Procjena dobi analizom slika lica

Perić, Nikola

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1477

PROCJENA DOBI ANALIZOM SLIKA LICA

Nikola Perić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1477

PROCJENA DOBI ANALIZOM SLIKA LICA

Nikola Perić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1477

Pristupnik:	**Nikola Perić (0036543871)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	prof. dr. sc. Sonja Grgić

Zadatak: **Procjena dobi analizom slika lica**

Opis zadatka:

Ljudsko starenje je proces popraćen promjenama specifičnim za dob, a te promjene su zajedničke za većinu ljudi. Analizom slika i detekcijom specifičnih promjena može se procijeniti dob osobe. Cilj zadatka je razviti sustav za procjenu dobi osobe na temelju analize slika lica. U radu je potrebno dati pregled postojećih pristupa za procjenu dobi iz slika lica kao što su, na primjer, analiza glavnih komponenti, linearna diskriminantna analiza, metoda k-najbližih susjeda te metode utemeljene na primjeni dubokoga učenja. Odabrati metodu, detaljnije ju objasniti i predložiti model sustava koji može na temelju ulazne slike lica procijeniti dob osobe. Implementirati i testirati programsko rješenje sustava u programskom jeziku po izboru. Za ispitivanje rada sustava koristiti javno dostupne baze slika lica u kojima je raspoloživ podatak o dobi osobe. Analizirati performanse sustava pružajući uvid u njegovu učinkovitost u predviđanju dobi na temelju slika lica. Navesti prednosti i ograničenja implementiranog rješenja i predložiti moguća poboljšanja.

Rok za predaju rada: 14. lipnja 2024.

Sadržaj

1. Uvod

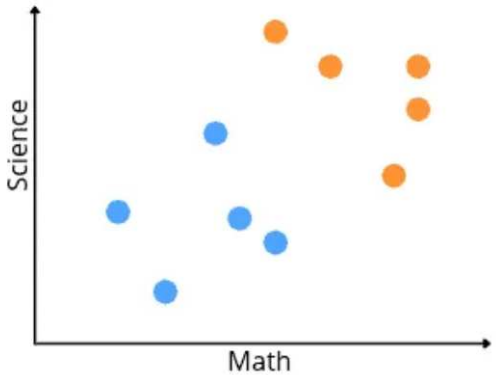
Dob osobe je vrlo korisna informacija u raznim područjima poput medicine, sigurnosti i marketinga. Starenjem dolazi do karakterističnih promjena na licu osobe kao što su bore, promjena teksture i gubitak elastičnosti kože, gubitak volumena oko usne, promjene oko očiju te brojne druge izmjene uvjetovane starenjem osobe. Prepoznavanjem tih karakteristika moguće je procijeniti dob osobe metodama strojnog učenja. U ovom radu dat će se pregled najčešćih pristupa i metoda za procjenu dobi osobe iz slike lice. To su analiza glavnih komponenti, linearna diskriminantna analiza, metoda k-najbližih susjeda, stroj s potpornim vektorima te model zasnovan na konvolucijskim neuronskim mrežama. Za svaku od navedenih metoda, objasnit će se glavne karakteristike, kako algoritam funkcionira te prednosti i nedostatci svake od njih. Implementirat će se programsko rješenje koje koristi analizu glavnih komponenti za smanjenje dimenzionalnosti podataka te stroj s potpornim vektorima za klasifikaciju tj. predviđanje dobi osobe. Korištenjem analize glavnih komponenti smanjuje se složenost modela i poboljšava efikasnost računanja, dok stroj s potpornim vektorima omogućava točnu klasifikaciju različitih dobnih skupina. Kao baza podataka za treniranje i testiranje modela, koristit će se UTKFace dataset. To je veliki skup podataka tj. slika lica osoba sa oznakama dobi, spola i etničke pripadnosti. Nakon implementacije, odabrani model će se ocijeniti pomoću matrice zabune i izvješća o klasifikaciji te će se argumentirati moguća poboljšanja algoritma.

2. Analiza postojećih postupaka za procjenu dobi

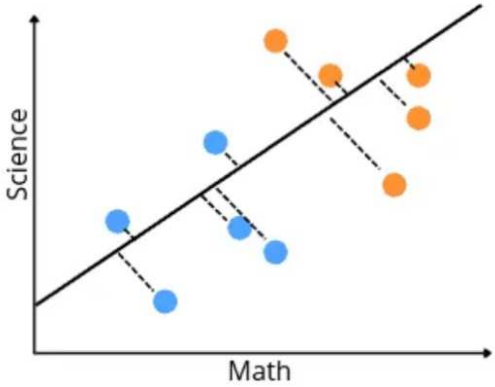
Ovo poglavlje dat će pregled nekih od najčešćih metoda za procjenu dobi. Metode poput analize glavnih komponenti i linearne diskriminantne analize služe za smanjenje dimenzionalnosti skupa podataka. S obzirom na to da su slike višedimenzionalni podaci, gdje svaki piksel predstavlja jednu dimenziju, smanjenje dimenzionalnosti omogućava brže izvođenje modela uz minimalan gubitak preciznosti. Metode kao što su konvolucijske neuronske mreže, stroj s potpornim vektorima te metoda k-najbližih susjeda služe za klasifikaciju tj. predviđanje klase kojoj neki podatak pripada – specifično, u ovom radu, to je dob osobe na slici.

2.1 Linearna diskriminantna analiza

Linearna diskriminantna analiza (engl. *linear discriminant analysis, LDA*) metoda je smanjenja dimenzionalnosti podataka najčešće korištena u algoritmima nadziranog strojnog učenja gdje je cilj klasifikacija podataka. Metoda proizlazi iz Fisherove linearne diskriminante koju je 1936. razvio sir Ronald Fisher, a 1948. C. R. Rao generalizirao da radi na više klasa. LDA pokušava pronaći pravac (ako su ulazni podaci u dvije dimenzije) ili hiperravninu (ako su podaci višedimenzionalni) koja najbolje razdvaja klase, i pritom pokušava smanjiti preklapanje unutar svake klase. Na primjer, na dvodimenzionalnom skupu podataka kao na slici (Sl. 2.1.1), LDA će pronaći pravac koji maksimizira udaljenost između srednjih vrijednosti (engl. *mean*) klasa i pritom minimizira varijancu unutar svake klase (Sl. 2.1.2).



Sl. 2.1.1 Ulazni skup podataka [4]

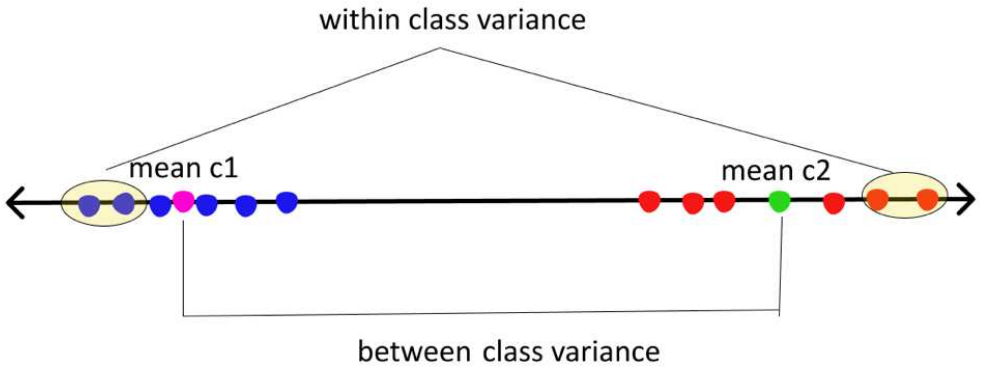


Sl. 2.1.2 Pravac koji razdvaja 2 klase skupa podataka [4]

Metoda preslikava prostor s n dimenzija u prostor s k dimenzija, tako da vrijedi $k \leq n - 1$, bez gubitka informacija o klasama podataka. LDA model ima nekoliko početnih pretpostavki koje je potrebno zadovoljiti kako bi algoritam pravilno funkcionirao. To su:

- Ulazni podaci imaju Gaussovu razdiobu (tj. grafički zapis podataka ima oblik zvona)
- Skup podataka je linearno razdvojiv tj. LDA može stvoriti granice odlučivanja (engl. *decision boundary*) koji razdvaja skup podataka prema klasama
- Svaka klasa ima istu kovarijacijsku matricu

LDA podatke preslikava u prostor manje dimenzionalnosti tako da računa varijancu između klasa (engl. *between-class variance*) tj. udaljenost između srednjih vrijednosti (engl. *mean*) klasa i varijancu unutar klase (engl. *within-class variance*) tj. razmak elemenata unutar jedne klase (Sl. 2.1.3)

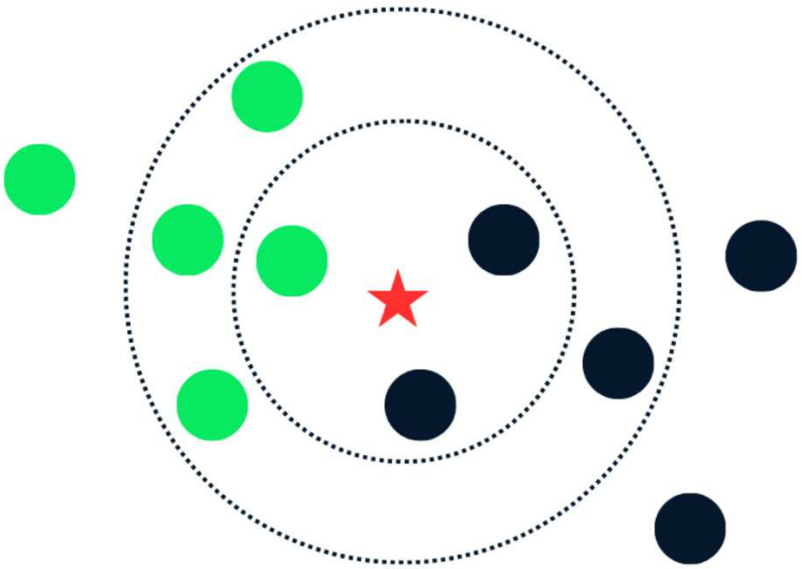


Sl. 2.1.3 Varijanca unutar i između klasa [2]

Linearna diskriminantna analiza jednostavna je za izvedbu i razumijevanje te je učinkovita kada je broj značajki (engl. *feature*) znatno veći od broja podataka tj. kada početni podaci imaju velik broj dimenzija kao što je slučaj u obradi teksta i prepoznavanju slika. S druge strane, LDA nailazi na probleme kada klase podataka imaju zajedničke ili slične srednje vrijednosti. Algoritam tada neće moći pravilno razdvojiti klase tj. neće moći stvoriti novu os koja bi linearno razdvojila klase. Osim toga, algoritam je osjetljiv na ekstreme u podacima koji mogu utjecati na srednje vrijednosti klasa i tako dovesti do lošijeg razdvajanja skupa podataka.

2.2 Metoda k-najbližih susjeda

Metoda k-najbližih susjeda (engl. *k-nearest neighbors, kNN*) jedan je od najjednostavnijih nadziranih modela strojnog učenja. Algoritam se primarno koristi za klasifikaciju, ali se uz blage modifikacije može primijeniti i na probleme regresije. Metodu su prvi razvili Evelyn Fix i Joseph Hodges 1951., a kasnije ju je proširio Thomas Cover. Algoritam kNN spada u obitelj lijenog učenja (engl. *lazy learner*) što znači da se model ne trenira, nego se samo koristi početni označeni skup podataka. Sve računanje odvija se tijekom predviđanja. Takvi algoritmi se još nazivaju i engl. *instance-based learning* algoritmi. Kako bi predvidio koja je klasa novog, nepoznatog podatka, algoritam traži k najbližih susjeda tog podatka. Postoji mnogo metrika udaljenosti za računanje najbližih susjeda, a najčešće su Euklidska udaljenost, Manhattan udaljenost, Minkowski udaljenost te Hammingova udaljenost. Kada pronađe susjede, algoritam gleda kojoj klasi pripada pojedinačni susjed te odabire najčešću klasu (kod problema regresije, algoritam bi uzeo prosjek vrijednosti najbližih susjeda). Na primjer, kako bi predvidio kojoj klasi pripada zvjezdica na slici (Sl. 2.2.1), algoritam gleda kojoj klasi pripada prvih k najbližih susjeda. U slučaju da je k = 3 (manji krug na slici), dva su crna i jedan svijetli element. Algoritam će zbog toga predvidjeti da novi podataka tj. zvjezdica pripada klasi 'crna'. Ako je k = 7 (veći krug na slici), algoritam će predvidjeti da novi podatak pripada klasi 'svijetla'.

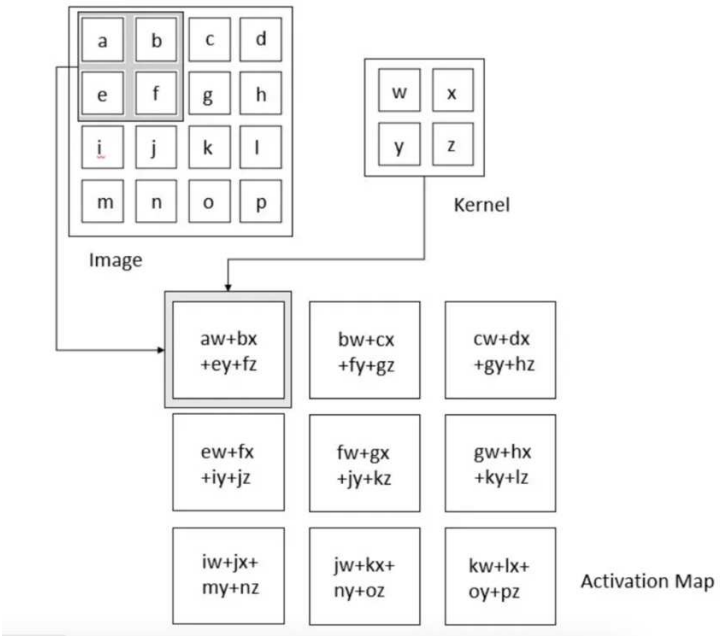


Sl. 2.2.1 Predviđanje klase podatka [6]

Kao što je vidljivo iz primjera, odabir vrijednosti k ima značajan utjecaj na performanse i rezultate algoritma. Manje vrijednosti k rezultiraju u višoj varijanci, ali manjoj pristranosti (engl. *bias*) algoritma, dok veće vrijednosti k uzrokuju manju varijancu, ali veću pristranost. Metoda k-najbližih susjeda jednostavna je za implementaciju te se pri dodavanju novih podataka za treniranje algoritam brzo prilagođava. Za razliku od većine algoritama strojnog učenja, kNN nema puno hiperparametara – potrebni su samo k i metrika za udaljenost koja se koristi. Međutim, algoritam se jako oslanja na memoriju za pohranu skupa podatka i zbog toga postiže lošije rezultate kada se koriste veliki skupovi podatka. kNN također pati od tzv. „prokletstva dimenzionalnosti“ (engl. *curse of dimensionality*) što znači da postiže loše rezultate kod podataka sa mnogo dimenzija. Male vrijednosti k sklone su prenaučenosti (engl. *overfitting*) algoritma, a velike vrijednosti k mogu podnaučiti algoritam (engl. *underfitting*).

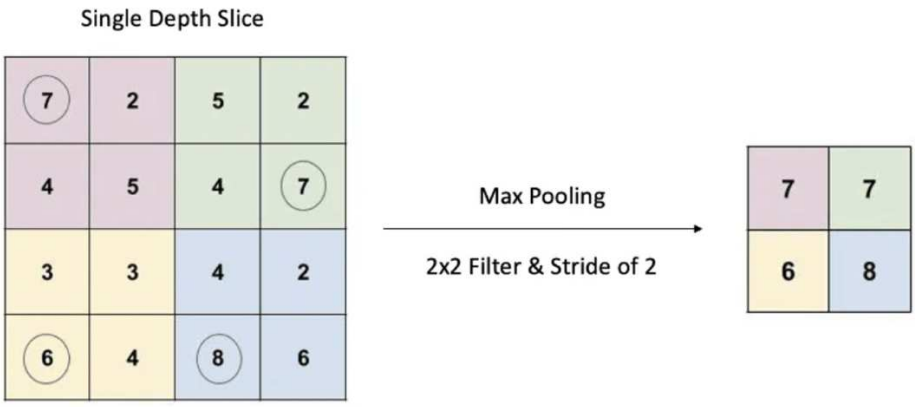
2.3 Konvolucijske neuronske mreže

Konvolucijska neuronska mreža (engl. *convolutional neural network, CNN*) tip je umjetne neuronske mreže koja se primarno koristi u klasifikaciji slika. Inspirirane su biološkim vidnim korteksom koji je odgovoran za obradu vizualnih informacija u životinjama. Glavna karakteristika algoritma je mogućnost da automatski uči važne značajke iz ulaznog skupa podataka. Za razliku od običnih neuronskih mreža, CNN ima barem jedan tzv. konvolucijski sloj. Kod običnih neuronskih mreža, svaki ulaz povezan je sa svakim neuronom u barem jednom skrivenom sloju mreže. Problem nastaje kod analize slika, jer bi tada svaki piksel bio jedan ulaz u neuronsku mrežu, što vodi do kombinatorne eksplozije Na primjer, za sliku rezolucije 500 x 500 piksela, mreža bi imala 250,000 ulaza. Konvolucijski sloj i sloj sažimanja služe kako bi se smanjio broj ulaza u potpuno povezanoj (engl. *fully connected*) neuronskoj mreži i pritom zadržalo što više važnih informacija o ulaznoj slici. CNN tipično ima 3 sloja: konvolucijski sloj (engl. *convolutional layer*), sloj sažimanja (engl. *pooling layer*) i potpuno povezani sloj (engl. *fully-connected layer*). Konvolucijski sloj sadrži filter matricu tj. jezgrenu funkciju (engl. *kernel*) i radi skalarni umnožak između jezgrene funkcije i ograničenog dijela slike (Sl. 2.3.1). Jezgrena funkcija "klizi" preko cijele slike i skalarnim umnoškom s elementima slike stvara tzv. aktivacijsku matricu (engl. *activation map*).



Sl. 2.3.1 Konvolucijski sloj mreže [10]

Sloj sažimanja služi za smanjenje dimenzionalnosti podataka i pritom zadržava najvažnije informacije o njima. To postiže primjenom funkcija sažimanja, od kojih su najčešće engl. *max pooling* koja vraća maksimalnu vrijednost, engl. *mean pooling* koja vraća prosječnu vrijednost i engl. *weighted mean pooling* koja vraća težinski prosjek. Na slici (Sl. 2.3.2) vidljivo je kako se pomoću engl. *max pooling* funkcije, koja iz svakog 2 x 2 bloka uzima element najveće vrijednosti, postiže smanjenje dimenzionalnosti podataka.



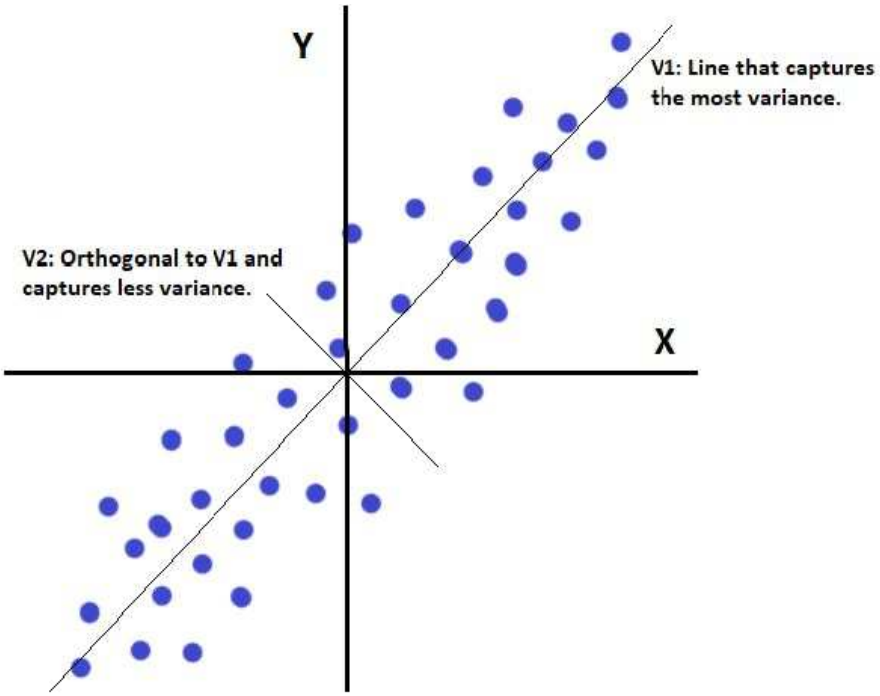
Sl. 2.3.2 Primjena max pooling funkcije [10]

Zadnji sloj konvolucijske neuronske mreže je potpuno povezani sloj (engl. *fully-connected layer*) koji služi za klasifikaciju/regresiju pomoću ulaznih podataka dobivenih iz sloja sažimanja. Svaki neuron u sloju povezan je sa svakim neuronom u prošlom i sljedećem sloju. Potpuno povezani sloj funkcionalno je identičan skrivenim slojevima običnih neuronskih mreža. Glavna prednost konvolucijskih neuronskih mreža je njihova sposobnost da automatski uče relevantne značajke iz ulaznih podataka. Otporni su na male promjene na slikama poput pozicije objekta unutar slike te, ako je skup podataka za učenje prikladne veličine, dobro generaliziraju. No, ako je skup za učenje malen ili pun šuma, algoritam je sklon prenaučenosti, tj. slabo generalizira. Osim toga, treniranje CNN-a računalno je skupo i zahtjeva velike količine podataka i računalnih resursa.

3. Segmentacija i klasifikacija

3.1 Segmentacija metodom analize glavnih komponenti

Analiza glavnih komponenti (engl. *principal component analysis, PCA*) metoda je smanjenja broja dimenzija tj. varijabli skupa podataka koja pritom pokušava sačuvati što više korisnih informacija o podacima. Algoritam je izumio Karl Pearson 1901. godine, a kasnije ju je neovisno razvio i imenovao Harold Hotelling. Glavne komponente su nove varijable stvorene kao linearne kombinacije početnih varijabli, konstruirane tako da budu međusobno nekorelirane tj. nezavisne. Većina važnih informacija iz originalnih podataka zapisana je u prvih nekoliko glavnih komponenti što omogućuje smanjenje dimenzionalnosti podataka. Matematički gledano, glavne komponente predstavljaju smjerove u podacima koji zapisuju najveću količinu varijance. Na primjer, na slici (Sl. 3.1.1) podaci su zapisani na grafu s 2 osi – x i y. PCA pronalazi os V1 koja hvata najveću količinu varijance u podacima i nakon toga pronalazi os V2 koja je ortogonalna tj. nekorelirana s osi V1 i zapisuje drugu najveću količinu varijance u podacima. Vidljivo je da je velika većina varijance zapisana pomoću osi V1 te odbacivanjem osi V2 moguće je smanjiti dimenzionalnost podataka i pritom zadržati veliku većinu korisnih informacija o podacima.

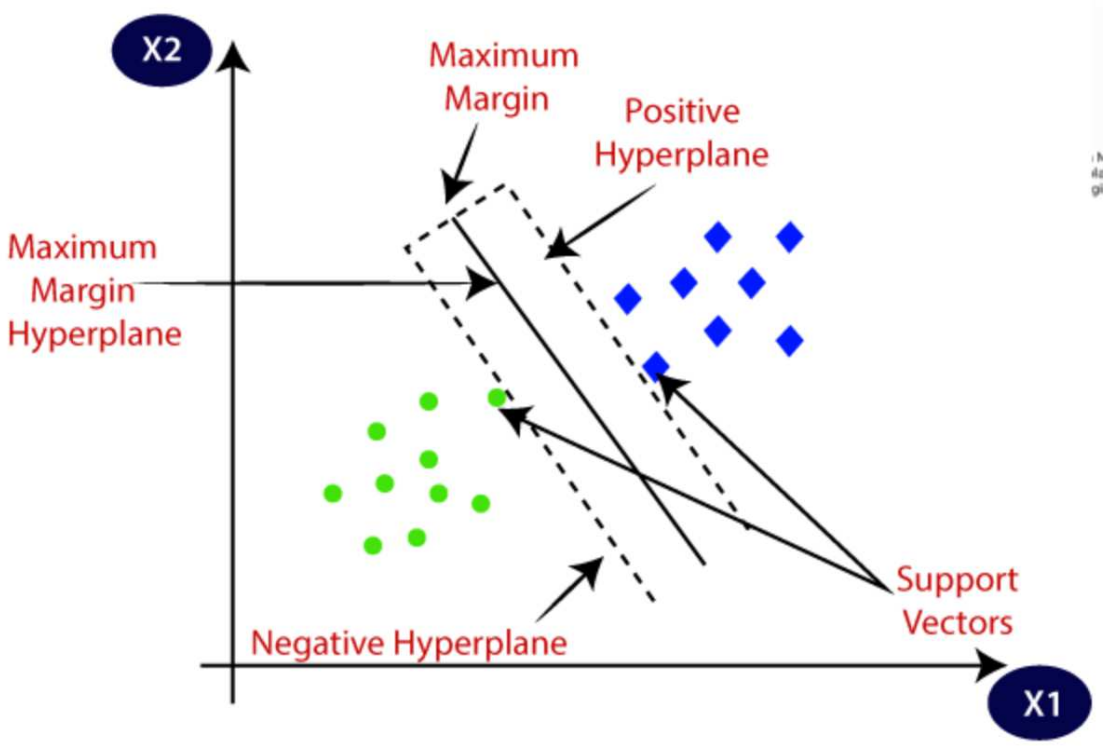


Sl. 3.1.1 Primjer analize glavnih komponenti [15]

PCA može stvoriti jednak broj varijabli kao što ih ima u originalnom skupu podataka. No, kako bi smanjili broj dimenzija podataka, potrebno je odbaciti neke od njih. Glavna prednost analize glavnih komponenti je u tome što nam omogućuje da zadržavanjem samo prvih nekoliko varijabli sačuvamo većinu važnih informacija o skupu podataka. Algoritam PCA prvo računa matricu kovarijance između varijabli skupa podataka. Kovarijanca je mjera korelacije između dvije varijable. Ako je kovarijanca pozitivna, to znači da se vrijednosti varijabli smanjuju tj. rastu zajedno (kada raste jedna, raste i druga). Ako je kovarijanca negativa, kada vrijednost jedne varijable raste, vrijednost druge se smanjuje – i suprotno. Ako je vrijednost kovarijance dvije varijable 0, to znači da su nekorelirane tj. jedna nema utjecaja na drugu. Varijable koje su jako korelirane stvaraju redundantne informacije. Računanjem matrice kovarijance, algoritam može prepoznati odnose između varijabli tj. koje varijable su međusobno korelirane, a koje nisu. Algoritam zatim računa svojstvene vektore (engl. *eigenvector*) i svojstvene vrijednosti (engl. *eigenvalue*) matrice kovarijance. Svojstveni vektori predstavljaju smjerove osi s najvećom varijancom tj. novonastale glavne komponente, dok su svojstvene vrijednosti koeficijenti tih vektora – oni označuju koliko je ukupne varijance pohranjeno u svakoj od glavnih komponenti. Algoritam zatim odabire koje od svojstvenih vektora će zadržati kao glavne komponente, a koje će odbaciti. U ovom koraku odvija se smanjenje dimenzionalnosti jer od n početnih vektora, algoritam ih zadržava p, gdje je $p \leq n$. U zadnjem koraku algoritma, podaci se zapisuju pomoću odabranih glavnih komponenti.

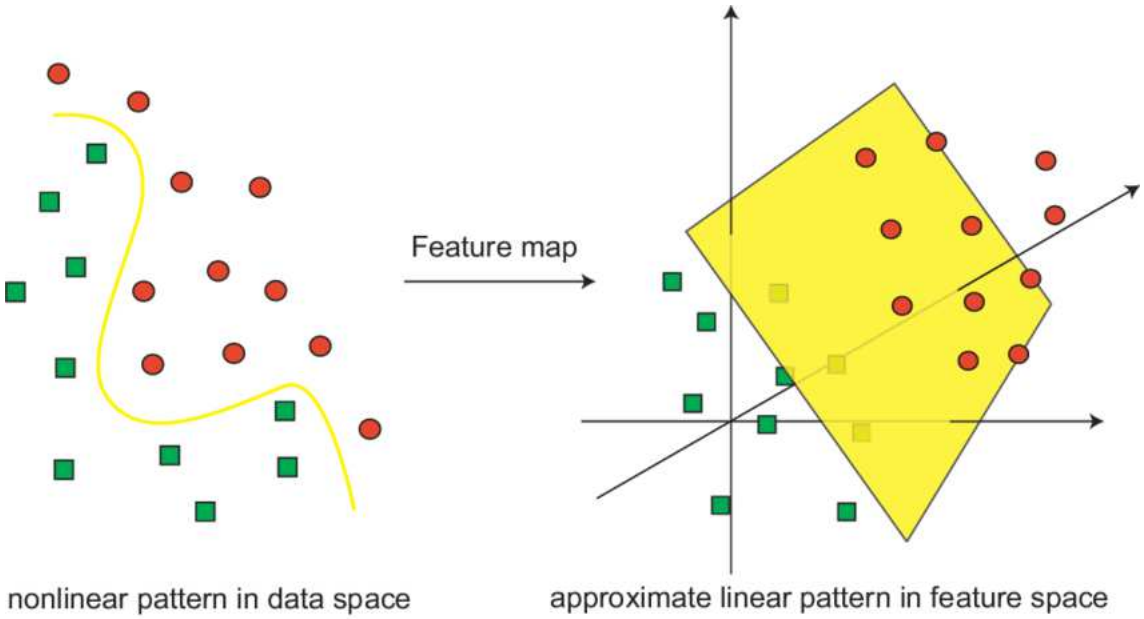
3.2 Stroj s potpornim vektorima

Stroj s potpornim vektorima (engl. *support vector machine, SVM*) nadzirani je algoritam strojnog učenja koji klasificira podatke tako što traži najbolju liniju tj. hiperravninu (engl. *hyperplane*) koja maksimizira udaljenost između svake klase u n-dimenzionalnom prostoru. Algoritam je, zajedno s kolegama, razvio Vladimir N. Vapnik 1964. i naknadno modificirao 1994. SVM pronalazi ravninu razdvajanja (engl. *decision boundary*) koja linearno dijeli dvije klase podataka. S obzirom na to da je moguće stvoriti više ravnina razdvajanja, algoritam traži ravninu razdvajanja koja maksimizira margine između klasa podataka, što dovodi do dobre generalizacije. Na slici (Sl. 3.2.1) prikazan je primjer linearno razdvojivog skupa podataka podijeljenih u dvije klase i ravnina koja razdvaja te dvije klase s maksimalnom veličinom margine između njih.



Sl. 3.2.1 SVM razdioba linearno razdvojivog skupa podataka [18]

Postoje dva tipa margina: čvrsta (engl. *hard margin*) i meka (engl. *soft margin*) margina. Kod čvrste margine, podaci su savršeno izvan rubova margine tj. „sa strane ulice“ kao na slici (Sl. 3.2.1). Meka margina je fleksibilnija i dozvoljava da dio podataka bude pogrešno klasificiran. Veličina margine može se podesiti pomoću hiperparametra C – veći C označava manju marginu (manje pogrešne klasifikacije podataka), a manji C označava veću marginu (više pogrešne klasifikacije). Algoritam se dijeli u 2 vrste: linearni i nelinearni SVM. Linearni SVM koristi se na linearno razdvojivim podacima tj. na podacima koje nije potrebno transformirati kako bi se razdvojili u različite klase. Primjer takvog modela prikazan je na slici (Sl. 3.2.1) na kojoj su se dvije klase podataka savršeno razdvojile uz pomoć jednog pravca. S obzirom na to da u većini situacija podatke nije moguće linearno razdvojiti, stvara se potreba za korištenjem nelinearnog SVM koji podatke transformira u prostor veće dimenzionalnosti pomoću jezgrenih funkcija (engl. *kernel function*). Primjer takvog modela prikazan je na slici (Sl. 3.2.2) na kojoj je vidljivo da početni skup podataka nije linearno razdvojiv zbog čega algoritam podatke transformira iz prostora s dvije dimenzije u trodimenzionalni prostor u kojem se dvije klase podataka mogu linearno razdvojiti.



Sl. 3.2.2 Primjer nelinearnog SVM modela [21]

Transformacija podataka u višedimenzionalni prostor računalno je skupo i povećava složenost modela. Zbog toga se podaci zapravo ne transformiraju, nego se koristi tzv. engl. „*kernel trick*“. Jezgrene funkcije ponašaju se kao modificirani skalarni produkt – primaju ulazne podatke u originalnim prostoru, te vraćaju skalarni produkt transformiranih vektora podataka u višedimenzionalnom prostoru. Neke od najpoznatijih jezgrenih funkcija su: engl. *polynomial kernel*, engl. *radial basis function kernel*, engl. *sigmoid kernel*. SVM su korisni kod podataka s velikim brojem dimenzija i kada klase podataka nisu linearno razdvojive. Zbog mogućnosti podešavanja margine preko parametra C, mogu ublažiti probleme prenaučenosti algoritma. S druge strane, SVM ima probleme kod podataka s velikom količinom šuma ili u situacijama gdje se klase podatka preklapaju. Osim toga, nisu praktični za velike skupove podataka zbog dugačkog perioda treniranja algoritma.

4. Implementacija programskog rješenja

Kako bi se procijenila dob osobe sa slike, koristit će se 2 metode – PCA i SVM. Primjenom PCA smanjit će se broj dimenzija skupa podataka i tako smanjiti računalna cijena algoritma, a SVM vršit će samu procjenu dobi. Kao baza podataka za treniranje i testiranje algoritma, koristit će se UTKFace dataset. To je skup podataka koji sadrži više od 20,000 slika lica s oznakama dobi, spola i etničke pripadnosti osobe. Dob se kreće od 0 do 116 godina, a većina osoba je između 20 i 40 godina. Prosječna dob je 33, a median je 29. Osim toga, u skupu ima značajan broj slika osoba od 1 godine. Primjer jedne osobe prikazan je na slici (Sl. 4.1), zajedno s dobnom skupinom tj. klasom kojoj pripadaju.

Sl. 4.1 Primjer osobe iz UTKFace baze podatka

U ovom radu neće se koristiti čitav skup podataka, već samo podskup od 9780 centriranih slika. Dob osobe podijeljena je u 8 klasa: 0-3, 4-10, 11-20, 21-30, 31-40, 41-50, 51-60 te 61+ godina. S obzirom na to da UTKFace nije balansiran skup, nad klasama 41-50 godina i 51-60 godina provedene su augmentacije kako bi se povećao skup za treniranje modela. To dovodi do bolje generalizacije i time boljih performansi sustava. Augmentacija se postiže rotacijom slike za nasumičan kut između -10 i 10 stupnjeva. Tako se klasa 41-50 godina povećava za 70 %, a

klasa 51-60 za 30 %. Osim toga, klasa 0-3 godina, koja inače ima 1871 slika, smanjena je da sadrži samo 1100 slika kako bi skup podataka bio uravnoteženiji. Nakon tih promjena, ukupan broj slika za treniranje i testiranje modela iznosi 9800 slika. Taj skup podataka dijeli se na skup za treniranje veličine 80 % originalnog skupa, i skup za testiranje koji iznosi 20 % originalnog skupa. Na slici (Sl. 4.2) prikazano je kako se početni skup dijeli na dva skupa, te se zatim računa prvih 100 glavnih komponenti skupa za treniranje pomoću fit() metode. Nakon toga se pomoću metode transform() cijeli originalni skup podataka (X_train i X_test) projicira na glavne komponente kako bi se broj dimenzija smanjio s 224 x 224 tj. 50,176 na samo 100. Nakon provođenja PCA, 91.86 % ukupne varijance u podacima zapisano je preko samo 100 dimenzija.

```
X_train, X_test, y_train, y_test = train_test_split(images,
                                                    labels,
                                                    test_size=0.2,
                                                    random_state=42)

pca = PCA(n_components=100).fit(X_train)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

Sl. 4.2 Primjena PCA na skupu podataka za treniranje

Algoritam zatim stvara SVM objekt pomoću metode SVC(). Odabrana jezgrena funkcija je "rbf" tj. engl. *radial-basis kernel*. Na slici (Sl. 4.3) odabrana je i vrijednost "balanced" za parametar *class_weight*. To je korisno za skupove podataka koji nisu balansirani tj. nemaju uniformnu razdiobu podatka po klasama. Nakon inicijalizacije objekta *svm*, potrebno ga je istrenirati na skupu podataka za treniranje X_train_pca. To se postiže metodom fit() koja kao parametre prima skup za treniranje i odgovarajuće labele tj. klase skupa.

```
svm = SVC(kernel='rbf', class_weight='balanced')
svm.fit(X_train_pca, y_train)
```

Sl. 4.3 Stvaranje i učenje SVM na skupu za treniranje

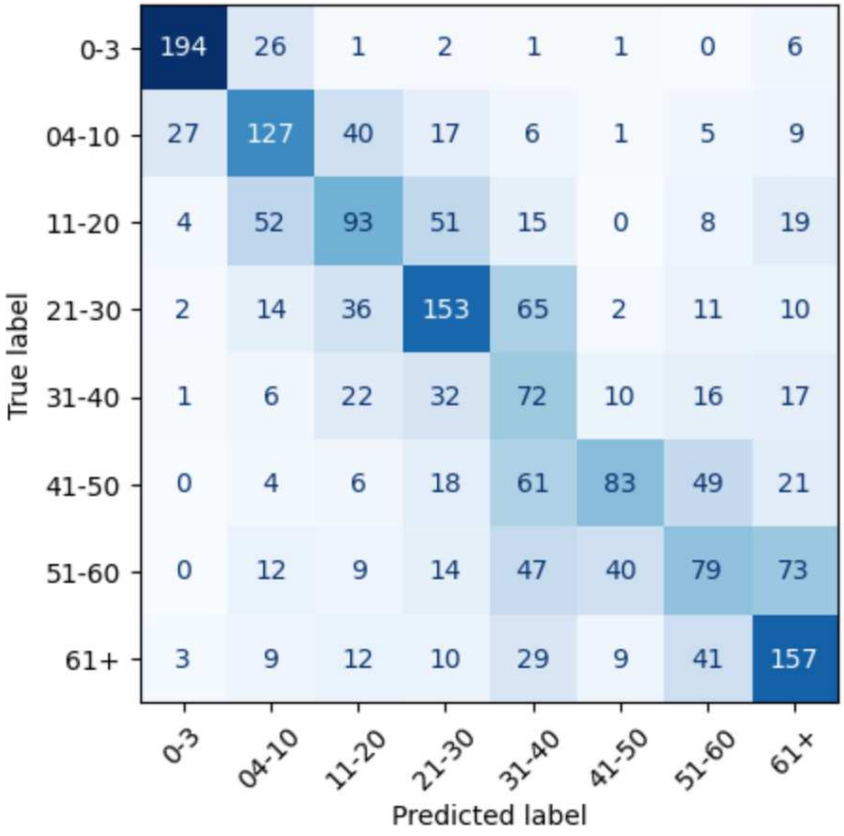
5. Rezultati programskog rješenja

Uz pomoć skupa za testiranje provjerava se točnost implementiranog rješenja koja iznosi 48.88 %. No, sama točnost ne daje detaljne uvide u rezultate algoritma pa se zbog toga koriste još neke mjere kao što su izvješće o klasifikaciji i matrica zabune. Na slici (Sl. 5.1) prikazano je izvješće o klasifikaciji na kojoj su vidljive vrijednosti metrika preciznosti, odaziva (engl. *recall*), f1-mjere i podrške (engl. *support*) za svaku klasu. Preciznost označava koliko je često algoritam predvidio neku klasu i pogodio. Na primjer, algoritam ima 0.84 preciznost za klasu 0-3 godina što znači da u 84 % slučajeva kada je algoritam odlučio da slika pripada klasi 0-3, bio je u pravu. Vidljivo je da je algoritam najviše problema imao s klasama 31-40 i 51-60 godina, dok je za klasu 0-3 postizao odlične rezultate. Odaziv označava koliki udio slika neke klase je algoritam prepoznao. Na primjer, ako je bila prikazana slika iz klase 61+, algoritam je u 58 % slučajeva točno predvidio da slika pripada toj klasi. Algoritam je nabolje rezultate odaziva postigao za klase 0-3 i 60+, dok je za klasu 51-60 imao najviše problema. F1-mjera predstavlja metriku koja spaja preciznost i odaziv. Visoka f1-mjera označava dobru ravnotežu između vrijednosti preciznosti i odaziva, dok niska f1-mjera ukazuje na loše performanse kod preciznosti ili odaziva. Podrška označava broj slika svake klase u skupu za testiranje. Vidljivo je da u skupu za testiranje ima najmanje primjeraka za klasu 31-40 što objašnjava slabije performanse tj. manje vrijednosti preciznosti i odaziva za tu klasu.

	precision	recall	f1-score	support
0-3	0.84	0.84	0.84	231
04-10	0.51	0.55	0.53	232
11-20	0.42	0.38	0.40	242
21-30	0.52	0.52	0.52	293
31-40	0.24	0.41	0.31	176
41-50	0.57	0.34	0.43	242
51-60	0.38	0.29	0.33	274
61+	0.50	0.58	0.54	270
accuracy			0.49	1960
macro avg	0.50	0.49	0.49	1960
weighted avg	0.50	0.49	0.49	1960

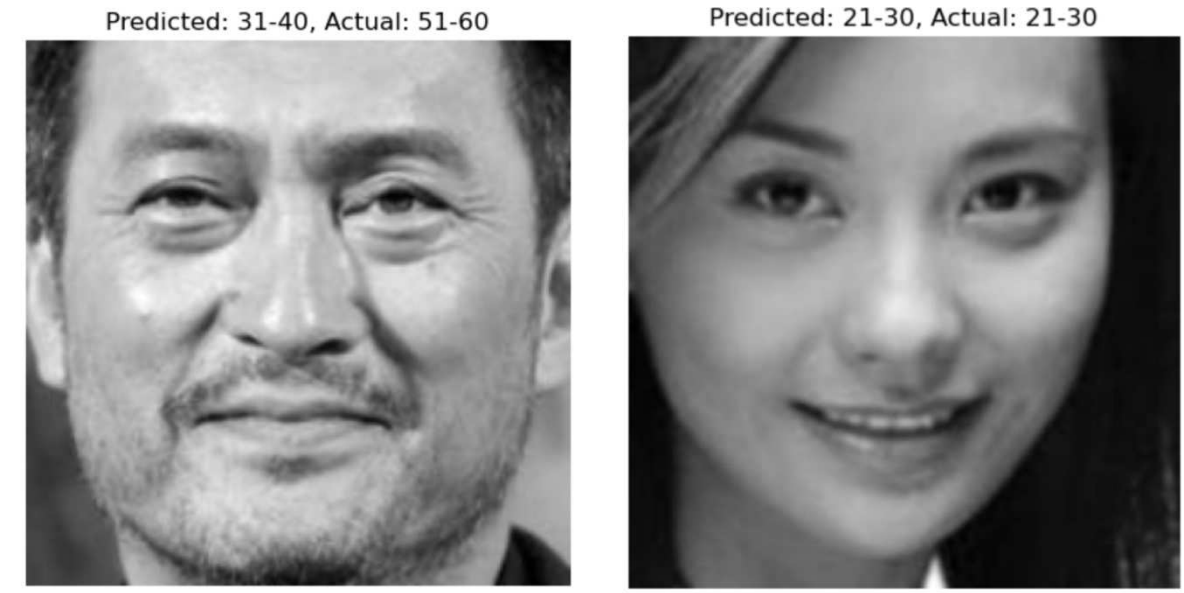
Sl. 5.1 Izvješće o klasifikaciji implementiranog algoritma

Osim izvješća o klasifikaciji, često se koristi i tzv. matrica zabune (engl. *confusion matrix*) koja prikazuje broj točnih i netočnih predikcija modela u usporedbi sa stvarnim vrijednostima klase. Redovi matrice predstavljaju točne klase, dok stupci označuju klase koje je model predvidio. Prema tome, dijagonala matrice označava točna predviđanja modela za svaku klasu. Na slici (Sl. 5.2) prikazana je matrica zabune implementiranog modela. Za klasu 0-3 godina, model je za 194 slike točno predvidio klasu, a za 26 slika je umjesto točne klase, predvidio klasu 4-10 godina. Vrijednosti na matrici najveće su blizu dijagonale, što znači da je model dosta dobro predviđao kojoj klasi slika pripada. Veće vrijednosti blizu dijagonale matrice sugeriraju da je model imao poteškoća s klasama koje su blizu jedna druge. Na primjer, za klasu 11-20, vidljivo je da je model često predviđao klasu 4-10 i 21-30 te da je povremeno predviđao da slika pripada klasi 11-20, kada je klasa zapravo bila 4-10 ili 21-30. To ima smisla s obzirom na to da je osoba koja ima 11 godina, i time pripada klasi 11-20, sličnija osobi koja ima 10 godina tj. pripada klasi 4-10 od osobe koja ima 19 godina. Sa slike je također vidljivo da je model imao problema s klasom 31-40 koju je nerijetko predviđao za sve klase između 21-30 i 61+. To se poklapa s činjenicom da je u skupu podatka za treniranje i testiranje bilo najmanje slika koje pripadaju klasi 31-40.



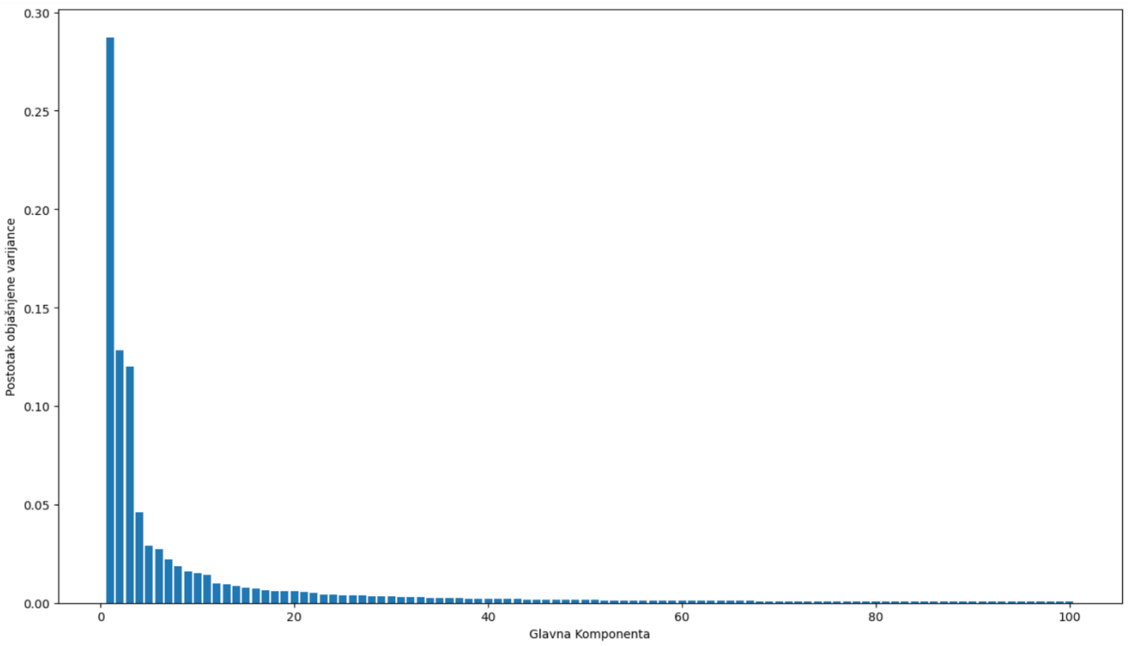
Sl. 5.2 Matrica zabune modela

Na slici (Sl 5.3) dan je primjer netočne (lijeva slika) i točne (desna slika) klasifikacije dobi osobe. Za lijevu sliku, model je predvidio da osoba pripada dobnoj skupini 31-40 umjesto klasi 51-60. To se poklapa s izgledom matrice zabune na kojoj je vidljivo da model predviđa klasu 31-40 za veliki raspon stvarnih klasa. Za desnu sliku, koja pripada klasi 21-30, algoritam je imao puno manje problema kao što je vidljivo i s izvješća o klasifikaciji i matrice zabune.



Sl. 5.3 Primjer netočne i točne klasifikacije

Na grafu (Sl. 5.4) prikazan je udio varijance objašnjen svakom od 100 odabranih glavnih komponenti. Kao što je vidljivo s grafa, prvih nekoliko komponenti objašnjava veliku većinu varijance u podacima. Specifično, u prve 3 komponente pohranjeno je 53.58 % varijance, dok je u prvih 50 komponenti pohranjeno 87.47 % ukupne varijance. Pomoću svih 100 glavnih komponenti, zapisano je čak 91.86 % varijance početnog skupa podataka. S obzirom na to da je broj dimenzija ulaznih podataka bio 224 x 224, to znači da je pomoću 0.199 % originalnog broja dimenzija podataka zapisano 91.86 % ukupne varijance.



Sl. 5.4 Udio varijance objašnjen pomoću glavnih komponenti

Da se na početnom skupu za treniranje nije koristila PCA, treniranje SVM bilo bi znatno računalno skuplje. Kako bi se to dokazalo, proveden je eksperiment na slikama smanjenih dimenzija – 64 x 64 piksela. Slike sada imaju 4096 dimenzija i direktnom primjenom SVM algoritma postiže se točnost od 49.77 %. Kako bi se model istrenirao i testirao potrebne su 3 minute i 6 sekundi. S druge strane, primjenom PCA metode sa 100 glavnih komponenti, postiže se točnost od 48.48 % sa samo 4.93 sekundi potrebnih za treniranje i testiranje SVM. To znači da smanjenjem broja dimenzija na 2.44 % broja originalnih, točnost je smanjena za 1.29 %, a izvođenje algoritma ubrzano je 37.7 puta.

6. Zaključak

U ovom radu, dan je pregled nekih od najčešćih metoda prepoznavanja dobi osobe pomoću slike lica. Objašnjene su glavne karakteristike linearne diskriminantne analize, metode k-najbližih susjeda i konvolucijskih neuronskih mreža. Implementirano je programsko rješenje koje se temelji na analizi glavnih komponenti i stroju s potpornim vektorima. Za treniranje i testiranje sustava korišteno je 9800 slika dobivenih iz UTKFace skup podataka. Algoritam je postigao preciznost od 48.88 %. Prepoznavanje dobi osobe kao problem klasifikacije ima neke važne nedostatke i probleme koje je potrebno uzeti u obzir. Glavni od njih je činjenica da se izgled osobe ne mijenja značajno ako je razlika u dobi manja od nekoliko godina. Zbog toga algoritam ima lošije performanse na rubovima klasa gdje se ne vide značajne razlike u izgledu osobe što dovodi do pogrešnih klasifikacija. Stoga, umjesto klasifikacije, zadatku bi se moglo pristupiti kao problem regresije što bi riješilo poteškoće na rubovima klasa tj. dobnih skupina. Osim toga, umjesto SVM, mogu se koristiti modeli temeljeni na CNN-u koji su značajno fleksibilniji i ostvaruju bolje rezultate nad velikim skupovima podataka. Implementirani algoritam također pati od nebalansiranog skupa podataka za treniranje te je moguće da bi se postigle bolje performanse kod uravnoteženog skupa.

7. Literatura

[1] IBM, *What is linear discriminant analysis (LDA)*?, (2023, studeni). Poveznica: https://www.ibm.com/topics/linear-discriminant-analysis

[2] Dash, S. K., *A Brief Introduction to Linear Discriminant Analysis*, (2024, veljača). Poveznica: https://www.analyticsvidhya.com/blog/2021/08/a-brief-introduction-to-linear-discriminant-analysis/

[3] StatQuest: *Linear Discriminant Analysis (LDA) clearly explained*, (2016, srpanj). Poveznica: https://www.youtube.com/watch?v=azXCzI57Yfc

[4] Alam, B., *Linear Discriminant Analysis in Machine Learning: A Beginner's Guide*, (2022, travanj). Poveznica: https://hands-on.cloud/implementation-of-linear-discriminant-analysis-lda-using-python/?utm_content=cmp-true

[5] Simplilearn, *KNN Algorithm In Machine Learning | KNN Algorithm Using Python | K Nearest Neighbor | Simplilearn*, (2018, lipanj). Poveznica: https://www.youtube.com/watch?v=4HKqjENq9OU

[6] Awan, A. A., *K-Nearest Neighbors (KNN) Classification with R Tutorial*, (2023, svibanj). Poveznica: https://www.datacamp.com/tutorial/k-nearest-neighbors-knn-classification-with-r-tutorial

[7] IBM, *What is the k-nearest neighbors (KNN) algorithm?* Poveznica: https://www.ibm.com/topics/knn

[8] Mishra, M., *Convolutional Neural Networks, Explained*, (2020, kolovoz). Poveznica: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[9] StatQuest: *Neural Networks Part 8: Image Classification with Convolutional Neural Networks (CNNs)*, (2021, ožujak). Poveznica: https://www.youtube.com/watch?v=HGwBXDKFk9I

[10] Umer, A., *Understanding Convolutional Neural Networks 🧠: A Beginner's Journey into the Architecture 🚀*, (2023, svibanj). Poveznica:

https://medium.com/codex/understanding-convolutional-neural-networks-a-beginners-journey-into-the-architecture-aab30dface10

[11] Tamanna, *Exploring Convolutional Neural Networks: Architecture, Steps, Use Cases, and Pros and Cons*, (2023, travanj). Poveznica: https://medium.com/@tam.tamanna18/exploring-convolutional-neural-networks-architecture-steps-use-cases-and-pros-and-cons-b0d3b7d46c71

[12] Jaadi, Z., *A Step-by-Step Explanation of Principal Component Analysis (PCA)*, (2024, veljača). Poveznica: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[13] Yaraghi, N., *Python PCA Tutorial: Image Classification using Principal Component Analysis*, (2023, travanj). Poveznica: https://www.youtube.com/watch?v=kX_asQvIuwk

[14] Santhosh Kumar, R., *Principal Component Analysis: In-depth understanding through image visualization*, (2019, rujan). Poveznica: https://towardsdatascience.com/principal-component-analysis-in-depth-understanding-through-image-visualization-892922f77d9f

[15] Frost, J., *Principal Component Analysis Guide & Example*.Poveznica: https://statisticsbyjim.com/basics/principal-component-analysis/

[16] IBM, *What is principal component analysis (PCA)?*, (2023, prosinac). Poveznica: https://www.ibm.com/topics/principal-component-analysis

[17] IBM, *What are support vector machines (SVMs)?*, (2023, prosinac). Poveznica: https://www.ibm.com/topics/support-vector-machine

[18] Javatpoint, *Support Vector Machine Algorithm*, Poveznica: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[19] StatQuest, *Support Vector Machines Part 1 (of 3): Main Ideas!!!*, (2019, rujan). Poveznica: https://www.youtube.com/watch?v=efR1C6CvhmE

[20] Wilimitis, D., *The Kernel Trick in Support Vector Classification*, (2018, prosinac). Poveznica: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

[21] Lee, Y., *Introduction to Support Vector Machines and Their Applications in Bankruptcy Prognosis*, (2012, srpanj). Poveznica:

https://www.researchgate.net/figure/The-illustration-of-nonlinear-SVM_fig1_228444578

[22] UTKFace dataset, Poveznica: https://susanqq.github.io/UTKFace/

8. Sažetak

Procjena dobi analizom slika lica

U ovom radu, objašnjene su neke od glavnih metoda prepoznavanja dobi po slici lica osobe kao što su linearna diskriminantna analiza, metoda k-najbližih susjeda, konvolucijske neuronske mreže, analiza glavnih komponenti i stroj s potpornim vektorima. Implementiran je model koji koristi analizu glavnih komponenti kako bi smanjio broj dimenzija sustava i zatim koristi stroj s potpornim vektorima kako bi klasificirao kojoj klasi tj. dobnoj skupini pripada osoba. Kao baza podataka koristi se UTKFace skup podatka koja sadrži oznake dobi, spola i etničke pripadnosti osobe. Implementirani model postiže točnost od 48.88 % te je, prema matrici zabune vidljivo da algoritam ima probleme s rubovima klasa što sugerira da bi se zadatku moglo pristupiti kao problemu regresije umjesto klasifikacije.

Ključne riječi: linearna diskriminantna analiza, metoda k-najbližih susjeda, konvolucijske neuronske mreže, analiza glavnih komponenti, stroj s potpornim vektorima

9. Summary

Age estimation using face images analysis

In this paper, some of the main methods for detecting the age of a person based on images are explained, such as linear discriminant analysis, the k-nearest neighbors method, convolutional neural networks, principal component analysis and support vector machines. A model was implemented that uses principal component analysis for dimensionality reduction of the dataset, after which a support vector machine is used to classify which class (age group) a person belongs to. As a database, the UTKFace dataset is used, which contains the labels for age, gender, and ethnicitiy of the person. The implemented model achieves an accuracy of 48.88 % and, based on the confusion matrix, it is evident that the algorithm struggles with the class edges, which suggests that the task could be approached as a regression problem instead of classification.

Keywords: linear discriminant analysis, k-nearest neighbors, convolutional neural networks, principal component analysis, support vector machines