

1. INTRODUCTION

Skin cancer is the abnormal growth of cells on the skin. There are three major types of skin cancer basal cell carcinoma, squamous cell carcinoma and melanoma. If it is not detected early it will spread to other parts of the body, where it becomes hard to treat and may be fatal. It is chiefly diagnosed visually, initiating with a clinical screening followed by dermoscopic analysis, histopathological assessment, and a biopsy at times it might be too late ,with the development of artificial intelligence and machine learning capabilities, there is shining potential to spare time and mitigate with errors saving millions of lives in the long run.

1.1 Purpose of the System

The main purpose of the project is to make an early prediction of the severity of tumour. Skin cancer prediction is an algorithm for identifying the malignancy of the tumor. The skin cancer at its severe stage might have already spread to the different parts of the body making it extremely dangerous. Our aim is to build a machine that would help the patient in an early detection of the cancer . Due to the fine-grained differences in the appearance of skin lesions, automated classification is quite challenging through images. To attain highly segregated and potentially general tasks against the finely grained object categorized, images are processed and features are extracted using the ABCD (Asymmetry, Border irregularity, Colour intensity, Diameter).Then we use a classifier such as Support Vector Machine and train the system over the features using machine learning.

1.2 Scope of the System

The scope of the project is to find if the provided image of the skin lesion is cancerous or non- cancerous. If it is cancerous then we would classify it based on its intensity as mild, moderate and severe. This is done by processing the image data set to extract the features and then training the model to give accurate predictions.

1.3 Current Scenario

Yet, diagnosis is still a visual process, which relies on the long-winded procedure of clinical screenings, followed by dermoscopic analysis, and then a biopsy and finally a histopathological examination. This process easily takes time and the need for many medical professionals. The sheer amount of time and technicalities it takes when diagnosing the patient (let alone beginning their treatment) and the many opportunities for human error, leaves thousands dead annually.

1.4 Proposed System

Our key goal is to develop a model using the concepts of machine learning model that would help in predicting the type of the cancer and if the cancer is benign or malignant. We use gaussianBlur to minimise the noise and then we perform segmentation using Otsu's thresholding. We train the model using support vector machine classifier. This achieves performance on par with all the tested experts across both tasks, demonstrating an artificial intelligence capable of classifying the skin cancer with a level of competence compared to the dermatologists. A toolkit is developed using Tkinter in order to input the images and display the predictions along with necessary metrics.

2. LITERATURE SURVEY

2.1 Introduction

In the past few decades, malignant melanoma has been leading as the most common cancer in Australia, America and Europe . If detected early, this type of cancer can be cured with a success rate of over 92% . A biopsy is usually performed to determine if a tumour is malignant or benign. However as this laboratory medical procedure involves a high cost and morbidity, an equally fast and convenient screening technique is automatic early detection .

Researchers in the field of dermatology imaging believe that melanoma diagnosis can be computerized building on specific physical features and colour details that are characteristic of skin cancer types . Major melanoma factors – both diagnostic and prognostic lesion colour, 3D shape and size and vertical thickness.

There have been many attempts by researchers at building an automated skin cancer detection system to improve the accuracy of diagnosis. It is important to study the path previously taken by these researchers to gain enough knowledge so as to try and achieve a reliable skin cancer detection system. The following literatures review these attempts.

2.2 Computer-aided diagnosis system

A mechanized detection procedure for skin cancer can bring down the false-positive or false-negative clinical diagnosis since it adds a quantitative observation to the regular human-eye observation. Early detection of skin lesion from cancer images is divided into four stages - preprocessing, segmentation, feature extraction and classification.

The skill of the dermatologists is also critical to achieve accurate diagnostic performance considering dermoscopy images. Considering the varied type of melanoma, non-melanoma skin lesions and dependency on the skill level of dermatologist, accurate diagnosis of melanoma is still a problem. The use of computer aided diagnosis can be used to tackle this problem.

Availability of advance image processing techniques and decision making mechanisms to build computer aided diagnostic system can provide a wholistic solutions to aid early diagnosis of skin cancer melanoma. The computer aided diagnostic systems are also referred to as “Computerized dermoscopy” .

2.2.1 Image acquisition/ methods for screening skin lesions

Visual inspection is a common clinical diagnosis in melanoma detection which may however, involve some error . There are different techniques which help dermatologists visualize morphological features that are not detectable by the naked eye. These include dermoscopy, solar scan , epiluminescence microscopy (ELM) , cross-polarization epiluminescence(XLM), and side transillumination (TLM) .

2.2.2 Pre-Processing

Any image considered for the purpose of cancer detection has to go through a pre-processing stage to mainly rid it of noise. It is mandatory to study and separate abnormalities in the background on the result and by noise, we mean parts of the image that are not required for detection purposes. Any redundant parts in the background will also be cut off, helping improve the image for the next process by retaining only the required information. Using a good pre-processing technique means better accuracy [95]. The pre-processing stage involves three processes, that of image enhancement, image restoration and hair removal.

Color Space Transformation

Detection systems use colour information as a vital component in demarcating one stage of skin cancer from another or to even differentiate between healthy and diseased. For the same reason scientists have worked on extracting the closest colour from images for further processing. Common colour representation models include RGB, HSV, HSI, CIELAB and CIEXYZ. The RGB colour code is frequently used for processing images and represents the three primary spectral colours red, green and blue. But RGB colour space comes with limitations in high-level processing leading to the use of other colour space representations.

HSV and HSI colour models perceives in a similar manner as that of the human eye when it comes to hue, saturation and intensity.

Since images used in skin cancer detection systems are required to show high level intensity variations in colour for proper detection of lesion edges, it would be optimal to use LAB colour space commonly represented as „Lab“ since it encompasses the entire range of colours. „L“, here, stands for lightness or colour space's brightness, „a“ is the axis along which red/green opponent colours are denoted while „b“ axis represents the yellow/blue opponent colours. This thesis applies Lab for image transformation from RGB using XYZ as an inbetween colour space. The main image is converted to a greyscale image which provides the values for Lightness (from „0“ to „100“) .

Noise Reduction

Various defects on an image can degrade it to a low quality one which cannot be used for disease detection. Hence, Image Restoration procedure is necessary to increase the quality of a degraded image by removing noise and blur among others .Imperfections commonly found on an image can be caused due to a bad imaging system, bad focusing or even movement consequently resulting in an image which is blurry. This process helps restore a bad image in more than one way. To apply the most appropriate algorithm for the purpose of de-noising, it is crucial to study the different noises that may be present as part of the image. Samples of four different noises simulated in Matlab are shown in figure 2.4. They are named Gaussian, Salt and Pepper, Poisson and Speckle .

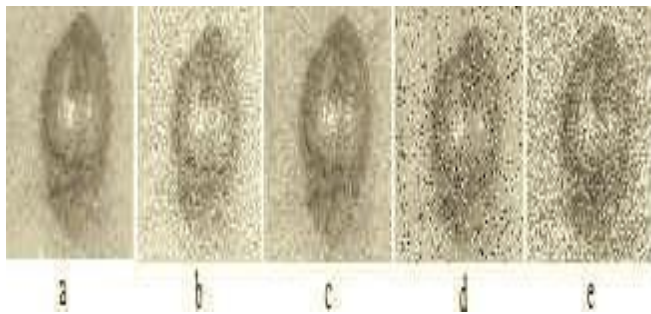


Figure 2.4. a) Image without noise b) Gaussian noise c) Poisson noise d) Salt and Pepper noise e) Speckle noise[107]

2.2.3 Segmentation

A major challenge for research and development in this field includes segmentation of skin cancer images. This process is essential in digital image processing as it is ultimately used for image description and classification. Various properties like shape, brightness, colour and/or texture are applied to assist in the segmentation of skin lesion. In the past few decades, many algorithms have been proposed for the detection of lesions in skin cancer images. Celebi et al]categorized the segmentation methods into:

- a. Histogram thresholding which separates the area of interest (ROI) and background using one or more threshold values
- b. Region-based methods which incorporate the pixels into their similar regions using region-splitting and region-merging algorithms.
- c. Edge-based methods in which the edges of lesions are determined using edge operators
- d. Active-contour methods in which the contours in the shape are made to evolve using curve evolution techniques .
- e. Morphological methods determine the seeds and employ watershed transform to identify contours of an object .
- f. Colour-clustering methods employ unsupervised clustering algorithms to generate homogeneous areas by separating the colour space .
- g. Soft-computing methods employ different soft-computing techniques to classify the pixels.
- h. Model-based methods, in which the image is considered as a random field and the model is parameterized using optimisation methods.

In segmentation, sets of objects with similar characteristics are classified into different groups. This process called clustering has been widely applied in many areas such as image processing, machine learning, pattern recognition, data mining and statistics. Recently, clustering algorithms found crucial applications in medical imaging field .

2.2.4 Feature Extraction

The process of feature extraction involves extraction of image parameters to characterize the dermatological features of melanoma and diagnosis based on these parameters. Medical experts rely on the features of melanoma. The method of diagnosis applied is important for feature selection. For instance, asymmetry and pigmented network are respectively the features in ABCD-rule and pattern analysis.

The features evaluation of melanoma diagnosis is visually very difficult because the content of information in dermoscopic images is very complicated and requires experienced physicians for analysis. The diagnosis methods by non-dermatologists to determine melanoma lesions in the screening process are listed as ABCD rule ABCD-E criteria [143] and Glasgow 7-point checklist .

The ABCD rule of dermoscopy consists of four criteria: Asymmetry, Border sharpness, Colour variegation and Differential structures. ABCDE has the added factor of E or Evolving. The 7-point checklist consists of seven criteria: Atypical pigment network, Bluewhitish veil, Atypical vascular pattern, Irregular streaks, Irregular dots/globules, Irregular blotches and Regression structures. Pattern analysis consists of Global patterns and Local .

According to , symmetry achieved the highest weight in the ABCD rule of dermoscopy. Stolz et al. [15] indicated that 96% of asymmetry in melanoma cases had score 2 (both axes represent asymmetry) while it was just about 24.2% in benign images. Many researches considered asymmetry according to the axis of symmetry in the tumour. In such studies, the axis of symmetry may be identified using Fourier transform , best-fit ellipse , diameter length and principal axis . Post that, both created areas by the axes are differentiated. In many studies, the roundness, compactness and thinness of lesion have been considered as appropriate properties of skin cancer images and in , they have been considered as accurate geometry variables. In , the symmetry distance (SD) was introduced as another measure in images. Seidenari et al. presented a method to estimate distribution in skin lesions. Their purpose was to determine the effectiveness of distribution parameters to identify melanoma

from the normal ones. They found out about the non-homogeneity of lesion region; they computed the mathematical parameters such as mean, variance, and Euclidean distance.

Manousaki et al. proposed estimation of distribution irregularity using the fractal dimension in the surface of the lesion. Also, they computed the standard deviation to measure the sharpness of borders. Lee et al. , presented an algorithm to search a convex and curvature maxima's locations in an image. In the standard deviation and mean are calculated in six colour spaces. In another approach the different statistical properties of standard deviation, energy, mean and entropy are computed as extracted features. The Neural Network has been trained using these features and the accuracy of 79% was achieved.

The significance of color features to classify skin lesions is put forth in]. A Kmeans clustering algorithm is incorporated to extract the color features. The Congenital nevi, combined nevi, Reed/Spitz nevi, melanomas, dysplastic nevi, blue nevi, dermal nevi, seborrheic keratosis and dermatofibroma lesion images are considered for evaluation. Using a symbolic regression algorithm the skin lesion are classified into benign or malignant types.

In [258] the authors consider that each dermoscopic image represents a Markov model. The parameters estimated from the model are considered as the features of the skin lesion. Classification is performed to identify the globular, reticular and homogeneous patterns in the pigmented cell.

The Gray Level Cooccurrence Matrix (GLCM) as another popular method to extract the image features has been employed by different researchers in various applications]. Many other researches have been reported on feature extraction of skin cancer in literature .

2.2.5 Classification

The final step in computerized analysis of melanoma detection, to estimate if a lesion is malignant or benign, is Lesion classification. To carry out the classification task, existing systems utilize different classification methods with feature descriptors that were extracted in the previous stage. The efficiency of these methods appertains to both extracted descriptors and selected classifier . There are different classifiers that currently exist namely

Discriminant Analysis, Artificial Neural Network, K-Nearest Neighbourhood, Support Vector Machine (SVM), Decision Trees and Self-Advising SVM.

In Different researches , Discriminant analysis was applied as a classifier to make predefined classes from a set of observations. It works by the values of determined measurements which are called predictors. Artificial Neural Networks (ANN) is another tool that was employed in . This approach connects the inputs and outputs for detecting the patterns in data. It is usually employed in classification problems. In another algorithm called knearest-neighbourhood (K-NN) was considered for distinguishing lesions as melanoma or benign. This classifier employs distance measure like Euclidean distance to evaluate the distribution of data and classify the objects according to their closeness to the training set. SVM displayed a powerful ability of solving problems of nonlinear classification in many applications (high dimension as well). Furthermore, SVMs prevented over-fitting by selecting a particular hyper-plane among many by separating the data in feature space . SVM has been used as a popular technique in for classifying skin cancer melanoma. In some other researche, Decision trees separated the data set into different groups according to its disparity and made the classification schema. Based on high-level intuitive features (HLIF) and SVM classifiers the diagnosis of melanomas and non-melanoma skin lesions is presented in . In addition to the HILF features, low-level features and their combinations are also considered. Dreiseitl et al. compared the different classification techniques of artificial neural network, k- nearest neighbourhood, support vector machine, logistic regression .

3 SYSTEM REQUIREMENTS

3.1 Functional Requirements

Functional requirements describe what the system should do, i.e. the services provided for the users and for other systems.

3.1.1 Input

- User takes input image.

3.1.2 Output

Display the prediction made by the model.

Display the metrics such as the several images and the features.

3.2 Modules

After a deep analysis of the system requirements the project has been developed by getting divided into the following modules. Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

3.3 Non-functional requirements

3.3.1 User interface and human factors

The Tkinter user interface is providing to do interaction with the user and the machine learning model to deploy the prediction of the input image.

The following human factors need to be considered:

- What users want and accepted.

- Users want full coverage in the target area.
- Technical characteristics and limitations of the computer hardware and software must also be considered.

3.3.2 Software Requirements

- Operating Systems: Windows 10/ 8/XP
- Programming Language: Python

3.3.3 Hardware Requirements

- System : Pentium IV 2.4 GHz.
- Hard Disk : 2TB.
- Floppy Drive : 1.44 Mb.
- Monitor : 14' Colour Monitor.
- Mouse : Optical Mouse.
- Ram : 4 GB Ram.
- Keyboard : 101 Keyboards.

4.SYSTEM DESIGN

4.1System Architecture

- A System architecture is a conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structure and behavior of the system.
- A representation of a system, including a mapping of functionally onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components.

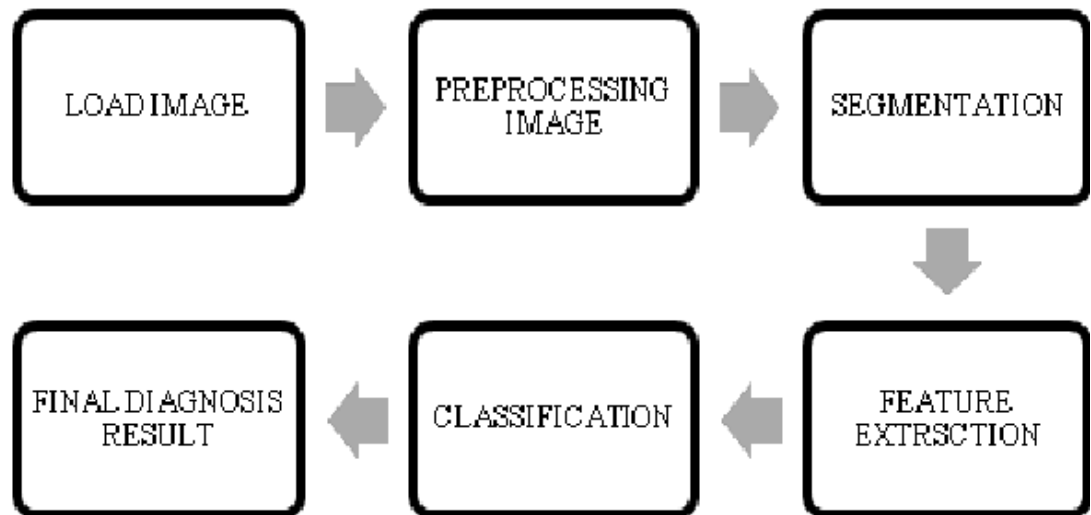


Fig 4.1.1 System Architecture

4.1.1 SYSTEM DESIGN INTRODUCTION:

The Unified Modeling Language (UML) allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. The Unified Modeling Language is commonly used to visualize and construct systems which are software intensive. Because software has become much more complex in recent years, developers are finding it more challenging to build complex applications within short time periods. Even when they do, these software applications are often filled with bugs, and it can take programmers weeks to find and fix them. This is time that has been wasted, since an approach could have been used which would have reduced the number of bugs before the application was completed.

Since UML is not a methodology, it does not require any formal work products. Yet it does provide several types of diagrams that, when used within a given methodology, increase the ease of understanding an application under development. There is more to UML than these diagrams, but for my purposes here, the diagrams offer a good introduction to the language and the principles behind its use. By placing standard UML diagrams in your methodology's work products, you make it easier for UML-proficient people to join your project and quickly become productive.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

- User Model View
- Structural model view
- Behavioral Model View
- Implementation Model View
- Environmental Model View

4.2 UML DIAGRAMS

Use Case Diagram

Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the Analysis phase of software development to articulate the high-level requirements of the system. The primary goals of Use Case diagrams include:

1. Providing a high-level view of what the system does.
2. Identifying the users ("actors") of the system.
3. Determining areas needing human-computer interfaces.

Class Diagram

Class diagrams identify the class structure of a system, including the properties and methods of each class. Also depicted are the various relationships that can exist between classes, such as an inheritance relationship. The Class diagram is one of the most widely used diagrams from the UML specification. Part of the popularity of Class diagrams stems from the fact that many CASE tools, such as Rational XDE, will auto-generate code in a variety of languages. These tools can synchronize models and code, reducing your workload, and can also generate Class diagrams from object-oriented code, for those "code-then-design" maintenance projects.

Sequence Diagram

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. Because UML is designed for object-oriented programming, these communications between classes are known as messages. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time.

Collaboration Diagram

Collaboration diagram displays an interaction organized around the objects and their links to one another. This type of diagram is a cross between an object diagram and a

sequence diagram. Unlike the Sequence diagram, which models the interaction in a column and row type format, the Collaboration diagram uses the free-form arrangement of objects as found in an Object diagram. This makes it easier to see all interactions involving a particular object. Numbers are used to show the sequence of messages. State diagram displays the sequences of states that an object of an interaction goes through during its life in response to received stimuli, Together with its responses and actions.

Activity Diagram

Activity diagram displays a special state diagram where most of the states are action states and most of the transitions are triggered by completion of the actions in the source states. This diagram focuses on flows driven by internal processing

Component Diagram

Component diagram displays the high level packaged structure of the code itself. Dependencies among components are shown, including source code components, binary code components, and executable components. Some components exist at compile time, at link time, at run times well as at more than one time.more capable of avoiding the black hole. Clearly, the random propagation phase is the key component that dictates the security and energy performance of the entire mechanism.

4.2.1 Use Case Diagram

Definition:

A use case diagram is a set of use cases, actors and relationships between actors and use cases. A use case is represented using oval shape.



4.2.1. List of Use cases

- Input image
- Processing input image
- Pre processing using Gaussian Blur
- Segmentaon(Otsu's Thresholding)
- Feature extracon
- Classificaon(SVM)
- Prediction

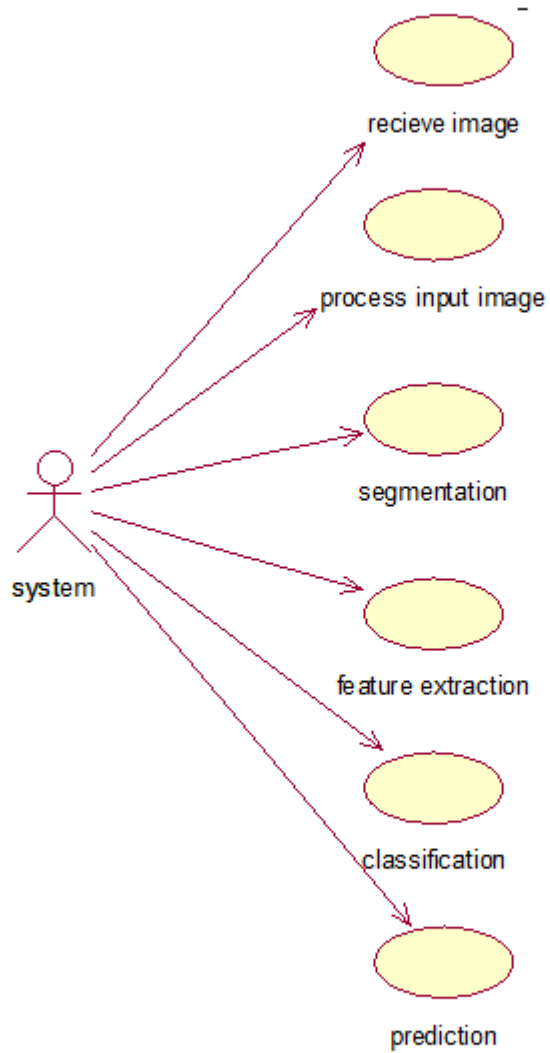


Fig: 4.2.1 Use Case Diagram

4.2.2 Class Diagrams

It shows a set of classes, interfaces, collaborations & their relationships. These address the static design view of a system. Class diagrams that include active classes address the static process view of a system. These diagrams are the most common diagram found in modelling object-oriented systems

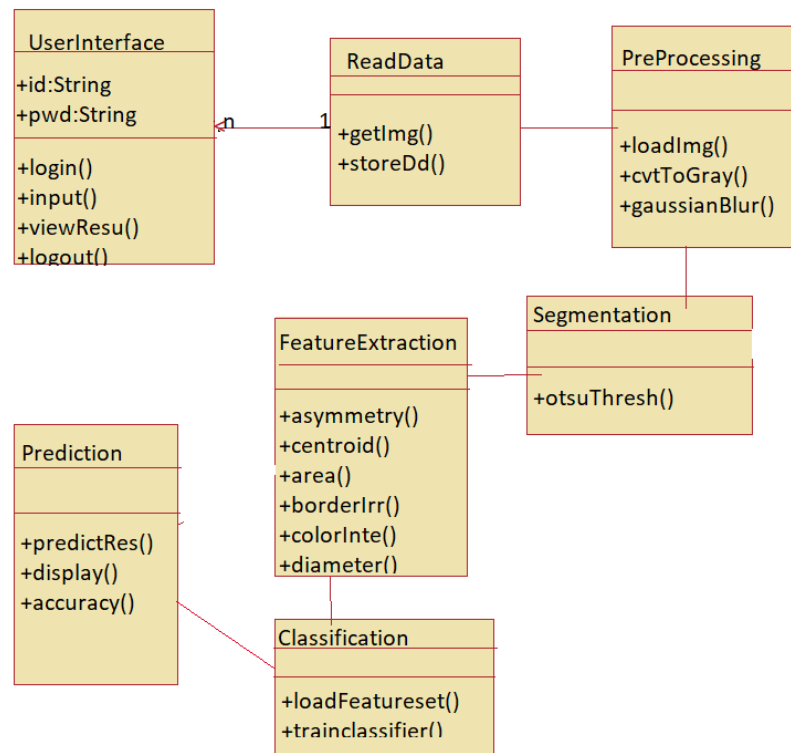


Fig 4.2.2 Class diagram

4.2.3 Collaboration Diagrams

A collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Graphically, collaboration diagrams are a collection of vertices and arcs. It emphasizes the organization of the objects that participate in the interaction. A collaboration diagram shows the relationship between the object and order of messages passed between them. The objects

are listed as icons and arrows indicate the message being between them. They are many acceptable sequence numbering schemes in UML. The purpose of collaboration diagram is similar to sequence diagram. But the specific purpose of collaboration diagram is to visualize the organization of objects and their interaction.

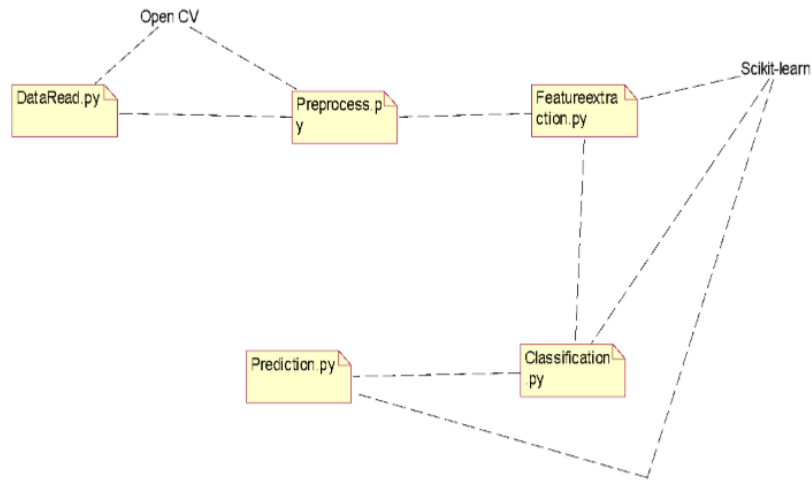


Fig 4.2.3 Collaboration Diagram

4.2.4 Sequence diagrams

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. A sequence diagram shows a set of objects and the messages sent and receives by those objects. The objects are typically named or anonymous instances of classes, but may also represent instances of other things, such as collaborations, components, and node.

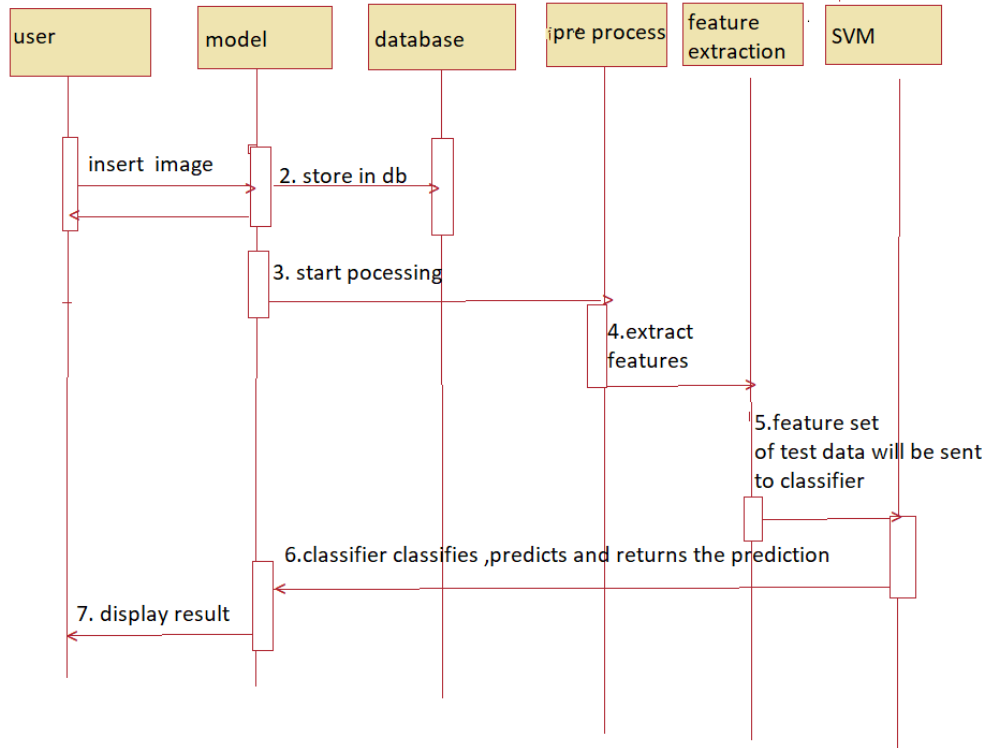


Fig 4.2.4 Sequence diagram for Sensors

4.2.5 Activity diagrams:

Activity diagrams are one of the five diagrams in the UML for modelling the dynamic aspects of the systems. An activity diagram is essentially a flow chart, showing the flow of control from activity to activity. We use activity diagrams to model the dynamic aspects of the system. For the most part, this involves modelling the sequential (and possibly concurrent) steps in a computational process. With an activity diagram, you can also model the flow of an object as it moves from state to state at different points in the flow of control. Activity diagrams may stand alone to visualize, specify, construct and document the dynamics of a society of objects, or they may be used to model the flow of control.

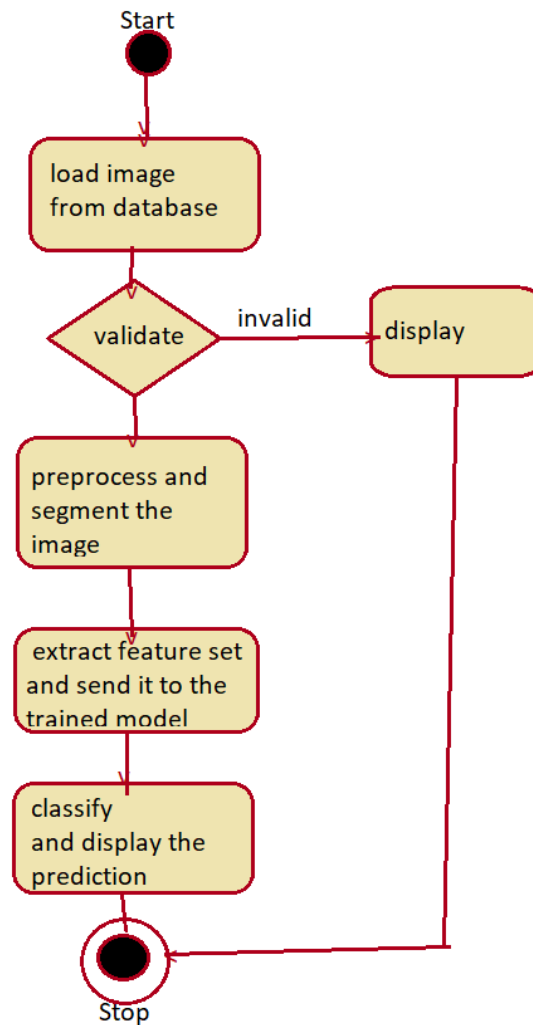


Fig 4.2.5 Activity Diagram

4.2.6 State Chart Diagram

A state chart diagram describes a state machine. Now to clarify it state machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Prediction of Skin Cancer Using Machine Learning

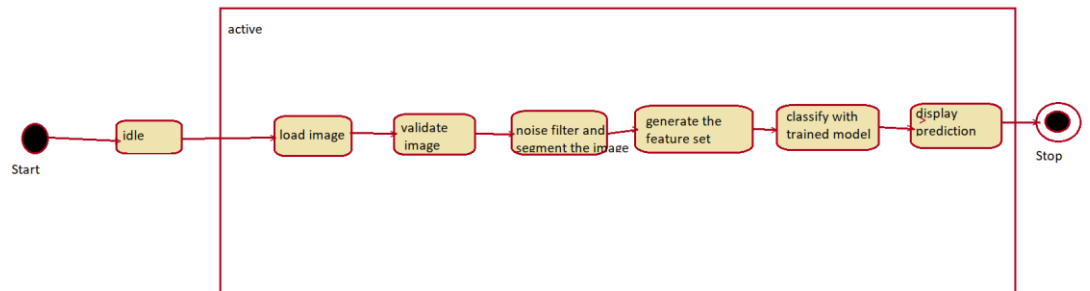


Fig 4.2.6 State Chart Diagram

4.2.7. Deployment Diagram

A deployment diagram shows the configuration of processing nodes and the components that live on. Deployment diagram addresses the stack deployment view of an architecture. They are related to component diagram in that a node typically encloses one or more components.

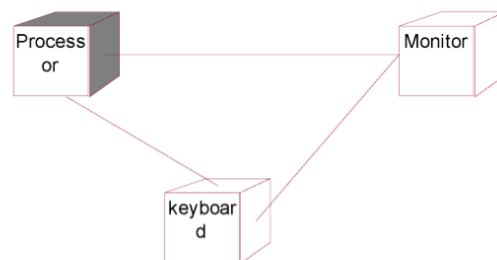


Fig 4.2.7 Deployment Diagram

4.3 working Principle

Prediction of skin cancer using machine learning is a dynamic system which predicts the stage of the lesion based on the image.

The flow of the system is

1. Take an input image.
2. Pre-process the input image by removing the noise using the Gaussian Blur technique.
3. Find the boundary and the effected area of the tumor by using the Otsu's Segmentation algorithm.
4. Extract the required features such as area, perimeter, centroid, color intensities, Asymmetry.
5. Predict the result of the given input image by providing the extracted features to the Trained Model.
6. Also calculate the accuracy of the system.
7. These analytics will be displayed in the User-interface.

5. SYSTEM IMPLEMENTATION

5.1 Image Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools;
- Analysing and manipulating the image;
- Output in which result can be altered image or report that is based on image analysis.

We performed the image processing using the OpenCV python. OpenCV is a Python open-source library. In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and video.

The system consists of three main components:

- 1) Image Preprocessing
- 2) Image Segmentation
- 3) Feature Extraction.

The system should be able to read the input image and perform the proper noise reduction pre processing and segmentation in order to have clear and accurate lesion. Also it should extract the features from the segmented output image. The features are consisting asymmetry, border, diameter and color of lesion.

5.1.1 Image Pre-processing: Pre process the image using the necessary noise reduction algorithms to minimize the noise such as Gaussian blur.

GaussianBlur(src, dst, ksize, sigmaX)

- **src** – A **Mat** object representing the source (input image) for this operation.
- **dst** – A **Mat** object representing the destination (output image) for this operation.
- **ksize** – A **Size** object representing the size of the kernel.
- **sigmaX** – A variable of the type double representing the Gaussian kernel standard deviation in X direction.

5.1.2 Image Segmentation: Image segmentation is the first step in the early detection of lesion. With image segmentation, the lesions can be detected and separated accurately and precisely. In proposed system image is segmented using the OTSU's method. This method is fully unsupervised method. It does not involve any user interaction.

OTSU's method is finest method for thresholding the objects (lesions) from the background skin. In this method the image is classified into two classes. Image with L gray levels is given as input. This method divides the image into two classes $C_0 = \{0, 1, 2, \dots, t\}$ and $C_1 = \{t+1, t+2, t+3, \dots, L-1\}$. The probability of occurrence of gray level i is given by:

$$p_i = \frac{n_i}{n}$$

P_i = Probability of occurrence of gray level i. n_i = Number of pixels in gray level i.
 n = Total number of pixels in an input image.

These classes C_0 and C_1 represent the object of interest (lesion) and the background. The probability of these classes W_0 and W_1 are given by

$$W_0 = \sum_{i=0}^t p_i \text{ and } W_1 = \sum_{i=t+1}^{L-1} p_i$$

Thus, the mean of the two classes can be computed as:

$$\mu_0(t) = \sum_{i=0}^t \frac{ip_i}{W_0(t) \cdot \mu_1(t)} \text{ and } \mu_1(t) = \sum_{i=t+1}^{L-1} \frac{ip_i}{W_1(t)}$$

An optimal threshold t^* can be obtained by maximizing the class variance:

$$t^* = \text{Arg} \left\{ 0 \leq i^{Max} \leq L - 1 \frac{\sigma_B^2}{\sigma_T^2} \right\}$$

Where, Class Variance is

$$\sigma_B^2 = W_0(\mu_0(t) - \mu_1(t))^2 + W_1(\mu_1(t) - \mu_1(T))^2$$

Total Variance is

$$\sigma_T^2 = \sum_{i=0}^{L-1} (i - \mu_T)^2$$

Total Mean is

$$\mu_T = \sum_{i=0}^{L-1} ip_i$$

Thus by putting all these values in t^* equation which results in

$$t^* = \text{Arg} \left\{ \underset{0 \leq i \leq L-1}{\text{Max}} W_0(\mu_0(t) - \mu_1(t))^2 + W_1(\mu_1(t) - \mu_T(t))^2 \right\}$$

5.1.3 Feature Extraction:

The mole have features like Asymmetry, border irregularity, colour and diameter these are ABCD features of melanoma skin cancer lesions.

Three features are extracted from lesion image. The features are Asymmetric(A), colour(C) and Diameter(D).

Prediction of Skin Cancer Using Machine Learning

Asymmetry: To assess asymmetry, the melanocytic lesion is bisected by two 90° axes that were positioned to produce the lowest possible asymmetry score. If both axes dermoscopically show asymmetric contours with regard to shape, colours and/or dermoscopic structures. Store the horizontal and the vertical asymmetricities.

Colour: Extracted different hue saturation values of the darkest shades of browns present in the lesion . HSV values of black, dark browns, light brown, whites are used to compare and get the maximum along with the maximum RGB values.

Diameter: Melanoma tends to grow larger than common moles, and especially the diameter of 6mm. Because the wound is often irregular forms, to find the diameter, draw from all the edge pixels to the pixel edges through the midpoint and averaged.

A report or a new data set of the obtained features of the medical image dataset is prepared after the image processing. This contains Four classes Non cancer, mild, moderate, severe.

1	class	area	perimeter	maxdia	mindia	h_asym	v_asym	maxr	maxg	maxb	minr	ming	minb	h	s	v
2	severe	210136	881.74	600	450	0.367	0.562	255	255	255	0	2	25	14	140	90
3	severe	188162	48.63	600	450	0.079	0.079	254	241	255	0	0	0	12	140	89
4	severe	190534	30.73	600	450	0.27	0.27	216	209	238	0	0	0	11	140	90
5	severe	246518	49.56	600	450	0.742	0.192	255	243	250	11	11	44	15	140	90
6	moderate	152221	71.94	600	450	0.894	0.631	214	211	255	0	0	52	9	255	147
7	moderate	239795	709.31	600	450	0.4	0.418	249	231	245	0	2	45	9	255	147
8	moderate	221596	15.9	600	450	0.279	0.687	211	193	252	35	24	101	2	187	147
9	moderate	221633	9.66	600	450	0.634	0.012	210	199	253	6	3	27	5	221	147
10	moderate	215275	14.83	600	450	0.394	0.754	190	182	240	17	20	77	6	207	147
11	severe	250307	574.5	600	450	0.719	0.154	255	243	242	7	0	7	7	137	88
12	moderate	258643	406.37	600	450	0.281	0.313	234	214	226	2	9	37	14	251	147
13	severe	171692	32.38	600	450	0.361	0.452	231	222	229	3	4	28	1	122	88
14	severe	233875	61.94	600	450	0.099	0.27	242	228	255	0	7	25	11	139	90
15	moderate	193000	43.21	600	450	0.013	0.013	187	172	251	23	16	98	5	201	147
16	moderate	250219	584.3	600	450	0.329	0.362	210	207	253	28	21	93	2	197	147
17	moderate	223373	41.21	600	450	0.645	0.502	255	237	255	52	46	62	6	163	147
18	moderate	258999	425.85	600	450	0.457	0.223	178	177	250	13	11	54	5	220	147
19	moderate	255943	492.66	600	450	0.13	0.494	255	253	255	0	0	64	12	255	147
20	moderate	256973	449.39	600	450	0.388	0.346	204	206	253	23	23	93	3	204	147

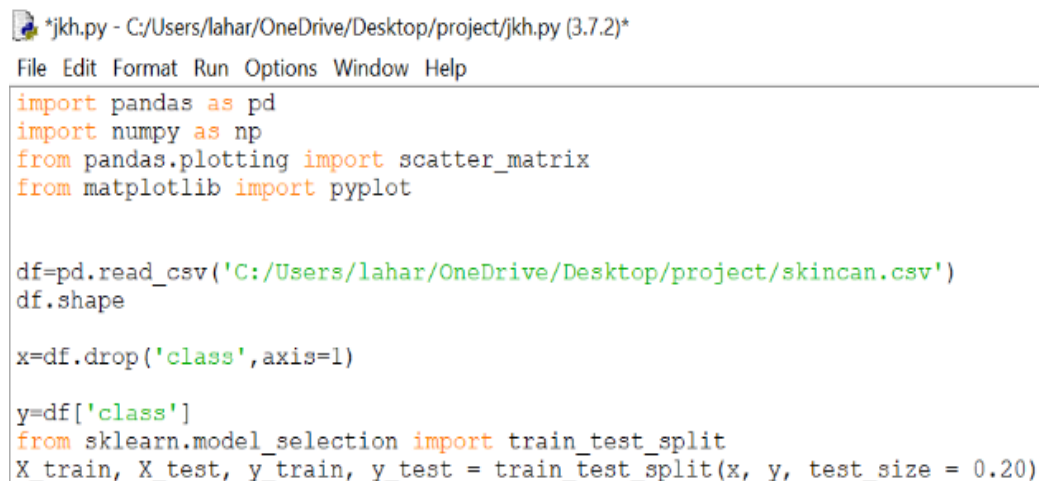
5.2 Support Vector Machine Algorithm

Train the machine learning model using the support vector machine algorithm. Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

5.2.1 Importing the dataset



```
*jkh.py - C:/Users/lahar/OneDrive/Desktop/project/jkh.py (3.7.2)*
File Edit Format Run Options Window Help

import pandas as pd
import numpy as np
from pandas.plotting import scatter_matrix
from matplotlib import pyplot

df=pd.read_csv('C:/Users/lahar/OneDrive/Desktop/project/skincan.csv')
df.shape

x=df.drop('class',axis=1)

y=df['class']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20)
```

Split the dataset accordingly for training as well as testing the model.

5.2.2 Fitting the SVM classifier to the training set and predicting:

Now the training set will be fitted to the SVM classifier. To create the SVM classifier, we will import **SVC** class from **Sklearn.svm** library.

In the below code, we have used **kernel='linear'**, as here we are creating SVM for linearly separable data. However, we can change it for non-linear data. And then we fitted the classifier to the training dataset(x_train, y_train), y is a new vector used to store the predicted value .

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
y=svclassifier.predict(X_test)
```

5.2.3 Performance Metrics :

Now we will see the performance of the SVM classifier that how many incorrect predictions are there as compared to the Logistic regression classifier. To create the confusion matrix, we need to import the confusion_matrix function of the sklearn library. After importing the function, we will call it using a new variable **cm**. The function takes two parameters, mainly y_true(the actual values) and y_pred (the targeted value return by the classifier). We also generate the accuracy_score of the model.

```
from sklearn.metrics import confusion_matrix, accuracy_score
a=accuracy_score(y_test, y)
print(confusion_matrix(y_test, y))
```

5.3 User Interface Using Tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

5.3.1 Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table –

Sr.No.	Operator & Description
1	<u>Button</u> The Button widget is used to display buttons in your application.
2	<u>Canvas</u> The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	<u>Checkbutton</u> The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	<u>Entry</u>

	<p>The Entry widget is used to display a single-line text field for accepting values from a user.</p>
5	<p><u>Frame</u></p> <p>The Frame widget is used as a container widget to organize other widgets.</p>
6	<p><u>Label</u></p> <p>The Label widget is used to provide a single-line caption for other widgets. It can also contain images.</p>
7	<p><u>Listbox</u></p> <p>The Listbox widget is used to provide a list of options to a user.</p>
8	<p><u>Menubutton</u></p> <p>The Menubutton widget is used to display menus in your application.</p>
9	<p><u>Menu</u></p> <p>The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.</p>
10	<p><u>Message</u></p> <p>The Message widget is used to display multiline text fields for accepting values from a user.</p>
11	<p><u>Radiobutton</u></p> <p>The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.</p>

12	<u>Scale</u> <p>The Scale widget is used to provide a slider widget.</p>
13	<u>Scrollbar</u> <p>The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.</p>
14	<u>Text</u> <p>The Text widget is used to display text in multiple lines.</p>
15	<u>Toplevel</u> <p>The Toplevel widget is used to provide a separate window container.</p>
16	<u>Spinbox</u> <p>The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.</p>
17	<u>PanedWindow</u> <p>A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.</p>
18	<u>LabelFrame</u> <p>A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.</p>
19	<u>tkMessageBox</u> <p>This module is used to display message boxes in your applications.</p>

Table 5.3.1 Widgets present in Tkinter

5.3.2 Standard attributes

Let us take a look at how some of their common attributes such as sizes, colors and fonts are specified.

- Dimensions
- Colors
- Fonts
- Anchors
- Relief styles
- Bitmaps
- Cursors

5.3.3 Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The *pack()* Method – This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The *grid()* Method – This geometry manager organizes widgets in a table-like structure in the parent widget.
- The *place()* Method – This geometry manager organizes widgets by placing them in a specific position in the parent widget.

6. IMPLEMENTATION CODE

6.1 Code for Image Processing

Feature.py:

```
import numpy as np
import cv2
import glob
import matplotlib.pyplot as plt

def extract(filename):
    img=cv2.imread(filename)
    r=img[..., 0].min()
    g=img[..., 1].min()
    b=img[..., 2].min()
    r1=img[..., 0].max()
    g1=img[..., 1].max()
    b1=img[..., 2].max()
    gray = cv2.cvtColor(img.copy(), cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (17, 17), )
    ret, thresh =
    cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    contours,hierarchy = cv2.findContours(thresh, 1, 2)
    c = contours[0]
    cnt = max(contours, key=cv2.contourArea)
    M = cv2.moments(c)
    if(M['m00']!=0):
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
    else:
        cx=cy=0
    centroid=(cx,cy)
    perimeter = round(cv2.arcLength(c, True),2)
    area=cv2.countNonZero(thresh)
```

```

if len(c)>5:
    rect = cv2.fitEllipse(c)
    cv2.ellipse(img,rect,(0,255,0),2)
    cv2.drawContours(img, contours, -1, (0, 0, 255), 3)
    cv2.circle(img, (cx, cy), 7, (255, 255, 255), -1)
    cv2.putText(img, "center", (cx - 25, cy - 25),
    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)
    cv2.imwrite('fimage.jpg',img)
    (x, y) = rect[0]
    (w, h) = rect[1]
    angle = rect[2]
    if w < h:
        if angle < 90:
            angle -= 90
    else:
        angle += 90

    rows,cols=thresh.shape
    rot = cv2.getRotationMatrix2D((x, y), angle, 1)
    cos = np.abs(rot[0, 0])
    sin = np.abs(rot[0, 1])
    nW = int((rows * sin) + (cols * cos))
    nH = int((rows * cos) + (cols * sin))

    rot[0, 2] += (nW / 2) - cols / 2
    rot[1, 2] += (nH / 2) - rows / 2
    warp_mask=cv2.warpAffine(thresh,rot,(nH,nW))
    warp_img = cv2.warpAffine(img, rot, (nH, nW))
    warp_img_segmented = cv2.bitwise_and(warp_img, warp_img,
                                         mask=warp_mask)
    xx, yy, nW, nH = cv2.boundingRect(cnt)

```

```

flipContourHorizontal = cv2.flip(warp_mask, 1)
flipContourVertical = cv2.flip(warp_mask, 0)
diff_horizontal = cv2.compare(warp_mask, flipContourHorizontal,
                              cv2.CV_8UC1)
diff_vertical = cv2.compare(warp_mask, flipContourVertical,
                            cv2.CV_8UC1)

diff_horizontal = cv2.bitwise_not(diff_horizontal)
diff_vertical = cv2.bitwise_not(diff_vertical)
h_asym = round(cv2.countNonZero(diff_horizontal)/area,3)
v_asym = round(cv2.countNonZero(diff_vertical)/area,3)
D1=max([nH,nW])
D2=min([nH,nW])
hsv = cv2.cvtColor(img.copy(), cv2.COLOR_BGR2HSV)
low=np.array([0, 0,0])
high=np.array([15,140,90])
mask = cv2.inRange(hsv,low,high)
imask = mask>0
black = np.zeros_like(img, np.uint8)
black[imask] = img[imask]
hsvv=cv2.cvtColor(black.copy(), cv2.COLOR_BGR2HSV)
h = hsvv[:, :,0].max()
s = hsvv[:, :,1].max()
v = hsvv[:, :,2].max()
if D1>600 or D1<600:
    h=s=v=0

elif(h!=0 and s!=0 and v!=0):
    x='severe'
else:

```

```
low=low=np.array([0, 80,0])
high=np.array([15,255,147])
mask = cv2.inRange(hsv,low,high)
imask = mask>0
brown = np.zeros_like(img, np.uint8)
brown[imask] = img[imask]
hsvv=cv2.cvtColor(brown.copy(), cv2.COLOR_BGR2HSV)
h = hsvv[:, :,0].max()
s = hsvv[:, :,1].max()
v = hsvv[:, :,2].max()
if h!=0 and s!=0 and v!=0:
    x=m'oderate'
elif h!=0 or s!=0 or v!=0:
    x='mild'
else:
    h=s=v=0

return(area,perimeter,D1,D2,h_asym,v_asym,r1,g1,b1,r,g,b,h,s,v)
```

6.2 Code for Support Vector Machine

modeltrained.py:

```
import pandas as pd

import numpy as np

from pandas.plotting import scatter_matrix

from matplotlib import pyplot

# Load dataset

def model(feats):

    X_test1=feats
```

```
print(X_test1)

df=pd.read_csv('C:/Users/lahar/OneDrive/Desktop/project/skincan.csv')

df.shape

x=df.drop('class',axis=1)

y=df['class']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20)

from sklearn.svm import SVC

svclassifier = SVC(kernel='linear')

svclassifier.fit(X_train, y_train)

y=svclassifier.predict(X_test)

from sklearn.metrics import confusion_matrix,accuracy_score

a=accuracy_score(y_test, y)

print(confusion_matrix(y_test,y))

if len(X_test1)==1:

    y_pred=['non cancer']

    return (y_pred,a)

else:

    y_pred = svclassifier.predict(np.reshape(X_test1,(1,-1)))

    return(y_pred,a)
```

6.3 Code for User Interface

Userinterface.py:

```
import tkinter as tk

import tkinter.ttk as ttk

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageTk

import cv2


gui=tk.Tk()

gui.title("Skin Cancer Prediction")

nb = ttk.Notebook(gui)

nb.pack()

gui.geometry("1000x1000")


tab1=ttk.Frame(nb)

nb.add(tab1,text='Main')


def data():

    global filename

    filename = askopenfilename(initialdir='C:\\\\',title = "select image")

    import pandas as pd

    global img

    import cv2
```

```
e1.insert(0,filename)

img=cv2.imread(filename)

tab2=ttk.Frame(nb)

nb.add(tab2,text='Image')

l6=tk.Label(tab2,text="INITIAL      IMAGE",fg="light      blue",bg="dark
blue",font="Times 20 bold italic")

l6.grid(row=0,column=0)

imge = ImageTk.PhotoImage(Image.open(filename))

imag =tk.Label(tab2, image=imge)

imag.imge=imge

imag.grid(row=1,column=0)


def process():

    global blur

    gray = cv2.cvtColor(img.copy(), cv2.COLOR_BGR2GRAY)

    blur = cv2.GaussianBlur(gray, (17, 17), 32)

    cv2.imwrite('gray.jpg',gray)

    tab3=ttk.Frame(nb)

    nb.add(tab3,text='Binary')

    l5=tk.Label(tab3,text="BINARY      IMAGE",fg="light      blue",bg="dark
blue",font="Times 20 bold italic")
```



```
l5.grid(row=0,column=0)

path='C:/Users/lahar/OneDrive/Desktop/project/gray.jpg'

image = ImageTk.PhotoImage(Image.open(path))

imag =tk.Label(tab3, image=image)

imag.image=image

imag.grid(row=1,column=0)

cv2.imwrite('blur.jpg',blur)

tab4=ttk.Frame(nb)

path='C:/Users/lahar/OneDrive/Desktop/project/blur.jpg'

nb.add(tab4,text='PreProcessed Image')

l4=tk.Label(tab4,text="PRE PROCESSED IMAGE",fg="light blue",bg="dark
blue",font="Times 20 bold italic")

l4.grid(row=0,column=0)

image = ImageTk.PhotoImage(Image.open(path))

imag =tk.Label(tab4, image=image)

imag.image=image

imag.grid(row=1,column=0)

def segment():

    global thresh

    ret,thresh =
cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

    cv2.imwrite('segmented.jpg',thresh)
```

```

tab5=ttk.Frame(nb)

nb.add(tab5,text='Segmented Image')

l3=tk.Label(tab5,text="SEGMENTED      IMAGE",fg="light      blue",bg="dark
blue",font="Times 20 bold italic")

l3.grid(row=0,column=0)

path='C:/Users/lahar/OneDrive/Desktop/project/segmented.jpg'

image = ImageTk.PhotoImage(Image.open(path))

imag =tk.Label(tab5, image=image)

imag.image=image

imag.grid(row=1,column=0)

def feature():

    global f

    import feature

    names                                     =
['area','perimeter','maxdia','mindia','h_asym','v_asym','maxr','maxg','maxb','minr','ming','m
inb','maxh','maxs','maxv']

    f=[]

    features=feature.extract(filename)

    if features[0]=='non':

        f.append(features[0])

        for i in range(1,len(features)):

            b.insert(i+1,names[i-1]+" "+str(features[i]))

    else:

        for i in range(0,len(features)):

```

```
b.insert(i+1,names[i]+" "+str(features[i]))

f.append((features[i]))

tab6=ttk.Frame(nb)

nb.add(tab6,text='Final Image')

l2=tk.Label(tab6,text="FINAL IMAGE",fg="light blue",bg="dark blue",font="Times
20 bold italic")

l2.grid(row=0,column=0)

path='C:/Users/lahar/OneDrive/Desktop/project/fimage.jpg'

image = ImageTk.PhotoImage(Image.open(path))

imag =tk.Label(tab6, image=image)

imag.image=image

imag.grid(row=1,column=0)

def prediction():

    import modeltrained

    predict=modeltrained.model(f)

    s=predict[0][0]

    y=predict[1]

    e.insert(0,s)

    a.insert(0,y)

    print("done")

def close_window():
```

```
gui.destroy()

l=tk.Label(tab1,text="Stage Prediction Of Skin Cancer",

           fg = "light blue",

           bg = "dark blue",

           font = "Times 20 bold italic" )

l1=tk.Label(tab1, text='Browse Image',fg="black",font="Times 13 ")

e1 = tk.Entry(tab1,text=")

br=tk.Button(tab1,text='Browse',fg="blue",font="Times 13",command=data)

pp=tk.Button(tab1,text='Preprocess',fg="blue",font="Times 13",command=process)

sg=tk.Button(tab1,text='Segmentation',fg="blue",font="Times 13",command=segment)

fe=tk.Button(tab1,text='Feature Extraction',fg="blue",font="Times

13",command=feature)

b=tk.Listbox(tab1)

pr=tk.Button(tab1,text='Predict',fg="blue",font="Times 13",command=prediction)
```

```
e=tk.Entry(tab1,text=' ',fg="red")

l8=tk.Label(tab1,text="Accuracy",fg="black",font="Times 13")

l8.grid(row=12,column=0)

a=tk.Entry(tab1,text="",fg='red')

a.grid(row=12,column=1)


ex=tk.Button (tab1, text = "exit",fg='blue',font="Times 13",command=close_window)

l.grid(row=0,column=1,padx=2,pady=2)


l1.grid(row=1,column=0)

e1.grid(row=1,column=1)

br.grid(row=2,column=1)

pp.grid(row=3,column=0)

sg.grid(row=3,column=2)

fe.grid(row=4,column=1,padx=3,pady=3)

b.grid(row=5,column=1,padx=15,pady=15)

pr.grid(row=10,column=0)

e.grid(row=10,column=1)

ex.grid(row=14,column=1,padx=15,pady=15)

gui.mainloop()
```

7.TESTING

7.1 Test Cases

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.2 Types of Tests

7.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.2.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.2.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

7.2.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.2.7 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

7.3 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format.
- No duplicate entries should be allowed.
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results

All the test cases mentioned above passed successfully. No defects encountered.

8. OUTPUT SCREENS



Fig 8.1 The User Interface

Prediction of Skin Cancer Using Machine Learning



Fig 8.2 After Browsing the image from the database of images



Fig 8.3 Binary image a stage of pre processing

Prediction of Skin Cancer Using Machine Learning



Fig 8.4 Pre Processed Noise reduced image

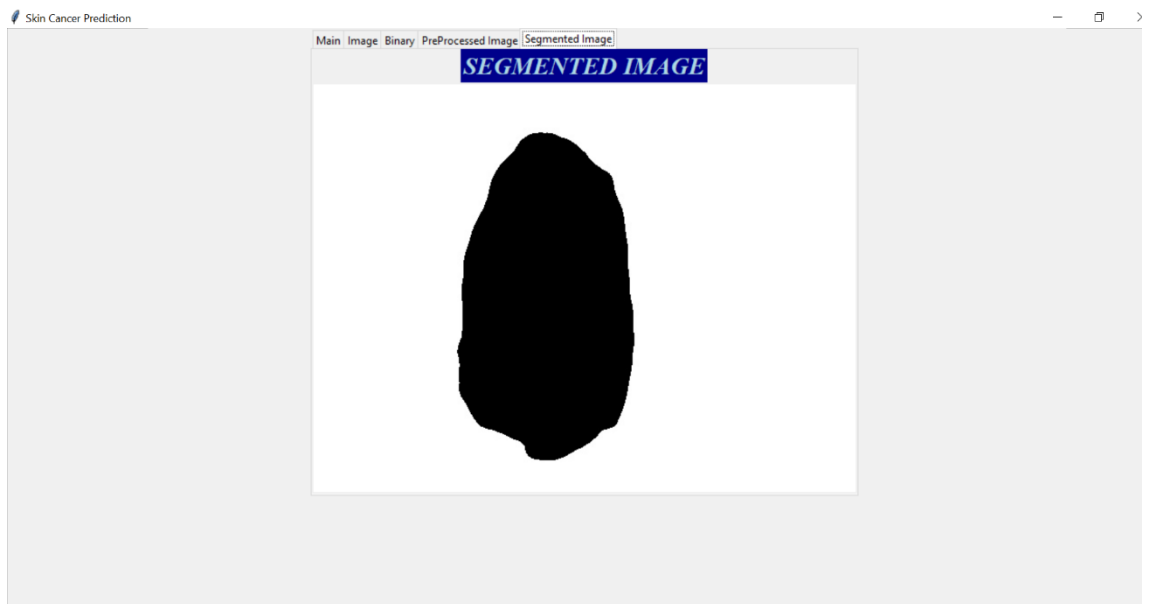


Fig 8.5 Segmented section of the lesion

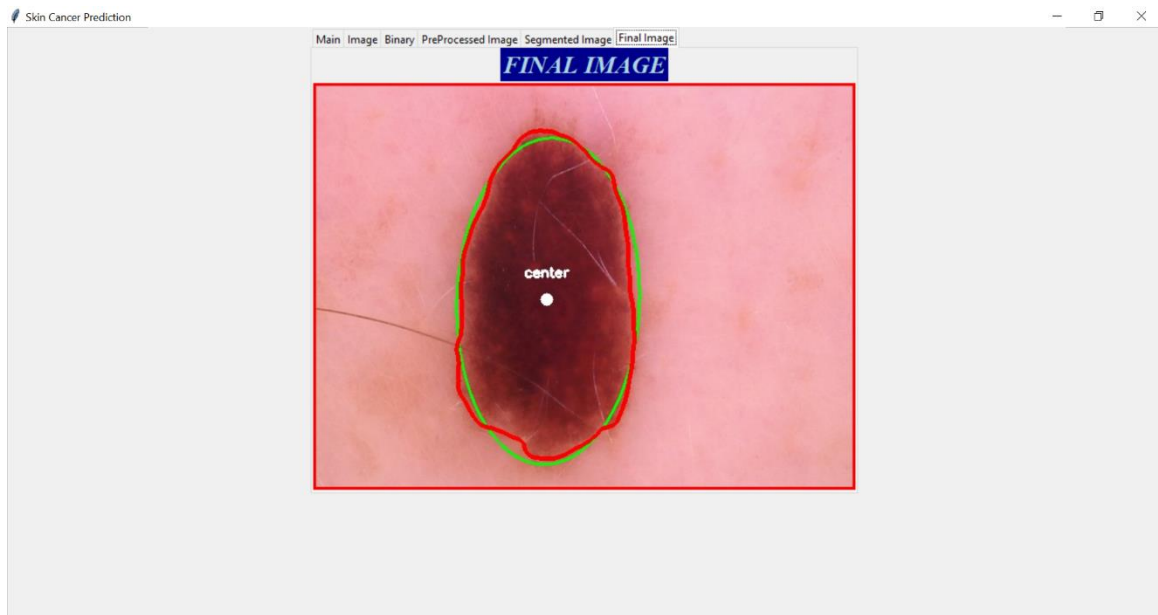


Fig 8.6 Boundary marked final image after feature selection

Prediction of Skin Cancer Using Machine Learning



Fig 8.7 Displaying the features of the input image



Fig 8.8 Final Prediction along with the Accuracy

9. Conclusion and Future Scope

By using this Machine learning model, it is easy to identify the severity of the skin lesion. This would be a very helpful tool in the hands of a dermatologist. In earlier days, detection of the skin lesion was done by using special dermoscopic machines along with biopsy. Our model would be an enhancement and guiding hand to obtain a faster results which would help in taking better decisions in ample amount of time.

This model which we built is fast, low expensive, efficient and make predictions with utmost accuracy.

The future scope of this project is by using adding more images to the dataset we can integrate the prediction of different skin diseases. This will help in distinguishing between the different type of skin disease. Also, could be enhanced to be made available to the user by using android, which would help in the home detection with necessary steps to be taken.

10. References

- [1] Arul N, Cho YY, “A Rising Cancer Prevention Target of RSK2 in Human Skin Cancer”, 2013 Aug 5; 3:201. doi: 10.3389/fonc.2013.00201. eCollection 2013.
- [2] American cancer society skin cancer prevention and early detection, in annual report(2013)
[“http://www.cancer.org/cancer/cancercauses/sunanduvexposure/skincancerpreventionandearlydetection/skin-cancer-prevention-and-early-detection-intro.”](http://www.cancer.org/cancer/cancercauses/sunanduvexposure/skincancerpreventionandearlydetection/skin-cancer-prevention-and-early-detection-intro.”)
- [3] Franz Nachbar, MD, Wilhelm Stolz, MD, TanjaMerkle, MD, Armand B. Cagnetta, MD, Thomas Vogt, MD, Michael Landthaler, MD, Peter Bilek, Otto Braun-Falco, MD, and GerdPlewig, MD, “The ABCD rule of dermatoscopy” in Journal of the American Academy of Dermatology ,Volume 30, Number 4.
- [4] International Journal of Scientific & Engineering Research Volume 4, Issue 2, February-2013.
- [5] Margarida Silveira, Member, IEEE, Jacinto C. Nascimento, Member, IEEE, Jorge S. Marques, André R. S. Marçal, Member, IEEE, Teresa Mendonça, Member, IEEE, Syogo Yamauchi, Junji Maeda, Member, IEEE, and Jorge Rozeira., “Comparison of Segmentation Methods for Melanoma Diagnosis in Dermoscopy Images” in IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING, VOL. 3, NO. 1, FEBRUARY 2
- [6] Marium A Sheha,Mai S Mabrouk and Amr Sharawy, "Automatic Detection of Melanoma skin cancer using Texture Analysis," International Journal of Computer Applications, Vol. 42 no 20, pp. 22-26, July 2011.
- [7] Nilkamal S. Ramteke and Shweta V. Jain, “ABCD rule based automatic computer-aided skin cancer detection using MATLAB” in Int.J.Computer Technology & Applications, Volume 4 (4), 691-697.