

DERMATOLOGICAL MANIFESTATIONS SEGMENTATION USING MACHINE LEARNING

**Submitted in partial fulfilment of the requirements for the
award of**

**Bachelor of Engineering Degree in
Computer Science and Engineering**

By

NEERAJA LAKSHMI KOPPOLU (38110264)

TATI AKHILA (38110019)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MARCH 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY



Accredited with "A" grade by NAAC
(DEEMED TO BE UNIVERSITY)

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 work of **Neeraja Lakshmi Koppolu (38110264)**, **Tati Akhila (38110019)** who carried out the project entitled "**DERMATOLOGICAL MANIFESTATIONS SEGMENTATION USING MACHINE LEARNING**" under my supervision November 2021 from to March 2022.

Internal Guide

Dr. K. Mohana Prasad, M.E., Ph.D.,

Head of the Department

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

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We **NEERAJA LAKSHMI KOPPOLU (38110264), TATI AKHILA (38110019)** Here by declare that the Project Report entitled “**DERMATOLOGICAL MANIFESTATIONS SEGMENTATION USING MACHINE LEARNING**” **done** by us under the guidance of **Dr. K. MOHANA PRASAD, M.E.,Ph.D.,** at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing Dr. L.Lakshmanan, M.E., Ph.D., and Dr. S. Vigneswari, M.E., Ph.D., Heads** of the Department 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 **Dr. K. MOHANA PRASAD, M.E., Ph.D.,** for his 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 have been significantly increasing in humans. Skin diseases are manifested either as infections or allergies. Some of the skin diseases such as ring worm infections, yeast infections, viral infections, eczemas etc. are spreading rapidly day by day. These type of diseases need to be identified in their earlier stages to avoid spreading. Some factors like clinical parameters 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 adults at 40-60 age.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF TABLES	ix
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
1.1.	Domain overview	2
1.2	Machine learning	3
1.3	Supervised learning	5
1.4	Regression	8
2	LITERATURE SURVEY	9
2.1	Segmentation and classification of skin lesions	9
2.2	Extraction of skin lesions from non dermoscopic images	9
2.3	An intelligent system to diagnosis the skin disease	10
2.4	Automatic segmentation of skin lesions from dermatological photographs	12
2.5	Survey on skin texture analysis	13
2.6	Existing system	13
2.7	Proposed system	14
3	AIM AND OBJECTIVES	15

TITLE	CHAPTER	PAGE NO
4	SYSTEM ANALYSIS	17
4.1	Existing system for system analysis	17
4.2	Proposed system for system analysis	17
4.3	System architecture	18
4.4	System requirements	19
4.5	Dataset	20
5	ALGORITHMS USED	21
5.1	KNN classification method	21
5.2	Naïve bayes method	27
6	SYSTEM IMPLEMENTATION	33
6.1	User module	33
6.2	Admin module	34
6.3	Disease prediction module	34
6.4	Best drug recommendation	35
7	DIAGRAMATIC REPRESENTATIONS	36
7.1	ER diagram	36
7.2	UML diagram	37
7.3	Use case diagram	38
7.4	Class diagram	39
7.5	Activity diagram	40
7.6	Sequence diagram	41
7.7	Collaboration diagram	42
8	RESULTS	43
9	CONCLUSION	50
10	REFERENCES	51
	APPENDIX	52
	A. SAMPLE CODING	52

LIST OF FIGURES

NO	TITLE	PAGE NO
1	SUPERVISED ARCHITECTURE	5
2	BLOCK DIAGRAM	7
3	SCHEMATIC REPRESENTATION OF SYSTEM ARCHITECTURE	18
4	DATASET	20
5	COLORS OF THE 3 FLOWER CATEGORIES IN OUR EXAMPLE DATASET, RED, YELLOW, AND GREEN	22
6	MEASURING THE DISTANCE BETWEEN THE CATEGORIES FROM OUR DATASET	23
7	SEEING HOW THE NEW DATA POINT BELONGS TO THE CATEGORY RED FROM OUR EXAMPLE DATASET	24
8	INTRODUCES A NEW DATA POINT TO ILLUSTRATE THE K-NEAREST NEIGHBOR ALGORITHM	25
9	ER DIAGRAM	36
10	USE CASE DIAGRAM	38
11	CLASS DIAGRAM	39
12	ACTIVITY DIAGRAM	40
13	SEQUENCE DIAGRAMS	41
14	COLLABORATION DIAGRAM	42
15	LOGIN	43
16	VALID USER	44
17	INVALID USER	45
18	PREDICTION	46
19	INPUT	47
20	PREDICTING THE SKIN DISEASE	48
21	RESULT	49

LIST OF TABLES

NO	TITLE	PAGE NO
1	LIKELIHOOD TABLE	15

LIST OF ABBREVIATIONS

SUPPORT VETOR MACHINE	SVM
K-NEAREST NEIGHBOR	KNN
AUDIENCE, BEHAVIOR, CONDITION, AND DEGREE	ABCD
ENTITY RELATIONSHIP MODEL	ER
UNIFIED MODELING LANGUAGE	UML
SINGLEFREQUENCY BIOMPEDANCE ANALYSIS	SF-BIA

CHAPTER 1

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). 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.

1.1. DOMAIN OVERVIEW :

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyze the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions about the future. Data is the core backbone of machine learning algorithms. With the help of the historical data, we are able to create more data by training these machine learning algorithms. For example, Generative Adversarial Networks are an advanced concept of Machine Learning that learns from the historical images through which they are capable of generating more images. This is also applied towards speech and text synthesis. Therefore, Machine Learning has opened up a vast potential for data science applications.

1.2. MACHINE LEARNING

Machine Learning combines computer science, mathematics, and statistics. Statistics is essential for drawing inferences from the data. Mathematics is useful for developing machine learning models and finally, computer science is used for implementing algorithms. However, simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyper parameters to reach an optimum result.

The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries. Therefore, Machine Learning has become an essential part of modern industry. Data is expanding exponentially and in order to harness the power of this data, added by the massive increase in computation power, Machine Learning has added another dimension to the way we perceive information. Machine Learning is being utilized everywhere. The electronic devices you use, the applications that are part of your everyday life are powered by powerful machine learning algorithms.

With an exponential increase in data, there is a need for having a system that can handle this massive load of data. Machine Learning models like Deep Learning allow the vast majority of data to be handled with an accurate generation of predictions. Machine Learning has revolutionized the way we perceive information and the various insights we can gain out of it. These machine learning algorithms use the patterns contained in the training data to perform classification and future predictions. Whenever any new input is introduced to the ML model, it applies its learned patterns over the new data to make future predictions. Based on the final accuracy, one can optimize their models using various standardized approaches. In this way, Machine Learning model learns to adapt to new examples and produce better results.

Types of Machine Learning

Machine Learning Algorithms can be classified into 3 types as follows –

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

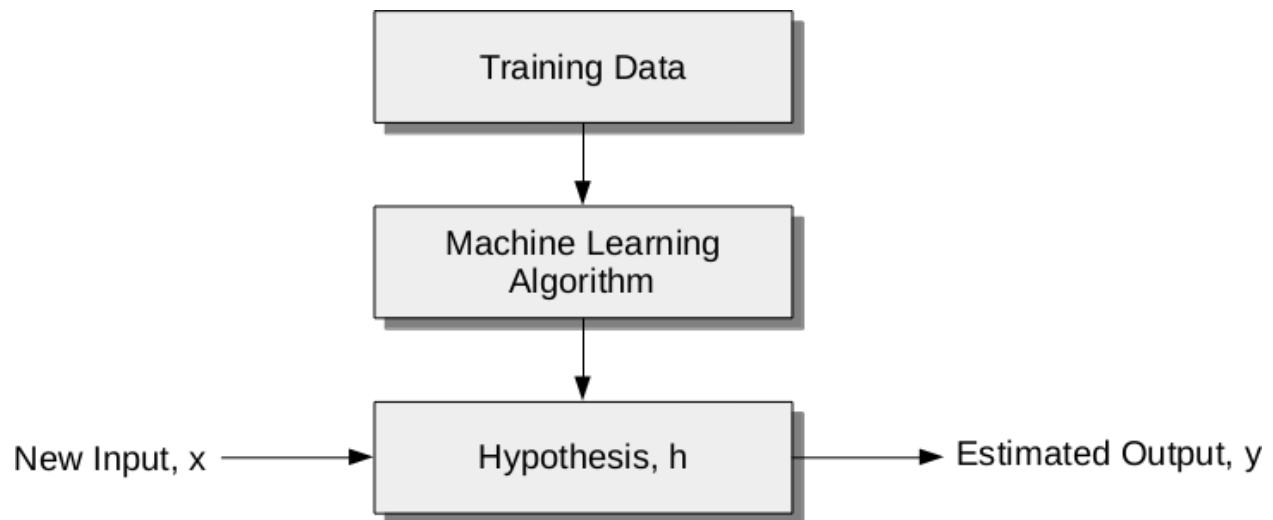


Fig -1 Supervised Architecture

SUPERVISED LEARNING

In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function $h(x)$ (sometimes called the “hypothesis”). “Learning” consists of using sophisticated mathematical algorithms to optimize this function so that, given input data x about a certain domain (say, square footage of a house), it will accurately predict some interesting value $h(x)$ (say, market price for said house).

$$h(x_1, x_2, x_3, x_4) = \theta_0 + \theta_1 x_1 + \theta_2 x_3^2 + \theta_3 x_3 x_4 + \theta_4 x_1^3 x_2^2 + \theta_5 x_2 x_3^4 x_4^2$$

This function takes input in four dimensions and has a variety of polynomial terms. Deriving a normal equation for this function is a significant challenge. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients. Predicting how an organism's genome will be expressed, or what the climate will be like in fifty years, are examples of such complex problems.

Under supervised ML, two major subcategories are:

- Regression machine learning systems: Systems where the value being predicted falls somewhere on a continuous spectrum.
- Classification machine learning systems: Systems where we seek a yes-or-no prediction.

In practice, x almost always represents multiple data points. So, for example, a housing price predictor might take not only square-footage (x_1) but also number of bedrooms (x_2), number of bathrooms (x_3), number of floors (x_4), year built (x_5), zip code (x_6), and so forth. Determining which inputs to use is an important part of ML design. However, for the sake of explanation, it is easiest to assume a single input value is used

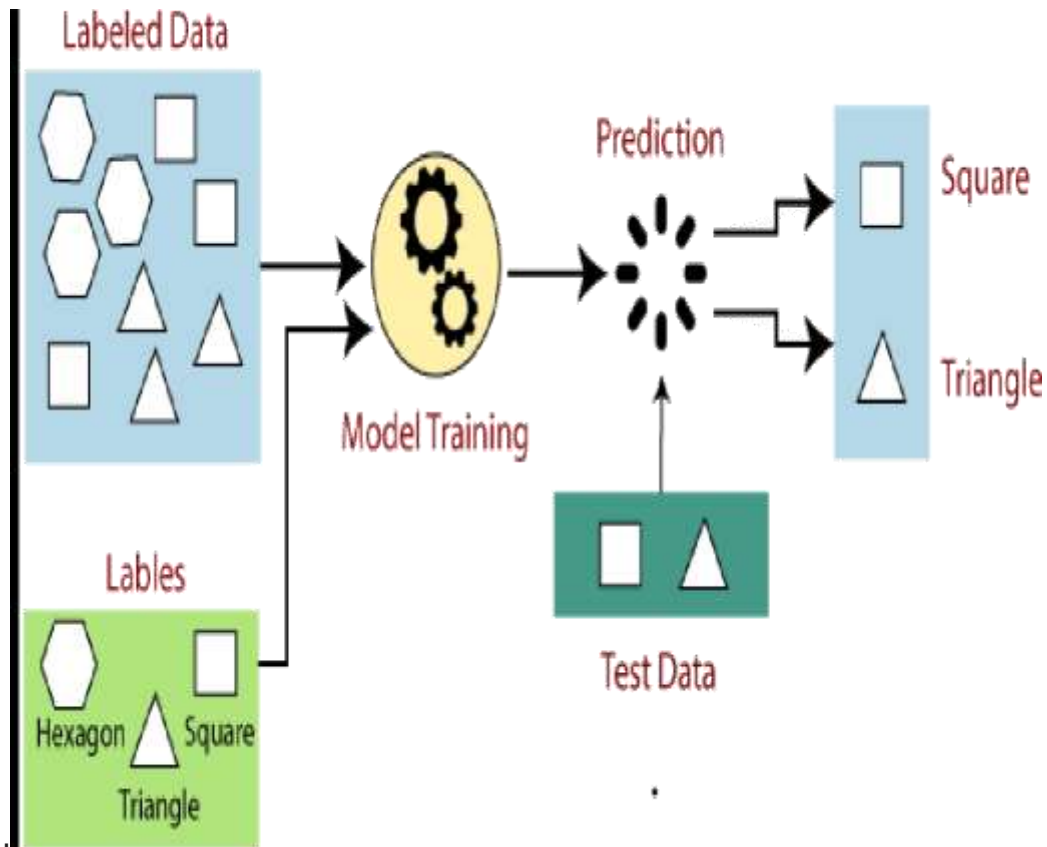


Fig-2 Block Diagram

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training dataset, test dataset, and validation dataset.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.

- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set.

REGRESSION

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc.

1. Linear Regression
2. Regression Trees
3. Non-Linear Regression
4. Bayesian Linear Regression
5. Polynomial Regression

CHAPTER 2

LITERATURE SURVEY

2.1. 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.

2.2. 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.

2.3. 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 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 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.

2.4. 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.

2.5. 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.

2.6. 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%.

Disadvantages of existing system

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

2.7. 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.

Advantages of proposed system:

- Easily analyze the disease
- High Security

CHAPTER 3

AIM AND OBJECTIVES

3.1. 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.

3.2. 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.

3.3 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.

CHAPTER 4

SYSTEM ANALYSIS

4.1. EXISTING SYSTEM

- In this project, 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%.

4.2. 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.

4.3. SYSTEM ARCHITECTURE

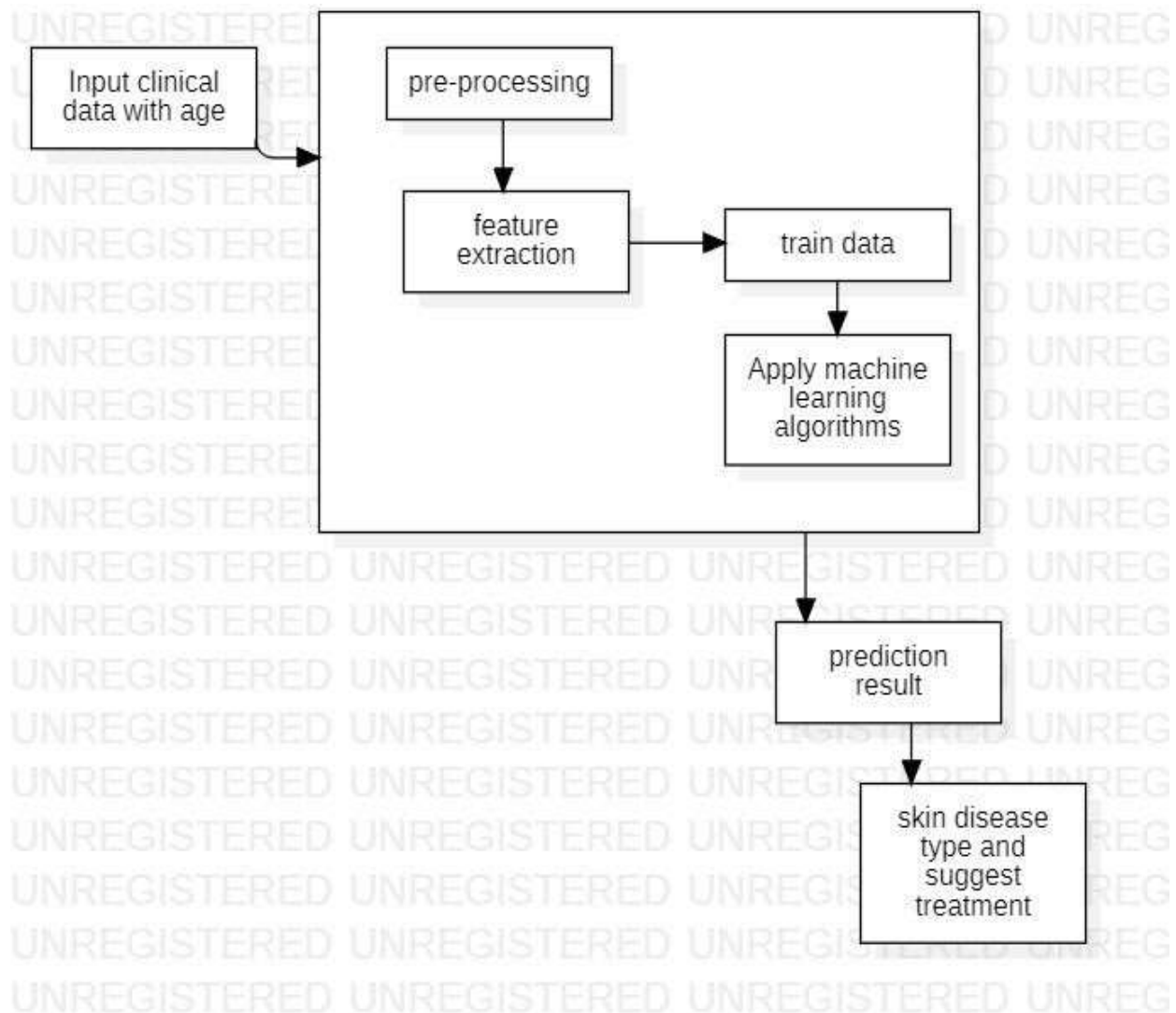


Fig 3: Schematic representation of system architecture

4.4. SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

- System : Pentium Dual Core.
- Hard Disk : 120 GB.
- Input Devices: Keyboard, Mouse
- Ram : 1GB.

SOFTWARE REQUIREMENTS:

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

4.5. DATASET

We have 700 + datasets in this Excel sheet, and it is downloaded in Kaggle website.

train - Excel

POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	126	51	20	25															
2	136	186	41	34															
3	136	120	39	23															
4	120	189	41	44															
5	132	170	43	50															
6	131	162	24	40															
7	119	138	25	46															
8	131	145	28	25															
9	135	101	23	32															
10	123	94	41	23															
11	132	166	26	38															
12	101	73	38	35															
13	101	95	47	42															
14	116	125	45	43															
15	143	79	28	23															
16	103	115	49	44															

Fig no 4: Dataset

CHAPTER 5

ALGORITHMS USED

5.1 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) is known

STEP 1: BEGIN

STEP 2: Input: $D = \{(x_1, c_1), \dots, (x_N, c_N)\}$

STEP 3: $x = (x_1, \dots, x_n)$ new instance to be classified

STEP 4: FOR each labelled instance (x_i, c_i) calculate $d(x_i, x)$

STEP 5: Order $d(x_i, x)$ from lowest to highest, $(i = 1, \dots, N)$

STEP 6: Select the K nearest instances to x : D_x^k

STEP 7: Assign to x the most frequent class in D_x^k

STEP 8: END

The KNN algorithm is a major classical machine learning algorithm that focuses on the distance from new points to existing labelled data points.

LABELLED DATA: In this data points have been labelled from our dataset and then plot them on a 2-dimensional graph; these data points belong to three categories represented by red, green, and yellow colors, as shown in figure 2.

UNLABELLED DATA: Next, a new unlabelled data point is considered and is represented by the black cross mark. Making it time to determine which category this new data point belongs to from the three colors. First, a random value, which is known to be the k-value is chosen. The k-value tells the number of nearest points to look for from the new unlabelled data point. Consider the k-value as 5. Next, the distance from the unlabelled data point to every data point on the graph is calculated and the top 5 shortest distances are selected

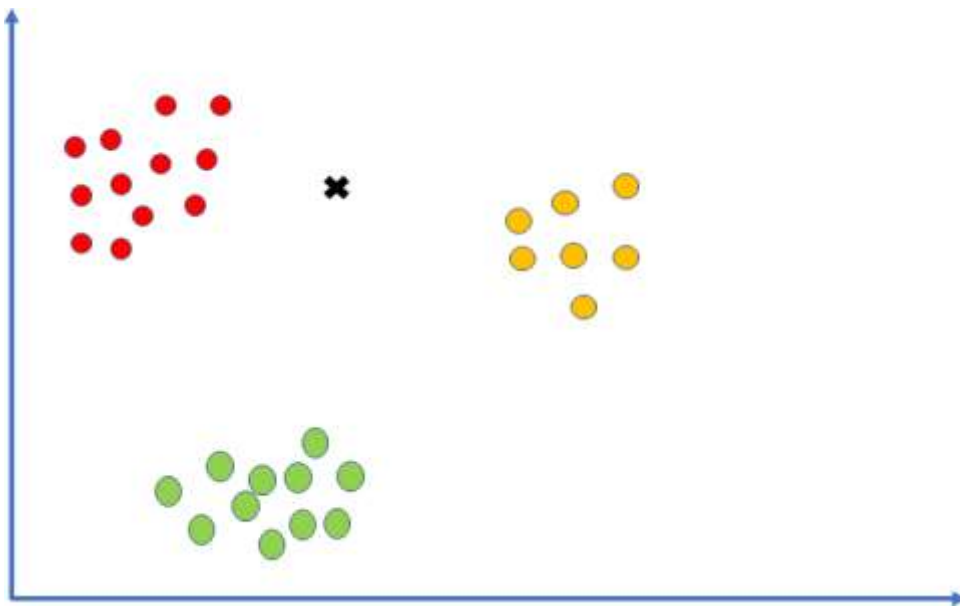


Fig 5: Colors of the three flower categories in our example dataset, red, yellow, and green.

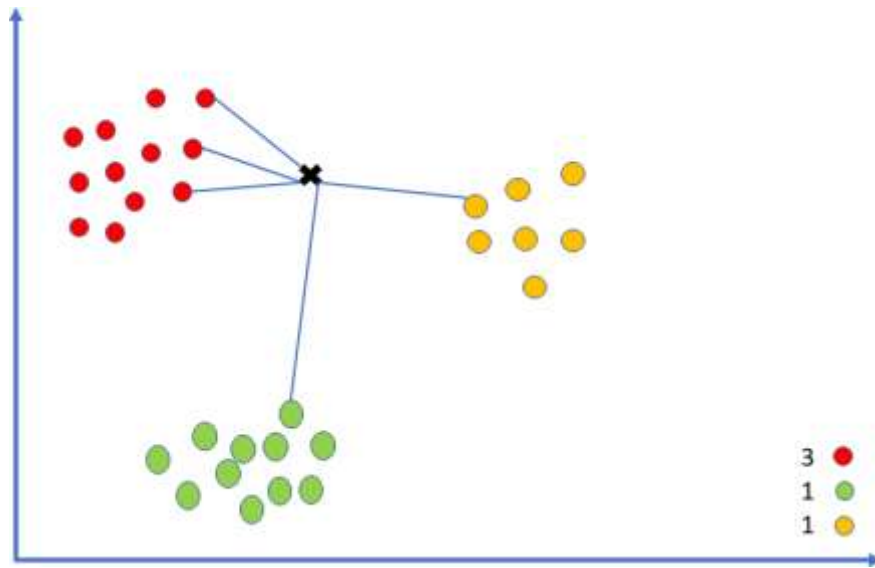


Fig 6: Measuring the distance between the categories from our dataset.

Amongst the top 5 nearest data points, 3 belong to category red, 1 belong to category green, and 1 belong to category yellow. These are known as $k(5)$ nearest neighbors of new data. It is now evident that the new data point belongs to category red as most of its nearest neighbors are from category red.

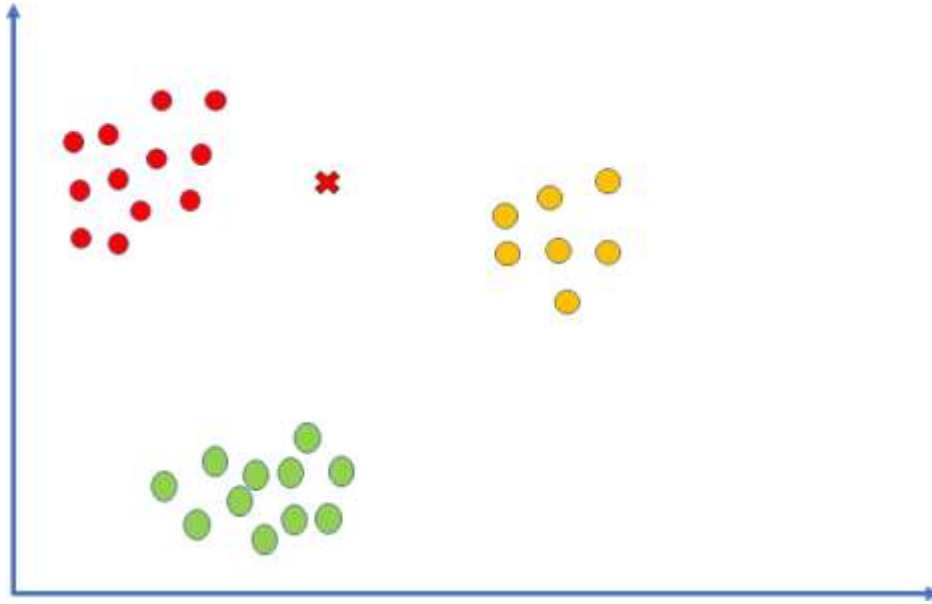


Fig 7: Seeing how the new data point belongs to the category red from our example dataset.

Similarly, in a regression problem, the aim is to predict a new data point's value, not the category to which it belongs. Again, to illustrate, plot a 2-dimensional graph consisting of data points from the given dataset. Since it is a 2-dimensional graph, there are 2 features for each data point. The x-axis represents feature-1, and the y-axis represents feature-2.

Next, we introduce a new data point for which only the feature-1 value is known, and we need to predict the feature-2 value. We take a k-value of 5 and get the 5 nearest neighboring points from the new data point. The predicted value of feature-2 for the new data point is the mean of the feature-2 of 5 nearest neighbors.

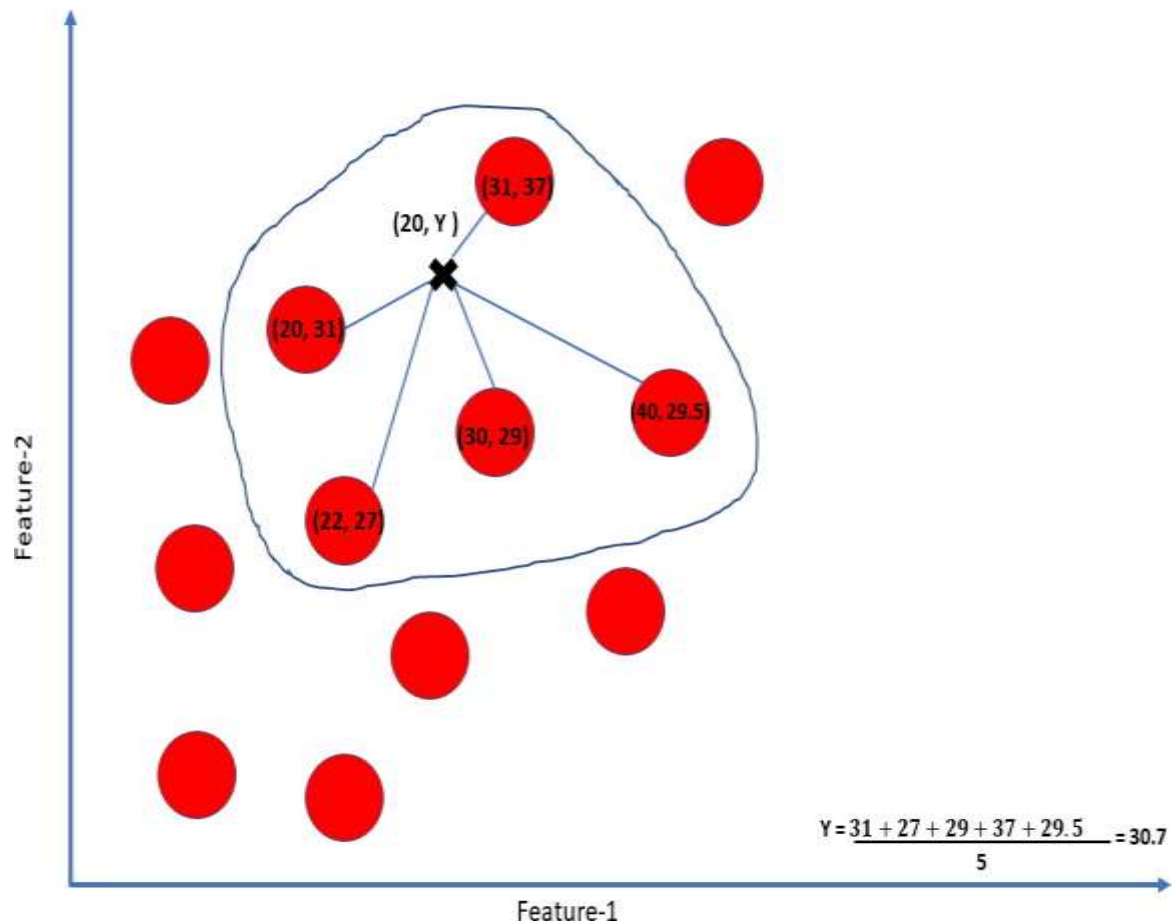


Fig 8: Our example introduces a new data point to illustrate the k-nearest neighbor (KNN) algorithm.

The KNN algorithm can compete with the most accurate models because it makes highly accurate predictions. Therefore, we can use the KNN algorithm for applications that require high accuracy but that do not require a human-readable model.

1. When the provided dataset for our task is small.

2. When data is labeled correctly, and the predicted value will be among the given labels. If there is category 1, category 2, and category 3, then the predicted category should be one of them, not any other category.
3. KNN is used to solve regression, classification, or search problems

Advantages

- It is straightforward and easy to implement the algorithm since it requires only two parameters: k-value and distance function.
- A good value of k will make the algorithm robust to noise.
- It learns a nonlinear decision boundary.
- There are almost no assumptions on the given data. The only thing that is assumed is nearby/similar instances belong to the same category.
- It is a non-parametric approach. No model fitting/training is required. The data speaks for itself.
- Since model training is not required, it is easy to update the dataset.

Disadvantages

- Inefficient for large datasets since distance has to be calculated throughout every point, looping every time the algorithm encounters a new data point.
- KNN assumes similar data points are close to each other. Therefore, the model is susceptible to outliers. A few outliers from a particular category can draw the new data towards it even in cases when the new data belongs to a different category.
- It cannot handle imbalanced data. When the data is imbalanced, there is a lot more data belonging to one particular category than the rest of the categories; the algorithm will be biased. Therefore, it needs to be handled explicitly.
- If our dataset requires a K that is a large number, it will increase the computational expense of the algorithm.

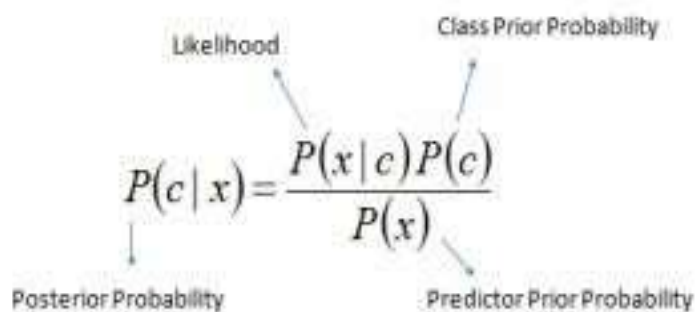
5.2 NAIVE BAYES ALGORITHM

It is a classification technique based on Bayes model 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:



The diagram shows the equation $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ with four labels and arrows pointing to the terms: 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

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

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*.

How Naive Bayes algorithm works?

Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

TABLE NO 1: LIKELIHOOD TABLE

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

$$P(\text{Yes} \mid \text{Sunny}) = P(\text{Sunny} \mid \text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$$

Here we have $P(\text{Sunny} \mid \text{Yes}) = 3/9 = 0.33$, $P(\text{Sunny}) = 5/14 = 0.36$, $P(\text{Yes}) = 9/14 = 0.64$

Now, $P(\text{Yes} \mid \text{Sunny}) = 0.33 * 0.64 / 0.36 = 0.60$, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

ADVANTAGES:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.

- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption)

DISADVANTAGES:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Applications of Naive Bayes Algorithms

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.

- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not.

CHAPTER 6

SYSTEM IMPLEMENTATION

MODULES:

6.1 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.

- **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.

6.2. 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.

6.2 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.

6.4 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.

CHAPTER 7

DIAGRAMATIC REPRESENTATIONS

7.1 ENTITY-RELATIONSHIP (ER) DIAGRAM

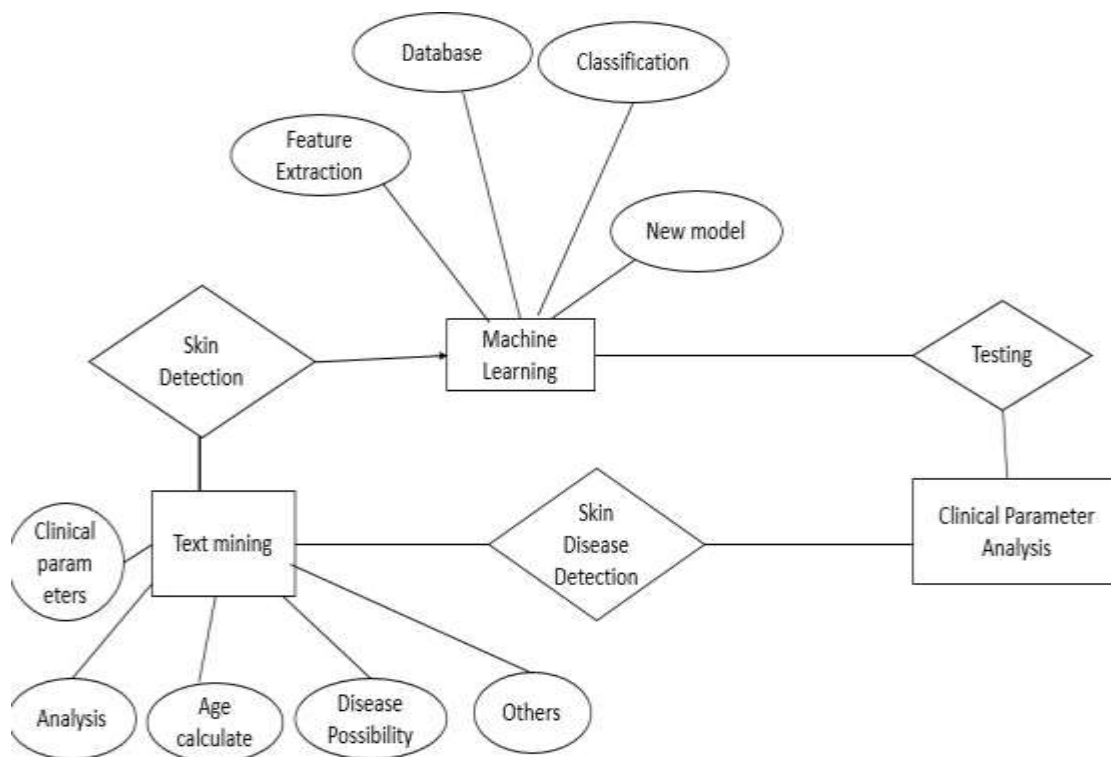


Fig no 9: ER DIAGRAM

7.2 UNIFIED MODELING LANGUAGE (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

7.3 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 development you will find that use cases provide significantly more value because they describe “the meant” of the actual requirements. A use case describes a sequence of action that provides something of measurable value to an action and is drawn as a horizontal ellipse.



Fig no 10: Use case diagram

7.4 CLASS DIAGRAM

A class diagram in the unified modeling language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

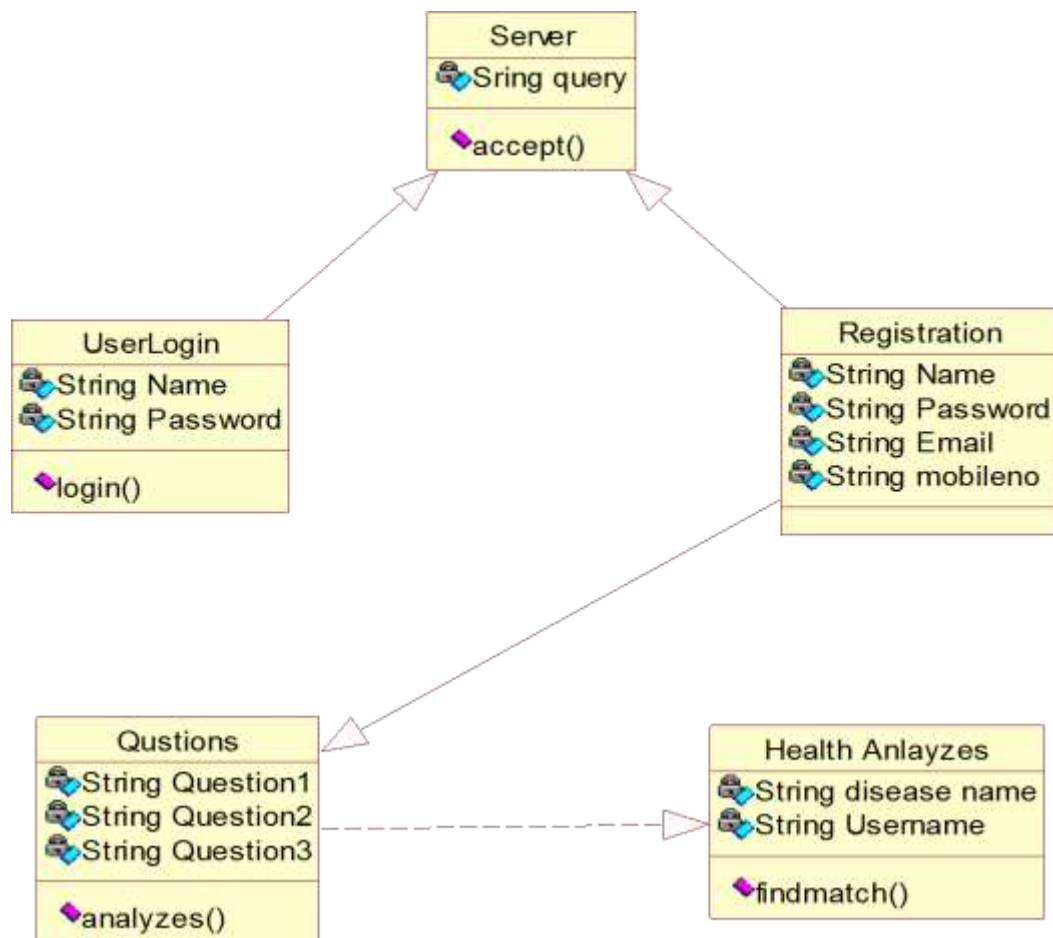


Fig no 11: Class diagram

7.5 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.

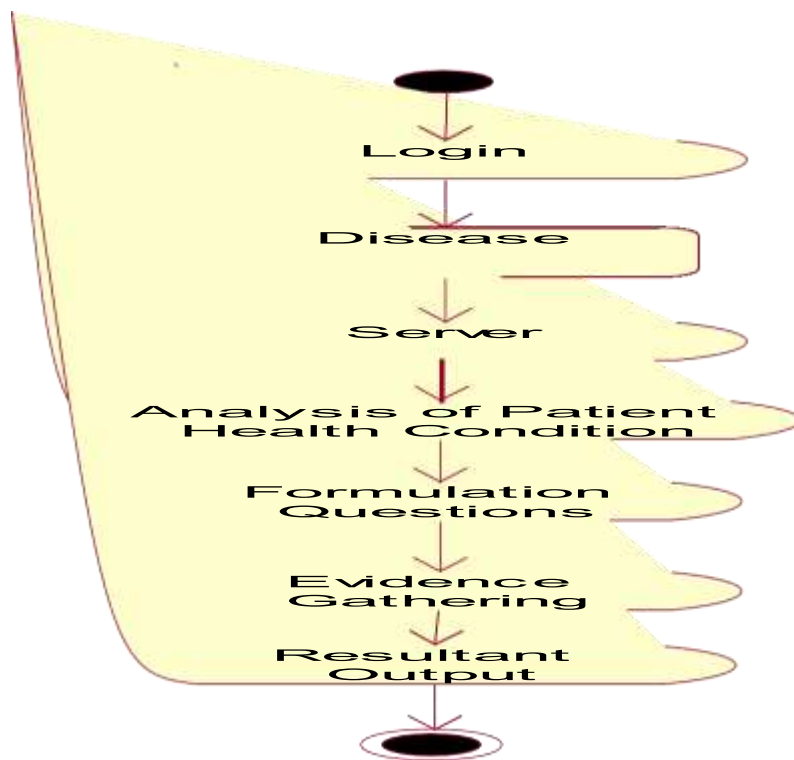


Fig no 12: Activity diagram

7.6 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.

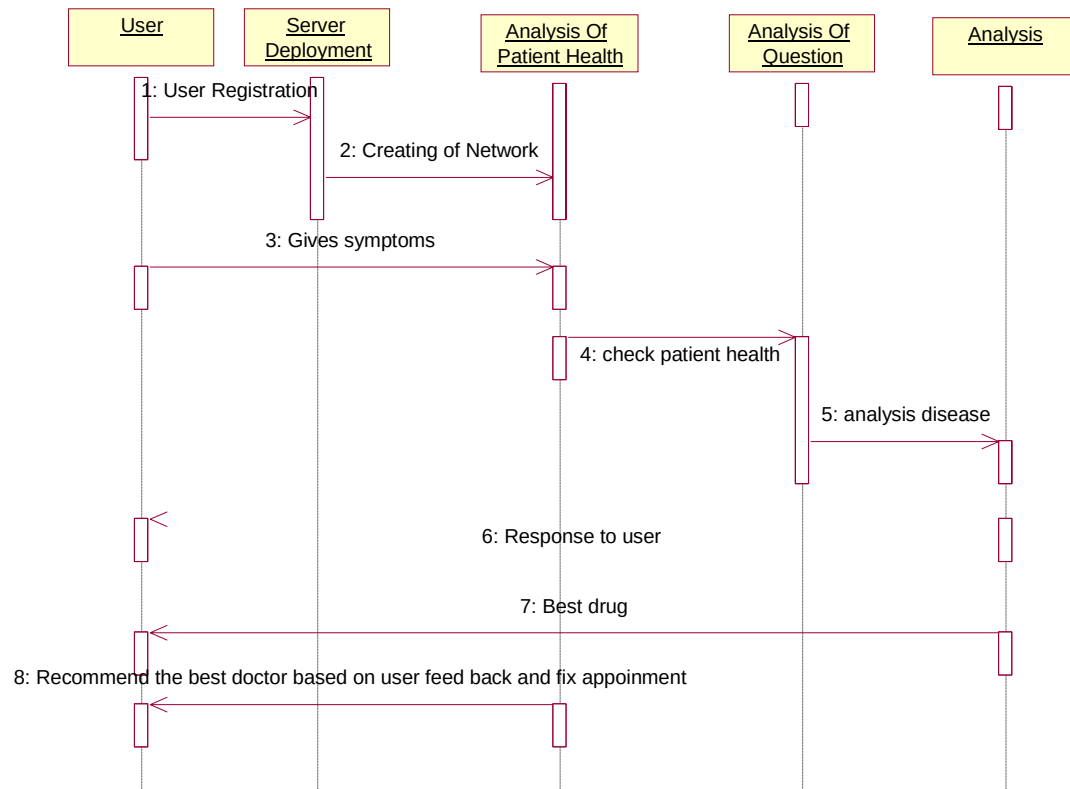


Fig no 13: Sequence diagram

7.7 COLLABORATION DIAGRAM

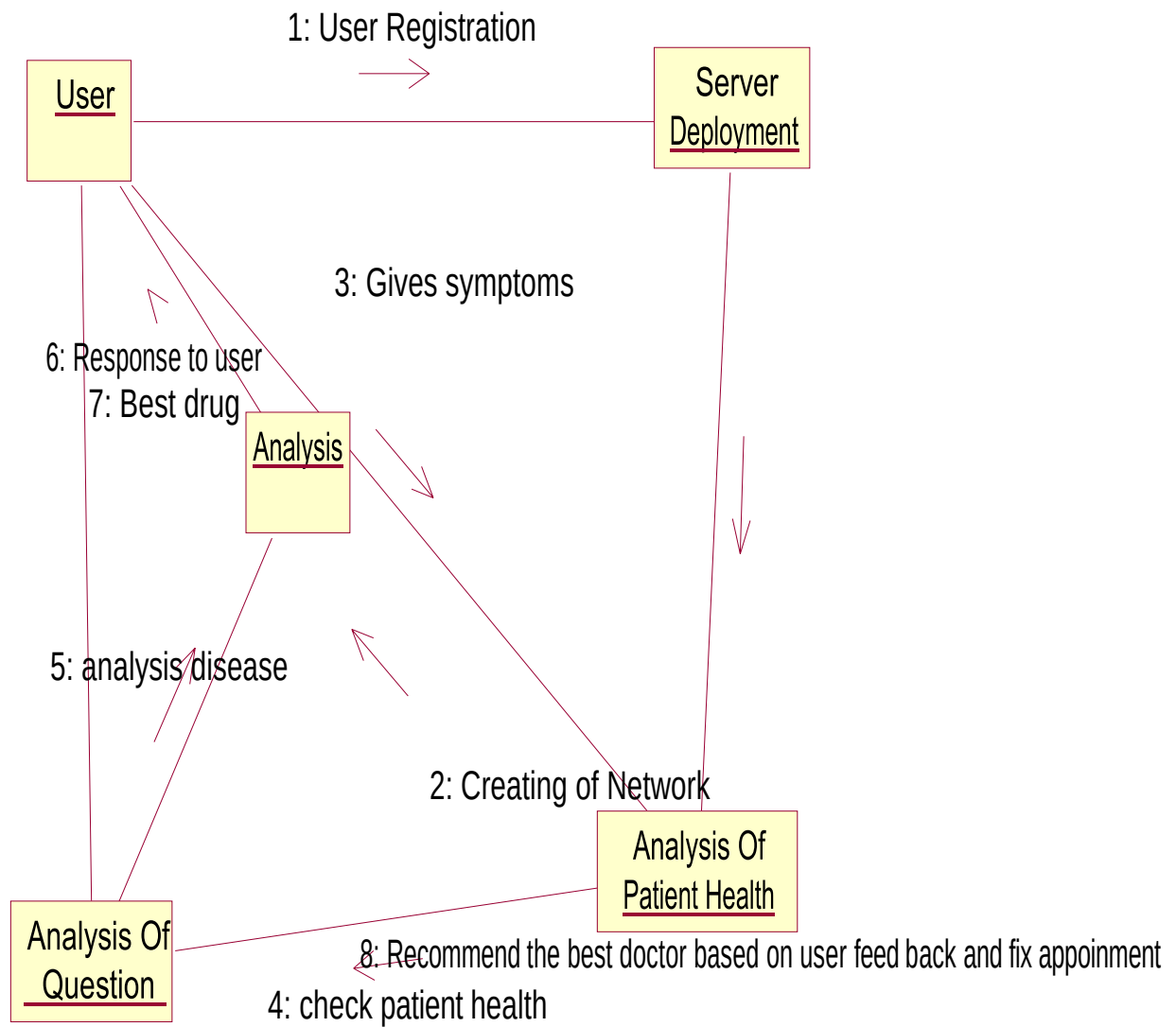


Fig no 14: collaboration diagram

CHAPTER 8

RESULT

This study highlights the importance of predicting Dermatological diseases using segmentation as depicted in the below figures.

SCREENSHOTS

- **LOGIN**

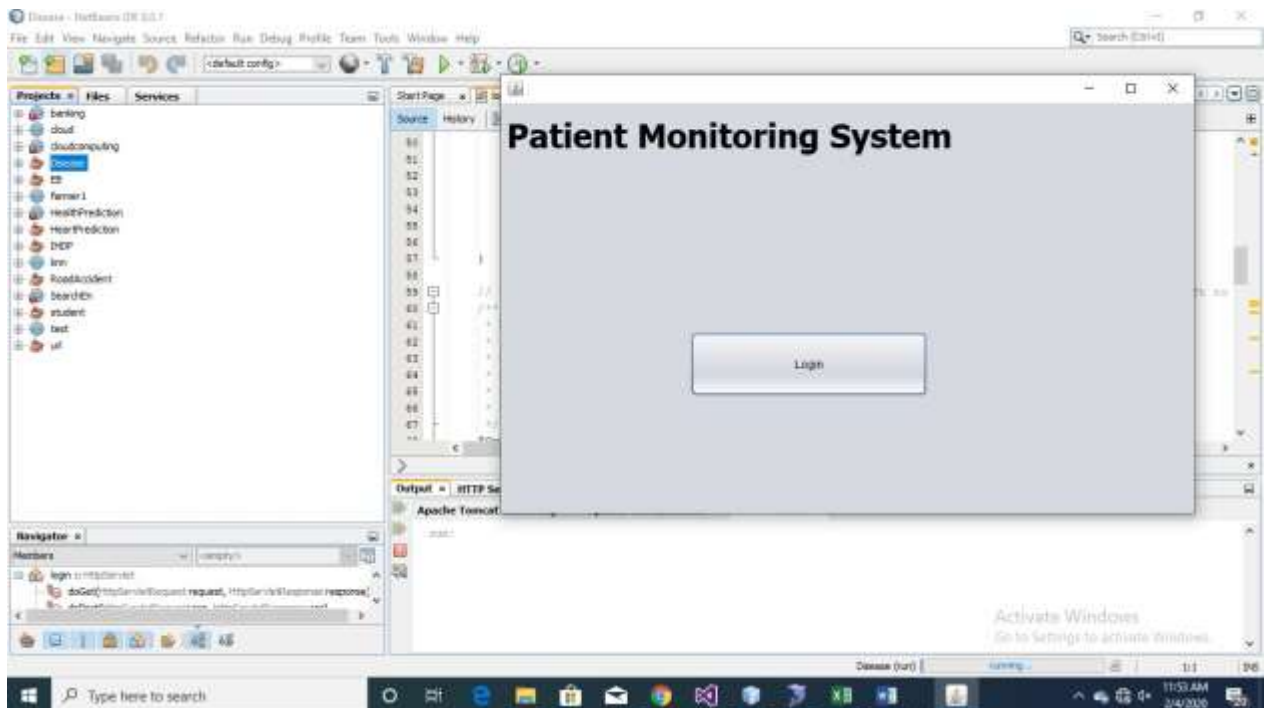


Fig no 15:login

- **VALID USER**

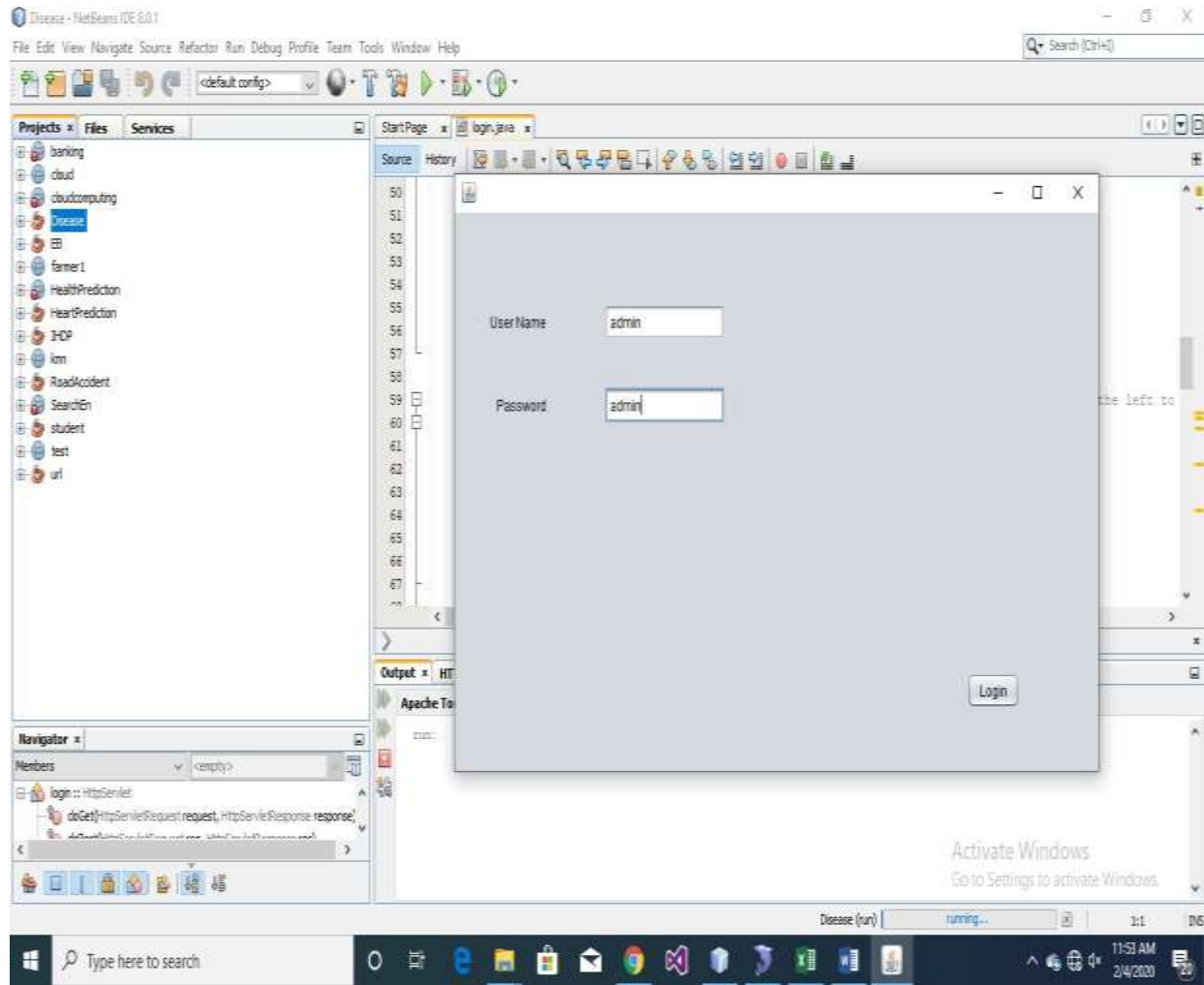


Fig no 16: Valid user

- **INVALID USER**

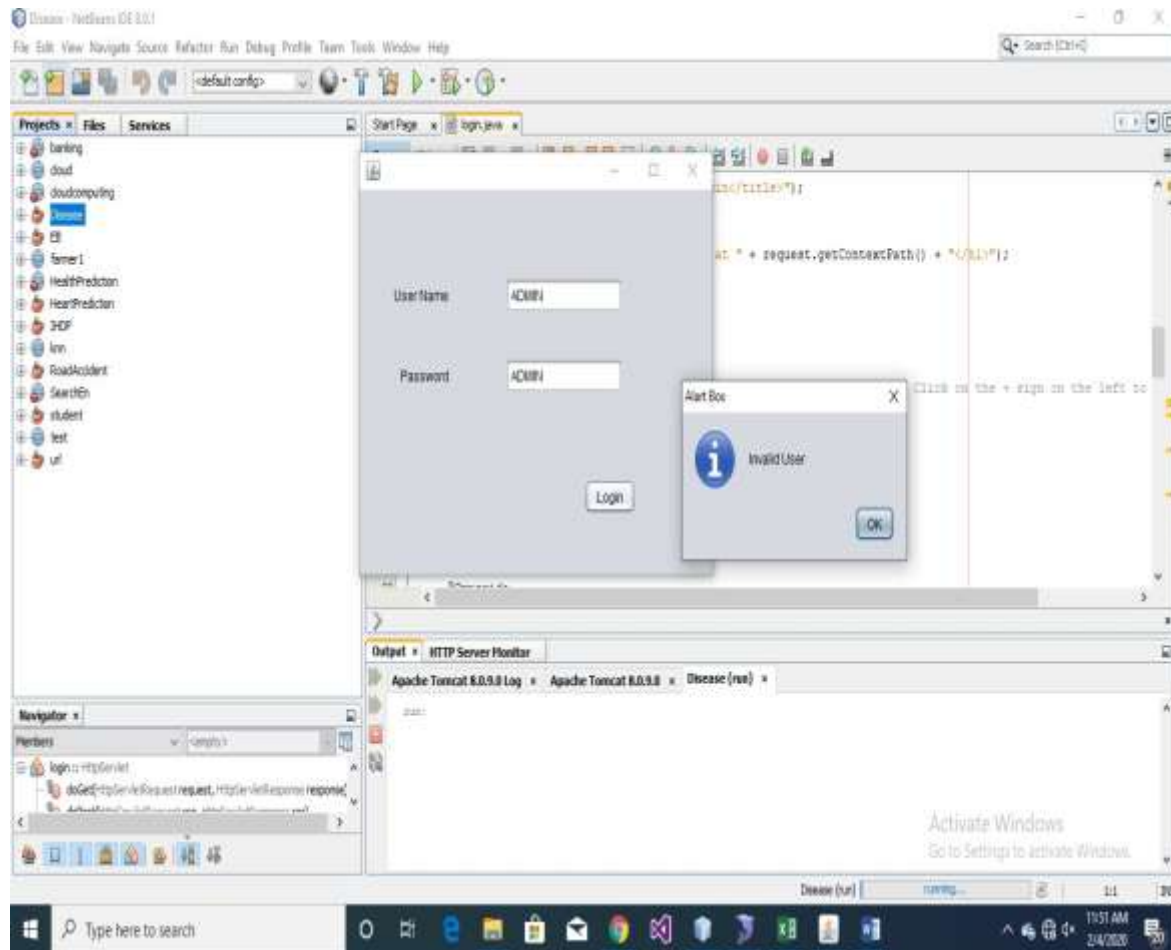


Fig no 17: InValid user

- **PREDICTION**

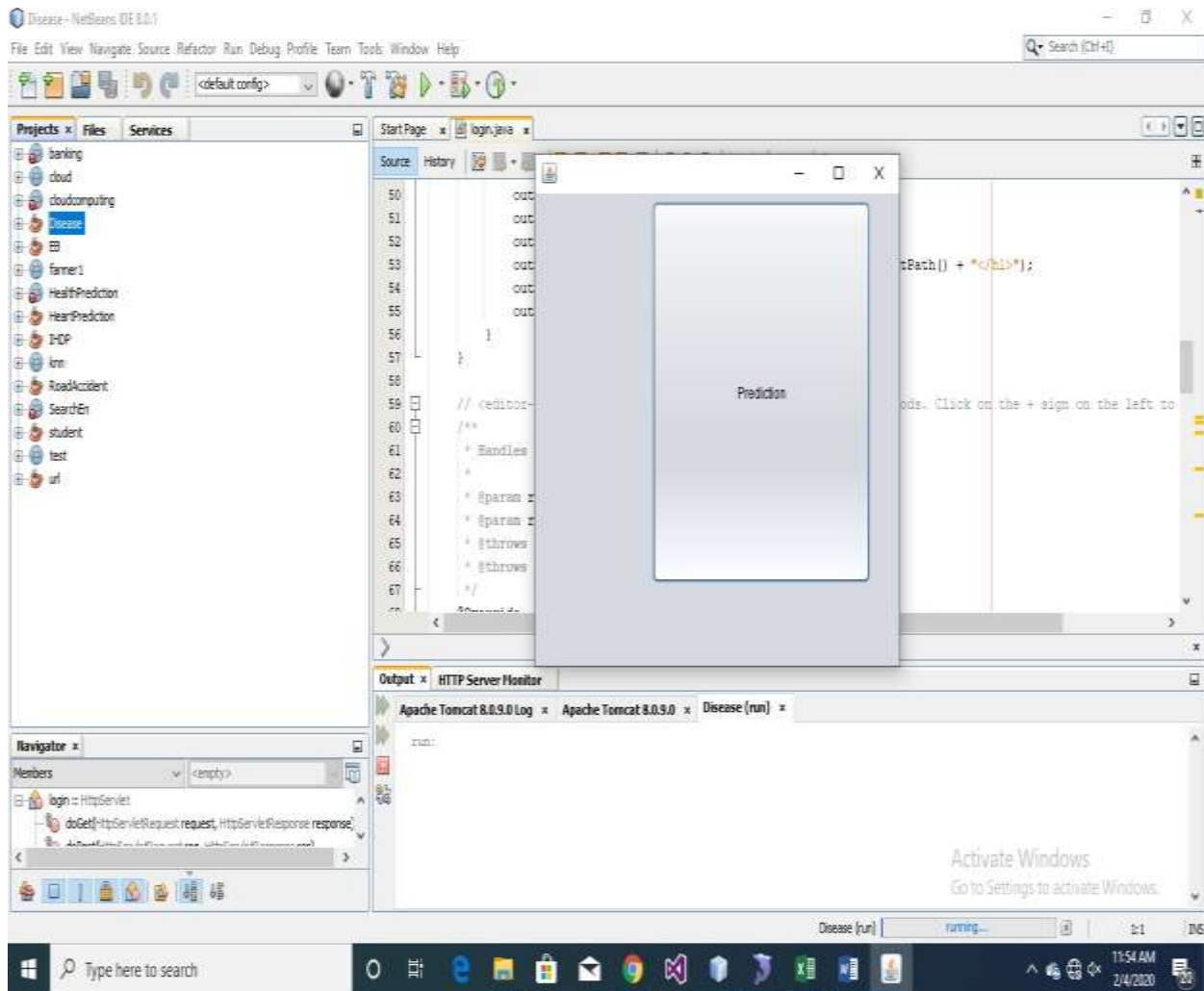


Fig no 18: Prediction

- **INPUT**

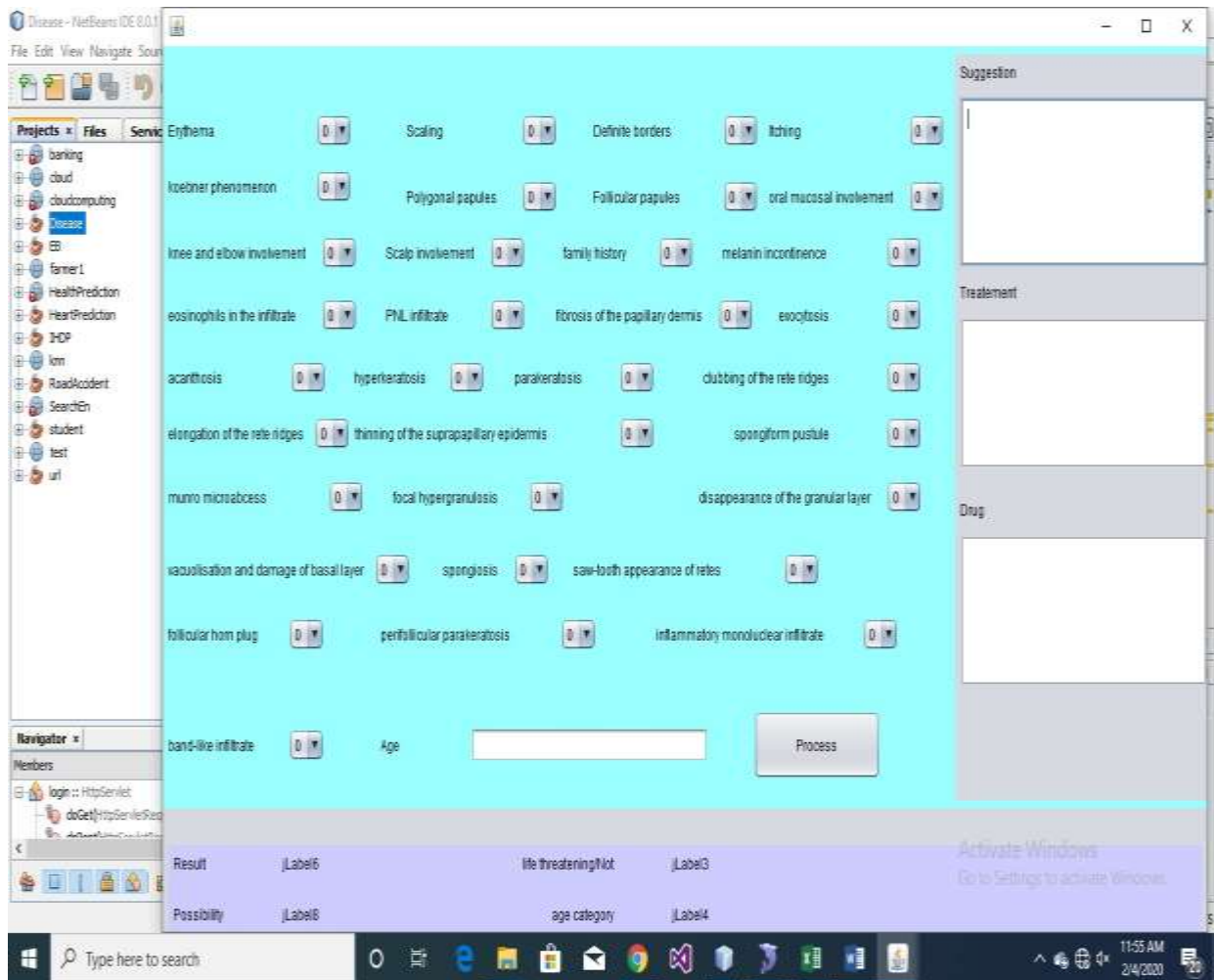


Fig no 19: Input

- **PREDICTION THE SKIN DISEASE**

The screenshot displays a NetBeans IDE window titled "Disease - NetBeans IDE 8.0.1". The main workspace shows a web application for predicting skin diseases. The application features a grid of input fields for various skin symptoms, each with a dropdown menu for selection. The symptoms listed are:

- Erythema (0)
- Scaling (0)
- Definite borders (3)
- Itching (0)
- Koebner phenomenon (0)
- Polygonal papules (0)
- Follicular papules (0)
- Oral mucosal involvement (0)
- Knee and elbow involvement (0)
- Scalp involvement (2)
- Family history (0)
- Melanin incontinence (0)
- Eosinophils in the infiltrate (0)
- PMN infiltrate (0)
- Fibrosis of the papillary dermis (0)
- Exocytosis (0)
- Acanthosis (0)
- Hyperkeratosis (3)
- Parakeratosis (0)
- Clubbing of the rete ridges (2)
- Elongation of the rete ridges (0)
- Thinning of the suprapapillary epidermis (1)
- Spongiform pustule (0)
- Munro microabscess (0)
- Focal hypergranulosis (0)
- Disappearance of the granular layer (0)
- Vacuolisation and damage of basal layer (0)
- Spongiosis (0)
- Saw-tooth appearance of rete (2)
- Follicular horn plug (0)
- Perifollicular parakeratosis (0)
- Inflammatory mononuclear infiltrate (0)
- Band-like infiltrate (0)

An "Age" input field is set to 21. A "Process" button is located at the bottom right of the input area. On the right side of the application, there are three panels:

- Suggestion:** A text area containing the suggestion "Moisturizer ;dates and protects skin from damage".
- Treatment:** A text area containing the suggestion "amines and in rare cases antiviral drugs can help".
- Drug:** A text area containing the suggestion "Steroid and Antihistamine drug".

At the bottom of the application, a purple status bar displays the following information:

- Result:** pityriasis rosea
- life threatening/Not:** No
- Possibility:** 82 %
- age category:** 10 to 35

The Windows taskbar at the bottom shows the time as 11:37 AM on 2/4/2020.

Fig no 20: Predicting the skin disease

- **RESULT**

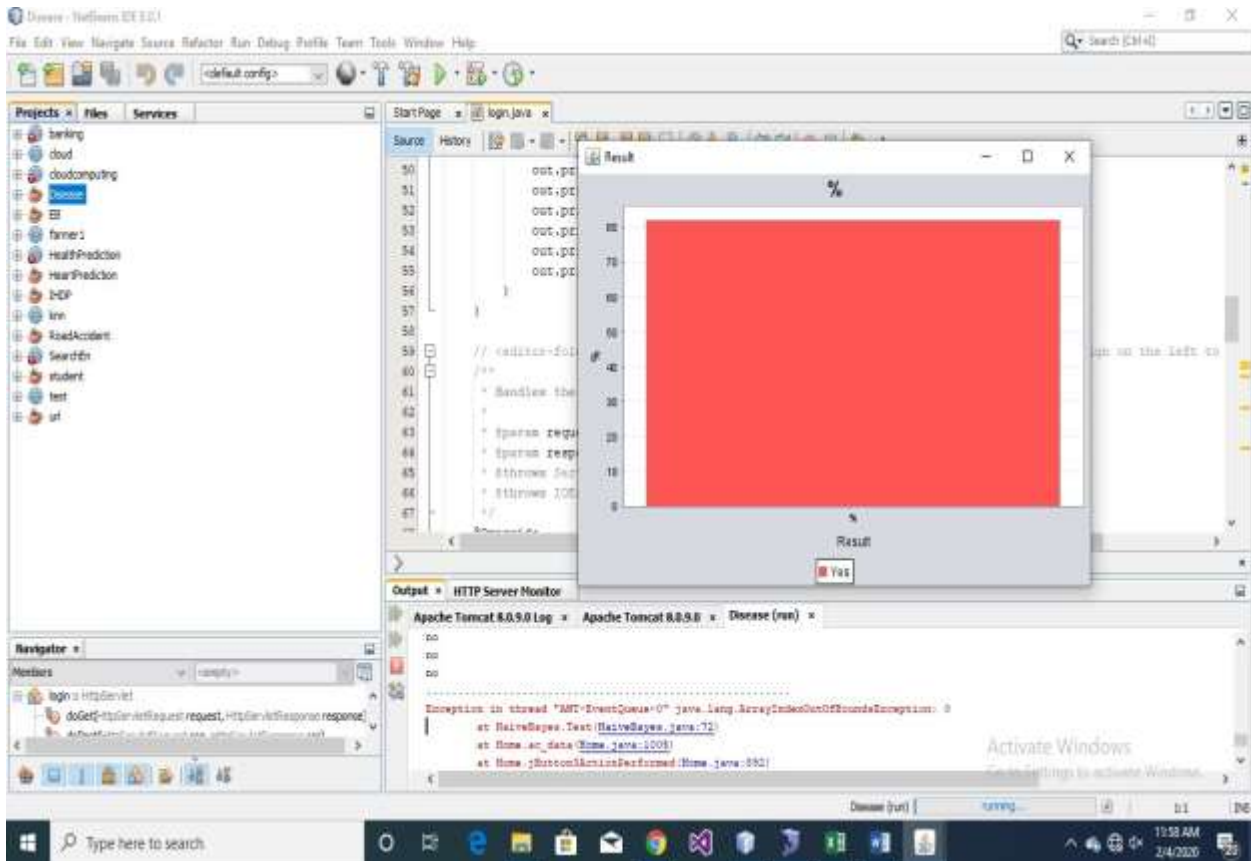


Fig no 21: Result

9. 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.

10. 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.

APPENDIX

A. SAMPLE CODING

1. InternalFrameDemo.java

```
import java.awt.Dimension;

import javax.swing.JDesktopPane;
import javax.swing.JInternalFrame;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.data.time.Minute;
import org.jfree.data.time.RegularTimePeriod;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
import org.jfree.data.xy.XYDataset;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.RefineryUtilities;

/**
 * A simple internal frame demo.
 */
public class InternalFrameDemo extends ApplicationFrame {

    /**
     * Creates a new instance of the demo.
     *
     * @param title the title.
     */
    public InternalFrameDemo(final String title) {
        super(title);
        final JDesktopPane desktopPane = new JDesktopPane();
        desktopPane.setPreferredSize(new Dimension(600, 400));
        // final JInternalFrame frame1 = createFrame1();
        // desktopPane.add(frame1);
        // frame1.pack();
        // frame1.setVisible(true);
        final JInternalFrame frame2 = createFrame2();
    }
}
```

```

        desktopPane.add(frame2);
        frame2.pack();
        frame2.setLocation(100, 200);
        frame2.setVisible(true);
        getContentPane().add(desktopPane);
    }

    /**
     * Creates an internal frame.
     *
     * @return An internal frame.
     */
    private JInternalFrame createFrame1() {
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(34.0, "Series 1", "Category 1");
        dataset.addValue(23.0, "Series 1", "Category 2");
        dataset.addValue(54.0, "Series 1", "Category 3");
        final JFreeChart chart = ChartFactory.createBarChart(
            "Bar Chart",
            "Category",
            "Series",
            dataset,
            PlotOrientation.VERTICAL,
            true,
            true,
            false
        );
        final ChartPanel chartPanel = new ChartPanel(chart);
        chartPanel.setPreferredSize(new Dimension(200, 100));
        final JInternalFrame frame = new JInternalFrame("Frame 1", true);
        frame.getContentPane().add(chartPanel);
        return frame;
    }

    /**
     * Creates an internal frame.
     *
     * @return An internal frame.
     */
    private JInternalFrame createFrame2() {
        final XYDataset dataset1 = createDataset("Series 1", 100.0, new Minute(), 200);

        final JFreeChart chart = ChartFactory.createTimeSeriesChart(
            "Time Series Chart",

```

```

        "Time of Day",
        "Value",
        dataset1,
        true,
        true,
        false
    );
    final ChartPanel chartPanel = new ChartPanel(chart);
    chartPanel.setPreferredSize(new Dimension(200, 100));
    final JInternalFrame frame = new JInternalFrame("Frame 2", true);
    frame.getContentPane().add(chartPanel);
    return frame;
}

/**
 * Creates a sample dataset.
 *
 * @param name the dataset name.
 * @param base the starting value.
 * @param start the starting period.
 * @param count the number of values to generate.
 *
 * @return The dataset.
 */
private XYDataset createDataset(final String name,
                                final double base,
                                final RegularTimePeriod start,
                                final int count) {

    final TimeSeries series = new TimeSeries(name, start.getClass());
    RegularTimePeriod period = start;
    double value = base;
    for (int i = 0; i < count; i++) {
        series.add(period, value);
        period = period.next();
        value = value * (1 + (Math.random() - 0.495) / 10.0);
    }

    final TimeSeriesCollection dataset = new TimeSeriesCollection();
    dataset.addSeries(series);

    return dataset;
}

// *****

```

```

// * JFREECHART DEVELOPER GUIDE *
// * The JFreeChart Developer Guide, written by David Gilbert, is available *
// * to purchase from Object Refinery Limited: *
// * *
// * http://www.object-refinery.com/jfreechart/guide.html *
// * *
// * Sales are used to provide funding for the JFreeChart project - please *
// * support us so that we can continue developing free software. *
// *****

/**
 * The starting point for the demo.
 *
 * @param args ignored.
 */
public static void main(final String[] args) {
    final InternalFrameDemo demo = new InternalFrameDemo("Internal Frame Demo");
    demo.pack();
    RefineryUtilities.centerFrameOnScreen(demo);
    demo.setVisible(true);
}
}

```

2. K.NearestNeighbours.java

```

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;
    }
}

```

3. NaiveBayes.java

```

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;
}
}

```

4. Person.java

```

public class Person {
    private double attributes[];
    private Outcome assigned; //yes or no

    public Person(String line, int mode) {
        String split[] = line.split(",");
        if(mode == 0) { //training data
            attributes = new double[split.length - 1];
            for(int i = 0; i < split.length - 1; i++) {
                attributes[i] = Double.parseDouble(split[i]);
            }
            if(split[split.length-1].equals("1")) {
                assigned = Outcome.yes;
            }
            else {
                assigned = Outcome.no;
            }
        }
        else { //test data
            attributes = new double[split.length];

```

```

        for(int i = 0; i < split.length; i++) {
            attributes[i] = Double.parseDouble(split[i]);
        }
        assigned = Outcome.unknown;
    }
}

public double[] getAttributes() {
    return attributes;
}

public Outcome getOutcome() {
    return assigned;
}
}

```

5. SSMO.java

```

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Map.Entry;
import java.util.Random;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.AbstractMap;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.jfree.ui.RefineryUtilities;

public class SSMO {
    ArrayList<ArrayList<Entry<Double, Double>> > rowList;
    static HashMap<Entry<Integer,Integer> ,Integer> map;
    static double[] lambdas;
    static double C;
    double epsilon;
    double[] b;
    double B;

```

```

        public SSMO(double[] lambdas, HashMap<Entry<Integer,Integer>,Integer> map,
double C) {
            SSMO.map = map;
            SSMO.lambdas = lambdas;
            SSMO.C = C;
        }

SSMO() {
    //To change body of generated methods, choose Tools | Templates.
}

    public ArrayList<ArrayList<Entry<Integer, Integer>>> init() {
        map = new LinkedHashMap<Entry<Integer,Integer>,Integer>();
        lambdas= new double[6];
        for(int i = 0; i < lambdas.length; i++) {
            lambdas[i] = 0;
        }
        b = new double[lambdas.length];
        SSMO.C = 2.5;
        this.epsilon = 0.00001;
        Entry<Integer,Integer> x1 = new
AbstractMap.SimpleEntry<Integer,Integer>(3,3);
        Entry<Integer,Integer> x2 = new
AbstractMap.SimpleEntry<Integer,Integer>(3,4);
        Entry<Integer,Integer> x3 = new
AbstractMap.SimpleEntry<Integer,Integer>(2,3);
        Entry<Integer,Integer> x4 = new
AbstractMap.SimpleEntry<Integer,Integer>(1,1);
        Entry<Integer,Integer> x5 = new
AbstractMap.SimpleEntry<Integer,Integer>(1,3);
        Entry<Integer,Integer> x6 = new
AbstractMap.SimpleEntry<Integer,Integer>(2,2);
        map.put(x1, 1);
        map.put(x2, 1);
        map.put(x3, 1);
        map.put(x4, -1);
        map.put(x5, -1);
        map.put(x6, -1);
        ArrayList<ArrayList<Entry<Integer,Integer>>> ijList = new
ArrayList<ArrayList<Entry<Integer,Integer>>>();
        ArrayList<Entry<Integer,Integer>> ijrow1 = new
ArrayList<Entry<Integer,Integer>>();
        ArrayList<Entry<Integer,Integer>> ijrow2 = new
ArrayList<Entry<Integer,Integer>>();
        ArrayList<Entry<Integer,Integer>> ijrow3 = new
ArrayList<Entry<Integer,Integer>>();

```

```

        ArrayList<Entry<Integer,Integer>> ijrow4 = new
ArrayList<Entry<Integer,Integer>>();
        ArrayList<Entry<Integer,Integer>> ijrow5 = new
ArrayList<Entry<Integer,Integer>>();
        ArrayList<Entry<Integer,Integer>> ijrow6 = new
ArrayList<Entry<Integer,Integer>>();
        ArrayList<Entry<Integer,Integer>> ijrow7 = new
ArrayList<Entry<Integer,Integer>>();
        ArrayList<Entry<Integer,Integer>> ijrow8 = new
ArrayList<Entry<Integer,Integer>>();
        ArrayList<Entry<Integer,Integer>> ijrow9 = new
ArrayList<Entry<Integer,Integer>>();
        ArrayList<Entry<Integer,Integer>> ijrow10 = new
ArrayList<Entry<Integer,Integer>>();
        Entry<Integer,Integer> ijrow1entry1 = new
AbstractMap.SimpleEntry<Integer,Integer>(0,1);
        Entry<Integer,Integer> ijrow1entry2 = new
AbstractMap.SimpleEntry<Integer,Integer>(1,2);
        Entry<Integer,Integer> ijrow1entry3 = new
AbstractMap.SimpleEntry<Integer,Integer>(2,3);
        Entry<Integer,Integer> ijrow1entry4 = new
AbstractMap.SimpleEntry<Integer,Integer>(3,4);
        Entry<Integer,Integer> ijrow1entry5 = new
AbstractMap.SimpleEntry<Integer,Integer>(4,5);
        ijrow1.add(ijrow1entry1);
        ijrow1.add(ijrow1entry2);
        ijrow1.add(ijrow1entry3);
        ijrow1.add(ijrow1entry4);
        ijrow1.add(ijrow1entry5);
        Entry<Integer,Integer> ijrow2entry1 = new
AbstractMap.SimpleEntry<Integer,Integer>(0,2);
        Entry<Integer,Integer> ijrow2entry2 = new
AbstractMap.SimpleEntry<Integer,Integer>(1,3);
        Entry<Integer,Integer> ijrow2entry3 = new
AbstractMap.SimpleEntry<Integer,Integer>(2,4);
        Entry<Integer,Integer> ijrow2entry4 = new
AbstractMap.SimpleEntry<Integer,Integer>(3,5);
        ijrow2.add(ijrow2entry1);
        ijrow2.add(ijrow2entry2);
        ijrow2.add(ijrow2entry3);
        ijrow2.add(ijrow2entry4);
        Entry<Integer,Integer> ijrow3entry1 = new
AbstractMap.SimpleEntry<Integer,Integer>(0,3);
        Entry<Integer,Integer> ijrow3entry2 = new
AbstractMap.SimpleEntry<Integer,Integer>(1,4);
        Entry<Integer,Integer> ijrow3entry3 = new

```

```

AbstractMap.SimpleEntry<Integer,Integer>(2,5);
    ijrow3.add(ijrow3entry1);
    ijrow3.add(ijrow3entry2);
    ijrow3.add(ijrow3entry3);
    Entry<Integer,Integer> ijrow4entry1 = new
AbstractMap.SimpleEntry<Integer,Integer>(0,4);
    Entry<Integer,Integer> ijrow4entry2 = new
AbstractMap.SimpleEntry<Integer,Integer>(1,5);
    ijrow4.add(ijrow4entry1);
    ijrow4.add(ijrow4entry2);
    Entry<Integer,Integer> ijrow5entry1 = new
AbstractMap.SimpleEntry<Integer,Integer>(0,5);
    ijrow5.add(ijrow5entry1);
    Entry<Integer,Integer> ijrow6entry1 = new
AbstractMap.SimpleEntry<Integer,Integer>(1,0);
    Entry<Integer,Integer> ijrow6entry2 = new
AbstractMap.SimpleEntry<Integer,Integer>(2,1);
    Entry<Integer,Integer> ijrow6entry3 = new
AbstractMap.SimpleEntry<Integer,Integer>(3,2);
    Entry<Integer,Integer> ijrow6entry4 = new
AbstractMap.SimpleEntry<Integer,Integer>(4,3);
    Entry<Integer,Integer> ijrow6entry5 = new
AbstractMap.SimpleEntry<Integer,Integer>(5,4);
    ijrow6.add(ijrow6entry1);
    ijrow6.add(ijrow6entry2);
    ijrow6.add(ijrow6entry3);
    ijrow6.add(ijrow6entry4);
    ijrow6.add(ijrow6entry5);
    Entry<Integer,Integer> ijrow7entry1 = new
AbstractMap.SimpleEntry<Integer,Integer>(2,0);
    Entry<Integer,Integer> ijrow7entry2 = new
AbstractMap.SimpleEntry<Integer,Integer>(3,1);
    Entry<Integer,Integer> ijrow7entry3 = new
AbstractMap.SimpleEntry<Integer,Integer>(4,2);
    Entry<Integer,Integer> ijrow7entry4 = new
AbstractMap.SimpleEntry<Integer,Integer>(5,3);
    ijrow7.add(ijrow7entry1);
    ijrow7.add(ijrow7entry2);
    ijrow7.add(ijrow7entry3);
    ijrow7.add(ijrow7entry4);
    Entry<Integer,Integer> ijrow8entry1 = new
AbstractMap.SimpleEntry<Integer,Integer>(3,0);
    Entry<Integer,Integer> ijrow8entry2 = new
AbstractMap.SimpleEntry<Integer,Integer>(4,1);
    Entry<Integer,Integer> ijrow8entry3 = new
AbstractMap.SimpleEntry<Integer,Integer>(5,2);

```

```

        ijrow8.add(ijrow8entry1);
        ijrow8.add(ijrow8entry2);
        ijrow8.add(ijrow8entry3);
        Entry<Integer,Integer> ijrow9entry1 = new
AbstractMap.SimpleEntry<Integer,Integer>(4,0);
        Entry<Integer,Integer> ijrow9entry2 = new
AbstractMap.SimpleEntry<Integer,Integer>(5,1);
        ijrow9.add(ijrow9entry1);
        ijrow9.add(ijrow9entry2);
        Entry<Integer,Integer> ijrow10entry1 = new
AbstractMap.SimpleEntry<Integer,Integer>(5,0);
        ijrow10.add(ijrow10entry1);
        ijList.add(ijrow1);
        ijList.add(ijrow2);
        ijList.add(ijrow3);
        ijList.add(ijrow4);
        ijList.add(ijrow5);
        ijList.add(ijrow6);
        ijList.add(ijrow7);
        ijList.add(ijrow8);
        ijList.add(ijrow9);
        ijList.add(ijrow10);
        return ijList;
    }

    public ArrayList<Entry<Integer, Integer>> init2() {
        map = new LinkedHashMap<Entry<Integer,Integer>,Integer>();
        lambdas= new double[6];
        for(int i = 0; i < lambdas.length; i++) {
            lambdas[i] = 0;
        }
        b = new double[lambdas.length];
        SSMO.C = 2.5;
        this.epsilon = 0.00001;
        Entry<Integer,Integer> x1 = new
AbstractMap.SimpleEntry<Integer,Integer>(3,3);
        Entry<Integer,Integer> x2 = new
AbstractMap.SimpleEntry<Integer,Integer>(3,4);
        Entry<Integer,Integer> x3 = new
AbstractMap.SimpleEntry<Integer,Integer>(2,3);
        Entry<Integer,Integer> x4 = new
AbstractMap.SimpleEntry<Integer,Integer>(1,1);
        Entry<Integer,Integer> x5 = new
AbstractMap.SimpleEntry<Integer,Integer>(1,3);
        Entry<Integer,Integer> x6 = new
AbstractMap.SimpleEntry<Integer,Integer>(2,2);

```

```

        map.put(x1, 1);
        map.put(x2, 1);
        map.put(x3, 1);
        map.put(x4, -1);
        map.put(x5, -1);
        map.put(x6, -1);
        ArrayList<Entry<Integer,Integer>> ijList2 = new ArrayList<>();
        ArrayList<Integer> shuffleList = new ArrayList<>();
        while(ijList2.size() < 1000) {
            for(int i = 0 ; i < lambdas.length ; i++) {
                shuffleList.clear();
                for(int j = 0 ; j < lambdas.length ; j++) {
                    shuffleList.add(j);
                }
                System.out.println("New Shuffle List: " + shuffleList);
                Random rand = new Random();
                if(shuffleList.size() == 6) {
                    shuffleList.remove(i);
                    Collections.shuffle(shuffleList, rand);
                    System.out.println("After shuffle:" + shuffleList);
                    Entry<Integer,Integer> entry = new
AbstractMap.SimpleEntry<Integer, Integer>(i,shuffleList.get(0));
                    ijList2.add(entry);
                    System.out.println(entry);
                }
            }
        }
        ijList2.remove(1001);
        ijList2.remove(1000);
        System.out.println(ijList2.size());
        return ijList2;
    }

    public SSMOReturn train2(ArrayList<Entry<Integer,Integer>> ijList2) {
        B = 0.0;
        System.out.println("epsilon: " + epsilon);
        double[] oldLambdas2 = new double[lambdas.length];
        double[] oldLambdas = new double[lambdas.length];
        while(true) {
            for(Entry<Integer,Integer> entry : ijList2) {
                System.out.println("-----WE ARE TRAINING IN TRAIN2--
-----");

                int i = entry.getKey();
                int j = entry.getValue();
                Entry<Integer,Integer> Xi = (Entry<Integer, Integer>

```

```

(map.keySet().toArray())[i];
Entry<Integer,Integer> Xj = (Entry<Integer,Integer>)
(map.keySet().toArray())[j];
int D = 2 * dotProduct(Xi,Xj) - dotProduct(Xi,Xi) - dotProduct(Xj,Xj);
System.out.println("D: " + D);
if( Math.abs(D) > epsilon) {
    // zi is map.get(Xi)
    // zj is map.get(Xj)
    double zi = map.get(Xi).doubleValue();
    double zj = map.get(Xj).doubleValue();
    double Ei = f(Xi) - zi;
    double Ej = f(Xj) - zj;
    oldLambdas[i] = lambdas[i];
    oldLambdas[j] = lambdas[j];
    lambdas[j] = lambdas[j] - zj * (Ei - Ej) / D;
    double h;
    double l;
    if( map.get(Xi) == map.get(Xj) ) {
        ArrayList<Double> listL = new ArrayList<>();
        listL.add(0.0);
        double elementL = lambdas[i] + lambdas[j] - C;
        listL.add(elementL);
        l = Collections.max(listL);
        ArrayList<Double> listH = new ArrayList<>();
        listH.add(C);
        double elementH = lambdas[i] + lambdas[j];
        listH.add(elementH);
        h = Collections.min(listH);
    } else {
        ArrayList<Double> listL2 = new ArrayList<>();
        listL2.add(0.0);
        double elementL2 = lambdas[j] - lambdas[i];
        listL2.add(elementL2);
        l = Collections.max(listL2);
        ArrayList<Double> listH2 = new ArrayList<>();
        listH2.add(C);
        double elementH2 = C + lambdas[j] - lambdas[i];
        listH2.add(elementH2);
        h = Collections.min(listH2);
    }
    if(lambdas[j] > h) {
        lambdas[j] = h;
    } else if(lambdas[j] >= l && lambdas[j] <= h) {
        lambdas[j] = lambdas[j];
    } else {
        lambdas[j] = l;
    }
}

```

```

        }
        lambdas[i] += zi*zj*(oldLambdas[j] - lambdas[j]);
        b[i] = B - Ei - zi * (lambdas[i] - oldLambdas[i]) *
dotProduct(Xi,Xi) - zj * (lambdas[j] - oldLambdas[j]) * dotProduct(Xi, Xj);
        b[j] = B - Ej - zi * (lambdas[i] - oldLambdas[i]) *
dotProduct(Xi,Xj) - zj * (lambdas[j] - oldLambdas[j]) * dotProduct(Xj, Xj);
        System.out.println("b values calculated ..... ");
        printArrayDouble(b);
        if(lambdas[i]<C && lambdas[i]>0) {
            B = b[i];
        } else if (lambdas[j]<C && lambdas[j]>0) {
            B = b[j];
        } else {
            B = ( b[i] + b[j] ) / 2;
        }
        System.out.println("B value at the end of this round : " + B);
        System.out.println("lambdas after this round");
        printArrayDouble(lambdas);
        System.out.println("oldLambdas after this round");
        printArrayDouble(oldLambdas);
    }
    boolean match = Arrays.equals(lambdas, oldLambdas);
    System.out.println("match value: " +match);
    if(match) {
        break;
    }
}
SSMOReturn sr = new SSMOReturn(lambdas, B);
return sr;
}
}

```

```

public SSMOReturn train(ArrayList<ArrayList<Entry<Integer, Integer>>> ijList) {
    B = 0.0;
    System.out.println("epsilon: " + epsilon);
    double oldLambdas[] = new double[lambdas.length];
    while(true) {
        for(ArrayList<Entry<Integer,Integer>> list : ijList) {
            for(Entry<Integer,Integer> entry: list) {
                System.out.println("WE ARE TRAINING");
                int i = entry.getKey();
                int j = entry.getValue();
                Entry<Integer,Integer> Xi = (Entry<Integer, Integer>
(map.keySet().toArray())[i];
                Entry<Integer,Integer> Xj = (Entry<Integer,Integer>

```

```
(map.keySet().toArray())[j];
```

```
dotProduct(Xj,Xj);
```

```
int D = 2 * dotProduct(Xi,Xj) - dotProduct(Xi,Xi) -
```

```
System.out.println("D: " + D);
```

```
if( Math.abs(D) > epsilon) {
```

```
    double zi = map.get(Xi).doubleValue();
```

```
    double zj = map.get(Xj).doubleValue();
```

```
    double Ei = f(Xi) - zi;
```

```
    double Ej = f(Xj) - zj;
```

```
    oldLambdas[i] = lambdas[i];
```

```
    oldLambdas[j] = lambdas[j];
```

```
    // zi is map.get(Xi)
```

```
    // zj is map.get(Xj)
```

```
    lambdas[j] = lambdas[j] - zj * (Ei - Ej) / D;
```

```
    double h;
```

```
    double l;
```

```
    if( map.get(Xi) == map.get(Xj) ) {
```

```
        ArrayList<Double> listL = new ArrayList<>();
```

```
        listL.add(0.0);
```

```
        double elementL = lambdas[i] + lambdas[j] - C;
```

```
        listL.add(elementL);
```

```
        l = Collections.max(listL);
```

```
        ArrayList<Double> listH = new ArrayList<>();
```

```
        listH.add(C);
```

```
        double elementH = lambdas[i] + lambdas[j];
```

```
        listH.add(elementH);
```

```
        h = Collections.min(listH);
```

```
    } else {
```

```
        ArrayList<Double> listL2 = new ArrayList<>();
```

```
        listL2.add(0.0);
```

```
        double elementL2 = lambdas[j] - lambdas[i];
```

```
        listL2.add(elementL2);
```

```
        l = Collections.max(listL2);
```

```
        ArrayList<Double> listH2 = new ArrayList<>();
```

```
        listH2.add(C);
```

```
        double elementH2 = C + lambdas[j] -
```

```
lambdas[i];
```

```
        listH2.add(elementH2);
```

```
        h = Collections.min(listH2);
```

```
    }
```

```
    if(lambdas[j] > h) {
```

```
        lambdas[j] = h;
```

```
    } else if(lambdas[j] >= l && lambdas[j] <= h) {
```

```
        lambdas[j] = lambdas[j];
```

```
    } else {
```

```
        lambdas[j] = l;
```

```

        }
        lambdas[i] += zi*zj*(oldLambdas[j] - lambdas[j]);
        b[i] = B - Ei - zi * (lambdas[i] - oldLambdas[i]) *
dotProduct(Xi,Xi) - zj * (lambdas[j] - oldLambdas[j]) * dotProduct(Xi, Xj);
        b[j] = B - Ej - zi * (lambdas[i] - oldLambdas[i]) *
dotProduct(Xi,Xj) - zj * (lambdas[j] - oldLambdas[j]) * dotProduct(Xj, Xj);
        System.out.println("B values calculated ..... ");
        printArrayDouble(b);
        if(lambdas[i]<C && lambdas[i]>0) {
            B = b[i];
        } else if (lambdas[j]<C && lambdas[j]>0) {
            B = b[j];
        } else {
            B = ( b[i] + b[j] ) / 2;
        }
        System.out.println("lambdas after this round");
        printArrayDouble(lambdas);
        System.out.println("oldLambdas after this round");
        printArrayDouble(oldLambdas);
    }
}

boolean match = Arrays.equals(lambdas, oldLambdas);
if(match) {
    break;
}

}
SSMOReturn sr = new SSMOReturn(lambdas, B);
return sr;
}

public double f(Entry<Integer,Integer> Xvector) {
    double sum = 0;
    for(int i=0; i< lambdas.length; i++) {
        Entry<Integer,Integer> Xi = (Entry<Integer,Integer>)
(map.keySet().toArray())[i];
        // System.out.println("Input to f(X): " +Xi);
        double zi = map.get(Xi);
        // System.out.println("Z values in f(X): " + zi);
        // System.out.println("B value: " + B);
        double element = lambdas[i] * zi * dotProduct(Xi, Xvector);
        // System.out.println("element value we are adding to sum in f(X): " +
element);
        sum+=element;
    }
    System.out.println("f(X) value: " + sum);
}

```

```

        return sum + B;
    }

    public static void main(String[] args)
    {
        System.out.println("In Main");
        SSMO ssmo = new SSMO(lambdas,map,C);
        ArrayList<ArrayList<Entry<Integer, Integer>> > ijList = ssmo.init();
        SSMOReturn values = ssmo.train(ijList);

        ArrayList<Entry<Integer,Integer>> ijList2 = ssmo.init2();
        SSMOReturn values2 = ssmo.train2(ijList2);
        System.out.println("b value for book's prescriptions: " + values.b);
        System.out.println("Lambda Values for book's prescriptions: ");
        printArrayDouble(values.lambdas);
        System.out.println("-----");
        System.out.println("b value for random selection of lambdas: " + values2.b);
        System.out.println("Lambda Values for random selection of lambdas:");
        printArrayDouble(values2.lambdas);

    }
    public void svm() throws IOException
    {
        Kmeans2d k1=new Kmeans2d();
        k1.loaddata("train.csv");
        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 = "SVM";
            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 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("Accuracy 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();

System.out.println("SVN 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(".....");

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);
    }
}
}
}
public void startsvm()
{
    System.out.println("In Main");
    SSMO ssmo = new SSMO(lambdas,map,C);
    ArrayList<ArrayList<Entry<Integer, Integer>>> ijList = ssmo.init();
    SSMOReturn values = ssmo.train(ijList);

```

```

        ArrayList<Entry<Integer,Integer>> ijList2 = ssmo.init2();
        SSMOReturn values2 = ssmo.train2(ijList2);
        System.out.println("b value for book's prescriptions: " + values.b);
        System.out.println("Lambda Values for book's prescriptions: ");
        printArrayDouble(values.lambdas);
        System.out.println("-----");
        System.out.println("b value for random selection of lambdas: " + values2.b);
        System.out.println("Lambda Values for random selection of lambdas:");
        printArrayDouble(values2.lambdas);
    }
    public static void printArray(int[] array) {
        for (int i : array) {
            System.out.print(i + ",");
        }
        System.out.println();
    }
    public static void printArrayDouble(double[] array) {
        System.out.print("[ ");
        for (double d : array) {
            System.out.print(d + ",");
        }
        System.out.print("]");
        System.out.println();
    }
    public int dotProduct( Entry<Integer,Integer> entry1, Entry<Integer,Integer> entry2) {
        int a = entry1.getKey();
        int b = entry1.getValue();
        int[] ab = new int[] {a,b};
        int c = entry2.getKey();
        int d = entry2.getValue();
        int[] cd = new int[] {c,d};
        int sum = 0;
        for (int i = 0; i < ab.length; i++) {
            sum += ab[i] * cd[i];
        }
        return sum;
    }
}

```

6. SSMOReturn.java

```

public class SSMOReturn {
    public double[] lambdas;
    public double b;

```

```
public SSMOReturn() {  
}  
public SSMOReturn(double[] l, double b) {  
    this.lambdas = l;  
    this.b = b;  
}  
}
```

