

پروژه نهایی درس شناسایی الگو (تعیین جنسیت از روی تصویر چهره)

نام و نام خانوادگی: فاطمه نباتی

شماره دانشجویی: 9812358038

استاد درس: آقای دکتر ختن لو

مقدمه :

در این پروژه قصد داریم یک مدل برای gender classification از روی تصاویر چهره افراد ایجاد نماییم.

به طور کلی gender classification در application های زیادی مثل targeted advertising ، سیستم های نظارت، تعامل انسان و ماشین، ایندکس کردن و سرچ کردن content based و ... کاربرد دارد.

همان طور که میدانیم تشخیص جنسیت یک فرد برای انسان ها کار بسیار ساده ای است اما برای ماشین ها این طور نیست.

به همین دلیل پژوهش ها در حوزه gender classification از سال 1990 شروع شده است و تا مدت ها بحث داغ پژوهش ها بوده تا با وجود مشکلات و چالش هایی در این حوزه وجود دارد، بتوانند به performance خوبی برسند.

1. توضیح پروژه :

هدف اصلی در این پروژه ارائه مدلی مبتنی بر الگوریتمهای یادگیری ماشین برای تعیین جنسیت افراد است. دیتاست در نظر گرفته شده برای این پروژه دیتاست database face Tech Georgia می باشد که شامل تصاویر چهره 50 شخص مختلف است که از هر شخص 15 تصویر در اشکال و زوایا متفاوت موجود است.

مراحل پیاده سازی پروژه در ادامه با جزئیات توضیح داده خواهد شد. اما به طور خلاصه، الگوریتم و روش های استفاده شده در این پروژه شامل الگوریتم Histogram of oriented gradient (HOG) برای feature extraction ، Haar cascade classifier برای face detection ، روش های Principal Component Analysis (PCA) و Linear Discriminant Analysis (LDA) برای dimensionality reduction می باشد.

و در نهایت از مدل استفاده شده برای این پروژه، مدل Support Vector Machine (SVM) است.

مراحل انجام پروژه شامل موارد زیر است :

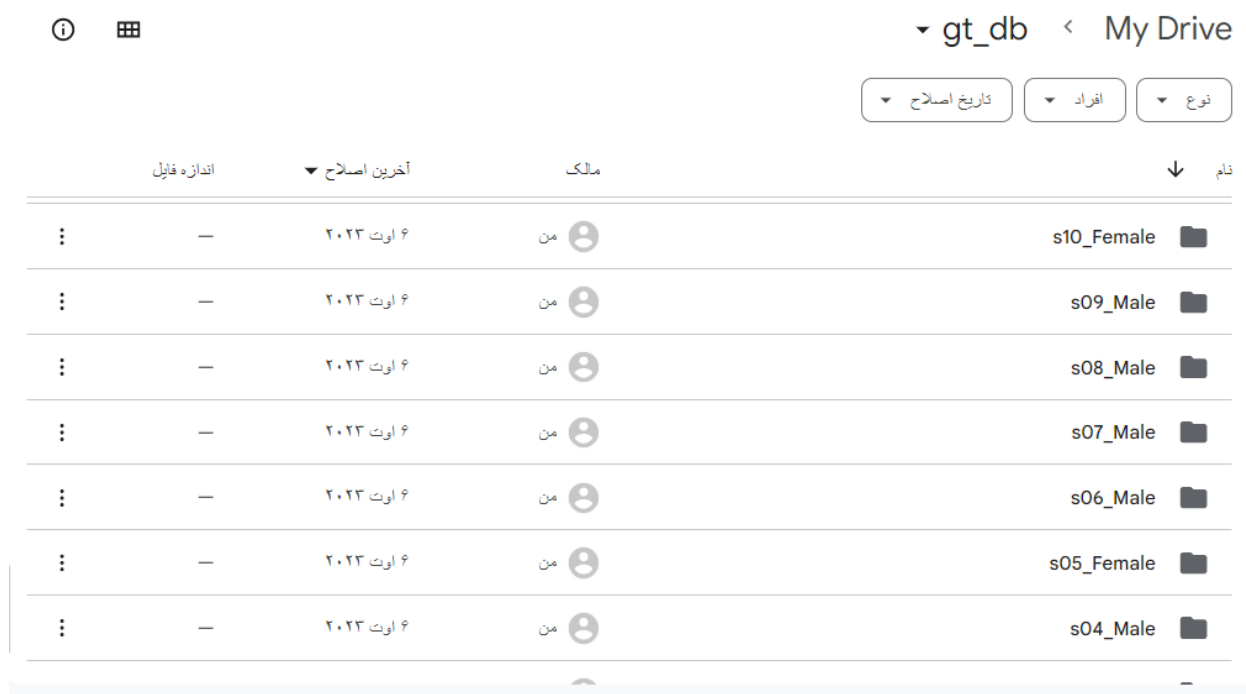
2. پیاده سازی پروژه :

2.1. کار های اولیه :

ابتدا دیتاست را دانلود می کنیم، آن را extract کرده و نام پوشه های آن را تغییر می دهیم. چون دیتاست به این صورت است که هر پوشه شامل تصاویر یک فرد است، به نام هر پوشه، جنسیت فردی که آن پوشه شامل تصاویر آن فرد است را اضافه می کنیم. در واقع نام پوشه ها به صورت s01_Male یا s01_Female می شوند.

بعدا می توانیم label تصاویر را با استفاده از نام پوشه ها، مشخص کنیم.

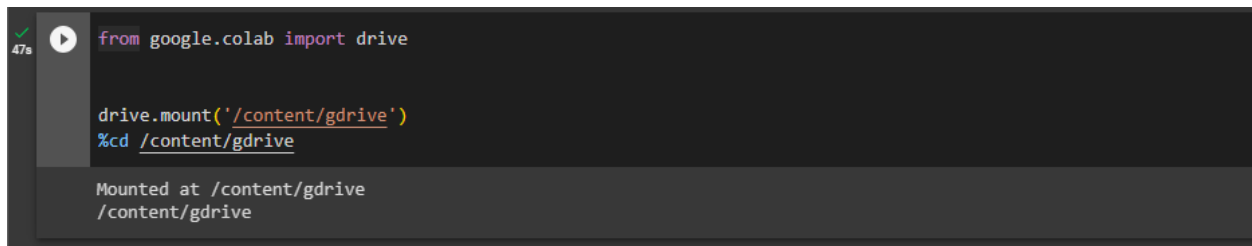
سپس دیتاست را در google drive آپلود می کنیم. (شکل 2.1 1)



| نام | مالک | آخرین اصلاح | اندازه فایل |
|------------|------|-------------|-------------|
| s10_Female | من | ۶ اوت ۲۰۲۳ | — |
| s09_Male | من | ۶ اوت ۲۰۲۳ | — |
| s08_Male | من | ۶ اوت ۲۰۲۳ | — |
| s07_Male | من | ۶ اوت ۲۰۲۳ | — |
| s06_Male | من | ۶ اوت ۲۰۲۳ | — |
| s05_Female | من | ۶ اوت ۲۰۲۳ | — |
| s04_Male | من | ۶ اوت ۲۰۲۳ | — |

شکل 2.1 1 نام های تغییر کرده پوشه های دیتاست

بعد باید به notebook ایجاد شده در google colab اجازه دسترسی به فایل های google drive را بدهیم تا بتوانیم با دیتاست کار کنیم. (شکل 2.1 2)



```
from google.colab import drive

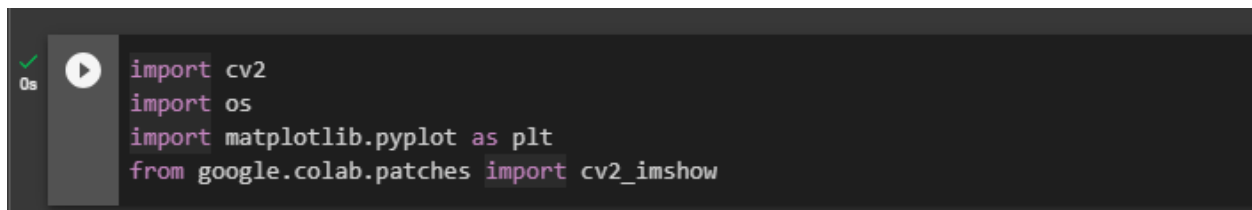
drive.mount('/content/gdrive')
%cd /content/gdrive

Mounted at /content/gdrive
/content/gdrive
```

شکل 2.1 2 کد مورد نیاز برای دسترسی به فایل های google drive

2.2. خواندن دیتاست و انجام تغییرات مورد نیاز در تصاویر و ایجاد label ها :

ابتدا تعدادی از کتابخانه های مورد نیاز را import می کنیم. (شکل 2 3)



```
import cv2
import os
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
```

شکل 2.2 1 تعدادی از کتابخانه های مورد نیاز

بعد آدرس دیتاست را مشخص کرده و دو list خالی به نام های images و labels برای نگه داری تصاویر و label های آن ها در نظر می گیریم.

```
# Set the path to the folder contains the images
path_to_images = "/content/gdrive/MyDrive/gt_db"

# Initialize two lists to hold the images and their labels
images = [] # images of dataset after required changes
labels = [] # labels of images
```

شکل 2.2 2 مشخص کردن آدرس دیتاست و تعریف کردن دو لیست برای ادامه کار

در این مرحله نیاز به یک حلقه بزرگ داریم که به ازای تمام پوشه های دیتاست کار های زیر را انجام دهیم (حلقه 1):

2.2.1.1. لود کردن تصاویر و تشخیص چهره ها :

اول با استفاده از یک حلقه داخلی دیگر (حلقه 2)، با استفاده از فانکشن `lisdire` در کتابخانه `os` ، روی تمام فایل های هر پوشه می چرخیم. (آدرس هر پوشه را از ترکیب آدرس دیتاست و نام آن پوشه به دست می آوریم). به ازای هر فایل (عکس) فانکشن `detect_faces` را کال می کنیم تا چهره های داخل هر عکس را پیدا کنیم.

2.2.1.1. تابع `detect_faces` :

بدنه این تابع به این صورت است که آدرس عکس را می گیرد و آن را با استفاده از تابع `imread` کتابخانه `opencv` لود می کند.

سپس چهره های انسان در تصویر را تشخیص می دهد و از `Haar cascade classifier` برای این کار استفاده می کند.

face detection با استفاده از Haar cascade یک روش بر اساس machine learning می باشد که در آن یک cascade function با مجموعه ای از داده های ورودی، train می شود. openCV شامل بسیاری از classifier های pre-train شده برای چهره، چشم، لبخند و ... است که در این تابع، از face classifier که در کتابخانه openCV وجود دارد، استفاده شده است.

```
def detect_faces(path_to_image: str):  
    # Load the image using OpenCV  
    img = cv2.imread(path_to_image)  
    # load face cascade  
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')  
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')  
    # Convert the image to grayscale  
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
    # search faces  
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)  
    return img, faces
```

شکل 2.2.1.1 1 بدنه تابع detect_faces

2.2.2. مرحله crop کردن و پیش پردازش کردن تصاویر :

سپس با استفاده از یک حلقه دیگر (حلقه 3)، تمام چهره های تشخیص داده شده داخل هر عکس را crop کرده و به یک لیست به نام detected_faces اضافه می کنیم.

زمانی که حلقه 2 به پایان برسد، به این معنی است که همه چهره های داخل یک پوشه تشخیص داده شده و در لیست detected_faces ذخیره شده است. پس بیرون حلقه 2 یک حلقه دیگر میگذاریم (حلقه 4) تا به ازای تمام

چهره های یک پوشه که مربوط به یک فرد است، آن ها را **resize** (پیش پردازش) کند و به لیست **images** اضافه کند.

در واقع حالا دیتاست ما به جای تصاویر اولیه، به تصاویر **crop** شده و پیش پردازش شده، تبدیل شده است.

سوال : هدف از تشخیص چهره و حذف قسمتهای اضافه تصویر چیست؟

حذف قسمت های اضافه تصویر باعث دقت و سرعت بیشتر در مراحل بعدی تشخیص جنسیت می شود. چون در یک تصویر، فقط قسمت چهره فرد برای تشخیص جنسیت اهمیت دارد و باید **feature** های آن **extract** شود و ممکن است **object** های دیگری که در تصویر وجود دارند، برای ما مشکل ایجاد کنند. پس ما قسمت چهره را در عکس تشخیص می دهیم و بقیه قسمت ها را حذف می کنیم تا بتوانیم **feature** ها مرتبط و مورد نیاز را از عکس به دست بیاوریم و به **performance** خوبی برسیم.

2.2.3. ایجاد label ها :

همان طور که در قسمت قبل گفته شد، تمام چهره های یک پوشه، مربوط به یک فرد است و نام این پوشه، جنسیت این فرد را نشان می دهد. پس در ادامه حلقه 4، **label** هر چهره را از روی نام پوشه تشخیص می دهیم (1 برای **male** و 0 برای **female**) و به لیست **labels** اضافه می کنیم.

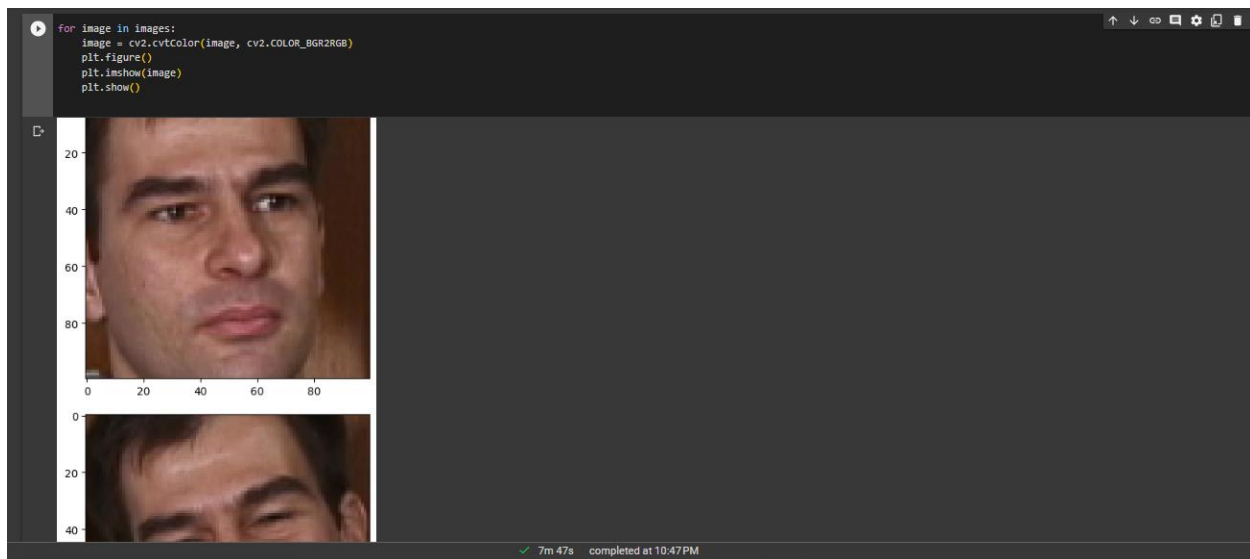
حالا لیست images و labels برای انجام مراحل بعد آماده هستند.

```
# Loop through all folders in the dataset (each folder contains 15 images of one person)
for folderName in os.listdir(path_to_images):
    print(folderName)
    detected_faces = []
    sub_path = path_to_images + "/" + folderName
    # Loop through all images in each folder
    for fileName in os.listdir(sub_path):
        if fileName.endswith(".jpg"):
            img, faces = detect_faces(os.path.join(sub_path, fileName))
            for (x1, y1, w, h) in faces:
                # cv2.rectangle(img, (x1, y1), (x1+w, y1+h), (0, 255, 0), 2)
                x2, y2 = x1 + w, y1 + h
                image = img[y1:y2, x1:x2]
                detected_faces.append(image)
    for image in detected_faces:
        # Resize the image to a fixed size (e.g. 100x100)
        resized = cv2.resize(image, (100, 100))
        # Add the resized image to the list of images
        images.append(resized)
        # Parse the label from the folderName (The folderName is in the format "number_gender.jpg")
        label = int(folderName.split("_")[1]) == "Male"
        # Add the label to the list of labels
        labels.append(label)
```

شکل 2.2 3 خواندن دیتاست و انجام تغییرات مورد نیاز در تصاویر و ایجاد label ها

2.2.4. چک کردن تصاویر و label ها :

حالا تصاویر را بعد از انجام تغییرات مورد نیاز در آن ها با استفاده از فانکشن `imshow` ، و `label` هایشان را با استفاده از `print` چاپ می کنیم تا درست بودن آن ها را بررسی کنیم.



شکل 2.2.4 1 نمایش تصاویر بعد از تغییرات مورد نیاز



شکل 2.2.4 2 نمایش label های تصاویر

2.3. استخراج ویژگی ها :

در مرحله بعد، برای همه تصاویر لیست images ، باید feature های آن ها را extract کنیم. یکی از روش هایی که برای این کار استفاده می شود، روش Histogram of Oriented Gradient (HOG) است. برای اعمال این روش، می توانیم از فانکشن hog از کتابخانه scikit-image استفاده کنیم. تصاویری که feature های آن ها extract شده را داخل لیست images میریزیم و بعد هر دو لیست images و labels را به numpy array تبدیل می کنیم.

```
[8] pip install numpy
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.23.5)

from skimage.feature import hog
from skimage import exposure
import numpy as np

X = []
# Extract features
for image in images:
    fd, hog_image = hog(image, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(2, 2), visualize=True, multichannel=True)
    # plt.axis("off")
    # plt.imshow(hog_image, cmap="gray")
    X.append(fd)

# Change images array
images = X

# Convert the lists of images and labels to numpy arrays
images = np.array(images)
labels = np.array(labels)
```

38s completed at 10:55PM

شکل 2.3 1 استخراج ویژگی ها با استفاده از روش HOG

2.4. کاهش ابعاد :

سپس ابعاد (feature های) تصاویر را یک بار با استفاده از روش PCA و یک بار با استفاده از روش LDA کاهش می دهیم تا نتیجه هر روش را روی خروجی بررسی کنیم.

تصاویری که ابعاد آن ها با استفاده از روش PCA کاهش پیدا کرده، در images_pca و تصاویری که ابعاد آن ها با استفاده از روش LDA کاهش پیدا کرده، در images_lda ذخیره شدند.

```

✓ [10] from sklearn.decomposition import PCA
1s

# Create a PCA object with 2 components
pca = PCA(n_components=2)

# Fit the PCA model to the data and transform the data
images_pca = pca.fit_transform(images)

✓ 2s
▶ from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Create an LDA object with 2 components
lda = LinearDiscriminantAnalysis(n_components=1)

# Fit the LDA model to the data and transform the data
images_lda = lda.fit_transform(images, labels)

```

شکل 2.4 1 کاهش ابعاد با استفاده از روش های PCA و LDA

2.5. جدا کردن داده های train و test و ایجاد مدل :

حالا همه چیز برای train کردن مدل آماده است :) کفایت با استفاده از فانکشن train_test_split ، 70 درصد داده ها را به عنوان داده train و 30 درصد را به عنوان داده test در نظر بگیریم.

یک بار همان لیست images را به عنوان آرگومان اول فانکشن train_test_split به آن می دهیم تا نتیجه کار را بدون انجام dimensionality reduction ببینیم.

سپس images_pca را به آن می دهیم تا نتیجه کار را بعد از انجام dimensionality reduction با استفاده از روش PCA ببینیم.

و در نهایت images_lda را به فانکشن می دهیم تا تاثیر روش LDA را با حالت های قبلی مقایسه کنیم.

در نهایت با استفاده از روش Support Vector Machine (SVM) ، مدل خود را train می کنیم.

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics

# Split the dataset into training and test sets (with 70% training data and 30% test data)
train_images, test_images, train_labels, test_labels = train_test_split(images, labels, test_size=0.3, random_state=42)

# Instantiate the Support Vector Classifier (SVC)
svc = SVC(C=1.0, random_state=1, kernel='linear')

# Fit the model
svc.fit(train_images, train_labels)
```

شکل 2.5 1 جدا کردن داده های train و test و ایجاد مدل svm

2.6. تست و ارزیابی مدل :

در نهایت با پیش بینی label های داده های test و مقایسه آن ها با label های واقعی، accuracy را حساب می کنیم.

```
# Make the predictions
predicted_labels = svc.predict(test_images)

# Measure the performance
print("Accuracy score %.3f" % metrics.accuracy_score(test_labels, predicted_labels))
```

شکل 2.6 1 تست و ارزیابی مدل

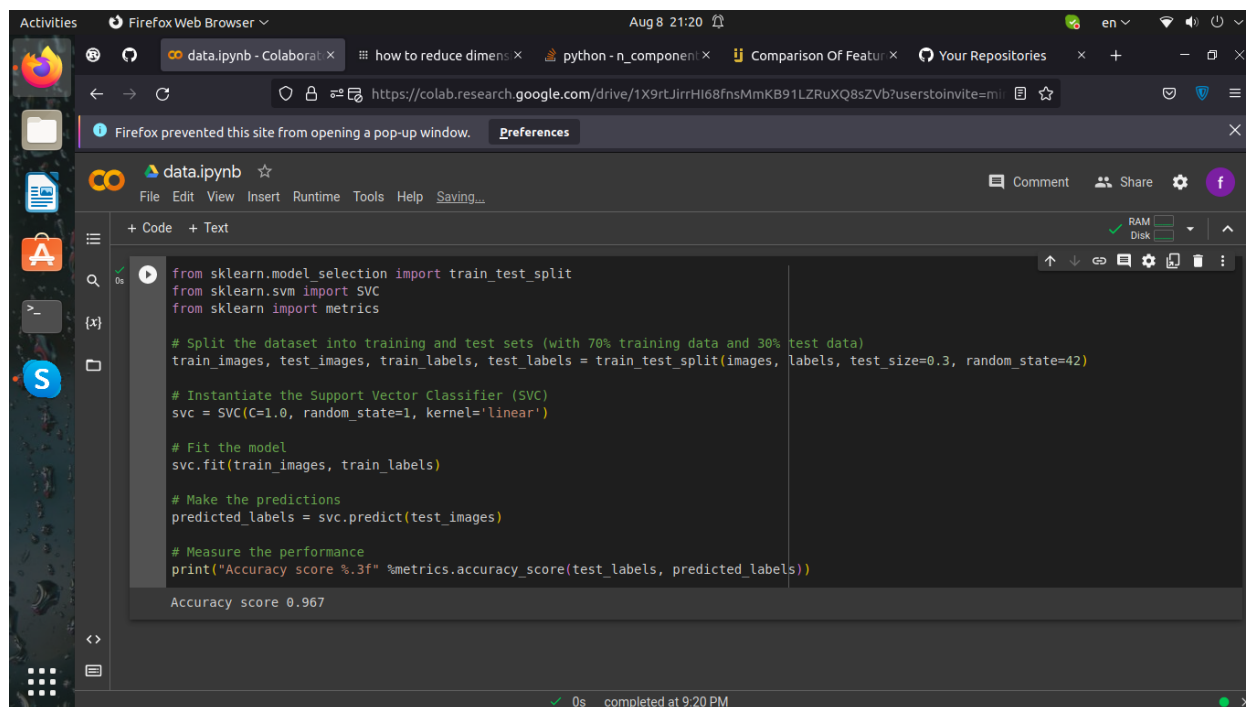
نتیجه بررسی روش های dimensionality reduction به صورت زیر می باشد :

دقت بدون اعمال روش های dimensionality reduction $\leq 96\%$

دقت با اعمال روش PCA $\leq 85\%$

دقت با اعمال روش LDA $\leq 100\%$

بنابراین روش PCA باعث کاهش دقت شده و تاثیر منفی روی نتیجه کار داشته است. اما روش LDA باعث افزایش دقت شده است.



```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics

# Split the dataset into training and test sets (with 70% training data and 30% test data)
train_images, test_images, train_labels, test_labels = train_test_split(images, labels, test_size=0.3, random_state=42)

# Instantiate the Support Vector Classifier (SVC)
svc = SVC(C=1.0, random_state=1, kernel='linear')

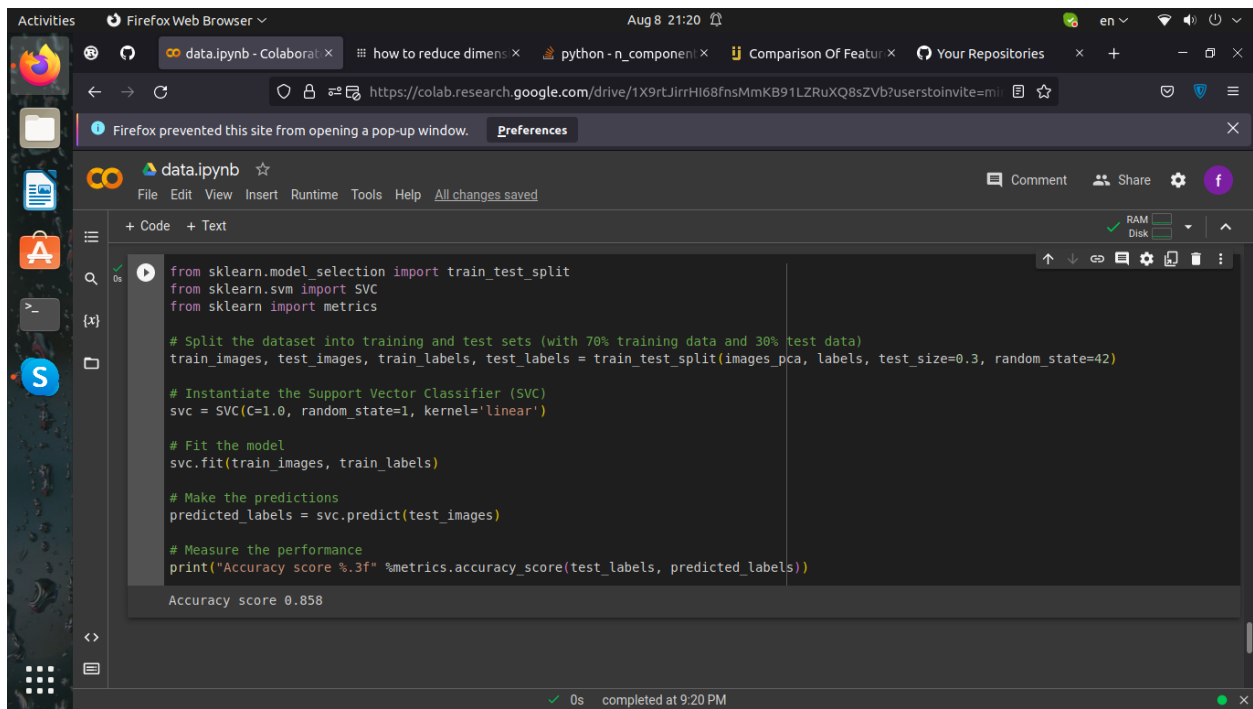
# Fit the model
svc.fit(train_images, train_labels)

# Make the predictions
predicted_labels = svc.predict(test_images)

# Measure the performance
print("Accuracy score %.3f" % metrics.accuracy_score(test_labels, predicted_labels))

Accuracy score 0.967
```

شکل 2.6 دقت بدون اعمال روش های کاهش ابعاد



The screenshot shows a web browser window with a Jupyter Notebook interface. The browser's address bar shows a Google Colab link. The notebook's code cell contains Python code for PCA and SVM. The code imports necessary libraries, splits the data, trains an SVM model, and prints the accuracy score. The output of the code cell shows an accuracy score of 0.858. The status bar at the bottom indicates the code was completed at 9:20 PM.

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics

# Split the dataset into training and test sets (with 70% training data and 30% test data)
train_images, test_images, train_labels, test_labels = train_test_split(images_pca, labels, test_size=0.3, random_state=42)

# Instantiate the Support Vector Classifier (SVC)
svc = SVC(C=1.0, random_state=1, kernel='linear')

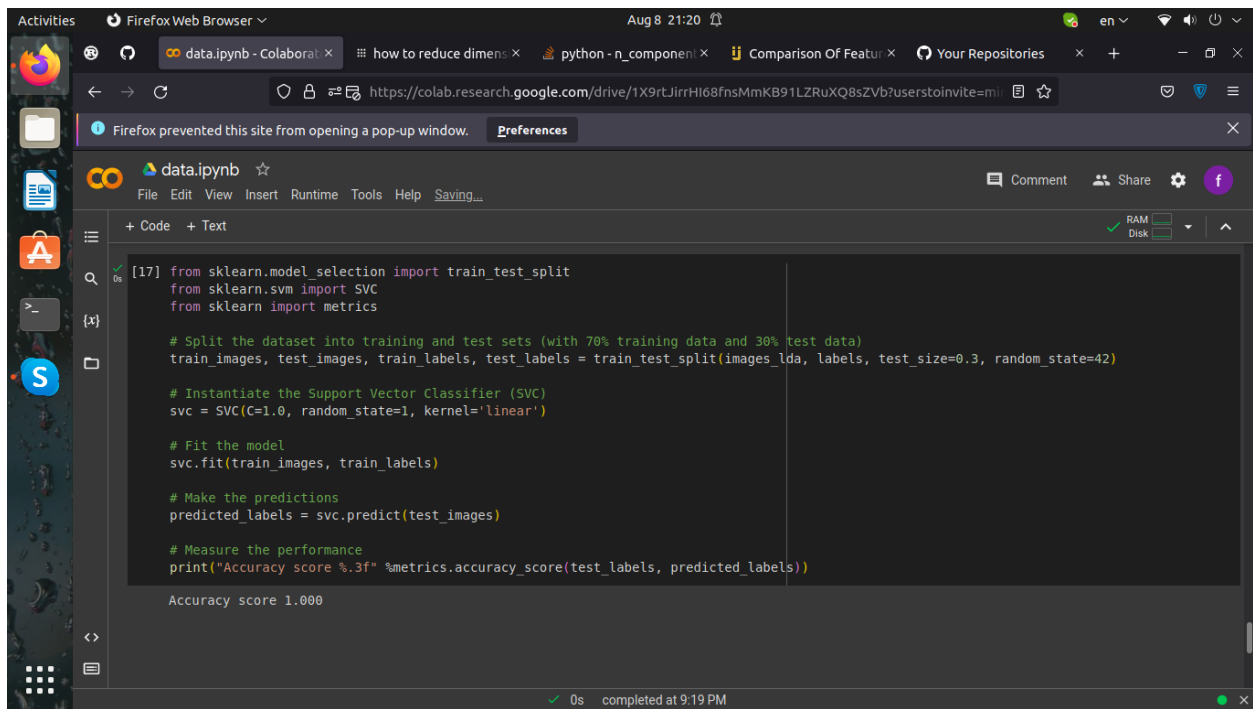
# Fit the model
svc.fit(train_images, train_labels)

# Make the predictions
predicted_labels = svc.predict(test_images)

# Measure the performance
print("Accuracy score %.3f" % metrics.accuracy_score(test_labels, predicted_labels))

Accuracy score 0.858
```

شکل 2.6 3 دقت با اعمال روش PCA



```
[17] from sklearn.model_selection import train_test_split
      from sklearn.svm import SVC
      from sklearn import metrics

      # Split the dataset into training and test sets (with 70% training data and 30% test data)
      train_images, test_images, train_labels, test_labels = train_test_split(images_lda, labels, test_size=0.3, random_state=42)

      # Instantiate the Support Vector Classifier (SVC)
      svc = SVC(C=1.0, random_state=1, kernel='linear')

      # Fit the model
      svc.fit(train_images, train_labels)

      # Make the predictions
      predicted_labels = svc.predict(test_images)

      # Measure the performance
      print("Accuracy score %.3f" % metrics.accuracy_score(test_labels, predicted_labels))

Accuracy score 1.000
```

شکل 2.6 4 دقت با اعمال روش LDA

لینک google colab پروژه :

<https://colab.research.google.com/drive/1X9rtJirrHI68fnsMmKB91LZRuXQ8sZVb#scrollTo=gXNRGGzpxtI9>

لینک github پروژه :

<https://github.com/fateme-nabati/Gender-detection>