**Skin Disease Prediction Using Machine Learning**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** |  |
|  | **INTRODUCTION** |  |
|  | * 1. General Introduction |  |
|  | * 1. Project Objectives |  |
|  | **SYSTEM** **PROPOSAL** |  |
|  | * 1. Existing System |  |
|  | * + 1. Disadvantages |  |
|  | * 1. Proposed System |  |
|  | 2.2.1 Advantages |  |
|  | * 1. Literature Survey |  |
|  | **SYSTEM DIAGRAMS** |  |
|  | * 1. Architecture Diagram |  |
|  | * 1. Flow Diagram |  |
|  | * 1. UML Diagrams |  |
|  | **IMPLEMENTATION** |  |
|  | * 1. Modules |  |
|  | * 1. Modules Description |  |
|  | **SYSTEM** **REQUIREMENTS** |  |
|  | * 1. Hardware Requirements |  |
|  | * 1. Software Requirements |  |
|  | * 1. Software Description |  |
|  | * 1. Testing of Products |  |
|  | **CONCLUSION** |  |
| **7.** | **FUTURE** **ENHANCEMENT** |  |
| **8.** | **SAMPLE** **CODING** |  |
| **9.** | **SAMPLE** **SCREENSHOT** |  |
| **10.** | **REFERENCES** |  |

**ABSTRACT**

Skin diseases are very common nowadays and spreading widely among people in present time. With the growth of computer-based technology and relevance of different machine learning methods in current decade, the development of skin disease prediction using classifier methods is analytical and exact. Therefore, development of data mining techniques can efficiently distinguish classes of skin disease is importance. A new method was developed using four types of classification methods which is used in this system. Skin diseases are a major global health problem associated with high number of people. With the rapid development of technologies and the application of various data mining techniques in recent years, the progress of dermatological predictive classification has become more and more predictive and accurate. Therefore, development of machine learning techniques, which can effectively differentiate skin disease classification, is of vast importance. The machine learning techniques applied to skin disease prediction so far, no techniques outperforms over all the others. We use skin disease dataset for analyzing various machine learning algorithms to classify the different classes of skin disease. The proposed data mining techniques were checked on skin disease datasets for analyzing 12 types of skin disease, which are Acne, Bowens, Chickenpox, Chiggers, and Dermatofibroma and so on. The system is developed the different machine learning algorithms such as random forest and decision tree. Finally, the experimental results shows that some performance metrics such as accuracy and error rate.

**CHAPTER 1**

**INTRODUCTION**

* 1. **General Introduction:**

The skin is the most significant part of human body. The skin protects the body from UV radiation infections, injuries, heat and harmful radiation, and also helps in the manufacture of vitamin D. The skin plays an important role in controlling body temperature, so it is important to maintain good health and protect the body from skin diseases.

The fast development of computer technology in present decades, the use of data mining technology plays a crucial role in the analysis of skin diseases. Researchers are constantly developing various prediction methods, but the largest researchers use only a few classification algorithms instead of ensemble methods. The ensemble method uses different data mining techniques and combines them to find predictions.

Melanoma is a malignant tumour which develops from the Pigment-containing cells known as melanocytes. It has the most rapidly increasing mortality rate among skin cancers. The American Cancer Society estimates that about 7,230 people are expected to die of melanoma and about 96,480 new melanomas is diagnosed in the United States in the year 2019. According to the statistics, the lifetime risk of getting melanoma is about 2.6% for whites, 0.1% for blacks, and 0.6% for Hispanics. Cutaneous melanoma is the most dangerous form of skin tumor that causes 90% of skin cancer mortality.

According to melanomas account for 90% of the deaths associated with cutaneous tumors. They also investigated that the incidence rate is around 25 new melanoma cases per 100,000 in Europe, and around 30 per 100,000 inhabitants in the United States. A lot of optical techniques exist for melanoma screening. These techniques are non-invasive, with fast response and are sensitive to biochemical and structural changes presented in skin cancer development. They include wide-field imaging, optical spectroscopy and microscopy imaging.

Skin Disease are occurring almost on all groups of ages among people. The rate of skin disease has been increased due to lifestyle and changing environments In the USA country, it is observed that every one out of five people are infected with any kind of skin disease.

They are usually caused by factors like different organism’s cells, a different diet, and internal and external factors, such as the hierarchical genetic group of cells, hormones, and immune system of conditions. These factors may act together or in a sequence of skin disease. There are chronic and incurable diseases, like eczema and psoriasis, and malignant diseases like malignant melanoma. Recent researchers have found the availability of cures for these diseases if they are detected in the early stages.

From the literature survey, authors of this paper found that, the creation of an expert application of skin disease detection using methods like Naive Bayes, CNN, SVM methods was felt to be very necessary to help all people who want to know about skin diseases that are being experienced or need information about skin diseases . To detect these diseases using the image processing method many research papers has been published and many researchers has contributed a lot which paved a way for our application and gave us a right direction. Without the previous works of these fellow researchers my work on developing any application would never have been easier .

Skin diseases are often quite hard to detect at an early stage and it is even harder to classify them separately. Recently, it is well known that, the most dangerous form of skin cancer among the other types of skin cancer is melanoma because it is much more likely to spread to other parts of the body if not diagnosed and treated early. In order to classify these skin diseases, “Support Vector Machine (SVM)” a Machine Learning Algorithm can be used Image classification is one of classical problems of concern in image processing. Support Vector Machine are classified under supervised learning models and is a part of machine learning algorithm which used to analyze structured and unstructured data such as text and images. As an input SVM always requires clean data. In skin disease detection, classifying the images into different types of skin diseases is the problem.

This paper gives us the complete overview on existing machine learning and image processing algorithms for detection of skin disease through android application development. It makes use of skin disease dataset for the analysis. From the literature review, it is found that CNN and SVM are the most suitable algorithm for skin disease detections.

Skin disease has more touchiness as compared to any other disease. Regular skin issues are dermatitis. The main focus of this research paper will be on dermatology database which contains different eryhemato-squamous diseases class as psoriasis, seboreic dermatitis, lichen planus, pityriasisrosea, cronic dermatitis and pityriasisrubrapilaris. Each record is a collection of 33 attributes which are linear values and one attribute of them is nominal. The 75% of the dataset utilise for demonstrating and keep down 25% for approval. The purpose of this article is to achieve the best-performing classifier that will communicate in the collection of dermatological information. Therefore, k-nearest neighbours and support vector machines are used. By using ten-fold cross validation and assess calculations utilising the accuracy metric. This is a gross metric which will prove the developed model is best one.

**1.2 Objectives:**

The main objective of our project is,

* To detect or to classify the skin disease effectively.
* To implement the machine learning algorithm such as DT and RF.
* To enhance the overall performance for classification algorithms.
* To implement the web application.

**CHAPTER 2**

**SYSTEM PROPOSAL**

**2.1 EXISTING SYSTEM:**

In existing system, melanoma detected by removing the part of effected skin in our body and send to check whether the cancer cells are present or not. If the cancer cells are present will checking then it will diagnosed. It is a time taken process and also it make patient life in risk. Melanoma detection using machine learning is difficult to handle large data so it may not provide best accuracy. It is technically challenging because of the large amount data collect by Images. So that, we bring a concept of melanoma detection using deep learning because it can handle large number of data set. Among state-of-the-art methods used for automated or computer assisted medical diagnosis, attention should be drawn to Deep Learning based on Convolutional Neural Networks, wherewith segmentation, classification and detection systems for several diseases have been implemented. The method proposed in this paper involves an initial stage that automatically crops the region of interest within a dermatoscopic image using the Mask and Region based Convolutional Neural Network technique, and a second stage based on a ResNet152 structure, which classifies lesions as either “benign” or “malignant”.

**2.1.1 DISADVANTAGES:**

* It doesn’t efficient for large volume of data’s.
* The performance is low.
* Time consumption is high.
* Training time is more.
  1. **PROPOSED SYSTEM:**

In this system, the skin disease dataset is collected from dataset repository. Then, we have to implement the image pre-processing step. In this step, we have to implement image resize and grey scale conversion. Then, we have to implement the image splitting into train is used for evaluation and test is used for prediction. After that, we have to implement the machine learning algorithm such as decision tree and random forest. The experimental results shows that the accuracy and error rate. Finally, we have to classify or predict the disease whether it is affected or not by using classification algorithm. Then, we can implement the framework for user friendly.

**2.2.1 ADVANTAGES:**

* It is efficient for large number of datasets.
* Time consumption is low.
* The training time is low.
* Prediction is more accurate.

**2.3 LITERATURE SURVEY:**

# **2.3.1 An improved border detection in dermoscopy images for density based clustering, 2011**

**Author*:*** S. Suer, S. Kockara, and M. Mete

# **Methodology:**

Dermoscopy is one of the major imaging modalities used in the diagnosis of melanoma and other pigmented skin lesions. In current practice, dermatologists determine lesion area by manually drawing lesion borders. Therefore, automated assessment tools for dermoscopy images have become an important research field mainly because of inter- and intra-observer variations in human interpretation. One of the most important steps in dermoscopy image analysis is automated detection of lesion borders. To our knowledge, in our 2010 study we achieved one of the highest accuracy rates in the automated lesion border detection field by using modified density based clustering algorithm. In the previous study, we proposed a novel method which removes redundant computations in well-known spatial density based clustering algorithm, DBSCAN; thus, in turn it speeds up clustering process considerably.

**Advantage:**

* It works on color images without any pre-processing and generates more accurate results than existing method.

**Disadvantage:**

* It cannot find any point density-reachable from the starting point.

**2.3.2 Digital monitoring by whole body photography and sequential digital dermoscopy detects thinner melanomas, 2010**

**Author**: M. Rademaker and A. Oakley

**Methodology:**

The value of digital epiluminescence microscopy (DELM) for the long-term follow-up of atypical nevi. Patients (*n*=530) were prospectively categorized into defined melanoma risk groups and followed by clinical and epiluminescence microscopy (ELM) examinations. Atypical nevi (*n*=7001) were additionally followed by DELM. During follow-up (median 32.2 months), we detected 53 melanomas among 637 excised lesions (8.3% overall chance of success). The chance of success for melanoma detection among lesions suspicious by ELM criteria was increased to 17% when additional DELM-documented changes were present. Moreover, 18 of the 53 melanomas were exclusively identified by DELM-documented changes, indicating that DELM increased the sensitivity of the ELM analysis by identifying additional melanomas. However, for lesions exclusively excised due to DELM changes, the chance of success was lower than for ELM (5.2 *vs* 11.8%). Excisions due to mere DELM changes detected 66.7% of melanomas in familial atypical mole and multiple melanoma (FAMMM) and 32.5% of melanomas in atypical mole syndrome (AMS) patients. We conclude that DELM is a valuable tool for the long-term follow-up of atypical nevi, especially in the high-risk groups of FAMMM and AMS patients. Randomized controlled trials are needed to validate the data from this clinical trial.

**Advantage:**

* It remains to be determined whether earlier diagnosis results in improved survival.

**Disadvantage:**

* It may not be detected until they are quite advanced.

**2.3.3: SKINcure: A real time image analysis system to aid in the malignant melanoma prevention and early detection, 2014**

**Author:** O. Abuzaghleh, B. D. Barkana, and M. Faezipour,

**Methodology:**

This paper proposes an innovative and fully functional smart-phone based  
application to assist in melanoma early detection and prevention. The application has two major components; the first component is a real-time alert to help users prevent skin burn caused by sunlight; a novel equation to compute the time for skin to burn is thereby introduced. The second component is an automated image analysis module which contains image acquisition, hair detection and exclusion, lesion segmentation, feature extraction, and classification. The proposed system exploits PH2 Dermoscopy image database from Pedro Hispano Hospital for development and testing purposes. The image database contains a total of 200 dermoscopy images of lesions, including normal, atypical, and melanoma cases.  
The experimental results show that the proposed system is efficient, achieving classification of the normal, atypical and melanoma images with accuracy of 96.3%, 95.7% and 97.5%, respectively.

**Advantage:**

* Portability and low cost. And can make a significant impact on health care delivery as assistive devices in underserved and remote areas.

**Disadvantage:**

* Their system didn’t allow the user to capture images using the smart phone.

**2.3.4: SkinScan: A portable library for melanoma detection on handheld  
devices, 2011**

**Author:** T. Wadhawan, N. Situ, K. Lancaster, X. Yuan, and G. Zouridakis

**Methodology:**

This paper is about to develop a portable library for automated detection of melanoma termed SkinScan that can be used on smartphones and other handheld devices. Compared to desktop computers, embedded processors have limited processing speed, memory, and power, but they have the advantage of portability and low cost. In this study we explored the feasibility of running a sophisticated application for automated skin cancer detection on an Apple iPhone 4. Our results demonstrate that the proposed library with the advanced image processing and analysis algorithms has excellent performance on handheld and desktop computers. Therefore, deployment of smartphones as screening devices for skin cancer and other skin diseases can have a significant impact on health care delivery in underserved and remote areas.

**Advantage:**

* It is possible to run sophisticated biomedical imaging applications on smartphones and other handheld devices, which have the advantage of portability and low cost, and therefore they can make a significant impact on health care delivery as assistive devices in underserved and remote areas.

**Disadvantage:**

* User cannot use this on other Portable device.

**2.3.5: A mobile automated skin lesion classification system, 2011**

**Author:** K. Ramlakhan and Y. Shang

**Methodology:**

Melanoma skin cancer accounts for less than 5% of skin cancer cases but causes the most deaths due to skin cancer. Convenient automated diagnosis of skin lesions and melanoma recognition can greatly improve early detection of melanomas. This paper presents a prototype of an image-based automated melanoma recognition system on Android smart phones. The system consists of three major components: image segmentation, feature calculation, and classification. It is designed to run on a mobile device with a camera, such as a smart phone or a tablet PC. A skin lesion image is converted to a monochrome image for outline contour detection. Color and shape features of the lesion are extracted and used as input to a kNN classifier. Initial experimental result shows that the system is efficient and works well on well-lighted test images, achieving an average accuracy of 66.7%, with average malignant class recall/sensitivity of 60.7% and specificity of 80.5%. New sensors allow simultaneous acquisition of 3D shape and colour data of skin at resolutions theoretically approaching cellular structures. We investigate whether the addition of 3D depth information increases classification rates relative to only using colour information for 5 non-melanoma skin lesions. The paper demonstrates that there is 6% increase in classification rates.

**Advantage:**

* Color and shape features of the lesion are extracted and used as input to a KNN classifier.

**Disadvantage:**

* It does not identified any previous non-Edinburgh research using 3D depth data for skin cancer analysis.

**CHAPTER 3**

**SYSTEM DIAGRAMS**

**3.1 SYSTEM ARCHITECTURE:**

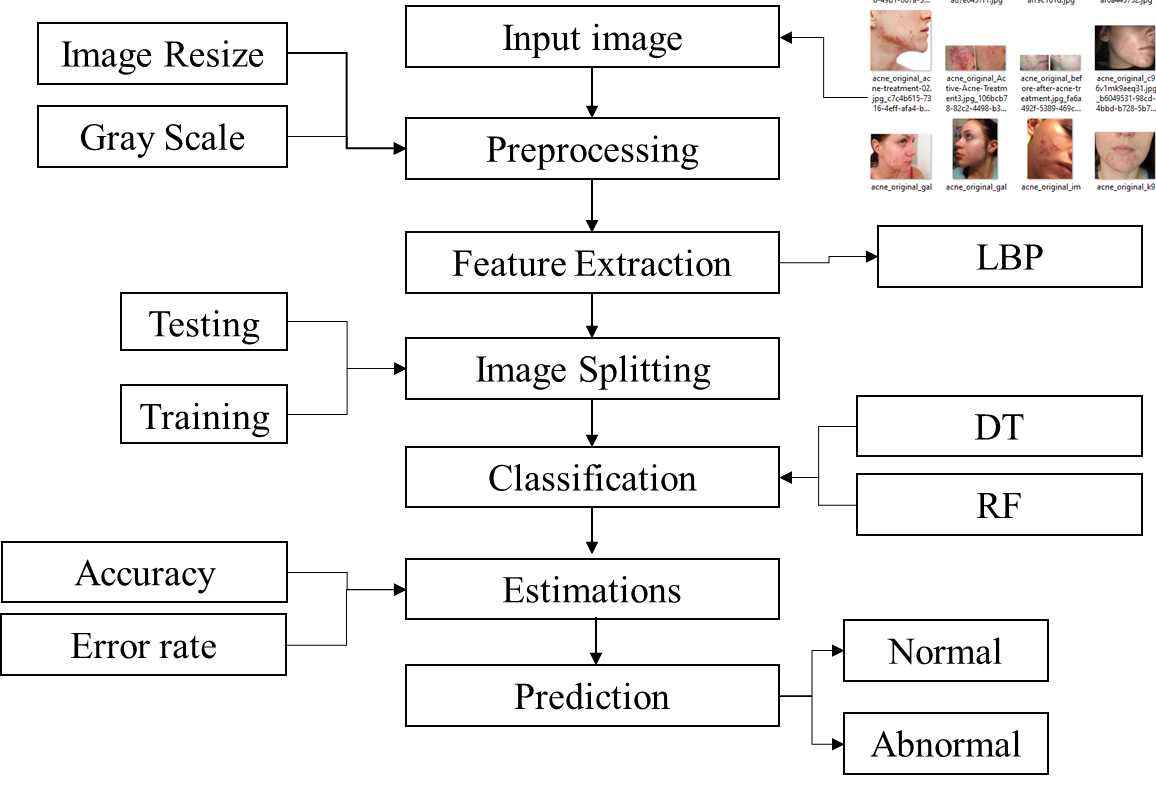


FIGURE 3.1: SYSTEM ARCHITECTURE

**3.2 FLOW DIAGRAM**

Input Image

Preprocessing

Feature Extraction

Classification

Performance analysis

FIGURE 3.2: FLOW DIAGRAM

**3.3 UML DIAGRAMS:**

**3.3.1 USE CASE DIAGRAM:**

System

User

FIGURE 3.3.1: USE CASE DIAGRAM

**3.3.2 ACTIVITY DIAGRAM:**

Input Image

Preprocessing

Feature Extraction

Performance metrics

Classification

FIGURE 3.3.2: ACTIVITY DIAGRAM

**3.3.3 SEQUENCE DIAGRAM:**

Input Image

Preprocessing

Feature extraction

Classification

Select image

Image Resize

DT and RF

Load Image

Grayscale Conversion

Accuracy

FIGURE 3.3.3: SEQUENCE DIAGRAM

**3.3.4 ER DIAGRAM:**

Image selection

Preprocessing

Feature extraction

Classification

FIGURE 3.3.4: ER DIAGRAM

**3.3.5 CLASS DIAGRAM:**

Select Image ()

Load Image ()

View Image ()

INPUT

LBP ()

FEATURE EXTRACTION

Accuracy ()

PERFORMANCE ANALYSIS

PREPROCESSING

Resize image ()

Gray Scale Conversion ()

RF ()

DT ()

CLASSIFICATION

FIGURE 3.3.5: CLASS DIAGRAM

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 MODULES:**

* Input image
* Preprocessing
* Feature Extraction
* Data splitting
* Classification
* Performance Estimation

**4.2 MODULES DESCRIPTION:**

**4.2.1: IMAGE SELECTION:**

* The dataset, **skin disease dataset** is implemented as input. The dataset is taken from dataset repository. The input dataset is in the format ‘.png, ‘.jpg.
* In this step, we have to read or load the input image by using the imread () function.
* The input image is used to detect or classify the input image.
* In our process, we are used the tkinter file dialogue box for selecting the input image.

**4.2.2: IMAGE PREPROCESSING:**

* In our process, we have to resize the image and convert the image into gray scale.
* To **resize an image**, you call the resize () method on it, passing in a two-integer tuple argument representing the width and height of the resized image.
* The function doesn't modify the used image; it instead returns another Image with the new dimensions.
* Convert an Image to **Grayscale** in Python Using the Conversion Formula and the matplotlib Library.
* We can also convert an image to grayscale using the standard RGB to grayscale conversion formula that is imgGray = 0.2989 \* R + 0.5870 \* G + 0.1140 \* B.

**4.2.3 FEATURE EXTRACTION:**

* In our process, we have to extract the features from preprocessed image.
* Local Binary Pattern (LBP) is a method that used to describe texture characteristics of the surfaces. By applying LBP, texture pattern probability can be summarised into a histogram. LBP values need to be determined for all of the image pixels.

**4.2.4: IMAGE SPLITTING:**

* During the machine learning process, data are needed so that learning can take place.
* In addition to the data required for training, test data are needed to evaluate the performance of the algorithm in order to see how well it works.
* In our process, we considered 70% of the input dataset to be the training data and the remaining 30% to be the testing data.
* Data splitting is the act of partitioning available data into two portions, usually for cross-validator purposes.
* One Portion of the data is used to develop a predictive model and the other to evaluate the model's performance.
* Separating data into training and testing sets is an important part of evaluating data mining models.
* Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.

**4.2.5: CLASSIFICATION:**

* In our process, we have to implement the two different machine learning algorithm such as DT and RF.
* A **decision tree** is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.
* **Random forest** is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

**4.2.6: RESULT GENERATION:**

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like

* **Accuracy** of classifier refers to the ability of classifier. It predicts the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

AC= (TP+TN)/ (TP+TN+FP+FN)

* Then, we can detect or to classify the input image is affected by skin disease or not.

**CHAPTER 5**

**SYSTEM REQUIREMENTS**

**5.1 HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz
* Hard Disk : 200 GB
* Mouse : Logitech.
* Keyboard : 110 keys enhanced
* Ram : 4GB

**5.2 SOFTWARE REQUIREMENTS:**

* O/S : Windows 7.
* Language : Python
* Front End : Anaconda Navigator – Spyder

**5.3 SOFTWARE DESCRIPTION:**

**5.3.1 Python**

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

## **5.3.2 Features of Python**

### **Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### **Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### **Free and Open Source**

Python is an example of a FLOSS (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

### **High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

### **Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like [Kivy](http://kivy.org) to create games for your computer and for iPhone, iPad, and Android.

### **Interpreted**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

### **Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

### **Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

### **Embeddable**

You can embed Python within your C/C++ programs to give scripting capabilities for your program's users.

### **Extensive Libraries**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

**5.4 TESTING PRODUCTS:**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. . A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

**5.4.1 UNIT TESTING:**

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’.

The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

**5.4.2 INTEGRATION TESTING:**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

i) Top-down integration testing. ii) Bottom-up integration testing.

**5.4.3 TESTING TECHNIQUES/STRATEGIES:**

* **WHITE BOX TESTING:**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we Derived test cases that guarantee that all independent paths within a module have been exercised at least once.

* **BLACK BOX TESTING:**

1. Black box testing is done to find incorrect or missing function
2. Interface error
3. Errors in external database access
4. Performance errors.
5. Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

**5.4.4 SOFTWARE TESTING STRATEGIES**

**VALIDATION TESTING:**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many,

But a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer

**USER ACCEPTANCE TESTING:**

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

**OUTPUT TESTING**:

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

**CHAPTER 6**

**CONCLUSION**

We conclude that, the dataset was taken from dataset repository. We are extracted the features from pre-processed image by using LBP. We are developed the machine learning algorithms such as decision tree and random forest. Finally, the experimental results shows that accuracy and error rate. Then, we are predicted or classified the disease is either affected or not.

**CHAPTER 7**

**FUTURE ENHANCEMENT**

In future work, we will hybrid the transfer learning or combine the two different machine learning algorithms or combine the two different deep learning algorithms for better performance or efficiency. This will help in detection of skin disease in rural parts of India where there is already a huge lack of basic medical facilities.

**CHAPTER 8**

**SAMPLE CODING**

import streamlit as st

import base64

# ================ Background image ===

def add\_bg\_from\_local(image\_file):

with open(image\_file, "rb") as image\_file:

encoded\_string = base64.b64encode(image\_file.read())

st.markdown(

f"""

<style>

.stApp {{

background-image: url(data:image/{"png"};base64,{encoded\_string.decode()});

background-size: cover

}}

</style>

""",

unsafe\_allow\_html=True

)

add\_bg\_from\_local('8.jpg')

def navigation():

try:

path = st.experimental\_get\_query\_params()['p'][0]

except Exception as e:

st.error('Please use the main app.')

return None

return path

if navigation() == "home":

st.title("Symptoms for Avilable Disease")

# st.title('Home')

# ================== REMEDIES ===========================

col1, col2, col3, col4, col5 = st.columns(5)

col6, col7, col8, col9, col10 = st.columns(5)

with col1:

st.text("ACNE")

st.image("Acne.jpg")

st.text("SYMPTOMS")

st.text("1.Crusting of skin bumps")

st.text("2.Cysts")

st.text("3.Papules ")

with col2:

st.text("Bowens")

st.image("Bowens.jpg")

st.text("SYMPTOMS")

st.text("1.Scaly skin")

st.text("2.Slow-growing")

st.text("3.Bleed ")

with col3:

st.text("ChickenBox")

st.image("ChickenBox.jpg")

st.text("SYMPTOMS")

st.text("1.Raised bumps")

st.text("2.fever")

st.text("3.Small fluid-filled blisters ")

with col4:

st.text("Chiggers")

st.image("Chiggers.jpg")

st.text("SYMPTOMS")

st.text("1.Red spots")

st.text("2.Severe itch")

st.text("3.Bites ")

with col5:

st.text("Eczema")

st.image("Eczema.jpg")

st.text("SYMPTOMS")

st.text("1.Dry, cracked skin")

st.text("2.Itchiness ")

st.text("3.Thickened skin")

elif navigation()=='login':

st.title("Welcome Login Page !!!")

import pandas as pd

# df = pd.read\_csv('login\_record.csv')

# Store the initial value of widgets in session state

if "visibility" not in st.session\_state:

st.session\_state.visibility = "visible"

st.session\_state.disabled = False

col1, col2 = st.columns(2)

with col1:

UR1 = st.text\_input("Login User Name",key="username")

psslog = st.text\_input("Password",key="password",type="password")

# tokenn=st.text\_input("Enter Access Key",key="Access")

agree = st.checkbox('LOGIN')

if agree:

try:

df = pd.read\_csv(UR1+'.csv')

U\_P1 = df['User'][0]

U\_P2 = df['Password'][0]

if str(UR1) == str(U\_P1) and str(psslog) == str(U\_P2):

st.success('Successfully Login !!!')

else:

st.write('Login Failed!!!')

except:

st.write('Login Failed!!!')

with col2:

UR = st.text\_input("Register User Name",key="username1")

pss1 = st.text\_input("First Password",key="password1",type="password")

pss2 = st.text\_input("Confirm Password",key="password2",type="password")

# temp\_user=[]

# temp\_user.append(UR)

if pss1 == pss2 and len(str(pss1)) > 2:

import pandas as pd

import csv

# field names

fields = ['User', 'Password']

# st.text(temp\_user)

old\_row = [[UR,pss1]]

# writing to csv file

with open(UR+'.csv', 'w') as csvfile:

# creating a csv writer object

csvwriter = csv.writer(csvfile)

# writing the fields

csvwriter.writerow(fields)

# writing the data rows

csvwriter.writerows(old\_row)

st.success('Successfully Registered !!!')

else:

st.write('Registeration Failed !!!')

elif navigation() == "about":

st.title("Welcome to About us Page !!!")

st.write("To develop an effective and efficient model which detects and predicts skin disease based on the user input. To achieve good accuracy. To develop a User Interface( UI ) that is user-friendly and takes input from the user and predicts the skin disease In future, this machine learning model may bind with a various website which can provide real-time data for skin disease prediction. Also, we may add large historical data on skin diseasewhich can help to improve the accuracy of the machine learning model. We can build an android app as a user interface for interacting with the user. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates, and train it on clusters of data ratherthan the whole dataset")

elif navigation() == "analysis":

st.title("Welcome to Prediction Page !!!")

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import warnings

warnings.filterwarnings('ignore')

from sklearn import preprocessing

import streamlit as st

import cv2

from PIL import Image

import matplotlib.image as mpimg

import matplotlib.pyplot as plt

import base64

# ================ INPUT IMAGE ======================

file\_up = st.file\_uploader("Upload an image", type="jpg")

# if file\_up==None:

# st.text("Browse")

# else:

# st.image(file\_up)

img = mpimg.imread("D:/Users/EGC/Downloads/streamlit\_navbar-main (1)/streamlit\_navbar-main/Data/Acne/acne\_original\_1rlnayp9x3n21111.jpg\_8fd49628-cd08-49b1-867a-3160a6e8a0b6.jpg")

st.image(img)

# ========= PREPROCESSING ============

img\_resize\_orig = cv2.resize(img,((50, 50)))

try:

gray1 = cv2.cvtColor(img\_resize\_orig, cv2.COLOR\_BGR2GRAY)

except:

gray1 = img\_resize\_orig

import os

# ========= IMAGE SPLITTING ============

data\_acne = os.listdir('Data/Acne/')

data\_bowens = os.listdir('Data/Bowens/')

data\_chick = os.listdir('Data/Chicken\_pox/')

data\_chiggers = os.listdir('Data/Chiggers/')

data\_dermto = os.listdir('Data/Dermatofibroma/')

data\_eczema = os.listdir('Data/Eczema/')

data\_entero = os.listdir('Data/Enterovirus/')

data\_kera = os.listdir('Data/Keratosis/')

data\_Meas = os.listdir('Data/Measles/')

data\_psor = os.listdir('Data/Psoriasis/')

data\_ringworm = os.listdir('Data/Ringworm/')

data\_scab = os.listdir('Data/Scabies/')

import numpy as np

dot1= []

labels1 = []

for img11 in data\_acne:

# print(img)

img\_1 = mpimg.imread('Data/Acne//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(1)

for img11 in data\_bowens:

# print(img)

img\_1 = mpimg.imread('Data/Bowens//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(2)

for img11 in data\_chick:

# print(img)

img\_1 = mpimg.imread('Data/Chicken\_pox//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(3)

for img11 in data\_chiggers:

# print(img)

img\_1 = mpimg.imread('Data/Chiggers//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(4)

for img11 in data\_dermto:

# print(img)

img\_1 = mpimg.imread('Data/Dermatofibroma//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(5)

for img11 in data\_eczema:

# print(img)

img\_1 = mpimg.imread('Data/Eczema//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(6)

for img11 in data\_entero:

# print(img)

img\_1 = mpimg.imread('Data/Enterovirus//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(7)

for img11 in data\_kera:

# print(img)

img\_1 = mpimg.imread('Data/Keratosis//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(8)

for img11 in data\_Meas:

# print(img)

img\_1 = mpimg.imread('Data/Measles//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(9)

for img11 in data\_psor:

# print(img)

img\_1 = mpimg.imread('Data/Psoriasis//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(10)

for img11 in data\_ringworm:

# print(img)

img\_1 = mpimg.imread('Data/Ringworm//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(11)

for img11 in data\_scab:

# print(img)

img\_1 = mpimg.imread('Data/Scabies//' + "/" + img11)

img\_1 = cv2.resize(img\_1,((50, 50)))

try:

gray = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2GRAY)

except:

gray = img\_1

dot1.append(np.array(gray))

labels1.append(12)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(dot1,labels1,test\_size = 0.2, random\_state = 101)

print("------------------------------------------------------------")

print(" Image Splitting")

print("------------------------------------------------------------")

print()

print("The Total of Images =",len(dot1))

print("The Total of Train Images =",len(x\_train))

print("The Total of Test Images =",len(x\_test))

# ===== CLASSIFICATION ======

from keras.utils import to\_categorical

x\_train11=np.zeros((len(x\_train),50))

for i in range(0,len(x\_train)):

x\_train11[i,:]=np.mean(x\_train[i])

x\_test11=np.zeros((len(x\_test),50))

for i in range(0,len(x\_test)):

x\_test11[i,:]=np.mean(x\_test[i])

y\_train11=np.array(y\_train)

y\_test11=np.array(y\_test)

train\_Y\_one\_hot = to\_categorical(y\_train11)

test\_Y\_one\_hot = to\_categorical(y\_test)

# === RF ===

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

rf.fit(x\_train11,y\_train11)

y\_pred = rf.predict(x\_train11)

from sklearn import metrics

accuracy\_test=metrics.accuracy\_score(y\_pred,y\_train11)\*100

print(accuracy\_test)

accuracy\_train=metrics.accuracy\_score(y\_train11,y\_train11)\*100

print(accuracy\_train)

acc\_overall\_rf=(accuracy\_test + accuracy\_train)/2

print("-------------------------------------")

print("PERFORMANCE ---------> (RF)")

print("-------------------------------------")

print()

print("1. Accuracy =", acc\_overall\_rf,'%')

print()

print("2. Error Rate =",100-acc\_overall\_rf)

# === DT ===

from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()

dt.fit(x\_train11,y\_train11)

y\_pred = dt.predict(x\_train11)

from sklearn import metrics

accuracy\_test=metrics.accuracy\_score(y\_pred,y\_train11)\*100

accuracy\_train=metrics.accuracy\_score(y\_train11,y\_train11)\*100

acc\_overall\_dt=(accuracy\_test + accuracy\_train)/2

print("-------------------------------------")

print("PERFORMANCE ---------> (DT)")

print("-------------------------------------")

print()

print("1. Accuracy =", accuracy\_test,'%')

print()

print("2. Error Rate =",100-acc\_overall\_dt)

Total\_length = len(data\_acne) + len(data\_bowens) + len(data\_chick) + len(data\_chiggers) + len(data\_dermto) + len(data\_eczema) + len(data\_entero) + len(data\_kera)+ len(data\_Meas) + len(data\_psor) + len(data\_ringworm) + len(data\_scab)

temp\_data1 = []

for ijk in range(0,Total\_length):

# print(ijk)

temp\_data = int(np.mean(dot1[ijk]) == np.mean(gray1))

temp\_data1.append(temp\_data)

temp\_data1 =np.array(temp\_data1)

zz = np.where(temp\_data1==1)

if labels1[zz[0][0]] == 1:

print('------------------------')

print()

print(' The Prediction = Acne')

print()

print('------------------------')

res1=" Affected by Acne"

st.text("--------------------")

st.write("Identified = ",res1)

st.text("--------------------")

elif labels1[zz[0][0]] == 2:

print('--------------------------')

print()

print('The Prediction = Bowens')

print()

print('-------------------------')

st.text("--------------------")

res1=" Affected by Bowens"

st.write("Identified = ",res1)

st.text("--------------------")

# st.text(res1)

elif labels1[zz[0][0]] == 3:

print('--------------------------')

print()

print('The Prediction = Chicken\_pox')

print()

print('-------------------------')

st.text("--------------------")

st.write("Identified = ",res1)

# res1=" Affected by Chicken\_pox"

st.text(res1)

st.text("--------------------")

elif labels1[zz[0][0]] == 4:

print('--------------------------')

print()

print('The Prediction = Chiggers')

print()

print('-------------------------')

res1=" Affected by Chiggers "

st.text(res1)

st.write("Identified = ",res1)

st.text("--------------------")

elif labels1[zz[0][0]] == 5:

print('--------------------------')

print()

print('The Prediction = Dermatofibroma')

print()

print('--------------------------------')

res1=" Affected by Dermatofibroma"

st.write("Identified = ",res1)

st.text("--------------------")

st.text(res1)

elif labels1[zz[0][0]] == 6:

print('--------------------------')

print()

print('The Prediction = Eczema')

print()

print('-------------------------')

res1=" Affected by sEczema"

st.text(res1)

elif labels1[zz[0][0]] == 7:

print('--------------------------')

print()

print('The Prediction = Enterovirus')

print()

print('-------------------------')

res1=" Affected by Enterovirus"

st.text(res1)

elif labels1[zz[0][0]] == 8:

print('--------------------------')

print()

print('The Prediction = Keratosis')

print()

print('-------------------------')

res1=" Affected by Keratosis"

st.text(res1)

elif labels1[zz[0][0]] == 9:

print('--------------------------')

print()

print('The Prediction = Measles')

print()

print('-------------------------')

res1=" Affected by Measles"

st.text(res1)

elif labels1[zz[0][0]] == 10:

print('--------------------------')

print()

print('The Prediction = Psoriasis')

print()

print('-------------------------')

res1=" Affected by Psoriasis"

st.text(res1)

elif labels1[zz[0][0]] == 11:

print('--------------------------')

print()

print('The Prediction = Ringworm')

print()

print('-------------------------')

res1=" Affected by Ringworm"

st.text(res1)

elif labels1[zz[0][0]] == 12:

print('--------------------------')

print()

print('The Prediction = Scabies')

print()

print('-------------------------')

res1=" Affected by Scabies"

st.text(res1)

elif navigation() == "results":

st.title('Results ')

rf=97

dt=95.7

st.text("-----------------------")

st.text("Performance Analysis")

st.text("-----------------------")

st.text(" ")

st.write("Random Forest = ", rf)

st.write("Decision Tree =", dt)

st.image("Result.png")

elif navigation() =="Contact":

st.title("Welcome to Contact Us Page!!!")

col1, col2, col3 = st.columns(3)

with col1:

st.image("1.jpg")

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"Person 1"}</h1>', unsafe\_allow\_html=True)

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"College Name"}</h1>', unsafe\_allow\_html=True)

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"Built framework "}</h1>', unsafe\_allow\_html=True)

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"College Name"}</h1>', unsafe\_allow\_html=True)

with col2:

st.image("1.jpg")

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"Person 2"}</h1>', unsafe\_allow\_html=True)

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"College Name"}</h1>', unsafe\_allow\_html=True)

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"Built framework "}</h1>', unsafe\_allow\_html=True)

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"College Name"}</h1>', unsafe\_allow\_html=True)

with col3:

st.image("1.jpg")

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"Person 3"}</h1>', unsafe\_allow\_html=True)

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"College Name"}</h1>', unsafe\_allow\_html=True)

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"Built framework "}</h1>', unsafe\_allow\_html=True)

st.markdown(f'<h1 style="color:#000000;font-size:14px;">{"College Name"}</h1>', unsafe\_allow\_html=True)

footer="""<style>

a:link , a:visited{

color: blue;

background-color: transparent;

text-decoration: underline;

}

a:hover, a:active {

color: red;

background-color: transparent;

text-decoration: underline;

}

.footer {

position: fixed;

left: 0;

bottom: 0;

width: 100%;

background-color: white;

color: black;

text-align: center;

}

</style>

<div class="footer">

<p>Developed with ❤ by Varun</a></p>

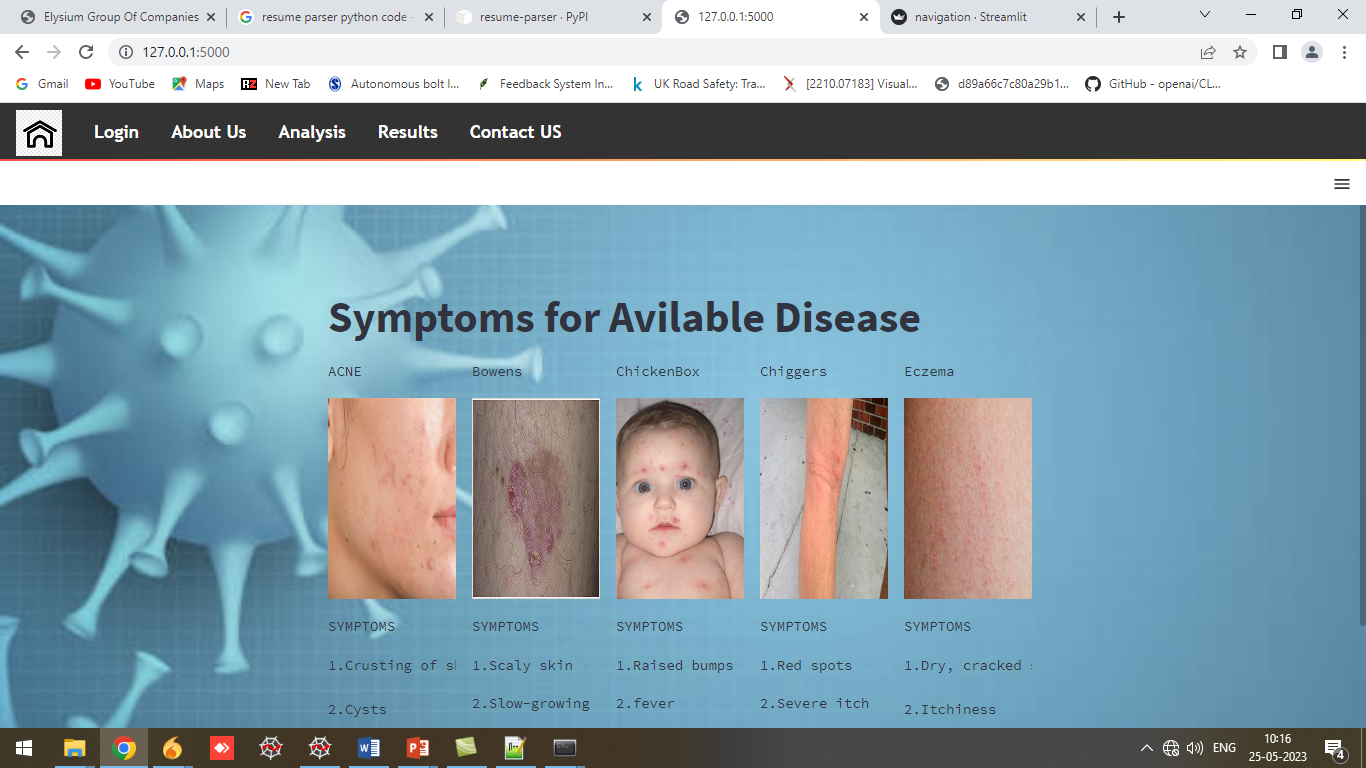
</div>

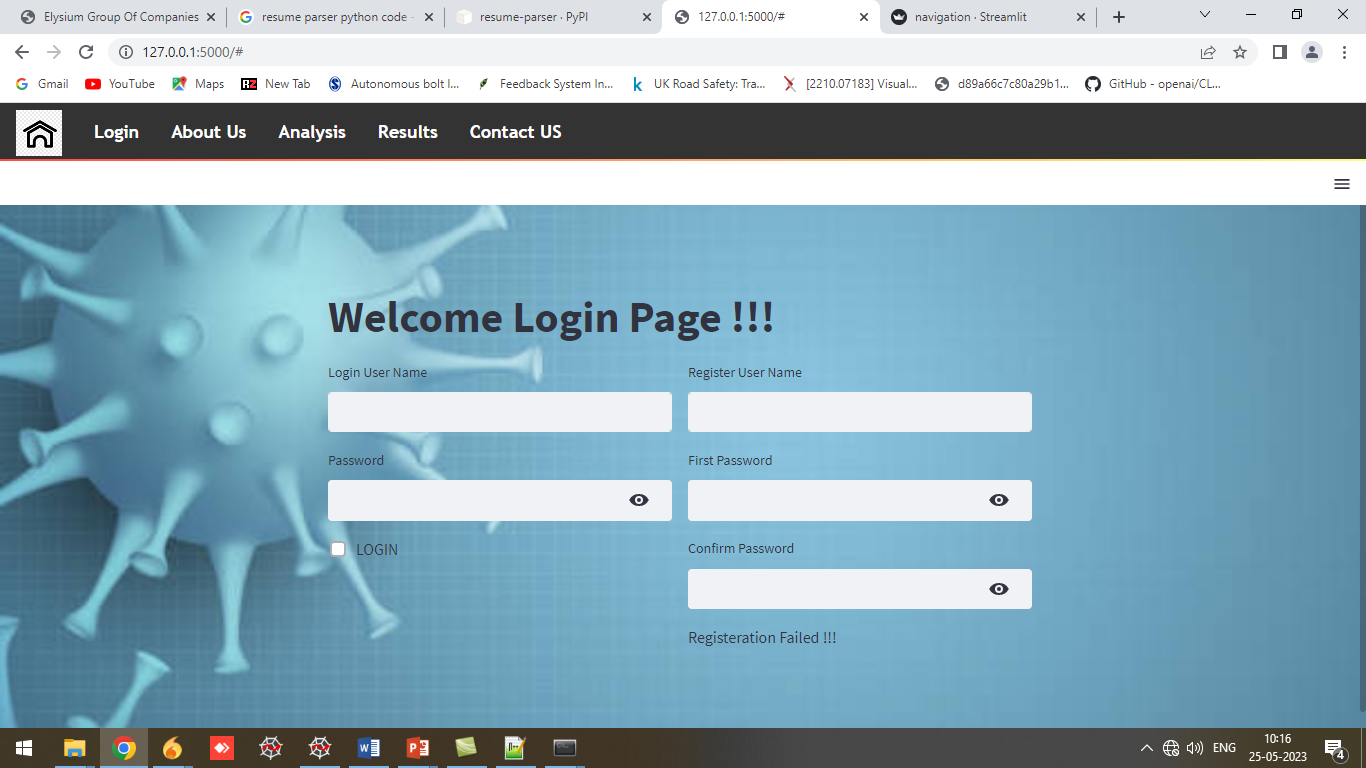
"""

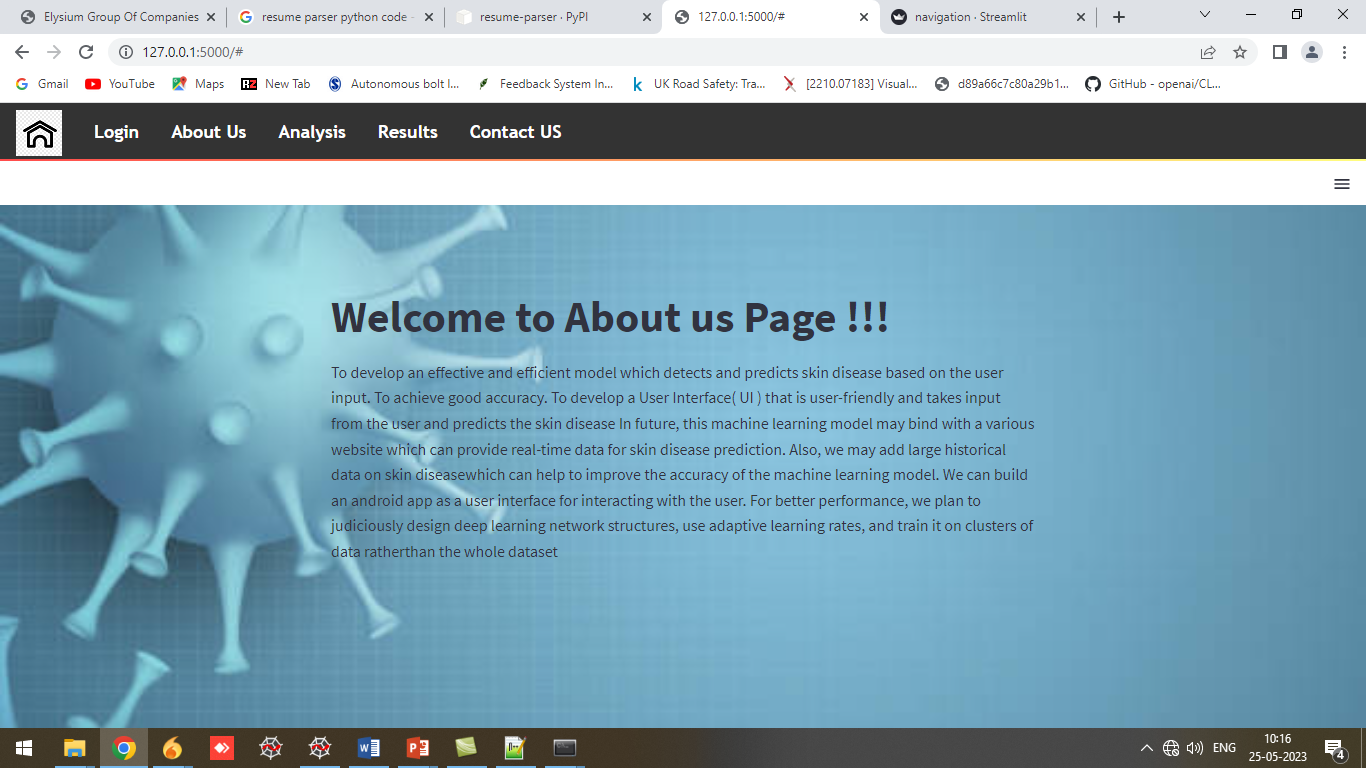
st.markdown(footer,unsafe\_allow\_html=True)

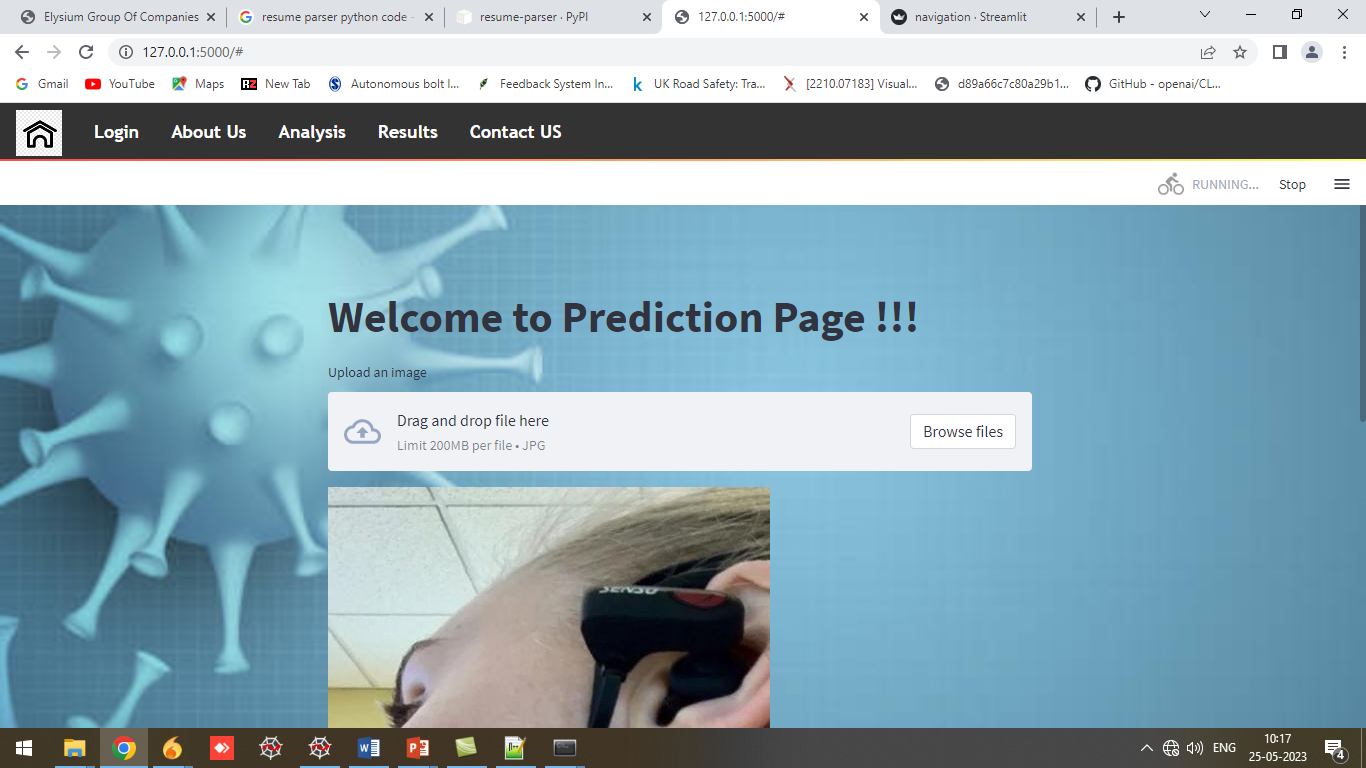
**CHAPTER 9**

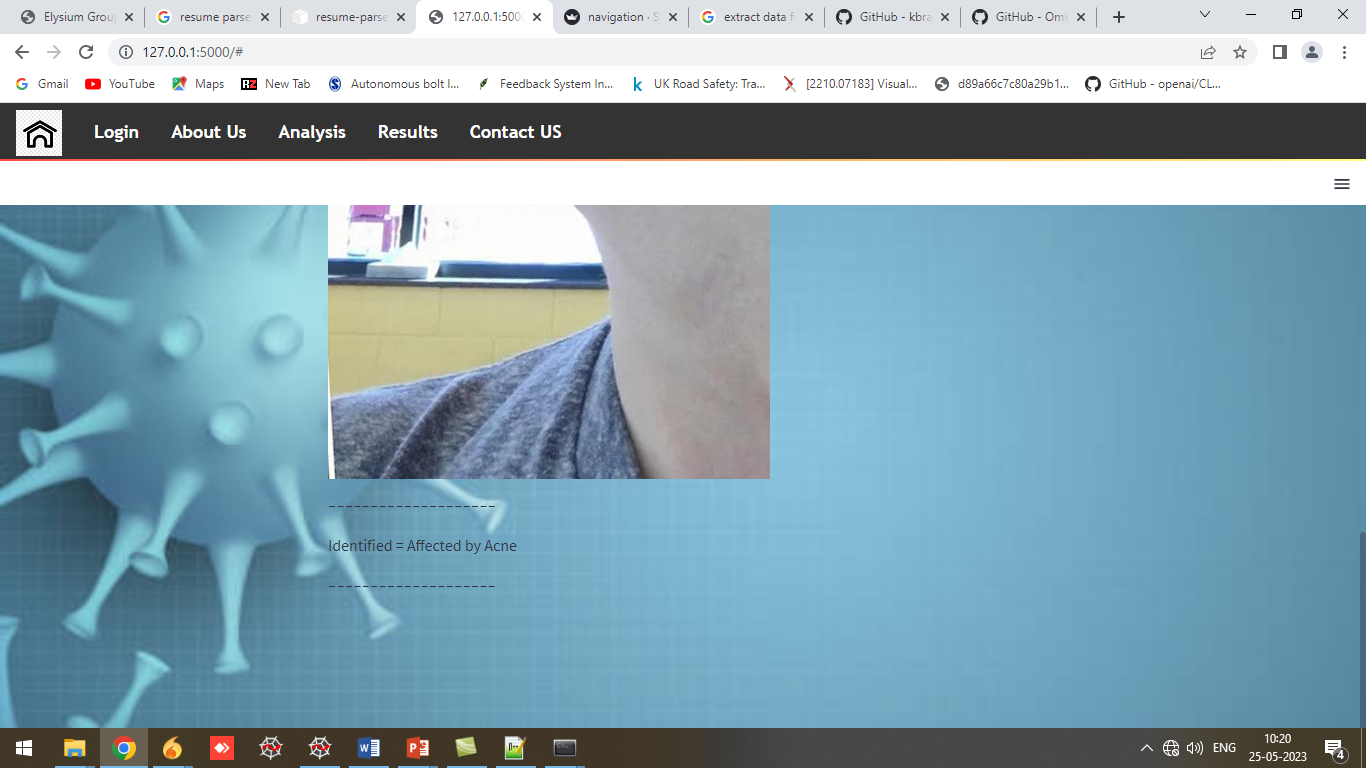
**SAMPLE SCREENSHOTS**

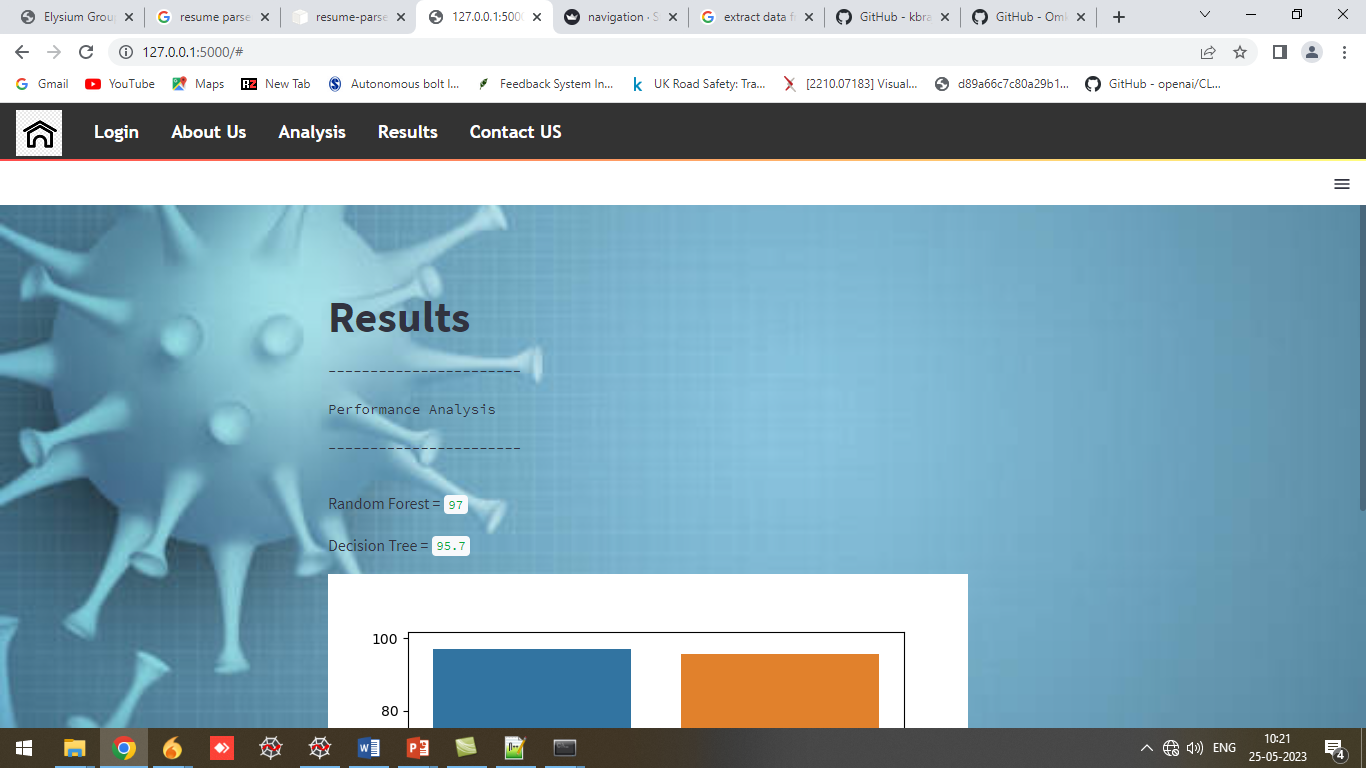


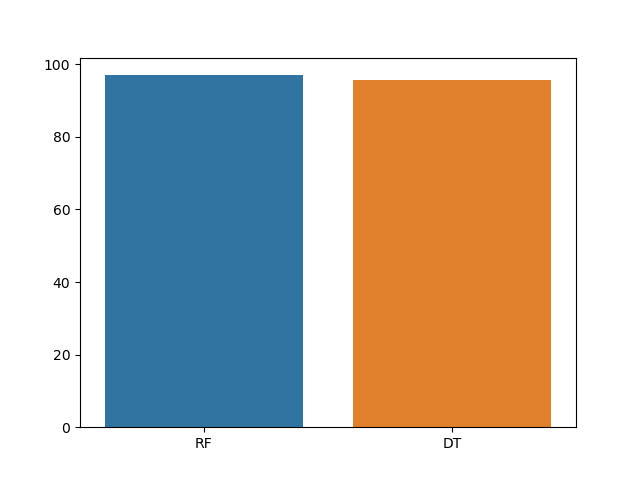












**CHAPTER 10**

**REFERENCES**

[1] M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. D. Jackel, U. Muller, and K. Zieba. Visualbackprop: visualizing cnns for autonomous driving. CoRR, abs/1611.05418, 2016. 3

[2] N. C. F. Codella, J. Cai, M. Abedini, R. Garnavi, A. Halpern, and J. R. Smith. Deep learning, sparse coding, and CNN for melanoma recognition in dermoscopy images. In Machine Learning in Medical Imaging, pages 118–126, MLMI 2015. 2

[3] N. C. F. Codella, D. Gutman, M. Emre Celebi, B. Helba, M. A. Marchetti, S. W. Dusza, A. Kalloo, K. Liopyris, N. K. Mishra, H. Kittler, and A. Halpern. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC). CoRR, abs/1710.05006, 2017. 2

[4] N. C. F. Codella, Q. D. Nguyen, S. Pankanti, D. Gutman, B. Helba, A. Halpern, and J. R. Smith. Deep learning ensembles for melanoma recognition in dermoscopy images. IBM Journal of Research and Development, 61(4/5), 2017.2 [4] F. Ccero, A. Oliveira, and G. Botelho. Deep learning and convolutional neural networks in the aid of the classification of melanoma. In Conference on Graphics, Patterns and Images, SIBGRAPI 2016. 2

[5] I. Gonzalez Diaz. Dermaknet: Incorporating the knowledge of dermatologists to convolutional neural networks for skin lesion diagnosis. IEEE Journal of Biomedical and Health Informatics, PP(99):1–1, 2018. 2,

[6]. Ali ARA, Deserno TM. A systematic review of automated melanoma detection in dermatoscopic images and its ground truth data. Proc SPIE Int Soc Opt Eng. 2012 Feb 28;8318:1–6. doi: 10.1117/12.912389. [CrossRef] [Google

Scholar]

[7 ]Fabbrocini G, De Vita V, Pastore F, D'Arco V, Mazzella C, Annunziata MC, Cacciapuoti S, Mauriello MC, Monfrecola A. Teledermatology: From prevention to diagnosis of nonmelanoma and melanoma skin cancer. Int J Telemed Appl. 2011 Sep 01;2011(17):125762. doi: 10.1155/2011/125762. doi: 10.1155/2011/125762. [PMC free article] [PubMed] [CrossRef] [CrossRef] [Google Scholar]

[8]. Foraker R, Kite B, Kelley MM, Lai AM, Roth C, Lopetegui MA, Shoben AB, Langan M, Rutledge NL, Payne PRO. EHR-based visualization tool: Adoption rates, satisfaction, and patient outcomes. EGEMS (Wash DC) 2015;3(2):1159. doi: 10.13063/2327-9214.1159. http://europepmc.org/abstract/MED/26290891. [PMC free article] [PubMed] [CrossRef] [Google Scholar]

[9]. Fabbrocini G, Betta G, Di Leo G, Liguori C, Paolillo A, Pietrosanto A, Sommella P, Rescigno O, Cacciapuoti S, Pastore F, De Vita V, Mordente I, Ayala F. Epiluminescence image processing for melanocytic skin lesion diagnosis based on 7-point check-list: A preliminary discussion on three parameters. Open Dermatol J. 2010 Jan 01;4(1):110–115. doi: 10.2174/1874372201004010110. [CrossRef] [Google Scholar]

[10]. Hart PE, Stork DG, Duda RO. Pattern Classification. 2nd edition. Hoboken, NJ: John Wiley & Sons; 2000. [Google Scholar]

[11] National Cancer Institute, PDQ Melanoma Treatment. Bethesda, MD, USA. (Nov. 4, 2019). PDQ Adult Treatment Editorial Board. Accessed: Dec. 9, 2019. [Online]. Available: <https://www.cancer.gov/> types/skin/hp/melanoma-treatment-pdq