanje udova kod osobe u ciklusima.



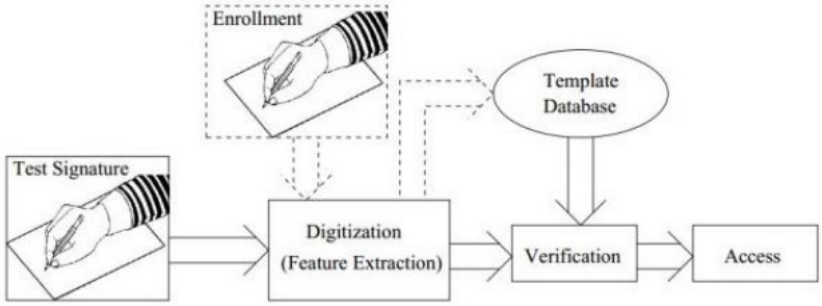

Slika 3. Dijagram prepoznavanja hoda[3]

Postupak je opisan u pet koraka gdje se u prvom koraku prikazuje prikupljanje podataka ljudskog hoda vizualnim senzorom, potom drugi korak prikazuje obradu podataka. Treći i četvrti korak se odnosi na odabir značajki kako bi dobili što precizniju procjenu, nakon toga dolazi postupak klasifikacije. Danas postoje aplikacije na mobilnim uređajima koje mogu analizirati ljudski hod vizualnim senzorima. Danas je metoda prepoznavanja hoda poželjna metoda kod autentikacije u unutarnjim i vanjskim prostorima gdje se nalazi puno ljudi. Na većim udaljenostima, prepoznavanje hoda radi bolje za razliku pod prepoznavanja glasa, te nije potreban input niti dozvola od strane korisnika. Također jedna od prednosti ove metode je da se ista može upotrijebiti koristeći CCTV (eng. *Closed-circuit television*) kamere za nadzor pa makar one imale nisku rezoluciju. Nedostatak ove tehnologije je da nije pouzdana kod ljudi koji nose težak teret iz razloga što se ljudski hod tada deformira. Također različita odjeća i obuća utječe na ljudski hod.

### 2.4.3. Metoda prepoznavanja potpisa osobe

Analiza ručnog potpisa osobe se već duže smatra načinom verificiranja osobe i ima široku primjenu u svim industrijama. Način na koji se može digitalno analizirati potpis osobe je da se potpisivanje izvrši nad površinom dodirnika uređaja računalnom olovkom.

[3] „Study On Most Popular Behavioral Biometrics, Advantages, Disadvantages And Recent Applications : A Review“ – stranica 17.




Slika 4. Metoda provjera potpisa[4]

U prvom potpisu korisnik koji nije upisan u bazu izrađuje testni potpis, a potom još jednom upisuje potpis koji se u drugom koraku digitalizira odnosno pretvara se u sliku. Nakon toga se niskim mjerenjima izrađuju značajke koje se spremaju u bazu podataka koje stvaraju digitalni identitet osobe. Kod procesa identifikacije korisnik se treba potpisati kako bi se trenutni potpis procijenio sa potpisom spremljenim u bazi podataka. Jedna od zanimljivih značajki koja je ujedno i prednost ove metode je da osoba koja nije pri svijesti ne može dati svoj potpis za razliku od nekih drugih biometrijskih metoda poput otiska prsta u kojem se otisak prsta osobe ili sken mrežnice može dobiti od osobe koja nije pri svijesti. Nedostatak ove metode je da sa vremenom osoba modificira svoj potpis koji više nije valjan kod procesa identifikacije. U nekim informacijskim sustavima poput sustava za upravljanje bazama podataka i sustavima za računalstvo u oblaku je uveden postupak analize potpisa osobe iz razloga što upravljanje i donošenje odluka u takvim sustavima može imati katastrofalne posljedice neodobrenim upravljanjem.

[4] „Study On Most Popular Behavioral Biometrics, Advantages, Disadvantages And Recent Applications : A Review“ – stranica 18.

# 3. Aplikacija za praćenje dinamike tipkanja

Računalna aplikacija koja se koristi u ovom radu izrađena je u **Windows Presentation Foundation** (skraćeno WPF) programskom okviru za izradu aplikacija u „Windows“ operacijskom sustavu. [5] Navedeni programski okvir je dio skupa alata razvijen od strane kompanije Microsoft 2006. godine. WPF programski okvir razdvaja programsku logiku koja je pisana u programskom jeziku **C#** i grafičko sučelje koje je napisano u opisnom jeziku **XAML** koji određuje vrstu i poziciju elemenata na prozoru aplikacije. WPF je izabran za izradu ove aplikacije iz razloga što je pouzdan i održavan programski okvir koji ima sve potrebne alate kako bi funkcionalnosti mogle biti izrađene, a iste funkcionalnosti korisne i pouzdane. Jedan dio aplikacije sadrži sučelje za interakciju s korisnicima, a drugi dio aplikacije sadrži programsku logiku. Aplikacija ima mogućnost istovremeno upravljati korisničkim sučeljem dok pokreće programsku logiku u pozadini programa, a takvo svojstvo je omogućeno upravljanjem dretvama koje se nalaze unutar jednog procesa. Kada se aplikacija pokrene putem uređivača programskog koda, u operacijskom sustavu pokreće se novi prozor koji u sebi ima korisničko sučelje. U određenim funkcionalnostima uvedena je mogućnost višezadaćnosti kako bi aplikacija radila u stvarnom vremenu. Cilj aplikacije je štopericom snimati unos zadane riječi i znakova korisnika dok korisnik ne utipka zadanu riječ. Nakon što korisnik unese ispravnu riječ, štoperica prestaje sa snimanjem i aplikacija sprema informacije u tekstualnu datoteku formata **comma-separated values** (skraćeno CSV). Ovakva vrsta tekstualne datoteke koristi specijalni znak za razdvajanje riječi kako bi se iste riječi mogle pretvoriti u stupce i retke. Ova funkcionalnost u formatu datoteke se koristi kako bi se podaci spremani u datoteku mogli lakše pročitati i obrađivati.

## 3.1. Struktura aplikativnog rješenja

Programski kod organiziran je u klase na način da svaka klasa sadrži svojstva i metode prema funkcionalnostima koje se koriste:

- Model dinamike tipkanja korisnika
- Dodatni alati za upravljanjem CSV datoteka
- Prezentacijski sloj korisničkog sučelja
- Programski sloj aplikativnog rješenja

- Model strojnog učenja

### 3.1.1. Model dinamike tipkanja

Model dinamike tipkanja korisnika opisana je u klasi KeystrokeData.cs te se navedena klasa koristi kao opisna klasa. Neki od svojstava klase su:

```
public bool CapsPressed { get; set; }

public bool BackSpacePressed { get; set; }

public long TotalTime { get; set; }
```

Kôd 1. Svojstva opisne klase dinamike tipkanja

Navedena 3 svojstva služe za primjer što opisujemo i kako svojstva izgledaju i funkcioniraju. Ključna riječ public (hrv. *javno*) se koristi kako bi navedenoj klasi mogla pristupiti neka druga klasa. U aplikaciji koja se koristi, klasa KeystrokeData mora biti javna kako bi joj pomoćna klasa za obradu CSV datoteke mogla pristupiti, uzeti podatke i spremiti ih u CSV datoteku. Druga ključna riječ bool se odnosi na Booleov logički izraz koji vraća rezultat da li je izraz istinit ili neistinit. Dolazi od engleske riječi boolean. Ovom ključnom riječju opisujemo podatkovni tip svojstva varijable u koju se može pospremiti jedna od tvrdnji: istinito (engl. *true*) ili neistinito (engl. *false*). Treća ključna riječ u liniji koda se odnosi na naziv svojstva koji mora biti približno odgovarajuće svojstvu koje opisuje klasu. Četvrta ključna riječ u nizu koda opisuje mogućnost svojstva da može dohvatiti podatke (engl. *get*) i pospremiti jedan podatak (engl. *set*). Kod prvog navedenog svojstva pod imenom CapsPressed nastojimo postaviti podatak Booleovom logikom da li je korisnik pritisnuo tipku CapsLock na tipkovnici dok je upisivao traženu riječ. S time možemo dobiti informaciju da li korisnik koristi navedenu tipku kako bi upisao velika tiskana slova. Drugo svojstvo također ima isti podatkovni tip, te se u njegovo svojstvo upisuje Booleovom logikom da li je korisnik koristio tipku Backspace za brisanje upisanog znaka što može upućivati da korisnik radi greške kod unosa teksta.

### 3.1.2. Alati za upravljanje CSV datotekom

„CsvHelper“ biblioteka je skup alata za čitanje i zapisivanje podataka u datoteke CSV formata putem C# koda. Ova biblioteka nije standardni alat koji dolazi uz WPF programski okvir, već proizvod otvorenog koda koji su pisali volonteri sa ciljem da se stvori i koristi

takav alat besplatno. Kako bi mogli koristiti funkcionalnosti CsvHelper-a, potrebno je dodati referencu biblioteke u aplikaciju, i putem dokumentacije pročitati javne dostupne metode čija prva ključna riječ u metodama počinje sa „public“. U aplikaciji za praćenje dinamike tipkanja izrađena je pomoćna metoda koja omogućava rad sa CsvHelper bibliotekom.

```
using (var stream = File.Open(PATH, FileMode.Append))
            using (var writer = new StreamWriter(stream))
            using (var csv = new CsvWriter(writer, config))
            {
                csv.WriteRecord(data);
                csv.NextRecord();
            }
```

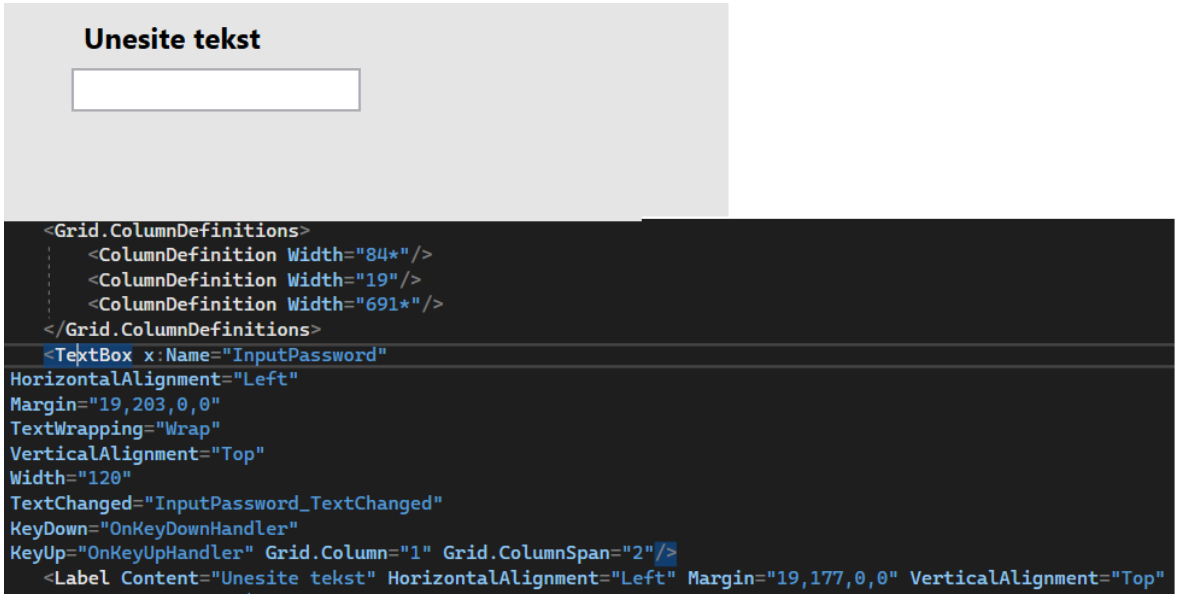
Kôd 2. Korištenje aplikacijskog programibilnog sučelja CsvHelper datoteke

U primjeru koda opisani su postupci zapisivanja podataka unutar CSV datoteke. Prva linija koda nam omogućava otvaranje CSV datoteke i njen način rada da dodaje nove retke u datoteku. Treća linija koda stvara objekt iz vanjske biblioteke CsvHelper kako bi mogli početi zapisivati u CSV datoteku. U petoj liniji koda parametar „data“ sadrži objekt koji je stvoren prema opisnoj klasi KeystrokeData i sadrži podatke o dinamici tipkanja korisnika koji je završio sa unosom tražene riječi. Slijedeća linija postavlja pokazivač u CSV datoteci na novi red.

### 3.1.3. Korisničko sučelje aplikativnog rješenja

Jezik korišten kako bi vizualno prikazao korisničko sučelje je **XAML** (engl. *Extensible Application Markup Language*) – deklarativni označni jezik .NET programskog okvira stvoren sa svrhom kako bi pojednostavnio izradu programskog sučelja u izradi desktop aplikacija . Glavna ideja zasebnog označnog jezika je kako bi odvojili programsku logiku od korisničkog sučelja. XAML označni jezik ima sličnu strukturu poput **XML** (engl. *Extensible Markup Language*) označnog jezika koji je industrijski standard u interoperabilnosti informacijskih sustava za razmjenu informacija. XML jezik može samo sadržavati informacije dok XAML jezik može osim informacija, meta podataka i svojstva podataka sadržavati dinamičke informacije kojima je moguće upravljati kada se program izvršava. Ujedno je moguće dodati događaje (engl. *Event*) na element XAML označnog koda koji

mijenja svoje stanje dogodi li se događaj poput klika tipke miša ili neki drugi događaj perifernih uređaja.




Slika 5. Primjer XAML koda koji oblikuje grafičko sučelje

Na slici se nalazi oznaka pod nazivom „Unesite tekst“ koja je vidljiva na ekranu na način da smo opisnim jezikom XAML upisali naziv objekta pod nazivom **Label** (hrv. *oznaka*). Upisivanjem naziva objekta u uređivač programskog koda određujemo koji element korisničkog sučelja se prikazuje na ekranu. Elementi korisničkog sučelja se definiraju u šiljastim zagradama kao na slici. Osim naziva elementa, moguće je dodati i atribute elementu koji definiraju poziciju elementa na ekranu, sadržaj elementa poput vidljivog teksta, oblika ili boje. Bijela kućica koja je vidljiva na slici je kućica za unos teksta koji unosi korisnik aplikacije. Kako bi mogli stvoriti funkcionalnu kućicu za unos teksta, potrebno je u XAML jeziku upisati oznaku **TextBox.** Osim atributa koji se koriste kako bi element sučelja imao specificiran vizualni oblik i njegovu poziciju na ekranu, također postoje metode i događaji koji se mogu pridodati elementima. U TextBox-u pod nazivom „InputPassword“ postoje tri definirana događaja. Element TextBox ima svojstvo okidanja programskog koda kada se dogodi događaj koji mu je dodijeljen. **TextChanged**, **KeyDown** i **KeyUp** atributi koriste se kako bi zaljepili funkcije koje će se okinuti kada se događaj dogodi. Primjer ukoliko korisnik pritisne bilo koju tipku na tipkovnici, okinuti će se KeyDown događaj koji će proslijediti parametre metodi koja je dodijeljena tom događaju, a to je **OnKeyDownHandler**. Naziv engleskog atributa **x:Name** koristi se kako bi se mogli referencirati na specifičan element podatkovnog sučelja.

### 3.1.4. Programski sloj aplikativnog rješenja

Kako bi korisničko sučelje napisano u XAML jeziku moglo imati funkcionalnosti poput upisa teksta u kućicu za unos teksta i potom obavila pohranu teksta, potrebna je programska logika koja se nalazi u programskom sloju aplikativnog rješenja. Datoteke označnog jezika XAML imaju ekstenziju datoteke pod nazivom **.xaml** dok datoteke u kojima se izvršava programska logika sadržavaju ekstenziju **.cs** što označava da je datoteka pisana u jeziku C#. Način na koji povezujemo obje datoteke je da ima pridodamo isti naziv, te potom u programskoj logici .cs datoteke klasu nazovemo istim imenom kao i XAML datoteku. Ukoliko se izradi XAML datoteka sa nazivom „MainWindow.xaml“ i u njoj se definiraju elementi korisničkog sučelja, potrebno je stvoriti C# datoteku sa nazivom „MainWindow.xaml.cs“. Na taj način .cs datoteka ima pristup elementima korisničkog sučelja kroz njihov x:Name atribut.

```
InputPassword.Text = string.Empty;
```

Prethodna linija koda predstavlja primjer funkcionalnosti referenciranja elementa grafičkog sučelja XAML datoteke u C# programsku datoteku. Prvi izraz do točke u liniji programskog koda označava naziv objekta pod atributom x:Name koji je „InputPassword“, a poslije točke pojam riječi „Text“ označava svojstvo elementa TextBox-a. Text svojstvo elementa se odnosi na tekst koji piše u kućici za upisivanje teksta, a navedena linija koda se koristi da se tekst u kućici postavi na „“ – odnosno prazan tekst. Na taj način brišemo tekst koji se nalazi u kućici.

### 3.1.5. Model strojnog učenja

Programski okvir za izradu modela strojnog učenja u programskom jeziku C# naziva se **ML.NET –** platforma otvorenog koda koja ima mogućnosti izvedbe na različitim operacijskim sustavima. ML.NET omogućava c# programerima izradu, treniranje i postavljanje u produkcijsko okruženje modele strojnog učenja. ML.NET sadrži alat pod nazivom „model builder“ koji se koristi kako bi olakšao izradu modela strojnog učenja. [6] Izrada modela počinje sa stvaranjem nove datoteke sa ekstenzijom **.mbconfig** koja označava datoteku strojnog učenja u jeziku C#. Datoteka se dodaje u projekt koji obuhvaća sve datoteke vezane za C# program koji izrađujemo. Nakon toga odabiremo scenarij koji nam odgovara za automatizirano strojno učenje koje će automatski uzeti nekoliko algoritama koji

spadaju pod određene vrste algoritama koji su kategorizirani prema zadaćama koje izvršavaju:

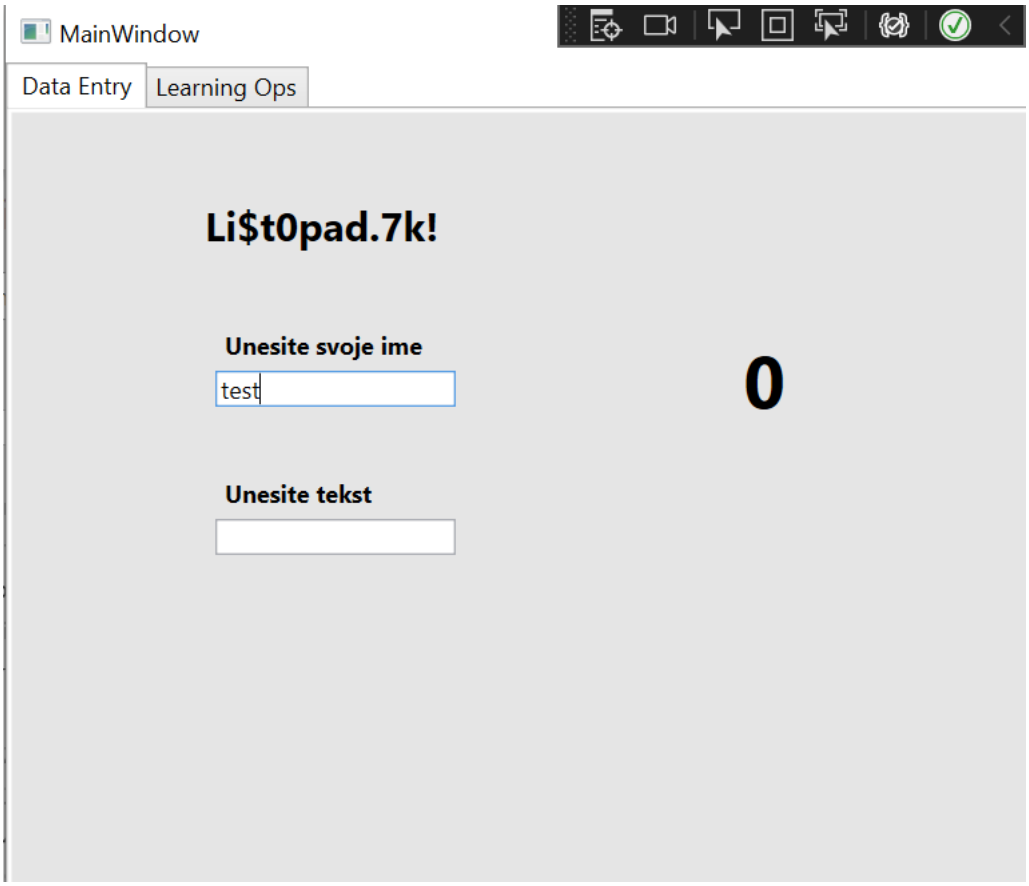
- Tablični podaci
  - Klasifikacija podataka
  - Predviđanje vrijednosti
  - Preporuke
  - Prognoziranje
- Slike
  - Klasifikacija slika
  - Uočavanje objekata
- Tekst
  - Klasifikacija teksta

Nakon odabira scenarija koji je ponuđen, slijedi odabir okruženja (engl. *environment*). Riječ okruženje odnosi se na računalo u kojem će se izvršavati treniranje i procjena odabranog modela strojnog učenja. Postoje dva moguća okruženja u ML.NET-u, a to su lokalno okruženje koje se odnosi na naše računalo, odnosno model strojnog učenja će koristiti resurse našeg računala ukoliko ima uvjete za korištenje. Uvjeti potrebni su dovoljna radna memorija i grafička kartica. Drugo okruženje se odnosi na računalstvo u oblaku (engl. *Cloud computing*) od strane poduzeća Microsoft pod nazivom **Azure** koje omogućava korištenje računalnih komponenti putem internetske mreže. Korisnik se spoji na postojeći Azure korisnički račun i odabire poslužiteljsko računalo koje sadrži računalne komponente jačih kapaciteta nego lokalno stolno ili prijenosno računalo. Nakon odabira okruženja, potrebno je odabrati izvor i format podataka. Kod modela strojnog učenja prema tabličnim podacima, format podataka može biti .csv ili tekstualna datoteka pod ekstenzijom txt. Drugi izvor podataka može biti **SQL Server** relacijska baza podataka stvorena od strane Microsoft-a. Nakon odabira izvora podataka i formata, odabire se oznaka koju će model pokušati predvidjeti. Pojam oznake se odnosi na postojeći stupac unutar tabličnih podataka koji se nastoji predvidjeti. Nakon toga slijedi odabir stupaca koji će se koristiti za treniranje modela strojnog učenja. Stupci unutar tablice koja se koristi kao izvor podataka se dijeli na dva dijela: oznaka koja se predviđa i značajke koje se koriste za treniranje modela. Odabirom značajki modelu se daje na znanje kako će koristiti upravo odabrane značajke kako bi pokušao pronaći određeni uzorak za predviđanje vrijednosti. Osim odabira oznake i značajki,

potrebno je odrediti podatkovni tip stupca koji se dijele na 4 glavne kategorije: broj, tekst, booleov izraz i datum i/ili vrijeme. Nakon odabira značajki i oznaka potrebno je pokrenuti automatizirano treniranje modela. Model se trenira sa nekoliko različitih algoritama u nekoliko iteracija. Nakon što je treniranje izvršeno, u sučelju se ispisuju rezultati. U rezultatima se mogu pronaći informacije poput: koji je algoritam ima najbolju preciznost u predviđaju oznake, kolika je preciznost u postocima – vrijednost koja se promatra je između 0 i 1. Nakon Treniranja modela, automatski su stvorene dvije datoteke pod nazivom Consumption.cs – datoteka pisana u C# programskom jeziku koja otvara mogućnost svojim API-jem (engl. *Application Programming Interface*) odnosno aplikacijsko programsko sučelje treniranog algoritma koje nudi poziv metode za korištenje modela. Druga datoteka opisuje rad treniranja modela pod nazivom Training.cs. u programskom jeziku C#. Nakon treniranja modela, dolazi korak procjene modela. Način na koji se provjerava procjena rada i točnosti modela strojnog učenja je tako da unesemo jedan red podataka koji je strukturiran kao i kod podataka koje smo koristili u tablici za treniranje modela. Nakon unosa jednog reda podataka, program će napraviti u određenom vremenskom periodu procjenu rada modela, te će nakon toga ispisati rezultate procjene rada modela. U procjeni rada modela opisani su rezultati preciznosti rada modela, vjerojatnost točne procjene i naziv oznaka koji je model predvidio.

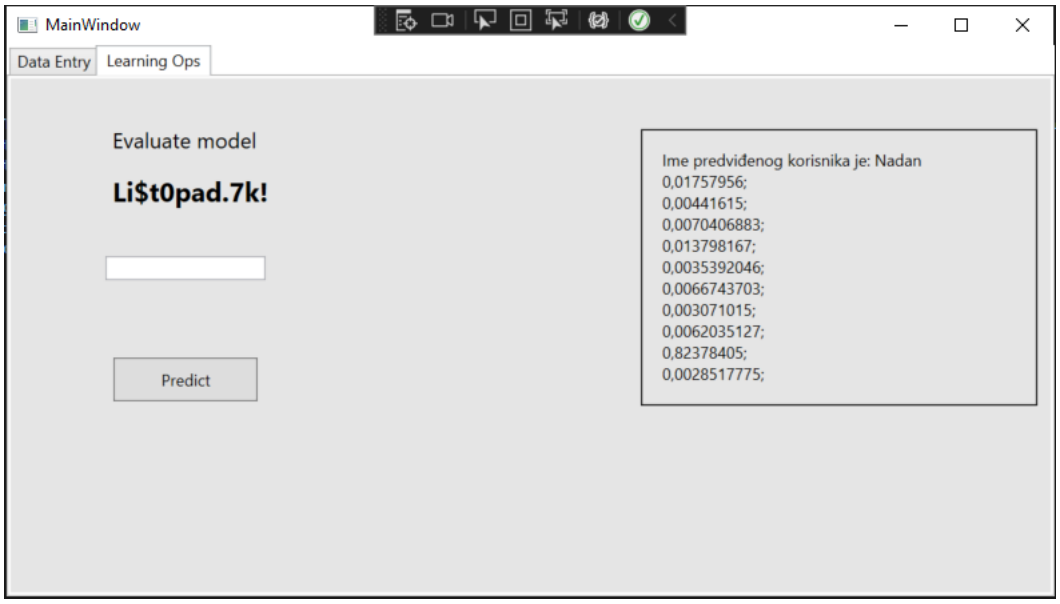
## 3.2. Opis korisničkog sučelja

Korisničko sučelje sastoji se od jednog prozora aplikacije na kojoj se nalaze dvije kartice. Prva kartica sa nazivom „Data Entry“ se koristi kako bi ispitanik unio podatke odnosno utipkao traženu riječ na ekranu. Unos podataka počinje kada ispitanik unese svoje ime ili nadimak koje se razlikuje od ostalih imena koji se koriste kao identifikatori. Nakon toga korisnik u kućicu za unos teksta upisuje traženi tekst. Nakon što se tekst ispravno unese, kućica se isprazni, te oznaka sa nazivom „counter_label“ promijeni svoju vrijednost sa 0 na 1. Zatraženo je od ispitanika da unese istu riječ dvadeset puta, te se sa svakim ispravnim unosom riječi oznaka koja služi kao brojač inkrementalno poveća za jedan. Nakon što korisnik unese dvadeset puta isti set znamenki, uputa dana korisniku je da pritiskom lijeve tipke miša u desnom gornjem putu na znak „X“. ugasi aplikaciju.




Slika 6. Grafičko sučelje generirano XAML programskim kodom

Na slici 4. vidljive su dvije tekstualne kućice, i četiri oznake. Oznaka koja sadrži tekst „Li$t0pad.7k!“ namijenjena je kako bi ispitanik ispred sebe vidio isti set znamenki koju treba upisivati. Oznaka sa trenutnom znamenkom „0“ predstavlja broj ispravni broj točnih upisa riječi koju treba upisati. Klikom na karticu „Learning Ops“ na istom prozoru mijenja se sadržaj grafičkog sučelja koje se koristi kako bi se napravila procjena modela strojnog učenja. Ispitanik nakon treniranja modela ponovo samo jednom unosi isti tekst i pritiskom miša na dugme „Predict“ pokreće programsku logiku sa procjenu preciznosti modela koji koristi **LightGBMMulti** klasifikacijski algoritam. Nakon pritiska dugmeta, ispisuje se lista vjerojatnosti predviđanja imena osobe upisana u tablicu koja je upisala tekst i pritisnula dugme. Lista vjerojatnosti koja se ispisuje je jedna oznaka koja je pri pokretanju programa prazna. Nakon izvršavanja procjene modela, tekst se upisuje sa svakom informacijom u novi red i time se postiže efekt kao da se prikazuje lista, a zapravo je jedan tekst u oznaci.




Slika 7. Grafičko sučelje kartice za procjenu modela

Na slici s desne strane nalazi se element sučelja pod nazivom „Rectangle“ (hrv. *pravokutnik*) koji je u sučelju dodan kako bi uokvirio tekst koji se dobije procjenom modela. Rezultat koji je vidljiv procjenom modela nakon pritiska dugmeta „Predict“ (eng. *predvidi*) prikazuje oznaku koju je model točno predvidio. Brojevi koji se nalaze na listi prikazuju su različite vjerojatnosti o pojedinim osobama u tablici. Cilj modela je da predvidi koji korisnik u tablici je upisivao tekst, te je uspio predvidjeti sa 82,3% vjerojatnosti da je ime predviđenog korisnika Nadan što je točna tvrdnja.

## 3.3. Opis programske logike

Programska logika koja upravlja aplikacijom i programskim sučeljem se nalazi u datoteci MainWindow.xaml.cs što upućuje da je datoteka pisana u C# programskom jeziku i da sa sobom referencira .xaml datoteku MainWindow.xaml. Navedena .cs datoteka je ulazna točka kod pokretanja aplikacije, što znači da kada se pokrene aplikacija iz sučelja uređivača koda, MainWindow prozor će se prvi otvoriti i generirati korisničko sučelje. Kada korisnik počne upisivati traženi tekst u TextBox pod oznakom „Unesite tekst“, pritiskom bilo koje tipke na tipkovnici pokreće se metoda pod nazivom „OnKeyDownHandler“. Metoda u prvom koraku pokreće metodu za provjeru da li je ispitanik koristio tipku CapsLock čije će se vrijednost kasnije koristiti kao značajka za procjenu oznake. Nakon toga pokreće se nova dretva u operacijskom sustavu koja pokreće štopericu. Štoperica staje nakon što korisnik upiše ispravan tekst u TextBox i otpusti sve tipke na tipkovnici. Uzima se mjereno vrijeme i

sprema se u stupac TotalTime unutar .csv datoteke koja se kasnije koristi za treniranje modela, a stupac TotalTime se koristi kao značajka. Kada korisnik otpusti pritisnutu tipku, pokreće se metoda OnKeyUpHandler koja se okida na događaj otpuštanja tipke na tipkovnici ukoliko je kontrola usmjerena na TextBox-u. Metoda provjerava da li je korisnik koristio tipku BackSpace, te istu informaciju naknadno zapisuje u redak tablice i koristi ju kao značajku za treniranje modela. Nakon toga se vrši provjera da li je tekst ispravno upisan. Ukoliko nije, pokreće se nova dretva sa štopericom koja snima vremenski interval između pritisnutih tipki opisano na slici 1.1. kao „Flight time“. Kada korisnik opet pritisne dugme na tipkovnici, završava snimanje vremena štoperice. Snimanje vremena u milisekundama za značajku „Flight time“ se vrši samo kod odabranih tipki na tipkovnici kako bi tablični podaci bili definirani i konzistentni. Tipke za koje se mjeri vremenski period su upravo one tipke koje korisnik mora pritisnuti kako bi upisao točni set znamenki koji se traži u aplikaciji.

```
private static long MeasureGeneralTimeInMiliseconds()
{
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    while (true)
    {
        if (correctPass)
        {
            break;
        }
    }
    stopwatch.Stop();
    long miliseconds = stopwatch.ElapsedMilliseconds;
    stopwatch.Restart();
    return miliseconds;
}
```

Kôd 3. Metoda za mjerenje ukupnog vremena

Prikazana je statička metoda koja se može pozvati kako bi se mjerilo ukupno vrijeme od pritiska tipke na tipkovnici do ispravno upisanog teksta koji je tražen. Metoda se poziva u

zasebnoj dretvi kako ne bi došlo do prekida programa, te metoda započinje mjerenjem vremena. Vrijeme se mjeri sve dok se booleov izraz u varijabli „correctPass“ zapiše kao istinit. Nakon toga prekida se beskonačna petlja naredbom „break“, te se zaustavlja mjerenje vremena štoperice. Štoperica potom mjereno vrijeme zapisuje u milisekunde, nakon toga se resetira kako ne bi ostao mjereni rezultat u memoriji objekta štoperice, te na kraju štoperica vraća rezultat u milisekundama.

```
private void OnKeyDownHandler(object sender, KeyEventArgs e) {
    UsedCaps(e);
    if (workerStarted == 0) {
    generalTimerTask=newTask<long>(()=>
    MeasureGeneralTimeInMiliseconds());
        generalTimerTask.Start();
        workerStarted++;
    }
    if (shiftKeyTimeTask != null) {
        measureShift = false;
        shiftKeyTimeTask.Wait();
        (long shiftKeyTime, string key) = shiftKeyTimeTask.Result;
        restKeyTimeDictionary[$"{key}"] = shiftKeyTime;
    }
    if (restKeyTimeTask != null)
    {
        measure = false;
        restKeyTimeTask.Wait();
        (long restKeyTime, string key) = restKeyTimeTask.Result;
        restKeyTimeDictionary[$"{key}"] = restKeyTime;
    }
}
```

Kôd 4.Metoda koja se okida prema događaju

Navedena metoda se okida prema događaju svaki put kada ispitanik pritisne tipku na tipkovnici ako je kontrola i fokus na TextBox-u koji osluškuje događaj. Osluškivanje događaja referencira se u XAML datoteci na način da se među atributima zapiše metoda za osluškivanje ukoliko grafički element ima te mogućnosti. „KeyDown“ i „KeyUp“ su metode za osluškivanje događaja dostupni na elementu TextBox, te se osluškivanje omogućuje ukoliko se metodi referencira metoda u programskom kodu odnosno u povezanoj C# datoteci. Na četvrtoj i petoj liniji koda opisana je funkcionalnost pozivanja „Task-a“ (hrv. *zadatak*) – jednostruka operacija koja se izvršava asinkrono na način da operacijski sustav dodijeli slobodnu dretvu kako bi se u njoj operacija izvršavala neometano. S obzirom da se operacija izvršava neometano, korisnik je u mogućnosti koristiti druge funkcionalnosti aplikacije. Metoda „Start“ (hrv. *pokreni*) koja se pokreće nad objektom koji predstavlja zadatak pokreće izvršavanje zadatka. Start metodu nije moguće pokrenuti dva puta nad istim zadatkom, s toga svaki puta kada se kod nalazi na poziciji da pokrene isti zadatak, novi zadatak se stvori pomoću ključne riječi „new“ što predstavlja stvaranje novog objekta u memoriji. Kod stvaranja novog zadatka, anonimnom funkcijom pridodajemo metodu koju će zadatak izvršavati. Zadatak ima mogućnost samostalnog završavanja, te nakon završavanja zadatak postavlja zastavicu „IsCompleted“ u istinitu vrijednost. U ovom aplikativnom rješenju se nad zadatkom koristi ključna riječ „Wait“ (hrv. *čekaj*) koja zaustavlja glavnu dretvu nad kojom se program izvršava, potom čeka da se zadatak koji je dodijeljen na drugoj dretvi izvrši do kraja. Nakon završavanja zadatka, glavna dretva može dalje nastaviti sa izvršavanjem programa. Razlog korištenja ključne riječi Wait je kako bi mogli izvršiti i zaustaviti vrijeme sa štopericom i time dobiti ispravno vrijeme razmaka između pritisnutih tipki.

```
            if (InputPassword.Text == "Li$t0pad.7k!") {
                correctPass = true;
                measure = false;
                measureShift = false;
                Task.WaitAll();
                totalTime = generalTimerTask.Result;
                KeystrokeData data = new KeystrokeData(isCaps,
pressedBackspace, totalTime, restKeyTimeDictionary, Username_Input.Text);
                CsvUtils.WriteToCsv(data);
```

```
            ResetData();
            typeCounter++;
            counter_label.Content = typeCounter.ToString();
        }
```

Kôd 5. Stvaranje objekta koji predstavlja redak u tablici

Prva linija koda provjerava da li TextBox sadrži traženi tekst. Ukoliko je tvrdnja točna, svi zadaci koji se trenutno izvršavaju se moraju do kraja završiti upotrebom WaitAll (hrv. *čekaj sve*) naredbe koja kao i Wait naredba zaustavlja izvršavanje programa na glavnoj dretvi. Nakon što smo dobili rezultate mjerenja vremenskih perioda putem zadataka za mjerenje, prosljeđuju se svi parametri u konstruktor klase KeystrokeData koji konstruira objekt koji sadrži svojstva proslijeđena parametrima. Informacije i podatkovni tipovi informacija koje se prosljeđuju objektu:

- Korištenje tipke CapsLock – Da/Ne
- Korištenje tipke BackSpace – Da/ne
- Ukupno mjereno vrijeme – Milisekunde
- Niz mjerenog vremena za svaku traženu tipku – Niz milisekundi od 14 članova
- Ime ispitanika

Nakon konstrukcije objekta, objekt sa svojim informacijama se putem alata za upravljanjem CSV datoteka sprema u .csv datoteku. Potom se svi podaci o mjerenju vremena u radnoj memoriji brišu, oznaka koja služi kao brojač se povećava za 1, te se vrijednost oznake ažurira za + 1. Gašenjem aplikacije svi podaci osim podataka u tablici se resetiraju.

# 4. Razvoj modela strojnog učenja

Modeli strojnog učenja izrađeni su u ML.NET platformi pomoću programskog jezika C#. U aplikativno rješenje dodani su projekti u obliku „konzolne“ aplikacije koje služe za vizualizaciju podataka, treniranje i procjenu modela strojnog učenja. U ovom radu opisan je automatizirani proces strojnog učenja kroz korisničko sučelje ML.NET konfiguracijske datoteke, kao i proces ručnog razvoja modela kroz pisanje programskog koda. Pri odabira kategorije algoritama odabrani su klasifikacijski algoritmi koji rade klasifikaciju na primjerima iz tabličnih podataka. Konzolna aplikacija „ModelTraining.csproj“ za testiranje i evaluaciju sadrži C# datoteku pod nazivom „Program.cs“ koja se koristi kao ulazna točka s kojom se pokreće treniranje modela sa različitim trenerima za klasifikaciju. Unutar navedenog projekta nalazi se direktorij koji sadrži opisne klase s kojima se učitavaju podaci iz tablice. Za jednu vrstu algoritama koristi se jedna vrsta uvozne i izvozne klase koja sadrži različite stupce u svrhu transformiranja podataka za potrebe treniranja modela. Konzolna aplikacija pokreće se putem uređivača koda „Visual Studio“, te se u datoteci Program.cs komentiranjem odabire klasa koja u sebi ima kodom definirane postupke za treniranje modela.

## 4.1. Tablični podaci

Tablični podaci su dobiveni na način da je ispitanik koristio aplikaciju za praćenje dinamike tipkanja opisanu u ovom radu. Ispitanici su skupina od 20 ljudi koji su po 20 puta upisivali istu riječ i s tim upisom upisali retke u tablicu.

| CapsPress | BackSpace | TotalTime | LeftShift | RightShift | LKey | IKey | D4Key | TKey | D0Key | PKey | AKey | DKey | OemPerio | D7Key | KKey | D1Key | Username |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FALSE | FALSE | 6418 | 632 | 0 | 142 | 243 | 606 | 354 | 321 | 166 | 117 | 495 | 424 | 708 | 129 | 0 | Nadan |
| FALSE | FALSE | 6224 | 231 | 0 | 257 | 240 | 212 | 527 | 348 | 213 | 239 | 788 | 965 | 309 | 90 | 0 | Nadan |
| FALSE | FALSE | 6288 | 211 | 0 | 213 | 350 | 208 | 287 | 362 | 63 | 0 | 496 | 920 | 868 | 100 | 1073 | Nadan |
| FALSE | TRUE | 43267 | 17308 | 0 | 1009 | 1396 | 2330 | 590 | 0 | 76 | 973 | 1780 | 2170 | 9444 | 719 | 17442 | draz |
| FALSE | FALSE | 13799 | 1501 | 0 | 637 | 1685 | 1618 | 1491 | 1673 | 32 | 106 | 1222 | 1103 | 1141 | 459 | 0 | draz |
| FALSE | FALSE | 21574 | 6847 | 0 | 1117 | 1651 | 2114 | 0 | 469 | 185 | 1549 | 898 | 1238 | 739 | 1671 | 7125 | draz |
| FALSE | FALSE | 17874 | 7228 | 0 | 498 | 824 | 803 | 1342 | 229 | 375 | 125 | 1365 | 0 | 1753 | 527 | 7359 | draz |
| FALSE | TRUE | 17513 | 627 | 0 | 587 | 1368 | 675 | 1442 | 478 | 360 | 300 | 1692 | 861 | 4333 | 237 | 7359 | draz |
| FALSE | FALSE | 13284 | 1072 | 0 | 909 | 1643 | 1142 | 2710 | 412 | 480 | 156 | 746 | 809 | 509 | 296 | 0 | draz |
| FALSE | FALSE | 14723 | 1573 | 0 | 2523 | 2214 | 1681 | 1574 | 224 | 216 | 298 | 1296 | 602 | 806 | 416 | 0 | draz |
| FALSE | FALSE | 9975 | 721 | 0 | 727 | 531 | 818 | 1202 | 181 | 57 | 147 | 753 | 766 | 517 | 285 | 0 | draz |
| FALSE | FALSE | 11536 | 1233 | 0 | 1477 | 471 | 1284 | 976 | 163 | 96 | 137 | 1707 | 1380 | 525 | 309 | 0 | draz |
| FALSE | TRUE | 13807 | 695 | 0 | 1001 | 637 | 755 | 1617 | 211 | 331 | 466 | 541 | 613 | 676 | 1221 | 0 | draz |

Slika 8. Tabelarni prikaz mjerenja dinamike tipkanja

U tablici se nalazi 18 stupaca od kojih stupac pod nazivom „Username“ predstavlja korisničko ime i koristi se kao oznaka koju model nastoji predvidjeti, dok se preostalih 17

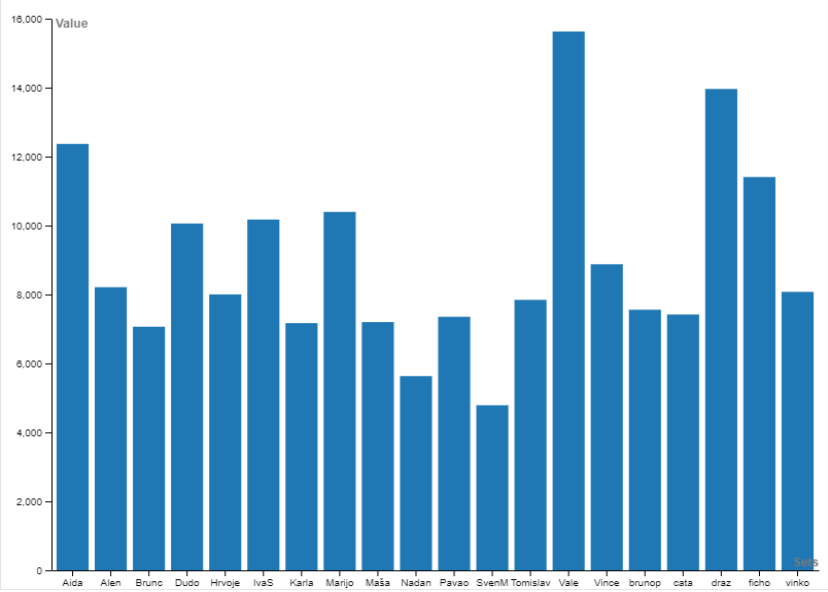
stupaca u tablici koristi kao niz značajki koji se koriste za treniranje i procjenu rada modela. Značajke se mogu grupirati u tri grupe:

1. Kategorijski podaci booleovog izraza
2. Ukupno vrijeme upisa
3. Vremenski raspon između otpuštanja tipke i pritiskom tipke

Pod prvu kategoriju značajki pripadaju stupci „CapsPressed“ i „BackspacePressed“. Pod drugu kategoriju stupac „TotalTime“ predstavlja ukupno vrijeme ispravnog upisa teksta zapisano u milisekundama. Od 4. do 17. stupca su podaci koji pripadaju kategoriji Vremenskog raspona između otpuštanja tipke i pritiskom slijedeće tipke (engl. *Flight Time*). Kategorije podataka su korištene kod treniranja modela te njihova kombinacija daje različite rezultate.

## 4.2. Vizualizacija podataka grafikonima

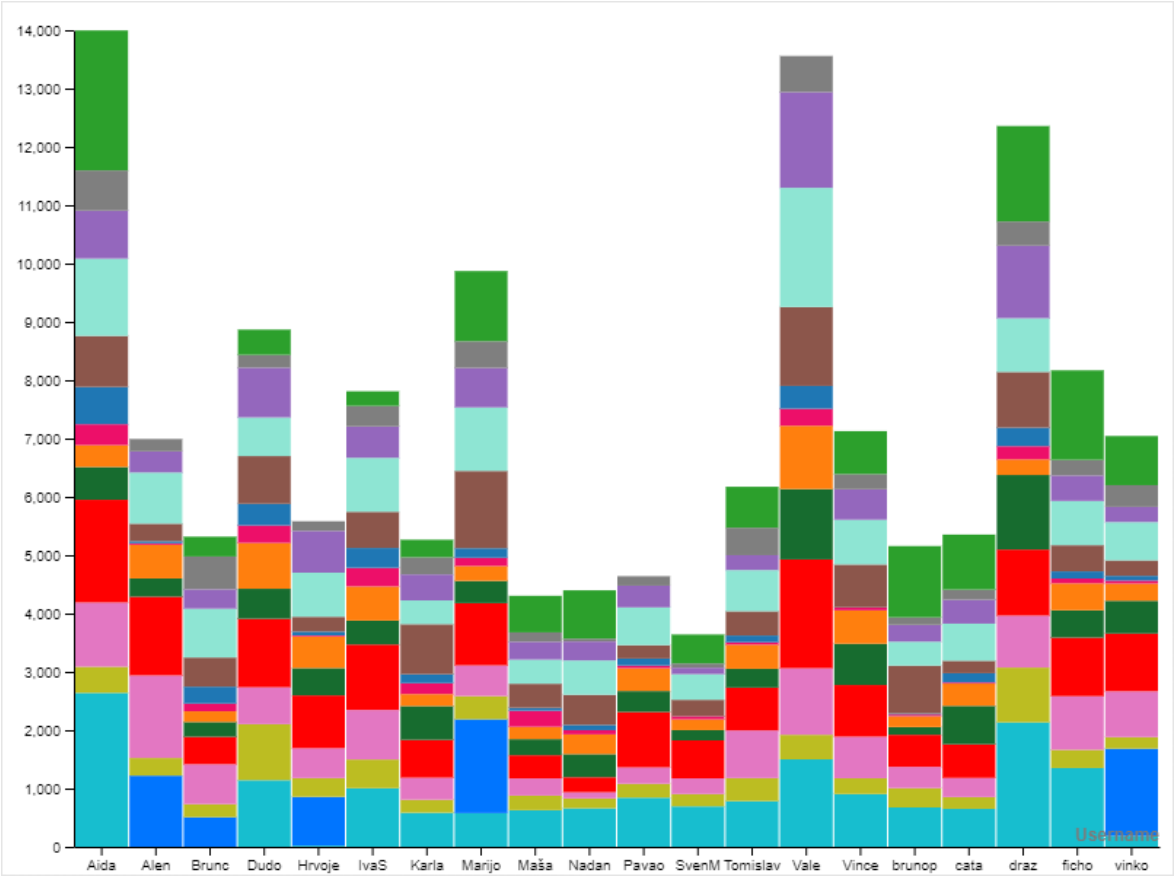
Značajke u tablici moguće je vizualizirati grafikonima i prikazati razliku između podataka u dvodimenzionalnom prostoru.




Slika 9. Prosjek stupca „TotalTime“ po osobi unutar tablice

Slika prikazuje dvodimenzionalni grafikon koji demonstrira prosjek ukupnog vremena unosa ispravnog teksta za svaku osobu koja se nalazi u tablici. Na Y osi grafikona prikazano je vrijeme u milisekundama u kojima je maksimalna vrijednost najbliža 16 sekundi.

Nekolicina osoba koje na ovom grafikonu imaju vrijednosti u milisekundama iznad 10000 su osobe koje ujedno ne koriste računalo u svakodnevnom radu.




Slika 10. Prosjek "Flight Time-a" po osobi

Na slici je prikazan vremenski raspon između otpuštanja tipke i pritiska tipke za svaku osobu. Jedan stupac prikazuje zbroj prosjeka Flight Time-a osobe, a broj stupaca tablice sažetih u jedan stupac u grafikonu je 14. U nekim stupcima na grafikonu nisu vidljive sve vrijednosti iz razloga što kod nekih ispitanika pojedina vrijednost za neke tipke je 0 Razlog tome je da pojedini ispitanici drže prst na tipkovnici i ne otpuštaju ga dok tipkaju drugim prstima. Na ovom grafikonu se vidi da također osobe koje ne koriste računalno u svakodnevnom radu imaju veći razmak između tipkanja.

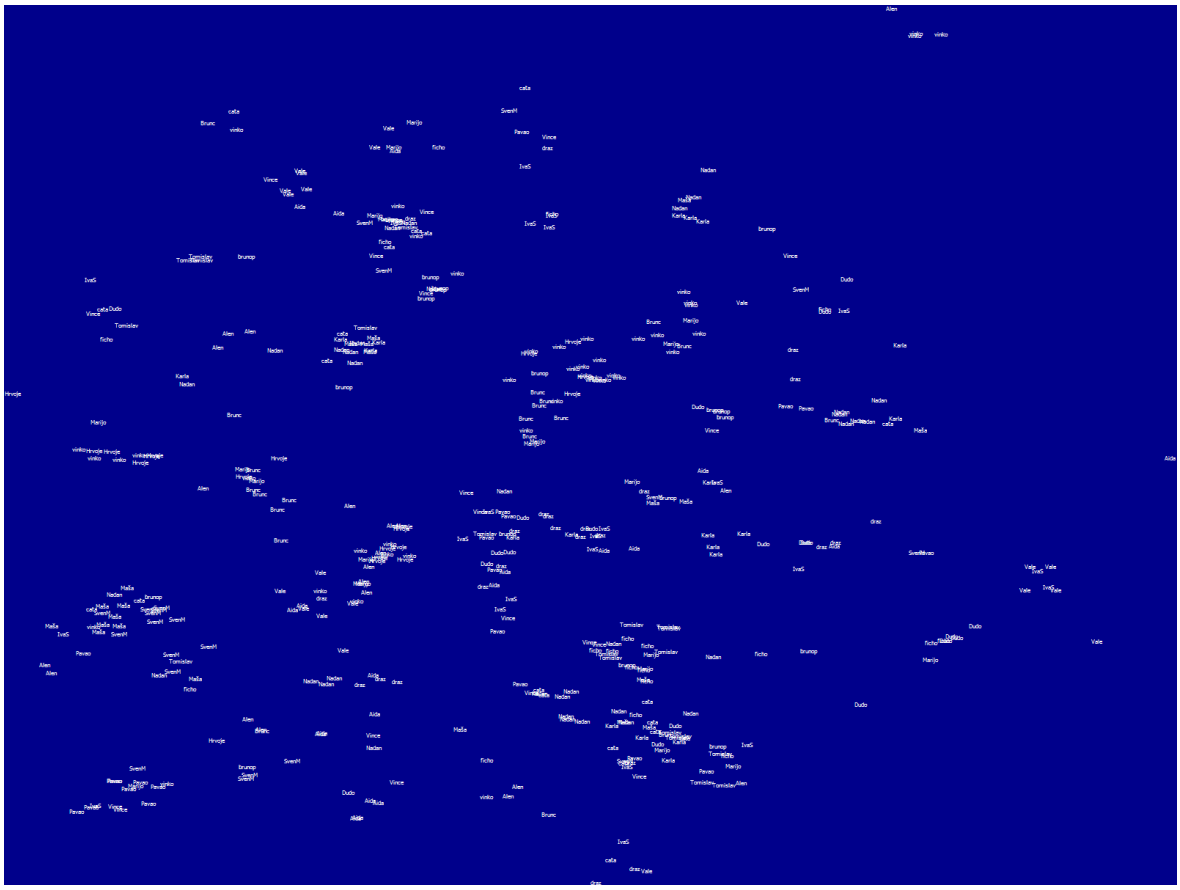
## 4.3. Vizualizacija niza značajki UMAP algoritmom

UMAP (engl. *Uniform Manifold Approximation and Projection*) je tehnika smanjenja dimenzija iz višedimenzionalnih podataka u dvodimenzionalni ili trodimenzionalni prostor. [7] Flight Time značajka se sastoji od niza 14 značajki pretvorene u jednu, što se smatra da

je to značajka od 14 dimenzija koju je moguće svesti na dvije dimenzije tako što ju dimenzijskom redukcijom pretvorimo u jednu točku na X i Y ili X, Y, Z koordinatama na koordinatnom sustavu.

Slika 11. Dvodimenzionalni prikaz Flight Time značajki UMAP algoritmom

Svaka boja na slici prikazuje jednog od 20 ispitanika i svaka točka jednog od ispitanika prikazuje Flight Time značajku u dvodimenzionalnom prostoru na osi X i Y izrađenom prema dimenzijskoj redukciji. Točke koje se nalaze na slici a nisu grupirane u jedan od formiranih klastera predstavljaju izuzetak primjera koji bi mogao loše utjecati na preciznost modela.



Slika 12. Dvodimenzionalan prikaz Flight Time-a prema ispitaniku

Svako ime na slici prikazuje točku koja predstavlja Flight Time. U klasterima u kojima su različita imena pretpostavlja se da iste osobe imaju sličan Flight Time. UMAP algoritam kod klasifikacijskih zadataka se može koristiti kao tehnika za odabir značajki u kojoj formirani klasteri prikazuju sličnosti u klasama. .NET implementacija je izrađena otvorenim kodom i javno dostupna putem „Github“ repozitorija otvorenog koda. [8] Trenutno postoje implementacije u 3 različita programska jezika a to su C#, JavaScript i Python. U svakoj implementaciji nalazi se minimalna razlika kod implementacije a odnosi se na parametre algoritma.

```
var umap = new Umap(distance:
Umap.DistanceFunctions.Euclidean, dimensions: 2a,
numberOfNeighbors: 2);
var nEpochs = umap.InitializeFit(allData.Select(entry =>
entry.MilisVector).ToArray());
for (var i = 0; i < nEpochs; i++)
    umap.Step();
```

Kôd 6. Implementacija UMAP algoritma

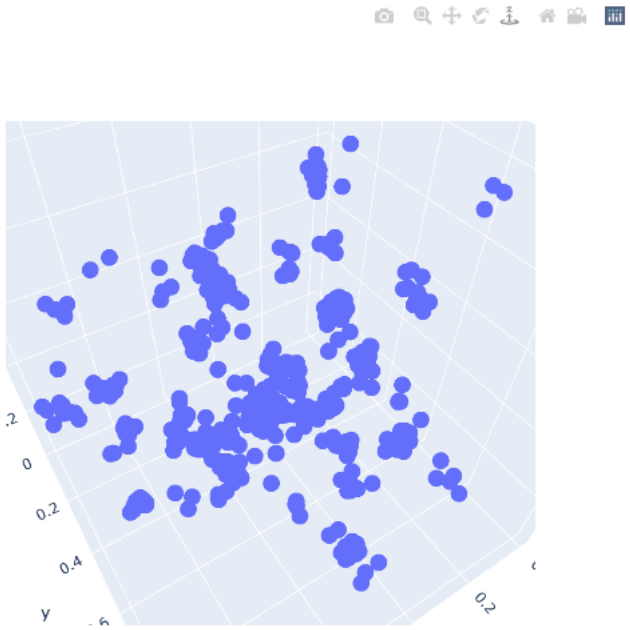
Prve tri linije koda objašnjavaju kako se UMAP inicijalizira kao objekt sa traženim parametrima. Prvi parametar je minimalna udaljenost između točaka na grafikonu koji se podešava između izbora euklidske udaljenosti i kosinusa. Implementacija u jeziku Python nudi normaliziran broj između 0 i 0.99. Drugi parametar nudi smanjenje redukcije za jednu, dvije ili tri dimenzije po izboru. Treći parametar predstavlja broj susjeda, odnosno veličinu nevidljivog radijusa svake točke koja se nalazi na grafikonu. Što je veći broj susjeda to je i radijus veći, a sve točke koje se dodiruju radijusom stvaraju susjedstvo odnosno klaster. Broj susjeda se koristi kako bi se formirala lokalna struktura između točaka na grafikonu. Četvrti parametar zvan broj epoha je opcionalni parametar koji definira broj iteracija s kojom će UMAP algoritam izvršiti dimenzijsku redukciju. Što je više iteracija to je algoritam precizniji kod redukcije. Preporuka je da broj iteracija ili epoha bude 200 sa većim brojem podataka, a 500 sa manjim brojem. Šesta i sedma linija koda predstavlja petlju u kojoj će se odvijati iteracija treninga algoritma koja je definirana brojem podataka u .csv datoteci.

```
var embeddings = umap.GetEmbedding()
    .Select(vector => new { X = vector[0], Y = vector[1] })
    .ToArray();
timer.Stop();
var minX = embeddings.Min(vector => vector.X);
var rangeX = embeddings.Max(vector => vector.X) - minX;
var minY = embeddings.Min(vector => vector.Y);
var rangeY = embeddings.Max(vector => vector.Y) - minY;
var scaledEmbeddings = embeddings
    .Select(vector => new { X = (vector.X - minX) / rangeX, Y
= (vector.Y - minY) / rangeY })
    .ToArray();
foreach (var (vector, username) in
scaledEmbeddings.Zip(allData, (vector, entry) => (vector,
entry.Username)))
    g.FillEllipse(colorsDict[username], vector.X * width,
vector.Y * height, 10, 10);
```

Kôd 7. Pretvaranje dimenzija u dvodimenzionalne vektore

Kod prikazuje definiranje X i Y veličina na koordinatnom sustavu, i definiranje svih točaka na koordinatnom sustavu koje su dobivene dimenzijskom redukcijom. Nakon definiranja vrijednosti, za svaku X i Y vrijednost pridodaje se „Username“ odnosno ispitanik kojem su pripadale vrijednosti Flight Time-a prije nego što su pretvorene u dvodimenzionalne vektore,

te se ime korisnika koristi kako bi pripadne točke prikazali u različitim bojama. Za svakog korisnika je zasebna boja u heksadekadskom zapisu. Potom se u petlji ispisuju i farbaju točke koje se stavljaju na grafikon i potom spremaju u unaprijed definiranu .png (engl. *Portable Network Graphic*) datoteku koja se otvara kao fotografija. Ukoliko se odabere trodimenzionalni prikaz kao parametar u konstruiranju UMAP objekta, onda je potrebno dodati i treću dimenziju kako bi se stvorili trodimenzionalni vektori. Trodimenzionalan prostor je moguće iscrtati „Plotly“ bibliotekom otvorenog koda koja ima funkcionalnost stvaranja grafikona u različitim dimenzijama. Kod izrade grafikona, „Plotly“ biblioteka stvara web stranicu u kojoj se nalazi grafikon sa proslijeđenim parametrima, a na web stranici dodani su alata s kojim je moguće manipulirati grafikonom.




Slika 13. 3D grafikon izrađen Plotly bibliotekom

Na slici u gornjem desnom kutu se nalaze ikone koje se mogu koristiti za rotiranje i zumiranje grafikona koji se nalazi na web stranici. Plave točke predstavljaju Flight Time značajku dobivenu redukcijom UMAP algoritmom.

## 4.4. Klasifikacija u strojnom učenju

Klasifikacija je nadzirana metoda strojnog učenja u kojoj model nastoji predvidjeti točnu oznaku pomoću ulaznih podataka. U klasifikaciji je model treniran koristeći podatke za

trening i testiran koristeći testne podatke koje dosad nije obrađivao kako bi nastojao predvidjeti točnu oznaku. [9] Klasifikacijski zadaci se dijele u tri glavne skupine:

- Binarna klasifikacija
- Višeklasna klasifikacija
- Klasifikacija višestrukim oznakama

U ovom radu se koriste algoritmi strojnog učenja koji pripadaju višeklasnoj i binarnoj klasifikaciji. U binarnoj klasifikaciji cilj zadatka je klasificirati ulazne podatke u dvije kategorije. Binarna klasifikacija se može primijeniti u filtriranju e-pošte na način da model nastoji prepoznati neželjenu poštu. Postupak bi bio uzeti svu poštu, i u setu podataka zapisati koja pošta je željena a koja neželjena. Treniranjem modela sa dvije klase a to su željena i neželjena pošta, model bi mogao sa slijedeći nepoznatim podacima klasificirati u koju klasu spada pošta sa određenom preciznošću. Kod višeklasne klasifikacije zadatak je predvidjeti klasu od danih dvije ili više klasa, za razliku od binarne klasifikacije čije rezultat booleovog logičkog izraza da li je za danu klasu vrijednost istinita ili nije. Primjer višeklasne klasifikacije bi bio zadatak kategorizacije vozila. Unosni podaci bi bio set fotografija gdje se na fotografiji nalazi vozilo poput kamiona, aviona, automobila i čamca. Ukoliko uz svaku fotografiju upišemo koja je to vrsta vozila, s tom oznakom smo definirali klasu. Fotografije i njihove oznake bi se koristile kao unosni podaci odnosno primjeri s kojim bi se trenirao model strojnog učenja. Nakon treninga modela, za procjenu preciznosti modela koriste se zasebni testni podaci koje algoritam još nije obradio. Rezultat procjene preciznosti je da model za svaki dani primjer vrati oznaku za koju misli da pripada fotografiji. Većina klasifikacijskih algoritama koji se koriste u binarnoj klasifikaciji, mogu se koristiti višeklasnoj klasifikaciji. Najpopularniji algoritmi koji se koriste u obje vrste klasifikacije su:

- Random Forests (hrv. *Slučajne šume*) temeljen na stablu odluke
- Naive Bayes
- K-nearest neighbor (hrv. *Najbliži k-ti susjed*)
- Logistička regresija
- Stroj potpornih vektora

Logistička regresija i stroj potpornih vektora prema zadanim postavkama nisu u mogućnosti izvoditi višeklasnu klasifikaciju, te je potrebno primijeniti višeklasni klasifikator koji izvodi klasifikaciju binarnim klasifikatorom. Algoritam pod nazvom „OneVsAll“ (hrv. *Jedan*

*protiv svih*) funkcionira na način da svaku klasu uspoređuje sa ostalim klasama. Ukoliko imamo 20 različitih klasa, svaka klasa će se redom uspoređivati sa ostalima.

## 4.5. Automatizirano strojno učenje

Nakon odabira značajki i oznaka, ML.NET platforma pokreće treniranje modela setom dostupnih algoritama za višeklasnu klasifikaciju čiji je zadatak predvidjeti stupac Username na temelju primjera. Rezultat treninga je prijedlog najpreciznijih algoritama, te nakon toga sastavlja rezultate preciznosti u tabličnom obliku.

| Algoritam | Mikro preciznost | Makro preciznost | Trajanje u sekundama |
|---|---|---|---|
| LightGBMMulti | 0,7781 | 0,7756 | 0,7 |
| SdcaMaximumEntropyMulti | 0,1831 | 0,1290 | 1 |
| FastTreeOva | 0,6353 | 0,6131 | 4 |
| FastForestOva | 0,6678 | 0,6900 | 4,6 |
| SdcaLogisticRegressionOva | 0,1692 | 0,1233 | 2,6 |

Tablica 2. Samohodno odabrani algoritmi za treniranje modela

| Algoritam | Mikro preciznost | Makro preciznost | Trajanje u sekundama | Iteracija |
|---|---|---|---|---|
| LightGBMMulti | 0,7781 | 0,7756 | 0,7 | 2 |
| FastTreeOva | 0,7207 | 0,7267 | 6,7 | 2 |
| FastForestOva | 0,7043 | 0,7182 | 6,9 | 2 |

Tablica 3. Algoritmi za treniranje modela sa najboljim preciznostima u višeklasnoj klasifikaciji

Makro preciznost je prosjek preciznosti na razini klase. Za svaku klasu mjeri se preciznost predviđanja, potom se uzima prosjek svih preciznosti. Mikro preciznost je agregirana vrijednost prosjeka preciznosti svih klasa te je mikro preciznost bolje mjerilo. Za oba pojma

mjerna jedinica je od 0 do 1 s time da što je vrijednost bliža 1 to je preciznost predviđanja bolja. [10] ML.NET platforma samohodno uzima algoritam koji je imao najvišu preciznosti za zadatak višeklasne klasifikacije i postavlja ga kao hipotezu modela koju je moguće procijeniti i konzumirati. [11] Algoritmi koji se nalaze na listi sa sufiksom „ova“ su algoritmi za binarnu klasifikaciju koji primjenom „OneVsaAll“ algoritma izvode višeklasnu klasifikaciju, a njihovi nazivi su „FastTree“ (hrv. *Brzo stablo*) i „FastForest“ (eng. *Brza šuma*) koji pripadaju skupu metodama stabla odlučivanja. Sufiks „Multi“ u nazivu „LightGBMMulti“ označava da se koristi „LightGBM“ algoritam za višeklasnu klasifikaciju koji također spada u skup metoda stabla odlučivanja. Kod koji je generiran u C# jeziku automatiziranim treniranjem modela koristi set podataka za trening kako bi nad njime radio procjenu preciznosti modela što ga čini nepouzdanim jer se procjena vrši nad testnim setom. Dobiveni programski kod se izmjeni svaki puta kada se ponovo pokrene treniranje modela što znači da se programski kod ne može izmijeniti kako bi pridodali druge funkcionalnosti ili pak promijenili parametre na algoritmu.

```
LightGbm(new
LightGbmMulticlassTrainer.Options(){NumberOfLeaves=6,NumberOf
Iterations=6,MinimumExampleCountPerLeaf=20,LearningRate=0.588
586716425705,LabelColumnName=@"Username",FeatureColumnName=@"
Features",ExampleWeightColumnName=null,Booster=new
GradientBooster.Options(){SubsampleFraction=0.999999776672986
,FeatureFraction=0.99999999,L1Regularization=2E-
10,L2Regularization=0.999999776672986},MaximumBinCountPerFeat
ure=206})
```

Kôd 8. Kod generiran automatskim treniranjem modela

Prikazani kod predstavlja stvaranje objekta sa parametrima koji su samostalno dodijeljeni automatiziranim treniranjem modela.

## 4.6. Klasifikacija metodom podizanja gradijenta

Metoda podizanja gradijenta (eng. *Gradient boosting*) izvodi se na način da se prilikom treniranja modela stvara ansambl stabala odluke koji imaju lošu predvidivost ishoda koja je malo bolja od metode slučajnog pogotka rezultata. U svakoj iteraciji treninga izgradi se stablo odluke te se rezultat predviđanja uspoređuje sa stvarnim rezultatom. Nakon usporedbe izgrađuje se novo stablo odluke koje nastoji naučiti prema greškama prijašnjeg stabla. Kod

klasifikacijskih zadataka konačni rezultat treniranog modela izrađuje se prema glasanju prema vrijednostima predviđanja svih stabala odluke. [12] Jedan od algoritama metode podizanja gradijenta je „LightGBM“ koji također spada u obitelj stabala odluka. Algoritam funkcionira na način da svaki puta kada na stablu se stvori list sa svojim rezultatima, ovisno o funkciji gubitka donosi se odluka gdje će na stablu niknuti novi list. Najbitniji parametri kod primjene algoritma su:

- Broj listova koji kontrolira kompleksnost stabla
- Broj iteracija u kojima se izgradi novo stablo
- Minimalni broj podatkovnih točki za rast novog lista
- Stopa učenja

Izmjenom navedenih parametara algoritam će imati različite preciznosti kod predviđanja klase. [13] Treniranjem modela LightGBM algoritmom dobiveni su rezultati preciznosti predviđanja klase. Za trening odabrani su svi stupci u tablici a oznaka koja se predviđa je korisničko ime. Podaci su podijeljeni na set podataka za trening koji sadrži 80% podataka, i 20% podataka za testiranje. Kategorijski podaci u tablici koji predstavljaju booleove vrijednosti pretvorene su u brojčane vrijednosti 0 i 1. U treniranju modela korišteni su najbitniji parametri: Broj listova, broj iteracija, minimalni broj točaka po listu, stopa učenja a ostali parametri su postavljeni prema početnim postavkama. Za mjerenje preciznosti mijenjan je broj listova koji najviše utječe na preciznost.

| Broj listova | Preciznost nad podacima za trening | Preciznost nad testnim podacima |
|---|---|---|
| 4 | 1 | 0,78 |
| 7 | 1 | 0,80 |

Tablica 4. Rezultati dobiveni treniranjem modela LightGBM višeklasnim algoritmom

S obzirom da je razlika u preciznosti između seta za trening i seta za test oko 20%, pretpostavljeno je da postoji problem u visokoj varijanci odnosno „prenaučenosti“ modela što znači da model neće dobro raditi klasifikaciju nad novim podacima. Kod odabira modela potrebno je pronaći model koji ima nisku razliku preciznosti između testnih i podataka za trening. LightGBM ima problem prenaučenosti sa modelima koji su trenirani nad malim setom podataka i taj problem se može riješiti povećanjem seta podataka ili odabirom značajki

za treniranje algoritma. Bolja preciznost modela se može dobiti binarnom klasifikacijom koja se u ovom radu može izvesti primjenom „OneVSAll“ strategijom.

## 4.7. Treniranje modela OneVSAll strategijom

„OneVsAll“ strategija se izvodi na način da se kao parametar uzima algoritam za binarnu klasifikaciju koji izvodi klasifikaciju tako što uspoređuje svaku klasu sa svim ostalim klasama. Kod usporedbe klasa, za svaku klasu se stvara novi klasifikator koji sa određenom preciznošću radi klasifikaciju. Algoritam uzima onaj klasifikator koji ima najbolju preciznost. U ovom primjeru izveden je trening modela sa 6 različitih algoritama za binarnu klasifikaciju primjenom „OneVsAll“ strategije s ciljem da se pronađe model sa najvišom preciznošću predviđanja klase. [14] Kod primjene treniranja modela korištene su sve značajke osim zadnjeg stupca pod nazivom „0Key“ jer je znatno snižavala preciznost kod LightGBM algoritma. u tablici a oznaka koja se predviđa je korisničko ime. Korištena je normalizacija brojčanih vrijednosti kako bi se poboljšala preciznost klasifikacije. Podaci su podijeljeni na set podataka za trening koji sadrži 80% podataka, i 20% podataka za testiranje.

| Algoritam | Mikro preciznost nad trening podacima | Makro preciznost nad trening podacima | Mikro preciznost nad testnim podacima | Makro preciznost nad testnim podacima | Vrijeme potrebno za trening u sekundama |
|---|---|---|---|---|---|
| LightGBM | 1 | 1 | 0,88 | 0,94 | 1,8 |
| FastTree | 1 | 1 | 0,88 | 0,94 | 3,7 |
| FastForest | 0,96 | 0,96 | 0,77 | 0,77 | 3,4 |
| SdcaLogisticRegression | 0,67 | 0,65 | 0,66 | 0,61 | 20,8 |
| LocalDeepSVM | 0,73 | 0,71 | 0,72 | 0,66 | 8,5 |
| LinearSVM | 0,55 | 0,53 | 0,33 | 0,33 | 15,6 |

Tablica 5. Rezultati treniranja modela binarnom klasifikacijom

LightGBM i FastTree algoritmi imaju istu preciznost iz razloga što oba algoritma pripadaju algoritmima podizanja gradienta i trenirani su sa istim osnovnim parametrima u kojima je broj listova sedam, međutim LightGBM ima gotovo dva puta brže izvršavanje za razliku od FastTree algoritma.

Izvedena je k-struka unakrsna provjera podjelom podataka na 5 dijelova u kojoj su u svakom dijelu 80% podaci za trening i 20% podataka za test. Pri provjeri korišten je LightGBM algoritam koji ima najvišu preciznost ali također ima najvišu razliku između preciznosti nad testnim i trening podacima što ukazuje na prenaučenost algoritma. Unakrsnom provjerom dobiveni su slijedeći rezultati:

| Skup učenja | Makro preciznost | Mikro preciznost |
|---|---|---|
| 1 | 0,82 | 0,82 |
| 2 | 0,75 | 0,77 |
| 3 | 0,79 | 0,79 |
| 4 | 0,78 | 0,80 |
| 5 | 0,74 | 0,73 |

Tablica 6. Unakrsna k-struka provjera

Unakrsna provjera pokazuje da model ima u prosjeku 78% preciznosti procjene vrijednosti što je dobar rezultat ali i dalje ostaje problem prenaučenosti jer je znatna razlika između rezultata treniranja i evaluacije. Tijekom pisanja ovog rada korištena je procjena kod zadatka višeklasne klasifikacije modelom stvorenim automatiziranim strojnim učenjem koji ima lošije rezultate predviđanja klase. Od 6 ispitanika model je uspješno procijenio 3 klase od 20. Pretpostavljeno je da kod izvršavanja procjene ispitanik nije koristio istu dinamiku tipkanja kakva je korištena kod stvaranja podataka za trening.

# 5. Testiranje sigurnosti servisa za autentikaciju dinamikom tipkanja

U radu „Latent Typing Biometrics in Online Collaboration Services“, Shane McCully i Vassil Roussev nastoje testirati sigurnost javno dostupnog komercijalnog servisa „TypingDNA“ biometrijskim napadima. TypingDNA navodi da njihov autentikacijski servis ima točnost 99% pri identifikaciji korisnika korištenjem dinamike tipkanja i dinamikom računalnog miša, te da je njihov servis otporan na kopiranja nečije dinamike tipkanja kako bi se identificirali kao željeni korisnik. Servis je testiran na način da je integriran sa web stranicom na kojoj se nalazi jedna kućica za unos teksta u koju je potrebno upisati zaporku. Nakon unosa zaporke objekt koji se šalje na servis sadrži u sebi identifikacijski broj korisnika i skup mjerenja dobiven dinamikom tipkanja. Nakon evaluacije, stranica vraća rezultat 0 ili 1 s time da broj 1 predstavlja uspješnu identifikaciju.

## 5.1. Napad krivotvorenim potpisom

Za testni scenarij odabrano je sveukupno 10 korisnika servisa TypingDNA od kojih su 5 korisnika koji imaju najbolji rezultat u točnosti klasifikacije od kojih nekolicina ima točnost od 100% i 5 koji imaju najlošiju točnost od oko 50% – 60% točnosti pri klasifikaciji . Jedan od zahtjeva testnog scenarija da korisnik ima pravo 5 puta na proces autentikacije prije nego što servis zaključa korisnički račun što znači da sveukupan broj pokušaja autentikacije je 50. Zatraženo je da svaki korisnik napiše isti tekst koji sadrži 200 znakova. Za testni scenarij su poznati parametri koji se šalju na TypingDNA servis, a s time je bilo lako oblikovati tražene podatke. Iz seta podataka od korisnika dobiveni su uzorci dinamike tipkanja koji su u obliku vremenskih intervala između tipkanja digrafa. Za svakog korisnika stvorena je dinamika tipkanja dobivena prosjekom vremenskih intervala, a potom je stvoreno 100 varijacija sa malim odstupanjima od prosjeka za svakog korisnika na temelju njegove dinamike tipkanja. Testiranje se izvodilo na način da se za svakog korisnika šalje jedna od varijacija dinamike tipkanja na TypingDNA servis sa traženim parametrima koje servis prima bez interakcije korisnika. Nakon uspješnih rezultata autentikacije odabrane su varijacije koje su imale uspjeh pri autentikaciji. Krajnji rezultati pokazuju da od 5 pokušaja autenticiranja krivotvorenim potpisom uspješno su autenticirani svi korisnici sa maksimalnom izvedbom

do 3 pokušaja. Dinamika tipkanja jednog od korisnika je replicirana na način da je imala 100% točnost pri klasifikaciji. Smatra se da značajke dinamike tipkanja nemaju visoku varijancu kao kod ostalih korisnika i da je iz tog razloga bila visoka točnost. [15]

# Zaključak

Dinamika tipkanja je dio ljudskog identiteta i sadrži informaciju o tome kako se osoba ponaša za računalom i nije moguće uvijek replicirati dinamiku tipkanja iste osobe što predstavlja izazov u analizi bihevioralnom biometrijom jer računalo ne može sa uvijek ustanoviti identitet osobe.

Iz ovog rada se može zaključiti da računalo putem strojnog učenja može analizirati i predvidjeti dinamiku tipkanja korisnika računala putem tipkovnice. Bilježenim podacima unosa korisnika i klasifikacijskim algoritmima se može stvoriti model za predviđanje korisničkog računa sa visokom preciznošću. Algoritmi stabala odluke pokazali su visoku preciznost pri klasifikaciji korisnika. Zadatak binarne klasifikacije je idealan autentikaciju korisnika iz razloga što kod autentikacije korisnika u informacijskom sustavu povratna informacija kod prijave korisnika treba vratiti informaciju da li je korisnik identificiran ili nije.

Trenutna praksa kod autentikacije u poslovnim organizacijama se izvodi na način da kod prijave u osobno računalo, korisnik unosi samo lozinku ili pin što dovodi do ugroze iz razloga što napadač ukoliko zna lozinku može se prijaviti kao korisnik u računalo. Poslovna praksa u autentikaciji nastoji zaobići ovaj problem na način da svakih 3 ili 6 mjeseci zaposlenik je primoran mijenjati postojeću lozinku za prijavu u računalo. Takvim strogim poslovnim politikama zaposlenici nastoje koristiti lozinke koje su već korištene ili modificirati postojeću lozinku za minimalnom izmjenom znakova što i dalje predstavlja sigurnosni problem. Bihevioralna biometrija strojnim učenjem bi se mogla primijeniti na način da zaposlenik koristi samo jednu lozinku dok je zaposlen u poduzeću, a da njegova dinamika tipkanja koja je naučena se koristi kao drugi faktor pri prijavi u informacijski sustav. Takvom vrstom dvofaktorske autentikacije bi napadaču bila otežana provala u informacijski sustav iz razloga što je dinamiku tipkanja teško replicirati.

Osim poslovnih sustava, ovakva vrsta autentikacije se može primijeniti i za privatne korisnike u web i mobilnim aplikacijama kako bi se smanjio broj neovlaštenih prijava i krađa podataka. U digitalnom bankarstvu već postoje programska rješenja koja izvode analizu ponašanja korisnika kako bi detektirali neobično ponašanje i spriječili ilegalne aktivnosti.

Student vlastoručno potpisuje Završni rad na prvoj stranici ispred Predgovora s datumom i oznakom mjesta završetka rada te naznakom:

*„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.*

*U Zagrebu, 28.3.2023.*

# Popis kratica

| | | |
|---|---|---|
| UMAP | *Uniform Manifold Approximation & Projection* | uniformna mnogostruka aproksimacija i projekcija |
| WPF | *Windows Presentation Foundation* | |
| XAML | *Extensible Application Markup Language* | proširiv aplikacijski označni jezik |
| XML | *Extensible Markup Language* | proširiv označni jezik |

# Popis slika

# Popis tablica

# Popis kôdova

# Literatura

Svaki autor piše popis literature na kraju rada. Popis literature se piše stilom literatura.


[1] BIOCATCH, What Is Behavioral Biometrics? (biocatch.com), 4. Prosinca 2022.

[2] RESEARCHGATE, THE PHYSIOLOGY OF KEYSTROKE DYNAMICS (Jeffery Jenkins, Quang Nguyen, Joseph Reynolds, 2011), 5. Prosinca 2022.

[3] RESEARCHGATE, A SURVEY ON BEHAVIORAL BIOMETRIC TECHNIQUES: MOUSE VS KEYBOARD DYNAMICS (Nilam Khairnar, 2013), 5. Prosinca 2022.

[4] RESEARCHGATE, STUDY ON MOST POPULAR BEHAVIORAL BIOMETRICS, ADVANTAGES, DISADVANTAGES AND RECENT APPLICATIONS: A REVIEW (Israa Alsaadi, 2021), 6. Prosinca 2022.

[5] MICROSOFT https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-6.0, 20. Prosinca 2022.

[6] MICROSOFT https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet#automl 22. Prosinca 2022.

[7] UMAP-LEARN, HTTPS://UMAP-LEARN.READTHEDOCS.IO/EN/LATEST/, 28. Prosinca 2022.

[8] GITHUB, HTTPS://GITHUB.COM/CURIOSITY-AI/UMAP-SHARP, 29. Prosinca 2022.

[9] DATACAMP, https://www.datacamp.com/blog/classification-machine-learning, 2. Siječnja 2023.

[10] MICROSOFT https://learn.microsoft.com/en-us/dotnet/machine-learning/resources/metrics, 29. Prosinca 2022.

[11] FER, NADZIRANO UČENJE (Jan Šnajder, Bojana Dalbelo Bašić, 2012), 3. Siječnja 2023.

[12] IMPLEMENTACIJA KLASIFIKACIJSKOG ALGORITMA PODIZANJA GRADIJENTA STABLIMA ODLUKE S PRIMJENOM NA PREDIKCIJU ISHODA NOGOMETNIH UTAKMICA (Dominik Hrastić, 2019), 5. Siječnja 2023.

[13] NEPTUNE HTTPS://NEPTUNE.AI/BLOG/GRADIENT-BOOSTED-DECISION-TREES-GUIDE, 7. Siječnja 2023.

[14] MICROSOFT https://learn.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.oneversusalltrainer?view=ml-dotnet, 8. Siječnja 2023.

[15] LATENT TYPING BIOMETRICS IN ONLINE COLLABORATION SERVICES (Shane McCulley, Vassil Roussev, 2018), 1. Veljače 2023.