**CHAPTER 1**

**INTRODUCTION**

* 1. **SKIN CANCER**

Skin cancer could be a dangerous infection. Skin has three (3) fundamental layers. Skin cancer starts in furthest layer, which is made up of first layer squamous cells, moment layer basal cells, and deepest or third layer melanocytes cell. Squamous cell and basal cell are in some cases called non-melanoma cancers. Non-melanoma skin cancer continuously reacts to treatment and rarely spreads to other skin tissues. Melanoma is more perilous than most other types of skin cancer [3]. In case it isn't recognized at beginning organize, it is rapidly attack adjacent tissues and spread to other parts of the body. Formal diagnosis method to skin cancer discovery is Biopsy method. A biopsy may be a strategy to evacuate a bit of tissue or a test of cells from quiet body so that it can be dissected in a research facility. It is awkward strategy. Biopsy Strategy is time expending for understanding as well as specialist since it takes parcel of time for testing.

* 1. **BIOPSY SKIN LESSION SEGMENTATION**

Biopsy is done by evacuating skin tissues (skin cells) which sample undergoes arrangement of laboratory testing [1]. There's plausibility of spreading of illness into other portion of body. It is more risky. Considering all the cases said over, So Skin cancer discovery utilizing svm is proposed. This strategy uses digital picture handling strategy and SVM for classification. This procedure has motivated the early location of skin cancers, and requires no oil to be connected to your skin to realize clear sharp pictures of your moles. In this way, it's quicker and cleaner approach. But, most imperatively, due to its higher amplification, Skin Cancer Location Utilizing SVM can prevent the superfluous extraction of flawlessly safe moles and skin injuries.

**1.3 MELANOMA SKIN CANCER**

Melanoma is a serious type of skin cancer and can occur anywhere on the body including areas that are never exposed to UV radiation. If caught late or left untreated, it can spread to other parts of the body. It is more common in fair-skinned people and can run in families. UV exposure is a common cause of melanoma, and intermittent high-intensity exposure is the most dangerous pattern of UV exposure for increasing the risk of developing melanoma. Melanoma typically shows up as a dark brown or black mole on the skin. Signs that a mole might be abnormal or might be a melanoma include the A, B, C, D, Es: Asymmetry, jagged or uneven Borders, multiple Colors, large Diameter (larger than the eraser on a pencil), and/or a mole that is Evolving or changing color or shape

* 1. **MACHINE LEARNING**

Machine learning (ML) is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions Recently, artificial neural networks have been able to surpass many previous approaches in performance.

Machine learning approaches have been applied to many fields including natural language processing, computer vision, speech recognition, email filtering, agriculture, and medicine.[4][5] ML is known in its application across business problems under the name predictive analytics. Although not all machine learning is statistically based, computational statistics is an important source of the field's methods.

The mathematical foundations of ML are provided by mathematical optimization (mathematical programming) methods. Data mining is a related (parallel) field of study, focusing on exploratory data analysis (EDA) through unsupervised learning.

* 1. **DIAGNOSTIC SYSTEM**

The diagnostic can be performed without any support, in the naked eye, although the result isn’t always reliable, therefore dermoscopy was created. It consists in using a device to take a picture of the lesion in order to analyze its features to determine whether the lesion is benign or not. As some people don’t have access to a dermatologist, and even with an experienced eye the result can be false, it is necessary to develop automatic methods in order to increase the accuracy of the diagnostic. Early detection is the most effective tool for controlling this kind of cancer. The main criterion to differentiate between benign and malignant skin lesion is the so-called ABCD rule. This criterion evaluates the asymmetry, edge, color and size of the skin lesion to generate a diagnostic

* 1. **SCREENING AND IDENTIFICATION**

Early detection is the most important factor in identifying and treating skin cancers. Many organizations, including the American Cancer Society, the American Academy of Dermatology, and the Skin Cancer Foundation, recommend regular skin self-examination. People who have already had skin cancer or who are known to be at high risk for developing skin cancer should be evaluated regularly by a health care clinician.

**1.6 PROBLEM DESCRIPTION**

Skin cancer can be defined as skin growths with differing causes and various degrees of malignancy. Skin cancer can also be referred to as Skin Neoplasm’s. Skin cancer develops on skin and so can be seen. The main cause of Skin cancer all over the world is UV radiations coming from the sun and it is estimated that Americans are greatly affected by skin cancers than the Africans and Asians. This is due to the fair complexion of their skin and so less melanin. Whereas Africans and Asians due to the high melanin content in the skin is far resistant to skin cancer [1]. It has been statistically proven that fairer skin toned people are much prone to tanning and so is prone to skin cancer. Cancer is the general name for a group of more than 100 diseases. Although there are different kinds of cancer, all cancers occur because abnormal cells grow out of control. Untreated cancers can cause serious illness and death.

Skin cancer is the most commonly occurring cancer. Skin cancer develops on skin and therefore from skin cells. Based on the type of skin cells, from which cancer arise, is classified into; Basal cell cancer [BCC], Squamous cell cancer [SCC], Melanoma Basal cell cancer Basal cell cancer is the most common skin cancer occurs in sun exposed areas. It rarely causes death as it rarely spreads. It is easily treated with surgery or radiation. Symptoms for basal cell cancer are: Raised, smooth, pearly bump on sun exposed skin (head, neck or shoulders).Small blood vessels are seen sometimes. Crusting and bleeding in the centre of the tumor. Squamous cell cancer It is less common than Basal cell cance cancer are red, scaling, thickened patch, ulceration and bleeding may occur and it develops into large mass if not treated.

**CHAPTER 2**

**LITERATURE SURVEY**

(Sumithra R, and et. al., 2015)Introduce unused strategy for automatic extricate injury zone and classified them as generous or malignant. Current paper begin with the method of de-noising and expelling the undesirable parts such as hair, earlier to segment the injury. The creator proposed a locale growing way to fragment the injury which initialized seed point automatically. Extricated injury is spoken to by texture features and colors. Two sorts of classifier (SVM and k-NN) are utilized or classification of extricated highlights [1].

(J Abdul Jaleel 2013)proposed Skin location based on Most extreme Entropy Limit, highlight extricated by utilizing Gray Level Co-occurrence Matrix(GLCM), and c1assification utilizing Fake Neural Network(ANN). Back-Propagation Neural (BPN) Organize is utilized for c1assification purpose[2]

(M.Chaithanya Krishna 2016): This paper employments division as different clustering procedure, highlights can be extricated by using ABCD (Asymmetry Record Border Colour List Distance across) strategy [3].

(A.A.L.C. Amarathunga 2015): This framework utilized run the show based and forward chaining approach to identify skin illness. Proposed system empowers client to distinguish children skin maladies by means of online and give valuable restorative proposals. Utilized different data mining classification calculations (AdaBoost, BayesNet, MLP and NaiveBayes) to foresee and analyze the skin disease. This as it were works for three skin maladies (Dermatitis, Impetigo and Melanoma) [4].

(Kawsar Ahmed 2013): In this paper analysts have utilized different Information Pre-processing strategies, Malady Diagnosis, Maximal Visit Itemset Calculation for preparing, K-means clustering for division and critical visit pattern for classification [5].

(Mariam A.Sheha,Mai S.Mabrouk, Amr Sharawy2012): This paper presents an strategy for melanoma determination connected on a set of computerized pictures. Highlights extricated by utilizing gray level Co-occurrence lattice (GLCM) and Utilizing Multilayer perceptron classifier (MLP) to classify between cancerous and noncancerous pictures [6].

**CHAPTER 3**

**METHODOLOGY**

**3.1 EXISTING SYSTEM**

In later days, skin cancer is seen as one of the foremost Dangerous frame of the Cancers found in People. Skin cancer is found in different sorts such as Melanoma, Basal and Squamous cell Carcinoma among which Melanoma is the foremost eccentric. The discovery of Melanoma cancer in early organize can be accommodating to remedy it. Computer vision can play critical part in Therapeutic Picture Determination and it has been demonstrated by numerous existing frameworks. In this paper, we show a computer helped strategy for the discovery of Melanoma Skin Cancer utilizing Picture Handling devices. The input to the system is the skin injury picture and after that by applying novel picture preparing methods, investigations it to conclude almost the nearness of skin cancer. The Injury Picture examination apparatuses checks for the different Melanoma parameters Like Asymmetry, Border, Colour, Diameter,(ABCD) etc. by surface, estimate and shape examination for picture division and include stages. The extricated highlight parameters are utilized to classify the picture as Ordinary skin and Melanoma cancer injury.

**Disadvantages of Existing System**

* Detection of the skin cancer with low variation is problematic
* Complexity of image based detection is identified
* The level variation is meant with different processing

**3.2 PROPOSED SYSTEM**

There are many types of the skin cancer, each type has a different color, size and features. Many skin features may have impact on digital images like hair and color, and other impacts such as lightness, and type of the scanner or digital camera. In the first stage, we have obtained the features related with images using discrete wavelet transformation. In the second stage, the features of skin images have been reduced using principle component analysis to the more essential features. In the classification stage, two classifiers based on supervised machine learning have been developed. The first classifier based on feed forward back-propagation artificial neural network and the second classifier based on k-nearest neighbor. The classifiers have been used to classify subjects as normal or abnormal skin cancer images. A classification with a success of 95% and 97.5% has been obtained by the two proposed classifiers and respectively. This result shows that the proposed hybrid techniques are robust and effective.

**Advantages of Proposed System**

* The skin cancer can be detected with a high accuracy
* The system implements with a good level of system
* The implementation work carried with a wide variety of system

**3.3 MODULES**

* Image acquisition
* Pre-processing
* Image sharpening
* Segmentation
* Feature Extraction
* Classification
* Cancer Detection

**3.4 MODULES DESCRIPTION**

**3.4.1 Image Acquisition**

Input to proposed system is dermoscopic images, dermoscopic images are images taken by dermatoscope. It is kind of magnifier used to take pictures of skin lesions (body part). It is hand held instrument make it very easier to diagnose skin disease. The skin image acquisition is the process where the image is given to the system. The image can be in any size to be uploaded by the trainer or tester.Here the skin image are prepared for the further analysis of the cancer detection system

**Fig 3.1 Image acquisition**

**3.4.2 Pre-Processing**

Goal of pre-processing is an improvement of image data that reduces unwanted distortions and enhances some image features important for further image processing. Image pre-processing involves three main things

1) Gray scale conversion

2) Noise removal.

**1) Grayscale conversion**

Grayscale image contains only brightness information. Each pixel value in grayscale image corresponds to an amount or quantity of light. The brightness graduation can be differentiated in grayscale image. Grayscale image measures only light intensity. 8 bit image will have brightness variation from 0 to 255 where ‘0’ represents black and ‘255’ represents white. In grayscale conversion colour image is converted into grayscale image shows in fig (6.2.1). Grayscale images are easier and faster to process than coloured images. All image processing technique are applied on grayscale image [4]. In our proposed system coloured or RBG image is converted into grayscale image by using weighted sum method by using following equations

Grayscale intensity= 0.299 R + 0.587 G + 0.114 B (6.1)

**Noise Removal**

The objective of noise removal is to detect and removed unwanted noise from digital image. The difficulty is in deciding which features of an image are real and which are caused by noise. Noise is random variations in pixel values. In our proposed system we are using median filter to remove unwanted noise shows in fig (4). Median filter is nonlinear filter, it leaves edges invariant. Median filter is implemented by sliding window of odd length [4]. Each sample value is sorted by magnitude, the centre most value is median of sample within the window, is a filter output.

**Fig4.2 Image processing**

**3.4.3 Image Sharpening**

The objective of image sharpening is to process an image to increase visibility of feature of interest. Here contrast enhancement is used to get better quality result. Image sharpening is an effect applied to digital images to give them a sharper appearance. Almost all lenses can benefit from at least a small amount of sharpening. Here the image sharpening is applied with the verification of the pre-processed image to get a vital pixel.

**3.4.4 Segmentation**

Segmentation is process of removing region of interest from given image. Region of interest containing each pixel similar attributes. Here we are using maximum entropy thresholding for segmentation [5]. First of all we have to take gray level of original image then calculate histogram of gray scale image then by using maximum entropy separate foreground from background. After maximum entropy we obtained binary image that is black and white image shows in fig 6.2.2. Image segmentation is performed by using our proposed automatic thresholding and masking operation in R, G and B planes. First, automatic thresholding proposed by Otsu12 is applied in each plane. Binary masks for each plane are obtained and then combined to produce a final lesion mask. We use 3-plane masking procedure to increase segmentation accuracy. Then edge detection is applied to further segmentation. The main prerequisite for extracting the features is that the lesion must be separated from the surrounding normal skin. But the segmented image may contain other smaller blobs which are not the skin lesion. To overcome this, we find the biggest blob in the segmented image. The segmented image obtained contains only the skin lesion.

**Fig3.3 Image segmentation**

The skin-images on which we are going to work contain the cancerous mole along with the skin part, so it is very important to remove the region of interest. So for this purpose we are going to use contour segmentation method.

**3.4.5 Feature Extraction**

To create a GLCM, the gray co-matrix function is used. The gray-level co-occurrence matrix (GLCM) is created by gray co matrix function. This is done by determining how often a pixel with the intensity (gray-level) value i occurs in a specific spatial relationship to a pixel with the value j. Each element (i, j) in GLCM is found by the sum of the number of times that the pixel with value i occurred in the specified spatial relationship to a pixel with value j in the input image. Because the processing is required to calculate a GLCM for the full dynamic range of an image is prohibitive, gray co-matrix scales the input image. The scaling is used by gray co-matrix for reducing the number of intensity values in grayscale image from 256 to eight. The size of the GLCM is determined by the number of gray levels. The number of gray levels the matrix called GLCM and intensity value scaling can be controlled by the Num Levels and the Gray Limits parameters. Certain properties of the spatial distribution of the gray scale image can be revealed by gray-level co-occurrence matrix.

For instance, when most of the values in the GLCM are clustered along the diagonal, the texture is coarse with respect to the specified offset. Several statistical measures can be derived from the GLCM. Segmentation is followed by feature extraction. No machine learning algorithm can work without predefined features set. The type of features can be broadly divided into following categories.

**Fig 3.4 Feature extraction**

The main features of the Melanoma skin Lesion are its Geometric Feature. Hence, we propose to extract the Geometric Features of segmented skin lesion.

Here, we used some standard geometry features (Area, Perimeter, Greatest Diameter, Circularity Index, and Irregularity Index) adopted. From the Segmented image containing only skin lesion, the image blob of the skin lesion is analyzed to extract the it’s geometrical features. The Different Features extracted are as follows: Area (A): Number of pixels of the lesion. Perimeter (P): Number of edge pixels. Major Axis Length or Greatest Diameter (GD): The length of the line passing through lesion centroid and connecting the two farthest boundary points.

* + 1. **FURIA Classification**

Using the FURIA rules for the melanoma skin cancer, we use some pre-defined thresholds in classification stage. The Feature Values Extracted in the Feature Extraction stage is compared and the skin lesion is classified as Melanoma Skin Cancer or normal skin or Mole. This classification method proves to be efficient for most of the skin images. FURIA is often referred to as technique for reducing the number of variables in a data set without loss of information and as a possible process for identifying new variables in to another smaller set the newly created variables are not usually easy to interpret. FURIA has been most successful in applications such as image compression where data reduction and not interpretation is of primary importance. FURIA allows one to identify the uncorrelated components of an ensemble of data. FURIA is used for classification, to classify the skin cancer. FURIA uses a method of analysis which involves finding the linear combination of a set of variables that has maximum variance and removing its effect and then testing and training is done. With the results of testing and training, FURIA will find whether the given values are benign or malignant. If the values are below 1 then it is benign. In case, the values are above 1, it’s a malignant. FURIA extends the well-known RIPPER algorithm, a state-of-the-art rule learner, while preserving its advantages, such as simple and comprehensible rule sets. With the generative rule set the detection of the algorithm can be done with the verified pattern which is used/ the possible feature identification with FURIA will tends a fuzzy classification with the generative rule scheme. The detection of the full training features and tested with the extracted values

**Fig 3.5 FURIA Segmentation**

Classifier is used to classify cancerous image from other skin diseases. For simplicity FURIA classifier is used here. FURIA takes set of images and predicts for each input image belongs to which of the two categories of cancerous and non-cancerous classes. The purpose of FURIA is create hyper plane that separates two classes with maximum gap between them.

**\**

**CHAPTER 4**

**WORKING OF THE SYSTEM**

**4.1 SYSTEM ARCHITECTURE**

A system architecture or systems architecture is the 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 structures and behaviors of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system.

There have been efforts to formalize languages to describe system architecture; collectively these are called architecture description languages (ADLs). Various organizations define systems architecture in different ways, including:

* An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.
* Architecture comprises the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.
* If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and priorities for future purchases, and a plan for upgrading and/or replacing dated equipment and software.

An architecture diagram is a graphical representation of a set of concepts that are part of architecture, including their principles, elements and components. Architecture diagram can help system designers and developers visualize the high-level, overall structure of their system or application, in order to ensure the system meets their users' needs. Using architecture diagram, you can also describe patterns that are used throughout the design. It's somewhat like a blueprint that you use as a guide, so that you and your colleagues can discuss, improve and follow.

**Fig 4.1 System architecture**

**4.2 ALGORITHM USED**

**4.2.1 FURIA**

FURIA is short for Fuzzy Unordered Rule Induction Algorithm. FURIA extends the well-known RIPPER algorithm, a state-of-the-art rule learner, while preserving its advantages, such as simple and comprehensible rule sets. In addition, it includes a number of modifications and extensions. In particular, FURIA learns fuzzy rules instead of conventional rules and unordered rule sets instead of rule lists. Moreover, to deal with uncovered examples, it makes use of an efficient rule stretching method. Experimental results show that FURIA significantly outperforms the original RIPPER, as well as other classifiers such as C4.5, in terms of classification accuracy.

**4.2.2 Algorithm Working**

public class **FURIA**

extends [Classifier](https://sci2s.ugr.es/keel/javadoc/keel/Algorithms/Fuzzy_Rule_Learning/Hybrid/FURIA/core/Classifier.html)

implements [WeightedInstancesHandler](https://sci2s.ugr.es/keel/javadoc/keel/Algorithms/Fuzzy_Rule_Learning/Hybrid/FURIA/core/WeightedInstancesHandler.html)

This class implements the FURIA algorithm proposed by Hühn and Hüllermeier 2009  
  
The FURIA algorithm is a fuzzy rule learner based on the JRip implementation (RIPRER). The main difference between FURIA and JRip is that FURIA makes no use of default rules. Furthermore FURIA has a changed pruning procedure, which means that the pruning during the IREP\* runs was deactivated permanently. It was found out experimentally that this improved the classification rate. The following description from the JRip class was altered to describe the methodology of FURIA:

Initialize RS = {}, and for each of both class DO:

1. Building stage:

Repeat 1.1 until the description length (DL) of the ruleset and examples is 64 bits greater than the smallest DL met so far, or there are no positive examples, or the error rate >= 50%.  
  
**1.1. Grow phase:**

Grow one rule by greedily adding antecedents (or conditions) to the rule until the rule is perfect (i.e. 100% accurate). The procedure tries every possible value of each attribute and selects the condition with highest information gain: p(log(p/t)-log(P/T)).  
**2. Optimization stage:**

after generating the initial ruleset {Ri}, generate and prune two variants of each rule Ri from randomized data using procedure 1.1 and X.1. But one variant is generated from an empty rule while the other is generated by greedily adding antecedents to the original rule. Moreover, the pruning metric used here is (TP+TN)/(P+N).Then the smallest possible DL for each variant and the original rule is computed. The variant with the minimal DL is selected as the final representative of Ri in the ruleset.After all the rules in {Ri} have been examined and if there are still residual positives, more rules are generated based on the residual positives using Building Stage again.

3. Delete the rules from the ruleset that would increase the DL of the whole ruleset if it were in it. and add resultant ruleset

ENDDO

**Fuzzification:**

For each rule r in every ruleset in RS DO

**4. Fuzzification of antecedents:**

Apply greedy strategy to fuzzify the existing antecedents in r the following way:  
4.1 Examine all possible support bounds and select the one which gains the highest purity on the training data.

4.2 Set the maximum support bound determined in 4.1 and restart with 4.1 but withouth the fuzzified antecedent.

**ENDDO**  
Incrementally prune each rule and allow the pruning of any final sequences of the antecedents;The pruning metric is (p-n)/(p+n) -- but it's actually 2p/(p+n) -1, so in this implementation we simply use p/(p+n) (actually (p+1)/(p+n+2), thus if p+n is 0, it's 0.5).

**Classification time:**

If an instance is not covered by any rule, apply a rule stretching mechanism: Cut every rule just in front of the first discriminating antecedent such that the this way stretched rule covers the instance. Doing this for all rules will lead to a set of rules in which each one covers the instance (or is empty and may be excluded). To determine the rule that assigns the class calculate the weight given by its purity using the m-measure on the one hand the Laplace-fraction of antecedents left in comparison to the original number of the respective rule. The rule that maximizes that value is from the class that will be assigned.

**CHAPTER 5**

**EXPERIMENTAL ANALYSIS**

**5.1 SYSTEM CONFIGUIRATION**

**5.1.1 HARDWARE SPECIFICATION**

* Processor : Intel core processor 2.6.0 GHz
* RAM : 8GB
* Hard disk : 256 GB

Keyboard : Standard keyboard

* Monitor : 15 inch color monitor

**5.1.2 SOFTWARE SPECIFICATION**

* Front End : PYTHON
* MySQL : MySQL
* IDE : Pycharm
* Platform : Windows 10

**5.2 SAMPLE CODE**

Ar\_master

import pymysql

class master\_flask\_code:

def \_\_init\_\_(self):

self.user = 'root'

self.password = ''

self.host = 'localhost'

self.database = 'python\_intelligent\_attendance'

def find\_max\_id(self,table):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute("SELECT id FROM "+table)

data = cursor.fetchall()

maxin = len(data)

if maxin == 0:

maxin = 1

else:

maxin += 1

return maxin

def insert\_query(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

result=cursor.execute(qry)

conn.commit()

conn.close()

return result

def select\_login(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute(qry)

data = cursor.fetchall()

check = len(data)

if check == 0:

return 'no'

else:

return 'yes'

def select\_single\_colum(self,table,colum):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

qry1=("select "+colum+" from "+table)

cursor = conn.cursor()

cursor.execute(qry1)

data = cursor.fetchall()

return data

def admin\_login(self,username,password):

if username == 'admin' and password == 'admin':

return 'yes'

else:

return 'no'

Median Filter

import shutil

from tkinter import \*

import os

import cv2

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

# image = Image.open(r"median.png")

# image = image.convert("L")

# image = image.filter(ImageFilter.FIND\_EDGES)

# image.save('img\_blur.png')

root.destroy()

import edge\_detection

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

im1 = Image.open(r"greyscale.png")

im2 = im1.filter(ImageFilter.MedianFilter(size=3))

im2.save('median.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Median Filter Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open('greyscale.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('median.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Median Filter",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Main.py

from tkinter import \*

from PIL import ImageTk,Image

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

dd=0

root.destroy()

import median\_filter

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

filapath=sample\_data.student.file\_path

img = Image.open(filapath).convert('L')

img.save('greyscale.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="RGB Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open(sample\_data.student.file\_path)

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('greyscale.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Grayscale",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Feature Extraction

import shutil

from tkinter import \*

import os

from tkinter import messagebox

import random

import cv2

import imagehash as imagehash

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.pyplot as plt1

import cv2

import accuracy\_value

import sample\_data

def image\_matching(a,b):

i1 = Image.open(a)

i2 = Image.open(b)

i1 = i1.resize((230, 200), Image.ANTIALIAS)

i2 = i2.resize((230, 200), Image.ANTIALIAS)

assert i1.mode == i2.mode, "Different kinds of images."

assert i1.size == i2.size, "Different sizes."

pairs = zip(i1.getdata(), i2.getdata())

if len(i1.getbands()) == 1:

# for gray-scale jpegs

dif = sum(abs(p1-p2) for p1,p2 in pairs)

else:

dif = sum(abs(c1-c2) for p1,p2 in pairs for c1,c2 in zip(p1,p2))

ncomponents = i1.size[0] \* i1.size[1] \* 3

xx= (dif / 255.0 \* 100) / ncomponents

return xx

def testing():

name = []

values = []

input\_image=sample\_data.student.file\_path

entries = os.listdir('train/')

for x in entries:

val=100

directory=x

name.append(x)

x1="train/"+x

arr = os.listdir(x1)

for x2 in arr:

path=x1+"/"+str(x2)

find=image\_matching(path,input\_image)

if(find<val):

val=find

values.append(val)

values\_lenght= len(values)

pos=0;

pos\_val=100

result="unknown"

for x in range(0, values\_lenght):

if values[x]<pos\_val:

pos=x

pos\_val=values[x]

print(pos\_val)

if(pos\_val<20):

result=name[pos]

messagebox.showinfo(title="info", message=str(result))

plt.close()

nn = accuracy\_value.sample()

nn.demo()

################################################################## read dataset

def read\_first\_data():

img\_file = 'img\_blur.png'

img = cv2.imread(img\_file, cv2.IMREAD\_COLOR) # rgb

alpha\_img = cv2.imread(img\_file, cv2.IMREAD\_UNCHANGED) # rgba

gray\_img = cv2.imread(img\_file, cv2.IMREAD\_GRAYSCALE) # grayscale

print(type(img))

print('RGB shape: ', img.shape) # Rows, cols, channels

print('ARGB shape:', alpha\_img.shape)

print('Gray shape:', gray\_img.shape)

print('img.dtype: ', img.dtype)

print('img.size: ', img.size)

hash0 = imagehash.average\_hash(Image.open(img\_file))

im = cv2.imread(img\_file)

vals = im.mean(axis=2).flatten()

counts, bins = np.histogram(vals, range(257))

plt.bar(bins[:-1] - 0.5, counts, width=1, edgecolor='none')

plt.xlim([0, 100])

plt.show()

################################################################## Next page

def next\_page():

name = "train"

if os.path.exists(name):

h = 0;

else:

os.mkdir(name)

name=txt1.get()

name1 = txt2.get()

if (name == ""):

messagebox.showinfo(title="info", message="Enter Disease Details")

elif(name1==""):

messagebox.showinfo(title="info", message="Enter Description")

else:

name1 = "train\\" + name

if os.path.exists(name1):

j = 0;

else:

os.mkdir(name1)

ri2 = Image.open(sample\_data.student.file\_path)

ri2 = ri2.resize((230, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

x = random.randint(10000000, 100000000)

str = name1 + '\\%d.jpg' % x

shutil.copy(sample\_data.student.file\_path, str)

messagebox.showinfo(title="info", message="Success")

################################################################## main loop

image = Image.open(sample\_data.student.file\_path)

width, height = image.size

hash0 = imagehash.average\_hash(Image.open(sample\_data.student.file\_path))

print(hash0)

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Feature Extraction",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

# message = Label(root, text="Feature Extraction",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

# message.place(x=400, y=170)

ri2 = Image.open('img\_blur.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

# txt=Entry(root,width=10)

# txt.place(x=400, y=200)

# lbl2 = Label(root)

# a1 = Image.open('img\_blur.png')

# a123 = a1.resize((200, 200), Image.ANTIALIAS)

# a12 = ImageTk.PhotoImage(a123)

#

# lbl2.configure(image=a12)

message = Label(root, text="Disease",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

txt1 = Entry(root, width=15, font=('times', 25, ' bold '))

txt1.place(x=400, y=190)

message = Label(root, text="Description",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=240)

txt2 = Entry(root, width=15, font=('times', 25, ' bold '))

txt2.place(x=400, y=260)

#

#

# txt2 = Entry(root, width=15, font=('times', 25, ' bold '))

# txt2.place(x=400, y=280)

######## button with command function

compare\_dataset = Button(root, text="Feature Extraction",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Training",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

resust\_dataset.place(x=400, y=310)

resust\_dataset = Button(root, text="Testing",width=16 ,height=1,command=testing,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

resust\_dataset.place(x=400, y=360)

root.mainloop()

Edge Detection

import shutil

from tkinter import \*

import os

import cv2

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

# image = Image.open(r"median.png")

# image = image.convert("L")

# image = image.filter(ImageFilter.FIND\_EDGES)

# image.save('img\_blur.png')

root.destroy()

import feature\_extraction

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

image = Image.open(r"median.png")

image = image.convert("L")

image = image.filter(ImageFilter.FIND\_EDGES)

image.save('img\_blur.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Median Filter Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Edge Detection",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open('median.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('img\_blur.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Edge Detection",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Ar\_master

import pymysql

class master\_flask\_code:

def \_\_init\_\_(self):

self.user = 'root'

self.password = ''

self.host = 'localhost'

self.database = 'python\_intelligent\_attendance'

def find\_max\_id(self,table):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute("SELECT id FROM "+table)

data = cursor.fetchall()

maxin = len(data)

if maxin == 0:

maxin = 1

else:

maxin += 1

return maxin

def insert\_query(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

result=cursor.execute(qry)

conn.commit()

conn.close()

return result

def select\_login(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute(qry)

data = cursor.fetchall()

check = len(data)

if check == 0:

return 'no'

else:

return 'yes'

def select\_single\_colum(self,table,colum):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

qry1=("select "+colum+" from "+table)

cursor = conn.cursor()

cursor.execute(qry1)

data = cursor.fetchall()

return data

def admin\_login(self,username,password):

if username == 'admin' and password == 'admin':

return 'yes'

else:

return 'no'

Median Filter

import shutil

from tkinter import \*

import os

import cv2

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

# image = Image.open(r"median.png")

# image = image.convert("L")

# image = image.filter(ImageFilter.FIND\_EDGES)

# image.save('img\_blur.png')

root.destroy()

import edge\_detection

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

im1 = Image.open(r"greyscale.png")

im2 = im1.filter(ImageFilter.MedianFilter(size=3))

im2.save('median.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Median Filter Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open('greyscale.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('median.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Median Filter",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Main.py

from tkinter import \*

from PIL import ImageTk,Image

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

dd=0

root.destroy()

import median\_filter

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

filapath=sample\_data.student.file\_path

img = Image.open(filapath).convert('L')

img.save('greyscale.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="RGB Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open(sample\_data.student.file\_path)

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('greyscale.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Grayscale",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Feature Extraction

import shutil

from tkinter import \*

import os

from tkinter import messagebox

import random

import cv2

import imagehash as imagehash

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.pyplot as plt1

import cv2

import accuracy\_value

import sample\_data

def image\_matching(a,b):

i1 = Image.open(a)

i2 = Image.open(b)

i1 = i1.resize((230, 200), Image.ANTIALIAS)

i2 = i2.resize((230, 200), Image.ANTIALIAS)

assert i1.mode == i2.mode, "Different kinds of images."

assert i1.size == i2.size, "Different sizes."

pairs = zip(i1.getdata(), i2.getdata())

if len(i1.getbands()) == 1:

# for gray-scale jpegs

dif = sum(abs(p1-p2) for p1,p2 in pairs)

else:

dif = sum(abs(c1-c2) for p1,p2 in pairs for c1,c2 in zip(p1,p2))

ncomponents = i1.size[0] \* i1.size[1] \* 3

xx= (dif / 255.0 \* 100) / ncomponents

return xx

def testing():

name = []

values = []

input\_image=sample\_data.student.file\_path

entries = os.listdir('train/')

for x in entries:

val=100

directory=x

name.append(x)

x1="train/"+x

arr = os.listdir(x1)

for x2 in arr:

path=x1+"/"+str(x2)

find=image\_matching(path,input\_image)

if(find<val):

val=find

values.append(val)

values\_lenght= len(values)

pos=0;

pos\_val=100

result="unknown"

for x in range(0, values\_lenght):

if values[x]<pos\_val:

pos=x

pos\_val=values[x]

print(pos\_val)

if(pos\_val<20):

result=name[pos]

messagebox.showinfo(title="info", message=str(result))

plt.close()

nn = accuracy\_value.sample()

nn.demo()

################################################################## read dataset

def read\_first\_data():

img\_file = 'img\_blur.png'

img = cv2.imread(img\_file, cv2.IMREAD\_COLOR) # rgb

alpha\_img = cv2.imread(img\_file, cv2.IMREAD\_UNCHANGED) # rgba

gray\_img = cv2.imread(img\_file, cv2.IMREAD\_GRAYSCALE) # grayscale

print(type(img))

print('RGB shape: ', img.shape) # Rows, cols, channels

print('ARGB shape:', alpha\_img.shape)

print('Gray shape:', gray\_img.shape)

print('img.dtype: ', img.dtype)

print('img.size: ', img.size)

hash0 = imagehash.average\_hash(Image.open(img\_file))

im = cv2.imread(img\_file)

vals = im.mean(axis=2).flatten()

counts, bins = np.histogram(vals, range(257))

plt.bar(bins[:-1] - 0.5, counts, width=1, edgecolor='none')

plt.xlim([0, 100])

plt.show()

################################################################## Next page

def next\_page():

name = "train"

if os.path.exists(name):

h = 0;

else:

os.mkdir(name)

name=txt1.get()

name1 = txt2.get()

if (name == ""):

messagebox.showinfo(title="info", message="Enter Disease Details")

elif(name1==""):

messagebox.showinfo(title="info", message="Enter Description")

else:

name1 = "train\\" + name

if os.path.exists(name1):

j = 0;

else:

os.mkdir(name1)

ri2 = Image.open(sample\_data.student.file\_path)

ri2 = ri2.resize((230, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

x = random.randint(10000000, 100000000)

str = name1 + '\\%d.jpg' % x

shutil.copy(sample\_data.student.file\_path, str)

messagebox.showinfo(title="info", message="Success")

################################################################## main loop

image = Image.open(sample\_data.student.file\_path)

width, height = image.size

hash0 = imagehash.average\_hash(Image.open(sample\_data.student.file\_path))

print(hash0)

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Feature Extraction",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

# message = Label(root, text="Feature Extraction",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

# message.place(x=400, y=170)

ri2 = Image.open('img\_blur.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

# txt=Entry(root,width=10)

# txt.place(x=400, y=200)

# lbl2 = Label(root)

# a1 = Image.open('img\_blur.png')

# a123 = a1.resize((200, 200), Image.ANTIALIAS)

# a12 = ImageTk.PhotoImage(a123)

#

# lbl2.configure(image=a12)

message = Label(root, text="Disease",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

txt1 = Entry(root, width=15, font=('times', 25, ' bold '))

txt1.place(x=400, y=190)

message = Label(root, text="Description",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=240)

txt2 = Entry(root, width=15, font=('times', 25, ' bold '))

txt2.place(x=400, y=260)

#

#

# txt2 = Entry(root, width=15, font=('times', 25, ' bold '))

# txt2.place(x=400, y=280)

######## button with command function

compare\_dataset = Button(root, text="Feature Extraction",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Training",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

resust\_dataset.place(x=400, y=310)

resust\_dataset = Button(root, text="Testing",width=16 ,height=1,command=testing,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

resust\_dataset.place(x=400, y=360)

root.mainloop()

Edge Detection

import shutil

from tkinter import \*

import os

import cv2

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

# image = Image.open(r"median.png")

# image = image.convert("L")

# image = image.filter(ImageFilter.FIND\_EDGES)

# image.save('img\_blur.png')

root.destroy()

import feature\_extraction

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

image = Image.open(r"median.png")

image = image.convert("L")

image = image.filter(ImageFilter.FIND\_EDGES)

image.save('img\_blur.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Median Filter Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Edge Detection",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open('median.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('img\_blur.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Edge Detection",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Ar\_master

import pymysql

class master\_flask\_code:

def \_\_init\_\_(self):

self.user = 'root'

self.password = ''

self.host = 'localhost'

self.database = 'python\_intelligent\_attendance'

def find\_max\_id(self,table):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute("SELECT id FROM "+table)

data = cursor.fetchall()

maxin = len(data)

if maxin == 0:

maxin = 1

else:

maxin += 1

return maxin

def insert\_query(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

result=cursor.execute(qry)

conn.commit()

conn.close()

return result

def select\_login(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute(qry)

data = cursor.fetchall()

check = len(data)

if check == 0:

return 'no'

else:

return 'yes'

def select\_single\_colum(self,table,colum):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

qry1=("select "+colum+" from "+table)

cursor = conn.cursor()

cursor.execute(qry1)

data = cursor.fetchall()

return data

def admin\_login(self,username,password):

if username == 'admin' and password == 'admin':

return 'yes'

else:

return 'no'

Median Filter

import shutil

from tkinter import \*

import os

import cv2

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

# image = Image.open(r"median.png")

# image = image.convert("L")

# image = image.filter(ImageFilter.FIND\_EDGES)

# image.save('img\_blur.png')

root.destroy()

import edge\_detection

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

im1 = Image.open(r"greyscale.png")

im2 = im1.filter(ImageFilter.MedianFilter(size=3))

im2.save('median.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Median Filter Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open('greyscale.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('median.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Median Filter",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Main.py

from tkinter import \*

from PIL import ImageTk,Image

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

dd=0

root.destroy()

import median\_filter

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

filapath=sample\_data.student.file\_path

img = Image.open(filapath).convert('L')

img.save('greyscale.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="RGB Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open(sample\_data.student.file\_path)

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('greyscale.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Grayscale",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

Feature Extraction

import shutil

from tkinter import \*

import os

from tkinter import messagebox

import random

import cv2

import imagehash as imagehash

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.pyplot as plt1

import cv2

import accuracy\_value

import sample\_data

def image\_matching(a,b):

i1 = Image.open(a)

i2 = Image.open(b)

i1 = i1.resize((230, 200), Image.ANTIALIAS)

i2 = i2.resize((230, 200), Image.ANTIALIAS)

assert i1.mode == i2.mode, "Different kinds of images."

assert i1.size == i2.size, "Different sizes."

pairs = zip(i1.getdata(), i2.getdata())

if len(i1.getbands()) == 1:

# for gray-scale jpegs

dif = sum(abs(p1-p2) for p1,p2 in pairs)

else:

dif = sum(abs(c1-c2) for p1,p2 in pairs for c1,c2 in zip(p1,p2))

ncomponents = i1.size[0] \* i1.size[1] \* 3

xx= (dif / 255.0 \* 100) / ncomponents

return xx

def testing():

name = []

values = []

input\_image=sample\_data.student.file\_path

entries = os.listdir('train/')

for x in entries:

val=100

directory=x

name.append(x)

x1="train/"+x

arr = os.listdir(x1)

for x2 in arr:

path=x1+"/"+str(x2)

find=image\_matching(path,input\_image)

if(find<val):

val=find

values.append(val)

values\_lenght= len(values)

pos=0;

pos\_val=100

result="unknown"

for x in range(0, values\_lenght):

if values[x]<pos\_val:

pos=x

pos\_val=values[x]

print(pos\_val)

if(pos\_val<20):

result=name[pos]

messagebox.showinfo(title="info", message=str(result))

plt.close()

nn = accuracy\_value.sample()

nn.demo()

################################################################## read dataset

def read\_first\_data():

img\_file = 'img\_blur.png'

img = cv2.imread(img\_file, cv2.IMREAD\_COLOR) # rgb

alpha\_img = cv2.imread(img\_file, cv2.IMREAD\_UNCHANGED) # rgba

gray\_img = cv2.imread(img\_file, cv2.IMREAD\_GRAYSCALE) # grayscale

print(type(img))

print('RGB shape: ', img.shape) # Rows, cols, channels

print('ARGB shape:', alpha\_img.shape)

print('Gray shape:', gray\_img.shape)

print('img.dtype: ', img.dtype)

print('img.size: ', img.size)

hash0 = imagehash.average\_hash(Image.open(img\_file))

im = cv2.imread(img\_file)

vals = im.mean(axis=2).flatten()

counts, bins = np.histogram(vals, range(257))

plt.bar(bins[:-1] - 0.5, counts, width=1, edgecolor='none')

plt.xlim([0, 100])

plt.show()

################################################################## Next page

def next\_page():

name = "train"

if os.path.exists(name):

h = 0;

else:

os.mkdir(name)

name=txt1.get()

name1 = txt2.get()

if (name == ""):

messagebox.showinfo(title="info", message="Enter Disease Details")

elif(name1==""):

messagebox.showinfo(title="info", message="Enter Description")

else:

name1 = "train\\" + name

if os.path.exists(name1):

j = 0;

else:

os.mkdir(name1)

ri2 = Image.open(sample\_data.student.file\_path)

ri2 = ri2.resize((230, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

x = random.randint(10000000, 100000000)

str = name1 + '\\%d.jpg' % x

shutil.copy(sample\_data.student.file\_path, str)

messagebox.showinfo(title="info", message="Success")

################################################################## main loop

image = Image.open(sample\_data.student.file\_path)

width, height = image.size

hash0 = imagehash.average\_hash(Image.open(sample\_data.student.file\_path))

print(hash0)

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Feature Extraction",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

# message = Label(root, text="Feature Extraction",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

# message.place(x=400, y=170)

ri2 = Image.open('img\_blur.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

# txt=Entry(root,width=10)

# txt.place(x=400, y=200)

# lbl2 = Label(root)

# a1 = Image.open('img\_blur.png')

# a123 = a1.resize((200, 200), Image.ANTIALIAS)

# a12 = ImageTk.PhotoImage(a123)

#

# lbl2.configure(image=a12)

message = Label(root, text="Disease",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

txt1 = Entry(root, width=15, font=('times', 25, ' bold '))

txt1.place(x=400, y=190)

message = Label(root, text="Description",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=240)

txt2 = Entry(root, width=15, font=('times', 25, ' bold '))

txt2.place(x=400, y=260)

#

#

# txt2 = Entry(root, width=15, font=('times', 25, ' bold '))

# txt2.place(x=400, y=280)

######## button with command function

compare\_dataset = Button(root, text="Feature Extraction",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Training",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

resust\_dataset.place(x=400, y=310)

resust\_dataset = Button(root, text="Testing",width=16 ,height=1,command=testing,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 20, ' bold '))

resust\_dataset.place(x=400, y=360)

root.mainloop()

Edge Detection

import shutil

from tkinter import \*

import os

import cv2

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

# image = Image.open(r"median.png")

# image = image.convert("L")

# image = image.filter(ImageFilter.FIND\_EDGES)

# image.save('img\_blur.png')

root.destroy()

import feature\_extraction

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

image = Image.open(r"median.png")

image = image.convert("L")

image = image.filter(ImageFilter.FIND\_EDGES)

image.save('img\_blur.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Median Filter Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Edge Detection",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open('median.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('img\_blur.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Edge Detection",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

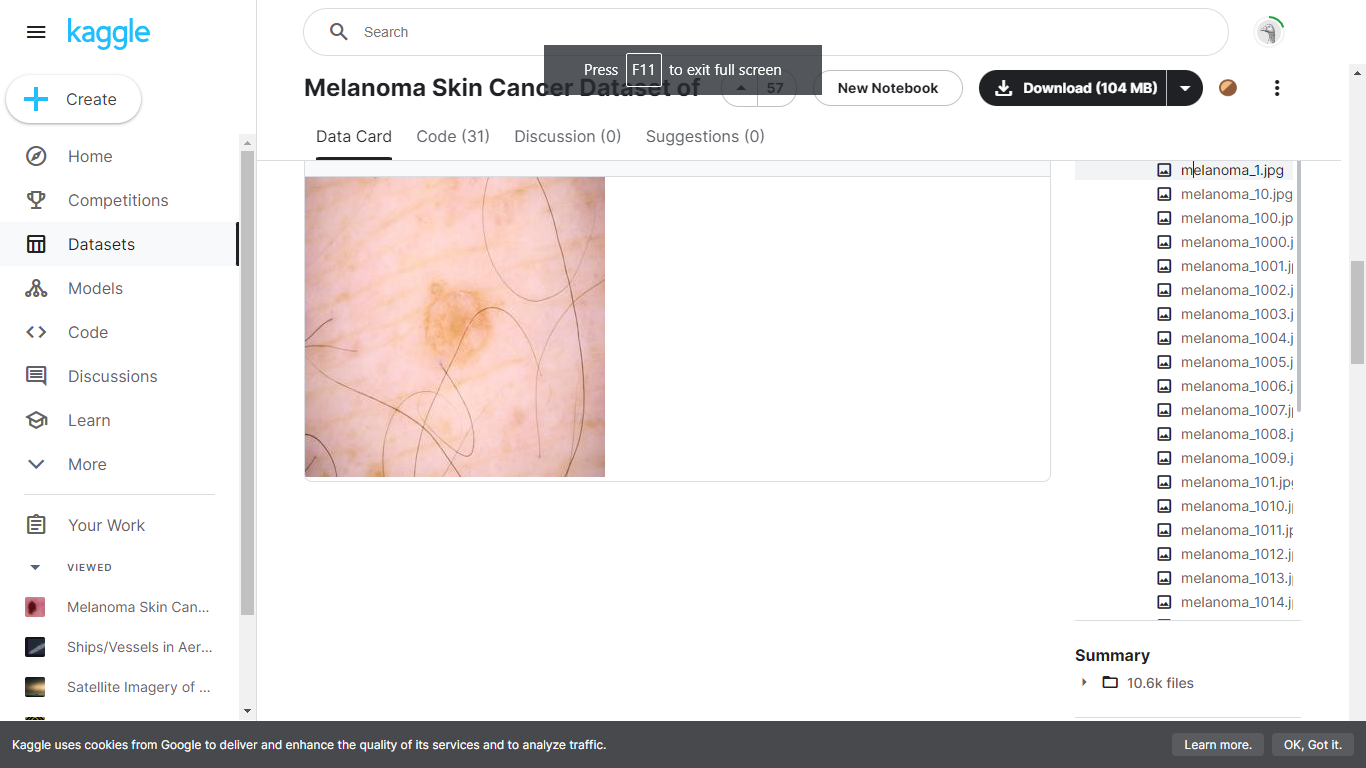
root.mainloop()

**5.3 DATASET USED**

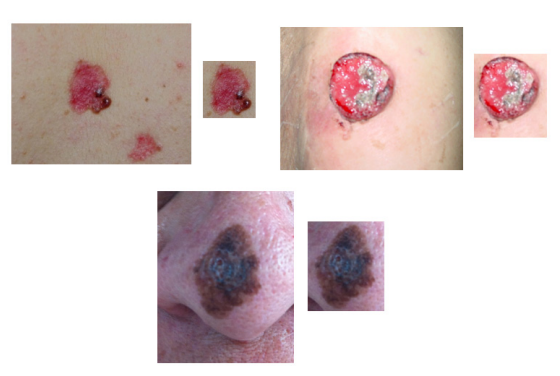
The dataset used here is melanoma skin cancer dataset functions where they can be gathered from the kaggle

Source: <https://www.kaggle.com/datasets/hasnainjaved/melanoma-skin-cancer-dataset-of-10000-images>

Melanoma Skin Cancer Dataset contains 10000 images. Melanoma skin cancer is deadly cancer, early detection and cure can save many lives. This dataset will be useful for developing the deep learning models for accurate classification of melanoma. Dataset consists of 9600 images for training the model and 1000 images for evaluation of model.

****

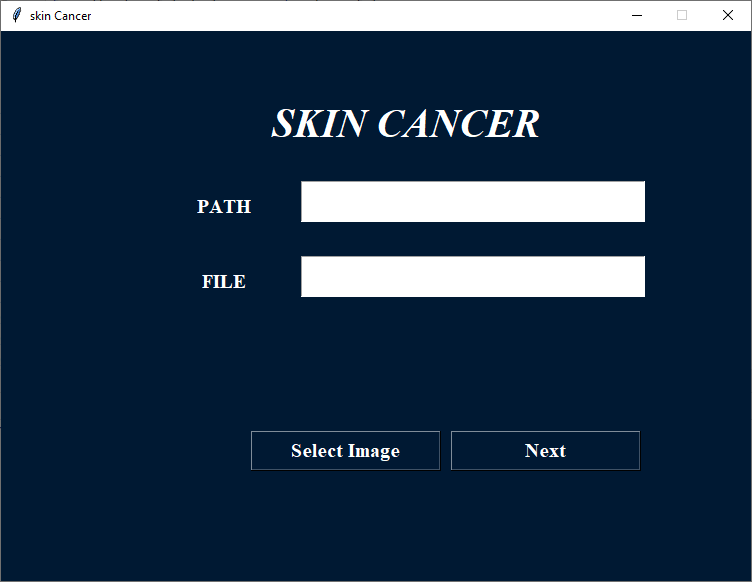
**Fig5.1skin cancer dataset**

****

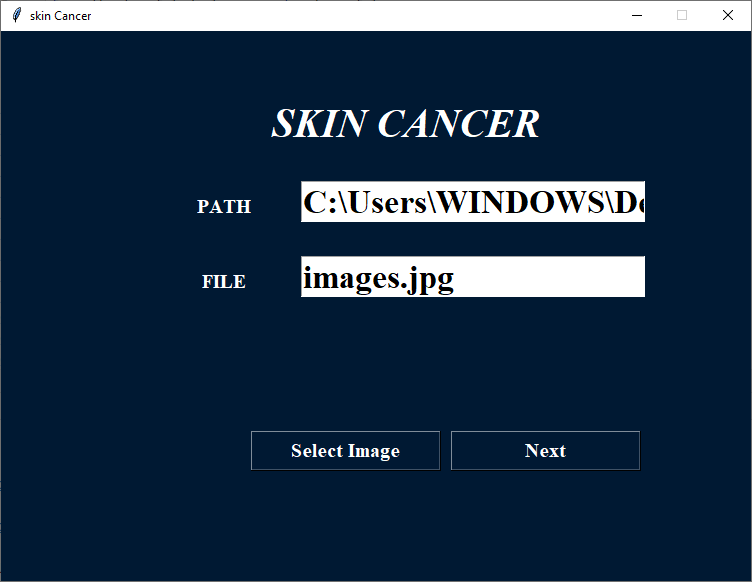
**Fig 5.2 Skin cancer Images**

**5.4 INPUT AND OUTPUT DESIGN**

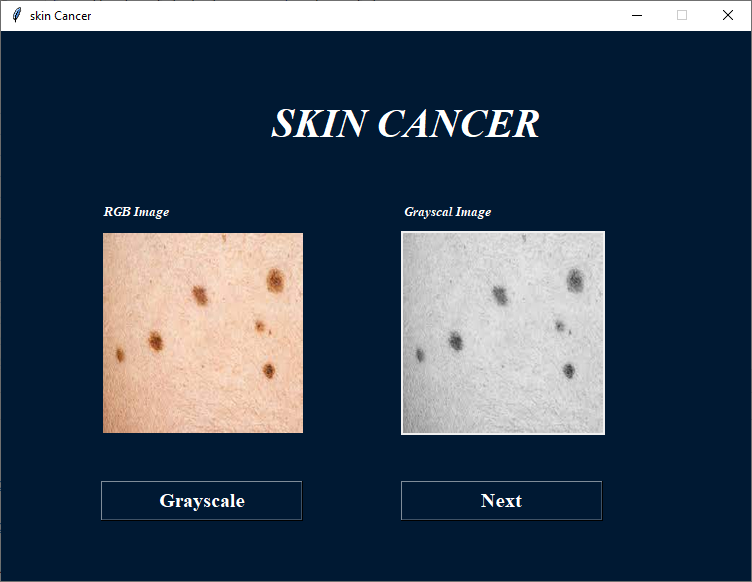
**5.4.1 INPUT DESIGN**

****

**Fig 5.2 Image loading of the input**

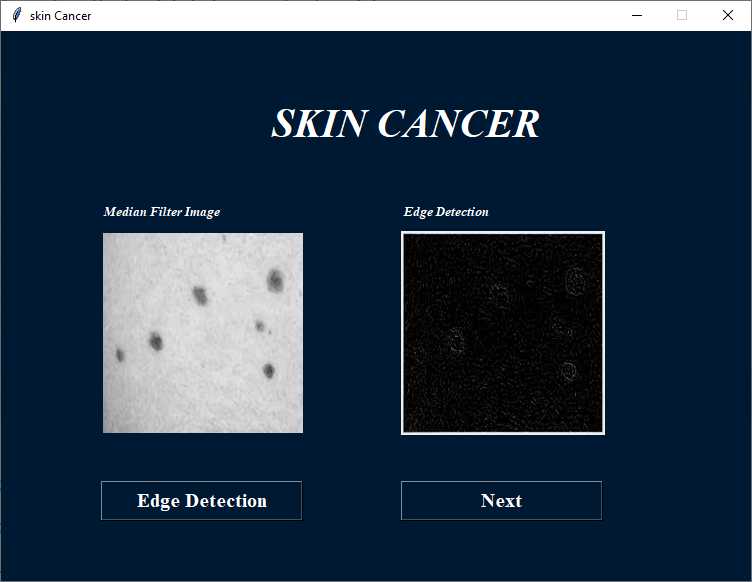


**Fig 5.3 Image loading**

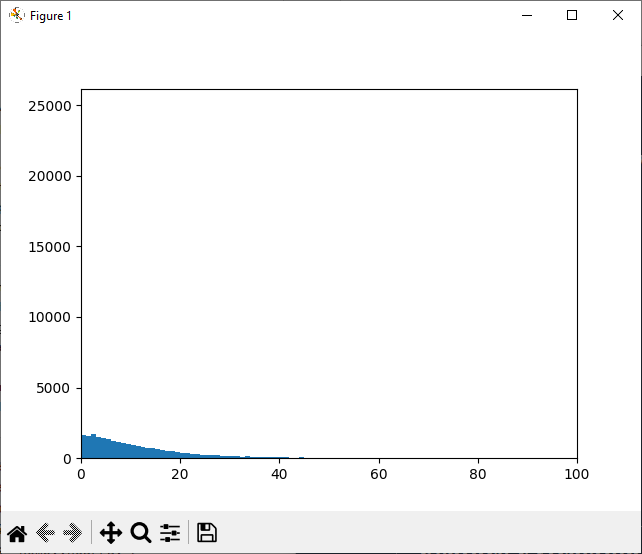


**Fig 5.4 Image view**

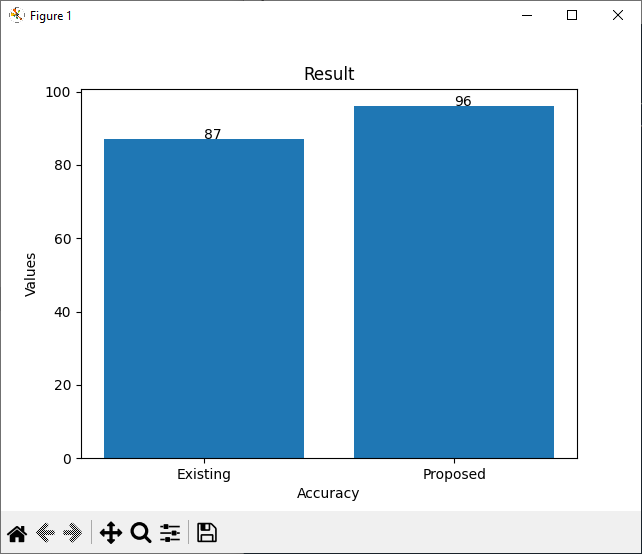
**5.4.2 OUTPUT DESIGN**

****

**Fig 5.5 Output design**

****

**Fig 5.6 Output design**

****

**Fig 5.7 Output design**

**5.5 RESULT AND DISCUSSION**

To begin with, the Skin Injury Cover is shaped and after that it is connected on the input picture to get the sectioned picture. As input picture of the skin injury is proficiently portioned for both mole as well as Melanoma Cancer Picture utilizing proposed division strategy.

To test the execution of proposed conclusion strategy we used 220 pictures, where 113 images are cancer and 107 image are non-cancer. Source of the pictures was (120 pictures from PH2 database and 100 images from the websites. After the division handle we compute the (TDS) for each picture. Agreeing to the esteem of (TDS) we classified all pictures accurately but 10 pictures (3 cancers and 7 non cancers) did not determination accurately

|  |  |  |  |
| --- | --- | --- | --- |
|  | Specificity | Sensitivity | Accuracy |
| Existing System | 82 | 81 | 86 |
| Proposed System | 97 | 94 | 98 |

**Table 5.1 comparison of the existing and proposed system**

**Fig 5.8 Comparison chart of the existing and proposed system**

The precision of proposed strategy compared with other strategies, the comes about appear in table 1. From table 1, it is obvious that the proposed strategy gives more accuracy to determination of cancer than the other strategies.

**CHAPTER 6**

**CONCLUSION AND RECOMMENDATION**

**6.1 CONCLUSION**

It can be easily concluded that the proposed system of skin cancer detection can be implemented using gray level co-occurrence matrix and support vector machine to classify easily whether image is cancerous or non-cancerous. Accuracy of proposed system is 95%. It is painless and timeless process than biopsy method. It is more advantageous to patients. This project we have discussed a computer-aided diagnosis system for melanoma skin cancer. It can be concluded from the results that the proposed system can be effectively used by patients and physicians to diagnose the skin cancer more accurately. This tool is more useful for the rural areas where the experts in the medical field may not be available. Since the tool is made more users friendly and robust for images acquired in any conditions, it can serve the purpose of automatic diagnostics of the Skin Cancer.

**6.2 RECOMMENDATION OF FUTURE WORK**

In future system, by comparing the first two techniques it is found that FURIA and takes less processing time, whereas ABCD method gives an accuracy of 90%. Third classification technique is the method called AIS (Artificial Immune System) using clonal selection method which is the future work.

**REFERENCES**

[1] A.W.Kopf, T.G. Salopek, J. Slade, A.A. Marghood, R.S. Bart, “Techniques of Cutaneous Examination for the Detection of Skin Cancer”, Cancer Supplement 75 (2) (1994) 684-690.

[2] I. Papamichail, N. Nikolaidis, I. Glava, K. Lentzas, Marmagkiolis, et al., “Merkel Cell Carcinoma of the Upper Extremity: Case Report and an Update”, J. Surg. On conf. 6(32) (2008). doi: 10.1186/1477-7819-6-32

[3] R. Siegel, D. Naishadham, and A. Jemal. “Cancer Statistics”, CA: A Cancer Journal for Clinicians, page 1029, 2012.

[4] National Institutes of Health. Sun Protection. “Cancer Trends Progress Report - 2009/2010”, Update, 2010.

[5] A. Jemal, R. Siegel, J. Xu, and E. Ward. “Cancer Statistics 2010”, CA: a cancer journal for Clinicians, 60(5):277-300, Epud, doi: 10.3322/caac.20073, 2010.

[6] Korotkov, K. & Garcia, R. (2012),“Computerized Analysis of Pigmented Skin Lesions:”, A review Artificial Intelligence in Medicine, Vol. 56, pp. 69-90.

[7] R. Sumithra, M. Suhil, and D. S. Guru, “Segmentation and Classification of Skin Lesions for Disease Diagnosis,” Procedia - Procedia Comput. Sci., vol. 45, pp. 76–85, 2015.

[8] N. Razmjooy, S. Mousavi et al, “A computer-aided diagnosis system for malignant melanomas,” Neural Computing & Applications, vol. 23, no. 7/8, pp. 2059– 2071, 2013. DOI: 10.1007/s00521-012-1149-1.

[9] M. A. Sheha, M.S.Mabrouk, A. Sharawy, “Automatic Detection of Melanoma Skin Cancer using Texture Analysis,” International Journal of Computer Applications, vol. 42, no. 20, pp. 22–26, 2012.

[10] W.Stolz, “ABCD rule of Dermatoscopy: Anew Practical Method forearly Recognition of Malignant Melanoma”, European Journal ofDermatology, Volume 4 (1994) 521–527.

**APPENDIX**

**A1. SOURCE CODE**

Ar\_master

import pymysql

class master\_flask\_code:

def \_\_init\_\_(self):

self.user = 'root'

self.password = ''

self.host = 'localhost'

self.database = 'python\_intelligent\_attendance'

def find\_max\_id(self,table):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute("SELECT id FROM "+table)

data = cursor.fetchall()

maxin = len(data)

if maxin == 0:

maxin = 1

else:

maxin += 1

return maxin

def insert\_query(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

result=cursor.execute(qry)

conn.commit()

conn.close()

return result

def select\_login(self,qry):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

cursor = conn.cursor()

cursor.execute(qry)

data = cursor.fetchall()

check = len(data)

if check == 0:

return 'no'

else:

return 'yes'

def select\_single\_colum(self,table,colum):

conn = pymysql.connect(user=self.user, password=self.password, host=self.host, database=self.database)

qry1=("select "+colum+" from "+table)

cursor = conn.cursor()

cursor.execute(qry1)

data = cursor.fetchall()

return data

def admin\_login(self,username,password):

if username == 'admin' and password == 'admin':

return 'yes'

else:

return 'no'

Median Filter

import shutil

from tkinter import \*

import os

import cv2

from PIL import ImageTk,Image

from tkinter.filedialog import askopenfilename

from PIL import Image, ImageFilter

import sample\_data

################################################################## read dataset

def read\_first\_data():

dd=0

lbl2.place(x=400, y=200)

# filapath=sample\_data.student.file\_path

# img = Image.open(filapath).convert('L')

# img.save('greyscale.png')

################################################################## Next page

def next\_page():

# image = Image.open(r"median.png")

# image = image.convert("L")

# image = image.filter(ImageFilter.FIND\_EDGES)

# image.save('img\_blur.png')

root.destroy()

import edge\_detection

# lbl2.configure(image=r2, background="#FFF")

# lbl2.place(x=400, y=200)

################################################################## main loop

im1 = Image.open(r"greyscale.png")

im2 = im1.filter(ImageFilter.MedianFilter(size=3))

im2.save('median.png')

root = Tk()

w=750

h=550

ws = root.winfo\_screenwidth()

hs = root.winfo\_screenheight()

x = (ws/2) - (w/2)

y = (hs/2) - (h/2)

root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample\_data.student.title)

root.resizable(False, False)

root.configure(background=sample\_data.student.background)

################################################################## components design

message = Label(root, text=sample\_data.student.titlec,fg=sample\_data.student.text\_color,bg=sample\_data.student.background, width=35,height=3, font=('times', 30, 'italic bold '))

message.place(x=00, y=20)

#######

message = Label(root, text="Grayscal Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=100, y=170)

message = Label(root, text="Median Filter Image",fg=sample\_data.student.text\_color,bg=sample\_data.student.background, font=('times', 10, 'italic bold '))

message.place(x=400, y=170)

ri2 = Image.open('greyscale.png')

ri2 = ri2.resize((200, 200), Image.ANTIALIAS)

r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample\_data.student.background)

lbl = Label(root, image=r2, background=sample\_data.student.background, fg=sample\_data.student.text\_color, font=('times', 15, ' bold '))

lbl.place(x=100, y=200)

lbl2 = Label(root)

a1 = Image.open('median.png')

a123 = a1.resize((200, 200), Image.ANTIALIAS)

a12 = ImageTk.PhotoImage(a123)

lbl2.configure(image=a12)

######## button with command function

compare\_dataset = Button(root, text="Median Filter",width=16,command=read\_first\_data ,height=1,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

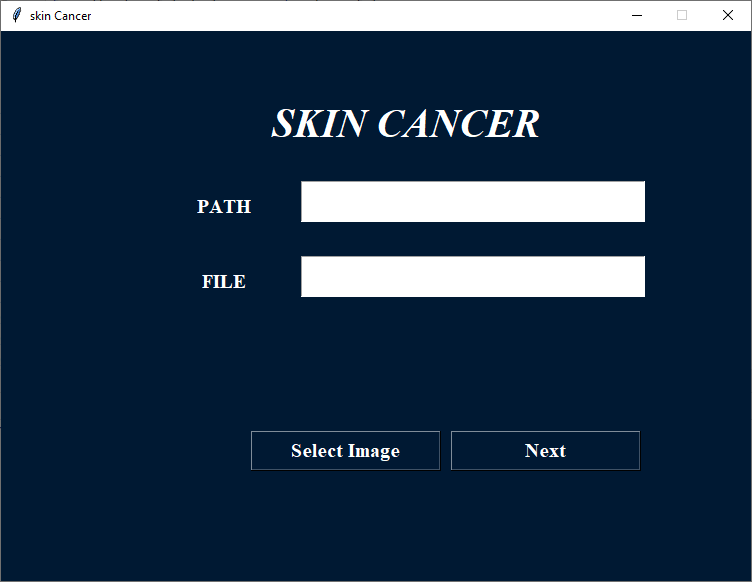
compare\_dataset.place(x=100, y=450)

resust\_dataset = Button(root, text="Next",width=16 ,height=1,command=next\_page,fg="#FFF",bg=sample\_data.student.background, activebackground = "#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))

resust\_dataset.place(x=400, y=450)

root.mainloop()

**A2. SCREENSHOT**

****

