# Supervised Machine Learning Framework for Glaucoma Classification using Retinal Fundus Images

*A project report submitted to MALLA REDDY UNIVERSITY*
*in partial fulfilment of the requirements for the award of degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING (AI & ML)**


**Submitted by**

| | | |
|---|---|---|
| **K. Sharath Chandra** | **:** | **2011CS020203** |
| **K. Varun** | **:** | **2011CS020204** |
| **L. Rakesh** | **:** | **2011CS020205** |
| **L. Ashrith** | **:** | **2011CS020206** |

*Under the Guidance of*

**Dr. A.Kiran Kumar**

**Associate Professor**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)**

**MALLA REDDY UNIVERSITY**

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)


2023

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

## COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled, **"Supervised Machine Learning Framework for Glaucoma Classification using Retinal Fundus Images"** submitted by K.Sharath Chandra(2011CS020203), K.Varun(2011CS020204), L.Rakesh(2011CS020205), L.Ashrith(2011CS020206) B. Tech III year I semester, Department of CSE (AI &ML) during the year 2022-2023. The resultsembodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

**INTERNAL GUIDE**

**Dr. A.Kiran Kumar**

**HEAD OF THE DEPARTMENT**

**Dr.ThayyabaKhatoon**
**CSE(AI&ML)**

# ABSTRACT

Glaucoma is a neuro-degenerative eye disease developed due to an increase in the Intra-ocular Pressure inside the retina. Being the second largest cause of blindness worldwide, it can lead the person towards complete blindness if an early diagnosis does not take place. With respect to this underlying issue, there is an immense need of developing a system that can effectively work in the absence of excessive equipment's, skilled medical practitioners and also is less time consuming. This is a systematic review on the main algorithms using machine learning (ML) in retinal image processing for glaucoma diagnosis and detection. ML has proven to be a significant tool for the development of computer aided technology. Furthermore, secondary research has been widely conducted over the years for ophthalmologists. Such aspects indicate the importance of ML in the context of retinal image processing.

## CONTENTS


| CHAPTER NO. | TITLE | | | PAGE NO. |
|---|---|---|---|---|
| 1 | INTRODUCTION | | | |
| | 1.1 | Problem definition | | 1 |
| | 1.2 | Objective of Project | | 1 |
| | 1.3 | Limitations of Project | | 1 |
| | | | | |
| 2 | ANALYSIS | | | |
| | 2.1 | Introduction | | 2 |
| | 2.2 | Requirement Specifications | | 2 |
| | | 2.2.1 | Software Requirements | 2 |
| | | 2.2.2 | Hardware Requirements | 2 |
| | 2.3 | Existing Systems | | 3 |
| | 2.4 | Proposed Systems | | 3 |
| | 2.5 | Modules | | 3 |
| | 2.6 | Architecture | | 4 |
| | | | | |
| 3 | DESIGN | | | |
| | 3.1 | Introduction | | 5 |
| | 3.2 | Data Flow Diagram | | 5 |
| | 3.3 | Data set Descriptions | | 5 |
| | 3.4 | Data Preprocessing Techniques | | 6 |
| | 3.5 | Methods & Algorithms | | 6 |
| | 3.6 | Building a Model | | 6 |
| | 3.7 | Evaluation | | 7 |
| | | | | |
| 4 | DEPLOYMENTS & RESULTS | | | |
| | 4.1 | Introduction | | 8 |
| | 4.2 | Source Code | | 8 |
| | 4.3 | Final Results | | 12 |

# 1. INTRODUCTION

## 1.1 PROBLEM DEFINITION:

Most current algorithms for automatic glaucoma assessment using fundus images rely on handcrafted features based on segmentation, which are affected by the performance of the chosen segmentation method and the extracted features. In this project, we develop a Deep learning architecture with Convolutional neural network for automated glaucoma diagnosis as CNN is known for its ability to learn highly discriminative features from raw pixel intensities.

## 1.2 OBJECTIVE OF PROJECT:

This paper describes supervised methods for glaucoma screening in retinal images. The studies reviewed in this article were categorized into deep learning and non-deep learning methods. Hence, its main objective is to evaluate the algorithms recently proposed by different groups, as well as to describe the preeminent steps in the development of an automated diagnosis system. Further, machine learning algorithms can be of significant contribution to the earlier and automated diagnosis of glaucoma, as well as for other abnormal ocular conditions.

## 1.3 LIMITATIONS OF THE PROJECT:

- It becomes difficult for the model to detect Glaucoma if the quality of Fundus image is low.
- It works only with fundus images.

# 2. ANALYSIS

## 2.1 INTRODUCTION:

Glaucoma is a condition where the eye's optic nerve, which provides information to the brain, is damaged with or without raised intraocular pressure. It causes damage to the optic nerve causing loss of vision due to many reasons. It causes blurred vision, glare, eye pain, headache, and narrowed vision. If untreated, this will' cause gradual vision loss and Blindness. The most basic diagnostic tool used for glaucoma diagnosis is the analysis of color fundus images. The word “fundus” derived from Latin language, refers to the part of the eyeball that lies opposite to the pupil. Thus, a photograph of the interior surface of the eye is regarded as the fundus image. Apart from funduscopic images, there are also other imaging techniques namely Scanning Laser Polarimetry, Heidelberg Retina Tomography (HRT) and Optical Coherence Tomography (OCT), but the cost of these equipment’s comes across as a barrier in the context of affordability for many of the hospitals worldwide. Therefore, we work on glaucoma detection using fundus images which are an affordable alternative for health practitioners.

## 2.2 SOFTWARE REQUIREMENT SPECIFICATION:

### Software Requirements:

- Jupyter Notebook.
- Anaconda navigator.
- Windows 7 and above
- Python 3.7 and above

### Hardware Requirements:

- CPU– Intel i5 6th gen and above.
- RAM 16/32GB DDR4
- GPU- For basic requirements an integrated GPU is required
- Storage -500GB and above.

## 2.3 EXISTING SYSTEM

- Usually, the doctors used to go with the traditional method of comparing images.
- It is very difficult for a human being to do classification for many numbers of sample.
- It is time consuming approach.

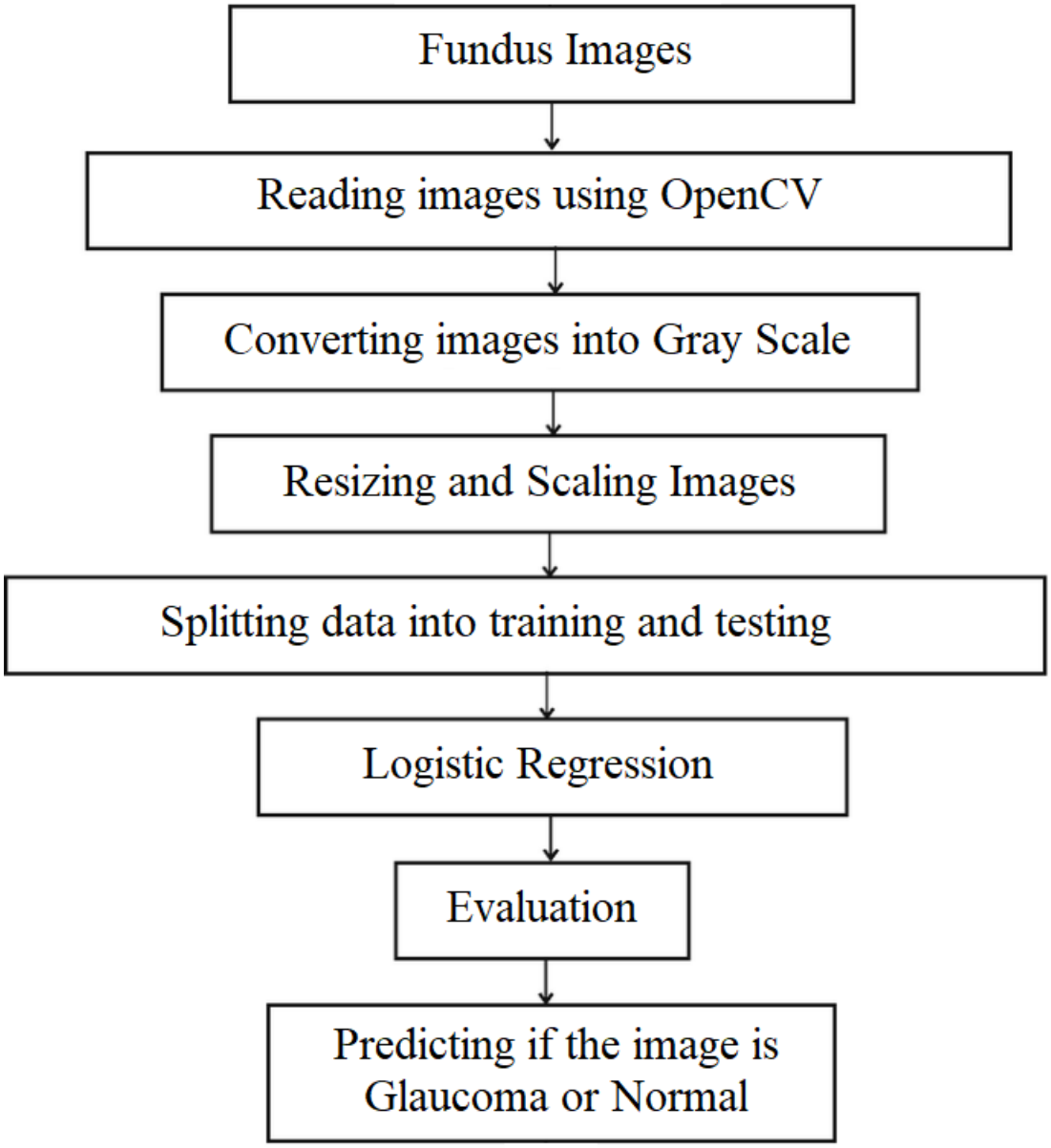
## 2.4 PROPOSED SYSTEM

- A machine learning model is created for detecting Glaucoma in an eye using fundus image.
- Fundus image of an eye is given as an input for this model, it detects whether the eye has a Glaucoma or not.
- It can find the results for many numbers of samples within a very short time.
- It saves a lot of time

## 2.5 Modules

- Numpy
- Pandas
- Matplotlib
- Seaborn
- OS
- OpenCV
- Scikit-learn

## 2.6 ARCHITECTURE

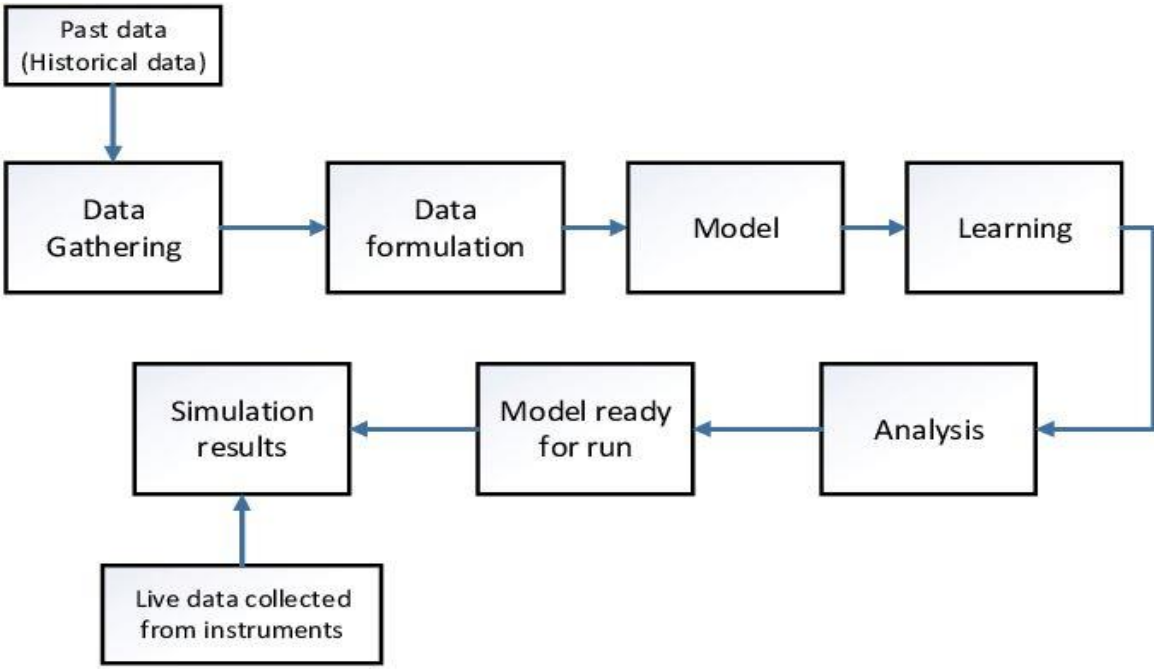


# 3.DESIGN

## 3.1 Introduction

Image classification is the process of categorizing images into one or more predefined classes based on their visual content. It is a common task in machine learning and computer vision, and there are many approaches to solving this problem. One of the most basic approaches to image classification is to use a simple machine learning algorithm, such as k-nearest neighbors or logistic regression, and train it on a dataset of images and their corresponding labels. These algorithms can be trained using various techniques, such as stochastic gradient descent or backpropagation, to learn the relationships between the images and their labels.

## 3.2 Data Flow Diagram




## 3.3 Data set Descriptions

A fundus image is a photograph of the interior surface of the eye, including the retina, optic disc, and blood vessels. Fundus images are often used for the detection and diagnosis of a variety of eye conditions, including glaucoma. In the context of glaucoma detection, a fundus image dataset may consist of a collection of fundus images that have been annotated with labels indicating the presence or absence of glaucoma.

## 3.4 Data Preprocessing Techniques

There are several data preprocessing techniques that can be applied to retinal fundus images for glaucoma detection:

- Image enhancement: This involves techniques such as contrast stretching, histogram equalization, and sharpening to improve the visibility and clarity of features in the image.
- Noise reduction: This involves techniques such as filtering and smoothing to remove noise and artifacts from the image.
- Segmentation: This involves techniques such as thresholding, region growing, and active contours to separate the relevant structures in the image from the background.
- Feature extraction: This involves techniques such as edge detection, texture analysis, and shape analysis to extract relevant features from the image that can be used for glaucoma detection.
- Data augmentation: This involves techniques such as rotation, translation, and scaling to generate additional training data from the existing dataset.

It is important to carefully preprocess the images to ensure that they are suitable for use in a glaucoma detection model.

## 3.5 Methods & Algorithms

Logistic regression is a statistical method for predicting binary outcomes, such as the probability of an event occurring. It is a supervised learning algorithm, which means that it requires a labeled dataset for training. The goal of logistic regression is to find the best coefficients for a set of features that will give the highest accuracy in predicting the outcome. In logistic regression, the predicted probability is transformed into a binary outcome using a threshold. For example, if the probability of an event occurring is greater than 0.5, the outcome is predicted as "yes," and if the probability is less than 0.5, the outcome is predicted as "no." The threshold of 0.5 can be adjusted based on the specific requirements of the problem. Logistic regression is a widely used method for classification tasks, and it has the advantage of being simple and efficient. It can be extended to handle multiclass classification by using one-versus-all or one-versus-rest schemes.

## 3.6 Building a Model

To build a model for glaucoma detection using retinal fundus images with supervised machine learning.

- Collect and label a dataset of retinal fundus images. This dataset should include both images with glaucoma and images without glaucoma.
- Preprocess the images to extract relevant features. This may involve techniques such as

image enhancement, noise reduction, segmentation, and feature extraction.

- Split the dataset into a training set and a test set. The training set will be used to train the model, and the test set will be used to evaluate the performance of the model.
- Train a machine learning model using the training set. There are several supervised machine learning algorithms that can be used for this task, including logistic regression, support vector machines, and decision trees.
- Evaluate the performance of the model on the test set. This will give you an idea of how well the model is able to detect glaucoma in new images.
- Fine-tune the model by adjusting the hyperparameters and/or the features used as input. This can help to improve the model's performance.
- Use the trained model to predict the presence or absence of glaucoma in new retinal fundus images.

## 3.7 Evaluation

To evaluate the performance of a model for glaucoma detection using retinal fundus images with supervised machine learning, you can use a variety of metrics. Some common evaluation metrics for classification tasks include:

- Accuracy: This is the proportion of correct predictions made by the model. It can be calculated as the number of correct predictions divided by the total number of predictions.
- Precision: This is the proportion of positive predictions that are correct. It can be calculated as the number of true positive predictions divided by the total number of positive predictions.
- Recall: This is the proportion of actual positive cases that are correctly predicted. It can be calculated as the number of true positive predictions divided by the total number of actual positive cases.
- F1 score: This is a balance between precision and recall. It can be calculated as the harmonic mean of precision and recall.
- Confusion matrix: This is a table that shows the number of true positive, true negative, false positive, and false negative predictions made by the model.

# 4. DEPLOYMENTS & RESULTS

## 4.1 Introduction

Once you have trained and evaluated a model for glaucoma detection using retinal fundus images with supervised machine learning, you can deploy the model in a production environment to make predictions on new images. This can involve integrating the model into a larger system or deploying it as a standalone application. When deploying the model, it is important to consider factors such as the computational resources required to run the model, the latency of the predictions, and the robustness of the model to changes in the input data distribution. To obtain the results of the model's predictions, you can use the model to make predictions on new images and compare the predicted labels to the true labels. This will allow you to evaluate the performance of the model on unseen data and determine how well the model generalizes to new images.

## 4.2 Source Code

In [1]:
```
#importing required modules
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import os
import cv2 as cv
import sklearn
```

In [3]:
```
#accessing folders inside the specified path
folders=[]
DIR=r'D:\AD\dataset\train'
for folder in os.listdir(DIR):
    folders.append(folder)
folders
```

Out[3]: ['class0', 'class1']

In [4]:
```
classes=['No Glaucoma','Glaucoma']
```

In [5]:
```
labels=[]
features=[]
```

In [7]:
```
#creating x(features) and y(labels) data
for folder in folders:
    path=os.path.join(DIR,folder)
    for image in os.listdir(path):
        if folder=='class0':
            labels.append(0)
        else:
            labels.append(1)
        image_path=os.path.join(path,image)
        img=cv.imread(image_path)
        gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)
        resized=cv.resize(gray,(290,290))
        features.append(resized)
```

In [10]:
```
#converting "features" into numpy array
x=np.array(features)
```

In [11]:
```
x.shape
```

Out[11]: (910, 290, 290)

In [12]:
```
#converting "labels" into numpy array
y=np.array(labels)
y.shape
```

Out[12]: (910,)

In [15]:
```
#creating x_test and y_test data
test_features=[]
test_labels=[]
DIR=r'D:\AD\dataset\test'
for folder in folders:
    path=os.path.join(DIR,folder)
    for image in os.listdir(path):
        if folder=='class0':
            test_labels.append(0)
        else:
            test_labels.append(1)
        image_path=os.path.join(path,image)
        img=cv.imread(image_path)
        gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)
        resized=cv.resize(gray,(290,290))
        test_features.append(resized)
```

In [18]:
```
x_test.shape
```

Out[18]: (88, 290, 290)

In [19]:
```
y_test=np.array(test_labels)
y_test
```

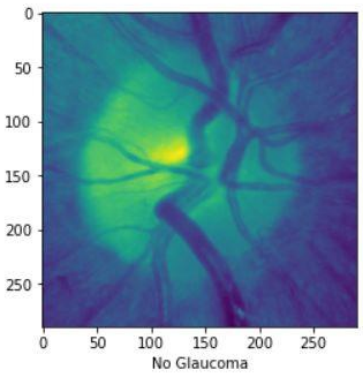
Out[19]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

In [20]:
```
y_test.shape
```

Out[20]: (88,)

In [21]:
```
#function to plot a sample data
def plot_sample(x,y,index):
    plt.imshow(x[index])
    plt.xlabel(classes[y[index]])
    plt.show()
```

In [22]:
```
plot_sample(x,y,34)
```




In [23]:
```
plot_sample(x,y,412)
```

In [43]:
```
true_positive=cm[0][0]
false_negative=cm[0][1]
false_positive=cm[1][0]
true_negative=cm[1][1]
print("True positive : ",true_positive)
print("False negative : ",false_negative)
print("False positive : ",false_positive)
print("True negative : ",true_negative)
```

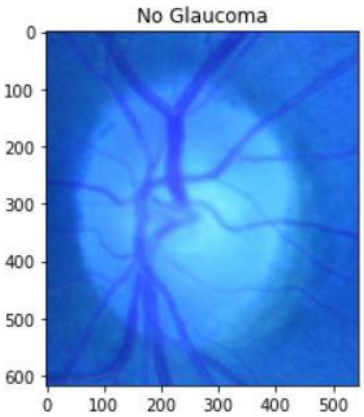
```
True positive :  43
False negative :  3
False positive :  1
True negative :  41
```

In [44]:
```
print(classification_report(y_test,y_predicted))
```

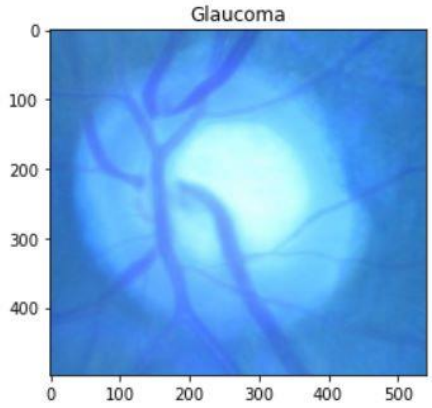
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.98 | 0.96 | 44 |
| 1 | 0.98 | 0.93 | 0.95 | 44 |
| accuracy | | | 0.95 | 88 |
| macro avg | 0.96 | 0.95 | 0.95 | 88 |
| weighted avg | 0.96 | 0.95 | 0.95 | 88 |

In [45]:
```
def predict_and_plot(path):
    img=cv.imread(path)
    gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)
    resized=cv.resize(gray,(290,290))
    x_var=np.array(resized)
    x_var=x_var.reshape(-1,290*290)/255
    predicted=model.predict(x_var)
    plt.imshow(img)
    plt.title(classes[predicted[0]])
    plt.show()
```

In [46]:
```
path=r'D:\AD\dataset\test\class0\250.jpg'
predict_and_plot(path)
```



In [47]:
```
path=r'D:\AD\dataset\test\class1\354.jpg'
predict_and_plot(path)
```




In [48]:
```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
```

In [54]:
```
results=[]
for model_name,classifier in models.items():
    model=classifier['model']
    model.fit(scaled_x,y)
    accuracy=model.score(scaled_x_test,y_test)
    y_predicted=model.predict(scaled_x_test)
    cm=confusion_matrix(y_test,y_predicted)
    true_positive=cm[0][0]
    false_negative=cm[0][1]
    false_positive=cm[1][0]
    true_negative=cm[1][1]
    precision=true_positive/(true_positive+false_positive)
    recall=true_positive/(true_positive+false_negative)
    f1_score=2*(precision*recall)/(precision+recall)
    results.append({
        "Model name":model_name,
        "Accuracy":accuracy,
        "Precision":precision,
        "Recall":recall,
        "F1 score":f1_score
    })
data=pd.DataFrame(results)
data
```

Out[54]:

| | Model name | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|---|
| 0 | svm | 0.840909 | 0.875000 | 0.795455 | 0.833333 |
| 1 | random_forest | 0.886364 | 0.826923 | 0.977273 | 0.895833 |
| 2 | logistic_regression | 0.954545 | 0.934783 | 0.977273 | 0.955556 |
| 3 | naive_bayes_gaussian | 0.636364 | 0.596774 | 0.840909 | 0.698113 |
| 4 | naive_bayes_multinomial | 0.693182 | 0.688889 | 0.704545 | 0.696629 |
| 5 | decision_tree | 0.840909 | 0.941176 | 0.727273 | 0.820513 |

In [56]:
```
#Adaboost with DecisionTree
from sklearn.ensemble import AdaBoostClassifier
adaBoost=AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),n_estimators=10)
adaBoost.fit(scaled_x,y)
```

Out[56]:
```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=10)
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [57]:
```
adaBoost.score(scaled_x_test,y_test)
```

Out[57]:
```
0.8295454545454546
```

In [58]:
```
#Adaboost with Logistic Regression
from sklearn.ensemble import AdaBoostClassifier
adaBoost2=AdaBoostClassifier(base_estimator=LogisticRegression(max_iter=1000),n_estimators=10)
adaBoost2.fit(scaled_x,y)
```

Out[58]:
```
AdaBoostClassifier(base_estimator=LogisticRegression(max_iter=1000),
                   n_estimators=10)
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [59]:
```
adaBoost2.score(scaled_x_test,y_test)
```

Out[59]:
```
0.9090909090909091
```

In [60]:
```
#Gradient Boosting
from sklearn.ensemble import GradientBoostingClassifier
gradBoost=GradientBoostingClassifier(n_estimators=10)
gradBoost.fit(scaled_x,y)
```

In [61]:
```
gradBoost.score(scaled_x_test,y_test)
```

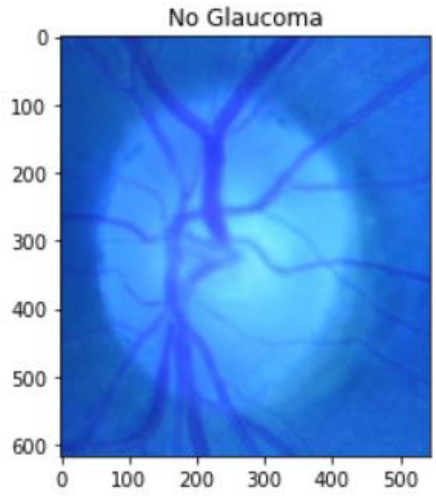
Out[61]:
```
0.7840909090909091
```

## 4.3 Final Results

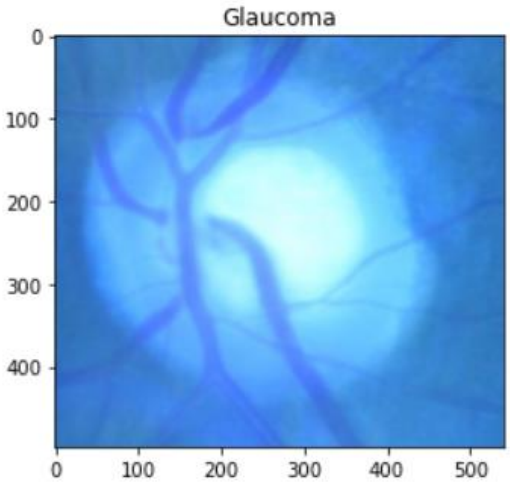
Out[54]:

| | Model name | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|---|
| 0 | svm | 0.840909 | 0.875000 | 0.795455 | 0.833333 |
| 1 | random_forest | 0.886364 | 0.826923 | 0.977273 | 0.895833 |
| 2 | logistic_regression | 0.954545 | 0.934783 | 0.977273 | 0.955556 |
| 3 | naive_bayes_gaussian | 0.636364 | 0.596774 | 0.840909 | 0.698113 |
| 4 | naive_bayes_multinomial | 0.693182 | 0.688889 | 0.704545 | 0.696629 |
| 5 | decision_tree | 0.840909 | 0.941176 | 0.727273 | 0.820513 |

In [46]:
```
path=r'D:\AD\dataset\test\class0\250.jpg'
predict_and_plot(path)
```




In [47]:
```
path=r'D:\AD\dataset\test\class1\354.jpg'
predict_and_plot(path)
```




## Conclusion

First, it is likely that such a method would be highly accurate, as it would be able to learn from a large number of labeled examples and make predictions based on patterns and features that are indicative of glaucoma. However, it is important to note that the accuracy of the method will depend on the quality and diversity of the training data, as well as the specific machine learning algorithm and model architecture used. It may also be necessary to validate the results on a separate test set to ensure that the model is not overfitting to the training data. Additionally, it is worth considering any ethical or privacy concerns that may arise from using machine learning for medical diagnosis, such as the potential for biased decision-making or the handling of sensitive patient data.