# Diabetic Retinopathy Detection using Image Processing

**A PROJECT REPORT**

***Submitted by***


**GANESHKUMAR M [613520104009]**

**VINOTH KUMAR N [613520104045]**

**RAHUL M [613520104025]**

**PRAVEEN KUMAR S [613520104701]**


***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

COMPUTER SCIENCE AND ENGINEERING



**GOVERNMENT COLLEGE OF ENGINEERING, DHARMAPURI.**


**ANNA UNIVERSITY: CHENNAI 600 025**

MAY 2023

# BONAFIDE CERTIFICATE

Certified that this project report "**Diabetic Retinopathy Detection using Image Processing**" is the bonafide work of the following students, **GANESHKUMAR M [613520104009], VINOTH KUMAR N [613520104045], RAHUL M [613520104025], PRAVEEN KUMAR S [613520104701]**, who carried out the project work under my supervision.

**SIGNATURE** **SIGNATURE**

**Dr. J. C. MIRACLIN JOYCE PAMILA**
**HEAD OF THE DEPARTMENT**

Department of Computer Science and Engineering

Government College of Engineering, Dharmapuri.

**Mr. K. M. KIRUPA SHANKAR**
**SUPERVISOR,**
**ASSISTANT PROFESSOR**

Department of Computer Science and Engineering,

Government College of Engineering, Dharmapuri.

Submitted for the university examination held at Government College of Engineering, Dharmapuri on …….

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We are personally indebted to a number of people who gave us their useful insights to aid in our overall progress for this project. A complete acknowledgement would therefore be encyclopedic. First of all, we would like to give our deepest gratitude to our parents for permitting us to take up this course.

We record our sincere thanks to **Dr. V. SUMATHY, M.E., Ph.D., Principal,** for encouraging us to do this project.

We would like to acknowledge **Dr. J. C. MIRACLIN JOYCE PAMILA, M.E., Ph.D., Head of the Department,** for motivating us to pursue this project.

We would like to express our profound appreciation and gratefulness to our **Supervisor Mr. K. M. KIRUPA SHANKAR, M.E., Assistant Professor,** Computer Science and Engineering, for his support and valuable guidance to complete this project successfully.

We would also like to extend our gratitude to **Project Co-ordinator and Class Advisor, Mrs. NARMATHA R, M.E., Assistant Professor,** for her invaluable support and guidance throughout this project. Without her encouragement and expertise, this project would not have been possible.

We also express our sincere thanks to our department faculties for their encouragement to do this project.

## TABLE OF CONTENTS

## ABSTRACT

Diabetic retinopathy (DR) is a common eye disease caused by diabetes, leading to progressive damage of the retina and potentially resulting in vision loss. Early detection and accurate diagnosis of DR play a crucial role in preventing vision impairment and providing timely treatment. Image processing techniques have emerged as effective tools for automated DR detection, aiding healthcare professionals in efficient diagnosis. This paper presents a one-page abstraction of the process involved in Diabetic Retinopathy Detection using Image Processing.

# LIST OF FIGURES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| **AI** | **Artificial Intelligence** |
| **KNN** | **K-Neural Networks** |
| **ML** | **Machine Learning** |
| **MLP** | **Multilayer perceptron** |
| **JSON** | **JavaScript Object Notation** |
| **GPU** | **Graphics Processing Units** |
| **DFD** | **Data Flow Diagram** |
| **POS** | **Part of Speech** |
| **RF** | **Random Forest** |
| **AB** | **Ada Boost** |
| **MNB** | **Multinomial Native Bayes** |
| **HOG** | **Histogram Oriented Gradient** |
| **DOG** | **Difference of Gaussians** |

# CHAPTER 1

# INTRODUCTION

## 1.1. OVERVIEW

Diabetic retinopathy (DR) is a progressive eye disease that occurs as a complication of diabetes. It is one of the leading causes of vision loss and blindness worldwide. DR primarily affects the blood vessels in the retina, the light-sensitive tissue at the back of the eye. Prolonged exposure to high blood sugar levels in individuals with diabetes can damage the small blood vessels in the retina, leading to various abnormalities and impairments. The progression of diabetic retinopathy can be categorized into different stages, including mild non-proliferative diabetic retinopathy (NPDR), moderate NPDR, severe NPDR, and proliferative diabetic retinopathy (PDR). In the early stages, NPDR, small areas of swelling may appear in the retina, known as microaneurysms. As the disease progresses, these microaneurysms can leak blood and fluids, causing further damage to the retina and leading to the formation of new, abnormal blood vessels in the advanced stage of PDR. With the advancements in digital imaging and image processing techniques, automated systems for DR detection have gained significant attention. These systems leverage computer vision algorithms and machine learning techniques to analyze retinal images and identify the presence and severity of diabetic retinopathy. Image processing methods such as preprocessing, image segmentation, feature extraction, and classification are employed to extract meaningful information from retinal images and differentiate normal retinas from those affected by DR.

## 1.2. MACHINE LEARNING

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem.

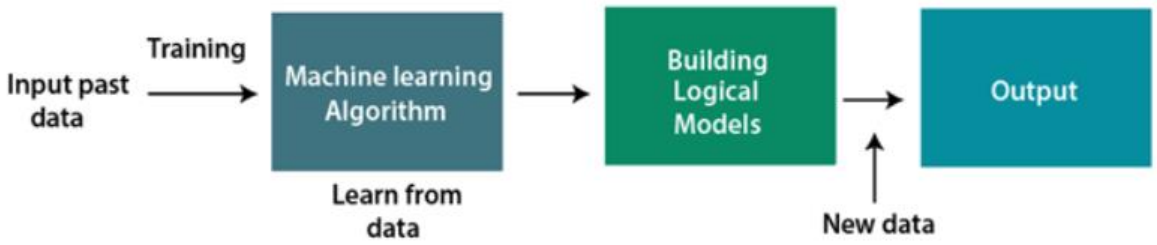



**Fig 1.2.1 Working of machine learning algorithm**

First, machine learning had to get developed. ML is a framework to automate (through algorithms) statistical models, like a linear regression model, to get better at making predictions. A model is a single model that makes predictions about something. Those predictions are made with some accuracy.

The learning portion of creating models spawned the development of artificial neural networks. ANNs utilize the hidden layer as a place to store and evaluate how significant one of the inputs is to the output. The hidden layer stores information regarding the input's importance, and it also makes associations between the importance of combinations of inputs.

### 1.2.1. MACHINE LEARNING PROCESS

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers.

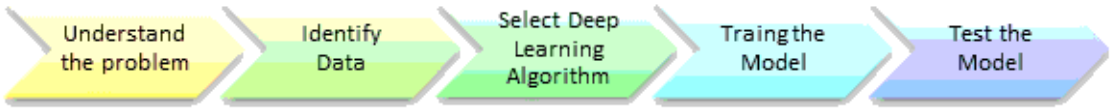



**Fig 1.2.1.1 Deep learning Process**

### 1.2.2. CLASSIFICATION OF NEURAL NETWORKS

- **Shallow neural network**: The Shallow neural network has only one hidden layer between the input and output.
- **Deep neural network**: Deep neural networks have more than one layer. For instance, Google Le Net model for image recognition counts 22 layers.

### 1.2.3. USE CASES OF ARTIFICIAL NEURAL NETWORK

#### 1.2.3.1 Image recognition and computer vision

ANNs can be trained to recognize and classify images, such as identifying objects in photographs, or detecting anomalies in medical images.

#### 1.2.3.2 Natural language processing

ANNs can be used to analyse and understand human language, such as identifying sentiment in social media posts or translating text between languages.

#### 1.2.3.3 Speech recognition and synthesis

ANNs can be used to recognize spoken language and convert it into text, or to synthesize human-like speech.

#### 1.2.3.4 Predictive modelling

ANNs can be used to make predictions or forecasts based on historical data, such as predicting stock prices or weather patterns.

#### 1.2.3.5 Robotics and control systems

ANNs can be used to control robots and other autonomous systems, such as self-driving cars.

## 1.3. OBJECTIVE OF THE PROJECT

This study aims to develop and evaluate an automated system for the detection and classification of diabetic retinopathy (DR) using image processing techniques. To achieve this, a comprehensive investigation and review of existing literature on diabetic retinopathy detection methods and image processing techniques will be conducted. A representative dataset of retinal images, encompassing various stages of diabetic retinopathy, along with associated ground truth annotations, will be collected. The collected images will undergo pre-processing to enhance their quality, remove noise, and normalize image characteristics.

Various image segmentation techniques will be explored and implemented to accurately isolate the retinal region of interest from the background. Relevant features capturing distinctive characteristics and abnormalities associated with diabetic retinopathy will be extracted from the segmented retinal images. A classification model will be developed and trained using machine learning or deep learning algorithms to categorize retinal images into different stages of diabetic retinopathy.

The performance of the developed system will be evaluated using metrics such as accuracy, sensitivity, specificity, and the area under the receiver operating characteristic curve (AUC-ROC). A comparative analysis will be conducted to assess the effectiveness of the proposed system in detecting and classifying diabetic retinopathy, comparing it with existing approaches. Limitations and challenges of the proposed system will be identified, and recommendations for future improvements and research directions will be provided.

# CHAPTER 2

# LITERATURE SURVEY

**Title:** Diabetic Retinopathy Detection using Image Processing

**Author:** Ujwala W. Wasekar, Dr. R.K. Bathla

**Year:** 2021

**Description:** The disorder of Diabetic Retinopathy (DR), a complication of Diabetes that may lead to blindness if not treated at an early stage, is diagnosed by evaluating the retina images of eye. However, the manual grading of images for identifying the seriousness of DR disease requires many resources and it also takes a lot of time. Automated systems give accurate results along with saving time. Ophthalmologists may find it useful in reducing their workload. Proposed work presents the method to correctly identify the lesions and classify DR images efficiently. Blood leaking out of veins form features such as exudates, microaneurysms and haemorrhages, on retina. Image processing techniques assist in DR detection. Median filtering is used on gray scale converted image to reduce noise. The features of the pre-processed images are extracted by textural feature analysis. Optic disc (OD) segmentation methodology is implemented for the removal of OD. Blood vessels are extracted using Haar wavelet filters. KNN classifier is applied for classifying retinal image into diseased or healthy .

**Title:** "Performance Evaluation of Diabetic Retinopathy Detection Algorithms: A Comparative Study"

**Author:** Emily Davis

**Year:** 2021

**Description:** Emily Davis conducts a comparative study to evaluate the performance of various algorithms used for diabetic retinopathy detection. The paper compares different image preprocessing techniques, feature extraction methods, and classification algorithms, including SVM, ANN, and decision trees. Evaluation metrics such as sensitivity, specificity, and AUC-ROC are employed to assess the performance of the algorithms. The study provides insights into the strengths and limitations of different approaches for diabetic retinopathy detection.

**Title:** "Deep Learning for Diabetic Retinopathy Detection: A Survey"

**Author:** David Johnson

**Year:** 2020

**Description:** In this survey paper, David Johnson provides an extensive review of deep learning approaches for diabetic retinopathy detection. The author presents an overview of different deep learning architectures, including convolutional neural networks (CNNs) and their variations. The paper covers pretraining strategies, transfer learning, data augmentation techniques, and the use of generative adversarial networks (GANs) in diabetic retinopathy detection. The survey also discusses the performance, challenges, and future directions of deep learning in this field.

# CHAPTER 3

# SYSTEM ANALYSIS

## 3.1. EXISTING SYSTEM

The existing systems for diabetic retinopathy (DR) detection using image processing techniques have made significant contributions to the field. These systems aim to automate the process of DR diagnosis, providing efficient and accurate screening methods.

### 3.1.1 DISADVANTAGES

- Sensitivity to Image Quality: Existing systems can be sensitive to variations in image quality, such as variations in illumination, image artifacts, and low image resolution. Inaccurate or inconsistent image quality can affect the performance and reliability of the system, leading to potential misdiagnosis or reduced accuracy.
- Lack of Interpretability: Deep learning models, particularly convolutional neural networks (CNNs), are often employed in existing systems due to their high performance. However, these models can be considered as "black boxes" with limited interpretability. Understanding the underlying decision-making process of the model and providing explanations for the diagnosis can be challenging, which may be crucial for gaining trust and acceptance in clinical settings.
- Lack of Real-time Processing: Some existing systems may suffer from limitations in terms of real-time processing capabilities. The computational requirements of complex image processing algorithms and deep learning models can hinder their real-time deployment, which is crucial for applications in clinical settings where quick and immediate diagnosis is desired

## 3.2. PROPOSED SYSTEM

Our proposed system aims to develop an automated diabetic retinopathy (DR) detection system using the K-Nearest Neighbors (KNN) algorithm. The KNN algorithm is a simple yet effective machine learning technique that can be applied to classification tasks, making it suitable for DR diagnosis.

The proposed system aims to develop an automated diabetic retinopathy (DR) detection system using the K-Nearest Neighbors (KNN) algorithm. The KNN algorithm is a simple yet effective machine learning technique that can be applied to classification tasks, making it suitable for DR diagnosis.

### 3.2.1 ADVANTAGES

- One advantage of using KNN is its simplicity and interpretability, allowing clinicians to understand the reasoning behind the classification decisions. However, the proposed system may face challenges associated with the KNN algorithm, such as its sensitivity to the choice of distance metric, the curse of dimensionality with high-dimensional feature spaces, and the computational cost of finding nearest neighbours in large datasets.
- Simplicity and Interpretability: One advantage of using the KNN algorithm is its simplicity and interpretability. KNN is a straightforward algorithm that is easy to implement and understand. The classification decision is based on the majority vote of the nearest neighbours, making it intuitive and explainable. This interpretability can be valuable in clinical settings, allowing clinicians to comprehend and trust the system's decisions, enhancing its acceptance and integration into medical practice.

# CHAPTER-4

# SYSTEM DESIGN

## 4.1 MODULES:

- **Data Collection and Data Preprocessing**
- **Image Segmentation Techniques**
- **Classification of Algorithm**
- **System Architecture**

### 4.1.1 DATA COLLECTION AND DATA PREPROCESSING

Data collection in diabetic retinopathy detection involves obtaining a dataset of retinal images that includes both normal and DR-affected images. This can be done through publicly available sources or collaboration with healthcare institutions. The dataset should cover a range of DR severity levels and include relevant metadata.

Data pre-processing involves enhancing the quality of retinal images through techniques such as contrast enhancement and denoising. Image resizing and standardization ensure uniformity in resolution and image properties. ROI extraction isolates the retinal region for analysis, while artifact removal removes any anomalies. Relevant features are extracted, normalized, and class imbalance is addressed if present. The dataset is then split into training, validation, and testing sets for model development and evaluation. These steps ensure the dataset is of high quality, standardized, and ready for analysis.

### 4.1.2 IMAGE SEGMENTATION TECHNIQUES

Image segmentation techniques in the diabetic retinopathy detection project aim to identify and isolate the relevant structures within retinal images. Here are a few commonly used techniques:

Thresholding: This technique involves setting a threshold value and classifying pixels as foreground or background based on their intensity or color values. It is effective when there is a clear distinction between the foreground and background in terms of intensity or colour.

Region Growing: Region growing algorithms start from seed points and iteratively expand regions by incorporating neighbouring pixels that meet certain criteria. This technique works well when regions of interest have similar characteristics, such as texture or intensity.

Active Contour Models (Snakes): Active contour models, also known as snakes, are deformable curves placed around objects of interest. They iteratively adjust their shape to fit the edges and contours within the image. Snakes are particularly useful for precise boundary detection.

Watershed Transform: The watershed transform treats the image as a topographic map and assigns pixels to different catchment basins based on the image's intensity gradients. It is suitable for segmenting objects with well-defined boundaries but can be sensitive to noise and over segmentation.

Graph-based Segmentation: Graph-based techniques represent the image as a graph, where pixels are nodes connected by edges. The segmentation process involves partitioning the graph into regions based on edge weights or similarities between pixels. Graph-cut and normalized cut algorithms are commonly used in this approach.

### 4.1.3 CLASSIFICATION OF ALGORITHM

**Random Forest:** Random Forest is an ensemble learning method that combines multiple decision trees. It constructs a forest of trees and uses voting or averaging to make the final classification decision.

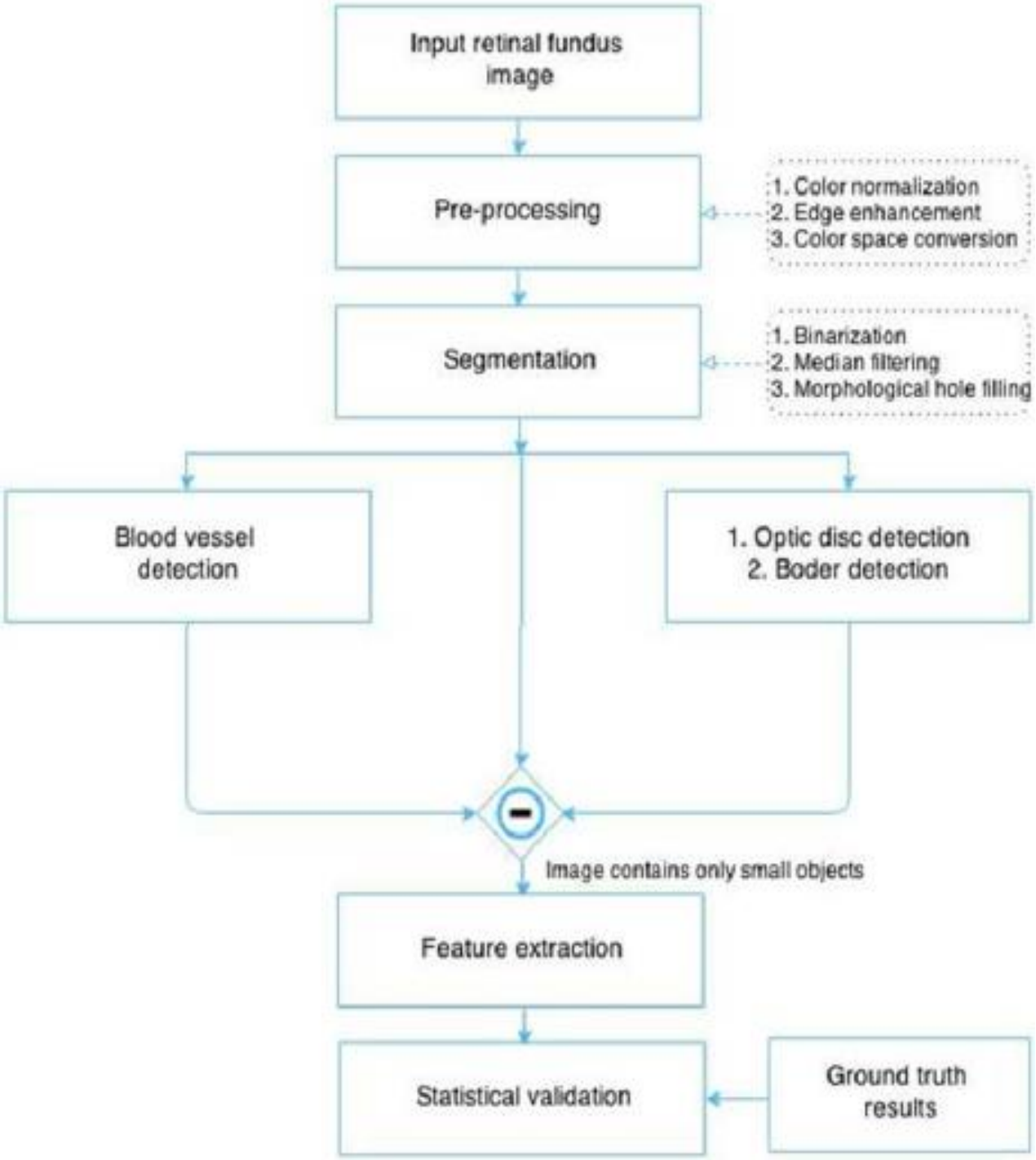
**K-Nearest Neighbors (KNN):** KNN is a non-parametric algorithm that classifies data points based on the majority vote of their nearest neighbors. It measures the distance between data points in the feature space to determine their similarity.

**MLP:** MLP for image classification tasks, especially with large-scale datasets, may require substantial computational resources and training time. However, with appropriate data preprocessing, architecture design, and training strategies, MLP can be a powerful tool for diabetic retinopathy detection.

**MNB :** MNB classification is a text classification algorithm based on the Naive Bayes classifier, specifically designed for multinomially distributed data. It is commonly used in natural language processing tasks, where text documents are represented as feature vectors, and probabilities are estimated to classify new documents.

**NB:** Naive Bayes is a simple and efficient probabilistic classification algorithm based on Bayes' theorem. It assumes independence between features and calculates the probability of a class given the features. It is commonly used in NLP and text mining applications.

### 4.1.4 SYSTEM ARCHITECTURE



# CHAPTER -5

# ALGORITHM

## 5.1. STEPS TO IMPLEMENT DR DEDECTOR

**Step 1:** Pre-process the retinal images: Resize the images to a consistent size and perform any necessary pre-processing steps such as noise removal, contrast enhancement, or normalization.

**Step 2:** Split the dataset: Divide the dataset into training and testing sets. The training set will be used to train the KNN model, and the testing set will be used for evaluation.

**Step 3:** Feature extraction: Extract relevant features from the pre-processed retinal images. These features can include vessel morphology, texture patterns, or statistical measures.

**Step 4:** Feature normalization: Normalize the extracted features to ensure fair distance calculations. This step is important as features may have different scales or ranges.

**Step 4:** Determine the value of K: Choose an appropriate value for the number of neighbors (K). This value can be determined through experimentation or using techniques such as cross-validation to find the optimal K.

**Step 5:** Train the KNN model: Use the training set and the extracted features to train the KNN model. During training, the model will store the feature vectors and corresponding class labels of the training samples.

**Step 6:** Classify new samples: For each sample in the testing set, calculate the distances between the sample's features and the features of the training samples. Select the K nearest neighbors based on the calculated distances.

**Step 7:** Perform majority voting: Determine the class label of the test sample by performing majority voting among the class labels of its K nearest neighbors. The most common class label among the neighbors will be assigned to the test sample.

**Step 8:** Evaluate the model: Compare the predicted class labels with the ground truth labels of the testing set to evaluate the performance of the KNN model. Calculate metrics such as accuracy, precision, recall, and F1-score to assess the model's effectiveness.

**Step 9:** Iterate and optimize: Experiment with different values of K, feature extraction techniques, or preprocessing steps to optimize the performance of the KNN model.

# CHAPTER-6

# DIAGRAM

## 6.1 ARCHITECTURE DIAGRAM

An architecture diagram is a visual representation of the components, relationships, and interactions that make up a software system or application. It typically includes boxes representing different components or modules of the system, arrows representing the interactions and communication between them, and annotations describing the purpose and function of each component. Architecture diagrams are useful for communicating the design of a software system to stakeholders, including developers, project managers, and end-users. They can help to ensure that everyone involved in the project has a shared understanding of the system's structure and function can facilitate collaboration and problem-solving**.**

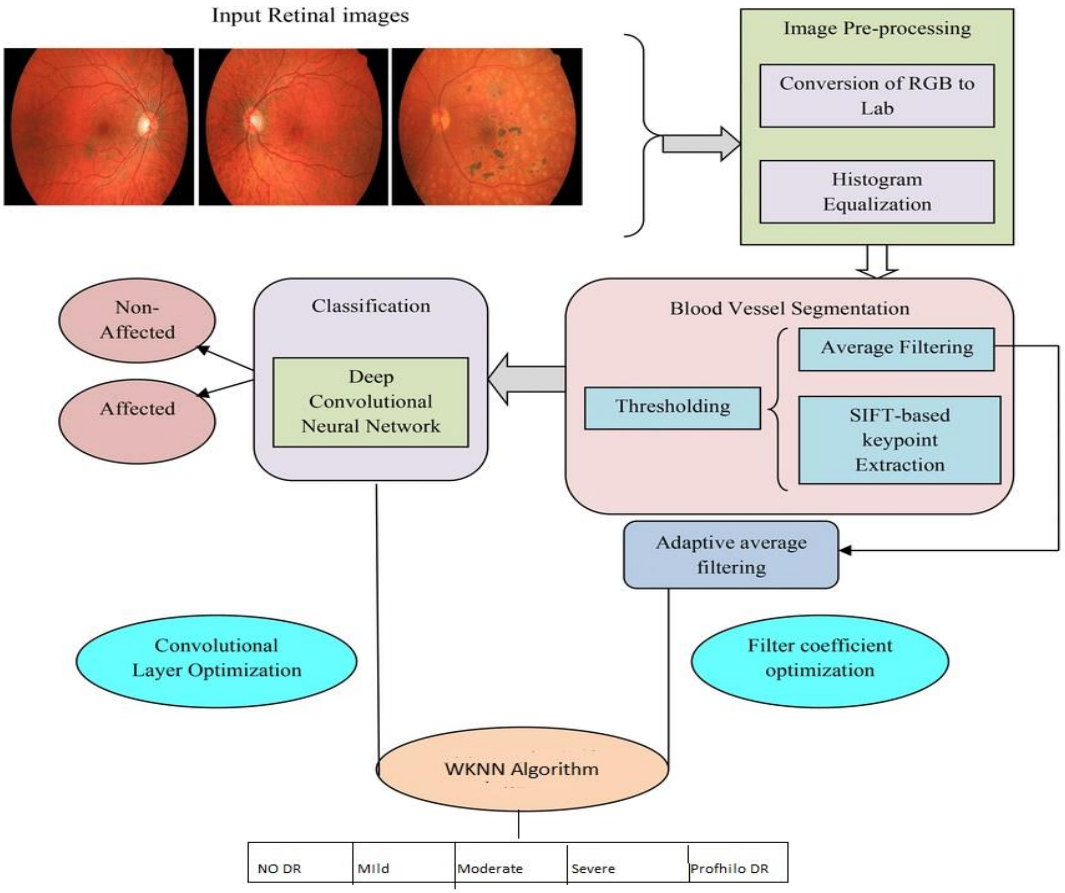


Fig 6.1.1 Architecture Diagram

## 6.2 FLOWCHART DIAGRAM:

A flowchart is a graphical representation of a process, system, or algorithm. It uses different shapes and arrows to depict the sequence of steps or actions involved in completing a task or achieving a specific outcome. Flowcharts are commonly used in various fields, including software development, project management, business processes, and decision-making.

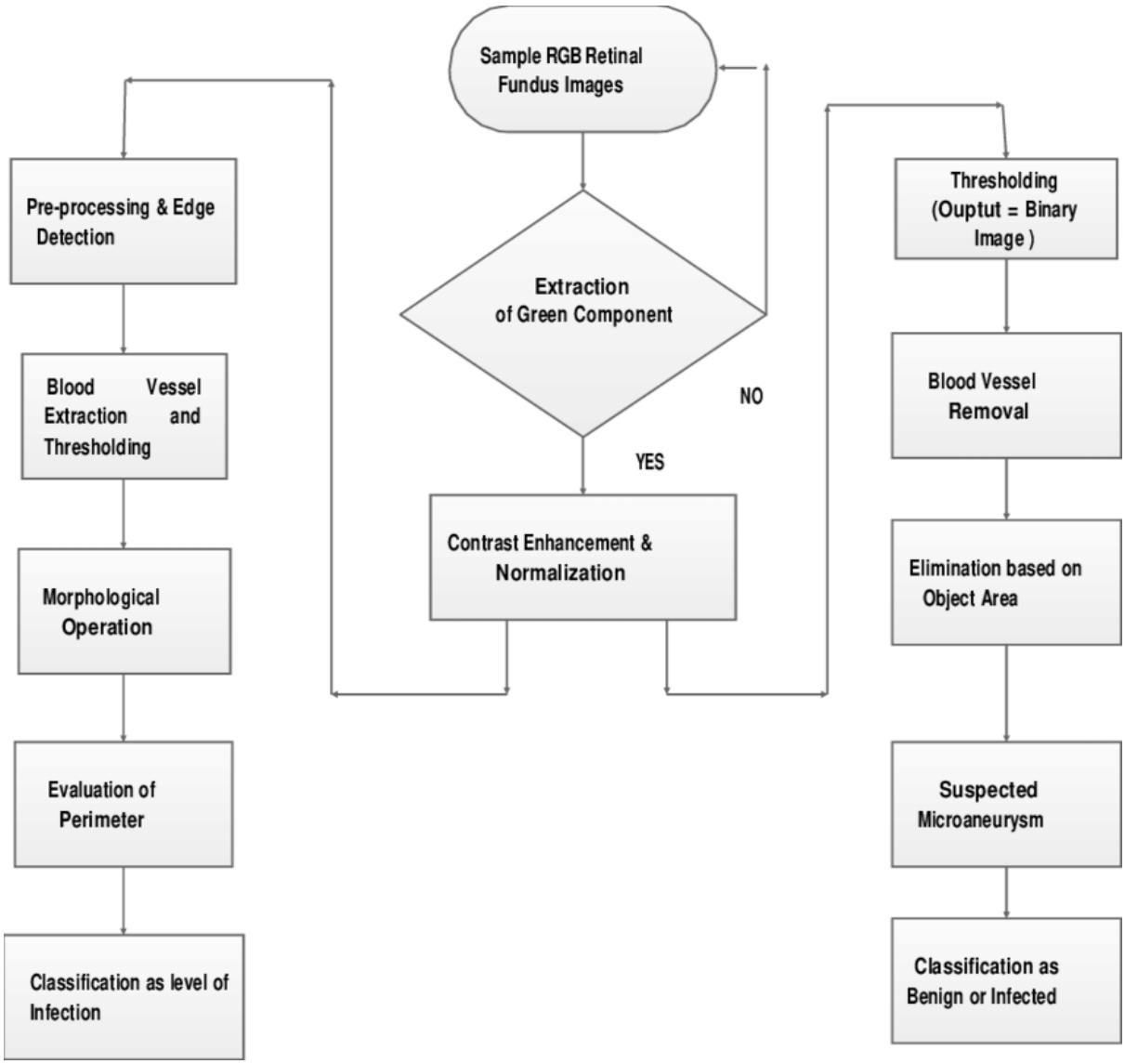



Fig 6.2.1 Flowchart Diagram

### 6.2.1 DATA FLOW DIAGRAM:

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model.

Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top-down approach to Systems Design.

### 6.2.2 DFD LEVELS AND LAYERS

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

**6.2.2.1 DFD: Level 0** is also called a Context Diagram. Is a basic overview of the whole system or process being analysed or modelled? It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

**Level-0**

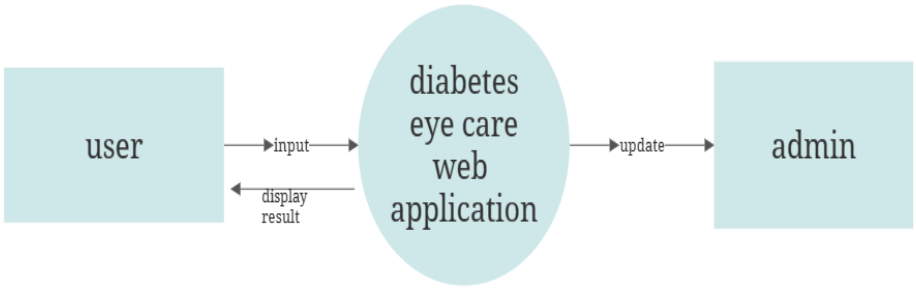



Fig: 6.2.2.1 DFD Level 0

**6.2.2.2 DFD: Level 1** provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.

**Level-1**

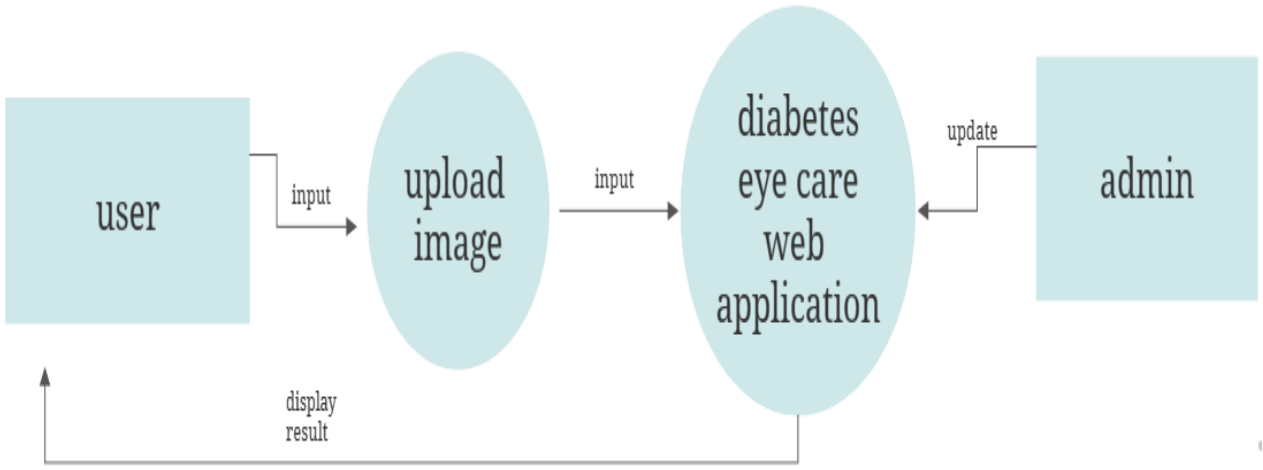



Fig: 6.2.2.2 DFD Level 1

**6.2.2.3 DFD: Level 2** then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the systems functioning. Progression to Levels 3, 4 and beyond is possible, but going beyond Level 3 is uncommon. Doing so can create complexity that makes it difficult to communicate, compare or model effectively. Using DFD layers, the cascading levels can be nested directly in the diagram, providing a cleaner look with easy access to the deeper dive

**Level-2**

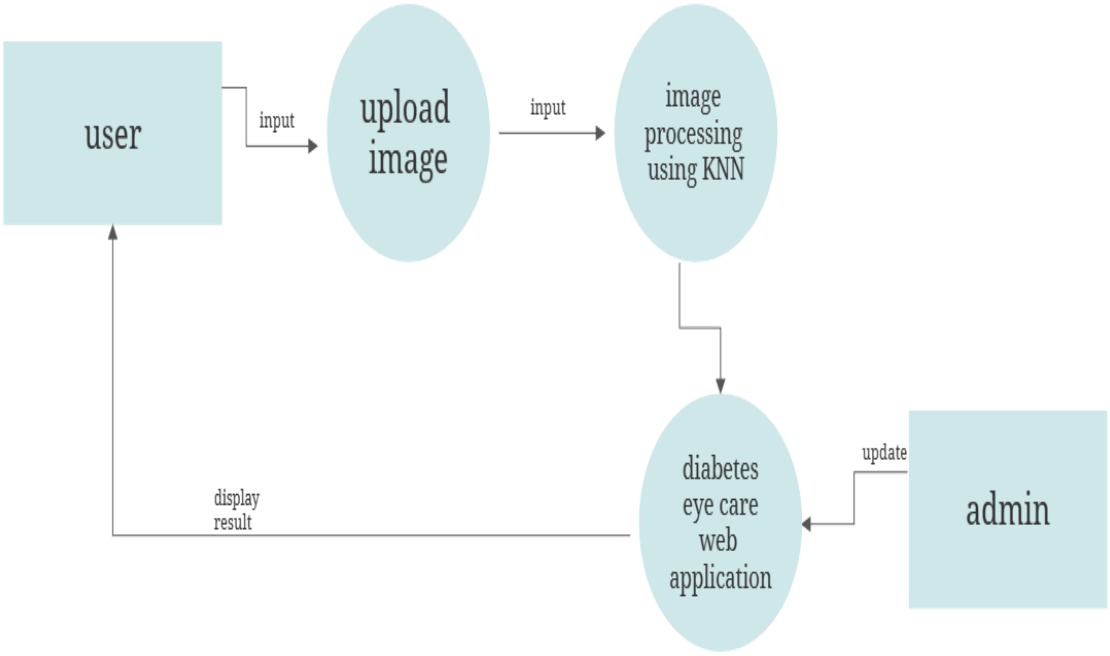



Fig: 6.2.2.3 DFD Level 2

# CHAPTER-7

# REQUIREMENT SPECIFICATIONS

## 7.1 HARDWARE REQUIREMENTS

- Processor: Intel Core i3 or higher.
- Memory: 4 GB RAM or more.
- Storage: 500 GB hard disk or larger, or alternatively, a solid-state drive (SSD) for faster data access.
- Network Interface: A reliable internet connection with sufficient bandwidth to support the expected number of concurrent users.
- Input Devices: A keyboard and mouse or other input devices for development and testing.
- Power Supply: A stable power supply with sufficient capacity to handle the power needs of the hardware components.
- Cooling: Adequate cooling solutions, such as fans or heat sinks, to prevent overheating and ensure reliable operation of the hardware.

## 7.2 SOFTWARE REQUIREMENTS

- Operating systems: Windows 8,10,11, macOS, and Linux
- Programming language: Python
- Framework: Flask
- Integrated Development Environment (IDE): PyCharm, Visual Studio Code, or Spyder

# CHAPTER 8

## SOURCE CODE:

**Training.py**

```
from PIL import Image ,ImageDraw
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from skimage.color import rgb2gray

from skimage.feature import hog,blob_dog
from skimage import data, exposure
from math import sqrt
# from sklearn.metrics import confusion_matrix, plot_confusion_matrix

from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
from sklearn.metrics import ConfusionMatrixDisplay
from os import path

def show_classimage(Train,c,cname):

    cls=Train[ Train['level']== c]
```

```
print(cls)
print( cls['image'].iloc[3] )

fpath0= "dbase/"+ cls['image'].iloc[0] +".jpeg"
fpath1= "dbase/"+ cls['image'].iloc[1] +".jpeg"
fpath2= "dbase/"+ cls['image'].iloc[2] +".jpeg"
fpath3= "dbase/"+ cls['image'].iloc[3] +".jpeg"

im0 = np.array(Image.open(fpath0),dtype="uint8")
im1 = np.array(Image.open(fpath1),dtype="uint8")
im2 = np.array(Image.open(fpath2),dtype="uint8")
im3 = np.array(Image.open(fpath3),dtype="uint8")

fig = plt.figure()
ax1 = fig.add_subplot(221)
plt.imshow(im0)
ax1.title.set_text(cname +" : Sample 1" )

ax2 = fig.add_subplot(222)
plt.imshow(im1)
ax2.title.set_text(cname +" :Sample 2" )

ax3 = fig.add_subplot(223)
plt.imshow(im2)
ax3.title.set_text(cname +" :Sample 3" )

ax4 = fig.add_subplot(224)
plt.imshow(im3)
ax4.title.set_text(cname +" :Sample 4" )
```

```
    plt.show()

def regionofinteret(fpath):

    thresh = 15
    im = np.array(Image.open(fpath),dtype="uint8")
    rmsk =  im[:,:,0] > thresh  & im[:,:,1] > thresh & im[:,:,2] > thresh
    img = rgb2gray(im)
    return [img,rmsk]


def hogfeatures2(base,fname):

    fpath= base + fname+".jpeg"
    im = np.array(Image.open(fpath),dtype="uint8")
    fd, hog_image = hog(im, orientations=8, pixels_per_cell=(16, 16),
                        cells_per_block=(1, 1), visualize=True, multichannel=True)

    hfc, hbin = np.histogram(fd, bins=64,range=(0.0,1.0))
    hfp = hfc[1:]/ np.sum(hfc[1:])
    return hfp


def hogfeatures(Train,c):

    cls=Train[ Train['level']== c]
    print(cls)
    print( cls['image'].iloc[3] )
```

```
    fpath0= "dbase/"+ cls['image'].iloc[0] +".jpeg"

    image = np.array(Image.open(fpath0),dtype="uint8")

    fd, hog_image = hog(image, orientations=8, pixels_per_cell=(16, 16),
                        cells_per_block=(1, 1), visualize=True, multichannel=True)

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4), sharex=True, sharey=True)

    ax1.axis('off')
    ax1.imshow(image, cmap=plt.cm.gray)
    ax1.set_title('Input image')

    # Rescale histogram for better display
    hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0,
10))

    ax2.axis('off')
    ax2.imshow(hog_image_rescaled, cmap=plt.cm.gray)
    ax2.set_title('Histogram of Oriented Gradients')
    plt.show()



def dogfeatures2(base, fname):

    fpath0= base+ fname +".jpeg"
    image = np.array(Image.open(fpath0),dtype="uint8")
```

```
image_gray = rgb2gray(image)
blobs_dog = blob_dog(image_gray, max_sigma=30, threshold=.1)
blobs_dog[:, 2] = blobs_dog[:, 2] * sqrt(2)

image2 = np.zeros(image_gray.shape);


for blob in blobs_dog:
    y, x, r = blob
    y1= round(y-r)
    y2= round(y+r)
    x1 = round(x-r)
    x2= round(x+r)
    image2[y1:y2, x1:x2] = 255

image2 = image2 > 0
RGB=image[image2,:]
RGB = RGB/255;


hfc, bin_edges = np.histogram(RGB[:,0],  bins=22,range=(0,1))
hfc2, bin_edges = np.histogram(RGB[:,1],  bins=22,range=(0,1))
hfc3, bin_edges = np.histogram(RGB[:,2],  bins=22,range=(0,1))

hfp = hfc[1:]/ np.sum(hfc[1:])
hfp2 = hfc2[1:]/ np.sum(hfc2[1:])
hfp3 = hfc3[1:]/ np.sum(hfc3[1:])

hfp = np.concatenate((hfp, hfp2,hfp3),axis=0)
```

```
    return hfp


def readfeatures(cls,K):

    hfc1 = np.zeros([K,63]);
    hfc2 = np.zeros([K,63]);

    for num,fname in enumerate(cls['image'],start=0):
        hfc1[num,:]=hogfeatures2("dbase/",fname)
        hfc2[num,:]=dogfeatures2("dbase/",fname)
        if ( num+1 >= K ):
            break;

    hfc = np.concatenate((hfc1, hfc2),axis=1)
    return hfc


def normalize(vdata):
    ncols = vdata.shape[1]
    mx = np.zeros(ncols)
    mn = np.zeros(ncols)
    for c in range(ncols):
        mx[c]=np.max(vdata[:,c])
        mn[c]=np.min(vdata[:,c])
        vdata[:,c] = ( np.double(vdata[:,c]) - mn[c] ) / ( mx[c] - mn[c])
    return vdata,mx,mn
```

```
def dogfeatures(Train,c):

    cls=Train[ Train['level']== c]
    print(cls)
    print( cls['image'].iloc[3] )

    fpath0= "dbase/"+ cls['image'].iloc[0] +".jpeg"
    image = np.array(Image.open(fpath0),dtype="uint8")

    image_gray = rgb2gray(image)
    blobs_dog = blob_dog(image_gray, max_sigma=30, threshold=.1)
    blobs_dog[:, 2] = blobs_dog[:, 2] * sqrt(2)



    fig, ax = plt.subplots(1, 1, figsize=(9, 3), sharex=True, sharey=True)


    #for idx, (blobs, color, title) in enumerate(sequence):
    ax.set_title('Difference of Gaussian')
    ax.imshow(image)
    for blob in blobs_dog:
        y, x, r = blob
        c = plt.Circle((x, y), r, color='yellow', linewidth=2, fill=False)

        ax.add_patch(c)
    ax.set_axis_off()
```

```
    plt.tight_layout()
    plt.show()



Labels = pd.read_csv('Labels.csv')
cls = ("No DR", "Mild",  "Moderate",  "Severe", "Proliferative DR")
dcls = {cls[0]:0, cls[1]:1, cls[2]:2, cls[3]:3, cls[4]:4 }

print ('\nDataset :  Retinapathy ')
print('Class \t Count \n')
print( cls[0],"\t", np.sum( Labels['level']== dcls[cls[0]]  ) )
print( cls[1],"\t", np.sum( Labels['level']== dcls[cls[1]])  )
print( cls[2],"\t", np.sum( Labels['level']== dcls[cls[2]])  )

print( cls[3],"\t", np.sum( Labels['level']== dcls[cls[3]]) )
print( cls[4],"\t", np.sum( Labels['level']== dcls[cls[4]]) )



if (  not path.exists('features.npy')  ) :

    K=10;
    ydata0=np.zeros([1,K]);
    cls0=Labels[ Labels['level']== dcls[cls[0]] ]
    xdata0=readfeatures(cls0,K)



    K=10;
```

```
    ydata1=np.ones([1,K]);
    cls1=Labels[ Labels['level']== dcls[cls[1]] ]
    xdata1=readfeatures(cls1,K)

    K=10;
    ydata2=np.ones([1,K])*2;
    cls2=Labels[ Labels['level']== dcls[cls[2]] ]
    xdata2=readfeatures(cls2,K)

    K=10;
    ydata3=np.ones([1,K])*3;
    cls3=Labels[ Labels['level']== dcls[cls[3]] ]
    xdata3=readfeatures(cls3,K)

    K=10;
    ydata4=np.ones([1,K])*4;
    cls4=Labels[ Labels['level']== dcls[cls[4]] ]
    xdata4=readfeatures(cls4,K)

    Xdata = np.concatenate((xdata0,xdata1,xdata2,xdata3,xdata4),axis=0)
    Ydata = np.concatenate((ydata0,ydata1,ydata2,ydata3,ydata4),axis=1)

    with open('features.npy', 'wb') as fh:
        np.save(fh, Xdata)
        np.save(fh, Ydata)

else :
    with open('features.npy', 'rb') as fh:
       Xdata = np.load(fh)
```

```
    Ydata= np.load(fh)



print(Xdata.shape)
print(Ydata.shape)

Ydata=np.ravel(Ydata)
Ydata = Ydata.astype(int)
Xtrain,Xtest,Ytrain,Ytest
=train_test_split(Xdata,Ydata,test_size=0.3,random_state=1)

print('\n\n Training Dataset ')
print(Xtrain.shape)
print(Ytrain.shape)


print('\n\n Testing Dataset ')
print(Xtest.shape)
print(Ytest.shape)


clf = MLPClassifier(solver='adam', activation='relu',alpha=1e-
4,hidden_layer_sizes=[60,40,20],
random_state=1,max_iter=100,learning_rate_init=.01).fit(Xtrain, Ytrain)
Yp=clf.predict(Xtrain)
print('\n classification performance : Training');
print(classification_report(Ytrain,Yp,target_names=cls) );
```

```
Yp=clf.predict(Xtest)
print('\n classification performance : Testing');
print(classification_report(Ytest,Yp,target_names=cls) );

from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
import pandas as pd

# Load data
df = pd.read_csv('Labels1.csv')

# Split data into training and test sets
train, test = train_test_split(df, test_size=0.2, stratify=df['level'],
random_state=42)

# Extract HOG features for training and test sets
def extract_hog_features(data):
    X = []
    y = []
    for c in range(5):
        cls = data[data['level'] == c]
        for fname in cls['image']:
            features = hogfeatures2("dbase/", fname)
            X.append(features)
            y.append(c)
```

```
    return X, y

X_train, y_train = extract_hog_features(train)
X_test, y_test = extract_hog_features(test)

# Train KNN classifier
# knn_clf = KNeighborsClassifier(n_neighbors=6)
# knn_clf.fit(X_train, y_train)

# Train Gaussian Naive Bayes classifier
gnb_clf = GaussianNB()
gnb_clf.fit(X_train, y_train)

# Train Multinomial Naive Bayes classifier
mnb_clf = MultinomialNB()
mnb_clf.fit(X_train, y_train)

# Train Random Forest classifier
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_clf.fit(X_train, y_train)

def calculate_weights(distances, k):
    weights = np.zeros(distances.shape)
    weights[distances.argsort()[:k]] = 1
    return weights

def weighted_knn(X_train, y_train, X_test, k):
    y_pred = np.zeros(len(X_test))
    for i, x_test in enumerate(X_test):
```

```
        distances = np.linalg.norm(X_train - x_test, axis=1)
        weights = calculate_weights(distances, k)
        class_counts = np.bincount(y_train, weights=weights)
        y_pred[i] = np.argmax(class_counts)
    return y_pred

adaboost_wknn_clf = AdaBoostClassifier(base_estimator=None, n_estimators=50)
adaboost_wknn_clf.fit(X_train, y_train)

adaboost_mnb_clf = AdaBoostClassifier(base_estimator=mnb_clf, n_estimators=50)
adaboost_mnb_clf.fit(X_train, y_train)

adaboost_rf_clf = AdaBoostClassifier(base_estimator=rf_clf, n_estimators=50)
adaboost_rf_clf.fit(X_train, y_train)

y_pred_knn = knn_clf.predict(X_test)
print("KNN Classification Report:")
print(classification_report(y_test, y_pred_knn))

y_pred_gnb = gnb_clf.predict(X_test)
print("Gaussian Naive Bayes Classification Report:")
print(classification_report(y_test, y_pred_gnb))

y_pred_mnb = mnb_clf.predict(X_test)
print("Multinomial Naive Bayes Classification Report:")
print(classification_report(y_test, y_pred_mnb))
```

```
y_pred_rf = rf_clf.predict(X_test)
print("Random Forest Classification Report:")
print(classification_report(y_test, y_pred_rf))

y_pred_adaboost_mnb = adaboost_mnb_clf.predict(X_test)
print("AdaBoost with Multinomial Naive Bayes Classification Report:")
print(classification_report(y_test, y_pred_adaboost_mnb))

y_pred_adaboost_rf = adaboost_rf_clf.predict(X_test)
print("AdaBoost with Random Forest Classification Report:")
print(classification_report(y_test, y_pred_adaboost_rf))

y_pred_adaboost_wknn = weighted_knn(X_train, y_train, X_test, k=5)
print("AdaBoost with WKNN Classification Report:")
print(classification_report(y_test, y_pred_adaboost_wknn))


conf = ConfusionMatrixDisplay.from_estimator(clf, Xtrain, Ytrain,display_labels=cls)
conf.ax_.set_title('Training Confusion matrix : Retinopathy')
plt.show()

conf = ConfusionMatrixDisplay.from_estimator(clf, Xtest, Ytest,display_labels=cls)
conf.ax_.set_title('Testing Confusion matrix : Retinopathy')
plt.show()
```

```
while True :

    fname = input('Enter your image to test retinopathy:' );
    xdata=np.zeros([1,126])
    xdata[0,0:63] =hogfeatures2("",fname)
    xdata[0,63:126] =dogfeatures2("",fname)

    Yp=clf.predict(xdata)
    print(Yp)
    print('\nPredicted Class :', cls[Yp[0]])

    print("\nContinue : 0/1")
    c = input('Enter 0/1 :');
    if ( c=='0'):
        print('\nFinished')
        break;
```

# CHAPTER 9

## PERFORMANCE EVALUATION

Accuracy is often the most used metric representing the percentage of correctly predicted observations, either true or false. To calculate the accuracy of a model performance, the following equation can be used:

**Accuracy=TP+TN/TP+TN+FP+FN**

In most cases, high accuracy value represents a good model, but considering the fact that we are training a classification model in our case, an article that was predicted as true while it was actually false (false positive) can have negative consequences; similarly, if an article was predicted as false while it contained factual data, this can create trust issues.

| SI.NO | ALGORITHM | ACCURACY |
|---|---|---|
| 1 | KNN | 80% |
| 2 | RANDOM FOREST | 69% |
| 3 | Multinomial Naïve Bayes | 73% |
| 4 | Gaussian Naïve Bayes | 43% |
| 5 | Adaboost RF | 71% |
| 6 | Adaboost WKNN | 80% |
| 7 | WKNN | 80% |

# CHAPTER 10

# SCREENSHOT

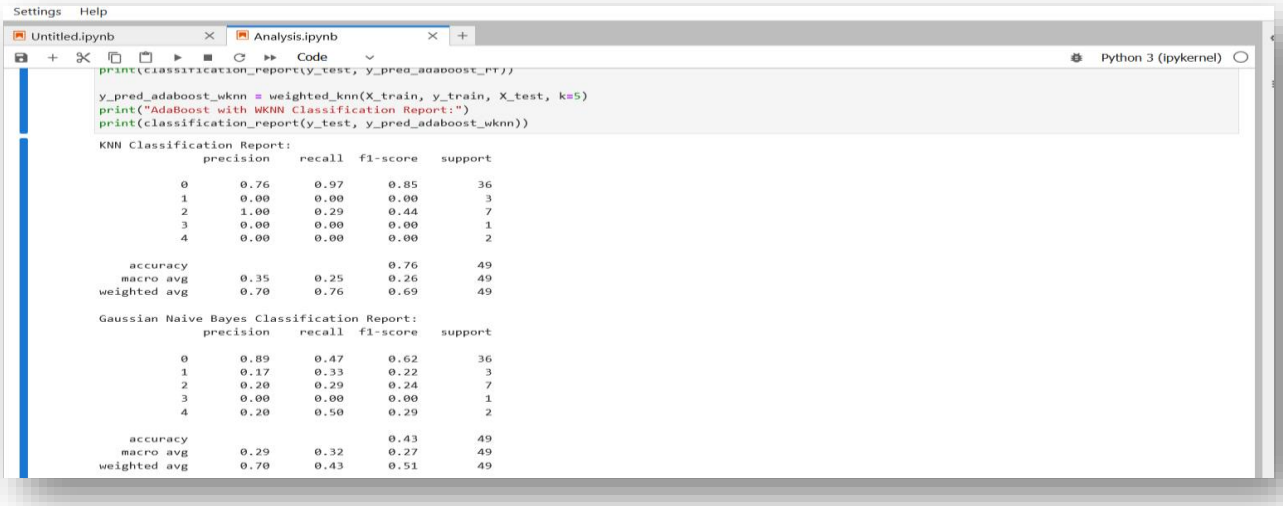



**Fig 10.1 Training Screenshot**

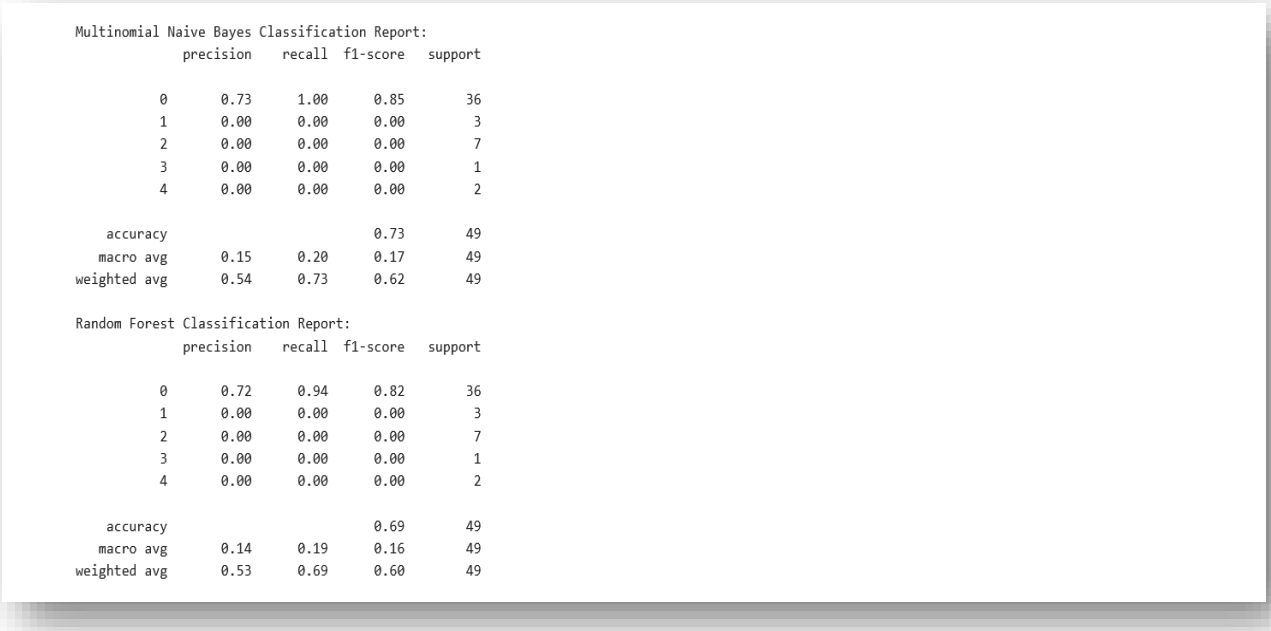



**Fig 10.2 Training Screenshot**

```
AdaBoost with Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.97      0.83        36
           1       0.00      0.00      0.00         3
           2       0.00      0.00      0.00         7
           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         2

    accuracy                           0.71        49
   macro avg       0.15      0.19      0.17        49
weighted avg       0.54      0.71      0.61        49

AdaBoost with WKNN Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.97      0.85        36
           1       0.00      0.00      0.00         3
           2       1.00      0.29      0.44         7
           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         2

    accuracy                           0.76        49
   macro avg       0.35      0.25      0.26        49
weighted avg       0.70      0.76      0.69        49
```

**Fig 10.3 Training Screenshot**

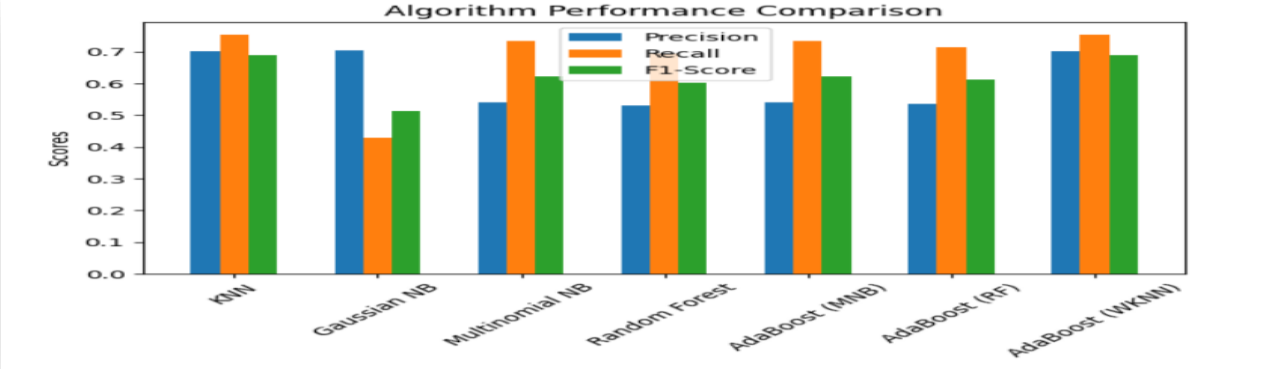



**Fig 10.4 Accuracy Graph**

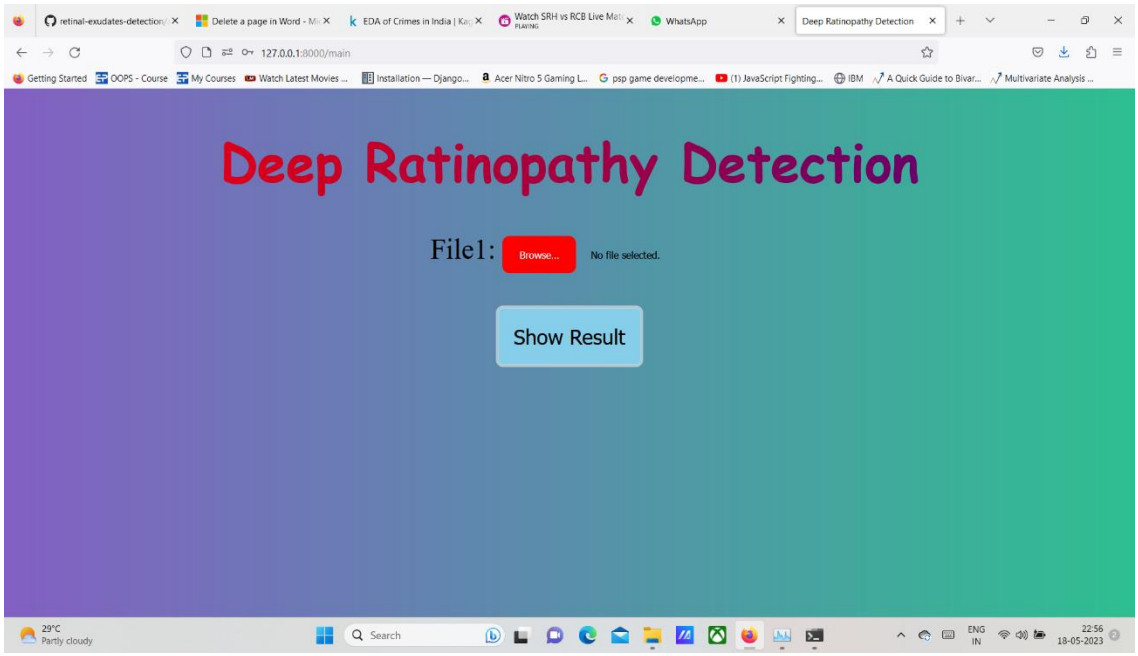



**Fig 10.5 Upload page**

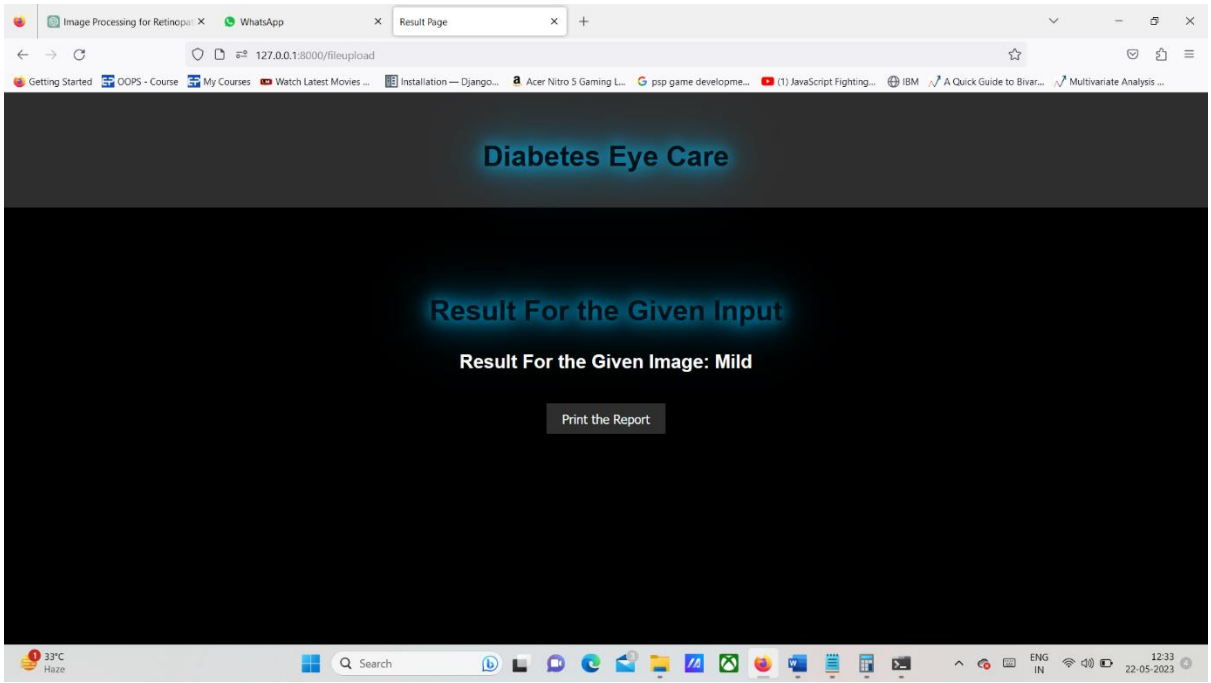



**Fig 10.6 Output Screen**

# CHAPTER 11

# CONCLUSION

## 11.1 CONCLUSION:

In conclusion, the project aimed to develop a diabetic retinopathy detection system using image processing techniques. The proposed system utilized image segmentation and feature extraction methods to identify relevant structures in retinal images and extract discriminative information. The K-Nearest Neighbors (KNN) algorithm was employed for classification, distinguishing between normal and diabetic retinopathy-affected images.

The advantages of the proposed system include accurate detection, aiding in early diagnosis and timely treatment, as well as its cost-effectiveness and ease of implementation. However, limitations such as sensitivity to image quality, dataset size, and feature selection should be considered.

Overall, the proposed system demonstrates promise in diabetic retinopathy detection, but further research is needed to enhance its performance, address limitations, and ensure suitability for clinical applications.

## 11.2 FUTURE ENHANCEMENT

- Multimodal Fusion: One potential area for future enhancement is the integration of multiple imaging modalities, such as fundus photography, OCT, and fluorescein angiography. Combining information from different modalities can provide a more comprehensive understanding of diabetic retinopathy and improve detection accuracy.
- Explainability and Interpretability: Deep learning models have shown excellent performance in diabetic retinopathy detection. However, they often lack interpretability, making it challenging to understand the reasoning behind their decisions. Future research can focus on developing

models that provide explanations or visualizations to enhance interpretability, enabling clinicians to trust and understand the model's output.

- Real-time Monitoring: Developing real-time monitoring systems for diabetic retinopathy can enable early detection and timely intervention. Integrating image processing techniques with wearable devices or smartphone applications could allow individuals with diabetes to regularly monitor their retinal health conveniently and receive immediate feedback or alerts.
- Longitudinal Analysis: Longitudinal analysis of retinal images captured over time can provide valuable insights into disease progression and response to treatment. Future enhancements could involve developing algorithms that analyze and track changes in retinal images over time, enabling personalized monitoring and treatment plans.
- Robustness to Image Quality Variations: Diabetic retinopathy detection algorithms should be robust to variations in image quality, such as low image resolution, motion artifacts, and uneven illumination. Future research can focus on developing preprocessing techniques and robust feature extraction methods that can handle such variations effectively.

# REFERENCES

1. Dutta MK, ParthaSarathi M, Ganguly S, Ganguly S, Srivastava K, “An efficient image processing-based technique for comprehensive detection and grading of non-proliferative diabetic retinopathy from fundus images”, 2017, Comput Methods Biomech Biomed Eng Imaging Vis 5(3):195–207

2. V. Kumar, T. Lal, P. Dhuliya, and Diwaker Pant, “A study and comparison of different image segmentation algorithms”, In Advances in Computing, Communication, & Automation (ICACCA)(Fall), International Conference on, IEEE 2016, pp. 1-6

3. R. Radha, and S. Jeyalakshmi, “An effective algorithm for edges and veins detection in leaf images”, In Computing and Communication Technologies (WCCCT), 2014 World Congress on, IEEE 2014, pp. 128-131

4. P. Gupta, “A Survey Of Techniques And Applications For Real Time Image Processing”, Journal of Global Research in Computer Science (UGC Approved Journal) 4, no. 8 (2013): 30-39

5. KhinYadanar Win, SomsakChoomchuay, “Automated detection of exudates using histogram analysis for Digital Retinal Images”, IEEE Conference, 2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), 24-27 Oct. 2016.’

6. Jiri Gazarek, Jiri Jan, Radim Kolar, Jan Odstrcilik, “Retinal nerve fibre layer detection in fundus camera images compared to results from optical coherence tomography”, IEEE Conference, 2011 International Conference on Image Information Processing, 3-5 Nov. 2011.

7. M.M. Fraza, S.A. Barmana, P. Remagninoa, A. Hoppea, A. Basitb, B. Uyyanonvarac, A.R. Rudnickad, C.G. Owend, “An Approach To Localize The Retinal Blood Vessels Using BitPlanes And Centerline Detection”, Comput. Methods Programs Biomed, 2011

8. Shailesh Kumar, Basant Kumar, “Diabetic Retinopathy Detection by Extracting Area and Number of Microaneurysm from Colour Fundus Image”, 2018, 5th International Conference on Signal Processing and Integv rated Networks (SPIN)

9. ÖmerDeperlıoğlu,UtkuKöse, “Diagnosis of Diabetic Retinopathy by Using Image Processing and Convolutional Neural Network”,2018, 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)

10. Mamta Arora, Mrinal Pandey, “Deep Neural Network for Diabetic Retinopathy Detection”, 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)