

SKIN CANCER DETECTION

FINAL REVIEW REPORT

Submitted by

Gokina Sri Sangathya (19BCB0048)

Sighakolli Susmitha (19BCB0056)

Rahul Gupta (19BCI0225)

Prepared For

CSE4019– IMAGE PROCESSING

Submitted To

Prof RajaKumar K

Associate Professor Grade 2

School of Computer Science and Engineering



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

FALL SEMESTER 2021-2022

Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

We would like to express my special thanks and gratitude to our teacher Prof. RajaKumar K. who gave us the golden opportunity to do this wonderful project on the topic IMAGE PROCESSING through which we came to know about so many new things. We are highly indebted for his guidance as well as teaching necessary information regarding the project & also for his support in completing the project.

We would like to express our gratitude toward our parents for their kind cooperation and encouragement which helped us in the completion of this project.

We would like to express our special gratitude and thanks to our friends for giving us such attention and time.

Special thanks and appreciations also go to our classmates in developing the project and to the people who have willingly helped us out with their abilities.

Gokina Sri Sangathya

Sighakolli Susmitha

Rahul Gupta

Table of Contents:

S.NO	TOPIC	PAGE.NO
1	Abstract	4
2	Introduction	4
3	Background of the problem	5
4	Literature survey	6
5	Data Set	8
6	Flowchart of the process	9
7	Detailed Description of the modules	10
8	Code	15
9	Sample Snippets	38
10	Output	39
11	Conclusion	43
12	Future Improvements	43
13	Learnings through the project	43
14	References	45

1. Abstract

Skin cancer is a term given to the uncontrolled growth of strange skin cells. It occurs whenever unrepaired DNA damage to skin cells triggers mutations, or any other genetic defects, that lead the skin cells to multiply readily and form malignant tumors.

Image processing is a commonly used method for skin cancer detection from the appearance of affected areas on the skin. The input to the system is the skin lesion image and then by applying novel image processing techniques, it analyses it to conclude about the presence of skin cancer. The Lesion Image analysis tools checks for the various Melanoma parameters Like Asymmetry, Border, Colour, Diameter, (ABCD rule) etc. by texture, size and shape analysis for image segmentation and feature stages. The extracted feature parameters are used to classify the image as Normal skin and Melanoma cancer lesion.

2. Introduction

In recent days, skin cancer is commonly seen as one of the most dangerous forms of the Cancers identified in Humans. Skin cancer is classified into various types such as Melanoma, Basal and Squamous Cell Carcinoma out of which Melanoma is the most unpredictable and the most common form of cancer. Melanoma could be a notably deadly variety of skin cancer, and though it justifies solely 4% of all types of skin cancers, it is responsible for 75% of all skin cancer deaths. Image processing is one of the most widely used methods for skin cancer detection. ‘ Dermoscopy ’ could be a non-invasive examination technique that supports the cause of incident light beam and oil immersion technique for the visual investigation of surface structures of the skin.

The detection of melanoma using Dermoscopy is higher than individual observation- based detection, but it's diagnostic accuracy depends on the factor of training the dermatologist. The diagnosis of melanoma is not very clear and easy to identify, especially in the early stage. Thus, an automatic diagnosis tool is more effective and essential. Other than ‘Dermoscopy’, a computerized skin lesion feature detector using image processing techniques has been adapted which is more efficient than the conventional one for classification.

3. Background of the problem

In today's modern world, Skin cancer is the most common cause of death amongst humans. Skin cancer is abnormal growth of skin cells most often develops on the body exposed to the sunlight, but can occur anywhere on the body. Most skin cancers are curable at early stages . Melanoma, a type of deadly skin cancer, affects the region of skin which is exposed directly to UV radiation which shows a rapid death chance. In order to lower the death rate early detection methods are adapted. According to the survey of 2014, it has been stated that the total effects of melanoma are around 76,100 and the deaths are around 9,710. In order to decrease the death rate, image processing is used for the detection of skin cancer. By using this methodology early detection of skin cancer can be achieved. It lessens the burden of Dermatologists.

A tumor develops when cells reproduce too quickly. A tumor can be cancerous or benign. A cancerous tumor is malignant, meaning it can grow and spread to other parts of the body. A benign tumor means the tumor can grow but will not spread.

Benign Tumors:

- Benign tumors are normal cells that divide and grow too much, but do not interfere with the function of normal cells around them.
- They do not have the ability to move from where they originated.
- They are not cancerous and usually do not become cancerous, no matter how large they grow.
- Benign tumors frequently stop growing once they reach a certain size and do not invade other tissues.

Malignant Tumors:

- Malignant tumors are overgrowths of abnormal cells (cancer) that divide without control and order.
- They do not stop growing, even when they come into contact with nearby cells.
- As malignant tumors grow, they squeeze surrounding healthy tissue and prevent their normal function.
- They also release certain signals that cause the creation of new blood vessels to feed the tumor.

4. Literature Survey

After carrying out a survey on around twenty-five research papers, we can draw the inference that There is still a lot of scope for research in the field of image processing for skin cancer detection and it can be furthermore used to reduce the number of deaths caused by melanoma and other kinds of cancer. Image-based computer aided diagnosis systems have a much significant potential for screening and early detection of malignant melanoma. We reviewed the state of the art in these systems and then examined the current practices, problems, and prospects of image acquisition, pre- processing, segmentation, feature extraction and selection, and classification of dermoscopic images.

The incidence of skin cancer incidents has been drastically elevating day-to-day. Skin cancer in the early stage could be cured easily by simple procedures or techniques but advanced skin cancer cannot be treated effectively by any medications. So, there is a need to detect and treat disease at an early stage. Overall, 4% of the cancer cases are melanoma. UV-A and B are mainly responsible for skin cancer. Outdoor workers are generally more prone to skin cancer because they get easily exposed to skin cancers. So, precautionary measures like application of sunscreen lotions need to be done. It can be treated at initial stages, as the duration is extended, the chances for treating skin cancer gets hastened. New molecular therapeutic approaches for skin cancer include several medications like cryosurgery, immunomodulation with imiquimod, 5- FU, photodynamic therapy etc.

Tomographic imaging of any soft tissue like skin, has a potential role in cancer detection. The penetration of infrared wavelengths makes a confocal approach based on laser feedback interferometry feasible. Experimental results agreed with numerical simulations and structural changes were evident which would permit discrimination of healthy tissue and tumour. Furthermore, cancer type discrimination was also able to be visualized using this imaging technique.

Following are the paper we have reviewed and followed by points we have tried to incorporate in out project:

The paper published by Uzma Bano Ansari and Tanuja Sarode [2017] is where they have used Support Vector Machines as a technique and GLCM for detection of skin cancer. Gray Level Co-occurrence Matrix (GLCM) is used to extract features from an image that can be used for classification.

<https://www.irjet.net/archives/V4/i4/IRJET-V4I4702.pdf>

In the research paper submitted by Shivangi Jain, Vandana Jagtap, Nitin Pise [2015] where melanoma skin cancer cells are detected with the help of dermoscopy. because of the melanoma.

Dermoscopy is a non-invasive examination technique based on the use of incident light and oil immersion to make possible the visual examination of subsurface structures of the skin. Also we understood about other algorithms such as the seven-point checklist, ABCD rule, and the Menzies method for detecting skin cancer.

https://ac.els-cdn.com/S1877050915007188/1-s2.0-S1877050915007188-main.pdf?_tid=5d61984e-deae-4ffb-93c3-a3204aba4001&acdnat=1540761803_39537903d1935bf744caef0c305de09f

In the research paper published by Chandrahas, Varun Vadigeri, Dixit Salech [2016] where they have talked about how the existing technique is not effective in detecting skin cancer and has low efficiency and have suggested the use of GLCM as a more efficient method to go ahead with.

<https://pdfs.semanticscholar.org/04fa/61818736921742b70e25da1e5344759d90f4.pdf>

In the research paper published by Sanjay Jaiswar, Mehran Kadri, Vaishali Gatty [2014] is where they have discussed how dangerous the Melanoma cells are and have stressed the early detection of these cells. The method they have used is Computer Vision methods with the best accuracy.

https://pdfs.semanticscholar.org/de71/c8f75f45a0ab95ab64aa2c69182a67b3cda3.pdf?_ga=2.116034693.2043151333.1540761846-504861824.1540761846

In the paper published by Sheeju Dianachrist , Ramamurthy B [2012] where they have talked about the Neural Networks as a potential method with Back Propagation algorithm

to predict the Stage and type of the Skin cancer.

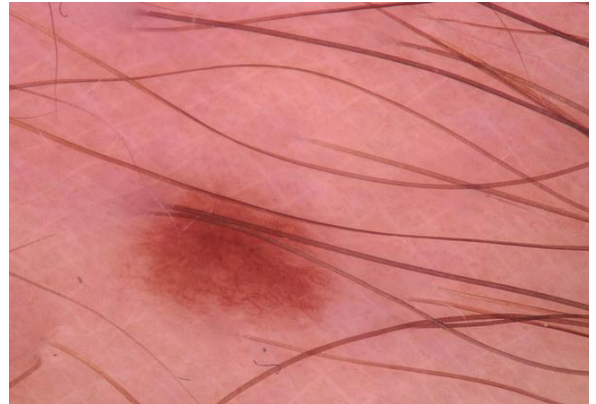
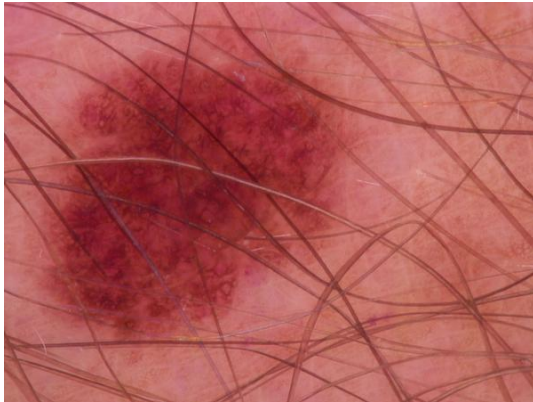
<https://www.sciencepubco.com/index.php/ijet/article/view/8643>

5. Data Set

The Data set was taken from:

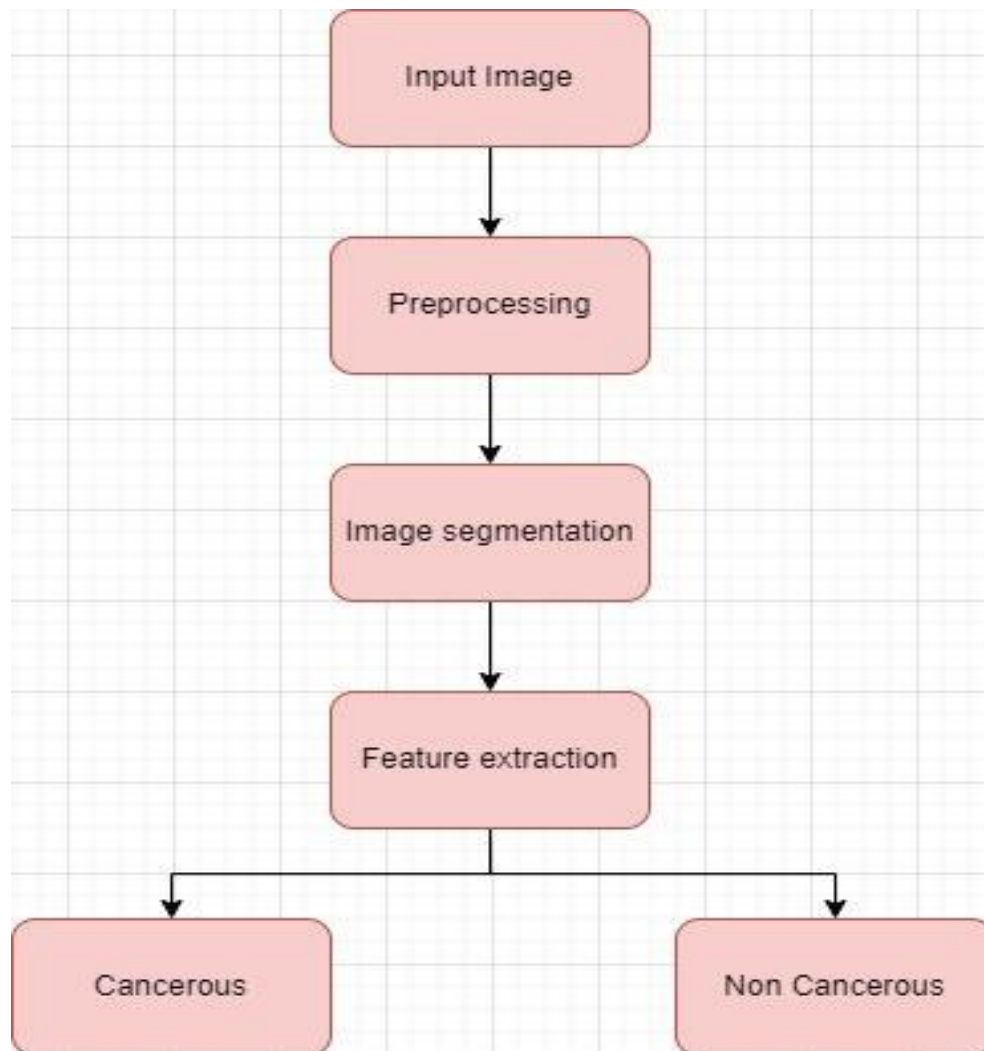
<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T>

Sample images from the dataset:



The images we took from this reference have been taken from a verified internet site.

6. Flowchart of the Process



7. Detailed description of the Modules

7.1 Input Image

Input to the proposed system are dermoscopic images, dermoscopic images are images taken by dermatoscope. It is a kind of magnifier used to take pictures of skin lesions (body part). It is a hand held instrument that makes it very easy to diagnose skin disease.

The image of skin lesions given to the computer diagnostic system can also be captured in any lighting condition or by using any camera.

Hence, it needs to pre-process.

7.2 Preprocessing

The preprocessing consists in the following 3 steps:

(i) Contrast-Limited Adaptive Histogram Equalization (CLAHE):

It is an algorithm for local contrast enhancement that uses histograms computed over different tile regions of the image.

The basic idea behind CLAHE (and also the ordinary histogram equalization) is that each pixel is transformed based on the histogram of a square surrounding the pixel using a transformation function. The transformation function is proportional to the cumulative distribution function (CDF) of pixel values in the neighbourhood. However, unlike the ordinary histogram equalization which tends to oversimplify the contrast (and thus causing noise to be amplified) in near-constant regions of the image, CLAHE limits the contrast amplification by clipping the histogram at a predefined value before computing the cumulative distribution function (CDF).

(ii) Morphological Closing:

The closing is a compound morphological operation that consists of an dilation followed by an erosion. As opposed to opening, closing can remove small dark spots and connect small bright cracks. This tends to “close” up (dark) gaps between (bright) regions.

In this project we used morphological closing to remove hairs). Images are processed using the 'morph_closing_each()' function which applies the morphological closing to the RGB image.

The closing is a compound morphological operation that consists of an dilation followed by an erosion.

(iii) **Median Filter:**

The Median Filter is a non-linear digital filtering technique, often used to remove noise from an image or signal. It is widely used in pre-processing steps to improve the results of later processing such as segmentation and feature extraction, in fact one of the main advantages is that it preserves edges while removing noise.

It is applied to each pixel of the image by using a window or structuring element having a simple shape as a disk or a square. The pixel under the origin of the structuring element is replaced with the median of the neighbouring pixels. The median is calculated by first sorting all the pixel values from the surrounding neighborhood into ascending order and then replacing the pixel being considered with the middle pixel value (if the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used).

In this work, we apply CLAHE in order to enhance the contrast and brightness of the images, while morphological closing is performed using a disk of radius 7 in order to remove hairs and stains. Morphological closing is then followed by the image filtering using the median filter in order to perform a fast and effective linear interpolation.

7.3 Image Segmentation

The `images_segmentation()` function, as its name suggests, is used to perform segmentation on the images. We use *region-based segmentation* which basically consists in finding the main region of interests (ROI) by using the watershed transform and then select among the found ROIs, the one that more probably represents the skin lesion. In particular, the segmentation process consists in the following steps:

1. Find an elevation map using the Sobel gradient of the image:

The Sobel filter is used in image processing and computer vision to detect edges and uses two 3×3 kernels which are convolved with the original image to calculate an

approximation of the gradient of the image intensity function (that is two derivatives - one for horizontal changes, and one for vertical).

2. Find markers of the background and the lesion based on the extreme parts of the histogram of gray values:

Markers are found using the threshold obtained through the IsoData filter and they are used to separate the background from the lesion region. IsoData is an histogram-based threshold, also known as Ridler-Calvard method or inter-means, based on an iterative algorithm that divides the image into two regions, the target object (foreground) and background. At the beginning, the algorithm computes the histogram of the grayscale image and then sets an initial threshold. At this point the algorithm iterates over the threshold value until it does not change by performing the following two steps:

- compute the mean of all the pixels whose gray level is equal or less than the threshold and the mean of all the pixels whose gray level is greater than the threshold.
- update the threshold to the average of the two means, that is:

$$threshold = \frac{mean(image \leq threshold) + mean(image > threshold)}{2}$$

3. Use the watershed transform to fill regions of the elevation map starting from the markers determined in the previous step:

Watershed is one of the main algorithms used for segmentation, that is, for separating different objects/regions in an image. The main idea behind watershed is that any grayscale image can be viewed as a topographic surface/map where high intensity denotes peaks and hills, while low intensity denotes valleys. The algorithm starts by filling every isolated valley (local minima) with different colored water (labels). As the water rises, depending on the peaks (gradients) nearby, water from different valleys, obviously with different colors, will start to merge. To avoid that, the algorithm builds barriers in the locations where water merges. It continues the work of filling water and building barriers until all the peaks are under water. Then the created barriers give you the segmentation result, that is they represent contours of segmented region

(i) Improve segmentation:

This is achieved through the following 3 steps:

- Fill small holes in the segmented regions using the `binary_fill_holes()` function of `scipy.ndimage` module.
- Remove small objects that have an area less than 800 pixels using the `remove_small_objects()` function of `skimage.morphology` module. This is useful to exclude the regions that do not represent a lesion.
- Clear regions connected to the image border using the `clear_border()` function of `skimage.segmentation` module. This operation is very useful when there are contour regions that have a big area and so they can be exchanged with the lesion. However, this can also create some issues when the lesion region is connected to the image borders. In order to (try to) overcome this issue, we use a (simple) lesion identification algorithm.

(ii) Apply connected components labeling algorithm:

It uses the `label()` method of `skimage.morphology` module in order to assign the same label to all the pixels that are in the same region. In this way different regions obtained after segmentation are labeled using distinct labels.

(iii) Apply the lesion identification algorithm:

The basic idea behind the lesion identification algorithm is to find the lesion among all the regions obtained from the previous steps and this is achieved by considering the area and extent features of each region. The area represents the number of pixels of the region, while the extent describes the ratio of pixels in the region to pixels in the total bounding box and is computed as $\text{area} / (\text{rows} * \text{cols})$. In fact, the basic assumption is that the region representing the lesion is the one having the greatest area (if the image has been properly filtered and segmented). Since there could be regions on the boundaries with big areas they could be potentially chosen instead of the skin lesion region. For this reason, all the contour regions are excluded at step 4. This solution works in most cases, however if the lesion is on the boundaries it is excluded and so either we haven't any label (if the lesion is the only region/object detected after the segmentation) or we have (usually) small regions that can be confused with the lesion region.

In order to (try to) solve both the above issues, the lesion identification algorithm works as follows: at the beginning it checks whether there is at least one region, it finds the largest region and then it checks if that region has an area of at least 1200 px. If so, the algorithm selects it as the target region, otherwise the lesion region may have been excluded and so in this case the algorithm backtracks to the original segmented image, applies again the connected components labeling algorithm and extracts the new region properties. Finally, in order to choose the target region, we use area and extent features by checking among the three largest regions (sorted in descending order, i.e. from the one having the largest area) the first one that has an extent greater than 0.5.

7.4 Feature Extraction

The `features_extraction()` function, as its name suggests, is used to extract some features from the segmented images in order to allow image classification.

Conventional diagnostic features are related to asymmetry, border irregularity, color variegation and diameter of skin lesion which is called ABCD analysis. The ABCD analysis has been performed by using the following parameters:

- **Asymmetry:** Asymmetry of the lesion is characterized through asymmetric index (AI) and eccentricity. In order to obtain the former parameter, AI, we first find the difference (the non-overlapping region) between the lesion image and its horizontal flip, and the one between the lesion image and its vertical flip. Then, we compute the ratio between those differences and the total lesion area and finally we take their average. AI can be expressed with the following formula:

$$AI = \frac{\frac{(image_area - image_hflip_area)}{image_area} + \frac{(image_area - image_vflip_area)}{image_area}}{2}$$

The latter represents the eccentricity of the ellipse that has the same second-moments as the region. By definition, eccentricity is the ratio of the focal distance (distance between focal points) over the major axis length.

- **Border irregularity:** Border irregularity is represented through the so-called compact index (CI), that is computed using the following formula:

$$CI = \frac{P^2}{4\pi A}$$

where P is the image perimeter and A is the image area.

- **Color variegation:** Color variegation is quantified by the normalized standard deviation of red, green and blue components of lesion. They are expressed by the following formulas:

$$C_r = \sigma_r / M_r, \quad C_g = \sigma_g / M_g, \quad C_b = \sigma_b / M_b.$$

where $\sigma_r, \sigma_g, \sigma_b$ are respectively the standard deviations of red, green and blue components of lesion area, while M_r, M_g, M_b are the maximum values of red, green and blue components in lesion region.

The normalization process is important because objects with higher normal skin pigmentation would also tend to show higher pigmentation in the lesion itself and this needs to be accounted for in order to make any comparison between cases.

- **Diameter:** The diameter of the lesion is computed as the diameter of a circle with the same area as the lesion region.

8. CODE

```
import matplotlib.pyplot as plt

import numpy as np

def image_show(image, size=None, cmap=None, no_axis=True):
```

```

"""

Plot a single image.

Parameters: image in form of nd array, size in form of length and
height tuple, string to choose color map,

            whether to show axis or not

"""

if size is None:

    size = plt.rcParams['figure.figsize']

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=size)

# dont want to display the grid
ax.grid(False)

if cmap is not None:

    ax.imshow(image, cmap=cmap)

else:

    ax.imshow(image)

# choose if axis needs to be shown or not.

if no_axis:

    ax.axis('off')

return fig, ax


def imshow_all(*images, **kwargs):

    """

    Plot a series of images side-by-side.

```


Convert all images to float so that images have a common intensity range.

Parameters

`limits : str`

Control the intensity limits. By default, 'image' is used set the min/max intensities to the min/max of all images. Setting `limits` to

'dtype' can also be used if you want to preserve the image exposure.

`titles : list of str`

Titles for subplots. If the length of titles is less than the number

of images, empty strings are appended.

`kwargs : dict`

Additional keyword-arguments passed to `imshow`.

"""

```
images = [img_as_float(img) for img in images]
```

```
titles = kwargs.pop('titles', [])
```

```
if len(titles) != len(images):
```

```
    titles = list(titles) + [''] * (len(images) - len(titles))
```

```
limits = kwargs.pop('limits', 'image')
```

```

if limits == 'image':

    kwargs.setdefault('vmin', min(img.min() for img in images))

    kwargs.setdefault('vmax', max(img.max() for img in images))

elif limits == 'dtype':

    vmin, vmax = dtype_limits(images[0])

    kwargs.setdefault('vmin', vmin)

    kwargs.setdefault('vmax', vmax)

nrows, ncols = kwargs.get('shape', (1, len(images)))

axes_off = kwargs.pop('axes_off', False)

size = nrows * kwargs.pop('size', 5)

width = size * len(images)

if nrows > 1:

    width /= nrows * 1.33

fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(width,
size))

for ax, img, label in zip(axes.ravel(), images, titles):

    ax.imshow(img, **kwargs)

    ax.set_title(label)

    ax.grid(False)

    if axes_off:

        ax.set_axis_off()

```

```

def imshow_with_histogram(image, **kwargs):

    """

    Plot an image side-by-side with its histogram.

    - Plot the image next to the histogram

    - Plot each RGB channel separately (if input is color)

    - Automatically flatten channels

    - Select reasonable bins based on the image's dtype

    """

    # create a new subplot made of 2 figures

    width, height = plt.rcParams['figure.figsize']

    fig, (ax_image, ax_hist) = plt.subplots(ncols=2, figsize=(2*width,
height))

    titles = kwargs.pop('titles', [])

    if len(titles) == 2:

        ax_image.set_title(titles[0])

        ax_hist.set_title(titles[1])

    # first plot the image

    kwargs.setdefault('cmap', plt.cm.gray)

    ax_image.imshow(image, **kwargs)

```

```

# then plot its histogram

if image.ndim == 2:

    hist, bin_centers = exposure.histogram(image)

    ax_hist.fill_between(bin_centers, hist, alpha=0.3,

                        color='black', **kwargs)

elif image.ndim == 3:

    # `channel` is the red, green, or blue channel of the image.

    for channel, channel_color in zip([channel for channel in
np.rollaxis(image, -1)], 'rgb'):

        hist, bin_centers = exposure.histogram(channel)

        ax_hist.fill_between(bin_centers, hist, alpha=0.3,

                            color=channel_color, **kwargs)

ax_hist.set_xlabel('intensity')

ax_hist.set_ylabel('# pixels')

# ax_hist.grid(False)

ax_image.set_axis_off()

# match axes height

plt.draw()

dst = ax_hist.get_position()

src = ax_image.get_position()

ax_hist.set_position([dst.xmin, src.ymin, dst.width, src.height])

return ax_image, ax_hist

```

```

# radius of structuring_element/kernel is 7. so total rows and cols are 15
structuring_element = morphology.disk(7)

# closing means dilation followed by an erosion
# Closing can remove small dark spots (i.e. "pepper") and connect small
bright cracks.

@adapt_rgb(each_channel)
def morph_closing_each(image, struct_element):
    return morphology.closing(image, struct_element)

@adapt_rgb(hsv_value)
def morph_closing_hsv(image, struct_element):
    return morphology.closing(image, struct_element)

@adapt_rgb(each_channel)
def median_filter_each(image, struct_element):
    return filters.median(image, struct_element)

```

```

@adapt_rgb(hsv_value)

def median_filter_hsv(image, struct_element):

    return filters.median(image, struct_element)


def preprocessing(image_name):

    """

    Apply some image processing techniques (CLAHE, morphology closing and
median
filter) to images.

    Parameters
    -----

    Returns training set (DataFrame), test set (DataFrame) and class
labels

    """

    print('-- PREPROCESSING --')

    # image_name="ISIC_0024320.jpg"

    segfolder = "C:/Users/My Pc/Desktop/Untitled Folder/segmeneted"

    # read the image

    image = io.imread(image_name)

```

```

# plot basic image info

print('{:_<100}'.format(''))

print('Image name: {}'.format(image_name))

print('Image sizes: {}'.format(image.shape))

print('Image data type: {}'.format(image.dtype))


# 1) Equalize the image contrast using the Contrast-Limited Adaptive
# Histogram Equalization (CLAHE) method.

equalized_adapthist = exposure.equalize_adapthist(image)


# 2) Use morphological closing to remove hairs (in order to simulate
the Razor filter).

# Images are processed using the 'morph_closing_each()' function
which applies

# the morphological closing to each channel of the RGB image.

img_morph_closing = morph_closing_each(

    equalized_adapthist, structuring_element)


# 3) Use the median filter to apply interpolation on the obtained
image.

# The median filter is implemented using the 'median_filter_each()'
# function which pass each of the RGB channels to the median filter
# one-by-one, and stitch the results back into an RGB image.

img_filtered = median_filter_each(img_morph_closing,
structuring_element)

```

```

# plot the original image with its histogram

imshow_with_histogram(image, titles=['Original image', 'Histogram'])

# plot the image obtained after applying the CLAHE method

imshow_with_histogram(equalized_adapthist, titles=[

    'Image equalized (CLAHE)', 'Histogram'])

# plot the images obtained using the morphological closing and the one
# obtained after applying the median filter

imshow_all(img_morph_closing, img_filtered, axes_off=True,

    titles=['Morphological Closing', 'Median Filter'])

plt.show()

# save the final processed image

finalpath = os.path.join(segfolder, image_name)

io.imwrite(finalpath, img_filtered)

def images_segmentation(image_name):

    print('-- REGION-BASED SEGMENTATION --')

```



```

# Python dictionaries containing the properties of the regions
representing

# the lesions

segmented_train_set = {}

# read image

segfolder = "C:/Users/My Pc/Desktop/Untitled Folder/segmeneted"

seg_path = os.path.join(segfolder, image_name)

image = io.imread(seg_path)

# convert the original image into grayscale

gray_img = color.rgb2gray(image)

# 1] Apply Sobel filter

elevation_map = filters.sobel(gray_img)

# 2] Build image markers using the threshold obtained through the
ISODATA filter

markers = np.zeros_like(gray_img)

threshold = filters.threshold_isodata(gray_img)

markers[gray_img > threshold] = 1

markers[gray_img < threshold] = 2

# 3] Apply Wathershed algorithm in order to segment the image filtered

# using the markers

segmented_img = morphology.watershed(elevation_map, markers)

```

```

# 4] Improve segmantation:

#     > Fill small holes

segmented_img = ndi.binary_fill_holes(segmented_img - 1)

#     > Remove small objects that have an area less than 800 px:

#         this could be useful to exclude some regions that does not
represent a lesion

segmented_img = morphology.remove_small_objects(
    segmented_img, min_size=800)

#     > Clear regions connected to the image borders.

#         This operation is very useful when there are contour regions
have a

#         big area and so they can be exchanged with the lesion.

#         However, this can also create some issues when the lesion
region is

#         connected to the image borders. In order to (try to) overcome
this

#         issue, we use a lesion identification algorithm (see below)

img_border_cleared = segmentation.clear_border(segmented_img)

# 5] Apply connected components labeling algorithm:

#     it assigns labels to a pixel such that adjacent pixels of the
same

#     features are assigned the same label.

# labeled_img, _ = ndi.label(segmented_img)

labeled_img = morphology.label(img_border_cleared)

```

```

# create a subplot of 3 figures in order to show elevation map,
# markers and the segmented image

fig, ax = plt.subplots(1, 3, figsize=(10, 8))

ax[0].imshow(elevation_map, cmap=plt.cm.gray)

ax[0].set_title('elevation map')

ax[0].set_axis_off()


ax[1].imshow(markers, cmap=plt.cm.nipy_spectral)

ax[1].set_title('markers')

ax[1].set_axis_off()


ax[2].imshow(segmented_img, cmap=plt.cm.gray)

ax[2].set_title('segmentation')

ax[2].set_axis_off()


plt.tight_layout()

plt.show()


# 6] Lesion identification algorithm:

# Compute properties of labeled image regions:

# it will be used to automatically select the region that contains
# the skin lesion according to area and extent

props = measure.regionprops(labeled_img)

# num labels -> num regions

```

```

num_labels = len(props)

# Get all the area of detected regions

areas = [region.area for region in props]

# If we have at least one region and the area of the region having the
# biggest area is at least 1200 px, we choose it as the region that
# contains the lesion because if properly segmented (i.e., after
removing
# small objects and regions on the image contours (since in most of
the
# images, the lesion is in the center))

if num_labels > 0 and areas[np.argmax(areas)] >= 1200:

    print('Num labels:', num_labels)

    print('Areas: {}'.format(areas))

    target_label = props[np.argmax(areas)].label

else:

    # ... otherwise we could have one of the following two cases:

    # 1] num_labels == 0:

    #     this can happen when there is only one region (the one
containing the lesion)

    #     but it has been deleted when applying the function
segmentation.clear_border()

    # 2] num_labels > 0 but areas[np.argmax(areas)] < 1200:

    #     it means that there exists at least one region but all
the regions have

    #     an area less than 1200 pixels.

```

```

        # This can happen when the region containing the lesion is
deleted

        # with segmentation.clear_border() but there still other
regions that

        # could be "exchanged for" a lesion region

        #

        # Since both cases can be due to the deletion of the segmented
area

        # because of the use of segmentation.clear_border(), the idea
is to

        # backtrack to the original segmented image (the one obtained

        # obtained before applying segmentation.clear_border()), apply
again the

        # connected components labeling algorithm and extract the new
region properties.

        # In addition, in order to find the region representing the
lesion,

        # we use area and extent features by checking among the three
largest regions

        # (if any because there could be only one or two regions)
sorted in ascending order

        # (i.e. from the one having the largest area) the first that
has an extent grater than 0.5.

labeled_img = morphology.label(segmented_img)

        # Get new region properties

props = measure.regionprops(labeled_img)

        # Get the new list of areas

areas = [region.area for region in props]

```

```

        # List of regions' extent.

        # Each extent is defined as the ratio of pixels in the region
to pixels

        # in the total bounding box (computed as: area / (rows *
cols))

    extents = [region.extent for region in props]

    print('Num labels: {}'.format(len(props)))

    print('Areas: {}'.format(areas))

    print('Extents: {}'.format(extents))

    # Get the index of the region having the largest area and if
there are

    # more than one or two regions, find also the index of the
second and

    # third most largest regions.

    region_max1 = np.argmax(areas)

    if len(props) > 1:

        areas_copy = areas.copy()

        areas_copy[region_max1] = 0

        region_max2 = np.argmax(areas_copy)

    if len(props) > 2:

        areas_copy[region_max2] = 0

        region_max3 = np.argmax(areas_copy)

    # If the largest region has an extent greater than 0.50, it is our
target region

```

```

if extents[region_max1] > 0.50:

    target_label = props[region_max1].label

    # ... else check if the extent of the second largest region is
greater than 0.5,

    # and if so we have found our target region

elif len(props) > 1 and extents[region_max2] > 0.50:

    target_label = props[region_max2].label

    # ... else if the third largest region has an extent greater
than 0.50,

    # it is (more probably) the one containing the lesion

elif len(props) > 2 and extents[region_max3] > 0.50:

    target_label = props[region_max3].label

    # ... otherwise we choose the largest region

else:

    target_label = props[region_max1].label

    # NOTE: another possible approach could be to select as the
target region

    #         the one having the largest extent among the 3 largest
regions found:

    # if len(props) > 2:

    #         extents_largest_reg = [val if idx in (region_max1,
region_max2, region_max3) else 0.0

    #                                     for idx, val in
enumerate(extents)]

    # elif len(props) > 1:

```

```

        # extents_largest_reg = [val if idx in (region_max1,
region_max2) else 0.0

        # for idx, val in
enumerate(extents)]

        # else:

        # extents_largest_reg = [val if idx in (region_max1,) else
0.0

        # for idx, val in
enumerate(extents)]

        # target_label = props[np.argmax(extents_largest_reg)].label

    # assign label 0 to all the pixels that are not in the target
region (that is

    # the region that more probably contains the lesion)

    for row, col in np.ndindex(labeled_img.shape):

        if labeled_img[row, col] != target_label:

            labeled_img[row, col] = 0

    # Convert the labeled image into its RGB version

    image_label_overlay = color.label2rgb(labeled_img, gray_img)

    print('Chosen label: {}'.format(target_label))

    # Plot the original image ('image') in which the contours of all
the

    # segmented regions are highlighted

    fig, axes = plt.subplots(1, 2, figsize=(8, 6), sharey=True)

    axes[0].imshow(image)

```



```

        axes[0].contour(segmented_img, [0.5], linewidths=1.2, colors='y')

        axes[0].axis('off')

        # Plot 'image_label_overlay' that contains the target region
highlighted

        axes[1].imshow(image_label_overlay)

        axes[1].axis('off')

    plt.tight_layout()

    plt.show()

    # Add the the found region into the proper dictionary according to
whether

    # the current image belongs to the training or the test set

    segmented_train_set[image_name] = props[target_label - 1]

    return segmented_train_set

import os

def features_extraction(segmented_region_train, image_name):

    print('-- FEATURE EXTRACTION --')

    segmented_regions = {**segmented_region_train} # Python 3.5

    # segmented_regions = {k: v for d in (segmented_region_train,
segmented_region_test) for k, v in d.items()}

```

```

print (segmented_regions)

for idx, image_name in enumerate(segmented_region_train):

    print('{:_<100}'.format(''))

    print('Image name: {}'.format(image_name))


    image = io.imread("ISIC_0024320.jpg")

    gray_img = color.rgb2gray(image)


    lesion_region = segmented_regions[image_name]


    # 1] ASYMMETRY

    area_total = lesion_region.area

    img_mask = lesion_region.image


    horizontal_flip = np.fliplr(img_mask)

    diff_horizontal = img_mask * ~horizontal_flip


    vertical_flip = np.flipud(img_mask)

    diff_vertical = img_mask * ~vertical_flip


    diff_horizontal_area = np.count_nonzero(diff_horizontal)

    diff_vertical_area = np.count_nonzero(diff_vertical)

```

```

    asymm_idx = 0.5 * ((diff_horizontal_area / area_total) +
(diff_vertical_area / area_total))

    ecc = lesion_region.eccentricity

    # mmr = lesion_region.minor_axis_length /
lesion_region.major_axis_length

    print('-- ASYMMETRY --')

    print('Diff area horizontal:
{:.3f}'.format(np.count_nonzero(diff_horizontal)))

    print('Diff area vertical:
{:.3f}'.format(np.count_nonzero(diff_vertical)))

    print('Asymmetric Index: {:.3f}'.format(asymm_idx))

    print('Eccentricity: {:.3f}'.format(ecc))

    # print('Minor-Major axis ratio: {}'.format(mmr))

    #imshow_all(img_mask, horizontal_flip,
diff_horizontal,titles=['image mask', 'horizontal flip', 'difference'],
size=4, cmap='gray')

    #imshow_all(img_mask, vertical_flip, diff_vertical,titles=['image
mask', 'vertical flip', 'difference'], size=4, cmap='gray')

    plt.show();

    # 2] Border irregularity:

    compact_index = (lesion_region.perimeter ** 2) / (4 * np.pi *
area_total)

    print('\n-- BORDER IRREGULARITY --')

```

```

print('Compact Index: {:.3f}'.format(compact_index))

# 3] Color variegation:

sliced = image[lesion_region.slice]

lesion_r = sliced[:, :, 0]

lesion_g = sliced[:, :, 1]

lesion_b = sliced[:, :, 2]

C_r = np.std(lesion_r) / np.max(lesion_r)

C_g = np.std(lesion_g) / np.max(lesion_g)

C_b = np.std(lesion_b) / np.max(lesion_b)

print('\n-- COLOR VARIEGATION --')

print('Red Std Deviation: {:.3f}'.format(C_r))

print('Green Std Deviation: {:.3f}'.format(C_g))

print('Blue Std Deviation: {:.3f}'.format(C_b))

# imshow_all(lesion_r, lesion_g, lesion_b)

# plt.show();

# Alternative method to compute colorfulness:

# https://www.pyimagesearch.com/2017/06/05/computing-image-colorfulness-with-opencv-and-python/

# compute rg = Red - Green

# rg = np.absolute(lesion_r - lesion_g)

```

```

# compute yb = 0.5 * (Red + Green) - Blue

# yb = np.absolute(0.5 * (lesion_r + lesion_g) - lesion_b)

#

# compute the mean and standard deviation of both `rg` and `yb`

# (rb_mean, rb_std) = (np.mean(rg), np.std(rg))

# (yb_mean, yb_std) = (np.mean(yb), np.std(yb))

#

# combine the mean and standard deviations

# std_root = np.sqrt((rb_std ** 2) + (yb_std ** 2))

# mean_root = np.sqrt((rb_mean ** 2) + (yb_mean ** 2))

#

# derive the "colorfulness" metric (color index)

# color_index = std_root + (0.3 * mean_root)


# 4] Diameter:

eq_diameter = lesion_region.equivalent_diameter

print('\n-- DIAMETER --')

print('Equivalent diameter: {:.3f}'.format(eq_diameter))

# optionally convert the diameter in mm, knowing that 1 px = 0.265
mm:

# 1 px : 0.265 mm = diam_px : diam_mm -> diam_mm = diam_px * 0.265

print('Diameter (mm): {:.3f}'.format(eq_diameter * 0.265))

```

```
image="ISIC_0024320.jpg"

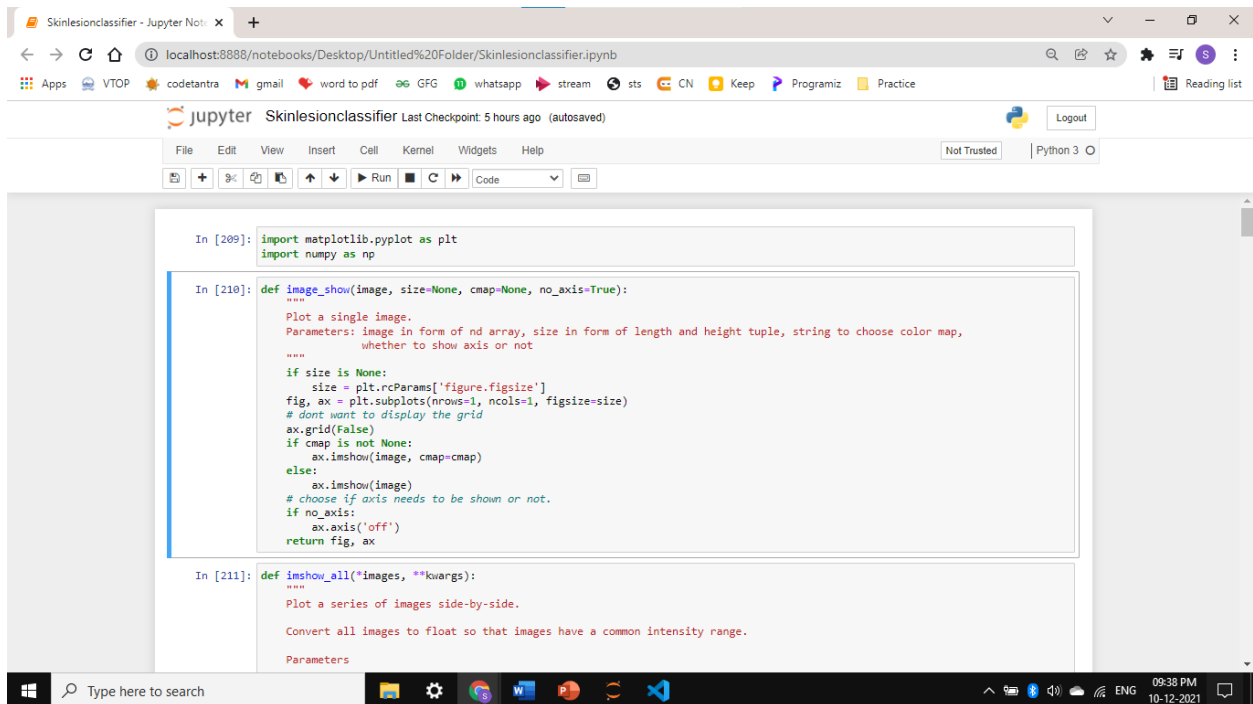
image_name="ISIC_0024320"

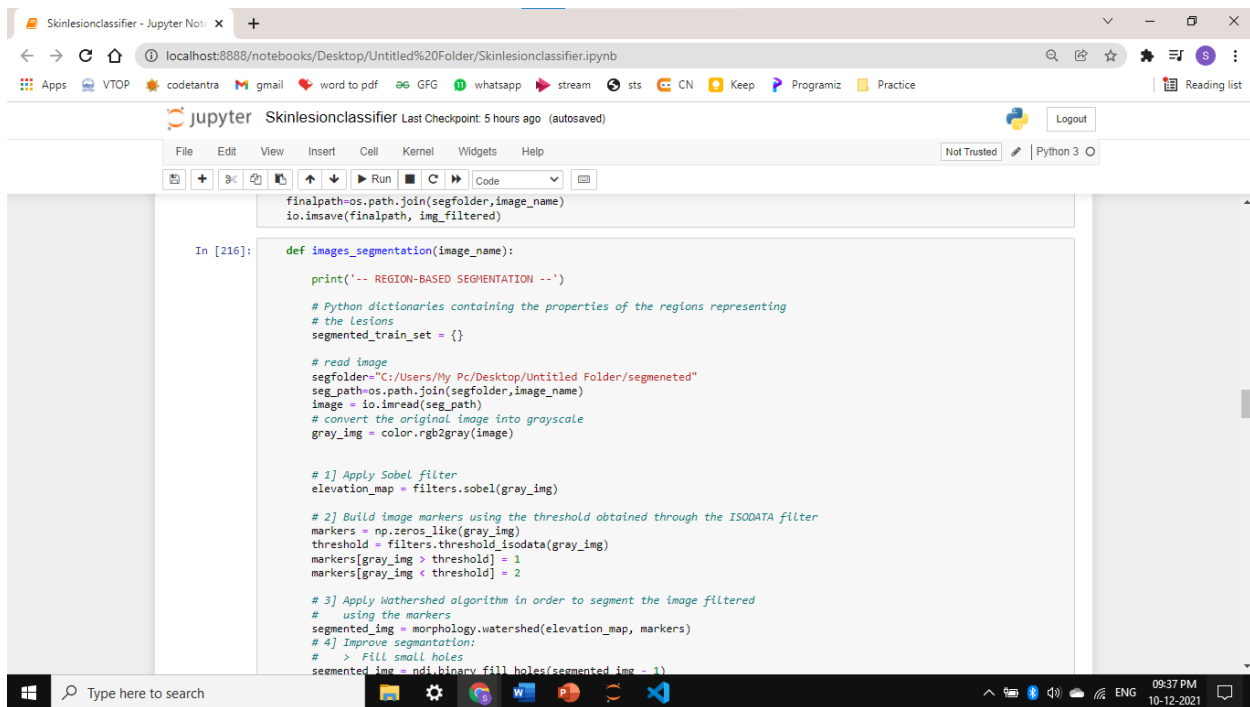
preprocessing(image)

segmented_region_train = images_segmentation(image)

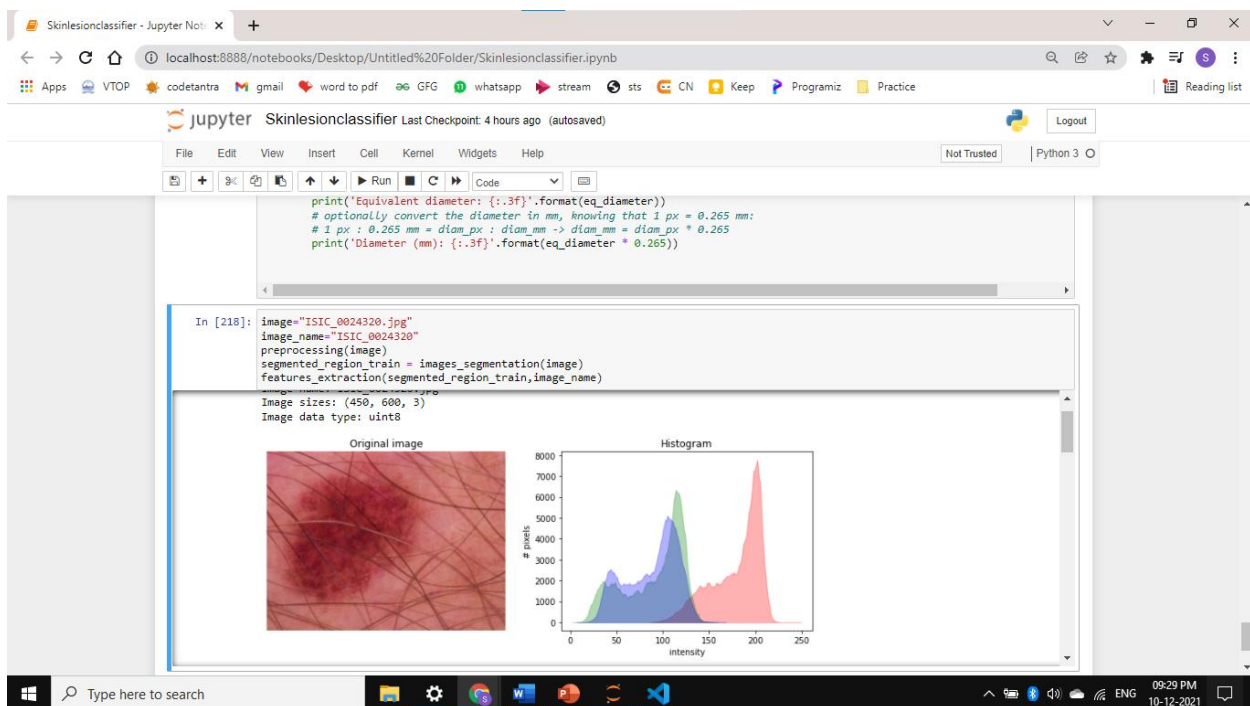
features_extraction(segmented_region_train,image_name)
```

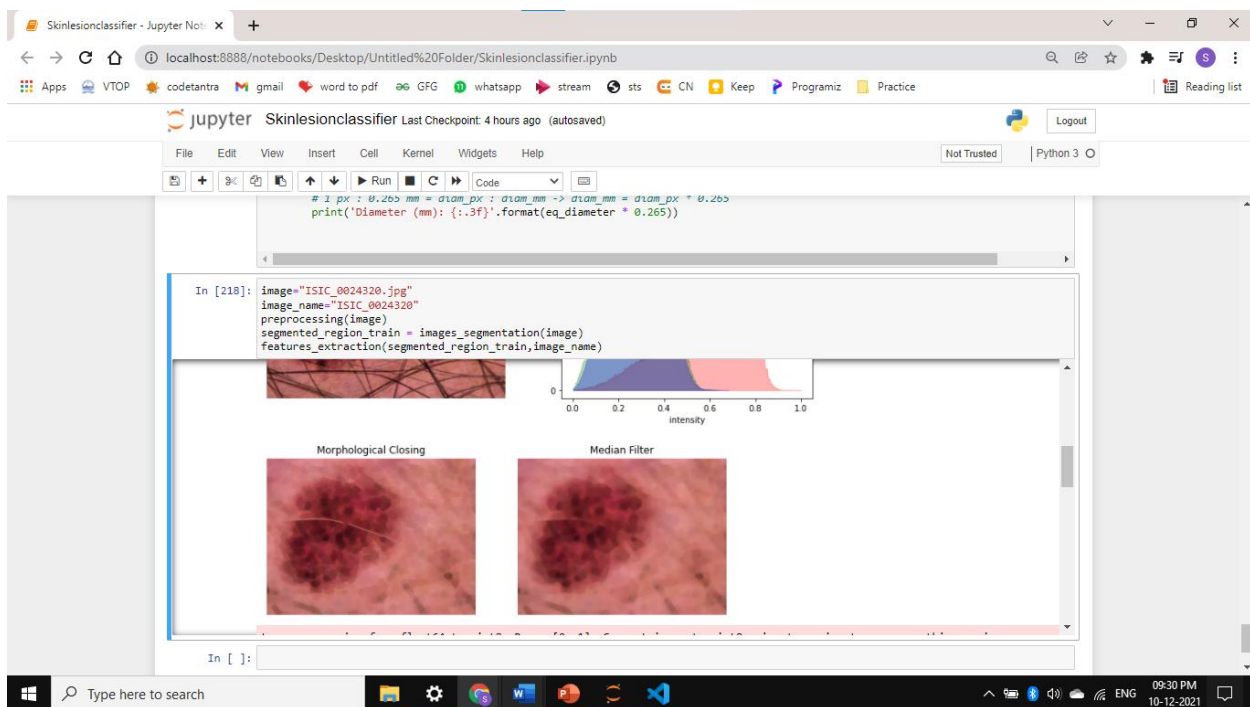
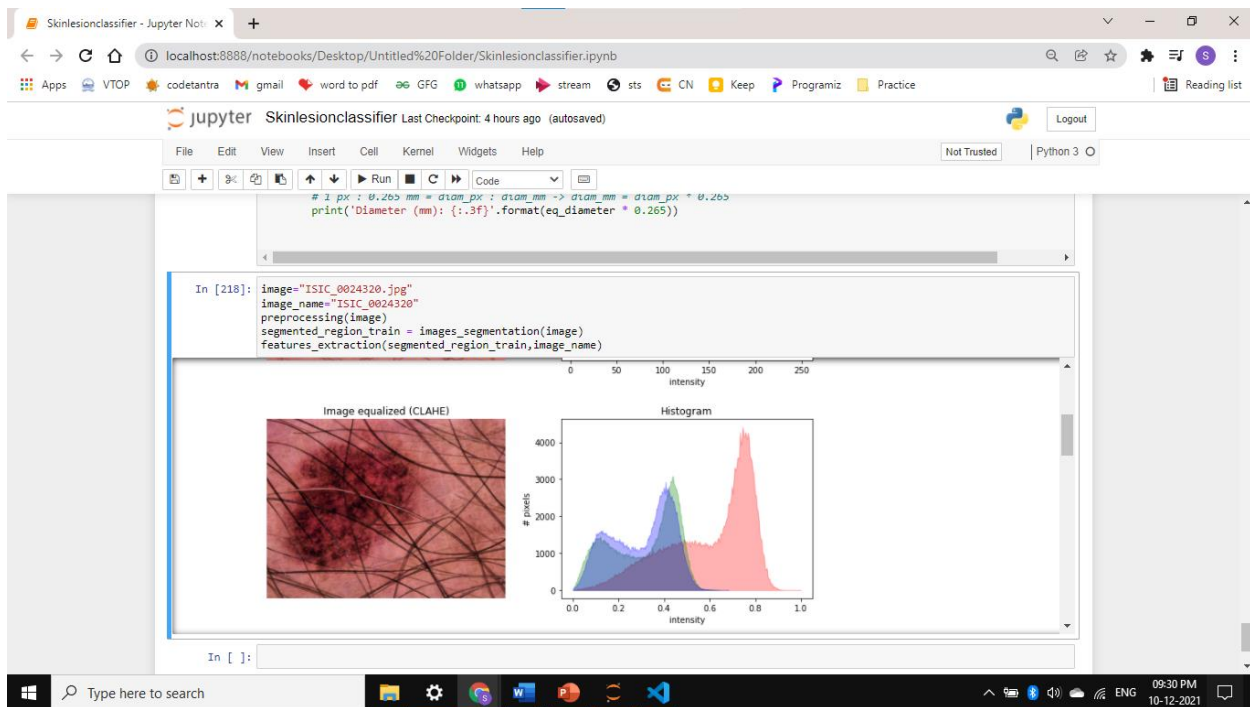
9. Sample Snippets





10. Output





Skinlesionclassifier - Jupyter Notebooks

localhost:8888/notebooks/Desktop/Untitled%20Folder/Skinlesionclassifier.ipynb

jupyter Skinlesionclassifier Last Checkpoint: 4 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

```
# 1 px : 0.265 mm = diam_px -> diam_mm = diam_px * 0.265
print('Diameter (mm): {:.3f}'.format(eq_diameter * 0.265))
```


In [218]:

```
image="ISIC_0024320.jpg"
image_name="ISIC_0024320"
preprocessing(image)
segmented_region_train = images_segmentation(image)
features_extraction(segmented_region_train, image_name)
```

s deprecated and will be removed in version 0.19. Use `skimage.segmentation.watershed` instead.

```
def watershed(image, markers=None, connectivity=1, offset=None, mask=None,
-- REGION-BASED SEGMENTATION --
```

elevation map markers segmentation



Num labels: 1
Areas: [75300]

In []:

Skinlesionclassifier - Jupyter Notebooks

localhost:8888/notebooks/Desktop/Untitled%20Folder/Skinlesionclassifier.ipynb

jupyter Skinlesionclassifier Last Checkpoint: 4 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

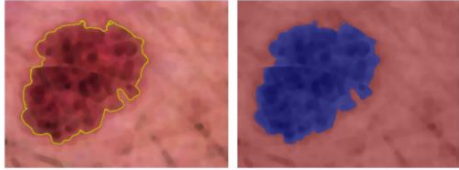
Not Trusted Python 3

```
# 1 px : 0.265 mm = diam_px -> diam_mm = diam_px * 0.265
print('Diameter (mm): {:.3f}'.format(eq_diameter * 0.265))
```

In [218]:

```
image="ISIC_0024320.jpg"
image_name="ISIC_0024320"
preprocessing(image)
segmented_region_train = images_segmentation(image)
features_extraction(segmented_region_train, image_name)
```

Chosen label: 1

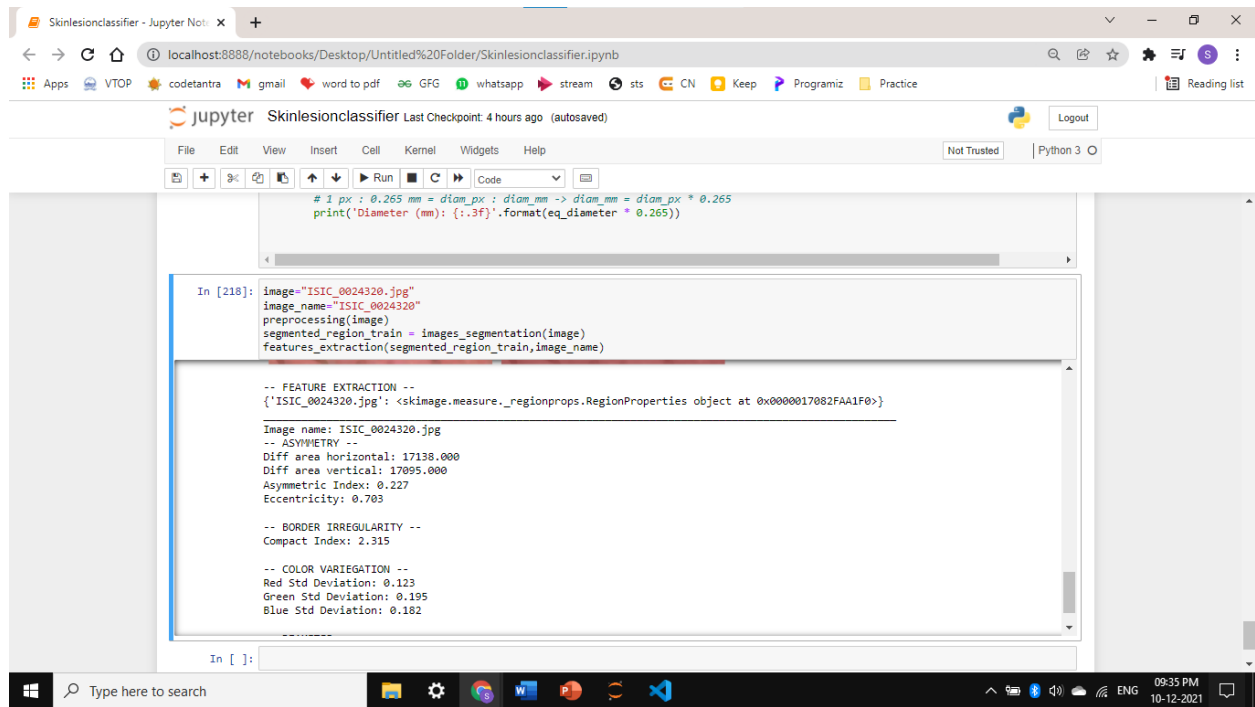


```
-- FEATURE EXTRACTION --
{'ISIC_0024320.jpg': <skimage.measure._regionprops.RegionProperties object at 0x0000017082FAA1F0>}
```

Image name: ISIC_0024320.jpg

In []:

After feature extraction for ABCD properties,



The screenshot shows a Jupyter Notebook titled 'Skinlesionclassifier' running on a local host. The code in the cell defines a function to calculate the diameter in mm from pixel dimensions and then performs feature extraction on an image named 'ISIC_0024320.jpg'. The output displays various morphological and color features.

```
# 1 px : 0.265 mm = diam_px : diam_mm -> diam_mm = diam_px * 0.265
print('Diameter (mm): {:.3f}'.format(eq_diameter * 0.265))

In [218]: image="ISIC_0024320.jpg"
          image_name="ISIC_0024320"
          preprocessing(image)
          segmented_region_train = images_segmentation(image)
          features_extraction(segmented_region_train,image_name)

-- FEATURE EXTRACTION --
{'ISIC_0024320.jpg': <skimage.measure._regionprops.RegionProperties object at 0x0000017082FAA1F0>}
```

```
Image name: ISIC_0024320.jpg
-- ASYMMETRY --
Diff area horizontal: 17138.000
Diff area vertical: 17095.000
Asymmetric Index: 0.227
Eccentricity: 0.703

-- BORDER IRREGULARITY --
Compact Index: 2.315

-- COLOR VARIATION --
Red Std Deviation: 0.123
Green Std Deviation: 0.195
Blue Std Deviation: 0.182
```

```
-- FEATURE EXTRACTION --
{'ISIC_0024320.jpg': <skimage.measure._regionprops.RegionProperties object at 0x00000170810822B0>}
```

```
Image name: ISIC_0024320.jpg
-- ASYMMETRY --
Diff area horizontal: 17138.000
Diff area vertical: 17095.000
Asymmetric Index: 0.227
Eccentricity: 0.703

-- BORDER IRREGULARITY --
Compact Index: 2.315

-- COLOR VARIATION --
Red Std Deviation: 0.123
Green Std Deviation: 0.195
Blue Std Deviation: 0.182

-- DIAMETER --
Equivalent diameter: 309.637
Diameter (mm): 82.054
```

11. Conclusion

We all know that skin cancer has multiplied to such an extent that it's very important to detect the disease at its initial stages. In order to solve this issue we have come up with a method to detect early signs of skin cancer due to raised concentration in certain parts of the skin. We have used a python code to detect the same and prove its efficiency.

Since the tool is made more feasible and robust for images acquired in any conditions, it can deliver the purpose of automatic diagnostics of Melanoma Skin Cancer. In the future, we could develop a computer algorithm for skin cancer diagnosis using Support Vector Machine, which is also an emerging technology nowadays.

In conclusion, this work aims to be a good starting point in developing advanced skin lesion classification systems capable of democratizing and distributing access to health care, in order to save many lives through early diagnosis and significantly reduce health costs.

12. Future Improvement

Segmentation does help us to detect cancer but it can be made much more accurate by improving its optimization results.

For that we can use Multiscale Optimization procedure Using better edge detection techniques we can preserve the object's topology and shape to even noisy images.

This algorithm does not depend on rigid threshold values thus very useful in unsupervised systems.

13. Learnings Through the Project

We learnt a lot of things while doing this project

- The creative nature of the project, which was typically self-selected by us and based on personal interests, strengthened our motivation to learn, particularly during a time when academic motivation and engagement tend to wane.
- It helped us gain more knowledge about our course that is IMAGE PROCESSING.

- The project typically required us to take on new responsibilities, be more self-directed, set goals, and follow through on commitments.
- Completing such a project boosted our self-esteem, built confidence, and taught us about the value of accomplishment.
- The project determined our proficiency (in the acquisition of knowledge and skills) or readiness (for college and work) by requiring us to demonstrate what we have learned throughout our project.
- It helped us with future planning, goal setting, postsecondary decisions, and career exploration.

14. References

[1] Danilo Barros Mendes and Nilton Correia da Silva. "Skin Lesions Classification Using Convolutional Neural Networks in Clinical Images". In: CoRR abs/1812.02316 (2018).

URL: <http://arxiv.org/abs/1812.02316>.

[2] M. Messadi, A. Bessaid, and A. Taleb-Ahmed. "Extraction of specific parameters for skin tumour classification". In: Journal of Medical Engineering & Technology 33.4 (2009), pp. 288-295.

[3] Zhishun She, Y. Liu, and Damatoa. "Combination of features from skin pattern and ABCD analysis for lesion classification". In: Skin Research and Technology 13.1 (2007), pp. 25-33.

[4] Noel C. F. Codella et al. "Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC)". In: CoRR abs/1902.03368 (2019).

URL: <http://arxiv.org/abs/1902.03366>.

[5] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. "The HAM10000 dataset, a large collection of multi-source dermoscopic images of common pigmented skin lesions". In: Scientific data 5 (2018), p. 180161.

THANK YOU