

# Detecting Skin Diseases Automatically Using Image Processing And Machine Learning

Joyce muchema

February 8, 2022

# 1 Milestone One

## 1.1 Understand the problem

Skin Diseases are a significant risk to human wellbeing. If skin diseases are not treated at earlier stage, they can cause complications inside the body in conjunction with spreading of the infection from one individual to the alternative. A system is needed to help skin specialist in getting better diagnosis and increase better treatment. There are some skin conditions that are a huge symptom of underlying health issues. Early detection of these disease can actually save thousands of lives. In poor areas, patients are not able to afford tests to determine what kind of a skin condition it is they are suffering from. These health practitioners need a cheap and a convenient way to detect all types of condition. Skin infections are conditions that affect your skin. These infections might cause rashes, aggravation, irritation or other skin changes. Some skin conditions might be hereditary, while way of life elements might cause others. Skin issues differ extraordinarily in manifestations and seriousness. They can be transitory or long-lasting, and might be effortless or difficult. Some have situational causes, while others might be hereditary. Some skin conditions are minor, and others can be dangerous. A patient can recuperate from skin illnesses in the event that it is distinguished furthermore treated in the beginning phases and this can accomplish fix proportions of more than 95%.Consequently, it is critical to recognize these sicknesses at their underlying stage to control them from spreading. Skin disease are set apart by their assortments and visibility. Dermatology is a field where pattern recognition and examination are key so experience is vital - having seen something beforehand makes it a lot simpler to remember it later on. Exact data taking and assessment are just about as significant as in some other field of medication. An efficient methodology is required, albeit this might become shortened with experience; nonetheless, even the most experienced specialist will have an intermittent troublesome situation where it is important to return to basics.

## 1.2 Formulating the solution

The solution for the problem mentioned above is to build a machine learning model that detects the type of a skin diseases based on its features. The system will have two main parts I.e. Image processing and machine learning multi-classification, in image processing filters are used on the image to increase the accuracy of the image, then machine learning will process

the data collected from the images. Below is a diagram that shows the implementation of the system.

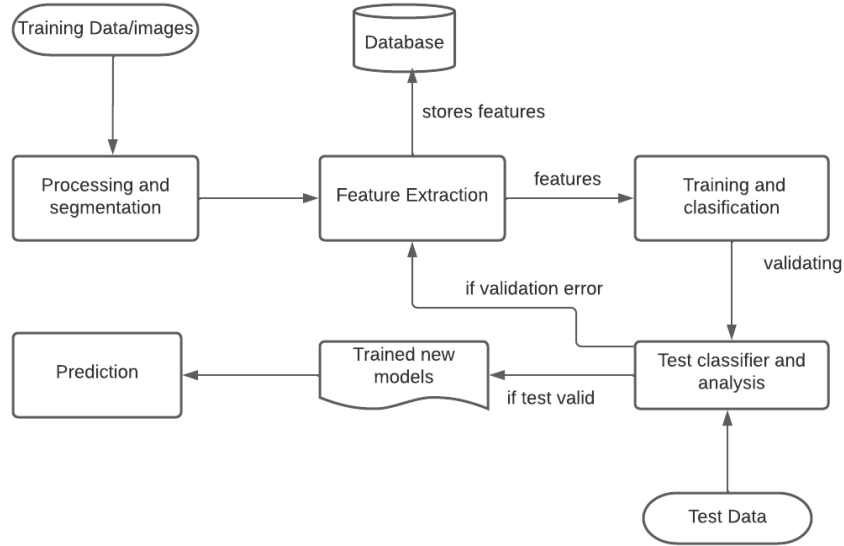


Figure 1: Flow chart showing the implementation of this project

## 1.3 Data And Attributes

### 1.3.1 Data

This project will use a public database for skin diseases from <https://www.kaggle.com/ismailpromus/skin-diseases-image-dataset>. The dataset contains over 10,000 images of skin diseases.

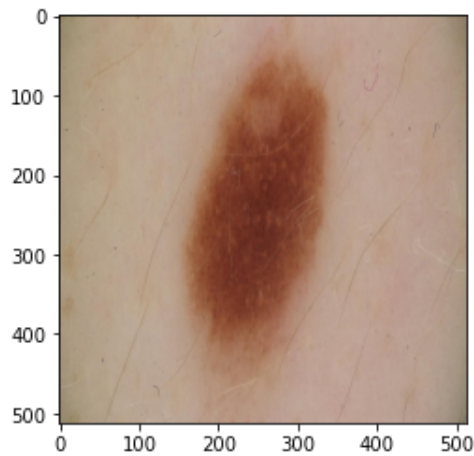
|   | Disease name                        | No_of_images |
|---|-------------------------------------|--------------|
| 0 | Atopic Dermatitis                   | 1257         |
| 1 | Basal Cell Carcinoma (BCC)          | 3323         |
| 2 | Benign Keratosis-like Lesions (BKL) | 2079         |
| 3 | Eczema                              | 1677         |
| 4 | Melanocytic Nevus (NV)              | 7970         |
| 5 | Melanoma                            | 3140         |
| 6 | Psoriasis pictures Lichen Planus    | 2055         |
| 7 | Seborrheic Keratoses                | 1847         |
| 8 | Tinea Ringworm Candidiasis          | 1702         |
| 9 | Warts Molluscum                     | 2103         |

## 1.4 Image Processing

### 1.4.1 Pre-processing

#### Read Image

In this stage, libraries are imported to store image dataset in a variable then create a function to load folders containing images into arrays. Below is a Sample image read from the dataset.



#### Remove noise

In this stage Median Filtering is utilized for limiting the impact of little constructions like thin hairs and it is effective at preserving edges in an

image after filtering out noise.

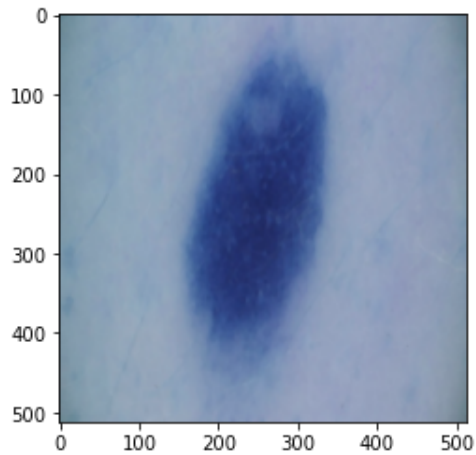


Figure 2: Filtered image

### 1.4.2 Segmentation

#### Threshold

Thresholding is a technique that identifies the required objects in an image. This technique replaces the pixels in the image with either black or white. If the intensity of a pixel is less than the threshold, then it is replaced the black and if is more, then the pixel is replaced with white.

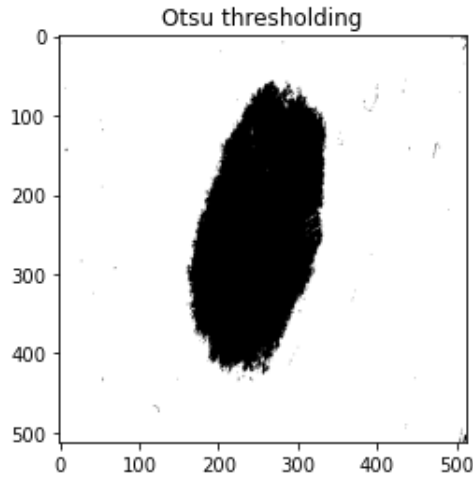


Figure 3: Thresholded image

### 1.4.3 Attributes

This project is centered around detecting/predicting types of skin diseases based on 10 attributes. These attributes are extracted from the images using techniques such as Gray-Level Co-occurrence Matrices (GLCMs). Attribute Information:

#### **Area**

Number of pixels of the region.

#### **Convex area**

The convex area of an object is the area of the convex hull that encloses the object. Convex area is by definition greater than or equal to the area of the region. It can be used to compute a shape factor known either as "convexity" or "solidity", defined as the ratio of area over convex area. It can be obtained by using the 'solidity' parameter in region props function. It is Number of pixels of convex hull image, which is the smallest convex polygon that encloses the region.

#### **Minor axis length**

The minor axis is the (x, y) endpoints of the longest line that can be drawn through the object whilst remaining perpendicular with the major-axis. The

length of the ellipse's minor axis has the same normalized second central moments as the region.

### **Perimeter**

The perimeter [length] is the number of pixels in the boundary of the object. Perimeter of object which approximates the contour as a line through the centers of border pixels using a 4-connectivity.

### **Equivalent diameter**

The equivalent diameter is defined as the diameter of a circle with an equal aggregate sectional area, which is calculated by  $d = 2 \text{ Area} / \text{pie}$ . The diameter of a circle with the same area as the region.

### **Mean intensity**

It measures the total intensity in an image by summing all of the pixel intensities (excluding masked pixels). This module will sum all pixel values to measure the total image intensity. The user can measure all pixels in the image or can restrict the measurement to pixels within objects. Value with the mean intensity in the region.

### **Solidity**

Solidity measures the density of an object. A measure of solidity can be obtained as the ratio of the area of an object to the area of convex hull of the object Ratio of pixels in the region to pixels of the convex hull image.

### **Eccentricity**

Eccentricity is the ratio of the length of the short (minor) axis to the length of the long (major) axis of an object: Eccentricity of the ellipse that has the same second-moments as the region

### **Box area**

The bounding box or bounding rectangle of an object is a rectangle which circumscribes the object. The dimensions of the bounding box are those of the major and minor axes. Number of pixels of bounding box.

### **Major axis length**

The major axis is the (x, y) endpoints of the longest line that can be drawn through the object. The length of the ellipse's major axis has the same normalized second central moments as the region.

The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and saving. The notebook title is "final project" and it shows the last checkpoint from 02/27/2022. The code in the cell is as follows:

```

In [32]: def get_attributes(name):
    properties = ['area', 'convex_area', 'bbox_area', 'major_axis_length', 'minor_axis_length',
                  'perimeter', 'equivalent_diameter', 'mean_intensity', 'solidity', 'eccentricity']
    dataframe = pd.DataFrame(columns=properties)
    # for file in filepaths:
    folder_path = 'C://Users//301CE//Desktop//Skin_Data//{}'.format(name)
    for root, dirs, files in os.walk(folder_path):
        for image_files in files:
            grayscale = rgb2gray(imread(os.path.join(root, image_files)))
            threshold = threshold_otsu(grayscale)
            binarized = grayscale < threshold
            closed = area_closing(binarized, 1000)
            opened = area_opening(closed, 1000)
            labeled = label(opened)
            regions = regionprops(labeled)
            data = pd.DataFrame(regionprops_table(labeled, grayscale, properties=properties))
            data = data[(data.index!=0) & (data.area>100)]
            dataframe = pd.concat([dataframe, data])
    return dataframe

Atopic_Dermatitis = get_attributes('Atopic Dermatitis')
Atopic_Dermatitis['class'] = 'Atopic Dermatitis'

Basal_Cell_Carcinoma = get_attributes('Basal Cell Carcinoma (BCC)')
Basal_Cell_Carcinoma['class'] = 'Basal Cell Carcinoma (BCC)'

Benign_Keratosis_like_Lesions = get_attributes('Benign Keratosis-like Lesions (BKL)')
Benign_Keratosis_like_Lesions['class'] = 'Benign Keratosis-like Lesions (BKL)'

Eczema = get_attributes('Eczema')
Eczema['class'] = 'Eczema'

Melanocytic_Nevi = get_attributes('Melanocytic Nevi (NV)')
Melanocytic_Nevi ['class'] = 'Melanocytic Nevi (NV)'

Melanoma = get_attributes('Melanoma')
Melanoma ['class'] = 'Melanoma'

```

## 2 Milestone Two

### 2.1 ML algorithm to be used

In this paper, SVM classifier will be used. SVM is a ML algorithm for data classification. The support vector machine (SVM) training is used for the optimization of a classification cost. The important advantage of SVM is that they provide a unified system in which different learning machine structures can be generated through an appropriate choice of kernel. Statistical and structural risk minimization is the principle used in SVM which minimizes the upper bound on the generalization error. The dataset is trained with 10,000 images belonging to ten classes and with the help of GLCM the texture and color features extracted from each of the ten classes are given as input to the SVM. The SVM will be divided into two different classes. The process involves two phases namely training phase and testing phase. In the



|    | A     | B           | C         | D                 | E                 | F           | G           | H              | I           | J            | K                             |
|----|-------|-------------|-----------|-------------------|-------------------|-------------|-------------|----------------|-------------|--------------|-------------------------------|
|    | area  | convex_area | bbox_area | major_axis_length | minor_axis_length | perimeter   | equivalent  | mean_intensity | solidity    | eccentricity | class                         |
| 1  | area  | convex_area | bbox_area | major_axis_length | minor_axis_length | perimeter   | equivalent  | mean_intensity | solidity    | eccentricity | class                         |
| 2  | 12726 | 17847       |           | 31302             | 249.4525221       | 90.22554661 | 742.6700936 | 127.291973     | 0.251911975 | 0.713061     | 0.932296765 Atopic Dermatitis |
| 3  | 1825  | 3938        |           | 7650              | 136.4008433       | 30.42174276 | 398.7289681 | 48.2043791     | 0.286603619 | 0.463433     | 0.974811141 Atopic Dermatitis |
| 4  | 1636  | 3644        |           | 6258              | 138.753188        | 32.72666447 | 376.9360749 | 45.6401128     | 0.266994213 | 0.448957     | 0.971786454 Atopic Dermatitis |
| 5  | 24103 | 32573       |           | 43512             | 250.9313798       | 166.0386835 | 1647.750396 | 175.182456     | 0.442290741 | 0.739969     | 0.749777885 Atopic Dermatitis |
| 6  | 4451  | 6163        |           | 6960              | 204.9797042       | 36.47800719 | 559.3208512 | 75.280736      | 0.406273011 | 0.722213     | 0.984037882 Atopic Dermatitis |
| 7  | 12243 | 16112       |           | 21312             | 235.7875182       | 76.15003466 | 980.9137803 | 124.853001     | 0.388626366 | 0.759868     | 0.946412474 Atopic Dermatitis |
| 8  | 27096 | 51435       |           | 65268             | 364.6951432       | 224.5011691 | 2281.006276 | 185.740945     | 0.465843606 | 0.526801     | 0.788069834 Atopic Dermatitis |
| 9  | 17365 | 27719       |           | 37518             | 232.7360438       | 141.4924331 | 1947.456024 | 148.693661     | 0.488017226 | 0.626466     | 0.793973486 Atopic Dermatitis |
| 10 | 4368  | 6652        |           | 11684             | 106.3496972       | 74.64694014 | 517.5340546 | 74.5755344     | 0.43048679  | 0.656645     | 0.712274689 Atopic Dermatitis |
| 11 | 1201  | 1424        |           | 2640              | 67.57126455       | 29.06943991 | 194.8050865 | 39.1044843     | 0.255311409 | 0.843399     | 0.902731616 Atopic Dermatitis |
| 12 | 7975  | 8883        |           | 13616             | 142.3373089       | 80.37865199 | 515.1259377 | 100.767482     | 0.107296587 | 0.897782     | 0.825292537 Atopic Dermatitis |
| 13 | 5814  | 8025        |           | 12528             | 144.091983        | 78.03159544 | 417.2670273 | 86.038449      | 0.39324198  | 0.724486     | 0.840674881 Atopic Dermatitis |
| 14 | 14944 | 18268       |           | 19992             | 352.400614        | 61.6256766  | 917.002092  | 137.93945      | 0.265523484 | 0.818042     | 0.984590835 Atopic Dermatitis |
| 15 | 1135  | 1849        |           | 3306              | 83.58809905       | 32.10283548 | 350.1726189 | 38.0148245     | 0.302809351 | 0.613845     | 0.923308186 Atopic Dermatitis |
| 16 | 9679  | 13209       |           | 25974             | 199.6980443       | 83.08111808 | 637.8416665 | 111.012096     | 0.300954756 | 0.732758     | 0.909349192 Atopic Dermatitis |
| 17 | 3134  | 3865        |           | 5004              | 139.7644033       | 32.39465141 | 426.155375  | 63.1690805     | 0.300298451 | 0.810867     | 0.972768132 Atopic Dermatitis |
| 18 | 1397  | 1582        |           | 1846              | 73.08389918       | 25.48585156 | 203.3969696 | 42.1748224     | 0.302261194 | 0.883059     | 0.937226779 Atopic Dermatitis |
| 19 | 1339  | 1588        |           | 2640              | 117.483015        | 18.35318956 | 272.6690476 | 41.2900442     | 0.280277272 | 0.843199     | 0.987722294 Atopic Dermatitis |
| 20 | 12398 | 27309       |           | 28416             | 305.0226169       | 111.7284194 | 1438.424494 | 125.640853     | 0.326408434 | 0.45399      | 0.93049857 Atopic Dermatitis  |
| 21 | 10102 | 15466       |           | 22815             | 191.8065248       | 97.48720602 | 848.1147904 | 113.41193      | 0.381533119 | 0.653175     | 0.86120484 Atopic Dermatitis  |
| 22 | 14537 | 18770       |           | 26040             | 191.8352459       | 119.3272065 | 1235.027525 | 136.048092     | 0.442765668 | 0.774481     | 0.782993706 Atopic Dermatitis |
| 23 | 21720 | 31764       |           | 40404             | 255.442028        | 132.4871927 | 1412.761543 | 166.297213     | 0.392794337 | 0.683793     | 0.854981465 Atopic Dermatitis |
| 24 | 11222 | 17051       |           | 18648             | 278.5906034       | 79.33198818 | 1435.110786 | 119.533653     | 0.394009876 | 0.658143     | 0.95859828 Atopic Dermatitis  |
| 25 | 10867 | 15762       |           | 27528             | 240.7329647       | 79.64056501 | 718.2264428 | 117.627778     | 0.254601858 | 0.689443     | 0.943692002 Atopic Dermatitis |
| 26 | 1241  | 2307        |           | 3895              | 96.21440002       | 25.81479992 | 315.527958  | 39.7503494     | 0.377469168 | 0.537928     | 0.96333403 Atopic Dermatitis  |
| 27 | 6478  | 13635       |           | 21312             | 270.3945978       | 62.82185049 | 1002.854906 | 90.8187523     | 0.403511135 | 0.475101     | 0.972636071 Atopic Dermatitis |
| 28 | 11515 | 24943       |           | 32412             | 311.8009227       | 114.7715558 | 1160.553391 | 121.084076     | 0.407162591 | 0.461653     | 0.929789208 Atopic Dermatitis |
| 29 | 1627  | 2004        |           | 2583              | 66.63955861       | 36.12107446 | 246.882251  | 45.5144014     | 0.468348415 | 0.811876     | 0.840354931 Atopic Dermatitis |
| 30 | 7219  | 14305       |           | 16206             | 298.9100169       | 63.23337293 | 1185.784884 | 95.8723958     | 0.408037425 | 0.504649     | 0.97736793 Atopic Dermatitis  |
| 31 | 8914  | 16501       |           | 17982             | 263.844579        | 66.04809208 | 1308.412301 | 106.53477      | 0.391978514 | 0.54021      | 0.968160669 Atopic Dermatitis |
| 32 | 18910 | 37372       |           | 42630             | 406.0200859       | 117.6830757 | 1480.234631 | 155.167522     | 0.363994285 | 0.505994     | 0.957073463 Atopic Dermatitis |
| 33 | 8419  | 20279       |           | 24642             | 320.2140551       | 94.76590758 | 1137.038672 | 103.534553     | 0.474062727 | 0.415159     | 0.955204828 Atopic Dermatitis |
| 34 | 13380 | 28605       |           | 29970             | 302.0887533       | 116.1660256 | 1158.162518 | 130.521819     | 0.500890776 | 0.46775      | 0.923107204 Atopic Dermatitis |
| 35 | 5651  | 7417        |           | 14018             | 160.2212912       | 63.30727971 | 554.4568901 | 84.8237978     | 0.153189917 | 0.761898     | 0.918627788 Atopic Dermatitis |
| 36 | 1096  | 1993        |           | 2704              | 113.3052999       | 18.67081081 | 287.5573952 | 37.3559974     | 0.348773939 | 0.549925     | 0.98632981 Atopic Dermatitis  |

Figure 4: Attributes table

training phase, the extracted features are fed to the classifier for training.in the testing phase, the unknown test pattern is fed and the knowledges ingined during the traing phase will classify the unknown pattern.

### 3 Milestone Three

#### 3.1 Tools used

The SVM algorithm will be implemented using Python programming language. AWS SageMaker services and Jupyter Notebook will be used for implementation purposes. required modules that will be instll include:

- sklearn
- pandas

- seaborn
- matplotlib
- numpy

```
#joyce muchema
# project implementation
import os
import pandas as pd
import numpy as np
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import cv2
import seaborn as sn
import openpyxl
import xlswriter
import csv
from skimage.io import imread, imshow
from skimage.color import rgb2gray
from skimage.measure import label, regionprops, regionprops_table
from skimage.filters import threshold_otsu
from skimage.morphology import area_closing, area_opening
from sklearn.model_selection import train_test_split
from skimage.filters import threshold_otsu
```

### 3.1.1 Sklearn

Scikit-Learn is a free AI library for Python. It upholds both administered and solo AI, giving different calculations to characterization, relapse, grouping, and dimensionality decrease. The library is constructed utilizing numerous libraries you may currently be acquainted with, like NumPy and SciPy. It additionally plays well with different libraries, like Pandas and Seaborn.

### 3.1.2 Pandas

Pandas is an open-source python bundle based on top of Numpy created by Wes McKinney. It is utilized as perhaps the main datum cleaning and investigation device. It gives quick, adaptable, and expressive information structures.

### 3.1.3 Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical

graphics.

### **3.1.4 Matplotlib**

Matplotlib is a low level diagram plotting library in python that fills in as a perception utility. Matplotlib was made by John D. Tracker. Matplotlib is open source and we can utilize it unreservedly. Matplotlib is generally written in python, a couple of fragments are written in C, Objective-C and JavaScript for Platform similarity.

### **3.1.5 Numpy**



Numpy, which represents Numerical Python, is a library comprising of complex cluster objects and an assortment of schedules for handling those exhibits. Utilizing Numpy, numerical and sensible procedure on clusters can be performed. This instructional exercise makes sense of the fundamentals of Numpy like its design and climate. It likewise talks about the different cluster capacities, sorts of ordering, and so forth.

## **4 Milestone Four**

### **4.1 Data processing**

#### **4.1.1 Missing Values**

Find missing values in a database is important since data insights and performance of machine learning algorithms could be impacted. To find the missing data the csv file ‘attributes.csv’ will be loaded and displayed.


final project
Last Checkpoint: 02/27/2022 (autosaved)
 Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

+

⌂

⌕

📄

⬆️

⬆️

▶️ Run

⏏️

🔄

⏏️

Code

⌵

📧

```

In [15]: #loading the datafile to find missing values
df = pd.read_csv('attributes.csv')
display(df)

```

|       | area  | convex_area | bbox_area | major_axis_length | minor_axis_length | perimeter   | equivalent_diameter | mean_intensity |
|-------|-------|-------------|-----------|-------------------|-------------------|-------------|---------------------|----------------|
| 0     | 12726 | 17847       | 31302     | 249.452522        | 90.225547         | 742.670094  | 127.291973          | 0.251912       |
| 1     | 1825  | 3938        | 7650      | 136.400843        | 30.421743         | 398.728968  | 48.204379           | 0.286604       |
| 2     | 1636  | 3644        | 6258      | 138.753188        | 32.726664         | 376.936075  | 45.640113           | 0.266994       |
| 3     | 24103 | 32573       | 43512     | 250.931380        | 166.038683        | 1647.750396 | 175.182456          | 0.442297       |
| 4     | 4451  | 6163        | 6960      | 204.979704        | 36.478007         | 559.320851  | 75.280736           | 0.406271       |
| ...   | ...   | ...         | ...       | ...               | ...               | ...         | ...                 | ...            |
| 26474 | 1046  | 1578        | 2223      | 55.259350         | 33.537532         | 265.195960  | 36.493952           | 0.411080       |
| 26475 | 9915  | 12657       | 17290     | 142.757406        | 103.520845        | 967.386868  | 112.357332          | 0.341821       |
| 26476 | 9915  | 12657       | 17290     | 142.757406        | 103.520845        | 967.386868  | 112.357332          | 0.341821       |
| 26477 | 1086  | 1192        | 1708      | 57.021174         | 24.643735         | 160.403066  | 37.185187           | 0.204781       |
| 26478 | 56859 | 91966       | 139200    | 389.338632        | 279.848786        | 6465.432175 | 269.063426          | 0.349431       |

26479 rows × 11 columns

<

>

The 'isnull ()' function will be used to find the missing values. If the field returns TRUE the field has missing values. If it returns FALSE then there are no missing values.

```
In [16]: #loading the datafile to find missing values
df = pd.read_csv('attributes.csv')
display(df.isnull())
```

|       | area  | convex_area | bbox_area | major_axis_length | minor_axis_length | perimeter | equivalent_diameter | mean_intensity |
|-------|-------|-------------|-----------|-------------------|-------------------|-----------|---------------------|----------------|
| 0     | False | False       | False     | False             | False             | False     | False               | False          |
| 1     | False | False       | False     | False             | False             | False     | False               | False          |
| 2     | False | False       | False     | False             | False             | False     | False               | False          |
| 3     | False | False       | False     | False             | False             | False     | False               | False          |
| 4     | False | False       | False     | False             | False             | False     | False               | False          |
| ...   | ...   | ...         | ...       | ...               | ...               | ...       | ...                 | ...            |
| 26474 | False | False       | False     | False             | False             | False     | False               | False          |
| 26475 | False | False       | False     | False             | False             | False     | False               | False          |
| 26476 | False | False       | False     | False             | False             | False     | False               | False          |
| 26477 | False | False       | False     | False             | False             | False     | False               | False          |
| 26478 | False | False       | False     | False             | False             | False     | False               | False          |

26479 rows × 11 columns

## 5 Milestone Five

### 5.1 Training. (Separate data using 80-20) use 80% to train

#### 5.1.1 Splitting

In this stage, our dataset will be split into training and testing parts to evaluate the models performance. 80% of the data has been assigned to the training part whereas the rest 20% to the testing part.

```
In [6]: #splitting and training the dataset
df = pd.read_csv('attributes.csv')

# input and outputs
x = df.drop('class', axis = 1)
y = df['class']

# Split dataset into training set and test set

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=100)
```

Figure 5: Splitting

### 5.1.2 Training

Training a network is a procedure of obtaining kernels in convolution layers and weights in fully connected layers that reduce differences on a training dataset between output predictions and specified ground truth labels. In our work, we used 80% of the data for training, through this stage so that the network that has been built learns by extracting features from skin disease images in order to learn from these features for each image to be distinguished on its basis. There two different conditions for training and testing. One is under the lab conditions, which means that the model is tested with the images from the same dataset from which it is used for both training and testing. The final accuracy obtained through this algorithm other condition is that field condition; this means that our model has tested with the images taken from the real world conditions. Since the lighting conditions and background properties of the images are totally different when we take samples from the real field, there is a chance that our model to produce a very low accuracy, when comparing to the accuracy values acquired during the lab conditions.

```
In [*]: #splitting and training the dataset
from sklearn.svm import SVC

df = pd.read_csv('attributes.csv')

# input and outputs
x = df.drop('class', axis = 1)
y = df['class']

# Split dataset into training set and test set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=100)

#used linear kernel to get the best accuracy

classifier1 = SVC(kernel='linear')

classifier1.fit(x_train, y_train)
```

Figure 6: Training

## 6 Milestone Six

### 6.1 Tests (20% of the data) and Evaluation Metrics (Accuracy, ROC Curve, Kappa, MSE)

#### 6.1.1 Testing

In this phase the dataset utilized to provide an impartial final design fit evaluation on the training set of data. In this stage, we use the groups that were trained in the previous step that was trained in SVM, and the features were extracted by learning the network when the data set passes from skin diseases on this network, we used 20% of the data for testing.

```
In [*]: #splitting and training the dataset
from sklearn.svm import SVC

df = pd.read_csv('attributes.csv')

# input and outputs
x = df.drop('class', axis = 1)
y = df['class']

# Split dataset into training set and test set

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=100)

#used linear kernel to get the best accuracy

classifier1 = SVC(kernel='linear')

classifier1.fit(x_train, y_train)

# testing the model
y_pred = classifier1.predict(X_test)
```

Figure 7: Testing

### 6.1.2 Evaluation Metrics

#### Accuracy

```
# printing the accuracy of the model  
print(accuracy_score(y_test, y_pred))
```

```
0.2177822177822178
```

Figure 8: Accuracy Score



## ROC Curve

ROC curve plots the true positive rate (Sensitivity) in function of the false positive rate for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold.

## Kappa

Kappa is used as a metric of comparison between Observed and Expected accuracy. Cohen's Kappa is a good way to measure the performance of a classifier.

```
In [39]: from sklearn.metrics import cohen_kappa_score
        cohen_score = cohen_kappa_score(y_test, y_pred)
        print(cohen_score)
0.13092415879218244
```

Figure 9: Kappa Cohen Score

## MSE

It returns the average of the sums of the square of each difference between the estimated value and the true value. The MSE is always positive, though it can be 0 if the predictions are completely accurate.

## 6.2 Discuss Results

The system proposed is a Skin Disease Detection System. This system uses images of skin captured with a camera to detect a particular type of disease.

```
In [42]: from sklearn.metrics import classification_report

# printing the report
print(classification_report(y_test, y_pred))
```

|                                     | precision | recall | f1-score | support |
|-------------------------------------|-----------|--------|----------|---------|
| Atopic Dermatitis                   | 0.29      | 0.39   | 0.33     | 100     |
| Basal Cell Carcinoma (BCC)          | 0.15      | 0.04   | 0.06     | 100     |
| Benign Keratosis-like Lesions (BKL) | 0.10      | 0.19   | 0.13     | 100     |
| Eczema                              | 0.24      | 0.17   | 0.20     | 100     |
| Melanocytic Nevi (NV)               | 0.26      | 0.39   | 0.31     | 100     |
| Melanoma                            | 0.28      | 0.28   | 0.28     | 100     |
| Psoriasis pictures Lichen Planus    | 0.30      | 0.13   | 0.18     | 101     |
| Seborrheic Keratoses                | 0.23      | 0.31   | 0.27     | 100     |
| Tinea Ringworm Candidiasis          | 0.18      | 0.21   | 0.20     | 100     |
| Warts Molluscum                     | 0.13      | 0.07   | 0.09     | 100     |
| accuracy                            |           |        | 0.22     | 1001    |
| macro avg                           | 0.22      | 0.22   | 0.21     | 1001    |
| weighted avg                        | 0.22      | 0.22   | 0.21     | 1001    |

Figure 10: Classification report

The SVM yielded a test accuracy of 22%.

## 7 Milestone Seven

### 7.1 Conclusion and Future work.

#### 7.1.1 Conclusion

The accurate Disease detection and classification of the human skin image is very important for the successful health development and this can be done using image processing. This paper discussed various techniques to segment the disease part of the skin. This paper discussed classification techniques to extract the features of infected skin and the classification of skin diseases through SVM classifier. This study is also highlighted to help the dermatologists in improving the diagnosis time and the accuracy of their intervention. Advantages of this proposed system is to reduce the human effort and automatic detect the diseases. Overall this work is implemented from scratch and produces a decent accuracy. In SVM computational complexity is reduced to quadratic optimization problem and it's easy to control complexity of decision rule and frequency of error. Drawback of SVM is it's difficult to determine optimal parameters when training data is not linearly separable. Also SVM is more complex to understand and implement.

### **7.1.2 Future work**

The Future scope in this proposed system is to increase the number of images present in the predefined database and to modify the architecture in accordance with the dataset for achieving better accuracy. This system can be improved to detect and classify more diseases as well as their severity. Also with the usage and the demand of the system we can expand the number of diseases which can be recognized by the system into considerable amount.