

Sveučilište Josipa Jurja Strossmayera u Osijeku
Fakultet elektrotehnike, računarstva i informacijskih tehnologija
Diplomski studij

Seminarski rad iz kolegija

Obrada slike i računalni vid

Raspoznavanje osmijeha na ljudskim licima pomoću SVM

Bljeona Jahaj

Osijek, kolovoz 2021.

Sadržaj

1. Uvod	3
2. Support-vector machine	4
2.1. Što je SVM?	4
2.2. Klasifikacija pomoću SVM	5
2.2.1. Kako se računa optimalna hiperravnina?	6
3. Sustav za prepoznavanje osmijeha pomoću SVM-a	9
3.1. Razvojna okolina i podaci za treniranje	9
3.1.1. Dataset za treniranje	9
3.1.2. Treniranje klasifikatora	13
4. Zaključak	15
Literatura	16

1. Uvod

Nasmijano lice najčešći je i najvažniji izraz lica koji u normalnim okolnostima ukazuje na emocije poput oduševljenja, zadovoljstva i uzbuđenja. Postoji mnogo znanstvenih istraživanja o prepoznavanju izraza lica, poput [1] i [4], ali nema mnogo istraživanja, specifično, o prepoznavanju osmijeha na licu. Ovakvo istraživanje bi poslužilo u *real-time* upotrebi, npr. promatranje vozača, promatranje pacijenata i njihova stanja. Trenutno je moderan algoritam koji omogućuje fotografiranje na temelju činjenice smije li se osoba na slici ili ne.

Prepoznavanje osmijeha je pomno proučavani algoritam u grani računalnog vida. Ovakvo prepoznavanje zahtjeva detaljnu obradu slike kako bi algoritam za prepoznavanje zadovoljavao kriterije s obzirom na kvalitetu slike, geometriju lica, položaja lica, osvjetljenja te mnogo drugih varijabli koje utječu na detekciju osmijeha.

Obrada slike uglavnom se odnosi na uporabu i primjenu matematičkih funkcija i transformacija nad slikama. To jednostavno znači da algoritam vrši neke transformacije na slici kao što su zaglađivanje, izoštravanje, rastezanje slike ili jednostavno izvlačenje informacija iz slike. Problem koji je usko povezan s prepoznavanjem osmijeha je prepoznavanje izraza lica.

Algoritam stroja potpornih vektora (engl. Support Vector Machines (SVM)) je model nadziranog učenja koji se može koristiti za klasifikaciju i regresiju. Osmislili su ga V. N. Vapnik i A. Y. Chervonenkis 1963. godine. Originalno je SVM bio razvijen za klasifikaciju (SVC). SVC je linearni binarni klasifikator koji može rješavati nelinearne probleme pomoću jezgrinog trika (B. E. Boser, I. M. Guyon i V. N. Vapnik, 1992.) te se može proširiti na višeklasne probleme na standardan način. Regresija pomoću SVM-a (SVR) predložena je 1996. godine.

U ovom radu će se teorijski opisati gore navedeni algoritam te će se implementirati sustav za raspoznavanje osmjeha na ljudskom licu. Istrenirat će se jednostavni binarni klasifikator koji će ulazne slike klasificirati na temelju toga.

2. Support-vector machine

2.1. Što je SVM?

Stroj s potpornim vektorima (engl. Support vector machines) je skup metoda nadziranog modela učenja s povezanim algoritmima učenja koji se koriste za klasifikaciju, regresiju i detekciju. Strojevi s potpornim vektorima su vrlo efikasni alati za korištenje, no njihovi zahtjevi za računanjem, procesiranjem i pohranom povećavaju se s brojem vektora za treniranje.

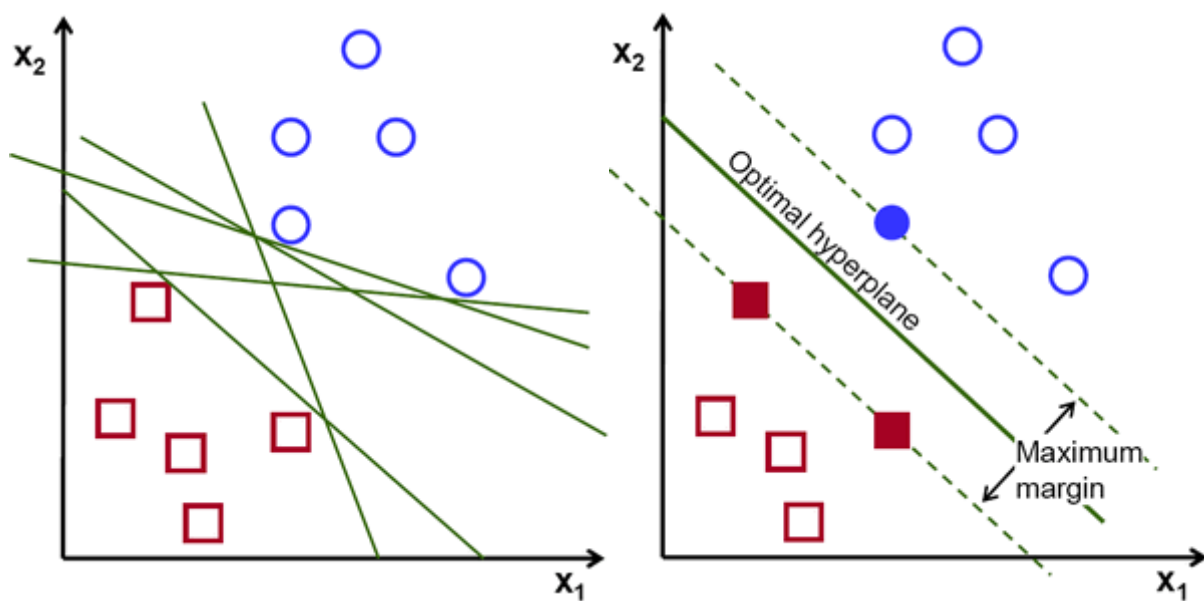
SVM konstruira hiperravninu ili skup hiperravnina u prostorima velikih ili čak i beskonačnih dimenzija, koji se onda koriste za klasifikaciju, regresiju ili za druge zadatke koji zahtijevaju sposobnosti SVM-a. Točno i efikasno razdvajanje postiže hiperravnina koja ima najveću udaljenost do najbliže točke podataka. Ova se udaljenost naziva funkcionalna margina. U slučaju SVM-a, što je veća vrijednost margine to je manja pogreška generalizacije klasifikatora.

SVM metoda koristi trening podatke kao točke u prostoru, i u postupku učenja određuje različite kategorije podataka i to na način da su podaci koji pripadaju različitim kategorijama podijeljeni jasnim što je moguće većim prostorom između njih. Tako se podaci formiraju u zasebne grupe. Ovako trenirani (naučeni) SVM se koristi za klasifikaciju novih podataka, pri čemu se ti novi podaci pridružuju grupama definiranim u postupku učenja.

Osim što provode linearnu klasifikaciju, SVM-ovi mogu učinkovito obavljati nelinearnu klasifikaciju koristeći se kernelima, implicitno preslikavajući svoje ulaze u više dimenzijske prostore značajki.

2.2. Klasifikacija pomoću SVM

Uobičajen problem strojnog učenja kada je zadan skup n - dimenzionalnih podataka, takvih da jedan dio podataka pripada jednoj klasi, a drugi pripada drugoj klasi, rješava se korištenjem upravo SVM algoritma koji utvrđuje kojoj klasi pripada određeni podatak.

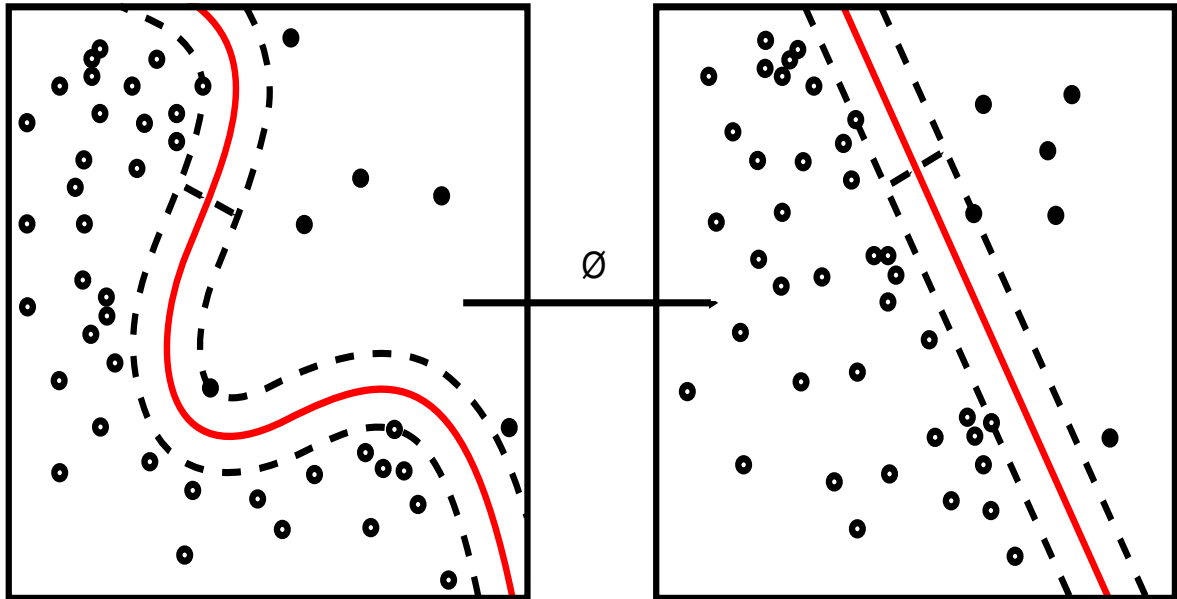


Slika 2.1. Prikaz podjele podataka i optimalne hiperravnine

2.2.1. Kako se računa optimalna hiperravnina?

Za prikaz i izračun hiperravnine koristi se izraz (2-1) gdje je β_0 poznat kao dijagonalni vektor, a β je težinski vektor.

$$f(x) = \beta_0 + \beta^T x, \quad (3-1)$$



Slika 2.2. Podjela (klasifikacija) podataka u dvije grupe

Optimalna hiperravnina može se predstaviti na beskonačan broj različitih načina skaliranjem β i β_0 . Sporazumno, među svim mogućim prikazima hiperravnina, onaj koji je izabran je:

$$|\beta_0 + \beta^T x| = 1 \quad (2-2)$$

gdje x simbolizira podatke za treniranje koji su najbliži hiperravnini. Općenito, podaci za treniranje koji su najbliži hiperravnini nazivaju se vektori podrške. Taj je prikaz poznat kao kanonska hiperravnina. Onda se koristi rezultat, dobiven geometrijski, koji daje udaljenost između točke x i hiperravnine:

$$udaljenost = \frac{|\beta_0 + \beta^T x|}{||\beta||} \quad (2-3)$$

Konkretno, za kanonsku hiperravninu, brojnik je jednak jedinici, a udaljenost do vektora podrške je:

$$udaljenost_{vektor\ potpore} = \frac{1}{||\beta||} \quad (2-4)$$

Margina koja je definirana u prethodnom ulomku ovdje je definirana kao M:

$$M = \frac{2}{||\beta||} \quad (2-5)$$

Na kraju, problem maksimiziranja M ekvivalentan je problemu minimiziranja funkcije $L(\beta)$ uz određena ograničenja. Ograničenja modeliraju zahtjev da hiperravnina pravilno razvrstava sve primjere x_i koji su se istrenirali.

$$\min_{\beta, \beta_0} L(\beta) = \frac{1}{2} ||\beta||^2 \quad (2-6)$$

$$(2-7)$$

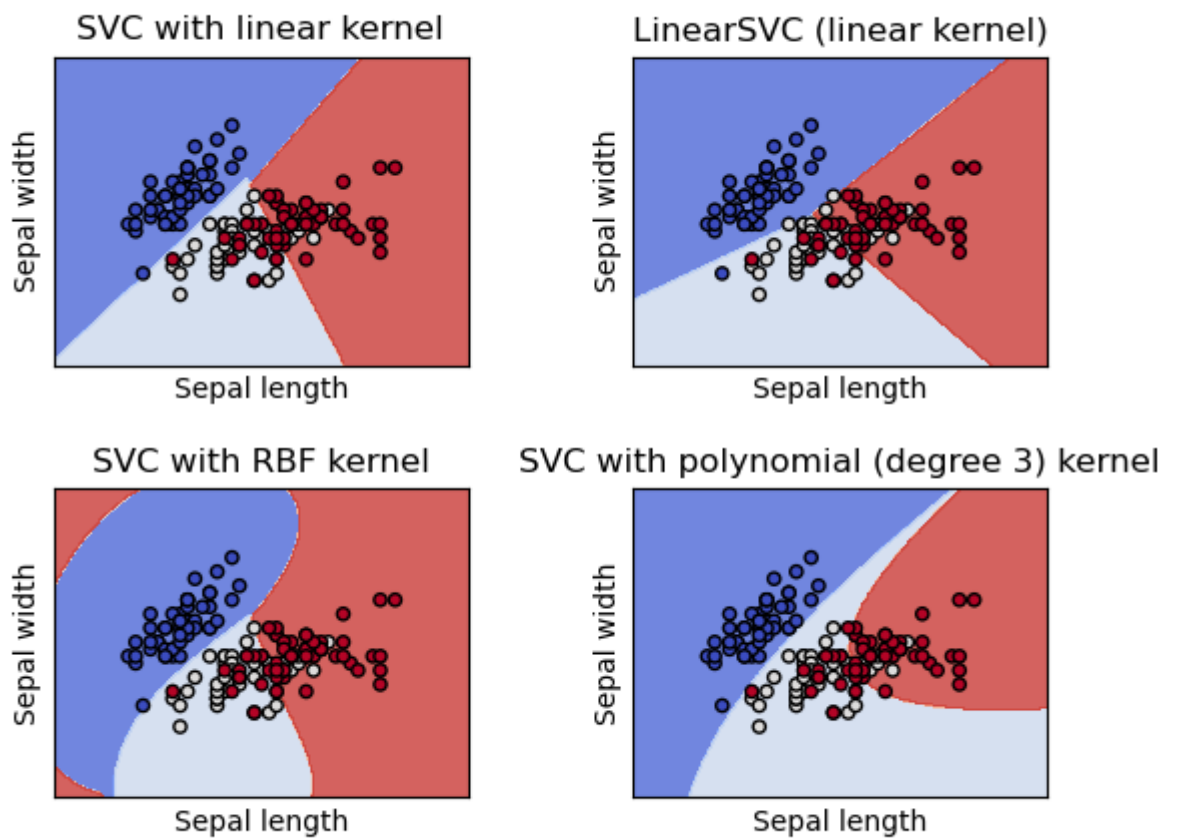
$$y_i(\beta^T x_i + \beta_0) \geq 1 \quad \forall i$$

gdje y_i predstavlja svaku od značajki istreniranih vektora.

Kad su podaci linearni, moguće ih je razdvojiti odgovarajućom hiperravninom. Postoji puno slučajeva kada podaci nisu linearni i onda ih je nemoguće razdvojiti. U takvim slučajevima koriste se različite funkcije jezgri koje nelinearno preslikavaju takve podatke u višedimenzionalan prostor. Četiri su vrste jezgre koje koristi SVM:

1. Linearna $K(x_i, x_j) = x_i^T x_j$
2. Polinomna $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0,$
3. RBF (eng. *Radial Basis Function*) $K(x_i, x_j) = e^{-\gamma(\|x_i - x_j\|^2)}, \gamma > 0$
4. Sigmoid $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

gdje su γ , r i d parametri jezgre.



Slika 2.2. Prikaz različitih plotova ovisno o vrsti jezgre koju koristi SVM

3. Sustav za prepoznavanje osmijeha pomoću SVM-a

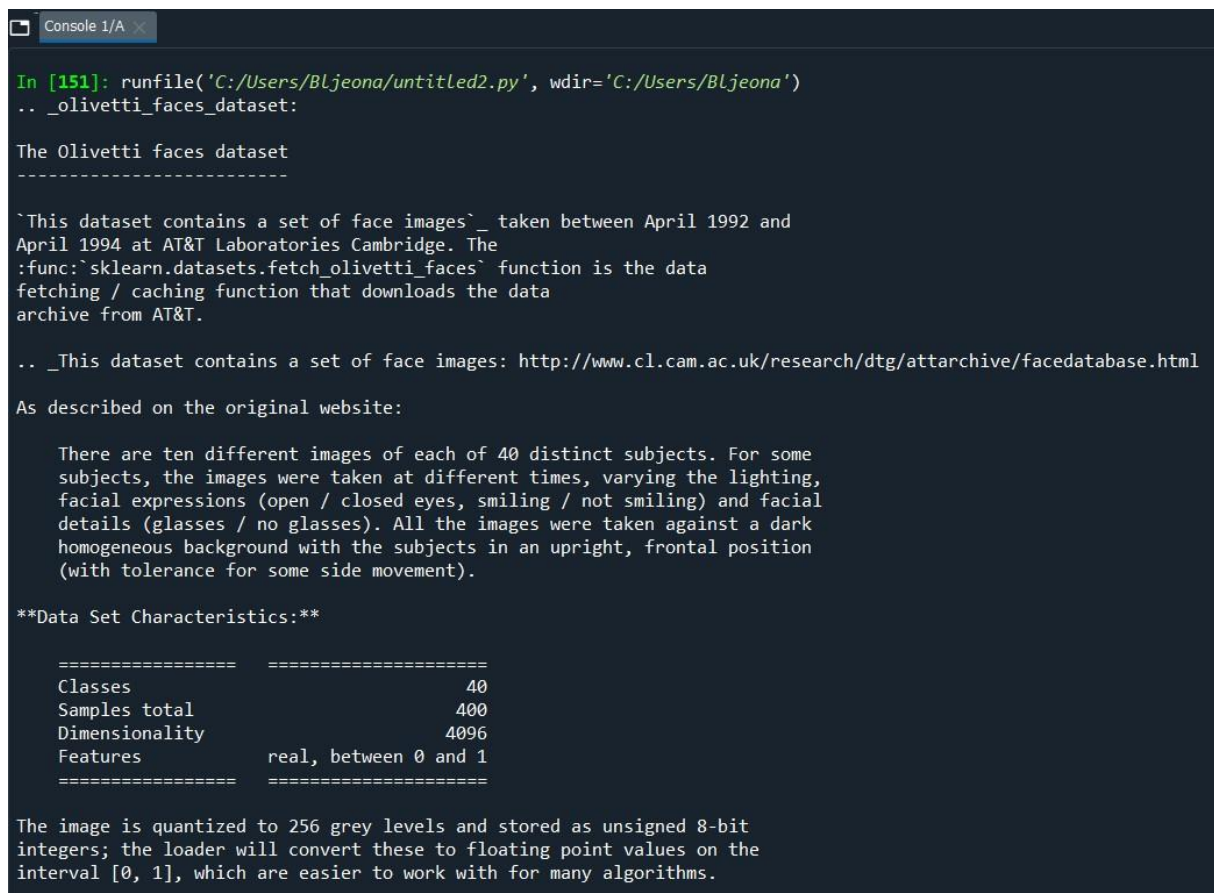
3.1. Razvojna okolina i podaci za treniranje

Za potrebe ovog seminarskog rada, provodit će se razvoj i treniranje sustava za prepoznavanje osmijeha na slikama u razvojnoj okolini JupyterLab ili u Spyderu koji dolazi uz instalaciju softvera Anaconda Cloud. Točnije, radit će se u Jupyter Notebook-u koji je prvobitno napravljen za programski jezik Python, ali je sada uz pomoć korištenja raznih kernela proširen i za druge programske jezike. Radit će se i u Spyder okruženju.

3.1.1. Dataset za treniranje

Za potrebe ovog rada koristimo dataset sa **scikit-learn**-a, *The Olivetti faces dataset*.

```
1. from sklearn.datasets import fetch_olivetti_faces
2. faces = fetch_olivetti_faces()
3. print (faces.DESCR)
```



```
In [151]: runfile('C:/Users/Bljeona/untitled2.py', wdir='C:/Users/Bljeona')
.. _olivetti_faces_dataset:

The Olivetti faces dataset
-----

`This dataset contains a set of face images`_ taken between April 1992 and
April 1994 at AT&T Laboratories Cambridge. The
:func:`sklearn.datasets.fetch_olivetti_faces` function is the data
fetching / caching function that downloads the data
archive from AT&T.

.. _This dataset contains a set of face images: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

As described on the original website:

There are ten different images of each of 40 distinct subjects. For some
subjects, the images were taken at different times, varying the lighting,
facial expressions (open / closed eyes, smiling / not smiling) and facial
details (glasses / no glasses). All the images were taken against a dark
homogeneous background with the subjects in an upright, frontal position
(with tolerance for some side movement).

**Data Set Characteristics:**

=====
Classes                40
Samples total          400
Dimensionality          4096
Features               real, between 0 and 1
=====

The image is quantized to 256 grey levels and stored as unsigned 8-bit
integers; the loader will convert these to floating point values on the
interval [0, 1], which are easier to work with for many algorithms.
```

Slika 3.1. Prikaz opisa dataseta kojeg koristimo za potrebe ovog rada u konzoli

Postoji deset različitih slika svakog od 40 različitih osoba. Za neke osobe, slike su snimljene u različito vrijeme, ovisno o osvjetljenju, izrazu lica (otvorene/zatvorene oči, nasmijan/nenasmijan) i pojedinostima (naočale/bez naočala). Sve su slike (primjer je slika 3.1.) snimljene nasuprot mračnoj pozadini sa subjektima u uspravnom, frontalnom položaju (s tolerancijom za neki pomak sa strane). Slika se kvantizira na 256 razina sive boje i pohranjuje kao 8-bitni cijeli broj; loader će ih pretvoriti u float vrijednosti na interval $[0, 1]$, s kojima je za mnoge algoritme lakše raditi.

```
1. for i in range(20):  
2.     face = faces.images[i]  
3.     subplot(4, 5, i + 1)  
4.     imshow(face.reshape((64, 64)), cmap='gray')  
5.     axis('off')
```



Slika 3.1. Prikaz dviju osoba iz dataseta iz različitih kutova

Sada kada se skup podataka učitava, izgradit ćemo jednostavni sustav za klasifikaciju 400 slika u dvije kategorije:

- nasmijano lice

- nenasmijano lice

```
1. class Trainer:
2.     def __init__(self):
3.         self.results = {}
4.         self.imgs = faces.images
5.         self.index = 0
6.
7.     def increment_face(self):
8.         if self.index + 1 >= len(self.imgs):
9.             return self.index
10.        else:
11.            while str(self.index) in self.results:
12.                print self.index
13.                self.index += 1
14.            return self.index
15.
16.    def record_result(self, smile=True):
17.        self.results[str(self.index)] = smile
```

Instanciramo klasu:

```
1. trainer = Trainer()
```

U nastavku su prikazane slike 3.2. i 3.3. nasmijanih i nenasmijanih lica koje smo klasificirali.

```
1. smiling_indices = [int(i) for i in results if results[i] == True]
2.
3. fig = plt.figure(figsize=(12, 12))
4. fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)
5. for i in range(len(smiling_indices)):
6.     # plot the images in a matrix of 20x20
7.     p = fig.add_subplot(20, 20, i + 1)
```

```

7. p.imshow(faces.images[smiling_indices[i]], cmap=plt.cm.bone)
8.
9. # Label the image with the target value
10.p.text(0, 14, "smiling")
11.p.text(0, 60, str(i))
12.p.axis('off')

```



Slika 3.2. Prikaz nasmijanih lica iz dataseta

```

1. not_smiling_indices = [int(i) for i in results if results[i] == False]
2. fig = plt.figure(figsize=(12, 12))
3. fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)
4. for i in range(len(not_smiling_indices)):
5. # plot the images in a matrix of 20x20
6. p = fig.add_subplot(20, 20, i + 1)
7. p.imshow(faces.images[not_smiling_indices[i]], cmap=plt.cm.bone)
8.
9. # Label the image with the target value
10.p.text(0, 14, "not smiling")
11.p.text(0, 60, str(i))
12.p.axis('off')

```



Slika 3.3. Prikaz nenasmijanih lica iz dataseta

3.1.2. Treniranje klasifikatora

U nastavku koristimo klasifikator vektora podrške za učenje od 400 osmijeha i „neosmijeha“ te predviđanje klasifikacije nove slike.

Prvo pokrećemo klasifikator importanjem SVC-a te ga u nastavku istreniramo:

```
1. from sklearn.svm import SVC
2. svc_1 = SVC(kernel='linear')
3.
4. ...
5.
6. train_and_evaluate(svc_1, X_train, X_test, y_train, y_test)
7.
8. ...
9.
10. def display_face_and_prediction(b):
11.     index = randint(0, 400)
12.     face = faces.images[index]
13.     display_face(face)
14.     print("Ova osoba se smije: {0}"
            .format(svc_1.predict(faces.data[index, :])==1))
```

Definicija „*display_face_and_prediction*“ nam prikazuje lice te ispisuje smije li se osoba na slici ili ne. U nastavku vidimo primjere za nasmijana lica te za nenasmijana.

1. Ova osoba se smije: [**True**]



Slika 3.3. Prikaz nasmijanih lica iz dataseta

1. Ova osoba se smije: [**False**]



Slika 3.3. Prikaz nenasmijanog lica iz dataseta

4. Zaključak

Sustav za prepoznavanje osmijeha na licu algoritam je koji se zasniva na prepoznavanju uzoraka koje detektira ili provjerava je li osoba nasmijana ili ne nasmijana lica na datoj slici ili videozapisu. Ovo je općenito učinjeno usporedbom crta lica dobivenih na slici s različitim standardnim licima dostupnih u bazama podataka. Kod detekcije osmijeha, potrebna je detaljna obrada slike kako bi se zadovoljili svi standardi za dobru kvalitetu prepoznavanja. Iz ovog rada, vidljivo je kako na pouzdanost detekcije osmijeha utječu osvjetljenje, položaj lica i geometrija lica. Za klasifikaciju osmijeha, dovoljno je pouzdan SVM klasifikator koji je opisan u ovom radu. Kod njega je specifično što koristi dijeljenje podataka hiperravninom koju stvara s obzirom na specifičnost podataka, te njihovo klasificiranje.

Literatura

- [1] M. Pantic, S. Member, and L. J. M. Rothkrantz, "Automatic Analysis of Facial Expressions: The State of the Art," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, pp.1424-1445, 2000.
- [2] B. Fasel and J. Luetttin, "Automatic Facial Expression Analysis: A Survey," Pattern Recognition, Vol. 36, pp. 259-275, 2003
- [3] Yu-Hao Huang(黃昱豪), 2 Chiou-Shann Fuh (傅楸善), „FACE DETECTION AND SMILE DETECTION 1“
- [4] Pedram Ghazi, Antti P. Happonen, Jani Boutellier, and Heikki Huttunen Tampere University of Technology Tampere, Finland, Embedded Implementation of a Deep Learning Smile Detector