Raspoznavanje slova znakovnog jezika korištenjem postupaka strojnog učenja

Takač, David

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

RASPOZNAVANJE SLOVA ZNAKOVNOG JEZIKA KORIŠTENJEM POSTUPAKA STROJNOG UČENJA

Završni rad

David Takač

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 03.09.2019.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	David Takač
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3995, 20.09.2018.
OIB studenta:	92780543146
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	Dr.sc. Bruno Zorić
Sumentor iz tvrtke:	
Naslov završnog rada:	Raspoznavanje slova znakovnog jezika korištenjem postupaka strojnog učenja
Znanstvena grana rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	03.09.2019.
Datum potvrde ocjene Odbora:	11.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 15.09.2019.

Ime i prezime studenta:	David Takač
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3995, 20.09.2018.
Ephorus podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Raspoznavanje slova znakovnog jezika korištenjem postupaka strojnog učenja**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora Dr.sc. Bruno Zorić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Potreba za jednostavnim sustavom učenja znakovnog jezika postoji otkad i sam znakovni jezik. Prije računala, jedini način na koji se moglo učiti znakovni jezik bilo je pomoću instruktora ili knjiga. Knjige ne daju povratnu informaciju o točnosti pokazanoga znaka, dok je unajmljivanje instruktora skupo i nezgodno. Kada instruktor nije tu, nije moguće vježbati znakove i dobiti povratnu informaciju o njihovoj točnosti. Cilj ovog rada je predstaviti metode klasičnog strojnog učenja koje će, zajedno s metodama obrade slike, omogućiti ljudima vježbanje znakovnog jezika pred kamerom. Korisnik može koristiti web-kameru svog računala da snimi sliku znaka kojeg pokazuje i računalo će pomoću modela strojnog učenja dati povratnu informaciju o točnosti pokazanog znaka. Ovo eliminira potrebu prisutnosti instruktora za povratnu informaciju i daje jednostavnu mogućnost samostalnog vježbanja.

U drugom poglavlju predstavlja se detaljniji opis problema, odabrana baza slika i već postojeća rješenja u ovom polju. Treće poglavlje opisuje korištene metode predobrade slika, strojnog učenja i vrednovanja. Četvrto poglavlje sadrži implementaciju algoritama o kojima je bilo riječi u trećem poglavlju pomoću programskog jezika Python kao i analizu rezultata. Peto poglavlje donosi zaključak i sažima najbolje rezultate.

1.1. Zadatak rada

U teorijskom dijelu rada potrebno je opisati postupke i korake strojnog učenja za klasifikaciju slova znakovnog jezika na temelju fotografija. U praktičnom dijelu rada usporediti uspješnost različitih postupaka na podatkovnom skupu slika dostupnih iz literature.

2. RASPOZNAVANJE ELEMENATA ZNAKOVNOG JEZIKA

U ovom poglavlju predstavlja se opis problema i jednostavan opis buduće implementacije. Bit će riječi o znakovnom jeziku i njegovim osobitostima koje su relevantne za ovaj rad. Predstaviti će se baza slika koja se koristi u eksperimentalnom dijelu rada i na kraju će se dati osvrt na postojeća rješenja u ovom području.

2.1. Opis problema

Prepoznavanje slova znakovnog jezika iz slike ljudske ruke važan je problem za gluhe ljude ili općenito one koji ne koriste verbalnu komunikaciju. Postojeće metode učenja znakovnog jezika uključuju unajmljivanje instruktora ili samostalno učenje kroz knjige ili videozapise na internetu. Od ovih metoda jedino će instruktor dati povratnu informaciju o točnosti pokazanog znaka, dok metode samostalnog učenja imaju za nedostatak upravo manjak povratne informacije.

Rješenje samostalnog učenja od kuće s povratnom informacijom predstavlja se u ovom radu. Implementirati će se raznim postupcima strojnog učenja i omogućiti korisniku samostalno učenje slova znakovnog jezika. Glavni problem u implementaciji ovog rješenja je izolacija bitnih informacija iz slike ruke. Naime, priroda slike je da u njoj ima puno smetnji koje štete vladanju korištenih algoritama i moći prepoznavanja. To su pojave poput razlike u osvjetljenju, sadržaji pozadine, boja ruke i slično. Ove pojave se razlikuju od korisnika do korisnika i kako bi krajnji model bio robustan potrebno je umanjiti njihov utjecaj algoritmima izdvajanja značajki. Zatim slijedi implementacija algoritama strojnog učenja i podešavanje njihovih parametara kako bi se vladali što bolje s korištenim podatcima. Za vrednovanje uspjeha algoritama koristit će se razne statističke metode.

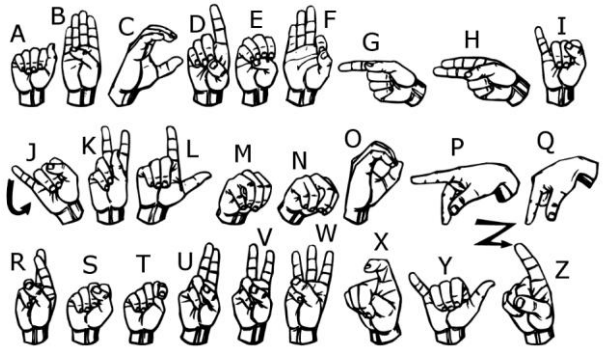
2.2. Znakovni jezik

Znakovni jezik je potpun i složen jezik koji koristi znakove napravljene pomicanjem ruke u kombinaciji s izrazom lica i govorom tijela. To je glavni jezik mnogih gluhih ljudi i onih koji se ne koriste verbalnom komunikacijom.

Sastoji se od posebnih gesti koje opisuju konkretne riječi i od znakova koji predstavljaju slova kojima se mogu „slovkati“ one riječi za koje ne postoji gesta.

2.2.1. Slova

Slova znakovnog jezika određena su položajem ruke i prstiju i položaj ruke svakog od znakova prikazan je slikom 2.1.



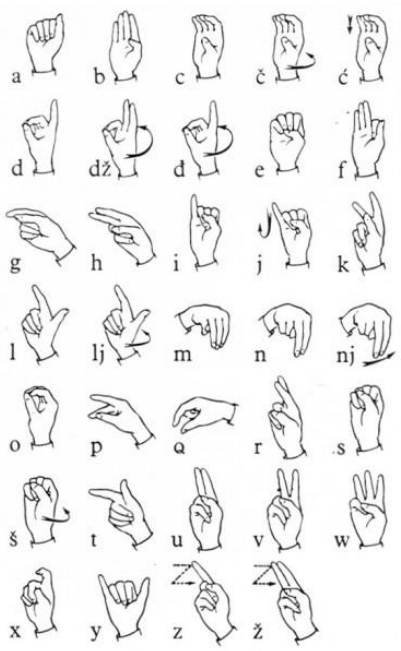
Sl. 2.1., Američka znakovna abeceda [1]

Ovaj rad usredotočen je na prepoznavanje slova osnovnog američkog znakovnog jezika jer za njega postoje najveće baze podataka označenih slika. Principi rada neovisni su o tome koja se baza slika koristi, pa će se postupci opisani ovdje moći primijeniti na prepoznavanje znakova bilo kojih znakovnih jezika. Jedina potrebna promjena bit će zamjena korištene baze slika za treniranje algoritma.

2.2.2. Različiti jezici

Znakovni jezik, kao i govorni, razlikuje se po lokalitetu. U SAD-u i Kanadi koristi se američki znakovni jezik (ASL) i postoje razne inačice znakovnog jezika za razna govorna područja.

U Hrvatskoj se koristi hrvatska znakovna abeceda koja je verzija američke znakovne abecede, ali s dodanim slovima koja se odnose na hrvatsko govorno područje (č, ć, đ, š, ž i dž). Ova dodatna slova zahtijevaju pokret ruke kako bi se opisala.



Sl. 2.2, Hrvatska znakovna abeceda [2]

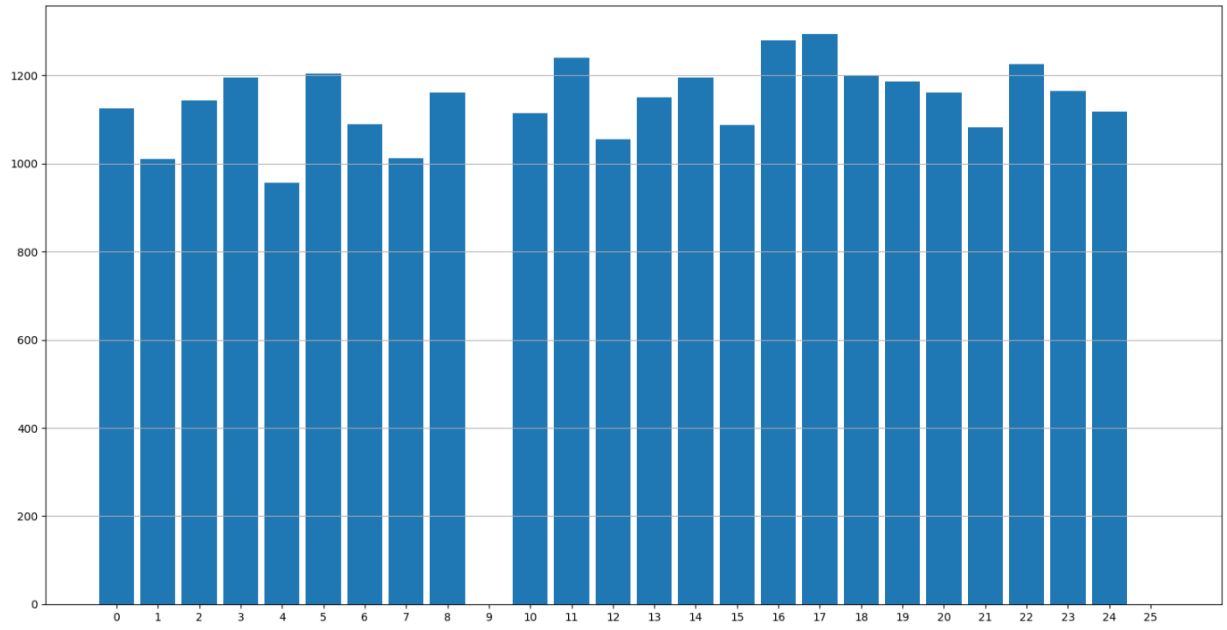
2.3. Načini snimanja i vrste slika

Načini snimanja ljudske ruke za svrhe klasifikacije uglavnom se dijele na dvije metode. Prva metoda koristi elektroničke rukavice koje imaju senzore za detekciju i snimanje gesti ruke [3]. Neke od takvih rukavica moraju biti spojene na računalo pomoću žica da bi radile, dok su neke spojene bežično. U svakom slučaju, takva oprema je skupa i nezgodna za korištenje. Također, instalacija takvih uređaja je kompliciranа i zahtijeva dosta vremena. Prednost ove metode je u mogućnosti opisa znakova koji zahtijevaju pokret ruke.

Druga metoda koristi računalni vid da bi prepoznala znakove. Kamerom se snimaju slike, odnosno videozapisi, koji zatim prolaze predobradu i izdvajanje značajki kako bi se mogli koristiti za treniranje algoritama za klasifikaciju. Na isti se način obrađuju novi primjeri koji se trebaju klasificirati. Ova metoda je intuitivnija, jeftinija i jednostavnija od elektroničke rukavice, ali mana joj se nalazi u tome što se pokreti ruke koji su potrebni za neke od znakova ne mogu jednostavno predstaviti. Drugi problem javlja se u obradi slika kako bi se mogle koristiti za klasifikaciju. Slike se mogu razlikovati u skali, rotaciji, osvjetljenju, pozadini, količini buke i sl., a za robusnost algoritma bitno je uzeti te faktore u obzir i eliminirati ih u što većoj mjeri.

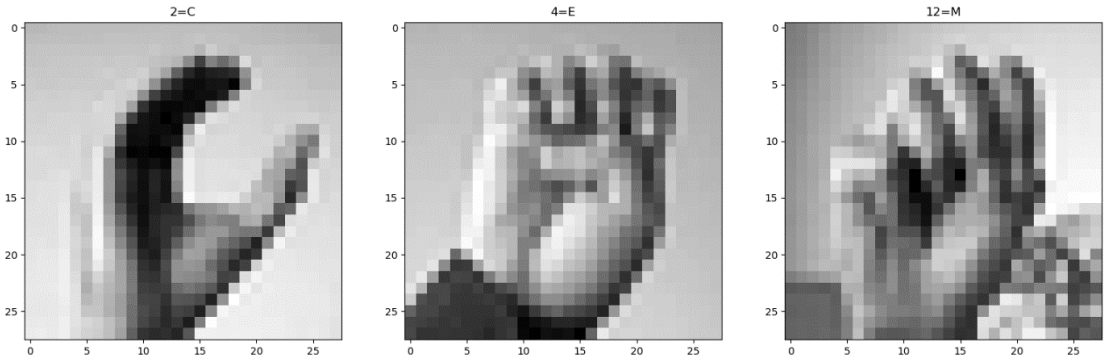
2.4. Baza slika

Baza slika koja se koristi za treniranje i testiranje metoda strojnog učenja u ovom radu je "*Sign language MNIST*" [4] baza. Ona se sastoji od 27455 primjera za treniranje i 7172 primjera za testiranje. Slike predstavljaju slova američke znakovne abecede bez slova J i Z jer ta slova zahtijevaju pokret ruke. Metode ovoga rada se općenito neće moći primijeniti na ona slova koja uključuju pokret ruke. Vrijednosti oznaka u bazi su brojevi od 0 do 25 i svaki broj odgovara slovu američke abecede od A do Z. Za znakove J=9 i Z=25 nema nijedan primjer. Dakle, ovo je višeklasni problem klasifikacije s 24 klase. Slika 2.3. pokazuje da su primjeri za treniranje jednoliko raspoređeni. Također vidimo da nema niti jedan primjer za oznake 9(J) i 25(Z), što je očekivano.



Sl. 2.3., Distribucija broja trening primjera po oznakama

Osim oznaka, svaki primjer se sastoji od vektora značajki sa 784 elementa, koji odgovaraju „razmotanoj" slici dimenzija 28x28 piksela. Pikseli mogu poprimiti vrijednosti od 0 do 255 i predstavljaju intenzitet slike u sivim tonovima. Tablica 2.1. prikazuje način na koji su primjeri spremljeni u bazu.



Sl. 2.4., Neki primjeri iz baze slika za treniranje

Tab. 2.1., Prvih sedam primjera iz baze slika za treniranje.

oznaka klase	piksel1	piksel2	...	piksel784
3	107	118	...	202
6	155	157	...	149
2	187	188	...	195
2	211	211	...	163
13	164	167	...	179
16	161	168	...	255
8	134	134	...	179

2.5. Primjene raspoznavanja znakovnog jezika strojnim učenjem

Jedna moguća primjena postupaka ovog rada je u samostalnom učenju abecede znakovnog jezika. Model strojnog učenja s fino podešenim parametrima može se implementirati u računalnoj ili mobilnoj aplikaciji. Takva aplikacija trebala bi imati mogućnost uslikati, spremiti i primijeniti istu vrstu obrade uslikane slike kao što je primijenjena na slikama koje su korištene za treniranje modela. Zatim se takva obrađena slika treba predati modelu strojnog učenja. Ovisno o izlazu iz modela i očekivanom znaku, korisniku se može dati povratna informacija o točnosti pokazanog znaka. Ako se oznaka znaka kojeg korisnik treba pokazati slaže s klasifikacijom slike pokazanog znaka, korisniku se potvrdi da je znak točan. U suprotnom, zamoli ga se da proba ponovo dok se oznake ne slažu.

Druga primjena je u pisanju pomoću znakovnog jezika. U tom slučaju je također potrebna računalna ili mobilna aplikacija s implementiranim modelom strojnog učenja i predobradom slike. Korisnik bi

mogao pokazivati znakove pred računalom i slike tih znakova bi se dale na ulaz modelu strojnog učenja. Takav model bi zatim prepoznao pokazani znak i aplikacija bi ispisala prikladno slovo.

2.6. Postojeća rješenja

2011. godine, Marek Hruz et al. koristili su histograme lokalnih binarnih uzoraka za izdvajanje značajki iz slike ruke osobe koja pokazuje znak [5]. Usporedili su performanse tih značajki sa značajkama geometrijskog momenta koje opisuju putanju i oblik ruke. Pokazalo se da značajke lokalnih binarnih uzoraka imaju bolje performanse od geometrijskih momenata, a kombinacijom lokalnih binarnih uzoraka i opisnika značajki lica (središte očiju, vrh nosa i središte gornje usne) postignuta je točnost od 57.54% na testovima neovisnim o osobi koja pokazuje znakove i 99.75% u testovima ovisnim o osobi koja pokazuje znakove. U ovom radu detaljnije se istražuju lokalni binarni uzorci kao metoda izdvajanja značajki.

Iste godine, Ruslan Kurdyumov et al. koristili su stroj s potpornim vektorima (SVM) s linearnim i Gaussovim kernelom, kao i algoritam *k-najbližih susjeda* (KNN) za klasifikaciju slova američke znakovne abecede [6]. Bazu slika stvorili su pomoću web-kamere na način da su po znaku imali dvije slike. Prva slika bila je slika pozadine, a u drugoj je slici dodana i ruka na istu pozadinu. Od druge slike su zatim oduzeli pozadinu, pretvorili svaku boju koja nije crna u bijelu i maknuli male smetnje kao bijele ili crne fleke i točkice. Zatim su obrezali su sliku oko ruke i obrisali zapešće. Takva je slika zatim pretvorena u dimenziju 20x20 piksela, što daje vektor značajki duljine 400 elemenata. Baza slika s takvim primjerima korištena je za treniranje spomenutih klasifikatora. Postignute točnosti na skupu od 25 znakova sa SVM-om su veće od 92%, dok je KNN imao točnost oko 84%. Osim toga, probali su napraviti 13 posebnih značajki koje su imale u cilju istaknuti rubove prstiju ruke i tako klasificirati znakove. SVM je tada imao točnost od preko 82%, dok je KNN imao točnost 78%.

2015. godine, Lucas Rioux-Maldague et al. implementiraju klasifikaciju slova američke znakovne abecede koristeći neuronske mreže dubinskog učenja [7]. Usredotočuju se na 24 statična znaka abecede i koriste jednu kameru za snimanje normalne slike, a jednu za snimanje dubinske slike. Obrađuju obje slike i kombiniraju njihove značajke u jedan vektor značajki. U testiranju na bazi slika koju klasifikator još nije vidio postižu 79% preciznosti i 77% odziva. Pojašnjavaju zašto klasifikator ima problema s prepoznavanjem sličnih znakova kao što su A, E, M, N, S i T. Naime, dubinska slika, kao i slika intenziteta, ne pomaže previše s takvim znakovima jer je ruka za svaki od tih znakova u

skoro istom položaju, što će dati vrlo sličnu dubinsku sliku, a i 2D reprezentacije u slici intenziteta su vrlo slične. Sličan problem javlja se i u ovom radu i bit će pokazano kako sličnosti tih znakova utječu na klasifikaciju.

3. POSTUPCI STROJNOG UČENJA ZA RASPOZNAVANJE SLOVA ZNAKOVNOG JEZIKA

Raspoznavanje slova znakovnog jezika problem je kojeg mogu riješiti algoritmi klasifikacije. Klasifikacija je postupak kojim se predviđa koja je klasa predstavljena ulaznim podacima. Uspješnost klasifikacije novih primjera ovisi o podacima koji su korišteni za treniranje klasifikatora i njegovim parametrima.

U ovom su poglavlju redom objašnjeni postupci obrade ulaznih podataka i strojnog učenja koji su potrebni za ostvarenje klasifikatora koji je prikladan korištenim podacima. U ovom slučaju ulazni podaci su slike znakova znakovnog jezika, no slični postupci se primjenjuju i u ostalim klasifikacijskim problemima gdje se koriste slike kao podaci za treniranje.

3.1. Predobrada slika

Predobrada slika je operacija koja na ulaz prima sliku u obliku dvodimenzionalne matrice intenziteta sivih tonova i na izlazu daje promijenjenu sliku. Svrha predobrade je izolacija bitnih informacija i smanjenje dimenzionalnosti podataka.

Prema [4], predobrada koja je primjenjena na korištenoj bazi slika je:

1. Na ulaz je dovedena neobrezana slika u boji koja prikazuje ruku u položaju jednog od znakova znakovne abecede, ali i ostale informacije koje nisu bitne kao čovjek koji pokazuje znak, podatci o pozadini itd.
2. Takva sirova slika je prvo obrezana na način da prikazuje samo ruku.
3. Zatim joj je veličina promijenjena u dimenzije 28x28 piksela.
4. Na kraju, slika je promijenjena iz prostora boje u prostor sivih tonova.

Slika 3.1. prikazuje konačan rezultat predobrade. Ovo je oblik slika u bazi i kao takve se koriste pri tvorbi modela strojnog učenja ovog rada.

Sl. 3.1., Konačni rezultat predobrade

Prije treniranja i testiranja klasifikatora potrebno je obraditi ulazne slike na isti način kao što su obrađene u korištenoj bazi slika. Kada se to ne bi napravilo, klasifikator ne bi mogao prepoznati nove primjere pravilno jer je treniran na drugačijoj vrsti podataka od one koja mu se prikazuje. Budući da je baza slika napravljena tako da već ima odvojene primjere koji su predviđeni za testiranje na koje je primijenjen isti postupak predobrade, u eksperimentalnom dijelu ovog rada predobrada se neće vršiti direktno. U konkretnoj implementaciji postupaka ovog rada u obliku računalne ili mobilne aplikacije potrebno je provesti ovu predobradu za svaki novi primjer kojega se želi klasificirati.

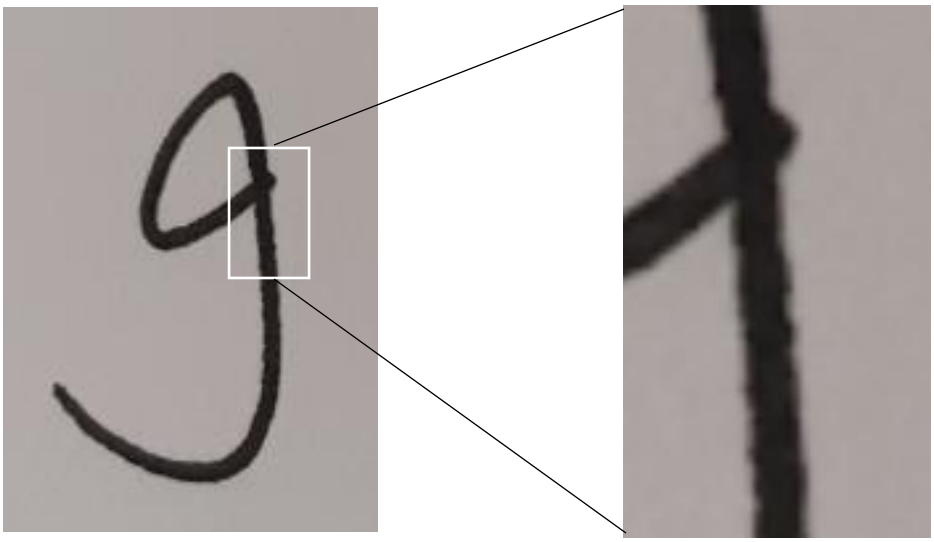
3.2. Izdvajanje značajki

U strojnom učenju, značajka je mjerljivo svojstvo ili karakteristika promatranog fenomena [8]. Značajke se prikazuju u obliku vektora značajki, što je zapravo samo skup vrijednosti svih značajki koje su organizirane u oblik vektora. Značajke se koriste kao opisnici podataka koje se promatra i za koje se želi trenirati klasifikator. Zato je vrlo bitno odabrati takav skup značajki koji će najbolje predstaviti ulazne podatke. Dobre značajke su informativne, što znači da pružaju dovoljno informacija kako bi se fenomeni koji se promatraju mogli razlikovati. Informativne značajke su one koje su međusobno neovisne, odnosno svaka značajka daje posebnu i korisnu informaciju koja poboljšava moć razlikovanja klasifikatora [9]. Značajke koje opisuju slike bi iz tog razloga trebale biti što otpornije na buku i nerelevantne informacije. Neki od primjera takve buke u slikama uključuju razlike

u osvjetljenju ili rotaciji, skali i poziciji promatranog fenomena na slici. Razlog iz kojega se mora raditi ovo kako bi se dobile dobre značajke je što računalo ne vidi slike isto kao i ljudi. Ljudski um ovo sve već radi automatski, ali klasifikator je samo matematički model koji je jedino dobar koliko i podaci na kojima je učio. Kao primjer može se zamisliti da se slike korištene za treniranje klasifikatora ne procesiraju tako da su neovisne o razini osvjetljenja i da su sve slike savršeno osvijetljene. Klasifikator treniran na takvim podacima neće znati prepoznati klasu primjera koji ima lošije osvjetljenje jer je učen samo na savršeno osvjetljenje. Isti princip primjenjiv je i na ostale poželjne osobine dobrih značajki.

Međutim, pronaći ovakve značajke u bazama podataka izazovan je zadatak. Baze podataka koje se koriste za probleme strojnog učenja često imaju intrinzičnu pravilnost i razlikovne osobine koje su lako primjetljive ljudima. Primjerice, nije problem iz slike rukom pisanog broja (sl. 3.2.) očitati koji je to broj. Uz smetnje i distorzije na slici čovjek svejedno može shvatiti koja informacija se želi prenijeti jer ljudski um automatski zanemaruje smetnje. Drugim riječima, ljudski mozak je naučio izdvojiti značajke i učiti se jedino na bitnim informacijama.

Na slici 3.2. lako je pročitati da je prikazan broj 9. Tek uvećanjem se otkriva da postoji dosta smetnji koje nisu bitne za prepoznavanje tog broja. Kada bi se klasifikator trenirao na takvim slikama naučio bi se na smetnje i nebitne informacije i imao bi poteškoća s generalizacijom na nove primjere. Ovo se u području strojnog učenja zove velika pristranost primjerima za treniranje, odnosno *overfitting*.



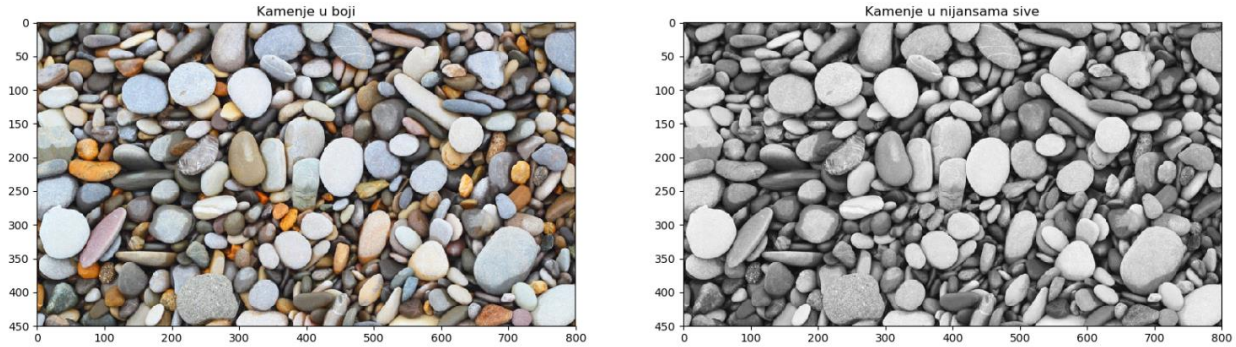
Sl. 3.2., Rukom pisani broj 9 s istaknutim nepravilnostima

Pronalazak značajki koje dobro opisuju podatke, odnosno koje ne uzimaju u obzir razne smetnje i reduntantne informacije, zove se *izdvajanje značajki*. Taj postupak će povećati točnost klasifikatora, a često i smanjiti vrijeme koje je potrebno za treniranje jer se dimenzionalnost vektora značajki smanjila [10]. Smanjenje dimenzionalnosti daje klasifikatoru bolju mogućnost generalizacije, a u isto vrijeme zadržava potrebne informacije za raspoznavanje primjera. Ovo nije uvijek slučaj jer nekad se broj značajki izdvajanjem može i povećati ako se time uvode bolje i kvalitetnije informacije.

3.2.1. Teksture

Tekstura je pojam koji nema točnu matematičku definiciju u računalnoj znanosti. Može se općenito definirati kao karakteristika ili izgled prizora, slike ili površine. Teksturu može tvoriti skupina malih elemenata kao zrna pijeska, većih elemenata kao stabljike kukuruza u polju ili ogromnih elemenata kao nebeska tijela u galaksiji. Također ju može tvoriti jedna površina koja ima razne udubine, izbočine i nesavršenosti koje ju opisuju. U digitalnim slikama, karakteristike teksture opisuju se prostornim varijacijama u intenzitetu slike, odnosno varijacijama u boji [11]. Opisnici tekstura zato dobro služe u opisu prizora koji se pojavljuju u prirodi jer ne ovise o točnoj boji koja je prikazana, već o odnosu boja na slici.

Metode za analizu tekstura razvijaju se na slikama u tonovima sive boje jer za dobar opis teksture informacija o boji nije potrebna. Ona može dodati interpretaciji teksture, ali time dodaje i veću računsku složenost, a ne donosi značajna poboljšanja. Dobar primjer tome je tekstura kamenja prikazana slikom 3.3.. Ljudima je vrlo je lako prepoznati da tekstura slike u nijansama sive predstavlja skupinu kamenja, dok dodatna informacija o boji ne donosi previše novih informacija osim „kamenje se razlikuje u boji“. Kada bi bilo potrebno napraviti model strojnog učenja koji određuje prikazuje li neka slika kamenje ili ne, čak je i bolje ako ne postoji informacija o boji, jer bi se klasifikator tada mogao naučiti razlikovati kamenje na osnovi boje, što na kraju nije bitno u pitanju „je li na slici kamenje?“. Dodano tome, ako je slika u boji dimenzija 800x450 piksela, kako bi računalo prikazalo informaciju o boji, mora prikazati sliku kao trodimenzionalnu matricu dimenzija 800x450x3 što je trostruko povećanje dimenzionalnosti u usporedbi sa slikom u sivim tonovima koja se lako prikaže matricom intenziteta dimenzija 800x450.



Sl. 3.3., Tekstura kamenja u boji i nijansama sive

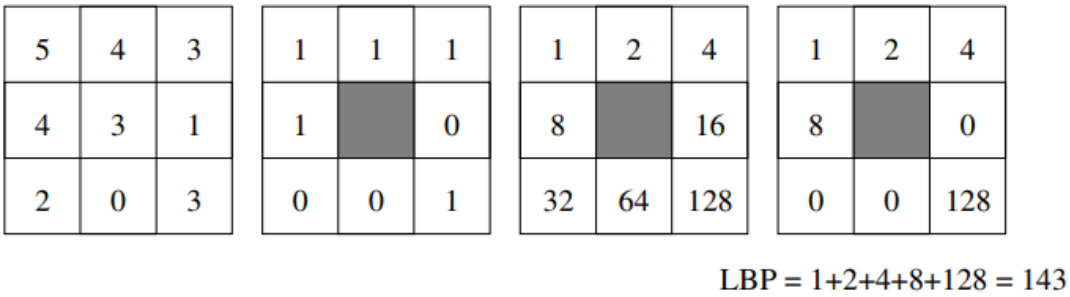
Sličan princip primjenjuje se i u korištenoj bazi slika gdje su informacije o boji ruke nebitne. Za klasifikaciju znaka nije bitna boja kože korisnika, već samo znak kojeg pokazuje.

3.2.2 Lokalni binarni uzorci

Lokalni binarni uzorci su jedna od metoda kojom je moguće opisati teksturu slike ili dijela slike. Vrlo je moćan opisnik tekstura koji je otporan na mnoge smetnje koje nastaju u procesu snimanja slika. Lokalni binarni uzorak računa se za svaki piksel slike u sivim tonovima.

Osnovna i najjednostavnija metoda računanja lokalnog binarnog uzorka za piksel slike računa se ovako:

1. Odabrati piksel i pronaći njegovih 8 susjeda
2. Ako susjedni piksel ima veći ili jednak intenzitet od odabranog piksela, na njegovo mjesto pisati „1“, a u suprotnom „0“
3. Pomnožiti dobivene vrijednosti s maskom težina svakog piksela
4. Lokalni binarni uzorak je zbroj pomnoženih vrijednosti
5. Ponoviti korake 1-4 za svaki piksel slike

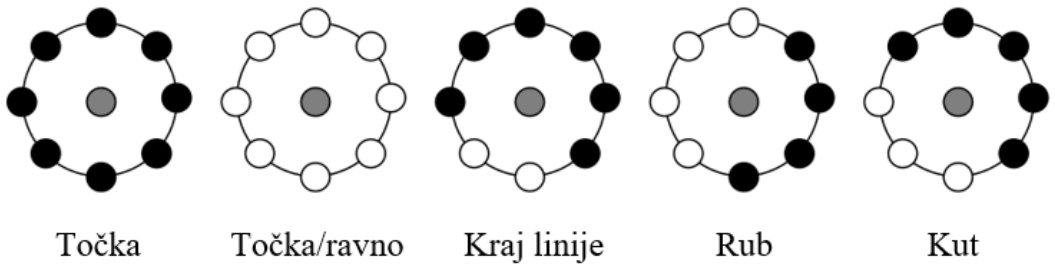


Sl. 3.4., Postupak računanja lokalnog binarnog uzorka (LBP) za središnji piksel [11]

Opisana metoda je originalna (engl. *default*) metoda koju su Ojala et. al predstavili u [12] i za 8 susjednih piksela daje 256 različitih lokalnih binarnih uzoraka. Općenito za p susjeda daje 2^p uzoraka.

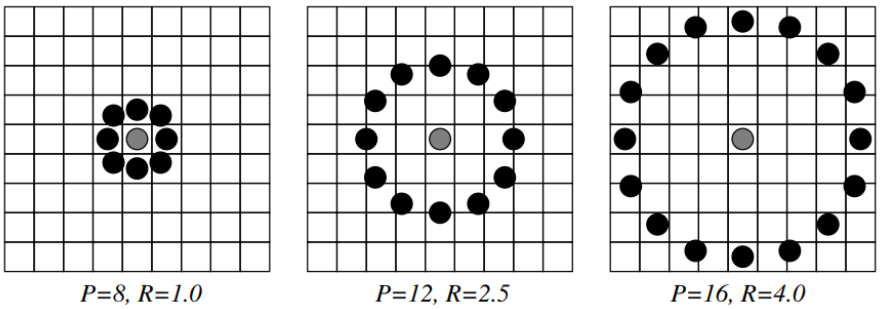
Postoji i uniformna metoda koja je varijacija originalne metode [13]. Za 8 susjednih piksela generira 58 uniformnih uzoraka, a sve ostale ne-uniformne uzorke broji kao jedan, što za 8 susjednih piksela daje 59 uzoraka. Uniformni uzorci su oni koji nemaju nijedan ili dva prijelaza iz 0 u 1 ili iz 1 u 0.

Uniformna metoda je bitna jer takvi uzorci dobro opisuju rubove, kutove, krajeve linija ili točke. Često su upravo takvi podaci zanimljivi pri razlikovanju objekata pomoću njihovih slika. Time se ne gubi informacija, a značajno se smanjuje dimenzionalnost s 256 značajki originalne metode na samo 59 značajki uniformne metode.



Sl. 3.5., Primjer uniformnih uzoraka sa susjedstvom 8 piksela i što predstavljaju po uzoru na [11]

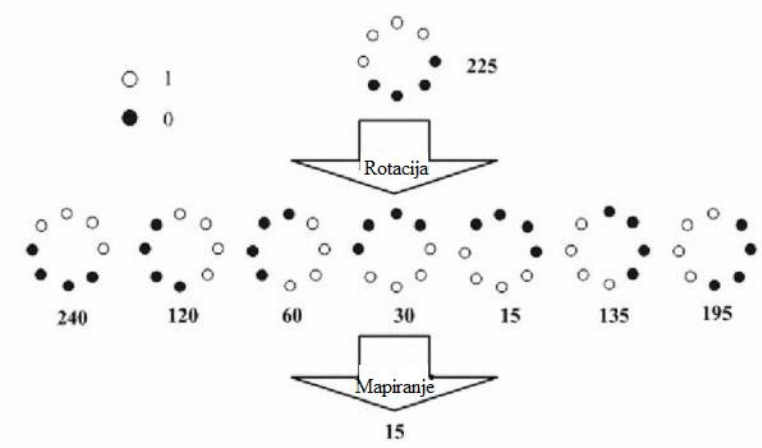
Originalna metoda koristi susjedstvo radijusa 1 oko promatranog piksela i uzima u obzir svih 8 susjednih piksela. Međutim, u kasnijim proširenjima originalnog algoritma, pokazalo se korisnim varirati oba od tih parametara [14]. Iz ulaznih podataka nije uvijek očito koji radijus i broj točaka je najbolji za pojedini problem, pa se dostupni parametri variraju i provjeravaju kako bi se utvrdilo koji su optimalni.



Sl. 3.6., Različiti radijusi i brojevi susjeda [11]

Lokalni binarni uzorci postižu neovisnost o monotonim razlikama osvjetljenja. To je očito jer ako se intenzitet svakog piksela poveća za istu vrijednost, odnosi piksela će ostati isti, a time će i lokalni binarni uzorci ostati isti.

Modifikacijom opisanih metoda moguće je postići i neovisnost o rotaciji. Isti se uzorci pojavljuju više puta, samo su drugačije rotirani. Ako se jednim predstavnikom prikažu sve moguće rotacije uzorka postiže se rotacijska neovisnost.



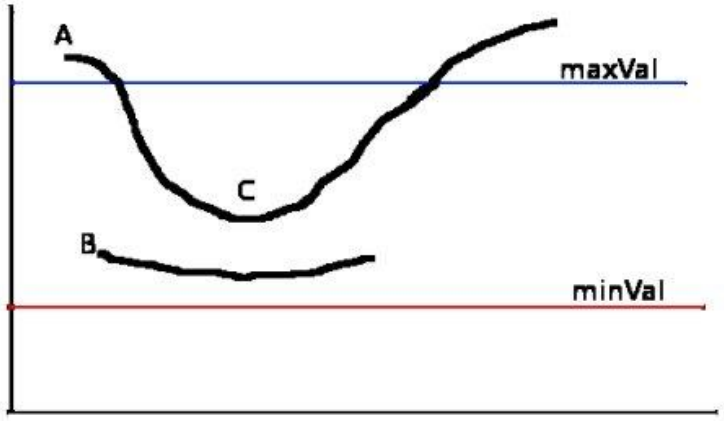
Sl. 3.7., Rotacijska neovisnost LBP po uzoru na [15]

Tvorba histograma uzoraka također rezultira neovisnošću o lokaciji teksture na slici. Primjerice, tekstura koja se nalazi u lijevom kutu slike imat će drugačiji poredak lokalnih binarnih uzoraka od iste slike koja tu teksturu ima u desnom kutu. Međutim, količina i vrsta lokalnih binarnih uzoraka će biti ista, što će nam dati isti konačan *broj* uzoraka, neovisno o njihovoj prostornoj lokaciji na slici.

3.2.3. Canny detekcija rubova

Canny detekcija rubova popularan je algoritam za detekciju rubova kojeg je razvio John F. Canny 1986. godine [16]. Postoji još dosta algoritama detekcije rubova, ali ovaj je bitan jer naglašava samo jake rubove i predstavlja ih tako da je svaki rub širine jednog piksela. Nadalje, ovaj algoritam binarizira sliku tako da na izlaz daje sliku gdje pikseli bijele boje predstavljaju otkrivene rubove, a crni pikseli predstavljaju sve što nisu rubovi (pozadina, buka, itd).

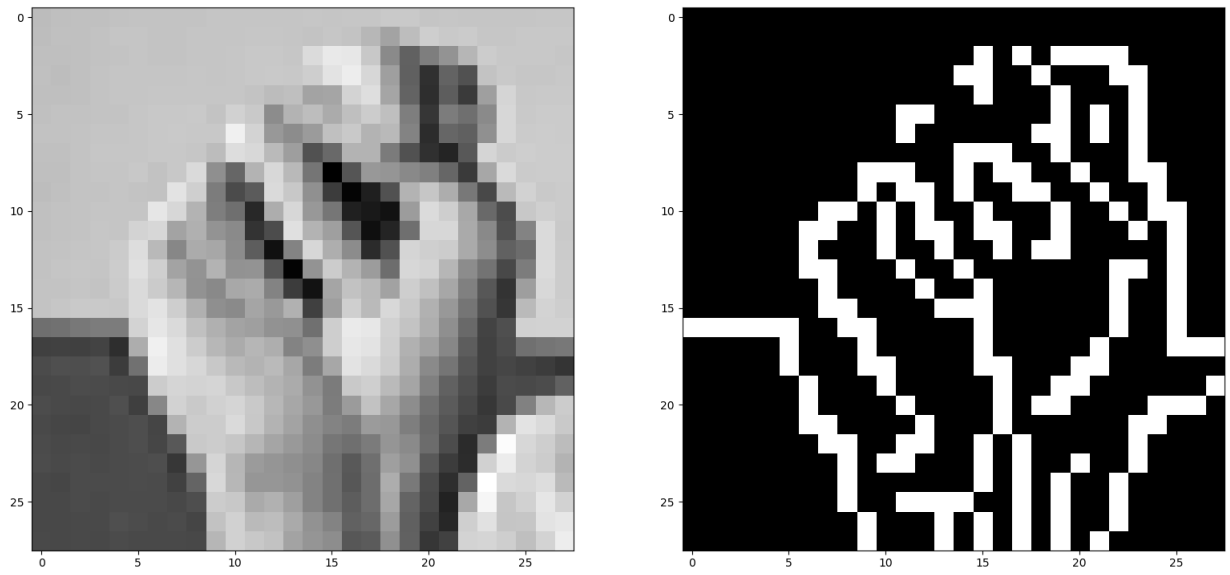
Algoritam prvo primjenjuje Gaussov filtar kako bi ugladio sliku i eliminirao buku koja bi se potencijalno detektirala kao rub. Zatim se filtrira Sobelovim operatorom što na izlaz daje gradijente i smjerove rubova svakog piksela slike. Za svaki se piksel slike zatim vrši provjera je li maksimum svog susjedstva u smjeru gradijenta. Ako je, onda je taj piksel rubni piksel i preživljava do sljedećeg koraka. Ako ne, odbacuje se. U zadnjem se koraku pomoću parametra donje granice detekcije ruba (*minVal*) i parametra gornje granice detekcije ruba (*maxVal*) određuju rubovi.



Sl. 3.8., Određivanje konačnih rubova u 4. koraku [17]

Rub A je siguran rub jer se nalazi iznad *maxVal*. Rub C nije siguran rub, ali jer je *spojen* s rubom A, smatra se sigurnim rubom. Slično tome, rub B ima veći gradijent intenziteta od *minVal*, ali jer nije iznad *maxVal* i nije spojen ni s jednim rubom koji je iznad *maxVal*, odbacuje se.

Bitno je pogoditi optimalne vrijednosti parametara *minVal* i *maxVal* za korištene slike kako bi se detektirali baš oni rubovi koji su značajni za specifični problem. Otkrivenim rubovima bit će dodijeljena vrijednost 255 (bijela boja). Ovaj korak će također ukloniti buku jer se za rubove uzimaju dugačke linije, dakle točkice i smetnje neće preživjeti ovaj korak.



Sl. 3.9., Primjer rezultata Canny detekcije rubova s *minVal*=100 i *maxVal*=200

3.2.4. Osnovna analiza komponenti (engl. *PCA*)

Osnovna analiza komponenti je statistička metoda smanjenja dimenzionalnosti koja nastoji komprimirati ulazne podatke, a pri tome očuvati što više originalne informacije. Drugim riječima, to je metoda koja projicira ulazne podatke u niže dimenzionalan prostor.

Vrlo se često koristi za probleme strojnog učenja jer može značajno smanjiti dimenzionalnost problema, a još uvijek predstaviti skoro istu količinu informacije. Smanjenjem dimenzionalnosti se smanjuje tendencija klasifikatora za *overfitting*, odnosno previše blisku prilagodbu na podatke za treniranje čime gubi sposobnost generaliziranja na nove primjere. Također značajno smanjuje vrijeme treniranja klasifikatora.

Ova metoda je osjetljiva na skalu podataka, pa je vrlo bitno prije primjene metode standardizirati podatke, a jedan od načina dan je izrazom

$$z = \frac{x - \mu}{\sigma} \qquad (3-1)$$

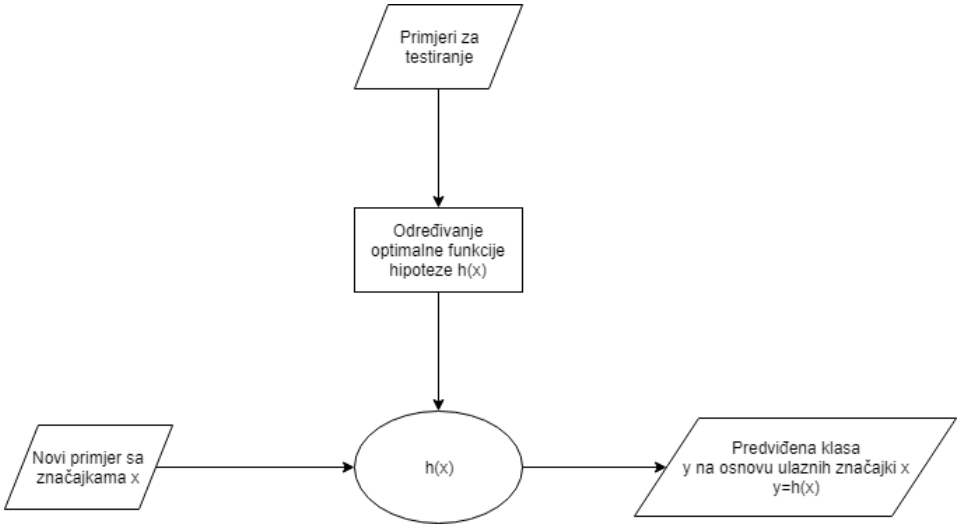
gdje je z standardizirana značajka, x originalna značajka, μ srednja vrijednost svih značajki tipa x, a σ standardna devijacija svih značajki tipa x.

Standardizacija podataka često se koristi u strojnom učenju jer se mnogi klasifikatori ne ponašaju najbolje kada se ulazni podatci jako razlikuju u skali (primjerice kad vrijednosti nekih značajki variraju od -3 do 1, a neke od 50 do 6000). Standardizacija sve te podatke svede u isti interval, odnosno standardno ih rasporedi (Gaussova distribucija).

3.3. Klasifikatori

Klasifikator je algoritam nadziranog strojnog učenja čija je svrha identificirati kojoj klasi pripada neki novi primjer koji je opisan određenim značajkama.

Jednostavnim riječima, klasificiranje se odvija pomoću funkcije koja je napravljena tako da najbolje odgovara primjerima za treniranje. Takva funkcija zove se funkcija hipoteze (engl. *hypothesis function*). Treniranje ili učenje klasifikatora, odnosno strojno učenje, je postupak pronalaska hipoteze koja najbolje odgovara podacima za treniranje. To je funkcija koja daje izlaz (predviđanje) u ovisnosti o ulaznim podacima (značajkama primjera).

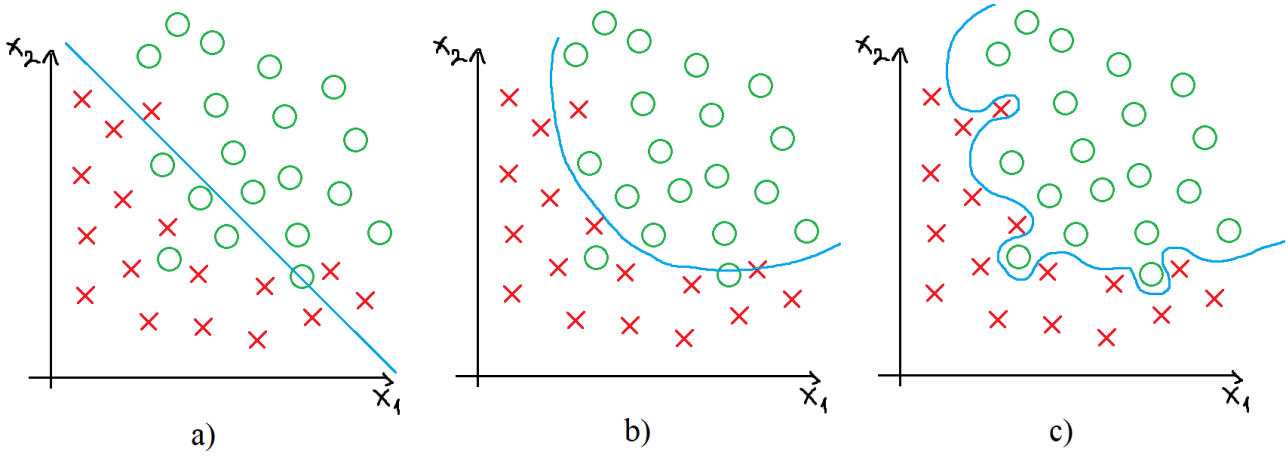


Sl. 3.10., Dijagram hipoteze i njezine svrhe

U problemu klasifikacije, naučena funkcija hipoteze tvori granicu odluke (engl. *decision boundary*). To je naučena granica koja razdvaja prostor značajki (engl. *feature space*) tako da svaki njegov dio predstavlja određenu klasu. Ako neki novi primjer ima značajke koje padaju u dio prostora određene klase, tada se taj novi primjer klasificira kao klasa prostora kojemu pripadaju te značajke.

Pronalazak granice odluke, odnosno hipoteze, razlikuje se od algoritma do algoritma. Pri određivanju potrebno je paziti na previše dobro odgovaranje podacima (engl. *overfitting*) i na slabo odgovaranje podacima (engl. *underfitting*).

Lako je postaviti klasifikator tako da savršeno dobro odgovara podacima na kojima je treniran, ali je tada upitno hoće li on dobro klasificirati novi primjer. Također je lako napraviti klasifikator koji previše generalizira i svejedno ne reagira dobro na nove primjere jer nije predstavio dovoljno informacija u skupu podataka za treniranje.



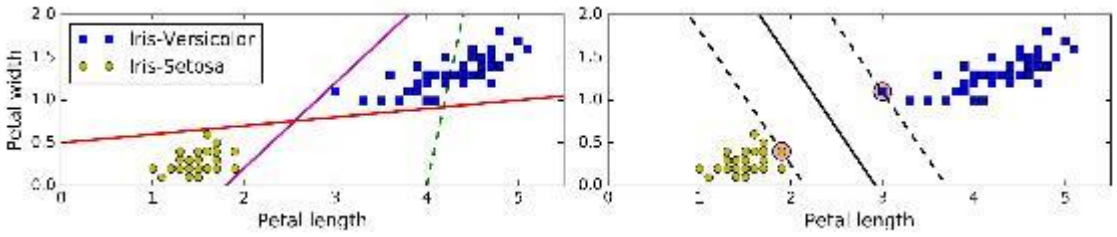
Sl. 3.11., *Underfitting* (a), dobra prilagodba (b) i *overfitting* (c) granice odluke na primjerima za treniranje

Slika 3.11. prikazuje različite oblike granice odluke koje su dobivene treniranjem klasifikatora na istom skupu za treniranje. Postoje dvije klase koje su predstavljene križićem i krugom, a klase su opisane dvjema značajkama x_1 i x_2. Slučaj a) pokazuje granicu odluke koja loše razdvaja klase jer nije dovoljno kompleksna. Ovo se još zove problemom visoke pristranosti (engl. *high bias*). Slučaj c) savršeno razdvaja primjere, ali takva je granica previše osjetljiva na buku i primjere uljeze (engl. *outliers*) koji se ne slažu s općenitim oblikom podataka. Drugim riječima, postoje neki krugovi koji su u prostoru križića i obratno, ali to nije općeniti slučaj za dotični skup podataka. To su samo slučajne pojave koje ne treba uzeti u obzir pri treniranju klasifikatora. Ovo se još zove problem visoke varijance (engl. *high variance*). Konačno, primjer b) pokazuje dobru granicu odluke. Uspješno zanemaruje uljeze i u većini slučajeva će napraviti točnu klasifikaciju. Intuitivno je vidljivo da je to logičan način za razdvojiti podatke koji su predstavljeni. Dobra granica odluke dobiva se optimalnim podešavanjem hiperparametara klasifikatora čime se kontrolira odnos između pristranosti i varijance.

3.3.1. Stroj s potpornim vektorima (engl. *Support Vector Machine*)

Stroj s potpornim vektorima (SVM) vrlo je popularan linearni klasifikator koji se može koristiti i kao nelinearni klasifikator pomoću jezgrenog trika (engl. *kernel trick*).

Glavni cilj SVM-a je dobiti granicu odluke koja razdvaja primjere, ali da se pritom ta granica nalazi što je dalje moguće od značajki primjera. Na prvom grafu slike 3.12. prikazane su neke od mogućih granica odluke. Kao prvo, primjeri su u potpunosti linearno razdvojivi i obje granice odluke savršeno razdvajaju primjere. Međutim očito je da one nisu najbolji izbor jer su relativno blizu primjerima i moglo bi doći do pogrešaka kada se predstavi novi primjer. Na drugom grafu prikazana je granica odluke koju je SVM dobio. Ovdje su primjeri puno bolje razdvojeni i granica odluke je najdalje što može biti od primjera. Ovo je upravo osobitost SVM-a jer on određuje najširu moguću marginu oko granice odluke i zato se još zove klasifikator velike margine (engl. *large margin classifier*).

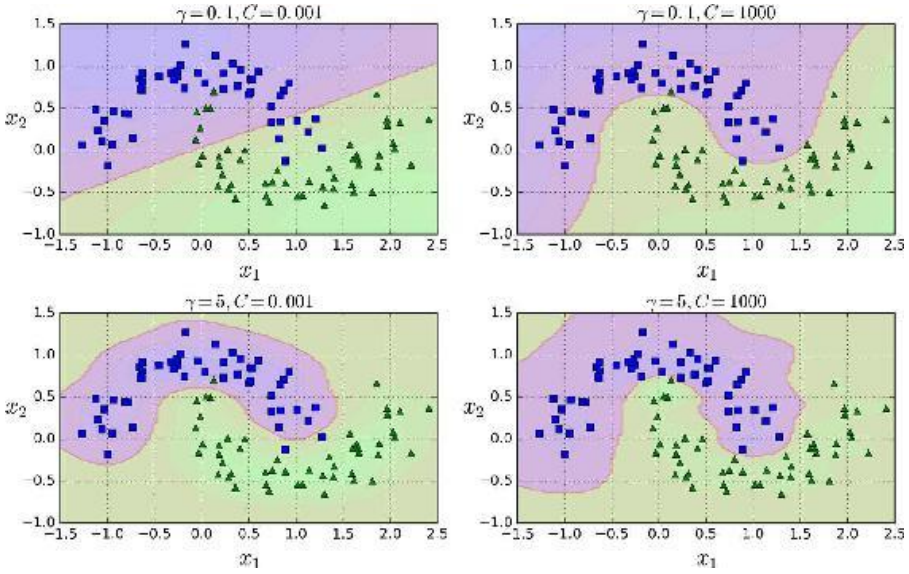


Sl. 3.12., Primjer granice odluke SVM-a [18]

Margine su određene s dva primjera koja su zaokružena. Bitno je primijetiti da se margine neće promijeniti ako se doda novi primjer za treniranje negdje izvan margina, odnosno margine ovise jedino o najbližim primjerima za testiranje koji „podupiru“ granicu odluke i oni se zovu *potporni vektori* [18].

U puno slučajeva primjeri nisu linearno razdvojivi i nemoguće je postići granicu odluke kao na slici 3.12. Ako se zahtijeva da su primjeri razdvojeni tako da nema niti jedan primjer unutar margina, ovo se zove klasificiranje s čvrstim marginama (engl. *hard margin*). Umjesto toga, u ovom radu koristit će se klasificiranje s mekim marginama (engl. *soft margin*) jer dopušta pojavu primjera unutar margina. Kod takvog pristupa manji hiperparametar C dat će veću marginu i više primjera u margini, a veći C dat će manju marginu i manje primjera u margini. Manji C rezultira većom regularizacijom što će smanjiti varijancu i obrnuto.

Drugo rješenje kod linearno nerazdvojivih podataka je uporaba jezgrenog trika. Često se kod linearno nerazdvojivih primjera uvode nove značajke većeg stupnja (kvadratnog, kubnog itd.), ali time se povećava dimenzionalnost i vrijeme koje je potrebno za treniranje. Pomoću jezgrenog trika postiže se učinak dodavanja značajki većeg stupnja bez da ih se zapravo doda, što čini SVM jednim od bržih klasifikatora. Postoje razne dostupne jezgre koje se mogu koristiti, ali najpopularnije su polinomna i Gaussova jezgra.



Sl. 3.13., Primjer Gaussove jezgre na linearno nerazdvojivim primjerima [18]

Slika 3.13. prikazuje razne granice odluke u ovisnosti o parametrima Gaussove jezgre γ i hiperparametru C. Vidljivo je da manji C daje dosta usku marginu i obrnuto. Povećanjem γ granica odluke poprima puno prirodniji oblik oko podataka i razdvaja plave primjere od zelenih.

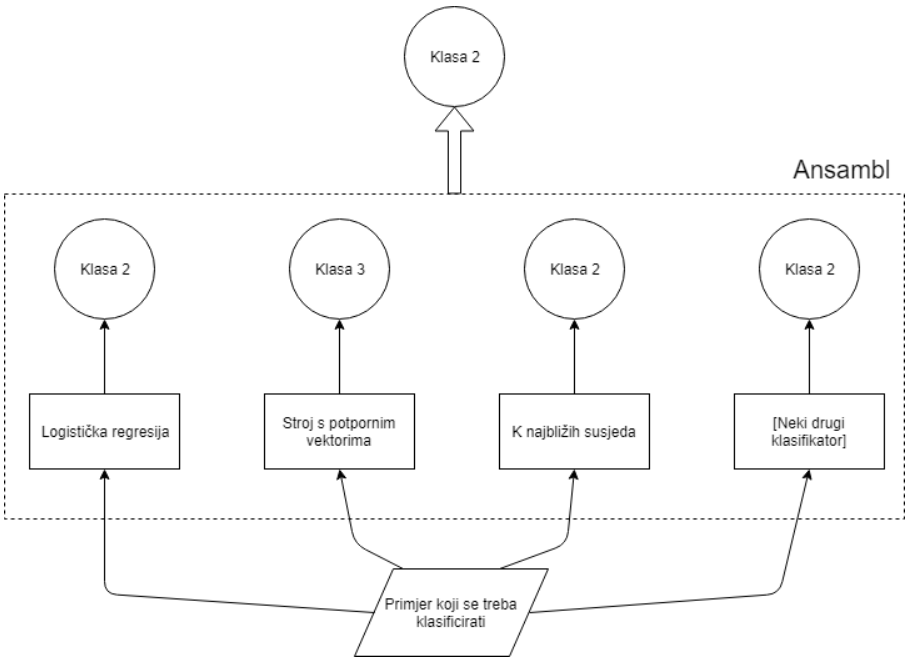
3.3.2. Šuma odluke (engl. *Random Forest*)

Ansambl metoda u strojnom učenju je metoda kojom se agregira više različitih klasifikatora i time dobiva veća moć klasifikacije od bilo kojeg pojedinačnog klasifikatora ansambla. Radi na principu „glasanja“, odnosno svaki klasifikator u ansamblu napravi svoju klasifikaciju i zatim se odabere ona klasa koja ima najviše glasova kao konačna predikcija.

Jedan pristup kojim se dobiva dobar ansambl je korištenje klasifikatora koji su na drugačije načine došli do svojih granica odluke. Ako se u ansambl uključi 50 strojeva s potpornim vektorima koji su trenirani na istim podacima, ne bi se vidjelo poboljšanje u usporedbi sa samostalnim SVM-om jer će

svaki od njih raditi iste pogreške i njihov glas će biti isti kao onaj samo jednog klasifikatora. Zato je poželjno imati više različitih klasifikatora.

Slika 3.14. prikazuje čvrsto glasanje (engl. *hard voting*) jer ovisi samo o predviđenim klasama. Konačna klasa se jednostavno odabere kao ona koja se najviše puta pojavila. Međutim, ako svi klasifikatori ansambla imaju mogućnost davanja sigurnosti predikcije u obliku vjerojatnosti da je klasa koju su predvidjeli zapravo ta klasa, može se implementirati meko glasanje (engl. *soft voting*) koje ima bolje performanse jer veću težinu daje sigurnijim predikcijama nego nesigurnim.



Sl. 3.14., Ilustracija ansambla i principa odluke

Drugi pristup efektivnom ansamblu je koristiti iste klasifikatore, ali trenirati svakog od njih na drugačijim podacima. Zbog različitosti podataka za treniranje klasifikatori će davati drugačije predikcije i griješiti na drugačijim stvarima, ovisno o tome kakve su podatke za treniranje dobili. Jedna od metoda kojom se dobivaju različiti podaci za treniranje zove se *bootstrap aggregating* ili *bagging* [19]. Ova metoda uzorkuje originalni skup primjera tako da nasumično odabere *n* primjera i zatim ih koristi za treniranje jednog od klasifikatora. Ovaj postupak se ponavlja za svaki od klasifikatora u ansamblu. Bitno je primijetiti da ova metoda dopušta duplikate primjera u uzorkovanju. Ista ova metoda, ali bez duplikata, zove se *pasting* [18].

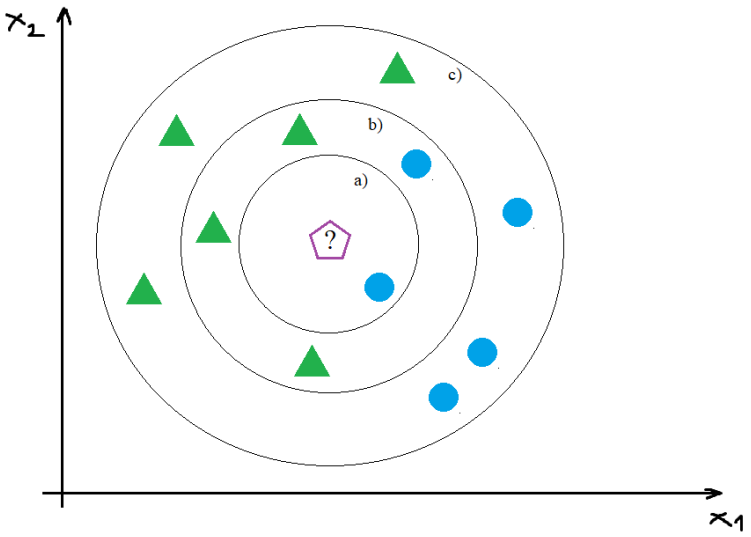
Šuma odluke je ansambl stabala odluke koji se treniraju *bagging* metodom. Metoda odluke koju ovaj ansambl koristi je meko glasanje jer stabla odluke imaju mogućnost davanja sigurnosti predikcije [20].

Uz *bagging*, šuma odluke također nasumično odabire određeni broj značajki koje će uzeti u obzir prilikom treniranja svakog od stabala odluke. Uzevši u obzir da se struktura stabala odluke mijenja s podacima za treniranje, šuma odluke koristi svaki od pristupa efektivnog ansambla i jedan je od jednostavnijih, ali i najefikasnijih algoritama strojnog učenja za višeklasnu klasifikaciju.

3.3.3. K-najbližih susjeda (engl. *K-nearest neighbors*)

K najbližih susjeda (KNN) jedan je od najjednostavnijih algoritama za višeklasnu klasifikaciju. Za razliku od već spomenutih algoritama, KNN ne konstruira generalni model (granicu odluke) pomoću koje klasificira nove primjere. Umjesto toga, sprema sve podatke za treniranje u memoriju i pomoću udaljenosti novog primjera od primjera koji su korišteni za treniranje provodi klasifikaciju.

Klasifikacija novog primjera vrši se određivanjem *k* najbližih primjera i njihovih klasa. Novi primjer bit će klasificiran kao najčešća klasa u tih *k* primjera. Na sl. 3.15. prikazan je primjer klasifikacije pomoću KNN u jednostavnom slučaju kada primjere opisuju dvije značajke i dvije klase (trokut i krug). U primjeru a) korišten je algoritam jednog najbližeg susjeda koji će klasificirati novi primjer kao najbliži primjer za treniranje, u ovom slučaju krug. Jedan najbliži susjed je algoritam koji je sklon velikoj pristranosti jer svoju odluku donosi samo na osnovi klase jednog primjera. Primjer a) je KNN s $k = 5$. U ovom slučaju novi primjer bit će klasificiran kao trokut. Primjer c) demonstrira KNN s $k = 11$ i tada je primjer klasificran kao trokut.



Sl. 3.15., Klasifikacija novog primjera koristeći KNN

Dobar odabir parametra *k* ovisi o primjerima za treniranje. Ako je *k* premalen, algoritam može patiti od velike varijance jer će u obzir uzeti samo prvi najbliži primjer. Tada će sustav biti jako osjetljiv na buku jer je moguće da se novi primjer baš nalazi najbliže primjeru za treniranje koji je uljez u cijelom skupu podataka. Povećanjem *k* dobiva se točnija klasifikacija i veća otpornost na smetnje, no povećava se računska složenost i pristranost.

3.4. Vrednovanje

Vrednovanje je postupak kojim se ocjenjuje kako se model strojnog učenja ponaša na ulaznim primjerima u ovisnosti o parametrima klasifikatora i o parametrima za predobradu primjera i izdvajanje značajki. Na osnovu provedenog vrednovanja mogu se mijenjati spomenuti parametri i tako podešavati model kako bi se ponašao što bolje na novim primjerima.

3.4.1. Stratificirana metoda izdvajanja

Za podešavanje hiperparametara klasifikatora i raznih metoda predobrade slike i izdvajanja značajki potrebni su neki primjeri na kojima se može testirati i vrednovati uspjeh određenih parametara. Jer su korišteni primjeri u bazi slika već podijeljeni na primjere za treniranje i primjere za testiranje naivan pristup bio bi za svaki skup parametara provesti predobradu i izdvajanje značajki na primjerima za treniranje i onda vrednovati kako se takav klasifikator ponaša na primjerima za testiranje.

Tada bi se odabrali oni parametri koji daju najbolje rezultate na primjerima za testiranje. No, kako tada dobiti ideju ideju o tome kako će se klasifikator ponašati s još neviđenim primjerima? Podešavanjem parametara klasifikatora na osnovu njegovih performansi na primjerima za testiranje rezultira klasifikatorom koji je fino podešen upravo tako da se ponaša najbolje na tim primjerima za testiranje. Tada je nemoguće saznati kako će se klasifikator ponašati s potpuno novim primjerima koje još nije vidio.

Primjere za testiranje potrebno je ostaviti netaknutima kako bi se na kraju podešavanja parametara modela mogla dobiti točna ocjena njegovog ponašanja na potpuno novim podacima. Stratificirana metoda izdvajanja omogućuje upravo to. Radi tako da cijeli skup primjera podijeli na dio za treniranje, testiranje i *validaciju*. Dio za validaciju je onaj dio na osnovu čijih performansi je moguće fino podešavati hiperparametre kako bi se ostvarile još bolje performanse. Tada ostaje netaknut skup primjera za testiranje pomoću kojega je moguće ocijeniti performansu modela za primjere koje još nikada nije vidio. Stratificiranost znači da će generirani skupovi primjera imati isti omjer klasa kao i originalni skup primjera.

Pri finom podešavanju hiperparametara klasifikator se trenira na generiranom skupu za treniranje i njegove performanse se provjeravaju na skupu za validaciju za svaku kombinaciju vrijednosti hiperparametara koje je potrebno provjeriti. Primjerice, ako se želi testirati koja od vrijednosti hiperparametara C = [0.1, 0.3, 1, 3, 10] najbolje odgovara stroju s potpornim vektorima, postupak bi se mogao izvesti ovako:

1. Učitati skup primjera za treniranje
2. Razdvojiti ga na nove skupove za treniranje, testiranje i validaciju
3. Vanjskom for-petljom iterirati kroz hiperparametre C
    a. U for petlji trenirati klasifikator na generiranom skupu za treniranje
    b. Testirati ga na skupu za validaciju
    c. Spremiti vrijednosti testiranja zajedno s hiperparametrima za koje su postignute te vrijednosti
4. Po završetku for-petlje odabrati optimalne hiperparametre kao one koji pripadaju maksimalnoj vrijednosti ocjene koju smo koristili za vrednovanje
5. Trenirati klasifikator na skupu za treniranje i testirati ga na skupu za testiranje kako bi se dobila ocjena njegove performanse u stvarnoj primjeni

3.4.2. Točnost

Točnost je najjednostavnija metoda vrednovanja performansi klasifikatora. Računa se kao postotak primjera koje je klasifikator točno predvidio. Zbog svoje jednostavne prirode, ova metoda je jedino točna ako primjeri za testiranje imaju ravnomjerno raspodijeljene klase, što često nije slučaj. Primjerice, ako je u pitanju problem klasifikacije s 10 klasa i trenira se klasifikator na nekim primjerima za treniranje i zatim se za testiranje koiristi skup primjera u kojemu prva klasa zauzima 90% skupa, a ostale klase se nalaze u preostalih 10%, tada bi klasifikator mogao biti samo jednostavna funkcija koja glasi: "za svaki skup ulaznih značajki, predvidi klasu 1". Time bi se postigla točnost od 90%, ali kada bi se distribucija klasa promijenila tako da svaka klasa zauzima 10% skupa, točnost bi se smanjila na 10%. Zato se koriste druge metode koje daju bolju ideju o tome kakve pogreške se javljaju i koje klase predstavljaju problem klasifikatoru neovisno o distribuciji skupova primjera i njihovoj simetričnosti.

3.4.3. Preciznost, odziv i F1 mjera

Preciznost (engl. *precision*) i odziv (engl. *recall*) su pojmovi koji su definirani za binarnu klasifikaciju. Binarna klasifikacija svodi se na pitanje: "predstavlja li ovaj primjer klasu koja me zanima?". Ako predstavlja, kaže se da je to pozitivan primjer, a ako ne, onda je to negativan primjer.

Matrica zabune C je takva matrica kojoj je element $C_{i,j}$ jednak broju predikcija koji pripadaju klasi *i*, ali su klasificirani kao klasa *j* [21]. Pomoću matrice zabune (tab. 3.1.) definiramo preciznost i odziv.

$$Preciznost = P = \frac{TP}{TP + FP} \qquad (3-2)$$

$$Odziv = R = \frac{TP}{TP + FN} \qquad (3-3)$$

Tab. 3.1., Matrica zabune kod binarne klasifikacije.

Matrica zabune	Zapravo +	Zapravo -
Klasificirano +	Stvarno pozitivan(TP)	Lažno pozitivan(FP)
Klasificirano -	Lažno negativan(FN)	Stvarno negativan(TN)

Dakle, preciznost se definira kao udio točno klasificiranih primjera u skupu pozitivno klasificiranih primjera, dok se odziv definira kao udio točno klasificiranih primjera u skupu svih pozitivnih primjera.

Preciznost i odziv zajedno daju bolju ideju o performansama klasifikatora i spajaju se u jedan broj pomoću mjere F1 koja se računa kao harmonijska sredina preciznosti i odziva.

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} = 2\frac{PR}{P+R} \qquad (3-3)$$

F1 mjera pridaje jednaku važnost preciznosti i odzivu, ali moguće je da to nije poželjno. Primjerice, ako se promatra problem u kojem se treba na osnovu krvne slike procijeniti ima li pacijent neku ozbiljnu bolest, poželjnije je napraviti neki broj lažnih pozitivnih klasifikacija, nego neki broj lažnih negativnih klasifikacija gdje će se predvidjeti da pacijent nema tu bolest, a zapravo ju ima. U ovom slučaju bitniji je odziv od preciznosti. Kako bi odziv bio što veći, potrebno je minimizirati broj FN.

Suprotno tome, ako se promatra problem u kojem je potrebno na osnovi riječi u e-pošti odrediti je li spam, poželjnije je napraviti što manje lažno pozitivnih klasifikacija jer bi neki bitan mail koji nije spam tada završio u spam mapi. Kod takvog problema bitnija je preciznost i kako bi ona bila što veća nastoji se minimizirati broj FP.

U općem slučaju mjera F računa se izrazom

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R} \qquad (3-4)$$

gdje se parametrom β naglašava preciznost ili odziv. Za $\beta < 1$, naglašava se preciznost, a za $\beta > 1$ naglašava se odziv [22].

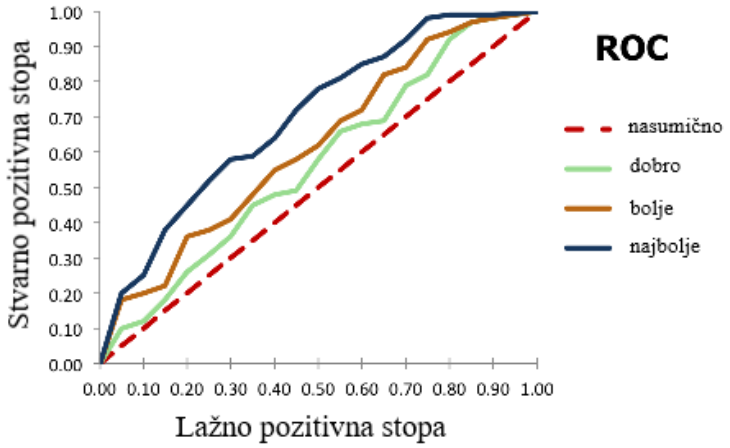
Preciznost, odziv i F1 mjera mogu se generalizirati i na višeklasne probleme. Za svaku klasu se posebno računa binarna matrica zabune tako da se predikcije te klase gledaju kao pozitivne, a predikcije svih ostalih klasa kao negativne. Kada se tako dobiju sve matrice zabune, one se stope na način da se zbroje vrijednosti svih ćelija kako bi se dobila jedna binarna matrica zabune. Za tu se matricu onda računa F1 mjera koja se zove *mikro F1-mjera*.

3.4.5. Površina ispod ROC krivulje

Receiver Operating Characteristic (ROC) krivulja opisuje odnos između stvarno pozitivne stope (osjetljivosti, odziva) (3-3) i lažno pozitivne stope (3-5) pri mijenjaju granice odluke binarnog klasifikatora.

$$FPR = \frac{FP}{FP + TN} \qquad (3-5)$$

Idealna ROC krivulja je ona koja ima stvarno pozitivnu stopu 1, a time lažno pozitivnu stopu 0. Ovo nije realistično, ali govori da je veća površina ispod ROC krivulje bolja. Idealna površina ispod ROC krivulje je 1. Površina ispod ROC krivulje od 0.5 odgovara nasumičnom klasifikatoru koji je bezvrijedan.



Sl. 3.16., Usporedbe ROC krivulja po uzoru na [23]

Površina ispod ROC krivulje koristi se kao parametar za vrednovanje binarnog klasifikatora jer je lakše vrednovati klasifikator koristeći jedan broj kao ocjenu umjesto krivulje.

U višeklasnim situacijama, ROC krivulje računaju se za svaku klasu posebno na principu „jedna protiv svih klasa“. Za sve te krivulje se zatim računa površina ispod krivulje. Srednja vrijednost svih površina ispod krivulja uzima se kao površina ispod krivulje tog klasifikatora.

4. EKSPERIMENTALNA ANALIZA

U ovom poglavlju implementirat će se postupci spomenuti u prošlim poglavljima i analizirati njihove performanse.

4.1. Korištene tehnologije

Kako bi se efikasno i jednostavno implementirali postupci strojnog učenja nema potrebe sve algoritme napisati samostalno jer je znanje koje je potrebno za to zaista opsežno. Umjesto toga, postoje već gotove, javno dostupne biblioteke za razne programske jezike koje daju jednostavan pristup raznim algoritmima predobrade slika, strojnog učenja i vrednovanja.

4.1.1. Python

Python je interpretirani programski jezik visoke razine s mogućnosti prilagodbe širokom spektru primjena i aplikacija. Podržava objektno-orijentirano i funkcionalno programiranje, a naglasak stavlja na urednost i čitljivost napisanog koda. Podržava pisanje ekstenzija i u drugim programskim jezicima kao C ili C++ koji su programski jezici niže razine (u usporedbi s Pythonom) i time imaju bolje vladanje memorijom i iskorištenjem sustavskih resursa. Ovo je bitno jer su algoritmi strojnog učenja izuzetno zahtjevni i vrijeme izvođenja nije zanemarivo. Tako su i biblioteke spomenute u ovom poglavlju većinom pisane u Pythonu, dok su neki dijelovi zbog boljih performansi pisani u C, C++ ili Cythonu.

Zbog toga postoje biblioteke koje implementiraju algoritme strojnog učenja i izlažu klase i metode koje su spremne za korištenje bez potrebe pisanja algoritama samostalno, što je vrlo zgodno jer za implementaciju algoritama strojnog učenja nije potrebno samo znanje o strojnom učenju, već i o naprednoj matematici. Python je odabran za eksperimentalnu analizu baš jer u njemu postoje sve bibilioteke koje su potrebne za brzu i efikasnu implementaciju opisanih algoritama bez potrebe za poznavanjem i pisanjem kompleksne matematike koja se nalazi ispod površine. Sve navedene biblioteke slobodne su za korištenje.

4.1.2. Scikit-learn

Scikit-learn [24] je opširna, besplatna Python biblioteka za strojno učenje koja pruža implementaciju mnoštva algoritama klasifikacije, regresije i grupiranja kao i algoritama za smanjenje

dimenzionalnosti, predobradu i odabir hiperparametara. Napravljena je tako da usko surađuje s postojećim numeričkim i znanstvenim bibliotekama NumPy i SciPy.

Od algoritama strojnog učenja koje nudi ova biblioteka, u ovom radu koriste se njihove implementacije šume odluke, k-najbližih susjeda i stroja s potpornim vektorima s linearnim kernelom. Također se koristi njihova implementacija osnovne analize komponenata i algoritama vrednovanja kao mjera F1, točnost, površina ispod ROC krivulje i matrica zabune.

4.1.3. Scikit-image i OpenCV

Scikit-image [25] i OpenCV [26] su Python biblioteke koje nude implementaciju raznih algoritama obrade slika. Nude funkcije za primjenu filtera, izdvajanje značajki popularnim metodama i slično. Od funkcionalnosti ovih biblioteka ovdje se koristi implementacija lokalnih binarnih uzoraka iz Scikit-image biblioteke i implementacija Canny detekcije rubova iz OpenCV biblioteke.

4.1.4. NumPy, Pandas i Matplotlib

NumPy [27] i Pandas [28] su popularne numeričke biblioteke Pythona koje omogućuju jednostavne i efikasne operacije s matricama. Pandas biblioteka se ovdje koristi za učitavanje skupova podataka iz .csv datoteka kao skupovi za treniranje, testiranje i validaciju. NumPy biblioteka se ovdje koristi za operacije nad matricama kao razdvajanje skupova podataka na dio koji predstavlja značajke i oznake primjera i za svaku ostalu operaciju koja zahtjeva manipuliranje matricama kao ulančavanje, dohvaćanje elemenata i slično. NumPy biblioteku interno koriste i biblioteke spomenute u prošlim potpoglavljima.

Matplotlib [29] je biblioteka koja služi za vizualizaciju podataka. Ovdje se koristi svaki puta kada treba vizualizirati podatke u obliku histograma, slika i grafova. Ima jednostavno aplikacijsko sučelje koje je puno mogućnosti i nudi jednostavan i pregledan prikaz željenih podataka što je od velike važnosti u problemima strojnog učenja.

4.2. Postavke eksperimenta

4.2.1. Postupak validacije

Kako bi se odabrala najbolja kombinacija hiperparametara korištena je stratificirana metoda izdvajanja u kojoj je originalni skup primjera za testiranje podijeljen na 60% primjera za validaciju i

40% primjera za testiranje. Svaka kombinacija hiperparametara validirana je na generiranom skupu primjera za validaciju pomoću mjere F1 i spremljena u odvojenu datoteku za kasniju analizu.

Tok provedbe validacije:

1. Učitati primjere za treniranje (27455 primjera) i validaciju (0.6 * 7172 = 4303 primjera)
2. Učitati polja koja sadrže hiperparametre koji se žele ispitati
3. Otvoriti .txt datoteke u koju će se zapisivati rezultati
4. Za svaku kombinaciju hiperparametara:
    a. Provesti izdvajanje značajki na skupu za treniranje i validaciju s trenutnim vrijednostima hiperparametara
    b. Normalizirati skupove
    c. Trenirati dotični klasifikator na obrađenim primjerima za treniranje
    d. Evaluirati mjeru F1 klasifikatora na primjerima za validaciju
    e. Spremiti rezultat zajedno s hiperparametrima u .txt datoteku
5. Odabrati najbolje hiperparametre pomoću najveće mjere F1

Bitno je primijetiti da u ovisnosti o metodi izdvajanja značajki i algoritmu strojnog učenja ovaj postupak može biti prilično dugačak (reda veličine nekoliko sati) jer je velik broj značajki i primjera za treniranje, a treniranje klasifikatora je vremenski najskuplji postupak.

4.2.2. Postupak testiranja na neviđenim primjerima

Kada je odabrana najbolja kombinacija hiperparametara validacijom, bitno je testirati takav sustav na još neviđenim podacima kako bi se utvrdile njegove stvarne performanse, odnosno kako bi se ponašao kada bi bio pušten u pogon na stvarnim slučajevima.

Tok testiranja i evaluacije klasifikatora je:

1. Učitati primjere za treniranje i testiranje
2. Učitati optimalne hiperparametre
3. Napraviti izdvajanje značajki i normalizaciju na primjerima za treniranje i testiranje s optimalnim hiperparametrima
4. Trenirati klasifikator na primjerima za treniranje
5. Evaluirati performansu klasifikatora pomoću točnosti, mjere F1, površine ispod ROC krivulje i matrice zabune

4.3. Izvedba eksperimenta i rezultati

4.3.1. Lokalni binarni uzorci

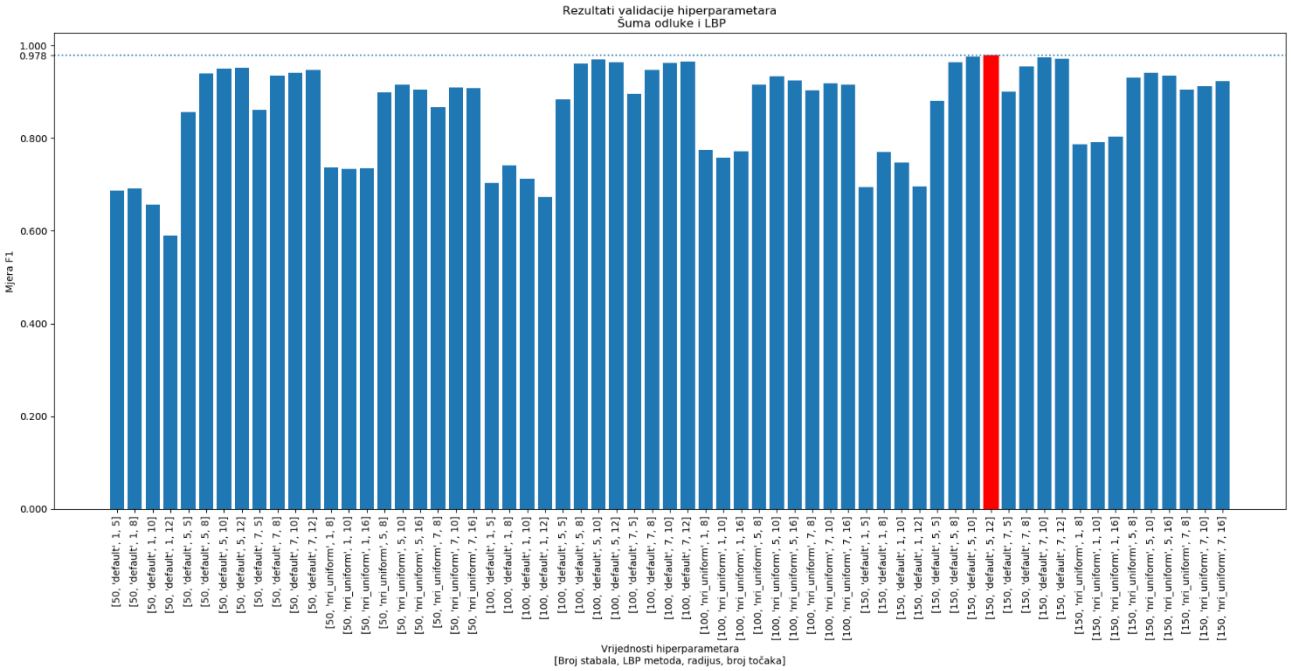
Pri validaciji lokalnih binarnih uzoraka testirana je „default“ i „uniform“ metoda. Bitno je napomenuti da je u scikit-learn implementaciji uniformne metode potrebno navesti metodu kao „nri_uniform“, a ne „uniform“. Prva metoda nije rotacijski neovisna, kao što je opisano u većini literatura, dok druga je. Vrijednosti točaka za default metodu idu do 12 jer se za taj broj točaka dobije vektor značajki duljine 4095. Sve iznad toga je preveliko i računski zahtjevno. Uniformna metoda omogućuje više točaka i manji vektor značajki pa zato može ići i do 16. Kombinacije hiperparametara koje su testirane navedene su u tab. 4.1.

Tab. 4.1., Testirani hiperparametri za LBP i klasifikatore.

Hiperparametar	Vrijednosti	
Metoda računanja LBP	„default“	„nri_uniform“
Broj točaka *p*	[5, 8, 10, 12]	[8, 10, 16]
Radijus *r*	[1, 5, 7]	
Broj stabala šume odluke	[50, 100, 150]	
Broj susjeda k najbližih susjeda	[3, 10, 30]	
Parametar regularizacije C stroja s potpornim vektorima	[0.01, 0.1, 1, 10]	

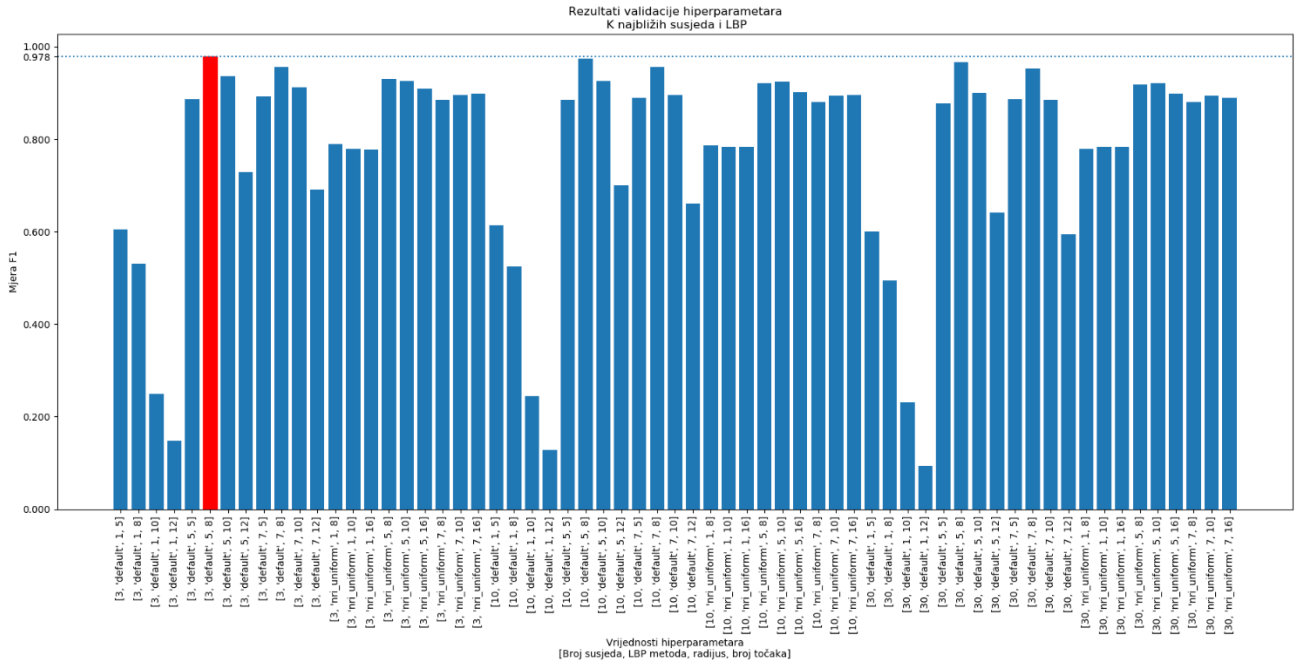
Značajke se formiraju izračunom lokalnih binarnih uzoraka. Zatim se tvori histogram uzoraka koji svakom uzorku pridjeljuje broj ponavljanja. Broj košara (engl. *bins*) jednak je broju lokalnih binarnih uzoraka kako ne bi došlo do gubitka informacije, no može se smanjiti ako je dimenzionalnost prevelika, naravno uz određeni gubitak informacija. Zatim se takav histogram normalizira i koristi za treniranje klasifikatora.

Slika 4.1. prikazuje stupčasti graf koji svakoj od 62 kombinacije hiperparametara za šumu odluke pridjeljuje dobivenu mjeru F1. Stupac označen crvenom bojom sadrži optimalne hiperparametre što je određeno najvećom mjerom F1. Može se zaključiti da šumi odluke najbolje odgovara kada je broj točaka i radijus veći od 5. Uniformna metoda u svakom se slučaju ponaša znatno lošije od default metode. Također se vidi da se povećanjem broja stabala povećava i mjera F1.



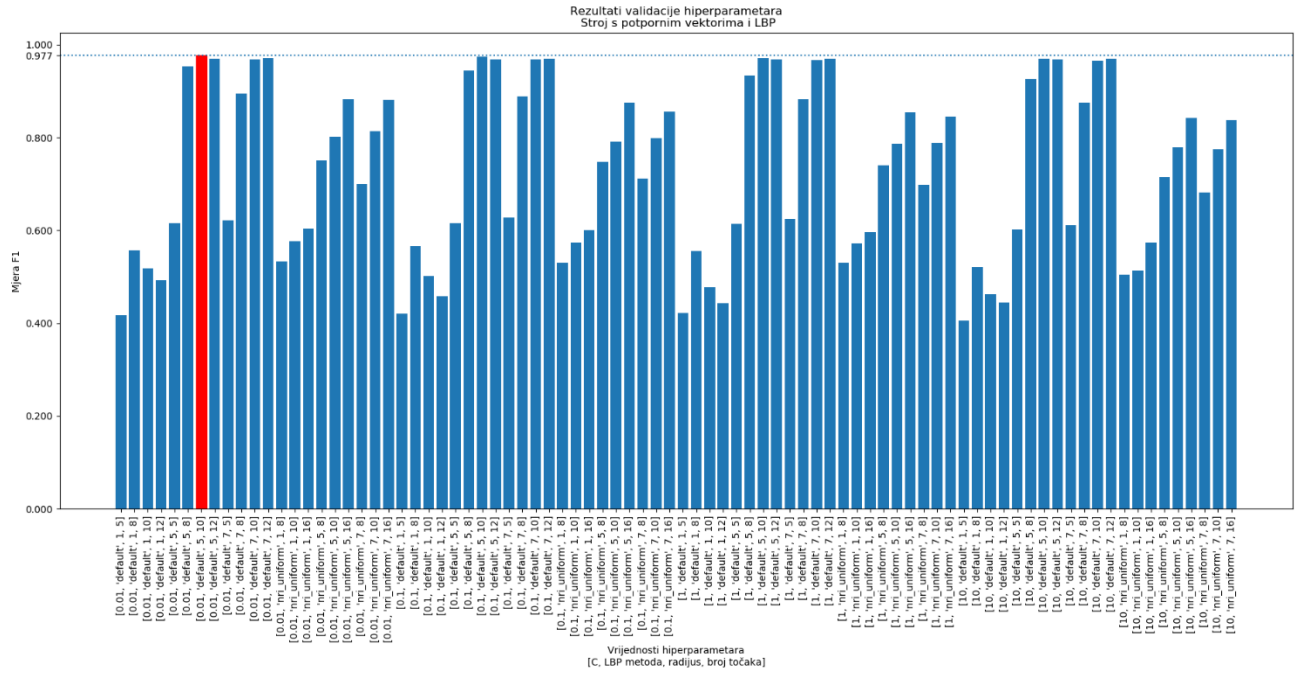
Sl. 4.1., Rezultati validacije hiperparametara za šumu odluke i LBP

Slika 4.2. prikazuje graf istog tipa, ali za KNN algoritam. Također se vidi da bolje reagira na veći broj točaka i radijus, kao i da mu default metoda općenito odgovara bolje od uniformne. Za kombinaciju LBP parametara od [‘default’, 5, 8] mijenjanje broja susjeda ne mijenja situaciju značajno. Svejedno, 3 susjeda su pobijedila s malom prednošću.



Sl. 4.2., Rezultati validacije hiperparametara za KNN i LBP

Prema slici 4.3., vidi se da za radijus 5, default metodu i broj točaka između 8 i 12, variranje hiperparametra C stroja potpornog vektora nema veliki utjecaj na mjeru F1. Vrijednost 0.01 je pobijedila s malom prednosti, ali najveći utjecaj imaju LBP parametri, što je bio slučaj i kod KNN.



Sl. 4.3, Rezultati validacije hiperparametara za SVM i LBP

Tablica 4.2. sažima najbolje rezultate validacije hiperparametara. Svi su klasifikatori postigli vrlo dobru mjeru F1, ali KNN pobjeđuje s malom prednošću.

Tab. 4.2., Optimalni hiperparametri lokalnih binarnih uzoraka i klasifikatora određeni mjerom F1.

	Hiperparam. klasifikatora	Metoda	Točke	Radijus	Mjera F1
Ansambl stabala odluke (RF)	Stabla: 150	default	12	5	0.978150
K najbližih susjeda (KNN)	Susjedi: 3	default	8	5	0.978382
Stroj s potpornim vektorima (SVM)	C: 0.01	default	10	5	0.976755

Zanimljivo je primijetiti da je vrijednost radijusa LBP za koju se dobio najbolji rezultat 5 za svaki klasifikator, što ukazuje na dobro pogođen hiperparametar. Također izgleda da je više točaka u ovom slučaju bolje. Niti jedan klasifikator nije preferirao uniformnu metodu. Moguć razlog ovome je što uniformna metoda stavlja naglasak samo na rubove, točke i slične značajke, a to vjerojatno ne daje potpuni opis teksture ruke. Svi ostali uzorci stavljaju se u jednu košaru i očito se pri tome gube bitne informacije.

SVM-u odgovara manji parametar C, odnosno veća regularizacija, što smanjuje *overfitting*. Isto se može reći i za RF koji preferira veći broj stabala. Međutim, KNN-u najbolje odgovaraju samo tri susjeda. Mali broj susjeda može lako dovesti do *overfitting*-a, ali u ovom slučaju klasifikatoru to odgovara i postiže najbolje rezultate s tako malenom vrijednosti.

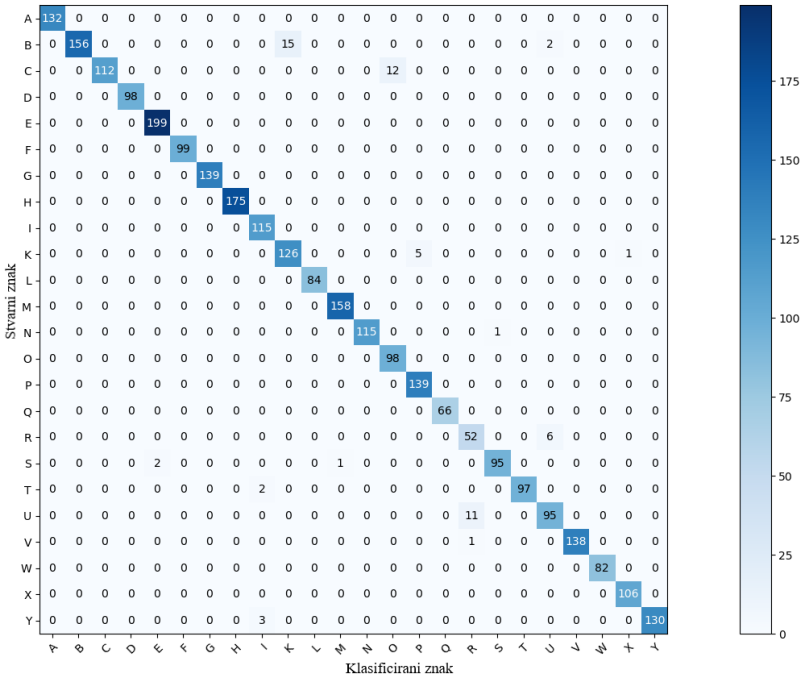
Nakon provedbe validacije, klasifikator je s optimalnim hiperparametrima testiran na odvojenom skupu za testiranje kojega do sada nije vidio. Mjera F1 i AUC izračunati su kao mikro srednja vrijednost (engl. *micro average*). Preciznost i odziv nisu posebno prikazani jer su u višeklasnoj klasifikaciji isti kao i mjera F1. Rezultati su prikazani tablicom 4.3.

Tab. 4.3., Rezultati testiranja klasifikatora s optimalnim hiperparametrima.

	Mjera F1	Površina ispod krivulje(AUC)	Točnost(ACC)
RF	0.978382	0.988721	0.978382
KNN	0.975244	0.987084	0.975244
SVM	0.975941	0.987447	0.975941

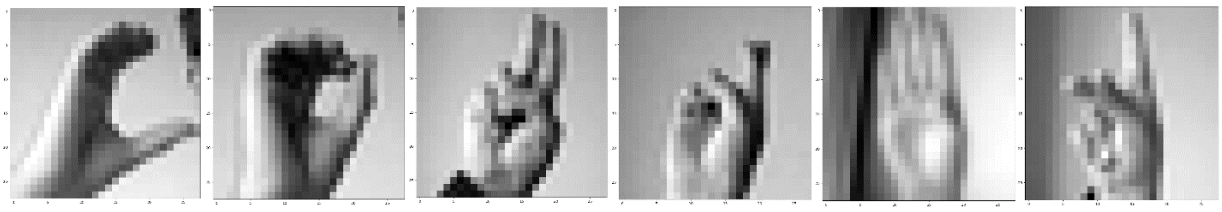
Pri testiranju klasifikatori imaju slične rezultate kao pri validaciji, međutim ovdje je RF bolji s malom prednošću. Za potpuni dojam treba komentirati matricu zabune koja će pokazati gdje svaki od klasifikatora ima poteškoće s razlikovanjem primjera.

Prema slici 4.4. RF ima najviše poteškoća s klasifikacijom znaka B kojega klasificira kao K, C kojega klasificira kao O i U kojega klasificira kao R. Ostale znakove skoro savršeno klasificira s malim iznimkama.



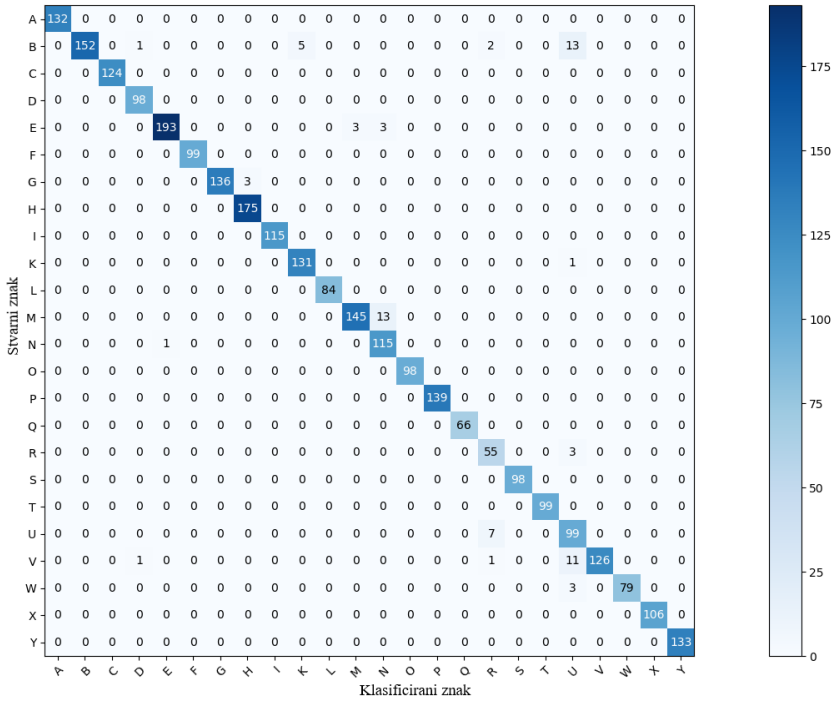
Sl. 4.4., Matrica zabune RF klasifikatora

Prema slici 4.5., pogreške u klasifikaciji znakova C i U imaju smisla jer su im znakovi O i R izgledom vrlo slični. Poboljšanje razlikovanja znakova C i O moglo bi se postići dodavanjem značajki koje ističu zatvorene petlje u ruci jer upravo zatvorena petlja razlikuje C od O. Poboljšanje razlikovanja znakova U i R je teže za odrediti jer su znakovi vrlo slični po obliku. Sofisticiraniji algoritam za detekciju rubova mogao bi biti od pomoći. Međutim, klasifikacija znaka B kao K je neočekivana i rješenje problema moglo bi biti skupljanje više primjera.



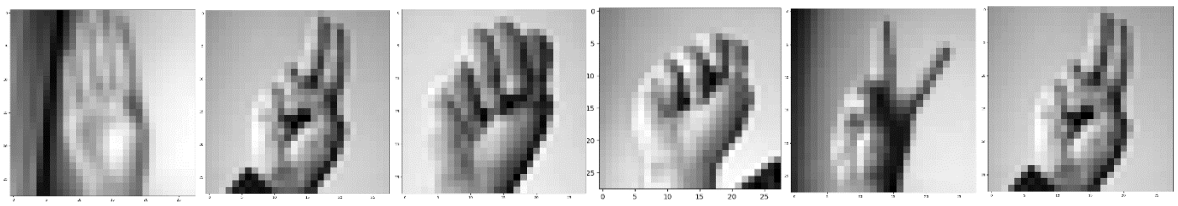
Sl. 4.5., Pogrešno klasificirani znakovi RF-a, s lijeva prema desno: C, O, U, R, B, K

Matrica zabune KNN klasifikatora na slici 4.6. pokazuje da KNN ima najviše poteškoća s klasifikacijom znaka B kao U, znaka M kao N i znaka V kao U. Ostale znakove skoro savršeno klasificira s malim odstupanjima.



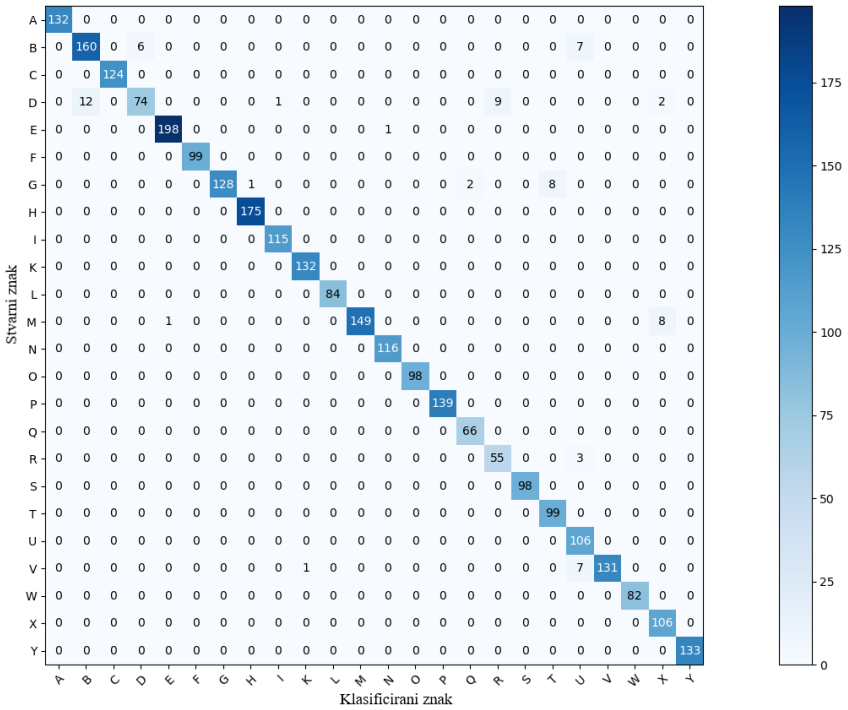
Sl. 4.6., Matrica zabune KNN klasifikatora

Prema slici 4.7., pogrešna klasifikacija znaka M kao N je razumljiva jer bi i čovjeku bilo teško razlikovati ta dva znaka iz slike. Razlika je samo u položaju palca ispod prstiju i teško je predložiti programsko rješenje ovog problema. Pogreška u klasificiranju znakova B i V kao znak U izgleda kao posljedica poteškoće u prepoznavanju broja podignutih prstiju i razlikovanje spojenosti i odvojenosti prstiju. U oba problema bi mogle pomoći značajke koje opisuju rubove.



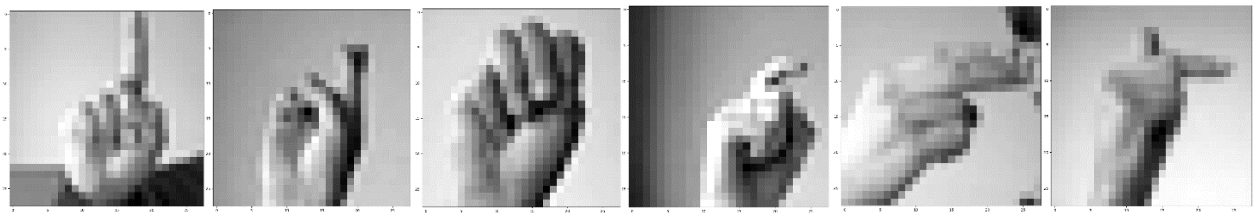
Sl. 4.7., Pogrešno klasicirani znakovi KNN-a, s lijeva prema desno: B, U, M, N, V, U

Slika 4.8. prikazuje matricu zabune za SVM klasifikator. Najčešće krivo klasificiran znak je D koji je 12 puta klasificiran kao B i 9 puta kao R. Osim toga M se klasificira kao X i G kao T.



Sl. 4.8., Matrica zabune SVM klasifikatora za LBP

Prema slici 4.9. SVM radi razumne pogreške i primjeri su stvarno teški za razlikovanje. Prepoznavanje znaka M moglo bi se poboljšati dodavanjem informacije o rubovima koja bi, teoretski, trebala dobro poslužiti kao opisnik prsta u znaku X i time razlikovati M i X. Za razlikovanje znaka G od T moglo bi poslužiti brojanje prstiju.



Sl. 4.9., Pogrešno klasificirani znakovi SVM-a, s lijeva na desno: D, R, M, X, T

Sve u svemu, lokalni binarni uzorci pokazali su se kao odlična metoda izdvajanja značajki koja dobro razlikuje skoro svaki primjer. Kao dodatak na značajke lokalnih binarnih uzoraka mogu se dodati neke nove značajke koje bi daljnje poboljšale klasifikaciju promaknutih primjera koji su opisani u ovom potpoglavlju. Kasnije će se proučiti kako se algoritam ponaša ako se vektoru značajki dodaju opisnici rubova.

4.3.2. Canny detekcija rubova

Kao i za LBP, provedena je validacija Canny detekcije rubova za razne hiperparametre koji su prikazani u tab. 4.4.

Tab. 4.4., Testirani hiperparametri za Canny i klasifikatore.

Hiperparametar	Vrijednosti
Gornja granica Canny detektora *maxVal*	[100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 255]
Donja granica Canny detektora *minVal*	*minVal = maxVal/2*
Broj stabala RF	[50, 100, 150]
Broj susjeda KNN	[3, 10, 30]
Regularizacija C SVM	[0.01, 0.1, 1, 10]

Značajke su tvorene izračunom Canny detekcije rubova i tvorbom dvaju histograma; jedan koji prati broj rubova za svaki redak i drugi koji prati broj rubova za svaki stupac. Ta dva histograma duljine 28 ulančavaju se u jedan rezultirajući vektorom značajki duljine 56. Takav vektor značajki se zatim normalizira i koristi za treniranje klasifikatora. Ideja ovog pristupa je izolirati informaciju o rubovima na slici što će, s imalo sreće, dati bolju informaciju o položaju prstiju u ovisnosti o vertikalnom i horizontalnom položaju i k tome držati vektor značajki kompaktnim.

Iz rezultata validacije prikazanih u tab. 4.5. očito je da sami histogrami rubova nisu dovoljni za zadovoljavajuće razdvajanje primjera. Najbolji rezultat ima šuma odluke, ali to svejedno nije

usporedivo s rezultatima lokalnih binarnih uzoraka. Daljnje testiranje i evaluacija neće se provoditi jer samostalno ova metoda ne daje dovoljno dobre rezultate.

Tab. 4.5., Optimalni hiperparametri Canny detektora.

	Hiperparam. klasifikatora	Niska granica(minVal)	Visoka granica(maxVal)	Mjera F1
RF	Stabla: 150	110	220	0.675499
KNN	Susjedi: 3	70	140	0.563691
SVM	C: 0.01	85	170	0.439562

Svejedno, zanimljivo je primijetiti da je samo 56 značajki dovoljno kako bi se postigla vrijednost mjere F1 od 0.67. Originalne slike imaju 784 značajke i dosta njihovih informacija je izgleda sadržano upravo u rubovima.

4.3.3. Osnovna analiza komponenata

Pri validaciji osnovne analize komponenata (PCA) mijenjan je postotak varijance. Prema *Scikit-learn* dokumentaciji, varijanca određuje broj generiranih osnovnih komponenti kao minimalan broj komponenti koji opisuje podatke s varijancom većom ili jednakom onoj koja je specificirana. Parametar poprima vrijednosti u rasponu <0,1>. Parametri koji su validirani prikazani su tablicom 4.6.

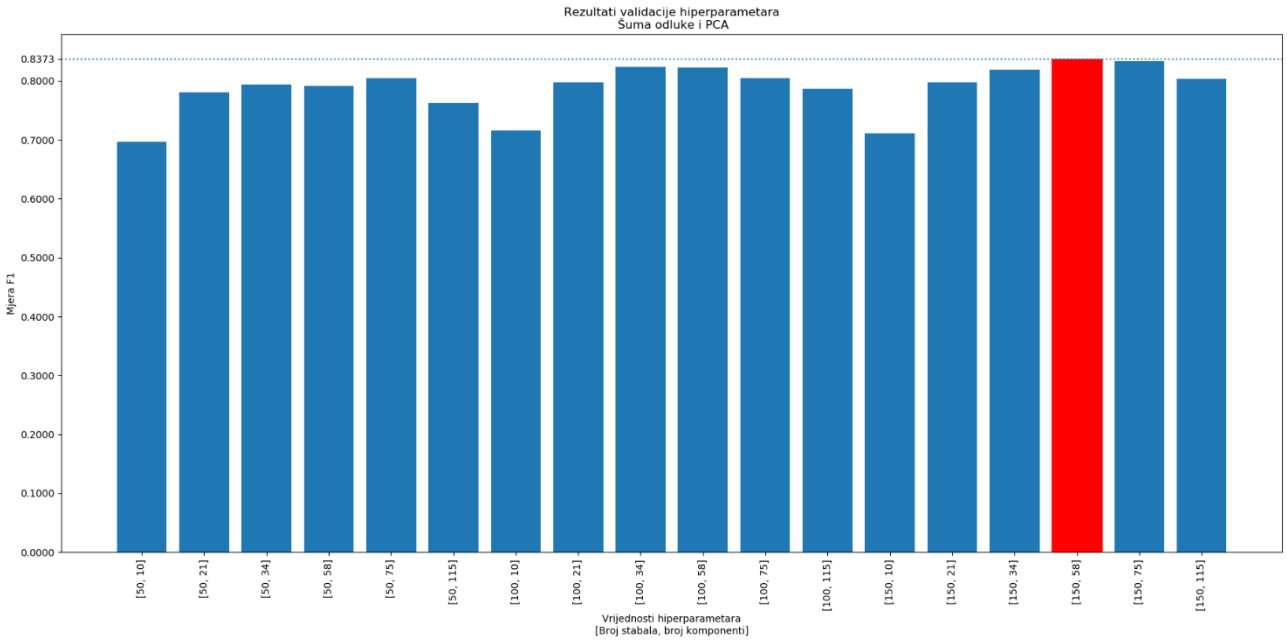
Tab. 4.6., Testirani hiperparametri za PCA i klasifikatore.

Hiperparametar	Vrijednosti
Varijanca	[0.7, 0.8, 0.85, 0.9, 0.92, 0.95]
Broj stabala RF	[50, 100, 150]
Broj susjeda KNN	[3, 10, 30]
Regularizacija C SVM	[0.01, 0.1, 1, 10]

PCA bi trebao pri dovoljno velikom parametru varijance očuvati što više originalnih informacija, ali značajno smanjiti broj značajki što će dati brže vrijeme izvođenja. Priroda korištene baze slika je da imaju dosta pozadinske buke koja nije bitna za klasifikaciju pa bi se broj značajki trebao dosta smanjiti, a performansa klasifikatora bi trebala ostati relativno nepromijenjena.

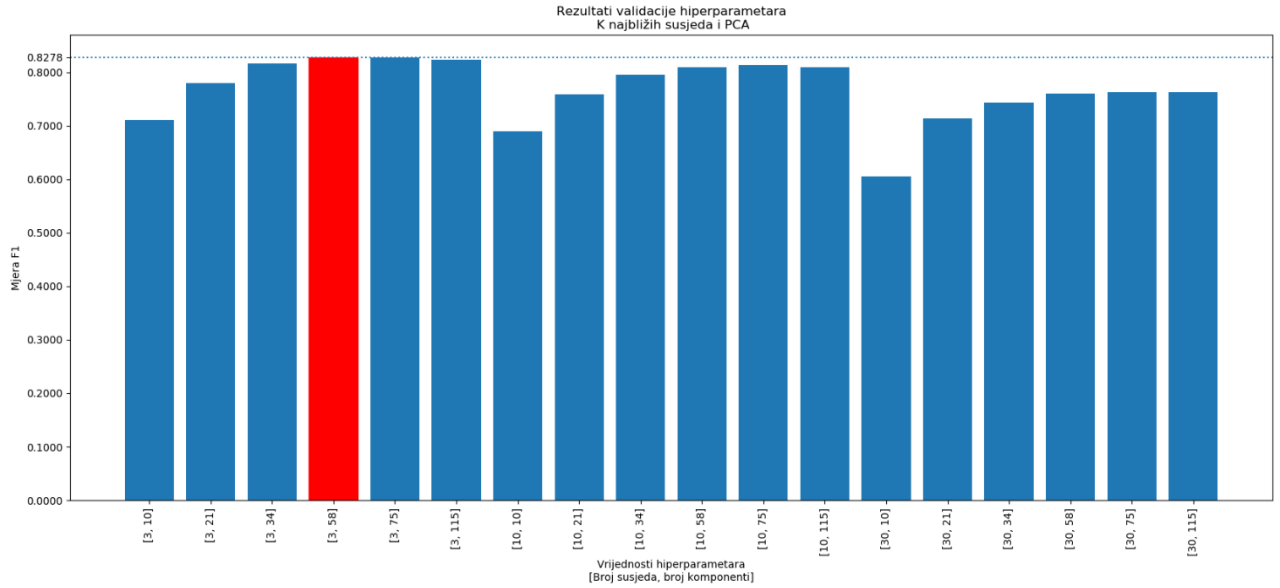
Slika 4.10. prikazuje rezultate variranja hiperparametara za šumu odluke s PCA algoritmom. Vidi se da je optimalan broj komponenti između 34 i 75 i da je najbolji rezultat postignut za 150 stabala i 58 komponenti. Zanimljivo i neočekivano je da se mjera F1 smanjuje za broj komponenti iznad 75 jer bi

imalo smisla da više informacija daje bolju razlikovnu moć. Isti postupak odluke za najbolju kombinaciju hiperparametara proveden je za SVM i KNN.

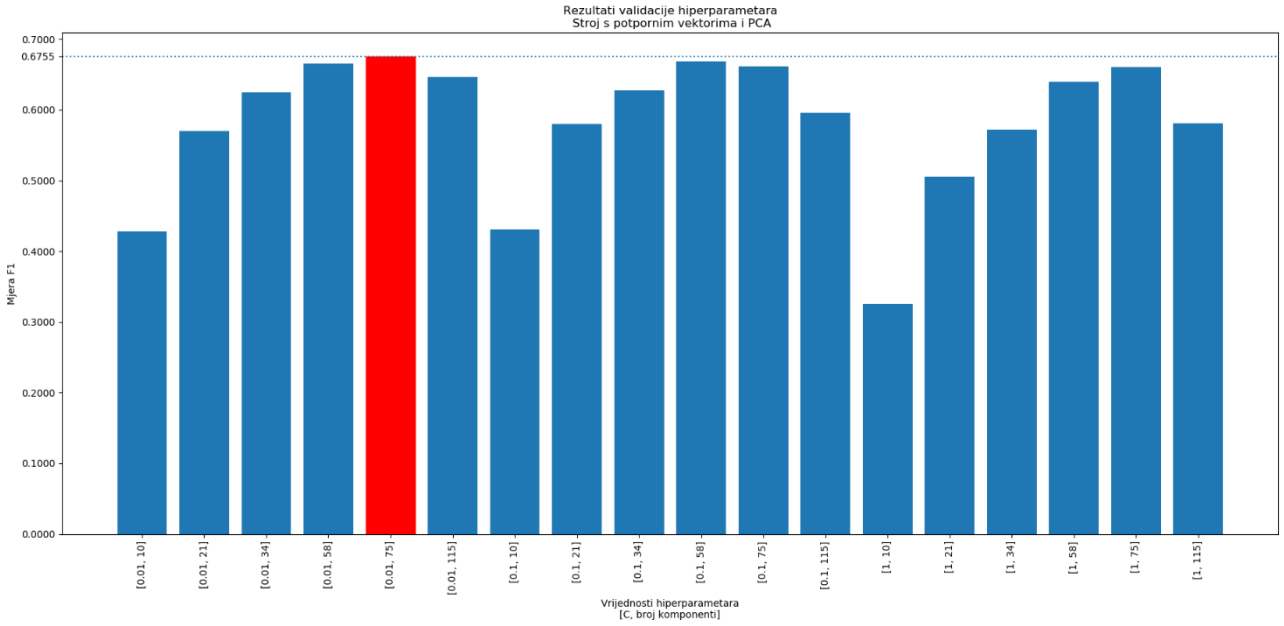


Sl. 4.10., Rezultati validacije hiperparametara šume odluke i PCA

Nešto slično se događa i kod KNN klasifikatora prema slici 4.11. gdje se povećanjem broja komponenti smanjuje mjera F1, ali je ovdje očito da se povećanjem broja susjeda također smanjuje mjera F1, što ukazuje na *underfitting* pri većem broju susjeda. SVM se ponaša na isti način (sl. 4.12.), ali se s njim *underfitting* postiže povećanjem parametra C što smanjuje regularizaciju.



Sl. 4.11., Rezultati validacije hiperparametara za KNN i PCA



Sl. 4.12., Rezultati validacije hiperparametara za SVM i PCA

Tab. 4.7., Optimalni hiperparametri PCA predobrade.

	Hiperparam. klasifikatora	Varijanca	Mjera F1	Broj značajki
RF	Stabla: 150	0.9	0.837284	58
KNN	Susjedi: 3	0.9	0.827754	58
SVM	C: 0.01	0.92	0.675499	75

Zanimljivo je što se sva tri klasifikatora najbolje ponašaju s vrijednosti varijance ispod 0.95. Očekivanje je bilo da će se s povećanjem očuvane informacije povećati i mogućnost razlikovanja primjera, ali izgleda da je 0.9 dobar balans između očuvanja informacija i broja značajki, odnosno dimenzionalnosti problema.

SVM s linearnim kernelom nije se uspio najbolje istrenirati i postigao je mjeru F1 od samo 0.67. Ovo ukazuje na linearnu nerazdvojivost primjera pri obradi osnovnom analizom komponenti. Međutim, KNN i RF dobro razdvajaju podatke jer su sposobni za nelinearne granice odluke.

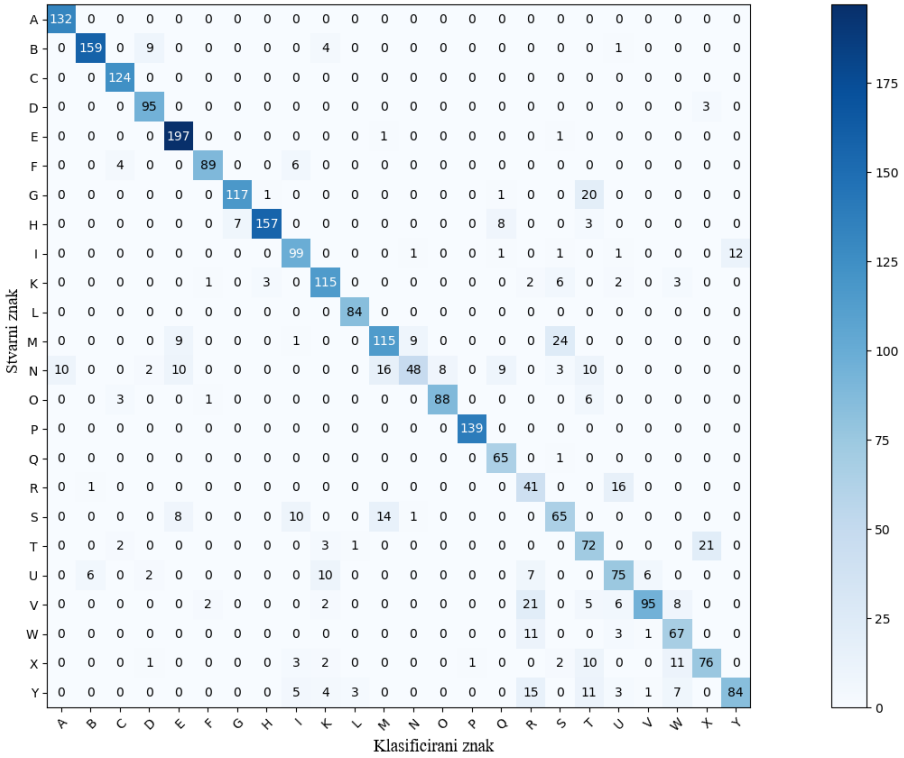
PCA nije algoritam izdvajanja značajki u smislu da će se njegovom provedbom dobiti kvalitetnija informacija kao kod lokalnih binarnih uzoraka ili detekcije rubova. On je algoritam smanjenja dimenzionalnosti koji komprimira podatke na način da prikaže istu količinu informacije manjim brojem značajki. Upravo broj osnovnih komponenti koje su generirane (tab. 4.7.) govori koliki je dio originalnih slika samo buka. Od 784 značajke, 58 značajki dovoljno je da klasifikator postigne dobru moć razlikovanja primjera. Time je eliminirana buka, ali očuvana inherentna informacija podataka.

Prema tab. 4.8. vidljivo je da su rezultati testiranja slični kao oni validacije. RF i KNN dovoljno dobro razlikuju još neviđene primjere i PCA je dobra opcija ako se ne želi gubiti vrijeme na proučavanju sofisticiranijih metoda izdvajanja značajki kao LBP. SVM s linearnim kernelom ne može dobro generalizirati na neviđene primjere kada se koristi samo PCA. Svejedno, za bolju analizu, treba pogledati matrice zabune klasifikatora za više informacija o vrsti poteškoća.

Tab. 4.8., Rezultati testiranja klasifikatora s optimalnim hiperparametrima.

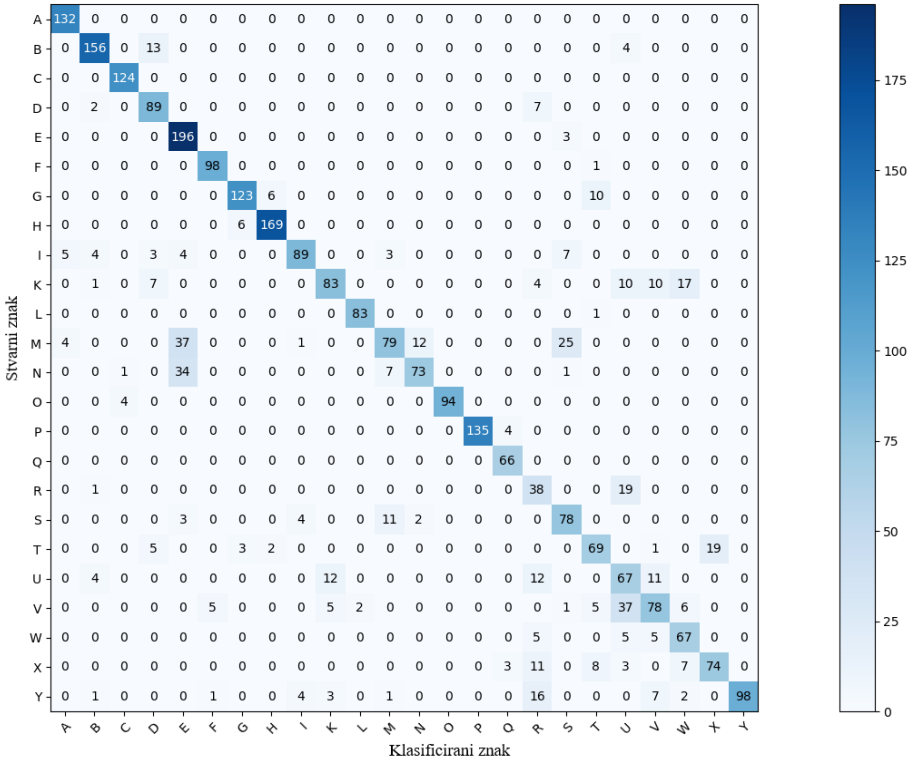
	Mjera F1	AUC	ACC
RF	0.829149	0.910860	0.829149
KNN	0.822176	0.907222	0.822176
SVM	0.561366	0.771147	0.561366

Prema slici 4.13., RF ima najviše problema s prepoznavanjem znaka N. Osim toga, dosta puta krivo identificira znakove kao R.

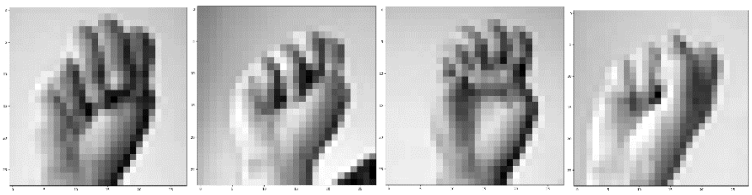


Sl. 4.13., Matrica zabune RF klasifikatora za PCA

Matrica zabune na slici 4.14. slična je onoj na sl. 4.13. i vidi se da KNN također ima poteškoća s krivom klasifikacijom znakova kao znak R. Također se znakovi M i N često klasificiraju kao E i S što ima smisla jer im je oblik skoro isti (sl. 4.15.), a time vjerojatno i vrijednosti intenziteta slike. Znak V je dosta puta krivo klasificiran kao U što se pojavilo i u lokalnim binarnim uzorcima, ali naravno ni blizu koliko ovdje.

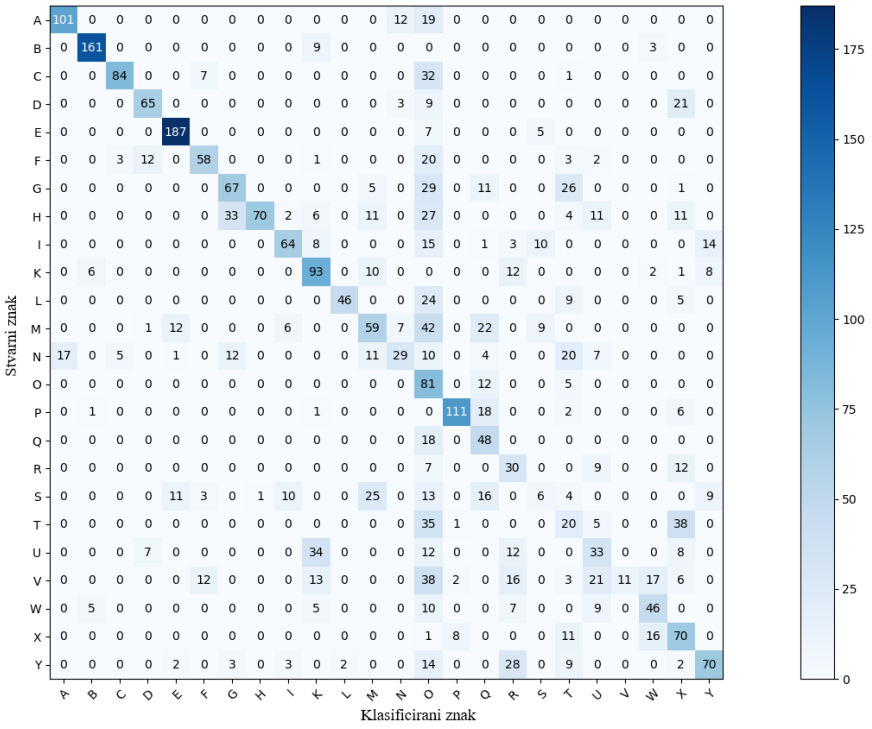


Sl. 4.14., Matrica zabune klasifikatora KNN za PCA



Sl. 4.15., Znakovi M, N, E i S

Konačno, sl. 4.16. prikazuje matricu zabune SVM klasifikatora gdje se vidi da on stvarno nije dobar kandidat za klasifikaciju koristeći samo PCA. Izuzetno puno primjera krivo klasificira kao znak O, dok znakove koji su slični ostalim znakovima kao M, N, S i V skoro uvijek klasificra kao neki drugi znak.



Sl. 4.16., Matrica zabune SVM klasifikatora za PCA

Sve u svemu, PCA je metoda kojom se smanjio broj značajki, a svejedno je ostavila dovoljno informacija kako bi klasifikatori s nelinearnim granicama odluke razlikovali primjere. Onaj tko ne želi potrošiti previše vremena na izdvajanje značajki može stati ovdje. SVM s linearnim kernelom nije uspio zadovoljavajuće razdvojiti podatke i ne bi se trebao koristiti u tom obliku. Poboljšanje bi se možda vidjelo primjenom polinomnog ili Gaussovog kernela koji imaju nelinearne granice odluke.

4.3.4. Kombinacija lokalnih binarnih uzoraka i detekcije rubova

Lokalni binarni uzorci samostalno su postigli odličan rezultat, no kako je rečeno u tom poglavlju, bilo je nekih problemima s razlikovanjem primjera gdje bi detekcija rubova mogla pomoći. U ovom potpoglavlju će se kombinirati histogrami lokalnih binarnih uzoraka i Canny detekcije rubova s najboljim parametrima u pokušaju rješenja tog problema.

Za klasifikator koristit će se šuma odluke jer se pokazala najboljom na primjerima za testiranje u oba slučaja. Za lokalne binarne uzorke i Canny detekciju rubova koristit će se najbolji hiperparametri (tab. 4.9.) za šumu odluke.

Tab. 4.9., Kombinirani, optimalni hiperparametri

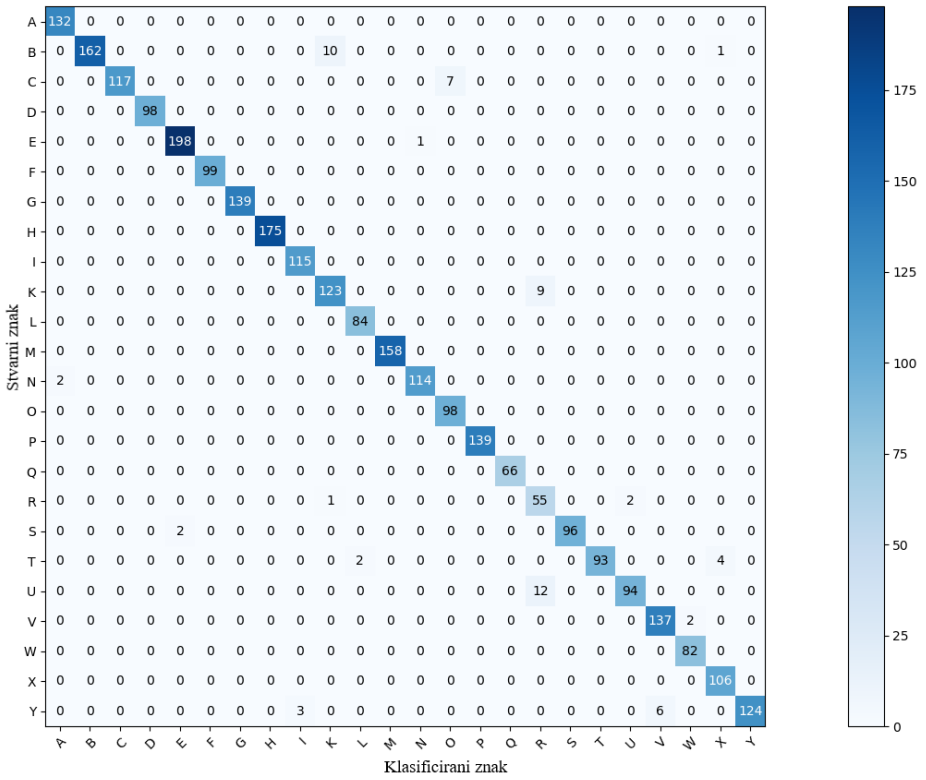
Klasifikator	Broj stabala	Canny *maxVal*	Canny *minVal*	LBP metoda	LBP točke	LBP radijus
Šuma odluke	150	140	70	„default“	12	5

Značajke se tvore tako da se iz sirovih značajki neovisno izračuna histogram lokalnih binarnih uzoraka i histogram Canny detektiranih rubova. Zatim se ta dva histograma ulančaju čime se dobije vektor od 4151 značajke. Nad skupom podataka s takvim značajkama provodi se testiranje. Usporedbu testiranja sa samim lokalnim binarnim uzorcima prikazuje tab. 4.10. Prema njoj, dodavanje Canny značajki nije značajno poboljšalo moć klasifikatora.

Tab. 4.10., Usporedba performanse RF klasifikatora za razne metode izdvajanja značajki

	Mjera F1	AUC	ACC
LBP	0.978382	0.988721	0.978382
LBP + Canny	0.977684	0.988357	0.977684

Usporedbom matrice zabune na sl. 4.17. s onom na sl. 4.4. vidimo da dodavanje značajki koje ističu rubove nisu imale utjecaj na performansu klasifikatora. Moguć razlog ovome je što se previše informacija izgubi pretvaranjem rubova slike u histograme ili što lokalni binarni uzorci u sebi već sadrže potrebne informacije o rubovima ruke.



Sl. 4.17., Matrica zabune RF klasifikatora s kombiniranim značajkama

U zaključku, nije vrijedno dodavati Canny značajke na ovaj način. Najbolje je držati se čistih lokalnih binarnih uzoraka jer daju i više nego zadovoljavajuće rezultate.

4.4. Osvrt i mogućnosti poboljšanja

Lokalni binarni uzorci pokazali su se kao odlična metoda izdvajanja značajki za ovaj problem. Svi klasifikatori, pa čak i SVM s linearnim kernelom, uspjeli su se vrlo dobro istrenirati na podacima i postigli su više nego zadovoljavajuće rezultate što se očituje mjerom F1 od 0.97 i više. SVM s linearnim kernelom se uspio vrlo dobro istrenirati što ukazuje na to da se obradom podataka LBP metodom postigla linearna razdvojivost klasa.

Eksperiment s Canny detekcijom rubova nije postigao najbolje rezultate, ali je svejedno bio zanimljiv jer je pokazao koliko informacija u podacima se može prikazati rubovima. Nažalost nije dovoljno dobra metoda sama od sebe jer su rezultati samo malo bolji od nasumičnog pogađanja. Osnovna analiza komponenti se također pokazala kao dobra metoda izdvajanja značajki za ovaj problem jer je značajno smanjila duljinu vektora značajki, a time postigla mjeru F1 veću od 0.8 za RF i KNN

klasifikatore. Međutim, SVM se nije uspio zadovoljavajuće istrenirati, vjerojatno jer su podaci obrađeni PCA algoritmom linearno nerazdvojivi. Da se pretpostaviti da će PCA raditi ovako dobro s ovim podacima za svaki klasifikator s mogućnosti nelinearne granice odluke.

Skupljanje više podataka za međusobno slične znakove moglo bi poboljšati moć razlikovanja s LBP algoritmom kao i dodatak značajki sofisticiranijeg algoritma detekcije rubova ili zatvorenih petlji. S tim na umu, moguće je da se takvo istraživanje i ne isplati jer su rezultati i više nego dobri ovakvi kakvi jesu, a poboljšanje bi bilo relativno malo. Još jedna mogućnost poboljšanja bila bi implementacija neuronske mreže umjesto metoda klasičnog strojnog učenja koje su implementirane u ovom radu.

5. ZAKLJUČAK

Ovim radom predstavljen je potpuni postupak stvaranja modela strojnog učenja za problem višeklasne klasifikacije za skup podataka koji se sastoji od slika znakova američkog znakovnog jezika. Teoretski su obrađene korištene metode predobrade slika, izdvajanja značajki, klasifikacije i vrednovanja u redoslijedu kojim se provode u eksperimentalnom dijelu. U predobradi slika opisani su postupci koji su već primijenjeni na skupu podataka *Sign Language MNIST* kojega se koristi u eksperimentalnom dijelu rada. Prije izdvajanja značajki pojašnjena je potreba za tim postupkom kao i pojam teksture. Zatim su opisane značajnije osobine lokalnih binarnih uzoraka, Canny detekcije rubova i osnovne analize komponenti. Ovi algoritmi implementirani su pomoću *Scikit-image* i *OpenCV* biblioteka. Od klasifikatora korištena je šuma odluke, stroj s potpornim vektorima i k-najbližih susjeda. Njihov način rada pojašnjen je u teorijskom dijelu, a implementiran pomoću *Scikit-learn* biblioteke. Za vrednovanje i podešavanje hiperparametara korištena je stratificirana metoda izdvajanja s mjerom F1 kao mjerilom ostvarenih performansi. U testiranju modela strojnog učenja s optimalnim hiperparametrima također je korištena točnost i površina ispod ROC krivulje. Svi ovi postupci opisani su u teorijskom dijelu rada, a implementirani pomoću *Scikit-learn* biblioteke.

Eksperimentalno su postignuti najbolji rezultati sa šumom odluke i lokalnim binarnim uzorcima gdje je vrijednost mjere F1 na skupu za testiranje bila 0.978, a površina ispod ROC krivulje 0.988. Šuma odluke i osnovna analiza komponenti zajedno postižu mjeru F1 0.829 i površinu ispod ROC krivulje 0.911, ali osnovna analiza komponenti ne izdvaja značajke tako da ih čini linearno razdvojivima, što je vidljivo slabim performansama SVM-a s linearnim kernelom. Eksperiment s Canny detekcijom rubova pokazao se nedovoljno dobrim za stvarnu primjenu, ali je ukazao na važnost rubova u raspoznavanju znakovnog jezika. S ciljem poboljšanja performansi lokalnih binarnih uzoraka i šume odluke, tim su značajkama ulančavanjem dodane značajke Canny detekcije rubova, što se pokazalo neuspješnim. Vjerojatno zbog toga što je i više nego dovoljno informacija o rubovima već prikazano lokalnim binarnim uzorcima.

Iz eksperimenta se zaključuje da šuma odluke zajedno s lokalnim binarnim uzorcima daje model strojnog učenja visokih performansi za raspoznavanje znakova znakovnog jezika. U budućem radu, poboljšanje bi se moglo ostvariti sofisticiranijom metodom detekcije rubova i zatvorenih petlji u slikama, skupljanjem dodatnih primjera za treniranje ili implementacijom rješenja pomoću neuronskih mreža.

LITERATURA

[1] D. Rakowski, „Fingerspelling ASL Type Font“, lifeprint.com, dostupno na: https://lifeprint.com/asl101/topics/wallpaper1.htm [28.08.2019.]

[2] „Primjer jednoručne abecede hrvatskog znakovnog jezika“, Udruga osoba oštećena sluha, dostupno na: http://uoosbbz.hr/images/uploads/hrvatska-jednorucna-abeceda.jpg [28.08.2019.]

[3] S. Joudaki, D. bin Mohamad, T. Saba, A. Rehman, M. Al-Rodhaan, A. Al-Dhelaan, “Vision-Based Sign Language Classification: A Directional Review”, IETE Technical Review, br. 5, sv. 31, str. 383-391, Listopad 2014

[4] „Sign Language MNIST: Drop-In Replacement for MNIST for Hand Gesture Recognition Tasks“, Kaggle.com, dostupno na: https://www.kaggle.com/datamunge/sign-language-mnist [28.08.2019.]

[5] M. Hruz, J. Trojanova, M. Železny, “Local Binary Pattern based features for Sign Language Recognition, Pattern Recognition and Image Analysis”, Springer-Verlag, br. 3, sv. 21, str. 398-401, 2011

[6] R. Kurdyumov, P. Ho, J. Ng, “Sign Language Classification Using Webcam Images”, Prosinac 2011

[7] L. Rioux-Maldague, P. Giguere, “Sign Language Fingerspelling Classification from Depth and Color Images using a Deep Belief Network”, Computer and Robot Vision, str. 6-9, 2014

[8] C. M. Bishop, “Pattern recognition and machine learning”, Springer, Singapore, 2006

[9] A. Ng, „Machine Learning“, coursera.org, dostupno na: https://www.coursera.org/learn/machine-learning [28.08.2019.]

[10] S. Feffer, „It's all about the features“, reality.ai, 2017, dostupno na: https://reality.ai/it-is-all-about-the-features [28.08.2019.]

[11] T. Mäenpää, “The Local Binary Pattern Approach To Texture Analysis – Extensions And Applications”, University of Oulu, Oulu, 2004

[12] T. Ojala, M. Pietikäinen, D. Harwood (1994), “Performance evaluation of texture measures with classification based on Kullback discrimination of distributions”, IEEE, Proceedings of 12th International Conference on Pattern Recognition, sv. 1, str. 582-585., Jerusalem, Israel, 1994

[13] M. Pietikainen, „Local Binary Patterns“, 2010, dostupno na: http://www.scholarpedia.org/article/Local_Binary_Patterns [28.08.2019.]

[14] T. Ojala, M. Pietikainen and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns", IEEE Transactions on Pattern Analysis and Machine Intelligence, br. 7, sv. 24, str. 971-987, Srpanj 2002

[15] “Facial Keypoints Detection - Scientific Figure”, researchgate.net, dostupno na: https://www.researchgate.net/figure/Diagram-of-Rotation-Invariant-LBP_fig9_320441974 [27.08.2019]

[16] J. Canny, „A Computational Approach To Edge Detection“, IEEE Transactions on Pattern Analysis and Machine Intelligence, br. 6, sv. 8, Studeni 1986

[17] A. Mordvintsev, Abid K., “Canny Edge Detection”, opencv.org, dostupno na: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html [29.08.2019.]

[18] A. Geron, “Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems”, O’Reilly Media, Inc., 2017

[19] L. Breiman, “Bagging Predictors”, Machine Learning, Springer, br. 2, sv. 24, str. 123-140, Kolovoz 1996

[20] “Decision Trees”, scikit-learn.org, dostupno na: https://scikit-learn.org/stable/modules/tree.html [29.08.2019.]

[21] “sklearn.metrics.confusion_matrix”, scikit-learn.org, dostupno na: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html [29.08.2019.]

[22] J. Šnajder, B. D. Bašić, „Strojno učenje: Bilješke s predavanja“, FER ZG, 2015

[23] „Drawing ROC Curve“, docs.eyesopen.com, 2018, dostupno na: https://docs.eyesopen.com/toolkits/cookbook/python/plotting/roc.html [29.08.2019.]

[24] “Scikit-learn: Machine Learning in Python”, scikit-learn.org, dostupno na: https://scikit-learn.org [29.08.2019.]

[25] “local_binary_pattern”, scikit-image.org, dostupno na: https://scikit-image.org/docs/dev/api/skimage.feature.html#skimage.feature.local_binary_pattern [29.08.2019.]

[26] „OpenCV 4.1.1“, opencv.org, dostupno na: https://opencv.org/ [29.08.2019.]

[27] “NumPy”, numpy.org, dostupno na: https://www.numpy.org/ [29.08.2019.]

[28] “Pandas: Python Data Analysis Library”, pandas.pydata.org, dostupno na: https://pandas.pydata.org/ [29.08.2019.]

[29] “Matplotlib”, matplotlib.org, dostupno na: https://matplotlib.org/ [29.08.2019.]

SAŽETAK

Rad donosi teorijsku podlogu potrebnu za razumijevanje procesa implementacije modela strojnog učenja za raspoznavanje znakovnog jezika pomoću uslikane ruke. Daje se pregled cijelog postupka od predobrade slika i izlučivanja značajki do treniranja klasifikatora, vrednovanja i odabira optimalnih hiperparametara. Rješenje je implementirano pomoću *Scikit-learn*, *Scikit-image* i *OpenCV* Python biblioteka. Najbolji rezultati postignuti su šumom odluke i značajkama dobivenim algoritmom lokalnih binarnih uzoraka, ali i stroj s potpornim vektorima i k-najbližih susjeda postižu slične rezultate s takvim značajkama. Analizirane su i druge metode izdvajanja značajki kao Canny detekcija rubova i osnovna analiza komponenti sa spomenutim klasifikatorima, iako su dale lošije rezultate.

Ključne riječi: izdvajanje značajki, klasifikacija slika, predobrada slika, strojno učenje, znakovni jezik

ABSTRACT

Title: Character letters recognition by using machine learning methods

This work presents the theoretical background necessary for understanding the implementation of a sign language classifier. An overview of the entire process is given; from image preprocessing and feature extraction to classifier training and cross-validation for fine tuning the machine learning model. The experimental portion was implemented using Python libraries such as *Scikit-learn*, *Scikit-image* and *OpenCV*. The best results were achieved using a random forest classifier with features extracted using the local binary patterns algorithm, although support vector machine and K-nearest neighbor classifiers achieved similar results with such features. Other feature extraction algorithms such as Canny edge detection and principal component analysis were implemented with those classifiers as well, although with poorer results.

Keywords: feature extraction, image classification, image preprocessing, machine learning, sign language

ŽIVOTOPIS

David Takač rođen je 22.07.1997. u Beogradu. S tri godine doselio se s roditeljima u Osijek gdje je pohađao osnovnu i srednju školu (Elektrotehnička i prometna škola Osijek). Stekao je strukovnu kvalifikaciju elektrotehničara u srednjoj školi i potom upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek gdje je trenutno student treće godine.

David Takač

_____

PRILOZI

Na CD-u:

1. Završni rad “Raspoznavanje slova znakovnog jezika korištenjem postupaka strojnog učenja” u *.docx* formatu
2. Završni rad “Raspoznavanje slova znakovnog jezika korištenjem postupaka strojnog učenja” u *.pdf* formatu
3. Izvorni kod eksperimentalnog dijela rada