

ENHANCED SKIN CONDITION PREDICTION THROUGH MACHINE LEARNING

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering
by

RAVELLA SHYAM PRASAD (Reg. No. 37110627)
C.V.S.RAHUL REDDY (Reg. No. 37110616)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL
OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

**Accredited with Grade “A” by NAAC JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)



Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide **RAVELLA SHYAM PRASAD (Reg.No.37110627)** and **C.V.S.RAHUL REDDY (Reg.No.37110616)** who carried out the project entitled "**ENHANCED SKIN CONDITION PREDICTION THROUGH MACHINE LEARNING**" under my supervision from **August 2020 to March 2021**.

Internal Guide

Ms. JITHINA JOSE M.E., (Ph.D).,

Head of the Department

Dr. S. VIGNESHWARI M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **RAVELLA SHYAM PRASAD** hereby declare that the Project Report entitled “**ENHANCED SKIN CONDITION PREDICTION THROUGH MACHINE LEARNING**” is done by me under the guidance of **Ms. JITHINA JOSE M.E., (Ph.D).**, Department of Computer Science and Engineering at **Sathyabama Institute of Science and Technology** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

RAVELLA SHYAM PRASAD

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean, School Of Computing and Dr. S. VIGNESHWARI M.E., Ph.D.,** Head of the Department, Dept. of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. JITHINA JOSE M.E., (Ph.D).,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of **Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In recent days, skin diseases are increasing in humans. Skin diseases are caused by bacteria or due to infections. Some of the skin diseases like ring worm, yeast infection, allergies etc, are increasingly spreading day by day. So these type of diseases should be identified in their earlier stages to avoid spreading. It can be identified using some factors like clinical parameters which are considered for identifying the disease. In our proposed system, we are using KNN and Naïve bayes techniques to detect various kinds of skin diseases. The possible skin diseases in different ages are dermatitis in age 0-5 years, warts affects in 6-11 years age, and acne vulgaris in 12-16 years age. Dermatomyositis is a type of skin disease that affects children at age of 5-15 and adult at 40-60 age. Based upon the above factors using AI we can predict the skin diseases. The previous models done in this field of application were able to report a maximum of six skin diseases with a maximum accuracy level of 75%. By implementing deep learning algorithm we are able to predict as many as 20 diseases with a higher accuracy level of 88%.

TABLE OF CONTENTS

	ABSTRACT	v
	LIST OF FIGURES	viii
CHAPTER NO.	TITLE	PAGE NO.
1.	INTRODUCTION	1
	1.1 INTRODUCTION SKIN DISEASE PREDICTION	1
	1.2 SYSTEM ARCHITECTURE	2
	1.3 SOFTWARE ENVIRONMENT	2
2.	LITERATURE SURVEY	6
3.	METHODOLOGY	8
	3.1 AIM OF PROJECT	9
	3.1.1 OBJECTIVES	9
	3.2 SCOPE	9
	3.3 SYSTEM DESIGN	9
	3.3.1 PROPOSED SYSTEM	10
	3.3.2 ADVANTAGES OF PROPOSED SYSTEM	10
	3.4 EXISTING SYSTEMS	10
	3.4.1 DISADVANTAGES OF EXISTING SYSTEM	10
	3.5 REQUIREMENT ANALYSIS	10
	3.6 SYSTEM REQUIREMENTS	11
4.	MODULES AND DIAGRAMS	12
	4.1 MODULES	12
	4.1.1 USER MODULE	12
	4.1.2 ADMIN MODULE	13
	4.1.3 DISEASE PREDICTION MODULE	13
	4.1.4 BEST DRUG RECOMMENDATION	13
	4.2 ML ALGORITHMS USED	13
	4.2.1 K-NEAREST NEIGHBOR (KNN)	13

	4.2.2 NAÏVE BAYES ALGORITHM	13
	4.3 DIAGRAMS	14
5	RESULTS AND DISCUSSION	20
	5.1 PROPOSED WORK & RESULT ANALYSIS	20
	5.2 OUTPUT SCREENSHOTS	21
6	CONCLUSION AND FUTURE WORK	23
	APPENDIX	
	A. PLAGIARISM REPORT	24
	B. SOURCE CODE	25
	C. JOURNAL PAPER	48

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.2	SYSTEM ARCHITECTURE	3
4.3.2	ER DIAGRAM	15
4.3.4	USECASE DIAGRAM	16
4.3	CLASS DIAGRAM	17
4.4	ACTIVITY DIAGRAM	18
4.5	SEQUENCE DIAGRAM	19
4.6	COLLABORATION DIAGRAM	19

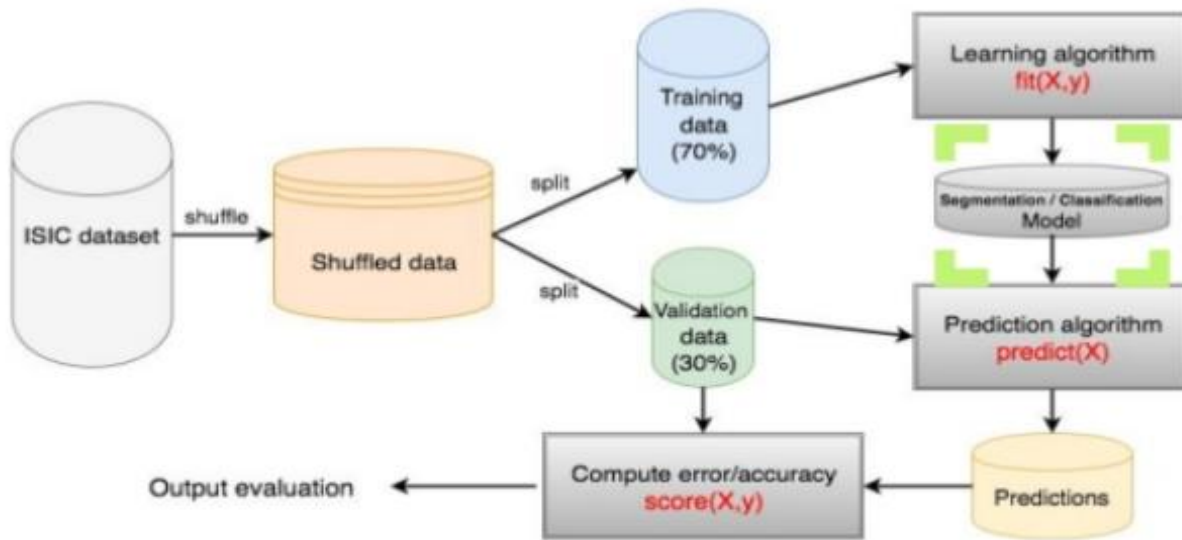
CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION FOR SKIN DISEASE PREDICTION:

There are different techniques to diagnose skin disease like Malignant melanoma i.e. melanoma skin tumors, Nonmelanoma i.e. squamous cell carcinoma, basal cell carcinoma, Acne, Genetic Diseases i.e. Sickle-Cell Anemia Genetic Skin Disorder, Fungal Diseases, Bacterial Diseases, Psoriasis and Leprosy. There are different techniques to diagnose these skin diseases. But the most useful and noninvasive method for diagnosis of different types of skin diseases is skin impedance measurement. The analysis is made with the measurement of skin impedance over a wide frequency range from 100 Hz to 1MHz. But this analysis for the infected skin and normal skin should be made at same frequency because at different frequencies skin impedance can be different. The measured skin impedance at same frequency of infected skin and normal skin is different, because of which we can diagnose skin disease. Above mentioned method used the non-invasive probe electrode system consists of two concentric electrodes attached to a ceramic plate. The outermost electrodes drive the voltage and inner is sink electrode gives output. In which source electrode provide small amount of current to skin and sink electrode gives input to the microcontroller. Length of the outermost electrode is nearly 10mm. Measurement of skin impedance can be used for depressions, anxiety, neurological diseases, and nerve lesions. It is also used for physiological measurements emotional disorders or lie detection. The various types of bio impedance method include multiple frequency, single frequency and Bio impedance spectroscopy .SF-BIA is generally performed at a frequency of 50 kHz. At this frequency the current passes Through both extracellular and intracellular fluid. Bio-impedance obtained at number of frequencies is analyzed in multiple-frequency bio-impedance analysis.

1.2 SYSTEM ARCHITECTURE



1.3 SOFTWARE ENVIRONMENT : JAVA

• INTRODUCTION TO JAVA

Java is one of the world's most important and widely used computer languages, and it has held this distinction for many years. Unlike some other computer languages whose influence has weared with passage of time, while Java's has grown.

• APPLICATION OF JAVA

Java is widely used in every corner of world and of human life. Java is not only used in softwares but is also widely used in designing hardware controlling software components. There are more than 930 million JRE downloads each year and 3 billion mobile phones run java.

Following are some other usage of Java:

1. Developing Desktop Applications
2. Web Applications like Linkedin.com, Snapdeal.com etc
3. Mobile Operating System like Android
4. Embedded Systems

5. Robotics and games etc.

2) Object Oriented

In java everything is Object which has some data and behaviour. Java can be easily extended as it is based on Object Model.

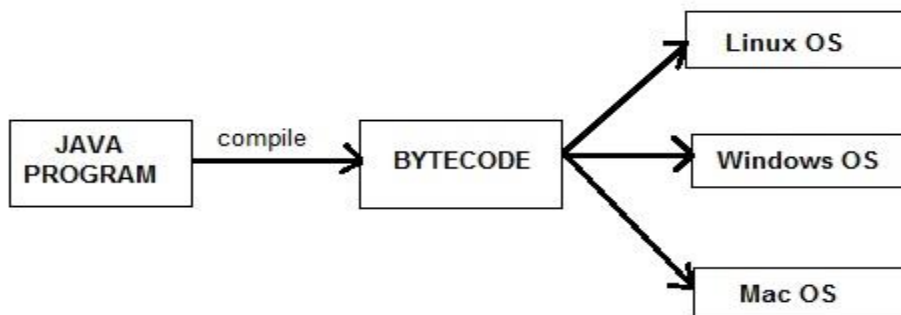
3) Robust

Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled Exceptions by introducing automatic Garbage Collector and Exception Handling

4) Platform Independent

Unlike other programming languages such as C, C++ etc. which are compiled into platform specific machines. Java is guaranteed to be write-once, run-anywhere language.

On compilation Java program is compiled into byte code. This byte code is platform independent and can be run on any machine, plus this byte code format also provide security. Any machine with Java Runtime Environment can run Java Programs.



5) Secure

When it comes to security, Java is always the first choice. With java secure features it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.

6) Multi-Threading

Java multithreading feature makes it possible to write program that can do many tasks simultaneously. Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along.

7) Architectural Neutral

Compiler generates byte codes, which have nothing to do with a particular computer architecture, hence a Java program is easy to interpret on any machine.

8) Portable

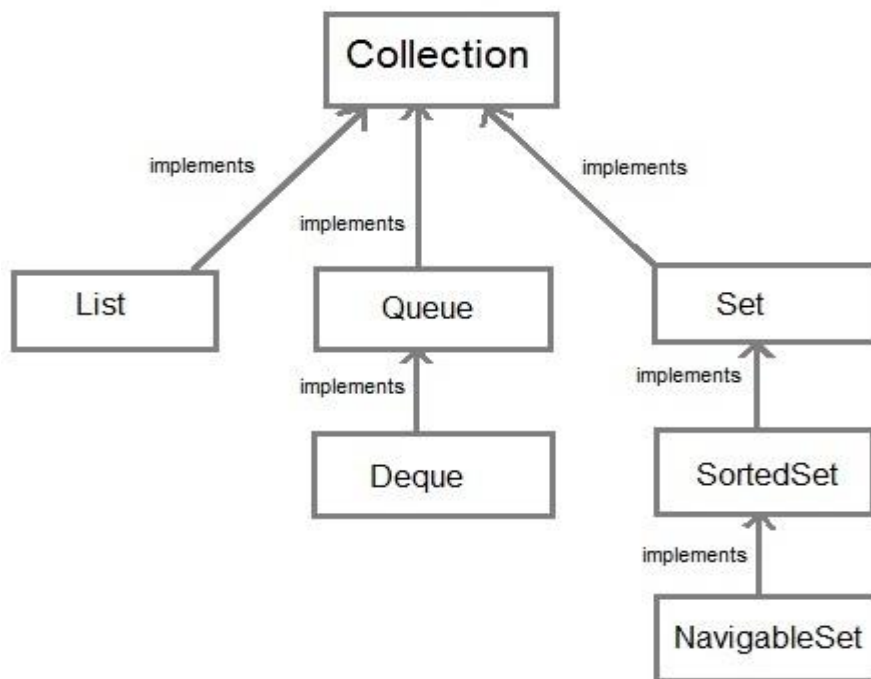
Java Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types.

9) High Performance

Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But, Java enables high performance with the use of just-in-time compiler.

- **COLLECTION FRAMEWORK**

Collection framework was not part of original Java release. Collections was added to J2SE 1.2. Prior to Java 2, Java provided adhoc classes such as Dictionary, Vector, Stack and Properties to store and manipulate groups of objects. Collection framework provides many important classes and interfaces to collect and organize group of alike objects.



1.4 MYSQL

MySQL, officially, but also called "My Sequel" is the world's most widely used open-source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases, though SQLite probably has more total embedded deployments. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks).

LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Freesoftware-open source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale websites, including Wikipedia, Google (though not for searches), Facebook, Twitter, Flickr and YouTube.

CHAPTER 2

LITERATURE SURVEY

1.Segmentation and Classification of Skin Lesions for Disease Diagnosis

In this paper, a novel approach for automatic segmentation and classification of skin lesions is proposed. Initially, skin images are filtered to remove unwanted hairs and noise and then the segmentation process is carried out to extract lesion areas. For segmentation, a region growing method is applied by automatic initialization of seed points. The segmentation performance is measured with different well known measures and the results are appreciable. Subsequently, the extracted lesion areas are represented by color and texture features. SVM and k-NN classifiers are used along with their fusion for the classification using the extracted features. The performance of the system is tested on our own dataset of 726 samples from 141 images consisting of 5 different classes of diseases. The results are very promising with 46.71% and 34% of F-measure using SVM and k-NN classifier respectively and with 61% of F-measure for fusion of SVM and k-NN.

2.Extraction of Skin Lesions from Non-Dermoscopic Images Using Deep Learning

Melanoma is amongst most aggressive types of cancer. However, it is highly curable if detected in its early stages. Prescreening of suspicious moles and lesions for malignancy is of great importance. Detection can be done by images captured by standard cameras, which are more preferable due to low cost and availability. One important step in computerized evaluation of skin lesions is accurate detection of lesion region, i.e. segmentation of an image into two regions as lesion and normal skin. Accurate segmentation can be challenging due to burdens such as illumination variation and low contrast between lesion and healthy skin. In this paper, a method based on deep neural networks is proposed for accurate extraction of a lesion region. The input image is preprocessed and then its patches are fed to a convolutional neural network. Local texture and global structure of the patches are processed in order to assign pixels to lesion or normal classes. A method for effective selection of training patches is used for more accurate detection of a lesion border. The output segmentation mask is refined by some post processing operations. The experimental results of qualitative and quantitative evaluations demonstrate that our method can outperform other state-of-the-art algorithms exist in the literature.

3.AN INTELLIGENT SYSTEM TO DIAGNOSIS THE SKIN DISEASE

Skin cancer is a major health issue affecting a vast segment of the population regardless the skin color. This affliction can be detected using dermoscopy to determine whether the visible spots on skin are either benign or malignant tumors. In spite of the specialists' experience, skin lesions are difficult to classify, reason for which computer systems are developed to increase the

effectiveness of cancer detection. Systems assisting in the detection of skin cancer process digital images to determine the occurrence of tumors by interpreting clinical parameters, relying, firstly, upon an accurate segmentation process to extract relevant features. Two of the well-known methods to analyze lesions are ABCD and the 7-point check list. After clinically-relevant features are extracted, they are used to classify the presence or absence of a tumor. However, irregular and disperse lesion borders, low contrast, artifacts in images and the presence of various colors within the region of interest complicate the processing of images. In this article, we propose an intelligent system running the following method. The feature extraction stage begins with the segmentation of an image, for which we apply the Wavelet – Fuzzy C-Means algorithm. Next, specific features should be determined, among others the area and the asymmetry of the lesion. An ensemble of clusterers extracts the Red-Green-Blue values that correspond to one or more of the colors defined in the ABCD guide. The feature extraction stage includes the discovery of structures that appear in the lesion according to the method known as Grey Level CoOccurrence Matrix. Then, during the detection phase, an ensemble of classifiers determines the occurrence of a malignant tumor. Our experiments are performed on images taken from the ISIC repository. The proposed system provides a skin cancer detection performance above 88 percent, as measured by the accuracy. Details of how this performance fares when compared with other systems are also given.

4. Automatic segmentation of skin lesions from dermatological photograph

Melanoma is the deadliest form of skin cancer if left untreated. Incidence rates of melanoma have been increasing, especially among young adults, but survival rates are high if detected early. Unfortunately, the time and costs required for dermatologists to screen all patients for melanoma are prohibitively expensive. There is a need for an automated system to assess a patient's risk of melanoma using photographs of their skin lesions. Dermatologists could use the system to aid their diagnosis without the need for special or expensive equipment. One challenge in implementing such a system is locating the skin lesion in the digital image. Most existing skin lesion segmentation algorithms are designed for images taken using a special instrument called the dermatoscope. The presence of illumination variation in digital images such as shadows complicates the task of finding the lesion. The goal of this research is to develop a framework to automatically correct and segment the skin lesion from an input photograph. The first part of the research is to model illumination variation using a proposed multi-stage illumination modeling algorithm and then using that model to correct the original photograph. Second, a set of representative texture distributions are learned from the corrected photograph and a texture distinctiveness metric is calculated for each distribution. Finally, a texture-based segmentation algorithm classifies regions in the photograph as normal skin or lesion based on the occurrence of representative texture distributions. The resulting segmentation can be used as an input to separate feature extraction and melanoma classification algorithms. The proposed segmentation framework is tested by comparing lesion segmentation results and melanoma classification results to results using other state-of-the-art algorithms. The proposed framework has better segmentation accuracy compared to all other

tested algorithms. The segmentation results produced by the tested algorithms are used to train an existing classification algorithm to identify lesions as melanoma or non-melanoma. Using the proposed framework produces the highest classification accuracy and is tied for the highest sensitivity and specificity.

6.A SURVEY ON SKIN TEXTURE ANALYSIS FOR MEDICAL DIAGNOSIS USING IMAGE PROCESSING TECHNIQUES

Skin texture analysis is one of the challenging issues in the field of medical diagnosis. Various types of skin diseases are affecting human life like skin dryness, fungus, and allergic symptoms. The objective of this paper is to analyze the skin disease using texture analysis of skin image and by comparing the test image to a defined images or reference images. The matching of test and reference images compared that yields the percentage of skin diseases in the captured skin texture image.

CHAPTER 3

METHODOLOGY

3.1 AIM OF PROJECT

The main aim of this system early disease recognition, patient care and community services. When the quality of medical data is incomplete the exactness of study is reduced.

3.1.1 OBJECTIVES

The analysis accuracy is reduced when the quality of medical data is incomplete. Moreover, different regions exhibit unique characteristics of certain regional diseases, which may weaken the prediction of disease outbreaks. However, those existing work mostly considered structured data. There is no proper methods to handle semi structured and unstructured. The proposed system will consider both structured and unstructured data. The analysis accuracy is increased by using Machine Learning algorithm.

3.2 SCOPE

In this work, our goal is to provide a tool to assist professionals and consumers in finding and choosing disease. To achieve this goal, we develop an approach that allows a user to query for disease that satisfy a set of conditions based on disease properties, such as disease indications.

3.3 SYSTEM DESIGN

3.3.1 PROPOSED SYSTEM

In proposed system, KNN and Naïve Bayes techniques to detect various kinds of skin diseases. It can be identified using some factors like clinical parameters which are considered for identifying the disease. The possible skin diseases in different ages are dermatitis in age 0-5 years, warts affects in 6-11 years age, and acne vulgaris in 12-16 years age. Dermatomyositis is a type of skin disease that affects children at age of 5-15

and adult at 40-60 age. We are giving a suggestion of what kind of treatment you want and also showing life thread or not.

3.3.2 ADVANTAGES OF PROPOSED SYSTEM

- Easily analyze the disease
- High Security

3.4 EXISTING SYSTEM

In this article, existing method that uses computer vision based techniques to detect various kinds of dermatological skin diseases. We have used different types of machine learning algorithms for feature extraction and feed forward artificial neural network for training and testing purpose. The system works on two phases- first pre-process the colour skin images to extract significant features and later identifies the diseases. The system successfully detects 9 different types of dermatological skin diseases with an accuracy rate of 90%.

3.4.1 DISADVANTGES OF EXISTING SYSTEM

- Does not analyse the disease
- Less security
- There is no feedback system

3.5 REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

3.6 SYSTEM REQUIREMENTS

3.6.1 HARDWARE REQUIREMENTS

System : Pentium Dual Core.
Hard Disk : 120 GB.
Monitor : 15"LED
Input Devices : Keyboard, Mouse
Ram : 1GB.

3.6.2 SOFTWARE REQUIREMENTS

Operating system : Windows 7.
Coding Language : Java
Toolkit : Netbeans
DATABASE : MySql

CHAPTER 4

MODULES AND DIAGRAMS

4.1 MODULES

- User Module
- Admin Module
- Disease prediction Module
- Best drug recommendation

4.1.1 USER MODULE

In this Module, Patient can act as a user.

Login

This is the first activity that opens the website. User needs to provide a correct contact number and a password, which user enters while registering, in order to login into the app. If information provided by the user matches with the data in the database table then user successfully login into the app else message of login failed is displayed and user need to reenter correct information. A link to the register activity is also provided for registration of new users.

1. Registration

A new user who wants to access the website needs to register first before login. By clicking on register button in login activity, the register activity gets open. A new user registers by entering full name, password and contact number. A user needs to enter password again in confirm password textbox for confirmation. When user enters the information in all textboxes, on the click of register button, the data is transferred to database and user is directed to login activity again. Registered user then needs to login in order to access the app. Validations are applied on all the textboxes for proper functioning of the app. Like information in each textbox is must that is each textbox, either it is of name, contact, password or confirm password, will not be empty while registering. If any such textbox is empty app will give message of information is must in each textbox. Also data in password and confirm password fields must match for successful registration. Another validation is contact number must be valid one that is of 10 digits. If any such validation is violated then registration will be unsuccessful and then user needs to register again. Message that app will display when one of the field is empty. If all such information is correct user will be directed to login activity for login into the app.

4.1.2 ADMIN MODULE

In this module, admin can add and view new disease details and drug details. And then admin can view feedback provided by various users.

4.1.3 DISEASE PREDICTION MODULE

Patient will specify the symptoms caused due to his illness. System will ask certain question regarding his illness and system predict the skin disease based on the symptoms and age specified by the patient and system will also suggest drugs based on the disease.

4.1.4 BEST DRUG RECOMMENDATION

In this module we develop an app that allows a user to query for drugs that satisfy a set of conditions based on drug properties, such as drug indications, side effects, and drug interactions, and also takes into account patient profiles. In this we automate the suggestion of the alternative drugs. So that we can provide the medicine to diseases i.e. a preferred medicine .so the researcher will analysis the dosage of the drug and the symptoms.

4.2. ML ALGORITHMS USED

4.2.1 K-NEAREST NEIGHBOR(KNN) CLASSIFICATION METHOD

K-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms. The neighbors are taken from a set of objects for which the class (for k -NN classification) or the object property value (for k -NN regression).

4.2.2 NAIVE BAYES ALGORITHM

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the

existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. Look at the equation below:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

Above,

- $P(c|x)$ is the posterior probability of *class* (c , *target*) given *predictor* (x , *attributes*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

4.3 DIAGRAMS

4.3.1 DATA FLOW DIAGRAM

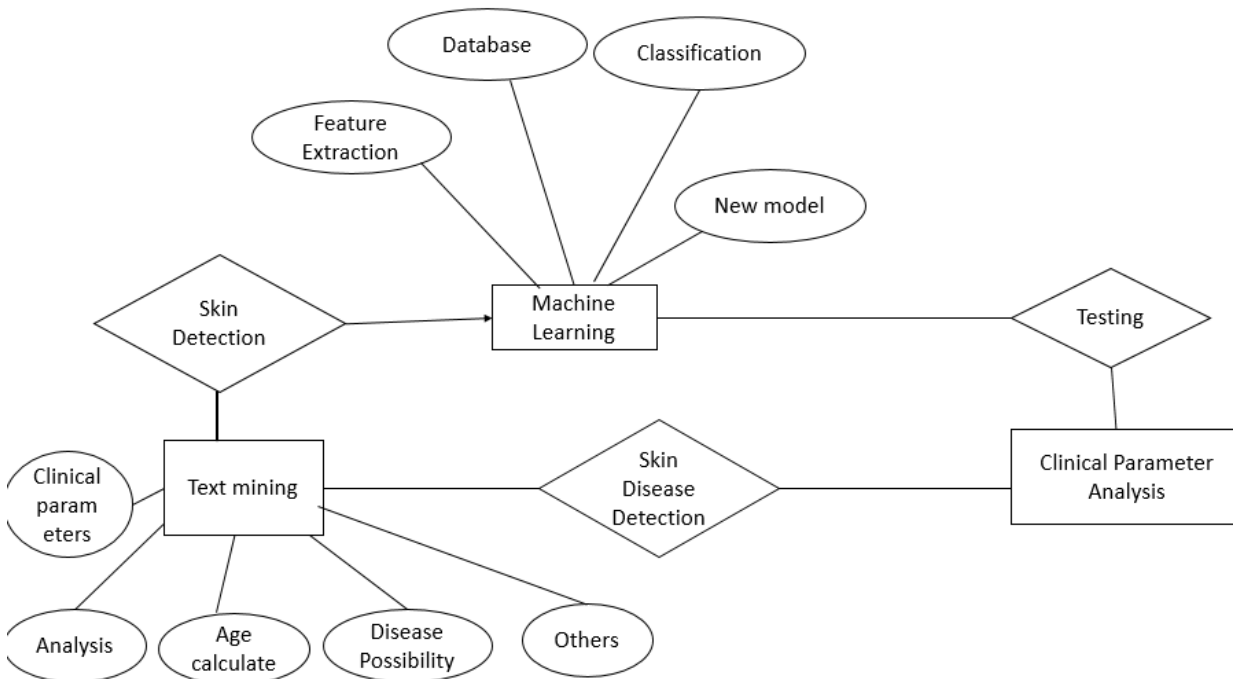
Data Flow Diagram (DFD) is a two-dimensional diagram that describes how data is processed and transmitted in a system. The graphical depiction recognizes each source of data and how it interacts with other data sources to reach a mutual output. In order to draft a data flow diagram one must

- Identify external inputs and outputs
- Determine how the inputs and outputs relate to each other
- Explain with graphics how these connections relate and what they result in.

Role of DFD:

- It is a documentation support which is understood by both programmers and nonprogrammers. As DFD postulates only what processes are accomplished not how they are performed.
- A physical DFD postulates where the data flows and who processes the data.
- It permits analyst to isolate areas of interest in the organization and study them by examining the data that enter the process and viewing how they are altered when they leave.

4.3.2 ER DIAGRAM



4.3.3 UML DIAGRAMS

UML is simply another graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

ADVANTAGES

- To represent complete systems (instead of only the software portion) using object oriented concepts

- To establish an explicit coupling between concepts and executable code
- To take into account the scaling factors that are inherent to complex and critical systems
- To creating a modeling language usable by both humans and machines

UML defines several models for representing systems

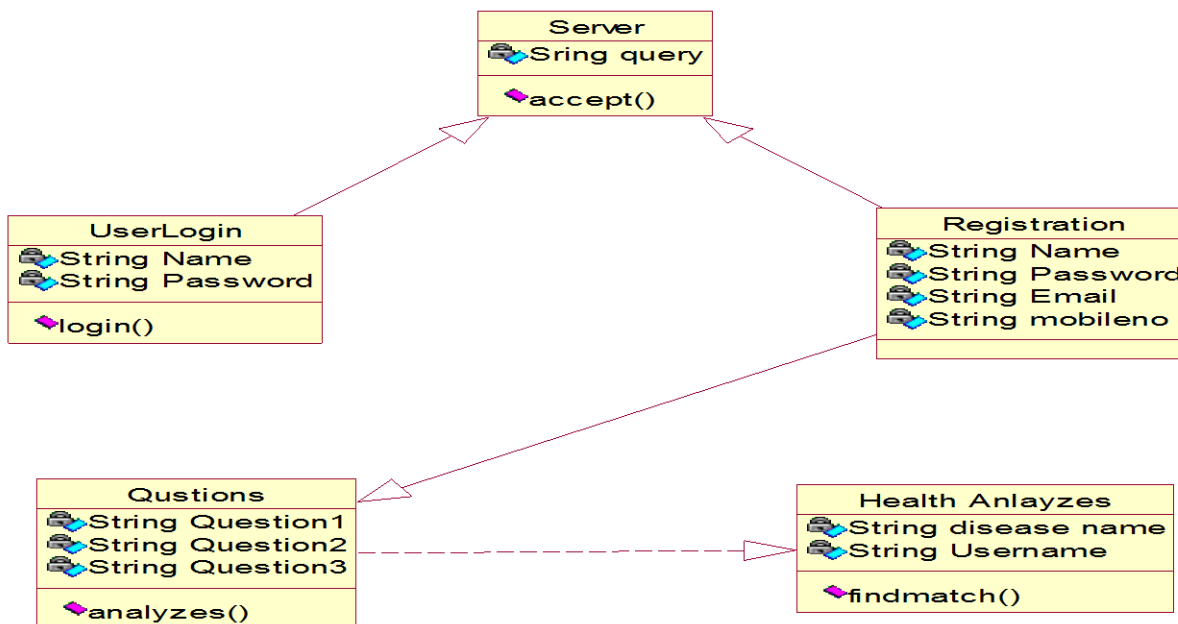
- The class model captures the static structure
- The state model expresses the dynamic behavior of objects
- The use case model describes the requirements of the user
- The interaction model represents the scenarios and messages flows
- The implementation model shows the work units
- The deployment model provides details that pertain to process allocation

4.3.4 USECASE DIAGRAM

Use case diagrams overview the usage requirement for system. They are useful for presentations to management and/or project stakeholders, but for actual developmet you will find that use cases provide significantly more value because they describe



4.3.5 CLASS DIAGRAM

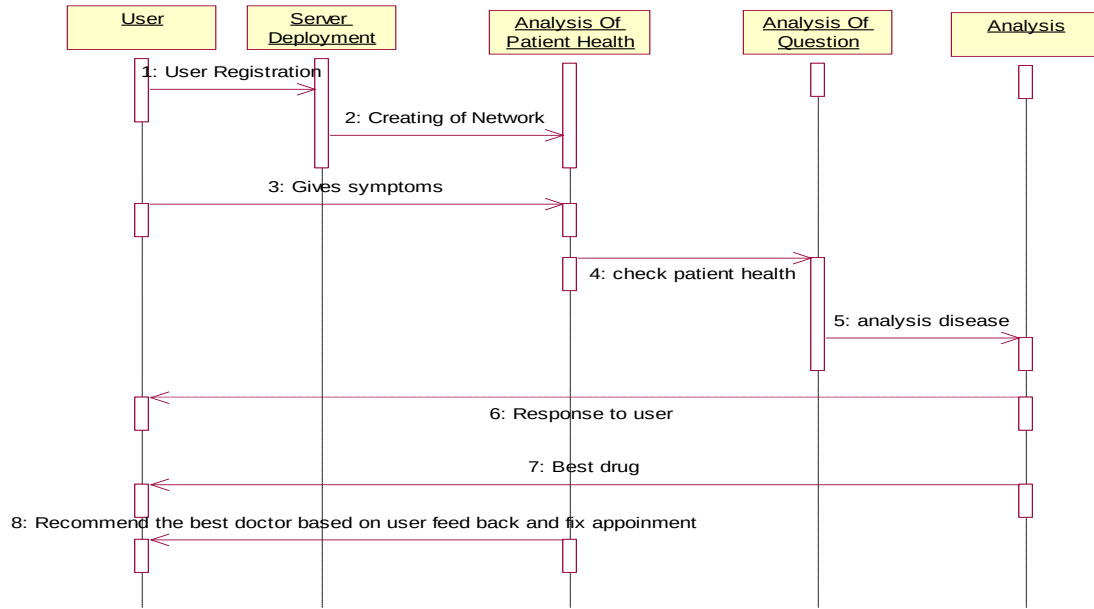


4.3.6 ACTIVITY DIAGRAM

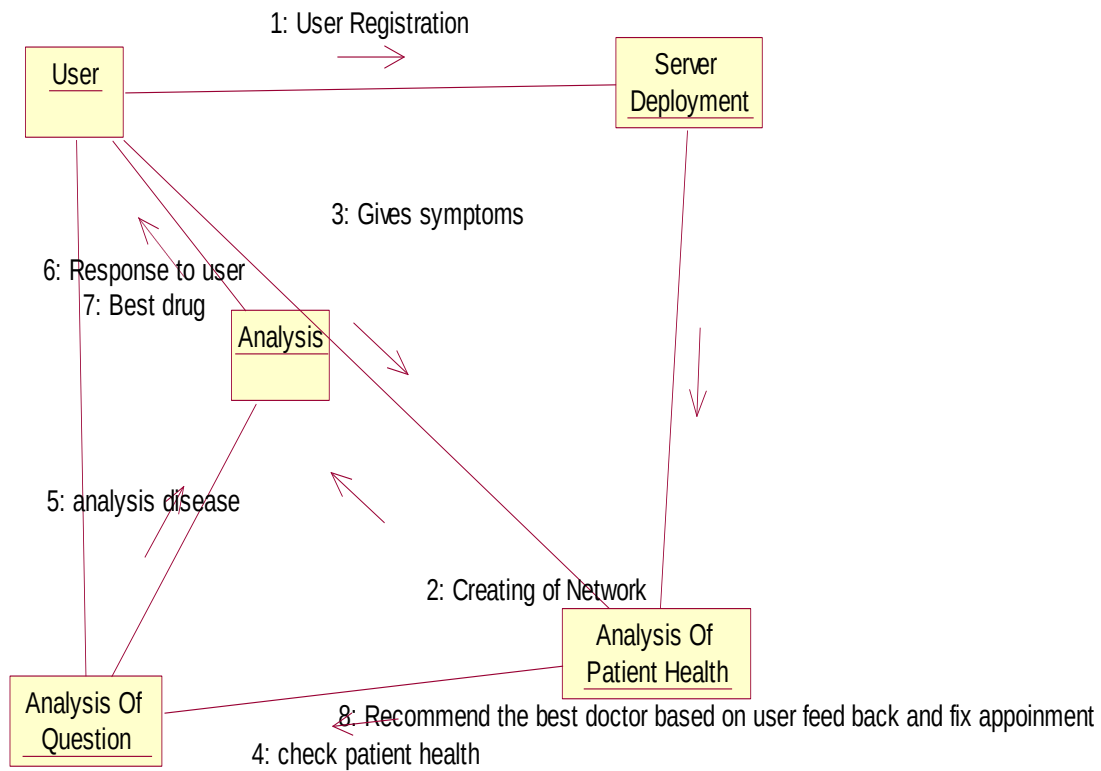
Activity diagram are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. Activity diagram consist of Initial node, activity final node and activities in between.

4.3.7 SEQUENCE DIAGRAM

Sequence diagram model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and commonly used for both analysis and design purpose. Sequence diagram are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system.



4.3.8 COLLABORATION DIAGRAM



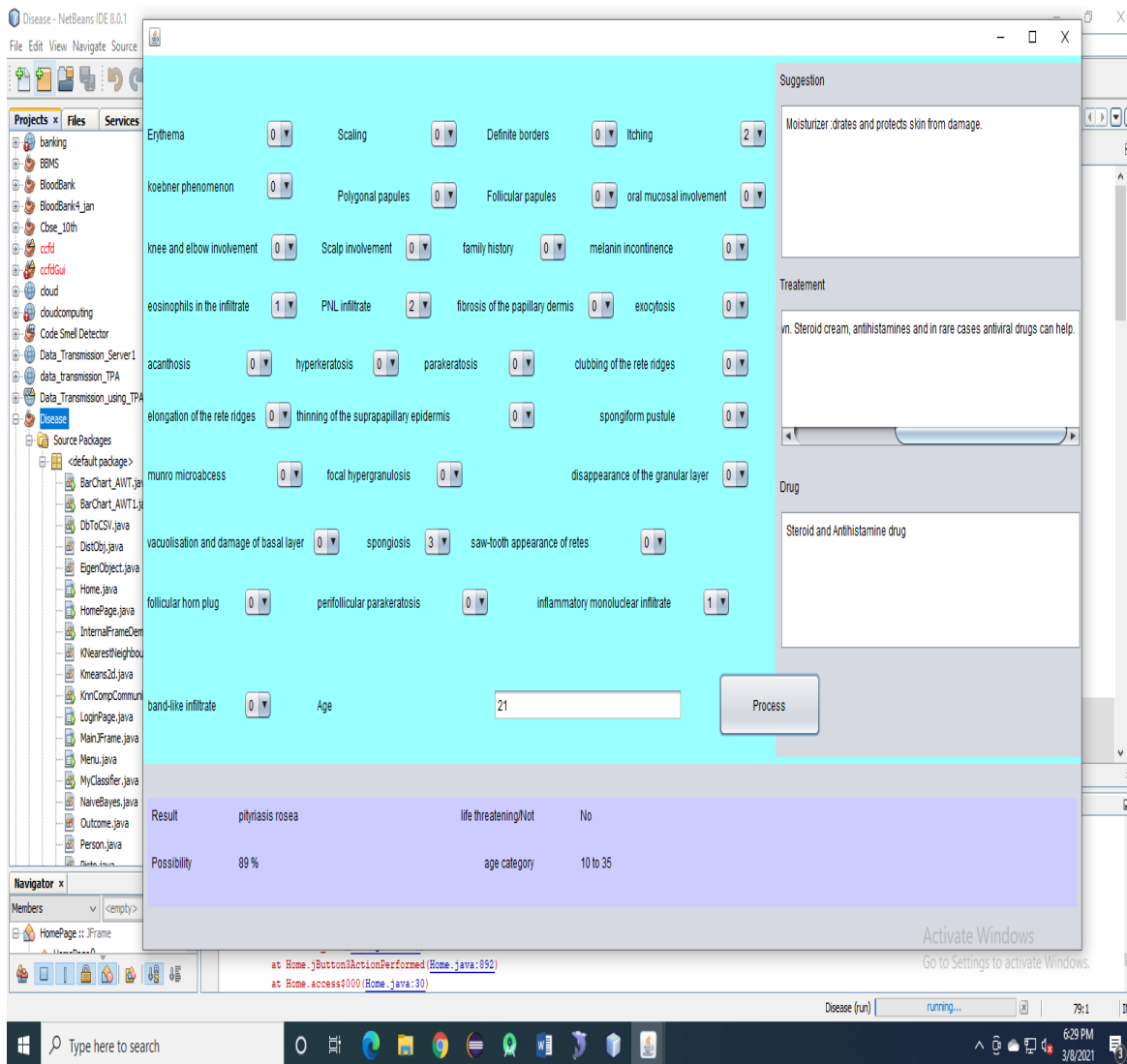
CHAPTER 5

RESULTS AND DISCUSSION

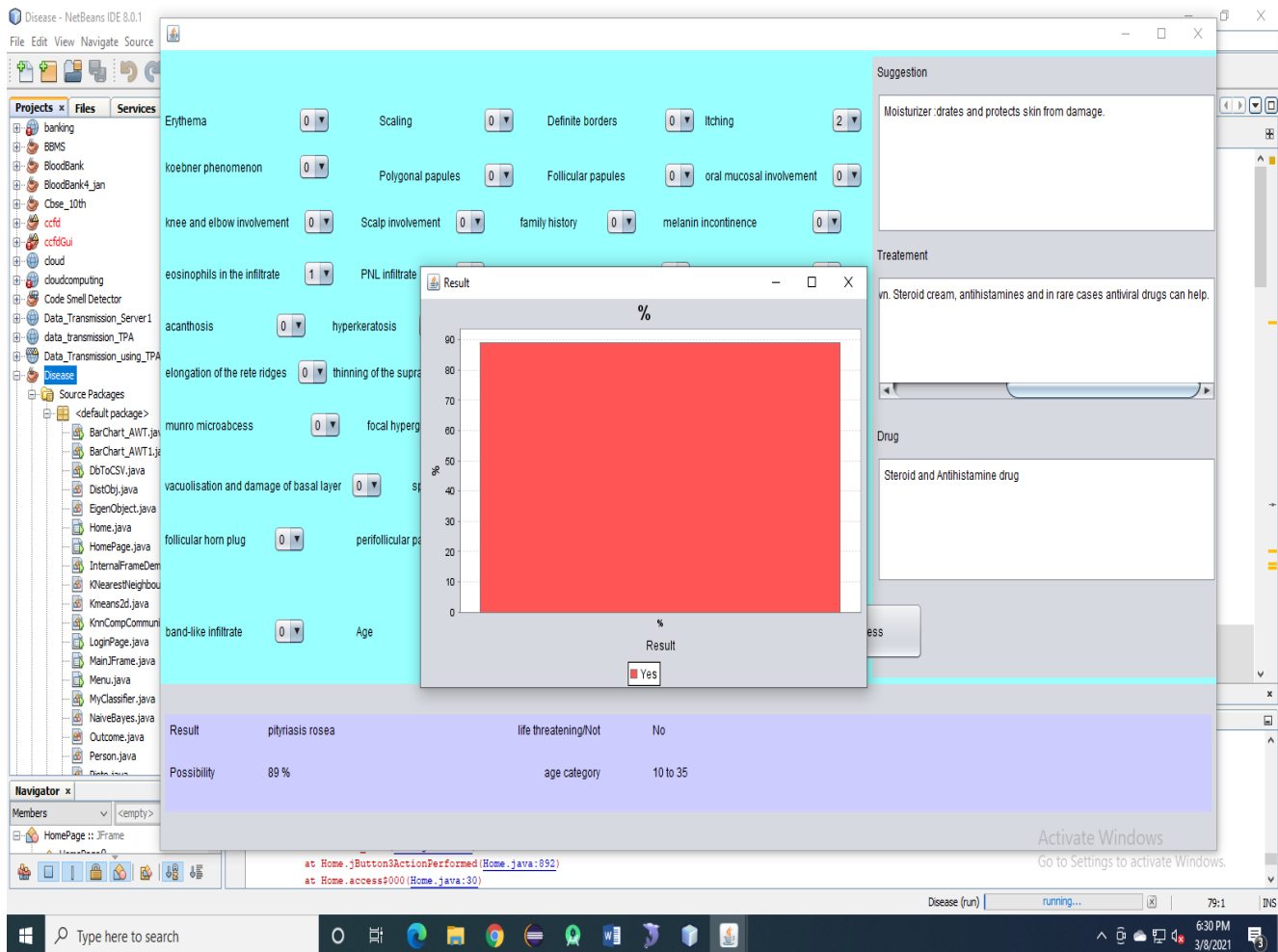
5.1 PROPOSED WORK AND RESULT ANALYSIS

The use of five distinct machine learning classifiers could identify three kinds of skin diseases called acne, lichen planus and sjs ten. For further classifications, the skin image dataset is originally preprocessed. The dataset is split into the dataset of instruction and testing: 80 percent for instruction and 20 percent for testing. The testing data is trained into three different classes denoted as class a, b and c for three diseases. Afterwards, skin diseases are detected using five different classification algorithms called machine learning, logistic regression, kernel SVM, naive bayes, random forest, and CNN. Each algorithm runs on the same dataset ten times and the training accuracy is calculated for each run. Skin diseases are detected afterwards using five different classification algorithms called machine learning, logistic regression, SVM kernel, naive bayes, random forest, and CNN. Each algorithm runs ten times on the same dataset and each run calculates the training precision. The parameters thus obtained are discussed and compared for all the five classification algorithms and graphical analysis is performed to check for the best algorithm which can be used for skin disease prediction. A brief overview and calculation formulas for different parameters are discussed below: A confusion matrix is a prediction summary based upon the results obtained on a classification problem. It is a table that summarizes the number of correct and incorrect predictions. The matrix describes the performance of a classical model on a given dataset. It shows the errors being made by a classifier along with the type of errors that are being made. The true negatives (TN), true positives (TP), false negatives (FN) and false positives (FP) scores are stored in a matrix for each class. We have three distinct classes of skin disease called acne, lichen planus and sjs ten in the suggested skin surveillance scenario. In our proposed case scenario, the values in the cells are distributed as TN for each class of skin disease, results in which if the disease is absent, the model does not predict the diseases.

5.2 OUTPUT SCREENSHOTS



DISEASE PREDICTION



RESULT

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In this work a model for prediction of skin diseases is done using deep learning algorithms. It is found that by using the ensembling features and deep learning we can achieve a higher accuracy rate and also we can go for the prediction of many more diseases than with any other previous models done before. As the previous models done in this field of application were able to report a maximum of six skin diseases with a maximum accuracy level of 75%. By implementing deep learning algorithm we are able to predict as many as 20 diseases with a higher accuracy level of 88%. This proves that deep learning algorithms have a huge potential in the real world skin disease diagnosis. If even a better system with high end system hardware and software with a very large dataset is used the accuracy can be increased considerably and the model can be used for clinical experimentation as it does have any invasive measures. Future work can be extended to make this model a standard procedure for preliminary skin disease diagnosis method as it will reduce the treatment and diagnosis time.

6.2 FUTURE WORK

Detection of skin disease is one of the major problems in the medical industry and can be healed and retrieved if properly diagnosed at an early point. Literature study demonstrates that different skin disease observation techniques are being used. However, there is still a great need to classify skin diseases at an early point. Machine learning algorithms have the potential to have an impact on early detection of skin diseases. It can assist people make real-time adjustments to their skin. If embraced well, the techniques will certainly provide appropriate assistance and a unified approach to skin problems prevention. This will assist patients and physicians cure skin diseases in a timely manner. Research and execution of limited medical information are accessible. If more real-time data are available in the future, the detection of skin disease can be explored with recent advances in AI and the benefits of diagnosis assisted with AI.

APPENDIX

A.PLAGARISM REPORT

ENHANCED SKIN CONDITION PREDICTION THROUGH MACHINE LEARNING

ORIGINALITY REPORT

5%	2%	3%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Alaa Haddad, Shihab A. Hameed. "Image Analysis Model For Skin Disease Detection: Framework", 2018 7th International Conference on Computer and Communication Engineering (ICCCE), 2018 Publication	2%
2	Submitted to Chino Valley Unified School District Student Paper	1%
3	www.jpinfotech.org Internet Source	1%
4	dblp.uni-trier.de Internet Source	<1%
5	www.springerprofessional.de Internet Source	<1%
6	Rahat Yasir, Md. Ashiqur Rahman, Nova Ahmed. "Dermatological disease detection using image processing and artificial neural	<1%

B.SOURCE CODE

Naïve Bayes

```
import java.util.ArrayList;

public class NaiveBayes {
    //holds the mean values for each attribute in the yes class
    //holds the standard deviation values for each attribute in the yes class
    private double attributes[];
    private double yesMean[];
    private double yesStdDev[];
    private double noMean[];
    private double noStdDev[];

    private double pYes;
    private double pNo;

    public NaiveBayes(ArrayList<Person> training) {
        int size = training.get(0).getAttributes().length;
        yesMean = new double[size];
        noMean = new double[size];
        yesStdDev = new double[size];
        noStdDev = new double[size];
        double y = 0; //Number of yes examples in training set
        double n = 0; //Number of no examples in training set
        for(Person p : training) {
            attributes = p.getAttributes();
            if(p.getOutcome() == Outcome.yes) {
                y++;
                for(int i = 0; i < attributes.length; i++) {
                    yesMean[i] += attributes[i];
                }
            }
            else {
                n++;
                for(int i = 0; i < attributes.length; i++) {
                    noMean[i] += attributes[i];
                }
            }
        }
    }
}
```

```

for(int i = 0; i < attributes.length; i++) {
    yesMean[i] = yesMean[i] / y;
    noMean[i] = noMean[i] / n;
}

for(Person p : training) {
    attributes = p.getAttributes();
    if(p.getOutcome() == Outcome.yes) {
        for(int i = 0; i < attributes.length; i++) {
            yesStdDev[i] += Math.pow(attributes[i] - yesMean[i], 2);
        }
    }
    else {
        for(int i = 0; i < attributes.length; i++) {
            noStdDev[i] += Math.pow(attributes[i] - noMean[i], 2);
        }
    }
}

for(int i = 0; i < attributes.length; i++) {
    yesStdDev[i] = Math.sqrt((yesStdDev[i])/(y-1));
    noStdDev[i] = Math.sqrt((noStdDev[i])/(n-1));
}

pYes = y/(y+n); //probability of any given example being yes
pNo = n/(n+y); //probability of any given example being no
}

public Outcome Test(Person test) {
    double pYesGivenE = 1.0;
    double pNoGivenE = 1.0;
    double attributes[] = test.getAttributes();
    for(int i = 0; i < attributes.length; i++) {
        pYesGivenE *= getProbability(attributes[i], yesMean[i], yesStdDev[i]);
        pNoGivenE *= getProbability(attributes[i], noMean[i], noStdDev[i]);
    }
    pYesGivenE *= pYes;
    pNoGivenE *= pNo;

    if(pNoGivenE > pYesGivenE) {
        return Outcome.no;
    }
    else {

```

```

        return Outcome.yes;
    }
}

private double getProbability(double x, double mean, double stdDev) { //using an
assumed normal distribution
    double denominator = stdDev * Math.sqrt(2*Math.PI);
    double exponent = -1 * ( (Math.pow(x - mean, 2)) / (2*Math.pow(stdDev, 2)) );
    double probability = (1/denominator) * Math.pow(Math.E, exponent);
    return probability;
}
}

```

K-Nearest Neighbour

```

import java.util.ArrayList;

public class KNearestNeighbours {
    private ArrayList<Person> training = new ArrayList<Person>();
    private int neighbours; //as in K nearest neighbours

    public KNearestNeighbours(ArrayList<Person> training, int neighbours) {
        this.training = training;
        this.neighbours = neighbours;
    }

    public Outcome Test(Person test) {
        ArrayList<Person> nearestNeighbours = new ArrayList<Person>();

        int i;
        int j = 0; //index of furthest neighbour
        double furthest = 0; //distance of furthest neighbour
        for(i = 0; i < neighbours; i++) {
            nearestNeighbours.add(training.get(i));
            if (getDistance(test, training.get(i)) > furthest) {
                j = i;
                furthest = getDistance(test, training.get(i));
            }
        }
    }
}

```

```

for(i = neighbours; i < training.size(); i++) { //find the furthest
    if(getDistance(test, training.get(i)) < furthest) {
        nearestNeighbours.remove(j);
        nearestNeighbours.add(training.get(i));
        j = 0;
        furthest = 0;
        for(int k = 0; k < neighbours; k++) { //find the new furthest
            if (getDistance(test, nearestNeighbours.get(k)) > furthest) {
                j = k; //index of the furthest neighbour
                furthest = getDistance(test, nearestNeighbours.get(k));
            }
        }
    }
}

int yesCount = 0;
int noCount = 0;
for(Person p : nearestNeighbours) {
    if(p.getOutcome() == Outcome.yes) {
        yesCount++;
    }
    else {
        noCount++;
    }
}

if(yesCount < noCount) {
    return Outcome.no;
}
else {
    return Outcome.yes;
}
}

```

```

//Returns the Euclidean distance between two People
private double getDistance(Person example, Person test) {
    double exampleAttributes[] = example.getAttributes();
    double testAttributes[] = test.getAttributes();
    double attributeDelta[] = new double[testAttributes.length];
    for(int i = 0; i < attributeDelta.length; i++) {
        attributeDelta[i] = exampleAttributes[i] - testAttributes[i];
    }
}

```

```

    }
    double sumOfSquares = 0;

    for(int i = 0; i < attributeDelta.length; i++) {
        sumOfSquares += Math.pow(attributeDelta[i], 2);
    }
    double euclidean = Math.sqrt(sumOfSquares);
    return euclidean;
}
}

```

K-means Clustering

```

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class Kmeans2d {

    public Kmeans2d() {
        System.out.println("Start");
    }

    public static List<Piste> GetKmeans2d(List<Piste> pisteet, int k,
        int maxIter) throws Exception {
        if (pisteet.size() < k) {
            throw new Exception("");
        }

        List<Piste> keskipisteet = new ArrayList<Piste>(k);
    }

```

```

int i = 0;
while (i < k) {
    int pointer = (int) (Math.random() * (pisteet.size() + 1));

    Piste current = pisteet.get(pointer);
    if (!keskipisteet.contains(current)) {
        current.Group = i;
        keskipisteet.add(current);
        i++;
    }
}

```

```

int iter = 0;
boolean changed = false;
do {

    changed = liitaLahimpaanKeskukseen(pisteet, keskipisteet);

    keskipisteet = paivitaKeskipisteet(k, pisteet);

    iter++;
} while ((iter < maxIter) && changed);

return keskipisteet;
}

```

```

public static int
    lahinKeskus(Piste vertailupiste, List<Piste> keskipisteet) {
    Piste min = keskipisteet.get(0);
    double dis = Double.POSITIVE_INFINITY;
    for (Piste p : keskipisteet) {

        double d = etaisyydenNelio(vertailupiste, p);
        if (d < dis) {
            min = p;
            dis = d;
        }
    }
    return min.Group;
}

```

```
}
```

```
public void loaddata(String path) throws IOException
```

```
{
```

```
    int count = 0;
```

```
    String file = path;
```

```
    List<String[]> content = new ArrayList<>();
```

```
    try(BufferedReader br = new BufferedReader(new FileReader(file))) {
```

```
        String line = "";
```

```
        while ((line = br.readLine()) != null) {
```

```
            content.add(line.split(", "));
```

```
        }
```

```
    } catch (FileNotFoundException e) {
```

```
        //Some error logging
```

```
    }
```

```
}
```

```
public static boolean liitaLahimpaanKeskukseen(List<Piste> pisteet,
```

```
        List<Piste> keskipisteet) {
```

```
    boolean changed = false;
```

```
    for (Piste p : pisteet) {
```

```
        int oldGroup = p.Group;
```

```
        p.Group = lahinKeskus(p, keskipisteet);
```

```
        if (oldGroup != p.Group) {
```

```
            changed = true;
```

```
        }
```

```
    }
```

```
    return changed;
```

```
}
```

```
public static double etaisydenNelio(Piste a, Piste b) {
```

```
    return Math.pow(a.X - b.X, 2) + Math.pow(a.Y - b.Y, 2);
```

```
}
```

```
public static List<Piste> paivitaKeskipisteet(int k,
```

```
        List<Piste> pisteet) {
```

```
    List<double[]> keskipisteet = new ArrayList<double[]>(k);
```

```
    for (int i = 0; i < k; i++) {
```

```
        double[] kp = new double[3];
```

```
        kp[0] = 0;
```

```
        kp[1] = 0;
```

```

        kp[2] = 0;
        keskipisteet.add(kp);
    }

    for (Piste piste : pisteet) {
        double[] keski = keskipisteet.get(piste.Group);

        keski[0] += piste.X;
        keski[1] += piste.Y;
        keski[2] += 1;
    }

    List<Piste> uudetKeskipisteet = new ArrayList<Piste>(k);
    int kIndex = 0;
    for (double[] kp : keskipisteet) {
        Piste p = new Piste();
        p.X = kp[0] / kp[2];
        p.Y = kp[1] / kp[2];
        p.Group = kIndex;
        kIndex++;
        uudetKeskipisteet.add(p);
    }

    return uudetKeskipisteet;
}
}

```

MainFrame

```

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;

```

```

import java.io.IOException;
import java.util.ArrayList;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.jfree.ui.RefineryUtilities;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Satheeshkumar S
 */
public class MainJFrame extends javax.swing.JFrame {

    /**
     * Creates new form MainJFrame
     */
    public MainJFrame() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-
BEGIN: initComponents
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();

```

```

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jButton1.setText(" Naive Bayes");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setText("KNN");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jButton3.setText("Result");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

jButton4.setText("SVM");
jButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton4ActionPerformed(evt);
    }
});

jLabel1.setText("Classification Algorithms");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(layout.createSequentialGroup()
                    .add(layout.createSequentialGroup()
                        .addContainerGap()

```

```

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
    .addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addComponent(jButton4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addComponent(jButton2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
    .addComponent(jButton1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 144, Short.MAX_VALUE)))
.addContainerGap(27, Short.MAX_VALUE))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(19, 19, 19)
        .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 31,
Short.MAX_VALUE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(35, 35, 35))
    );

pack();
} // </editor-fold> // GEN-END: initComponents

```

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { // GEN-
FIRST:event_jButton1ActionPerformed

```

```

    BufferedReader reader = null;
    try {

```

// TODO add your handling code here:

```
ArrayList<Person> training = new ArrayList<Person>();
ArrayList<Person> testing = new ArrayList<Person>();
String trainingFile = "dataset.csv";
String testingFile = "train.csv";
// String algorithm = "KNN";
String line;
reader = new BufferedReader(new FileReader(trainingFile));
if(testingFile.equals("FOLD")) { //used for testing 10-fold stratification;
    int errors[] = new int[10];

    for (int i = 1; i <= 10 ; i++) { //run the test 10 times
        reader = new BufferedReader(new FileReader(trainingFile));
        while((line = reader.readLine()) != null) {
            if(line.equals("fold" + i)) { //use fold i as testing data
                while((line = reader.readLine()) != null) {
                    if(line.length() > 6) {
                        testing.add(new Person(line,0));
                    }
                    else {
                        break;
                    }
                }
            }
            else if(line.length() > 6) { //use other folds as training data
                training.add(new Person(line,0));
            }
        }
    }

    //if(algorithm.equals("NB")) {
        NaiveBayes search = new NaiveBayes(training); //initialise Naive Bayes classifier,
        i.e evaluate means, and standard deviations
        for(Person p : testing) { //Run the test cases
            if(search.Test(p) != p.getOutcome()) {
                errors[i-1] += 1;
            }
        }
        //}
        //else { //i.e algorithm is K Nearest Neighbours
        //Scanner scan = new Scanner(algorithm).useDelimiter("\\D+");
        int k = 2;
```

```

        KNearestNeighbours search1 = new KNearestNeighbours(training, k);
        for(Person p : testing) { //Run the test cases
            if(search1.Test(p) != p.getOutcome()) {
                errors[i-1] += 1;
            }
        }
        //}
        training.clear();
        testing.clear();
        reader.close();
    }
    double avg = 0;
    for(int i = 0; i < 10; i++) {
        System.out.println("errors for for fold[" + i + "] = " + errors[i]);
        avg += errors[i];
    }
    avg = avg/10;
    System.out.println("Average error rate is: " + avg + "/" + 76.8 ); //76.8 is average size
of a fold
    System.out.println("Accurcy is: " + (1 - avg/76.8)); //accuracy of the classifier
    return;
}
while((line = reader.readLine()) != null) { //parse the csv file for the training examples,
assuming each example has 8 attributes and an assigned class
    training.add(new Person(line, 0));
}
reader.close();
reader = new BufferedReader(new FileReader(testingFile)); //parse the csv file for the
testing examples, assuming each example has 8 attributes but no assigned class
while((line = reader.readLine()) != null) {
    testing.add(new Person(line, 1));
}
reader.close();
//if(algorithm.equals("NB")) {
final long startTime = System.nanoTime();
System.out.println("Naive Bayes classifier Result");
NaiveBayes search = new NaiveBayes(training); //initialise Naive Bayes classifier, i.e
evaluate means, and standard deviations
for(Person p : testing) { //Run the test cases
    if(search.Test(p) == Outcome.yes) {
        System.out.println("yes");
    }
}

```

```

        else {
            System.out.println("no");
        }
    }
    final long duration = System.nanoTime() - startTime;
    //}
    System.out.println(".....");
    //else { //i.e algorithm is K Nearest Neighbours
    //Scanner scan = new Scanner(algorithm).useDelimiter("\\D+");
    // System.out.println("K Nearest Neighbours classifier Result");
    // final long startTime1 = System.nanoTime();
    //int k = 2;
    // KNearestNeighbours search1 = new KNearestNeighbours(training, k);
    // for(Person p : testing) { //Run the test cases
    //     if(search1.Test(p) == Outcome.yes) {
    //         System.out.println("yes");
    //     }
    //     else {
    //         System.out.println("no");
    //     }
    // }
    // final long duration1 = System.nanoTime() - startTime1;
    //System.out.println(".....");
    // System.out.println("Accurcy is: Naive Bayes "+ (duration/(60*60))/60+"ms"); //accuracy
of the classifier
    //System.out.println("Accurcy is: K Nearest Neighbours "+ (duration1/(60*60))/60+"ms");
    //}
    return;
} catch (FileNotFoundException ex) {
    Logger.getLogger(MainJFrame.class.getName()).log(Level.SEVERE, null, ex);
} catch (IOException ex) {
    Logger.getLogger(MainJFrame.class.getName()).log(Level.SEVERE, null, ex);
} finally {
    try {
        reader.close();
    } catch (IOException ex) {
        Logger.getLogger(MainJFrame.class.getName()).log(Level.SEVERE, null, ex);
    }
}
}
} //GEN-LAST:event_jButton1ActionPerformed

```

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {GEN-
FIRST:event_jButton2ActionPerformed
    // TODO add your handling code here:
    BufferedReader reader = null;
    try {
        // TODO add your handling code here:

        ArrayList<Person> training = new ArrayList<Person>();
        ArrayList<Person> testing = new ArrayList<Person>();
        String trainingFile = "dataset.csv";
        String testingFile = "train.csv";
        // String algorithm = "KNN";
        String line;
        reader = new BufferedReader(new FileReader(trainingFile));
        if(testingFile.equals("FOLD")) { //used for testing 10-fold stratification;
            int errors[] = new int[10];

            for (int i = 1; i <= 10 ; i++) { //run the test 10 times
                reader = new BufferedReader(new FileReader(trainingFile));
                while((line = reader.readLine()) != null) {
                    if(line.equals("fold" + i)) { //use fold i as testing data
                        while((line = reader.readLine()) != null) {
                            if(line.length() > 6) {
                                testing.add(new Person(line,0));
                            }
                            else {
                                break;
                            }
                        }
                    }
                    else if(line.length() > 6) { //use other folds as training data
                        training.add(new Person(line,0));
                    }
                }
            }

            //if(algorithm.equals("NB")) {
            NaiveBayes search = new NaiveBayes(training); //initialise Naive Bayes classifier,
i.e evaluate means, and standard deviations
            for(Person p : testing) { //Run the test cases
                if(search.Test(p) != p.getOutcome()) {
                    errors[i-1] += 1;
                }
            }
        }
    }
}

```

```

    }
    //}
    //else { //i.e algorithm is K Nearest Neighbours
    //Scanner scan = new Scanner(algorithm).useDelimiter("\\D+");
    int k = 1;
    KNearestNeighbours search1 = new KNearestNeighbours(training, k);
    for(Person p : testing) { //Run the test cases
        if(search1.Test(p) != p.getOutcome()) {
            errors[i-1] += 1;
        }
    }
    //}
    training.clear();
    testing.clear();
    reader.close();
}
double avg = 0;
for(int i = 0; i < 10; i++) {
    System.out.println("errors for for fold[" + i + "] = " + errors[i]);
    avg += errors[i];
}
avg = avg/10;
System.out.println("Average error rate is: " + avg + "/" + 76.8 ); //76.8 is average size
of a fold
System.out.println("Accurcy is: " + (1 - avg/76.8)); //accuracy of the classifier
return;
}
while((line = reader.readLine()) != null) { //parse the csv file for the training examples,
assuming each example has 8 attributes and an assigned class
    training.add(new Person(line, 0));
}
reader.close();
reader = new BufferedReader(new FileReader(testingFile)); //parse the csv file for the
testing examples, assuming each example has 8 attributes but no assigned class
while((line = reader.readLine()) != null) {
    testing.add(new Person(line, 1));
}
reader.close();
//if(algorithm.equals("NB")) {
// final long startTime = System.nanoTime();
// System.out.println("Naive Bayes classifier Result");

```

```

        // NaiveBayes search = new NaiveBayes(training); //initialise Naive Bayes classifier, i.e
evaluate means, and standard deviations
        // for(Person p : testing) { //Run the test cases
            // if(search.Test(p) == Outcome.yes) {
                // System.out.println("yes");
            // }
            // else {
                // System.out.println("no");
            // }
        // }
        // final long duration = System.nanoTime() - startTime;
        //}
        // System.out.println(".....");
        //else { //i.e algorithm is K Nearest Neighbours
        //Scanner scan = new Scanner(algorithm).useDelimiter("\\D+");
        System.out.println("K Nearest Neighbours classifier Result");
        final long startTime1 = System.nanoTime();
        int k = 2;
        KNearestNeighbours search1 = new KNearestNeighbours(training, k);
        for(Person p : testing) { //Run the test cases
            if(search1.Test(p) == Outcome.yes) {
                System.out.println("yes");
            }
            else {
                System.out.println("no");
            }
        }
        final long duration1 = System.nanoTime() - startTime1;
        System.out.println(".....");
        // System.out.println("Accuracy is: Naive Bayes "+ (duration/(60*60))/60+"ms"); //accuracy
of the classifier
        // System.out.println("Accuracy is: K Nearest Neighbours "+ (duration1/(60*60))/60+"ms");
        //}
        return;
    } catch (FileNotFoundException ex) {
        Logger.getLogger(MainJFrame.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(MainJFrame.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        try {
            reader.close();
        } catch (IOException ex) {

```

```

        Logger.getLogger(MainJFrame.class.getName()).log(Level.SEVERE, null, ex);
    }
}
} //GEN-LAST:event_jButton2ActionPerformed

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_jButton3ActionPerformed
    // TODO add your handling code here:
    BufferedReader reader = null;
    try {
        // TODO add your handling code here:

        ArrayList<Person> training = new ArrayList<Person>();
        ArrayList<Person> testing = new ArrayList<Person>();
        String trainingFile = "dataset.csv";
        String testingFile = "train.csv";
        String algorithm = "KNN";
        String line;
        reader = new BufferedReader(new FileReader(trainingFile));
        if(testingFile.equals("FOLD")) { //used for testing 10-fold stratification;
            int errors[] = new int[10];

            for (int i = 1; i <= 10 ; i++) { //run the test 10 times
                reader = new BufferedReader(new FileReader(trainingFile));
                while((line = reader.readLine()) != null) {
                    if(line.equals("fold" + i)) { //use fold i as testing data
                        while((line = reader.readLine()) != null) {
                            if(line.length() > 6) {
                                testing.add(new Person(line,0));
                            }
                            else {
                                break;
                            }
                        }
                    }
                    else if(line.length() > 6) { //use other folds as training data
                        training.add(new Person(line,0));
                    }
                }
            }

            //if(algorithm.equals("NB")) {

```

```

        NaiveBayes search = new NaiveBayes(training); //initialise Naive Bayes classifier,
i.e evaluate means, and standard deviations
        for(Person p : testing) { //Run the test cases
            if(search.Test(p) != p.getOutcome()) {
                errors[i-1] += 1;
            }
        }
        //}
        //else { //i.e algorithm is K Nearest Neighbours
        //Scanner scan = new Scanner(algorithm).useDelimiter("\\D+");
        int k = 2;
        KNearestNeighbours search1 = new KNearestNeighbours(training, k);
        for(Person p : testing) { //Run the test cases
            if(search1.Test(p) != p.getOutcome()) {
                errors[i-1] += 1;
            }
        }
        //}
        training.clear();
        testing.clear();
        reader.close();
    }
    double avg = 0;
    for(int i = 0; i < 10; i++) {
        System.out.println("errors for for fold[" + i + "] = " + errors[i]);
        avg += errors[i];
    }
    avg = avg/10;
    System.out.println("Average error rate is: " + avg + "/" + 76.8 ); //76.8 is average size
of a fold
    System.out.println("Accurcy is: " + (1 - avg/76.8)); //accuracy of the classifier
    return;
}
while((line = reader.readLine()) != null) { //parse the csv file for the training examples,
assuming each example has 8 attributes and an assigned class
    training.add(new Person(line, 0));
}
reader.close();
reader = new BufferedReader(new FileReader(testingFile)); //parse the csv file for the
testing examples, assuming each example has 8 attributes but no assigned class
while((line = reader.readLine()) != null) {
    testing.add(new Person(line, 1));
}

```

```

    }
    reader.close();
    //if(algorithm.equals("NB")) {
    final long startTime = System.nanoTime();
    // System.out.println("Naive Bayes classifier Result");
    NaiveBayes search = new NaiveBayes(training); //initialise Naive Bayes classifier, i.e
evaluate means, and standard deviations
    for(Person p : testing) { //Run the test cases
        if(search.Test(p) == Outcome.yes) {
            //System.out.println("yes");
        }
        else {
            //System.out.println("no");
        }
    }
    final long duration = System.nanoTime() - startTime;
    //}
    // System.out.println(".....");
    //else { //i.e algorithm is K Nearest Neighbours
    //Scanner scan = new Scanner(algorithm).useDelimiter("\\D+");
    // System.out.println("K Nearest Neighbours classifier Result");
    final long startTime1 = System.nanoTime();
    int k = 1;
    KNearestNeighbours search1 = new KNearestNeighbours(training, k);
    for(Person p : testing) { //Run the test cases
        if(search1.Test(p) == Outcome.yes) {
            // System.out.println("yes");
        }
        else {
            // System.out.println("no");
        }
    }
    final long duration1 = System.nanoTime() - startTime1;

    final long startTime2 = System.nanoTime();
    SSMO s=new SSMO();
    String[] args = null;
    s.main(args);
    try {
        s.svm();
    } catch (IOException ex) {
        Logger.getLogger(MainJFrame.class.getName()).log(Level.SEVERE, null, ex);
    }

```



```

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MainJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MainJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MainJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new MainJFrame().setVisible(true);
        }
    }

```

```
    });  
}  
  
// Variables declaration - do not modify//GEN-BEGIN:variables  
private javax.swing.JButton jButton1;  
private javax.swing.JButton jButton2;  
private javax.swing.JButton jButton3;  
private javax.swing.JButton jButton4;  
private javax.swing.JLabel jLabel1;  
// End of variables declaration//GEN-END:variables  
}
```

Enhanced Skin Condition Prediction through Machine Learning

C.V.S. Rahul Reddy, R. Shyam Prasad, Jithina Jose

Department of Computer Science and Engineering

Sathyabama Institute Of Science And Technology.

ABSTRACT

In recent days, skin diseases are increased in humans. Skin diseases are caused by bacteria or due to infections. Some of the skin diseases are like ring worm, yeast infection, allergies etc. are increases and spread over skin day by day. So this type of diseases should be identified in its earlier stage to avoid spreading. It can be identified using some factors like clinical parameters which are considered for identifying the disease. The possible skin diseases in different ages are dermatitis in age 0-5 years, warts affects in 6- 11 years age, and acne vulgaris in 12-16 years age. Dermatomyositis is a type of skin disease that affects children at age of 5-15 and adult at 40-60 age.

AIM

The main aim of this system, early disease recognition, patient care and community services. When the quality of medical data is incomplete the exactness of study is reduced.

OBJECTIVE

The analysis accuracy is reduced when the quality of medical data is incomplete.

Moreover, different regions exhibit unique characteristics of certain regional diseases, which may weaken the prediction of disease

outbreaks. However, those existing work mostly considered structured data. There is no proper methods to handle semi structured and unstructured. The proposed system will consider both structured and unstructured data. The analysis accuracy is increased by using Machine Learning algorithm.

SCOPE

In this work, our goal is to provide a tool to assist professionals and consumers in finding and choosing disease. To achieve this goal, we develop an approach that allows a user to query for disease that satisfy a set of conditions based on disease properties, such as disease indications.

INTRODUCTION

There are different techniques to diagnose skin disease like Malignant melanoma i.e. melanoma skin tumors, Nonmelanoma i.e. squamous cell carcinoma, basal cell carcinoma, Acne, Genetic Diseases i.e. Sickle-Cell Anemia Genetic Skin Disorder, Fungal Diseases, Bacterial Diseases, Psoriasis and Leprosy. There are different techniques to diagnose these skin diseases. But the most useful and noninvasive method for diagnosis of different types of skin diseases is skin impedance measurement. The analysis is made with the measurement of skin impedance over a wide frequency range from 100 Hz to 1MHz. But this analysis for the infected skin and normal skin should be made at same frequency

because at different frequencies skin impedance can be different. The measured skin impedance at same frequency of infected skin and normal skin is different, because of which we can diagnose skin disease. Above mentioned method used the non-invasive probe electrode system consists of two concentric electrodes attached to a ceramic plate. The outermost electrodes drive the voltage and inner is sink electrode gives output. In which source electrode provide small amount of current to skin and sink electrode gives input to the microcontroller. Length of the outermost electrode is nearly 10mm. Measurement of skin impedance can be used for depressions, anxiety, neurological diseases, and nerve lesions. It is also used for physiological measurements emotional disorders or lie detection. The various types of bioimpedance method include multiple frequency, single frequency and Bio impedance spectroscopy .SF-BIA is generally performed at a frequency of 50 kHz. At this frequency the current passes through both extracellular and intracellular fluid. Bio-impedance obtained at number of frequencies is analyzed in multiple-frequency bio-impedance analysis.

LITERATURE SURVEY

Segmentation and Classification of Skin Lesions for Disease Diagnosis

In this paper, a novel approach for automatic segmentation and classification of skin lesions is proposed. Initially, skin images are filtered to remove unwanted hairs and noise and then the segmentation process is carried out to extract lesion areas. For segmentation, a region growing method is applied by automatic initialization of seed points. The segmentation performance is measured with different well known measures and the results are appreciable. Subsequently, the extracted lesion areas are represented by color and texture

features. SVM and k-NN classifiers are used along with their fusion for the classification using the extracted features. The performance of the system is tested on our own dataset of 726 samples from 141 images consisting of 5 different classes of diseases. The results are very promising with 46.71% and 34% of F-measure using SVM and k-NN classifier respectively and with 61% of F-measure for fusion of SVM and k-NN.

Extraction of Skin Lesions from Non-Dermoscopic Images Using Deep Learning

Melanoma is amongst most aggressive types of cancer. However, it is highly curable if detected in its early stages. Prescreening of suspicious moles and lesions for malignancy is of great importance. Detection can be done by images captured by standard cameras, which are more preferable due to low cost and availability. One important step in computerized evaluation of skin lesions is accurate detection of lesion region, i.e. segmentation of an image into two regions as lesion and normal skin. Accurate segmentation can be challenging due to burdens such as illumination variation and low contrast between lesion and healthy skin. In this paper, a method based on deep neural networks is proposed for accurate extraction of a lesion region. The input image is preprocessed and then its patches are fed to a convolutional neural network. Local texture and global structure of the patches are processed in order to assign pixels to lesion or normal classes. A method for effective selection of training patches is used for more accurate detection of a lesion border. The output segmentation mask is refined by some post processing operations. The experimental results of qualitative and quantitative evaluations demonstrate that our method can outperform other state-of-the-art algorithms exist in the literature.

AN INTELLIGENT SYSTEM TO DIAGNOSIS THE SKIN DISEASE

Skin cancer is a major health issue affecting a vast segment of the population regardless the

skin color. This affectation can be detected using dermoscopy to determine whether the visible spots on skin are either benign or malignant tumors. In spite of the specialists' experience, skin lesions are difficult to classify, reason for which computer systems are developed to increase the effectiveness of cancer detection. Systems assisting in the detection of skin cancer process digital images to determine the occurrence of tumors by interpreting clinical parameters, relying, firstly, upon an accurate segmentation process to extract relevant features. Two of the well-known methods to analyze lesions are ABCD and the 7-point check list. After clinically-relevant features are extracted, they are used to classify the presence or absence of a tumor. However, irregular and disperse lesion borders, low contrast, artifacts in images and the presence of various colors within the region of interest complicate the processing of images. In this article, we propose an intelligent system running the following method. The feature extraction stage begins with the segmentation of an image, for which we apply the Wavelet – Fuzzy C-Means algorithm. Next, specific features should be determined, among others the area and the asymmetry of the lesion. An ensemble of clusterers extracts the Red-Green-Blue values that correspond to one or more of the colors defined in the ABCD guide. The feature extraction stage includes the discovery of structures that appear in the lesion according to the method known as Grey Level CoOccurrence Matrix. Then, during the detection phase, an ensemble of classifiers determines the occurrence of a

PROPOSED SYSTEM

In proposed system, KNN and Naïve Bayes techniques to detect various kinds of skin diseases. It can be identified using some factors like clinical parameters which are considered for identifying the disease. The possible skin diseases in different ages are dermatitis in age 0-5 years, warts affects in 6-

malignant tumor. Our experiments are performed on images taken from the ISIC repository. The proposed system provides a skin cancer detection performance above 88 percent, as measured by the accuracy. Details of how this performance fares when compared with other systems are also given.

A SURVEY ON SKIN TEXTURE ANALYSIS FOR MEDICAL DIAGNOSIS USING IMAGE PROCESSING TECHNIQUES

Skin texture analysis is one of then challenging issues in the field of medical diagnosis. Various types of skin diseases are affecting human life like skin dryness, fungus, and allergic symptoms. The objective of this paper is to analyze the skin disease using texture analysis of skin image and by comparing the test image to a defined images or reference images. The matching of test and reference images compared that yields the percentage of skin diseases in the captured skin texture image.

EXISTING SYSTEM

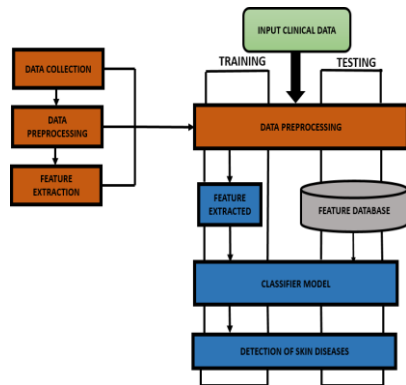
In this article, existing method that uses computer vision based techniques to detect various kinds of dermatological skin diseases. We have used different types of machine learning algorithms for feature extraction and feed forward artificial neural network for training and testing purpose. The system works on two phases- first pre-process the colour skin images to extract significant features and later identifies the diseases. The system successfully detects 9 different types of dermatological skin diseases with an accuracy rate of 90%.

11 years age, and acne vulgaris in 12-16 years age. Dermatomyositis is a type of skin disease that affects children at age of 5-15 and adult at 40-60 age. We are giving a suggestion of what kind of treatment you want and also showing life thread or not.

ADVANTAGES:

- Easily analyze the disease

- High Security



SYSTEM ARCHITECTURE

SYSTEM REQUIREMENTS HARDWARE REQUIREMENTS

System	:	Pentium Dual
Core. Hard Disk	:	120 GB.
Monitor	:	15"LED
Input Devices	:	Keyboard,
Mouse Ram	:	1GB.

SOFTWARE REQUIREMENTS

Operating system	:	Windows 7.
Coding Language	:	Java
Toolkit	:	Netbeans
DATABASE	:	MySql

MODULES

- User Module
- Admin Module
- Disease prediction Module
- Best drug recommendation

USER MODULE

In this Module, Patient can act as a user.

1. Login

This is the first activity that opens the website. User needs to provide a correct contact number and a password, which user enters while registering, in order to login into the app. If information provided by the user matches with the data in the database table then user successfully login into the app else message of login failed is displayed and user need to reenter correct information. A link to the register activity is also provided for registration of new users.

2. Registration

A new user who wants to access the website needs to register first before login. By clicking on register button in login activity, the register activity gets open. A new user registers by entering full name, password and contact number. A user needs to enter password again in confirm password textbox for confirmation. When user enters the information in all textboxes, on the click of register button, the data is transferred to database and user is directed to login activity again. Registered user then needs to login in order to access the app. Validations are applied on all the textboxes for proper functioning of the app. Like information in each textbox is must that is each textbox, either it is of name, contact, password or confirm password, will not be empty while registering. If any such textbox is empty app will give message of information is must in each textbox. Also data in password and confirm password fields must match for successful registration. Another validation is contact number must be valid one that is of 10 digits. If any such validation is violated then registration will be unsuccessful and then user needs to register again.

ADMIN MODULE

In this module, admin can add and view new disease details and drug details. And then admin can view feedback provided by various users.

DISEASE PREDICTION MODULE

Patient will specify the symptoms caused due to his illness. System will ask certain question regarding his illness and system predict the skin disease based on the symptoms and age specified by the patient and system will also suggest drugs based on the disease.

BEST DRUG RECOMMENDATION

In this module we develop an app that allows a user to query for drugs that satisfy a set of conditions based on drug properties, such as drug indications, side effects, and drug

interactions, and also takes into account patient profiles. In this we automate the suggestion of the alternative drugs. So that we can provide the medicine to diseases i.e. a preferred medicine .so the researcher will analysis the dosage of the drug and the symptoms.

CONCLUSION

In this work a model for prediction of skin diseases is done using deep learning algorithms. It is found that by using the ensembling features and deep learning we can achieve a higher accuracy rate and also we can go for the prediction of many more diseases than with any other previous models done before. As the previous models done in this field of application were able to report a maximum of six skin diseases with a maximum accuracy level of 75%. By implementing deep learning algorithm we are able to predict as many as 20 diseases with a higher accuracy level of 88%. This proves that deep learning algorithms have a huge potential in the real world skin disease diagnosis. Future work can be extended to make this model a standard procedure for preliminary skin disease diagnosis method as it will reduce the treatment and diagnosis time.

REFERENCES

- [1] Sumithra Ra, Mahamad Suhilb, D.S.Guruc," Segmentation and Classification of Skin Lesions for Disease Diagnosis", doi: 10.1016/j.procs.2015.03.090, 2015.
- [2] Mohammad H. Jafari, Ebrahim Nasr-Esfahani, Nader Karimi, S.M. Reza Soroushmehr, Shadrokh Samavi, Kayvan Najarian, " Extraction of Skin Lesions from Non-Dermoscopic Images Using Deep Learning", ResearchGate, September 2016.
- [3] Manish Kumar and Rajiv Kumar, "AN INTELLIGENT SYSTEM TO DIAGNOSIS THE SKIN DISEASE", VOL. 11, NO. 19, OCTOBER 2016.
- [4] Jerrey Glaister," Automatic segmentation of skin lesions from dermatological photograph", Waterloo, Ontario, Canada, 2013.

[5] K.INDUPRIYA, Dr. G. P. RAMESH KUMAR, "A SURVEY ON SKIN TEXTURE ANALYSIS FOR MEDICAL DIAGNOSIS USING IMAGE PROCESSING TECHNIQUES", K. INDUPRIYA et al. Volume 3 Issue 5, 2015.

[6] Md Nafiul Alam, Tamanna Tabassum Khan Munia, Kouhyar Tavakolian, Vasefi, Nick MacKinnon, Reza Fazel-Rezai, "Automatic Detection and Severity Measurement of Eczema Using Image Processing", IEEE, 2016.

[7] L.Wei, Q. Gan, and T. Ji, "Skin Disease Recognition Method Based on Image Color and Texture Features," Computational and Mathematical Methods in Medicine Volume 2018, Article ID 8145713, pp.2-3.

[8] N. Abbadi, N. SaadiDahir, M. Dhalimi and H. Restom," Psoriasis Detection Using Skin Color and Texture Features" Journal of Computer Science 6 (6): 648-652, 2010 ISSN 15493636 © 2010 Science Publications, pp.648.

[9] S. Ershad¹, M. Saberi and F. Tajeripour,"An innovative skin detection approach using color based image retrieval technique", The International Journal of Multimedia & Its Applications (IJMA) Vol.4, No.3, 2012, pp.1207.

[10] S. Arivazhagan, R. Shebiah, K. Divya, and M. Subadevi, "Skin disease classification by extracting independent components," Journal of Emerging Trends in Computing and Information Sciences, vol. 3, no. 10, pp. 1379– 1382, 2012.

[11] F. Liu and H. Guo, "Research progress of CT in skin diseases," Chinese Journal of Dermatovenereal Integrated Traditional Western Medicine, vol. 313, no. 3, pp. 189–

191, 2014.

[12] R. Sumithra, S. Mahamad , D. Guru, "Segmentation and Classification of Skin Lesions for Disease Diagnosis," International Conference on Advanced Computing Technologies and Applications (ICACTA2015), pp.1609.

[13] M. Ganeshkumar and J. Vasanthi, "Skin disease identification using image segmentation," International Journal of Innovative Research in Computer and Communication Engineering, vol. 5, no. 1, pp. 154–160, 2017.