

**Institute of Engineering & Technology**

|  |
| --- |
| **FINAL REPORT**  **On**  **MALARIA DETECTION USING ML**  **Submitted by**  **INDAR PAWANI**  **171500143**  **SUSHANT RASTOGI**  **171500350**  **SOURABH BAINSLA**  **171500343**  **Department of Computer Engineering & Applications**  **Institute of Engineering & Technology**    **GLA University**  **Mathura- 281406, INDIA**  **2019** |



**Department of computer Engineering and Applications**

**GLA University, Mathura**

**17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,**

**Mathura – 281406**

**Declaration**

We hereby declare that the work which is being presented in the Mini project “Malaria Detection using ML” in fulfilment of the requirements for project viva voce, is an authentic record of my own work carried under the supervision of our mentor Mr. Pankaj Sharma Sir.

Indar Pawani (171500143)

Sushant Rastogi (171500350)

Sourabh Bainsla (171500343)

**ACKNOWLEDGEMENT**

It is indeed with a great pleasure and immense sense of gratitude that we acknowledge the help of these individuals. We are highly indebted to our Mentor Mr. Pankaj Sharma for the facilities provided to accomplish this mini project.

We would like to thank our mentor for this constructive criticism throughout our project. We extremely grateful to our Departmental staff members, Lab technicians and Non-teaching staff members for their extreme help throughout our project.

Finally, we express our heartfelt thanks to all of our professors who is helping us in the completion of this project.

**ABSTRACT**

Malaria is mosquito-borne blood disease caused by parasites of the genus Plasmodium. Conventional diagnostic tool for malaria is the examination of stained blood cell of patient in microscope.

The blood to be tested is placed in a slide and is observed under a microscope to count the number of infected RBC. An expert technician is involved in the examination of the slide with intense visual and mental concentration. This is tiresome and time consuming process.

We construct a new image processing system for detection and quantification of plasmodium parasites in blood smear slide, later we develop Machine Learning algorithm to learn, detect and determine the types of infected cells according to its features.

**INDEX**

1. INTRODUCTION

1.1 General Introduction

1.2 Problem Introduction

1.3 Objective

2. REQUIREMENTS SPECIFICATION

2.1 Software requirements

2.2 Install libraries

3. METHODOLOGY

4. IMPLEMENTATION DETAILS

5. RBC IMAGE DATA SET

6. CODE

7. CONTRIBUTION SUMMARY

8. REFRENCE

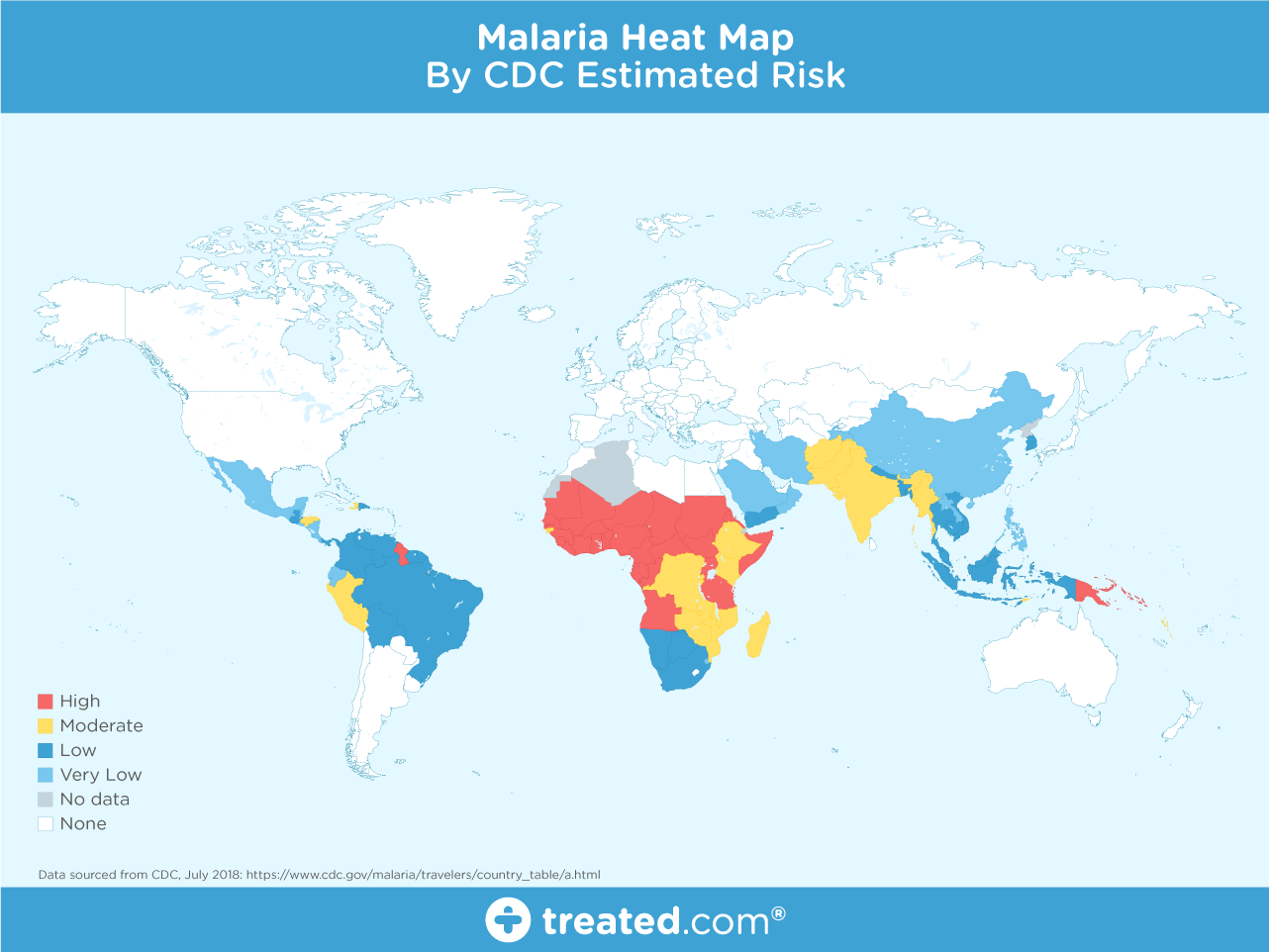
**1. INTRODUCTION**

**1.1 General Introduction:**

Malaria is a mosquito-borne life threatening disease caused by Plasmodium parasite. Globally, an estimated 3.2 billion people are at high risk (>1 in 1000 chance of getting malaria in a year). According to the report, there were 212 million new cases of malaria worldwide in 2015 (range 148–304 million). The WHO African Region accounted for most global cases of malaria (90%), followed by the South-East Asia Region (7%) and the Eastern Mediterranean Region (2%). Visual detection and recognition of Plasmodium in RBC is possible via chemical process. The staining process somewhat colorizes the RBCs but highlight Plasmodium, WBCs and platelets. The detection of Plasmodium requires detection of the stained objects. However, we need to analyzed stained objects further to determine if they are parasites or not to prevent false diagnosis. Several methods exist for malaria detection.

Malaria parasite (MP) in blood sample can be identified by using image segmentation and feature extraction using minimum distance classifier. Based on Image Acquisition, Image Preprocessing, Image Smoothing, Thresholding and Dilation image segmentation is done. Feature extraction uses two phases in architectural model: 1) Training Phase and 2) Recognition Phase which helps to recognize the MP.

In this work, we focus 1)automated detection and quantification of malaria detection, 2) strategy to determine infected image using machine learning 3) discuss to improve the predictive value for detection of infected cells .

****

**1.2 Problem Introduction:**

The main problem we are dealing nowadays regarding health is the detection of diseases before its worst phase. Some diseases shows its symptoms when it reaches or tend to reach its worst phase. Sometime the worst phase of any disease can cause death also.

In this project we are concerned about this problem and want to reduce it by developing a algorithm which works for detecting Malaria by using image processing technique.

**1.3 Objective:**

The main objective regarding this project is to reduce the time taken for detecting Malaria. Malaria symtoms could be observe before the last stage but in some cases it could not be observed and for detecting it we had to go through the big procedure.

The procedure for detecting Malaria is both time taking and costly. So for avoiding it we can use this algorithm, The requirement for this algorithm to be used is just a picture of the blood cell and we can detect it in seconds whether the person is infected or not

**2. REQUIREMENT SPECIFICATION**

**2.1Software Requirements:**

Python IDE-**Python** is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [general-purpose](https://en.wikipedia.org/wiki/General-purpose_programming_language) [programming language](https://en.wikipedia.org/wiki/Programming_language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991, Python's design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its notable use of [significant whitespace](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) and [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is [dynamically typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigms), including [structured](https://en.wikipedia.org/wiki/Structured_programming) (particularly, [procedural](https://en.wikipedia.org/wiki/Procedural_programming),) object-oriented, and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). Python is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

* 1. **Install libraries:**
* Pandas-. **pandas** is a [software library](https://en.wikipedia.org/wiki/Software_library) written for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)) for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and [time series](https://en.wikipedia.org/wiki/Time_series). It is [free software](https://en.wikipedia.org/wiki/Free_software) released under the [three-clause BSD license](https://en.wikipedia.org/wiki/3-clause_BSD_license). The name is derived from the term "[panel data](https://en.wikipedia.org/wiki/Panel_data)", an [econometrics](https://en.wikipedia.org/wiki/Econometrics) term for data sets that include observations over multiple time periods for the same individuals.
* Scikit-learn-**Scikit-learn** (formerly **scikits.learn** and also known as **sklearn**) is a [free software](https://en.wikipedia.org/wiki/Free_software) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) [programming language](https://en.wikipedia.org/wiki/Programming_language).[[3]](https://en.wikipedia.org/wiki/Scikit-learn#cite_note-jmlr-3) It features various [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) algorithms including [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine), [random forests](https://en.wikipedia.org/wiki/Random_forests), [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting), [*k*-means](https://en.wikipedia.org/wiki/K-means_clustering) and [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN), and is designed to interoperate with the Python numerical and scientific libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and [SciPy](https://en.wikipedia.org/wiki/SciPy).
* Joblib-Joblib is a set of tools to provide **lightweight pipelining in Python**. In particular:

1. transparent disk-caching of functions and lazy re-evaluation (memoize pattern)
2. easy simple parallel computing

Joblib is optimized to be **fast** and **robust** on large data in particular and has specific optimizations for *numpy* arrays. It is **BSD-licensed**.

* Opencv-**OpenCV** (*Open source computer vision*) is a [library of programming functions](https://en.wikipedia.org/wiki/Library_(computing)) mainly aimed at real-time [computer vision](https://en.wikipedia.org/wiki/Computer_vision). Originally developed by [Intel](https://en.wikipedia.org/wiki/Intel_Corporation), it was later supported by [Willow Garage](https://en.wikipedia.org/wiki/Willow_Garage) then Itseez (which was later acquired by Intel). The library is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) and free for use under the [open-source](https://en.wikipedia.org/wiki/Open-source_software) [BSD license](https://en.wikipedia.org/wiki/BSD_license).

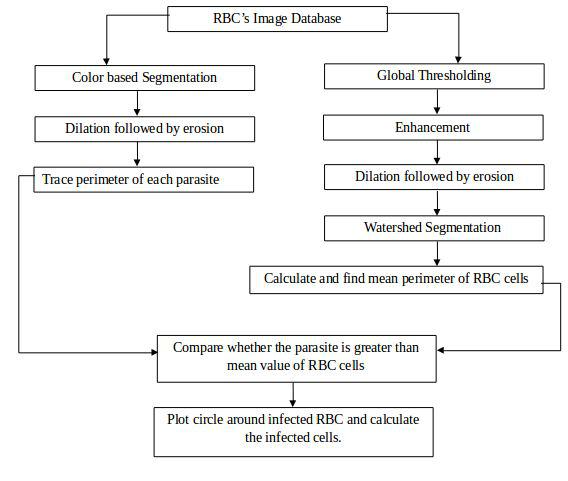
OpenCV supports some models from [deep learning](https://en.wikipedia.org/wiki/Deep_learning) frameworks like [TensorFlow](https://en.wikipedia.org/wiki/TensorFlow), [Torch](https://en.wikipedia.org/wiki/Torch_(machine_learning)), [PyTorch](https://en.wikipedia.org/wiki/PyTorch) (after converting to an ONNX model) and [Caffe](https://en.wikipedia.org/wiki/Caffe_(software)) according to a defined list of supported layers.[[3]](https://en.wikipedia.org/wiki/OpenCV#cite_note-3). It promotes OpenVisionCapsules. [[4]](https://en.wikipedia.org/wiki/OpenCV#cite_note-4), which is a portable format, compatible with all other formats

**3. METHODOLOGY**

Our data consists of heterogeneous datasets, we have lot of variability in the images. So it might be easy to develop an algorithm that works for particular image but it is much different challenge to find an algorithm that works across the board of heterogeneous datasets.There are couple of approaches we can take we can try an algorithm that is robust to the heterogeneity or we can cached the data into more homogeneity state.

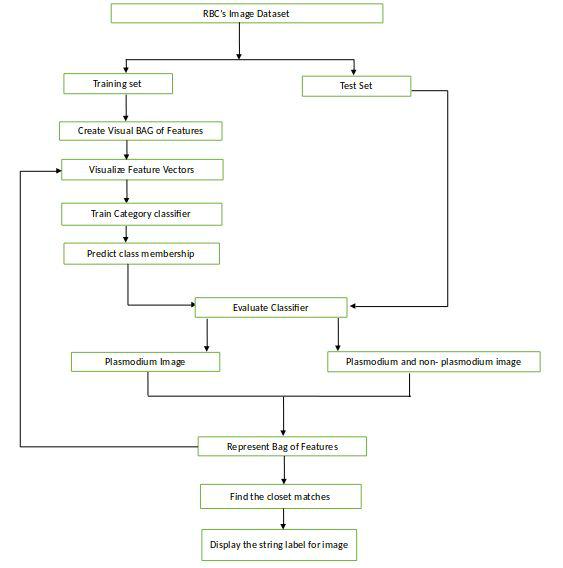
**4. IMPLEMENTATION DETAILS**

Using malaria infected images which is of less noisy and devoid of artifact this research can be done. The method used for the counting malaria infected RBCs is shown in Figure

****

Instead of finding how many cells are infected we will find what types of infection it is using machine learning.

A machine learning algorithm takes examples of inputs and outputs associated with a task and produces a program that can automatically differentiate them. In this research we are not going to discuss about machine learning but we will show the architecture we used for this research.



**5. RBC IMAGE DATA SET**-

PARACITIZED UNINFECTED

C:\Users\jeetu\AppData\Local\Microsoft\Windows\INetCache\Content.Word\C33P1thinF_IMG_20150619_114756a_cell_179.png C:\Users\jeetu\AppData\Local\Microsoft\Windows\INetCache\Content.Word\C1_thinF_IMG_20150604_104722_cell_15.png

C:\Users\jeetu\AppData\Local\Microsoft\Windows\INetCache\Content.Word\C189P150ThinF_IMG_20151203_142224_cell_84.png C:\Users\jeetu\AppData\Local\Microsoft\Windows\INetCache\Content.Word\C1_thinF_IMG_20150604_104722_cell_66.png

C:\Users\jeetu\AppData\Local\Microsoft\Windows\INetCache\Content.Word\C33P1thinF_IMG_20150619_114756a_cell_181.png C:\Users\jeetu\AppData\Local\Microsoft\Windows\INetCache\Content.Word\C1_thinF_IMG_20150604_104722_cell_73.png

C:\Users\jeetu\AppData\Local\Microsoft\Windows\INetCache\Content.Word\C33P1thinF_IMG_20150619_114756a_cell_182.png C:\Users\jeetu\AppData\Local\Microsoft\Windows\INetCache\Content.Word\C1_thinF_IMG_20150604_104722_cell_79.png

**6. CODE**

Run the FIRST CODE to read cell images and generate the CSV file for training and testing data

|  |
| --- |
|  |
|  | import numpy as np |
|  | import csv |
|  | import glob |
|  |  |
|  | label = "Parasitized" |
|  | dirList = glob.glob("cell\_images/"+label+"/\*.png") |
|  | file = open("csv/dataset.csv","a") |
|  |  |
|  | for img\_path in dirList: |
|  |  |
|  | im = cv2.imread(img\_path) |
|  |  |
|  | im = cv2.GaussianBlur(im,(5,5),2) |
|  |  |
|  |  |
|  |  |
|  | im\_gray = cv2.cvtColor(im,cv2.COLOR\_BGR2GRAY) |
|  |  |
|  | ret,thresh = cv2.threshold(im\_gray,127,255,0) |
|  | contours,\_ = cv2.findContours(thresh,1,2) |
|  |  |
|  | for contour in contours: |
|  | cv2.drawContours(im\_gray, contours, -1, (0,255,0), 3) |
|  |  |
|  |  |
|  | cv2.imshow("window",im\_gray) |
|  |  |
|  | break |
|  |  |
|  |  |
|  | file.write(label) |
|  | file.write(",") |
|  |  |
|  | for i in range(5): |
|  | try: |
|  | area = cv2.contourArea(contours[i]) |
|  | file.write(str(area)) |
|  | except: |
|  | file.write("0") |
|  |  |
|  | file.write(",") |
|  |  |
|  |  |
|  | file.write("\n") |
|  |  |
|  |  |
|  | cv2.waitKey(19000)  After the FIRST CODE you can see that the CSV file has been created in the CSV folder  So, as we have our data set prepared lets go head and built a classifier using scikit learn   |  | | --- | |  | |  | |  | from sklearn.model\_selection import train\_test\_split | |  | from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier | |  | from sklearn import metrics | |  | import joblib | |  |  | |  | ##Step1: Load Dataset | |  |  | |  | dataframe = pd.read\_csv("csv/dataset.csv") | |  | #print(dataframe.head()) | |  |  | |  | #Step2: Split into training and test data | |  | x = dataframe.drop(["Label"],axis=1) | |  | y = dataframe["Label"] | |  | x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=42) | |  |  | |  | ##Step4: Build a model | |  |  | |  | model = RandomForestClassifier(n\_estimators=100,max\_depth=5) | |  | model.fit(x\_train,y\_train) | |  | joblib.dump(model,"rf\_malaria\_100\_5") | |  |  | |  |  | |  | ##Step5: Make predictions and get classification report | |  |  | |  | predictions = model.predict(x\_test) | |  |  | |  | print(metrics.classification\_report(predictions,y\_test)) | |  | print(model.score(x\_test,y\_test)) | |  |  | |

OUTPUT FORMAT

Precision recall f1-score support

Parasitized 0.90 0.90 0.90 2773

Uninfected 0.90 0.89 0.90 2739

Micro avg 0.90 0.90 0.90 5512

Macro avg 0.90 0.90 0.90 5512

Weighted avg 0.90 0.90 0.90 5512

(Finished in 12.6s)

As you can see that we are getting average precision and recall of about 0.9 which is not state of the hour but not bad ether. The precision over here will give you sense of how much error is present in your model and recall will tell you how many times you are getting error. There is usually a tradeoff between these two so something like f1-score which is the harmonic mean of these two gives a better sense of how well your model is performing.

**7. CONTRIBUTION SUMMARY**

Our group consists of three people and each person has contributed equally according to their skills.

Indar Pawani, Sushant Rastogi and Sourabh bainsla all three are working together for algorithm using feature extraction technique called contour detection and then using RandomForest Classifier

**8. Reference**

* Data was collected by a hospital in Bangladesh and made public by researcher at NIH
* Researchgate.net
* Sciencedirect.com
* Wikipedia.com