

IMAGE CLASSIFICATION OF ABNORMAL RED BLOOD CELLS

Major project report submitted in partial fulfillment of the requirements for the
award of the degree of

BACHELOR OF TECHNOLOGY

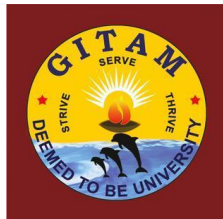
IN

COMPUTER SCIENCE AND ENGINEERING

Under the esteemed guidance of

Name of the Project Guide

Anagani Navya Sree (Assistant Professor)



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM (Deemed to be University)

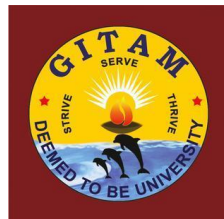
VISAKHAPATNAM

MARCH 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM (Deemed to be University)



DECLARATION

I, hereby declare that the project review entitled " Image Classification of Abnormal Red Blood Cells" is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering.

The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 31-03-2022

Registration NO:

121810303013

121810303029

121810303052

121810303059

Name

Ch. Sadhwika

M. Gowtham

G. Ramya Devi

P. Sai prakash

Signature

Ch.Sadhwika

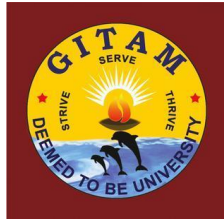
M.Gowtham

G.Ramya

P.Saiprakash

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM INSTITUTE OF TECHNOLOGY GITAM

(Deemed to be University)



CERTIFICATE

This is to certify that the project report entitled " Image Classification Of Abnormal Red Blood Cells " is a bonafide record of work carried out by Ch. Sai Lakshmi Sadhwika (121810303013), G. Ramya Devi(121810303052), M. Goutham(121810303029), P. Sai Prakash(121810303059) students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

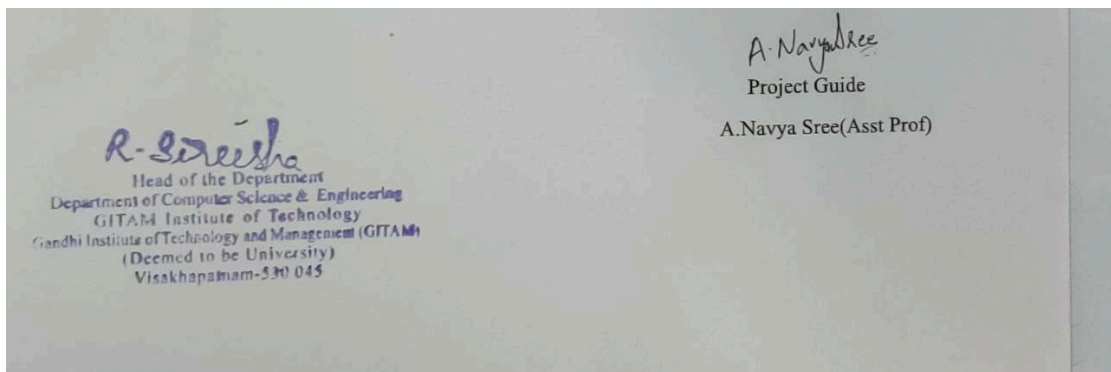


TABLE OF CONTENTS

S.no	Topic	Pg.no
	Abstract	5
1.	Introduction	6-11
2.	Literature Survey	12
3.	Problem analysis	13-36
4.	System design	37-46
5.	Implementation	47-54
6.	Testing	55-61
7.	Results and discussion	62-68
8.	Conclusion	69
9.	References	70-71

ABSTRACT

The study aims to increase the number of abnormal red blood cells that can be detected using image processing. This study used Random forest Algorithm as a machine learning algorithm in classifying. As a result, the system detected and classified a total of ten abnormal red blood cells. Images used in the system came from hospitals' past patients. In addition, a camera is used to capture the slides. The image was then inserted into the program. The system processed and classified the image. In effect, the results show the name of the abnormal red blood cells detected in the image within the system including the soft copy of the list.

1. INTRODUCTION

Blood is the life-maintaining fluid that flows through the heart, arteries, and other organs and it circulates through the whole body. Moreover, a blood is made up of many parts, mostly the red blood cells, white blood cells, platelets and plasma. Abnormalities of red blood cells vary through size or anisocytosis, through shape or poikilocytosis, in color and even through the presence of inclusion bodies. Detecting these irregularities in the shapes of the red blood cells is significant to one's health as it can determine whether the blood is healthy or not. Medical technicians, Pathologists and Hematologists usually used a manual microscopic method, to classify abnormal shapes of red blood cells. This methodology somehow is difficult and prone to human error. Thus, classifying the abnormal red blood cells using image processing is created using the high technologies. Mohammad Syahputra Et. Al (2017) said that

Morphological examination of peripheral blood smears done manually is less efficient and the shapes of the abnormal red blood cells found is not always the same for every analyst because of precision factor, concentration, and lack of knowledge. Radial Basis Function Network was used together with several stages which included input image, pre-processing, and feature extraction. Research result shows that by using this method, the accuracy to classify abnormal red blood cell types is 83.3%.

The objective of this study is to create a system that can classify 10 abnormal red blood cells and to know the reliability rate of classification of each abnormal red blood cell. Previous studies are usually limited to two to four abnormal red blood cells. Thus, the proponents aimed to create a maximized system. The proponents proposed a system that automatically classifies ten (10) abnormal red blood cells (see Figure 1), the spherocyte, codocyte, stomatocyte, ovalocyte, elliptocyte, degmacyte, drepanocyte, dacrocytes, acanthocyte and echinocyte using the Decision-Tree Algorithm. The algorithm is commonly used in classification and regression analysis. A decision tree is a simple representation for classifying an example that splits on its nodes. It uses a question on an attribute and splits the node that results to a branch or to end up with an output. The idea of the whole system of the decision tree is to divide the data set into smaller data set based on the descriptive features or the attributes until it reaches a specific abnormal red blood cell.

1.1MOTIVATION

Abnormalities of red blood cells vary through size or anisocytosis, through shape or poikilocytosis, in color and even through the presence of inclusion bodies. Detecting these irregularities in the shapes of the red blood cells is significant to one's health as it can determine whether the blood is healthy or not. Medical technicians, Pathologists and Hematologists usually used a manual microscopic method, to classify abnormal shapes of red blood cells. This methodology somehow is difficult and prone to human error..

1.2 Existing System

- Sickle-cell anemia is one of the most important types of anemia. This paper presents an algorithm for detecting blood cells characteristic of sickle-cell anemia. First, I discuss the construction of an algorithm that can be used to detect and count benign or distorted red blood cells (RBCs) in a microscopic colored image, even if those cells are hidden or overlapped. Second, I explain the process for checking and analyzing the constructed RBC data by applying two important techniques in data mining: the neural network (NN) and the decision tree. I then review experiments demonstrating that these models show high accuracy when predicting the counts of benign or distorted cells. In these experiments, the algorithm has segmented around 99.98% of all input cells, helping to improve the diagnosis of sickle-cell anemia. The NN has shown a 96.9% agreement with the algorithm's prediction outcomes, and the classification and regression tree has achieved 92.9%.

1.2.1 Limitations of existing system

- It required a high level of knowledgeable person.

1.3 Objectives

The objective of project is to Abnormal Red Blood Cells Using Decision Tree Algorithm

1.4 Applications

It can be used in Hospitals

1.5 STRUCTURE OF PROJECT (SYSTEM ANALYSIS)

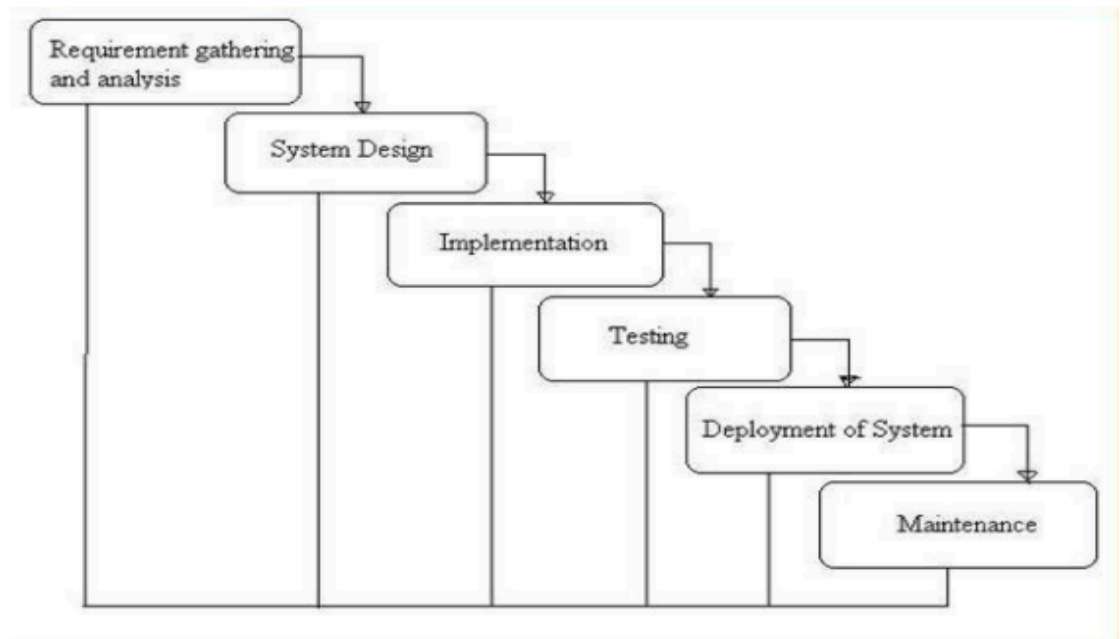


Fig: 1 Project SDLC

- Project Requisites Accumulating and Analysis
- Application System Design
- Practical Implementation
- Manual Testing of My Application
- Application Deployment of System
- Maintenance of the Project

1.6.1 REQUISITES ACCUMULATING AND ANALYSIS

It's the first and foremost stage of the any project as our is a an academic leave for requisites amassing we followed of IEEE Journals and Amassed so many IEEE Relegated papers and final culled a Paper designated "Individual web revisitation by setting and substance importance input and for analysis stage we took referees from the paper and did literature survey of some papers and amassed all the Requisites of the project in this stage

1.6.2 SYSTEM DESIGN

In System Design has divided into three types like GUI Designing, UML Designing with avails in development of project in facile way with different actor and its utilizer case by utilizer case diagram, flow of the project utilizing sequence, Class diagram gives information about different

class in the project with methods that have to be utilized in the project if comes to our project our UML Will utilizable in this way The third and post import for the project in system design is Database design where we endeavor to design database predicated on the number of modules in our project

1.6.3 IMPLEMENTATION

The Implementation is Phase where we endeavor to give the practical output of the work done in designing stage and most of Coding in Business logic lay comes into action in this stage its main and crucial part of the project

1.6.4 TESTING UNIT TESTING

It is done by the developer itself in every stage of the project and fine-tuning the bug and module predicated additionally done by the developer only here we are going to solve all the runtime errors

MANUAL TESTING

As our Project is academic Leave, we can do any automatic testing so we follow manual testing by endeavor and error methods

1.6.4 DEPLOYMENT OF SYSTEM AND MAINTENANCE

Once the project is total yare, we will come to deployment of client system in genuinely world as its academic leave we did deployment i our college lab only with all need Software's with having Windows OS .

The Maintenance of our Project is one-time process only

1.7 FUNCTIONAL REQUIREMENTS

1.Data Collection

2.Data Preprocessing

3.Training And Testing

4.Modiling

5.Predicting

1.8 NON FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a

software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

1.8.1 EXAMPLES OF NON-FUNCTIONAL REQUIREMENTS

Here, are some examples of non-functional requirement:

1.8.1.1 Users must upload dataset

1.8.1.2 The software should be portable. So moving from one OS to other OS does not create any problem.

1.8.1.3 Privacy of information, the export of restricted technologies, intellectual property rights, etc. should be audited.

1.8.2 ADVANTAGES OF NON-FUNCTIONAL REQUIREMENT

Benefits/pros of Non-functional testing are:

- The nonfunctional requirements ensure the software system follow legal and compliance rules.
- They ensure the reliability, availability, and performance of the software system
- They ensure good user experience and ease of operating the software.
- They help in formulating security policy of the software system.

1.8.3 DISADVANTAGES OF NON-FUNCTIONAL REQUIREMENT

Cons/drawbacks of Non-functional requirement are:

- Non functional requirement may affect the various high-level software subsystem
- They require special consideration during the software architecture/high-level design phase which increases costs.
- Their implementation does not usually map to the specific software sub-system,
- It is tough to modify non-functional ones once you pass the architecture phase.

1.8.4 KEY LEARNING

The character of the time period, the length of road, the weather, the bus speed and the rate of road usage are adopted as input vectors in Support Vector machine

2. LITERATURE SURVEY

2.1 Pooja Tukaram Dalvi and Nagaraj Vernekar (2016). Computer Aided Detection of Abnormal Red Blood Cells

Red blood cell classification and counting plays a very important role in detecting diseases like iron deficiency anemia, vitamin B12 deficiency anemia etc. In this research we intend to develop a standalone application that can classify the red blood cells into four abnormal types namely elliptocytes, echinocytes, tear drop cells and macrocytes. We will also provide the total red blood cell count. Thirteen Geometric features have been used to classify the red blood cells into the four abnormal types. We have used two data mining classifiers namely Artificial Neural Network and Decision Tree Classifier and we have compared the results of the two classifiers with respect to accuracy in classifying the red blood cells. The proposed method exhibits an accuracy of 95.27% for detecting elliptocytes, 96.06% for echinocytes, 85.82% for tear drop cells 85.82% for macrocytes and 89.76% for normal red blood cells. Red blood cells (RBCs) are the most abundant cells present in the human body. Normal RBCs are biconcave and disk shaped. Any abnormality in the shape of RBC indicates the presence of disease. The number of RBCs also plays an important role in detecting anemia. A decrease in the number of RBCs and an abnormality in RBCs shape is a clear indicator of presence of blood related disorders. Presence of tear drop cells, echinocytes, elliptocytes, macrocytes indicate presence of diseases like myelofibrosis, severe iron deficiency, uremia, hereditary elliptocytosis, haemolytic anemia etc [14]. Anemia and blood related disorders are prevalent in almost 24.5% of the world population.

2.2 Vishwas Sharma Et. Al (2010). Detection of Sick Cell Anemia and Thalassaemia Causing Abnormalities in Thin Smear of Human Blood Sample Using Image Processing.

Blood is a connective tissue in which Red blood cells function to transport oxygen and it is normally in disk shape. The inherited disorder of blood includes hemoglobinopathies which are a major public health problem in India. Sick cell disease refers to a group of genetic disorders characterized by presence of sickle hemoglobin, anemia, acute and chronic tissue injury to blockage of blood flow by abnormally shaped red cells. Sick cell disease is Sick cell anemia. It is a disorder in which the body makes sickle shaped red blood cells. "Sickle-shaped means the red blood cells are crescent shaped. Sick cell anemia is also a serious disorder problem in Chhattisgarh state. It is highly prevalent among scheduled caste, scheduled tribe and other backward classes. In Chhattisgarh the highest percentage of sickle cell diseases are found in Sahu, Mahar, Gond, Devangan, Kurmi and Halba etc. This paper proposed a method to recognize the sickle shaped red blood cells present in the blood smear by using fractal dimension. Fractal Dimension is used to recognize the shape of the red blood cells and segmentation the sickle shaped red blood cells for shape analysis to find the percentage of sickle cell anemia. Results exhibit the future aspect of the technique, which overcomes traditional shape recognition and analysis methods found in various literatures. sickle cell disease (SCD), also known as sickle cell anemia, is a serious disease in which the body makes an altered form of hemoglobin, the protein in red blood cells that carries oxygen throughout the body. This genetic alteration causes the body to produce abnormal sickle- or crescent-shaped red blood cells.

3. PROBLEM ANALYSIS

3.1 EXISTING APPROACH:

Sickle-cell anemia is one of the most important types of anemia. This paper presents an algorithm for detecting blood cells characteristic of sickle-cell anemia. First, I discuss the construction of an algorithm that can be used to detect and count benign or distorted red blood cells (RBCs) in a microscopic colored image, even if those cells are hidden or overlapped. Second, I explain the process for checking and analyzing the constructed RBC data by applying two important techniques in data mining: the neural network (NN) and the decision tree. We then review experiments demonstrating that these models show high accuracy when predicting the counts of benign or distorted cells. In these experiments, the algorithm has segmented around 99.98% of all input cells, helping to improve the diagnosis of sickle-cell anemia. The NN has shown a 96.9% agreement with the algorithm's prediction outcomes, and the classification and regression tree has achieved 92.9%..

3.1.1 Drawbacks

It required a high level of knowledge.

3.2 Proposed System

The proponents proposed a system that automatically classifies ten (10) abnormal red blood cells the spherocyte, codocyte, stomatocyte, ovalocyte, elliptocyte, degmacyte, drepanocyte, dacrococytes, acanthocyte and echinocyte using the Decision-Tree Algorithm. The algorithm is commonly used in classification and regression analysis. A decision tree is a simple representation for classifying an example that splits on its nodes. It uses a question on an attribute and splits the node that results to a branch or to end up with an output. The idea of the whole system of the decision tree is to divide the data set into smaller data set based on the descriptive features or the attributes until it reaches a specific abnormal red blood cell.

3.2.1 Advantages

By using this we can get results more efficient and easy identification

3.3 Software And Hardware Requirements

SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

- **Python idle 3.7 version (or)**
- **Anaconda 3.7 (or)**
- **Jupyter (or)**
- **Google colab**

HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- **Operating system : windows, linux**
- **Processor : minimum intel i3**
- **Ram : minimum 4 gb**
- **Hard disk : minimum 250gb**

3.4 About Dataset

White blood cell data collected form kaggle

For this classification we take datasets. This dataset consists of nineteen columns and 401 records. Out of nineteen columns 18 columns consist of attributes and the last column consists of class labels.

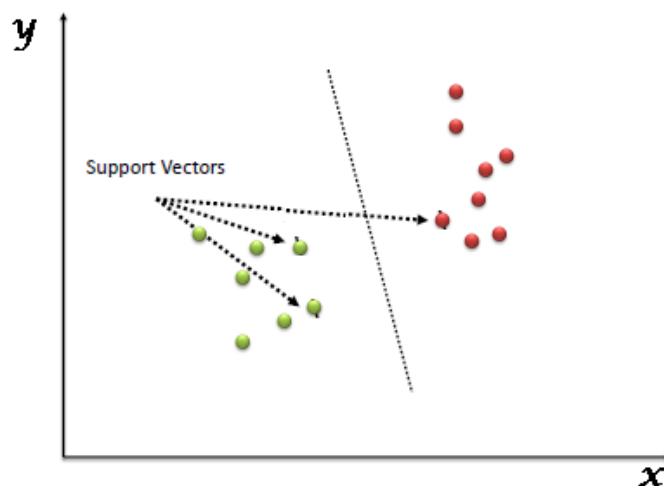
Contains 4 categories

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		area	perimeter	physiologic	physiologic	aspect_ratio	rectangularity	circularity	mean_r	mean_g	mean_b	stddev_r	stddev_g	stddev_b	contrast	correlation	inverse_diff	entropy	class		
2	0	1898	164.36753	52	48	1.0833333	1.3150685	14.234291	41.942431	84.950833	147.17181	80.386339	106.7005	24.650697	108.37764	0.9870923	0.4726736	8.3329637	2		
3	0	9441	395.74011	91	120	0.7583333	1.2958866	18.553517	78.756806	88.28472	133.45938	90.544628	91.329781	31.500503	125.9242	0.9872521	0.3930962	9.8928665	1		
4	0	9122.5	457.84062	120	120	1	1.5785147	22.978135	77.102292	92.618819	127.06556	88.450453	92.357954	32.701174	117.6612	0.9891558	0.3732655	10.242833	1		
5	0	3956.5	321.92388	91	91	1	2.0930115	26.193602	65.744514	73.586389	143.17743	91.034857	91.890791	30.218365	127.99939	0.9864418	0.4528215	9.0905226	1		
6	0	10613.5	462.52691	120	120	1	1.3567626	20.156512	89.694306	84.854375	131.62264	89.209763	85.16135	26.328184	79.496383	0.9919475	0.4218264	9.7086789	1		
7	0	6995.5	439.09545	120	120	1	2.0584662	27.561263	112.08104	161.80792	128.27396	101.29152	84.285629	23.526304	93.122613	0.9890815	0.2368452	11.418837	1		
8	0	2388.5	183.58073	61	53	1.1509434	1.3535692	14.110063	105.02222	170.52854	134.45222	105.77333	90.629696	22.044975	66.180372	0.9921181	0.2715913	10.771922	2		
9	0	7902	395.88225	94	120	0.7833333	1.4274867	19.833302	83.670625	105.70069	130.57507	94.399336	97.751209	27.457868	102.64524	0.9892609	0.3759926	9.9983959	1		
10	0	9909.5	456.66905	120	120	1	1.453151	21.04512	86.667222	129.31319	127.47708	90.475705	94.01066	25.624709	80.395819	0.9913746	0.3152286	10.603359	1		
11	0	10134.5	457.84062	120	120	1	1.420889	20.683099	106.77667	135.92507	120.44569	93.396362	88.465948	24.076729	82.676724	0.990807	0.2676453	11.314682	1		
12	0	6893.5	386.61017	120	92	1.3043478	1.6015087	21.682371	92.332639	144.02681	123.19917	98.43325	93.540991	25.397663	82.276945	0.9911712	0.3040421	10.835416	1		
13	0	7443	445.5391	120	120	1	1.9347037	26.670038	76.70875	122.96167	129.82764	92.088999	99.004769	25.704018	82.450833	0.9911358	0.3589497	10.118868	1		
14	0	11186.5	460.18377	120	120	1	1.2872659	18.930773	89.196875	135.78965	120.7909	87.883146	95.426053	25.290434	78.338944	0.9925266	0.3135048	10.818588	3		
15	0	3131.5	222.40916	65	69	0.942029	1.432221	15.796211	75.967361	122.02111	132.40597	95.665868	100.98648	24.464689	84.720332	0.9905372	0.3556044	10.057279	1		
16	0	8991	508.08326	120	120	1	1.6016016	28.71189	81.748472	109.84181	130.16417	94.356647	96.914751	26.054995	90.908455	0.9904389	0.3697621	10.027625	1		
17	0	2417	189.68124	62	53	1.1698113	1.3959366	14.885797	34.951458	121.29236	133.23896	71.853726	114.52016	26.614098	65.694478	0.9940069	0.4808799	8.3230034	2		
18	0	8335	446.71068	120	120	1	1.7276545	23.941263	121.53688	151.54125	123.43222	97.17474	84.49639	23.6882	64.377069	0.9925346	0.2851587	11.35702	1		
19	0	5051.5	384.75231	94	120	0.7833333	2.2330001	29.305026	141.33264	109.02271	121.2491	100.43577	72.096783	21.425597	82.495968	0.9899837	0.23991	11.436506	1		
20	0	2421.5	184.75231	58	56	1.0357143	1.3413174	14.09598	75.943125	186.79889	133.43056	101.13677	83.847239	23.247321	69.645999	0.9926264	0.3020137	10.247152	2		
21	0	7569	443.19596	120	120	1	1.902497	25.950939	100.49715	144.53722	126.67667	98.055359	91.213455	24.840621	91.61755	0.9898001	0.2701582	11.171757	1		
22	0	7459.5	398.46804	120	95	1.2631579	1.5282526	21.285177	76.628264	135.1441	129.54993	84.378607	99.010166	25.15309	71.18834	0.9933797	0.309022	10.559417	3		
23	0	8281.5	446.12489	120	120	1	1.7388154	24.032774	88.026528	57.271319	140.08319	87.848494	78.376753	32.993884	95.17075	0.9906472	0.51322	8.2905663	1		
24	0	6235	386.50967	94	120	0.7833333	1.8091419	23.959859	50.820833	54.483056	132.41021	79.360954	82.978046	29.978837	97.69967	0.9918059	0.5542664	7.515777	1		
25	0	8903.5	450.81118	120	120	1	1.6173415	22.025936	69.781667	61.133819	129.56257	85.949917	81.204223	30.293477	96.673291	0.9912498	0.497282	8.3875554	1		
26	0	1844.5	165.78174	47	54	0.8703704	1.3759827	14.900291	24.818333	28.340208	175.52243	58.577524	68.361864	31.565642	73.002201	0.9902728	0.65752	6.5023225	2		
27	0	6751	396.2254	120	93	1.2903226	1.6530884	23.255009	71.999861	61.188958	157.46972	90.135123	82.367031	34.231786	66.662216	0.9928774	0.5019489	8.7611534	3		
28	0	5393.5	392.61017	95	120	0.7916667	2.1136553	28.579354	77.679028	82.864861	149.66743	96.738388	93.004413	32.407827	95.504275	0.988794	0.4495751	9.3931089	1		
29	0	3534.5	308.61017	53	120	0.4416667	1.7994059	26.945887	78.924514	77.643264	147.58632	98.182054	91.283814	33.313868	89.827256	0.9894881	0.4461676	9.2709254	1		
30	0	2676.5	196.40916	58	63	0.9206349	1.3652158	14.413061	37.49625	28.073333	167.61694	69.266866	57.727477	25.244363	31.869009	0.9948818	0.715822	5.2908089	2		

3.5 Algorithms

Support Vector Machine

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).



Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line).

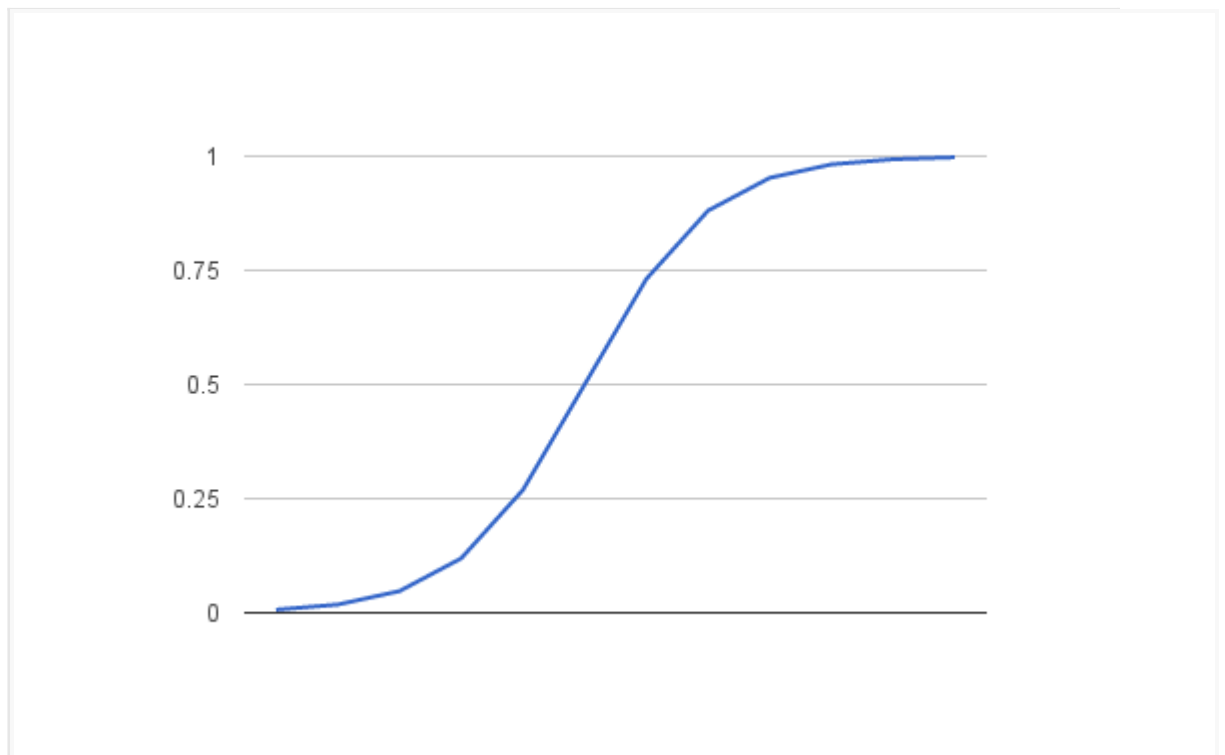
Logistic regression

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

$$1 / (1 + e^{-\text{value}})$$

Where e is the base of the natural logarithm (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.



Logistic Function

Now that we know what the logistic function is, let's see how it is used in logistic regression.

Naive Bayes Classifiers:

For binary (two-class) and multi-class classification issues, Naive Bayes is a classification algorithm. When stated with binary or categorical input values, the technique is the easiest to grasp.

Because the probabilities for each hypothesis are simplified to make their calculation tractable, it is known as naïve Bayes or stupid Bayes. Instead of trying to determine the values of each attribute value $P(d_1, d_2, d_3|h)$, they are considered to be conditionally independent given the goal value and computed as $P(d_1|h) * P(d_2|H)$, and so on.

This is a strong assumption that is unlikely to be true in real data, namely that the qualities do not interact. Nonetheless, the technique works very well on data that contradicts this premise.

Random Forest Algorithm

Random Forest is a well-known machine learning algorithm that uses the supervised learning method. In machine learning, it can be utilized for both classification and regression issues. It is based on ensemble learning, which is a method of integrating several classifiers to solve a complex problem and increase the model's performance.

"Random Forest is a classifier that contains a number of decision trees on various subsets of a given dataset and takes the average to enhance the predicted accuracy of that dataset," according to the name. Instead of relying on a single decision tree, the random forest collects the forecasts from each tree and predicts the final output based on the majority votes of predictions.

Decision tree Algorithm

Instances are classified using decision trees by sorting them along the tree from the root to a leaf node, which yields the classification. As indicated in the above diagram, an instance is categorized by starting at the root node of the tree, checking the attribute given by this node, and then progressing along the tree branch according to the attribute value. The subtree rooted at the new node is then processed in the same way.

AdaBoost Classifier

Yoav Freund and Robert Schapire proposed Ada-boost (Adaptive Boosting) in 1996 as one of the ensemble boosting classifiers. To improve the accuracy of classifiers, it mixes numerous classifiers. Iterative ensemble approach AdaBoost AdaBoost classifier creates a powerful classifier by merging many low-performing classifiers, resulting in a high-accuracy, high-performance classifier. The primary idea behind Adaboost is to train the data sample and set the weights of classifiers in each iteration to provide accurate predictions of uncommon observations. If the basic classifier takes weights from the training set, any machine learning method may be applied. Two requirements must be met by Adaboost:

1. The classifier should be trained interactively on various weighted training examples.
2. In each iteration, it tries to provide an excellent fit for these examples by minimizing training error.

3.6 SDLC

What is SDLC?

SDLC stands for Software Development Life Cycle. A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software.

SDLC is the process consisting of a series of planned activities to develop or alter the software products.

Benefits of the SDLC Process

The intent of a SDLC process is to help produce a product that is cost-efficient, effective, and of high quality. Once an application is created, the SDLC maps the proper deployment and decommissioning of the software once it becomes a legacy. The SDLC methodology usually contains the following stages: Analysis (requirements and design), construction, testing, release, and maintenance (response). Veracode makes it possible to integrate automated security testing into the SDLC process through use of its cloud based platform.

1. Requirements Gathering:

In this phase we gather all the requirements from the client, i.e. what are the client's expected input, output.

2. Analysis:

In this phase based upon the client requirements we prepare one documentation called “High Level Design Document”. It contains Abstract, Functional Requirements, Non Functional Requirements, Existing System, Proposed System, SRS.

3. Design:

It is difficult to understand the High Level Design Document for all the members, so to understand it easily we use “Low Level Design Document”. To design this document we use UML (Unified Modeling Language). In this we have Use case, Sequence, Collaboration.

4. Coding:

In this phase we develop the coding module by module. After developing all

the modules we integrate them.

5. Testing:

After developing we have to check whether client requirements are satisfied or not. If not, we are again going to develop.

6. Implementation:

In the testing phase if client requirements are satisfied, we go for implementation. i.e. we need to deploy the application on some server.

7. Maintenance:

After deployment, if at all any problems come from the client side; we are providing maintenance for that application.

3.7 FEASIBILITY STUDY

Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running systems. All systems are feasible if they have unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any additional hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned into an information system. That will meet the organization's operating requirements. Operational

feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Does the proposed equipment have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability

and security. The software and hardware requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

3.8 TECHNOLOGIES USED

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it to you.

Python concepts

If you're not interested in the hows and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

- Open source general-purpose language.
- Object Oriented, Procedural, Functional
- Easy to interface with C/ObjC/Java/Fortran
- Easy-ish to interface with C++ (via SWIG)
- Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include –

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.

- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs Static

Types Python is a dynamic-typed language. Many other languages are statically typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it,

your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn't matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it's more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types –

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of

strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated. Tuples can be thought of as **read-only** lists.

Python Dictionary

Python's dictionaries are a kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

Python libraries

1. Requests. The most famous http library written by kenneth reitz. It's a must have for every python developer.
2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won't use any other.
3. wxPython. A gui toolkit for python. I have primarily used it in place of tkinter. You will really love it.
4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.
5. SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.
6. BeautifulSoup. I know it's slow but this xml and html parsing library is very useful for beginners.
7. Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.
8. NumPy. How can we leave this very important library ? It provides some advanced math functionalities to python.
9. SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
10. matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.
11. Pygame. Which developer does not like to play games and develop them ? This library will help you achieve your goal of 2d game development.
12. Pyglet. A 3d animation and game creation engine. This is the engine in which the famous python port of minecraft was made

13. pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI's for my python scripts.
14. pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.
15. Scapy. A packet sniffer and analyzer for python made in python.
16. pywin32. A python library which provides some useful methods and classes for interacting with windows.
17. nltk. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.
18. nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.
19. SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.
20. IPython. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

Numpy

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

- Offers Matlab-ish capabilities within Python
- Fast array operations
- 2D arrays, multi-D arrays, linear algebra etc.

matplotlib

- High quality plotting library.

Python class and objects

These are the building blocks of OOP. class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most

programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early '90s, I realized functions were just like subroutines so there wasn't much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you'll realize that it's actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it's not mandatory to use objects in your code in a way that works best; maybe you don't need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you've already seen, Python can do just fine with functions. Unlike languages such as Java, you aren't tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code
Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here's a brief list of Python OOP ideas:

- The class statement creates a class object and gives it a name. This creates a new namespace.
- Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: Class Name Attribute.
- Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.
- Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

- Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.
- Using the term `self` identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it. To elaborate, by subclassing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components.

As we've seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python's object model and the object interfaces are more consistent to Python's built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python's built-in operation methods

Exceptions

I've talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify a program's flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn't exist, or when the program reaches a marker, such as the completion of a loop. Exceptions, by definition, don't occur very often; hence, they are the "exception to the

rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

- Accessing a non-existent dictionary key will raise a `KeyError` exception.
- Searching a list for a non-existent value will raise a `ValueError` exception.
- Calling a non-existent method will raise an `AttributeError` exception.
- Referencing a non-existent variable will raise a `NameError` exception.
- Mixing data types without coercion will raise a `TypeError` exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don't need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exception, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don't put any overhead on the code processing.

Because exceptions aren't supposed to happen very often, they aren't processed until they occur.

Exceptions can be thought of as a special form of the `if/elif` statements. You can realistically do the same thing with `if` blocks as you can with exceptions.

However, as already mentioned, exceptions aren't processed until they occur; `if` blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using `if` blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling `if` statements, it can be difficult to read, modify, and debug your

program.

User-Defined Exceptions

I won't spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won't have to do this very often but it's nice to have the option when necessary.

However, before making your own exceptions, make sure there isn't one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*; definitions from a module can be *imported* into other modules or into the *main* module.

Testing code

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. The entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

To carry out that specific task, the function might or might not need multiple inputs. When the task is carried out, the function can or can not return one or more values.

There are three types of functions in python: `help()`, `min()`, `print()`.

Python Namespace

Generally speaking, a namespace (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e. the naming of people in firstname and family name (surname).

An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames.

Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces.

(Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- **global names** of a module
- **local names** in a function or method invocation
- **built-in names**: this namespace contains built-in functions (e.g. `abs()`, `cmp()`, ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs The Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX : Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree – based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format templating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks

fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

Comparing web frameworks

There is also a repository called [compare python web framework](#) where the same web application is being coded with varying Python web frameworks, templating engines and object.

Web framework resources

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
- Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
- What is a web framework? is an in-depth explanation of what web frameworks are and their relation to web servers.
- Django vs Flask vs Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
- Python's web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
- What web frameworks do you use and why are they awesome? is a language agnostic Reddit discussion on web frameworks. It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.

- This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?". The votes aren't as important as the list of the many frameworks that are available to Python developers.

Web frameworks learning checklist

1. Choose a major Python web framework (Django or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the deployments section to make it accessible on the web.

Python-Database Communication

Connector/Python provides a `connect()` call used to establish connections to the MySQL server. The following sections describe the permitted arguments for `connect()` and describe how to use option files that supply additional arguments.

4. SYSTEM DESIGN

UML DIAGRAMS

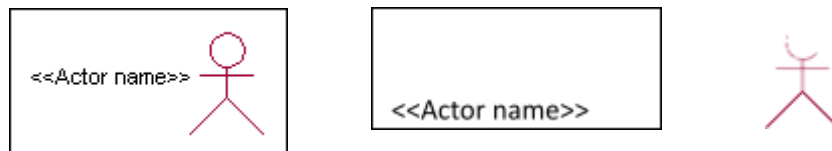
The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



Actor

An actor is someone or something that:

Interacts with or uses the system.

Provides input to and receives information from the system.

Is external to the system and has no control over the use cases. Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system. Questions to identify actors:
 - Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task

- Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
- Which external hardware or systems (if any) use the system to perform tasks?
- What problems does this application solve (that is, for whom)?
- And, finally, how do users use the system (use case)? What are they doing with the system?

The actors identified in this system are:

- System Administrator**
- Customer**
- Customer Care**

Identification of use cases:

Use Case: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

- Pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor Use cases provide a means to:
- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guidelines for identifying use cases:

- For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
- Name the use cases.
- Describe the use cases briefly by applying terms with which the user is familiar. This makes the description less ambiguous

Questions to identify use cases:

- What are the tasks of each actor?
- Will any actor create, store, change, remove or read information in the system?
- What use case will store, change, remove or read this information?
- Will any actor need to inform the system about sudden external changes?
- Does any actor need to inform about certain occurrences in the system?
- What use cases will support and maintain the system?

Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
 - Use case/actor interactions
 - Data needed by the use case
 - Normal sequence of events for the use case
 - Alternate or exceptional flows
- Construction of Use Case diagrams:

Use-case diagrams graphically depict system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may depict all or some of the use cases of a system.

A use-case diagram can contain:

- actors ("things" outside the system)
- use cases (system boundaries identifying what the system should do)

- Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Relationships in use cases:

1. **Communication:**

The communication relationship of an actor in a use case is shown by connecting the actor symbol to the use case symbol with a solid path. The actor is said to communicate with the use case.

2. **Uses:**

A Uses relationship between the use cases is shown by the generalization arrow from the use case.

3. **Extends:**

The extended relationship is used when we have one use case that is similar to another use case but does a bit more. In essence it is like a subclass.

SEQUENCE DIAGRAMS

A sequence diagram is a graphical view of a scenario that shows object interaction in a time- based sequence: what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

Object:

An object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

The object icon is similar to a class icon except that the name is underlined:

An object's concurrency is defined by the concurrency of its class.

Message:

A message is the communication carried between two objects that trigger an event. A message carries information from the source focus of control to the destination focus of control. The synchronization of a message can be modified through the message specification. Synchronization means a message where the sending object pauses to wait for results.

Link:

A link should exist between two objects, including class utilities, only if there is a relationship between their corresponding classes. The existence of a relationship between two classes symbolizes a path of communication between instances of the classes: one object may send messages to another. The link is depicted as a straight line between objects or objects and class instances in a collaboration diagram. If an object links to itself, use the loop version of the icon.

CLASS DIAGRAM:

Identification of analysis classes:

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items. There are 4 approaches for identifying classes:

- a. Noun phrase approach:
- b. Common class pattern approach.
- c. Use case Driven Sequence or Collaboration approach.
- d. Classes , Responsibilities and collaborators Approach

1. Noun Phrase Approach:

The guidelines for identifying the classes:

- Look for nouns and noun phrases in the use cases.
- Some classes are implicit or taken from general knowledge.
- All classes must make sense in the application domain; Avoid computer implementation classes – defer them to the design stage.
- Carefully choose and define the class names After identifying the classes we have to eliminate the following types of classes:

- Adjective classes.

2. Common class pattern approach:

The following are the patterns for finding the candidate classes:

- Concept class.
- Events class.
- Organization class
- Peoples class
- Places class
- Tangible things and devices class.

3. Use case driven approach:

We have to draw the sequence diagram or collaboration diagram. If there is a need for some classes to represent some functionality then add new classes which perform those functionalities.

4. CRC approach:

The process consists of the following steps:

- Identify classes' responsibilities (and identify the classes)
- Assign the responsibilities
- Identify the collaborators. Identification of responsibilities of each class:

The questions that should be answered to identify the attributes and methods of a class respectively are:

- a. What information about an object should we keep track of?
- b. What services must a class provide? Identification of relationships among the classes:

Three types of relationships among the objects are:

Association: How are objects associated?

Super-sub structure: How are objects organized into super classes and subclasses?

Aggregation: What is the composition of the complex classes?

Association:

The questions that will help us to identify the associations are:

- a. Is the class capable of fulfilling the required task by itself?
- b. If not, what does it need?
- c. From what other classes can it acquire what it needs? Guidelines for identifying the tentative associations:

- A dependency between two or more classes may be an association. Association often corresponds to a verb or prepositional phrase.

- A reference from one class to another is an association. Some associations are implicit or taken from general knowledge.

Some common association patterns are:

Location association like part of, next to, contained in..... Communication association like talk to, order to

We have to eliminate the unnecessary association like implementation associations, ternary or n- ary associations and derived associations.

Super-sub class relationships:

Super-sub class hierarchy is a relationship between classes where one class is the parent class of another class (derived class). This is based on inheritance.

Guidelines for identifying the super-sub relationship, a generalization are

1. Top-down:

Look for noun phrases composed of various adjectives in a class name. Avoid excessive refinement. Specialize only when the sub classes have significant behavior.

2. Bottom-up:

Look for classes with similar attributes or methods. Group them by moving the common attributes and methods to an abstract class. You may have to alter the definitions a bit.

3. Reusability:

Move the attributes and methods as high as possible in the hierarchy.

4. Multiple inheritances:

Avoid excessive use of multiple inheritances. One way of getting benefits of multiple inheritances is to inherit from the most appropriate class and add an object of another class as an attribute.

Aggregation or a-part-of relationship:

It represents the situation where a class consists of several component classes. A class that is composed of other classes doesn't behave like its parts. It behaves very differently. The major properties of this relationship are transitivity and anti symmetry. The **questions** whose answers will determine the distinction between the part and whole relationships are:

- Does the part class belong to the problem domain?
 - Is the part class within the system's responsibilities?
- There are three types of aggregation relationships. They are:

Assembly:

It is constructed from its parts and an assembly-part situation physically exists.

Container:

A physical whole encompasses but is not constructed from physical parts.

Collection member:

A conceptual whole encompasses parts that may be physical or conceptual.

USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

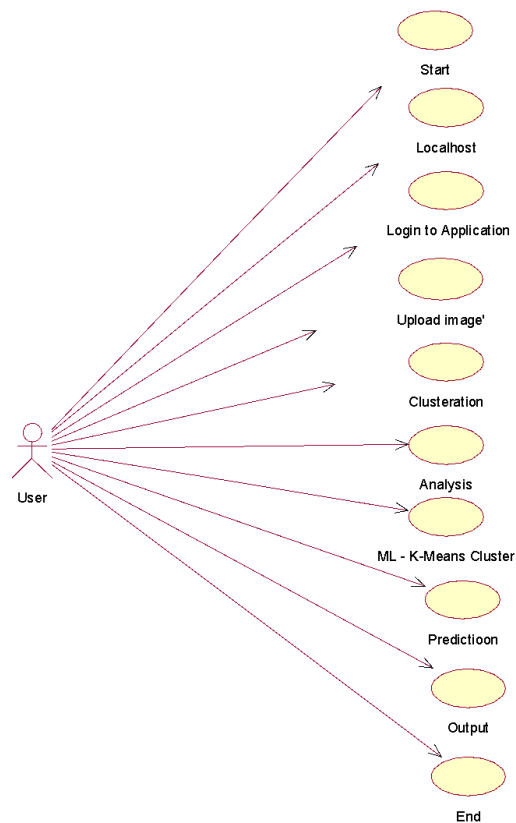


Fig 1: Use Case Diagram

CLASS DIAGRAM

In software engineering, 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 (or methods), and the relationships among the classes. It explains which class contains information.

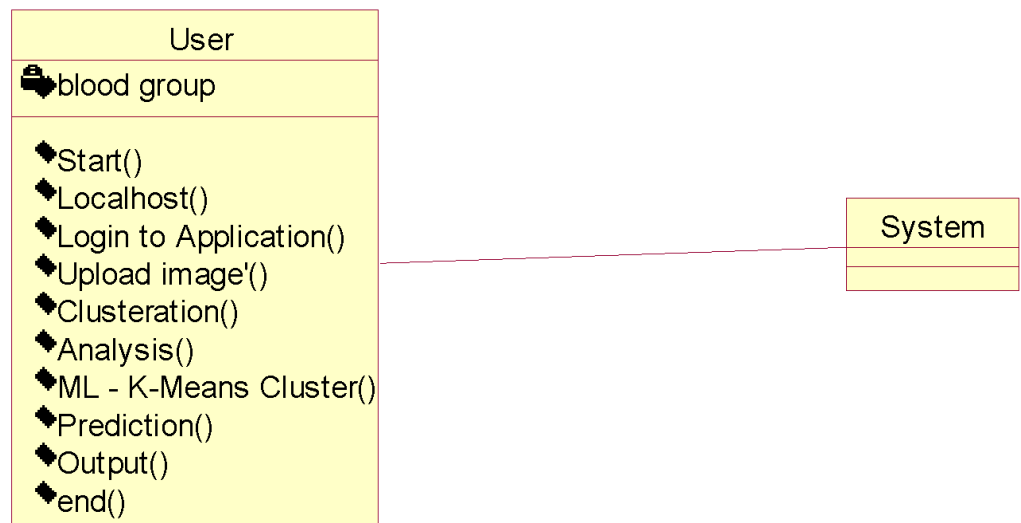


Fig 2:Class Diagram

SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

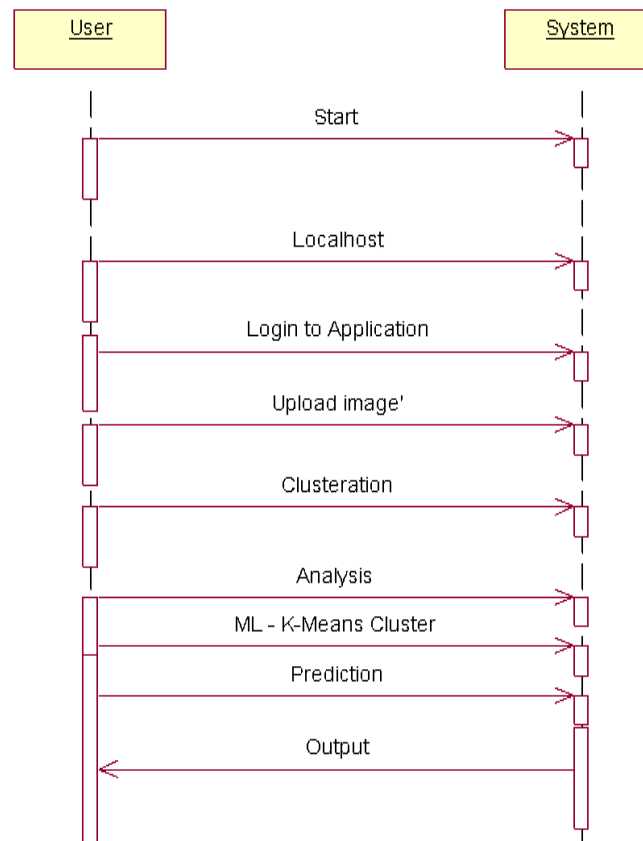


Fig 3: Sequence Diagram

5.IMPLEMENTATION

5.1 FLOW CHART:



5.2 Architecture



5.3 Code

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from imutils import paths
from tkinter.filedialog import askopenfilename

import numpy as np
import pandas as pd

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import
train_test_split, KFold, cross_val_score, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier,
AdaBoostClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import LinearSVC

import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
import seaborn as sns
import cv2
import mahotas as mt

import warnings
warnings.filterwarnings('ignore')

main = tkinter.Tk()
main.title("Blood Cell classification")
main.geometry("1300x1200")

class test:
```



```

def upload():
    global filename
    text.delete('1.0', END)
    filename = askopenfilename(initialdir = "Dataset")
    pathlabel.config(text=filename)
    text.insert(END, "Dataset loaded\n\n")

def csv():
    global data
    text.delete('1.0', END)
    data=pd.read_csv(filename)
    text.insert(END, "Top Five rows of dataset\n"+str(data.head())+"\n")
    text.insert(END, "Last Five rows of dataset\n"+str(data.tail()))
    data.drop('Unnamed: 0',axis=1,inplace=True)

def splitdataset():
    text.delete('1.0', END)
    print(data.columns)
    X = data.iloc[:, :-1]
    Y = data.iloc[:, -1]
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2,
random_state = 7)
    text.insert(END, "\nTrain & Test Model Generated\n\n")
    text.insert(END, "Total Dataset Size : "+str(len(data))+"\n")
    text.insert(END, "Split Training Size : "+str(len(X_train))+"\n")
    text.insert(END, "Split Test Size : "+str(len(X_test))+"\n")
    return X_train, X_test, y_train, y_test

def MLmodels():
    global model_final
    X_train, X_test, y_train, y_test=test.splitdataset()
    text.delete('1.0', END)
    models=[]
    models.append(('SVM-Linear',LinearSVC()))
    models.append(('RandomForest',RandomForestClassifier()))
    models.append(('DecisionTree',DecisionTreeClassifier()))
    models.append(('Adaboost',AdaBoostClassifier()))
    models.append(('Bagging',BaggingClassifier()))
    results=[]
    names=[]
    predicted_values=[]
    text.insert(END, "Machine Learning Classification Models\n")
    text.insert(END, "Predicted values,Accuracy Scores and S.D values from ML
Classifiers\n\n")
    for name,model in models:
        kfold=KFold(n_splits=10,random_state=7)

cv_results=cross_val_score(model,X_train,y_train,cv=kfold,scoring='accuracy')
    model.fit(X_train,y_train)

```

```

        predicted=model.predict(X_test)

        predicted_values.append(predicted)
        results.append(cv_results.mean()*100)
        names.append(name)
        text.insert(END, "\n"+str(name)+" "+"Predicted Values on Test
Data:"+str(predicted)+"\n\n")
        text.insert(END, "%s: %f\t\t(%f)\n"
%(name,cv_results.mean()*100,cv_results.std()))
        if name == 'Bagging':
            model_final=model
    return results

def graph():
    results=test.MLmodels()
    bars = ('SVM-Linear','RandomForest','DecisionTree','Adaboost','Bagging')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, results)
    plt.xticks(y_pos, bars)
    plt.show()
def singleImage():
    global testfile
    main_img = cv2.imread(testfile)
    #Preprocessing
    img = cv2.cvtColor(main_img, cv2.COLOR_BGR2RGB)
    gs = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
    blur = cv2.GaussianBlur(gs, (25,25),0)
    ret_otsu,im_bw_otsu =
cv2.threshold(blur,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    kernel = np.ones((50,50),np.uint8)
    closing = cv2.morphologyEx(im_bw_otsu, cv2.MORPH_CLOSE, kernel)

    #Shape features
    contours, image =
cv2.findContours(closing,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    cnt = contours[0]
    M = cv2.moments(cnt)
    area = cv2.contourArea(cnt)
    perimeter = cv2.arcLength(cnt,True)
    x,y,w,h = cv2.boundingRect(cnt)
    aspect_ratio = float(w)/h
    rectangularity = w*h/area
    circularity = ((perimeter)**2)/area

    #Color features
    red_channel = img[:, :,0]
    green_channel = img[:, :,1]
    blue_channel = img[:, :,2]
    blue_channel[blue_channel == 255] = 0
    green_channel[green_channel == 255] = 0

```

```

red_channel[red_channel == 255] = 0

red_mean = np.mean(red_channel)
green_mean = np.mean(green_channel)
blue_mean = np.mean(blue_channel)

red_std = np.std(red_channel)
green_std = np.std(green_channel)
blue_std = np.std(blue_channel)

#Texture features
textures = mt.features.haralick(gs)
ht_mean = textures.mean(axis=0)
contrast = ht_mean[1]
correlation = ht_mean[2]
inverse_diff_moments = ht_mean[4]
entropy = ht_mean[8]
vector = [area,perimeter,w,h,aspect_ratio,rectangularity,circularity,
red_mean,green_mean,blue_mean,red_std,green_std,blue_std,
          contrast,correlation,inverse_diff_moments,entropy]
return vector

def pred():
    global model_final
    global testfile
    testfile = askopenfilename(initialdir = "Dataset")
    text.insert(END,"Predict File Selected\n\n")
    vector = test.singleImage()
    test_data = pd.DataFrame([vector])

    pred=model_final.predict(test_data)
    classess=['Eosinophils','Lymphocytes','Lymphocytes','Monocytes']
    print(pred)
    text.insert(END,"For Image "+str(filename)+" Predicted Output
is"+str(classess[pred[0]])+"\n")

font = ('times', 16, 'bold')
title = Label(main, text='Image Classification of Abnormal Red Blood Cells Using
Decision Tree Algorithm')
title.config(bg='sky blue', fg='black')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 14, 'bold')
upload = Button(main, text="Upload Dataset", command=test.upload)
upload.place(x=700,y=100)
upload.config(font=font1)

pathlabel = Label(main)

```

```

pathlabel.config(bg='dark orchid', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=700,y=150)

df = Button(main, text="Reading Data ", command=test.csv)
df.place(x=700,y=200)
df.config(font=font1)

split = Button(main, text="Train_Test_Split ", command=test.splitdataset)
split.place(x=700,y=250)
split.config(font=font1)

ml= Button(main, text="All Classifiers", command=test.MLmodels)
ml.place(x=700,y=300)
ml.config(font=font1)

graph= Button(main, text="Model Comparison", command=test.graph)
graph.place(x=700,y=350)
graph.config(font=font1)

pre= Button(main, text="Predict", command=test.pred)
pre.place(x=700,y=400)
pre.config(font=font1)

font1 = ('times', 12, 'bold')
text=Text(main,height=30,width=80)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=100)
text.config(font=font1)

main.config(bg='wheat1')
main.mainloop()
import os
import cv2
import numpy as np
import pandas as pd
import mahotas as mt
from matplotlib import pyplot as plt

filename1 = 'Class Labels of Dataset 1.csv'
filename2 = 'Class Labels of Dataset 2.csv'
data1 = pd.read_csv(filename1)
data2 = pd.read_csv(filename2)

listOfFiles = list()
names =
['area','perimeter','physiological_length','physiological_width','aspect_ratio','rectangul
arity','circularity','mean_r','mean_g','mean_b','stddev_r','stddev_g','stddev_b',
'contrast','correlation','inverse_difference_moments','entropy']

```

```

    ]
df = pd.DataFrame([], columns=names)
for (dirpath, dirnames, filenames) in os.walk('dataset'):
    print(dirnames)

print(type(dirnames))
dirname=None
for file in filenames:
    print(file)
    if file.endswith(".png"):
        continue
    imgpath = os.path.join(dirpath, file)
    main_img = cv2.imread(imgpath)

    #Preprocessing
    img = cv2.cvtColor(main_img, cv2.COLOR_BGR2RGB)
    gs = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
    blur = cv2.GaussianBlur(gs, (25,25),0)
    ret_otsu,im_bw_otsu =
cv2.threshold(blur,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    kernel = np.ones((50,50),np.uint8)
    closing = cv2.morphologyEx(im_bw_otsu, cv2.MORPH_CLOSE, kernel)

    #Shape features
    contours, image =
cv2.findContours(closing,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    cnt = contours[0]
    M = cv2.moments(cnt)
    area = cv2.contourArea(cnt)
    perimeter = cv2.arcLength(cnt,True)
    x,y,w,h = cv2.boundingRect(cnt)
    aspect_ratio = float(w)/h
    rectangularity = w*h/area
    circularity = ((perimeter)**2)/area

    #Color features
    red_channel = img[:, :,0]
    green_channel = img[:, :,1]
    blue_channel = img[:, :,2]
    blue_channel[blue_channel == 255] = 0
    green_channel[green_channel == 255] = 0
    red_channel[red_channel == 255] = 0

    red_mean = np.mean(red_channel)
    green_mean = np.mean(green_channel)
    blue_mean = np.mean(blue_channel)

    red_std = np.std(red_channel)
    green_std = np.std(green_channel)

```

```

blue_std = np.std(blue_channel)

#Texture features
textures = mt.features.haralick(gs)
ht_mean = textures.mean(axis=0)
contrast = ht_mean[1]
correlation = ht_mean[2]
inverse_diff_moments = ht_mean[4]
entropy = ht_mean[8]

vector = [area,perimeter,w,h,aspect_ratio,rectangularity,circularity,
red_mean,green_mean,blue_mean,red_std,green_std,blue_std,
contrast,correlation,inverse_diff_moments,entropy
]

df_temp = pd.DataFrame([vector],columns=names)
df = df.append(df_temp)
dirname=dirpath.split('/')

columns1=data1.columns
columns2=data2.columns
class1 =list(data1[columns1[1]])
class1.extend(list(data2[columns2[1]]))
df['class'] = class1
df.to_csv('data.csv')

```

6.TESTING

6.1 SOFTWARE TESTING

Testing

Testing is a process of executing a program with the aim of finding errors. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

6.1.1 Types of Testing

1. White Box Testing
2. Black Box Testing
3. Unit testing
4. Integration Testing
5. Alpha Testing
6. Beta Testing
7. Performance Testing and so on

White Box Testing

Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers

Black Box Testing

A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality.

Unit Testing

Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team.

Integration Testing

The phase in software testing in which individual software modules are combined and

tested as a group. It is usually conducted by testing teams.

Alpha Testing

Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end users.

Beta Testing

Final testing before releasing application for commercial purpose. It is typically done by end- users or others.

Performance Testing

Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.

Black Box Testing

Blackbox testing is testing the functionality of an application without knowing the details of its implementation including internal program structure, data structures etc. Test cases for black box testing are created based on the requirement specifications. Therefore, it is also called specification-based testing. Fig.4.1 represents the black box testing:

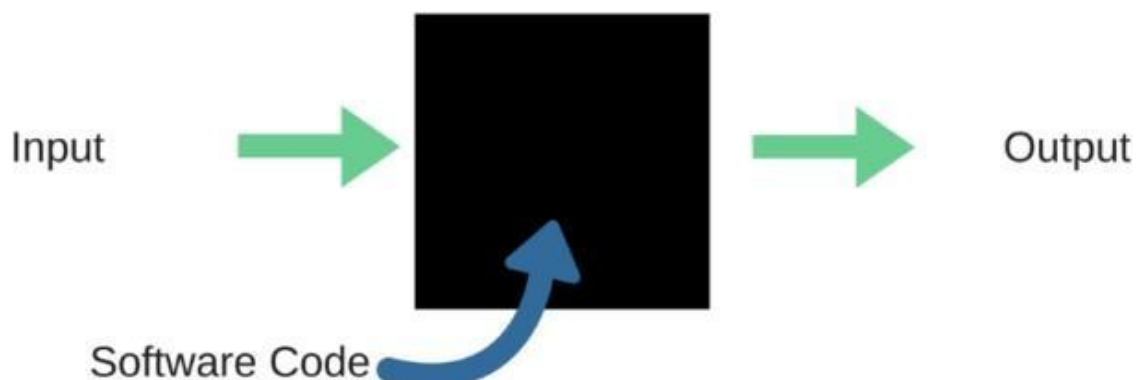


Fig6.1.1.:Black Box Testing

When applied to machine learning models, black box testing would mean testing machine learning models without knowing the internal details such as features of the machine learning

model, the algorithm used to create the model etc. The challenge, however, is to verify the test outcome against the expected values that are known beforehand.

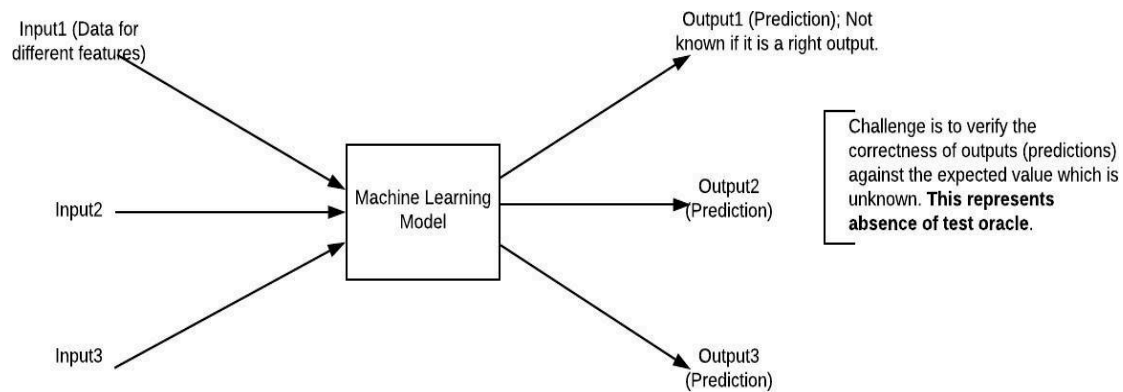


Fig.6.1.2:Black Box Testing for Machine Learning algorithms

The above Fig. represents the black box testing procedure for machine learning algorithms.

Input	Actual Output	Predicted Output
[16,6,324,0,0,0,22,0,0,0,0,0]	0	0
[16,7,263,7,0,2,700,9,10,1153,832,9,2]	1	1

Table.6.1:Black box Testing

The model gives out the correct output when different inputs are given which are mentioned in Table 4.1. Therefore the program is said to be executed as expected or the correct program.

Testing

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

7.2.2 Types of Testing

1. White Box Testing
2. Black Box Testing
3. Unit testing
4. Integration Testing
5. Alpha Testing
6. Beta Testing
7. Performance Testing and so on

White Box Testing

Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers

Black Box Testing

A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality.

Unit Testing

Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team.

Integration Testing

The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.

Alpha Testing

Type of testing a software product or system conducted at the developer's site.

Usually it is performed by the end users.

Beta Testing

Final testing before releasing application for commercial purpose. It is typically done by end- users or others.

Performance Testing

Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.

Black Box Testing

Blackbox testing is testing the functionality of an application without knowing the details of its implementation including internal program structure, data structures etc. Test cases for black box testing are created based on the requirement specifications. Therefore, it is also called as specification-based testing. Fig.4.1 represents the black box testing:

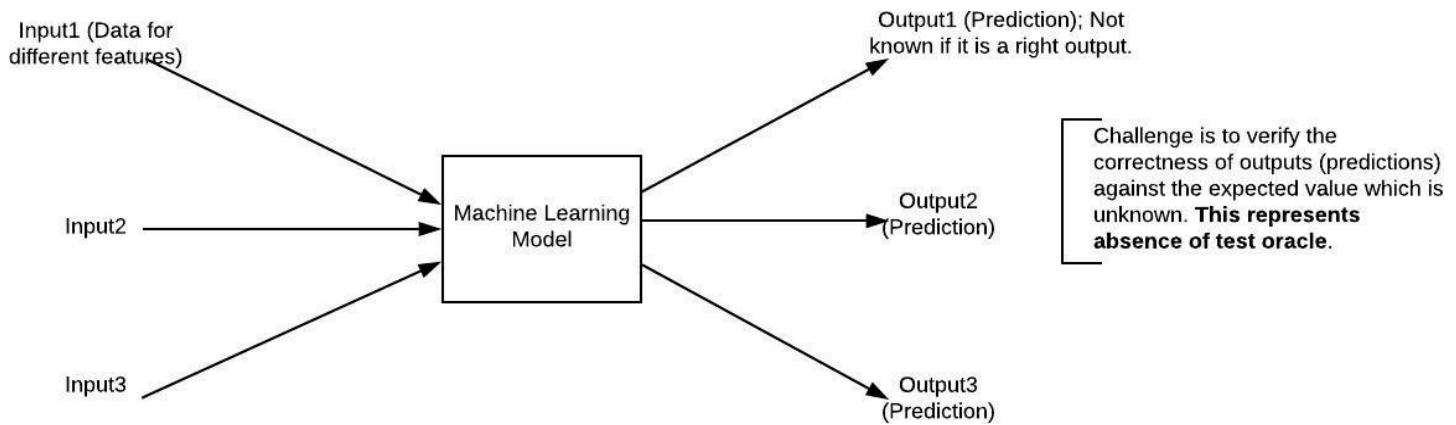


Fig.6.1.3:Black Box Testing

When applied to machine learning models, black box testing would mean testing machine learning models without knowing the internal details such as features of the machine learning

model, the algorithm used to create the model etc. The challenge, however, is to verify the test outcome against the expected values that are known beforehand.

Fig.6.1.4:Black Box Testing for Machine Learning algorithms



The above Fig.4.2 represents the black box testing procedure for machine learning algorithms.

Input	Actual Output	Predicted Output
[16,6,324,0,0,0,22,0,0,0,0,0]	0	0
[16,7,263,7,0,2,700,9,10,1153,832,9,2]	1	1

Table.6.2:Black box Testing

The model gives out the correct output when different inputs are given which are mentioned in Table 6.2. Therefore the program is said to be executed as expected or correct program

Test Case Id	Test Case Name	Test Case Description	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Start the Application	Host the application and test if it starts making sure the required software is available	If it doesn't Start	We cannot run the application.	The application hosts success.	High	High
02	Home Page	Check the deployment environment for properly loading the application.	If it doesn't load.	We cannot access the application.	The application is running successfully	High	High
03	User Mode	Verify the working of the application in freestyle mode	If it doesn't Respond	We cannot use the Freestyle mode.	The application displays the Freestyle Page	High	High
04	Data Input	Verify if the application takes input and updates	If it fails to take the input or store in The Database	We cannot proceed further	The application updates the input to application	High	High

7.RESULTS AND DISCUSSIONS

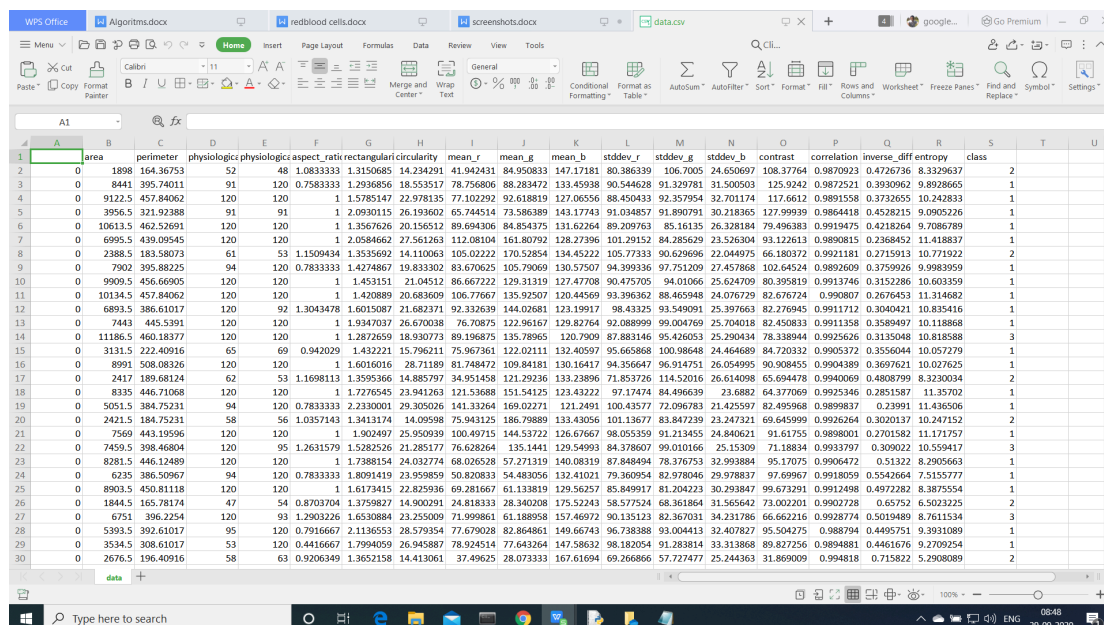
Blood is the life-maintaining fluid that flows through the heart, arteries, and other organs and it circulates through the whole body. Moreover, a blood is made up of many parts, mostly the red blood cells, white blood cells, platelets and plasma. Abnormalities of red blood cells vary through size or anisocytosis, through shape or poikilocytosis, in color and even through the presence of inclusion bodies. Detecting these irregularities in the shapes of the red blood cells is significant to one's health as it can determine whether the blood is healthy or not. Medical technicians, Pathologists and Hematologists usually used a manual microscopic method, to classify abnormal shapes of red blood cells. This methodology somehow is difficult and prone to human error. Thus, classifying the abnormal red blood cells using image processing is created using the high technologies.

Data Description:

White blood cell data collected form kaggle

For this classification we take datasets. This dataset consists of nineteen columns and 401 records. Out of nineteen columns 18 columns consist of attributes and the last column consists of class labels.

Contains 4 categories



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	area	perimeter	physiologic	physiologic	aspect_ratio	rectangularity	circularity	mean_r	mean_g	mean_b	stddev_r	stddev_g	stddev_b	contrast	correlation	inverse_diff	entropy	class			
2	0	1696	164.36753	52	48	1.0833333	1.3150685	14.234291	41.942431	84.950833	147.17181	80.386339	106.7005	24.650697	108.37764	0.9870923	0.4726736	8.3329637	2		
3	0	8441	395.74011	91	120	0.7583333	1.2936856	18.553517	78.756806	88.283472	133.45938	90.544628	91.329781	31.500503	125.9242	0.9872521	0.3930962	9.8928665	1		
4	0	9122.5	457.84062	120	120	1	1.5785147	22.978135	77.102292	92.618819	127.06556	88.450433	92.357954	32.701174	117.6612	0.9891558	0.3732655	10.242833	1		
5	0	3956.5	321.92388	91	91	1	2.0930115	26.193602	65.744514	73.586389	143.17743	91.034857	91.890791	30.218365	127.99939	0.9864418	0.4528215	9.0905226	1		
6	0	10613.5	462.52691	120	120	1	1.3567626	20.156512	89.694306	84.854375	131.62264	89.209763	85.16135	26.328184	79.496383	0.9919475	0.4218264	9.7086789	1		
7	0	6995.5	439.09545	120	120	1	2.0584662	27.561263	112.08104	161.80792	128.27396	101.29152	84.285629	23.526304	93.122613	0.9890815	0.2368452	11.418837	1		
8	0	2388.5	183.58073	61	53	1	1.1509434	1.3535692	14.110063	105.02222	170.52854	134.45222	105.77333	90.629696	22.044975	66.180372	0.9921181	0.2715913	10.771922	2	
9	0	7902	395.88225	94	120	0.7833333	1.4274867	18.833302	83.670625	105.79606	130.57507	94.299326	97.751209	27.457868	102.54524	0.9892609	0.3759926	9.9883959	1		
10	0	9909.5	456.66905	120	120	1	1.453151	21.04512	86.667222	129.31319	127.47708	90.475705	94.01066	25.624709	80.395819	0.9913746	0.3153286	10.603259	1		
11	0	10134.5	457.84062	120	120	1	1.420889	20.683609	106.77667	135.92507	120.44569	93.396362	88.465948	24.076729	82.676724	0.990807	0.2676453	11.314682	1		
12	0	6893.5	386.61017	120	92	1.3043478	1.6015087	21.682371	92.332639	144.02681	123.19917	98.43325	93.549091	25.397663	82.276945	0.9911712	0.3040421	10.835416	1		
13	0	7443	445.5391	120	120	1	1.9347037	26.670038	76.70875	122.96167	129.82764	92.088999	99.004769	25.704018	82.450833	0.9911358	0.3589497	10.118868	1		
14	0	11186.5	460.18377	120	120	1	1.2872659	18.930773	89.196875	135.78965	120.7909	87.883146	95.426053	25.290434	78.338944	0.9925626	0.3135048	10.818588	3		
15	0	3131.5	222.40916	65	69	0.942029	1.432221	15.796211	75.967361	122.02111	132.40597	95.665868	100.98648	24.464689	84.720332	0.9905372	0.3556044	10.057279	1		
16	0	8991	508.08326	120	120	1	1.6016016	28.71189	81.748472	109.84181	130.16417	94.256647	96.914751	26.054995	90.908455	0.9904389	0.3697621	10.027625	1		
17	0	2417	189.68124	62	53	1.1698113	1.3955366	14.885797	34.951458	121.29236	133.23896	71.853726	114.52016	26.614098	65.694478	0.9940069	0.4808799	8.3230034	2		
18	0	8335	446.71068	120	120	1	1.7276545	23.941263	121.53688	151.54125	123.43222	97.14744	84.496639	23.6882	64.377069	0.9925346	0.2851587	11.35702	1		
19	0	5051.5	384.75231	94	120	0.7833333	2.2330001	29.305026	141.33264	166.02271	121.2491	100.43577	72.096783	21.425597	82.495968	0.9899837	0.23991	11.436506	1		
20	0	2421.5	184.75231	58	56	1.0357143	1.3413174	14.09598	75.943125	186.79889	133.43056	101.13677	83.847239	23.247321	69.645999	0.9926264	0.3020137	10.247152	2		
21	0	7569	443.19596	120	120	1	1.902497	25.950939	100.49715	144.53722	126.67667	98.055359	91.213455	24.840621	91.61755	0.9898001	0.2701582	11.171757	1		
22	0	7459.5	398.46804	120	95	1.2631579	1.5282526	21.285177	76.628264	135.1441	129.54993	84.378607	99.010166	25.15309	71.18834	0.9933797	0.309022	10.559417	3		
23	0	8281.5	446.12489	120	120	1	1.7388154	24.032774	68.026528	57.271319	140.08319	87.848494	78.376753	32.993884	95.17075	0.9906472	0.51322	8.2905663	1		
24	0	6235	386.30967	94	120	0.7833333	1.8091419	23.959859	50.820833	54.480356	132.41021	70.360554	82.378046	29.978837	97.99967	0.9918059	0.5421664	7.5155777	1		
25	0	8903.5	450.81118	120	120	1	1.6173415	22.825936	69.281667	61.13819	129.56257	85.849917	81.204223	30.293847	99.673291	0.9912498	0.4972282	8.3875554	1		
26	0	1844.5	165.78174	47	54	0.8703704	1.3759827	14.900291	24.818333	28.340208	175.52243	58.577524	68.361864	31.565642	73.002201	0.9902728	0.65752	6.5023225	2		
27	0	6751	396.2254	120	93	1.2903226	1.6530884	23.255009	71.999861	61.188958	157.46972	90.135123	82.367031	34.231786	66.662216	0.9928774	0.5019489	8.7611534	3		
28	0	5393.5	392.61017	95	120	0.7916667	2.1136553	28.579354	77.679028	82.864861	149.66743	96.738388	93.004413	32.407827	95.504275	0.988794	0.4495751	9.3931089	1		
29	0	3534.5	308.61017	53	120	0.4416667	1.7994059	26.945887	78.924514	77.643264	147.58632	98.182054	91.283814	33.313868	89.827256	0.9894881	0.4461676	9.2709254	1		
30	0	2676.5	196.40916	58	63	0.9206349	1.3652158	14.413061	37.49625	28.073333	167.61694	69.268666	57.727477	25.244363	31.869009	0.994818	0.715822	5.2908089	2		

Steps for Machine Learning Algorithms

1. Install Anaconda Latest Version
2. Open anaconda Prompt
3. Conda create -n tf python=3.7
4. Conda activate tf
5. Install require softwares

scikit-image==0.17.2

scikit-learn==0.23.2

pandas==1.1.1

matplotlib==3.3.1

Pillow==7.2.0

plotly==4.10.0

opencv-python==4.4.0.42

spacy==2.3.2

lightgbm==3.0.0

mahotas==1.4.11

matplotlib==3.3.1lightgbm==3.0.0

mahotas==1.4.11

nltk==3.5

matplotlib==3.3.1

xgboost==1.2.0

Jupyter

6. Activate environment for jupyter notebook(For execute the in jupyter notebook)

```
python -m ipykernel install --user --name=
```

7. Goto project Directory

Note: For Text related projects. Need to Download

1. Open anaconda Prompt
2. Python
3. Import nltk

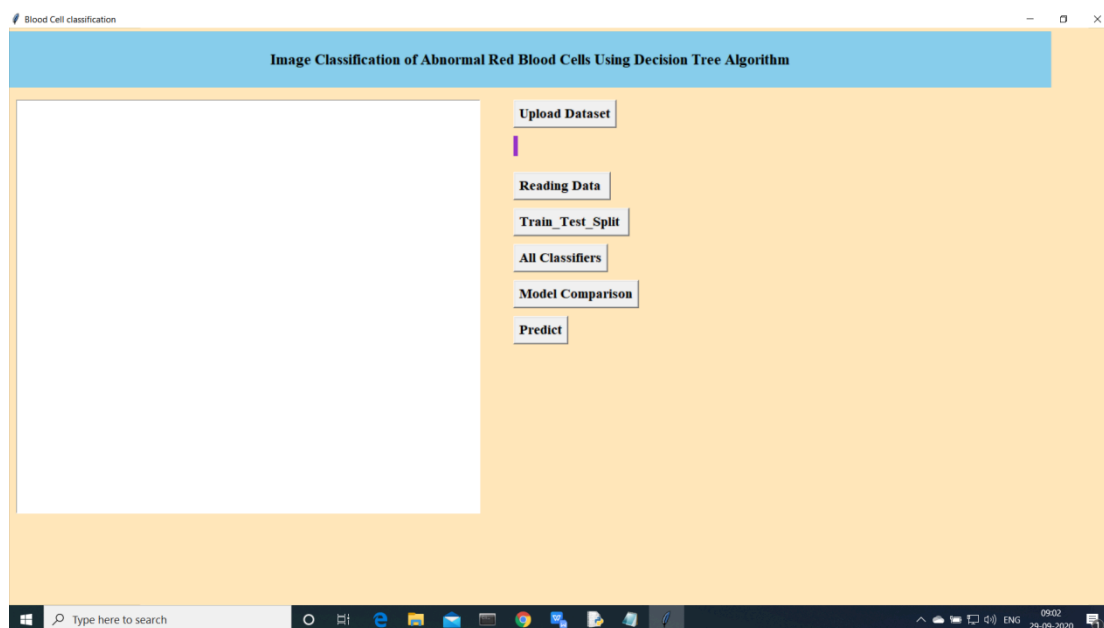
4. `Nltk.download()`

Project Development Modules:

1. **Data Collection:**Collect sufficient data samples and legitimate software samples. 📁
2. **Feature Extraction:**For each image extract the features using image processing and save in '.csv' extension 📄
3. **Train and Test Modeling:** Split the data into train and test data Train will be used for training the model and Test data to check the performance
4. **Modeling :** SVM Naive bayes, Random FOrEst,KNN,Ada boost, Decision tree, Ada boost with random forest . Combine the training using machine learning algorithms and establish a classification model.

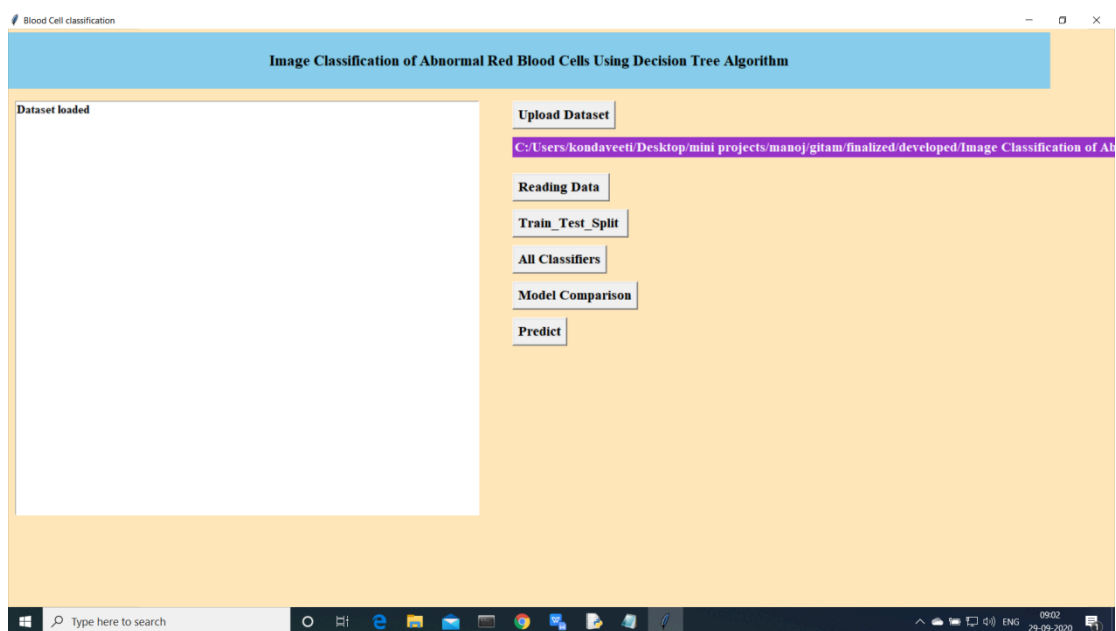
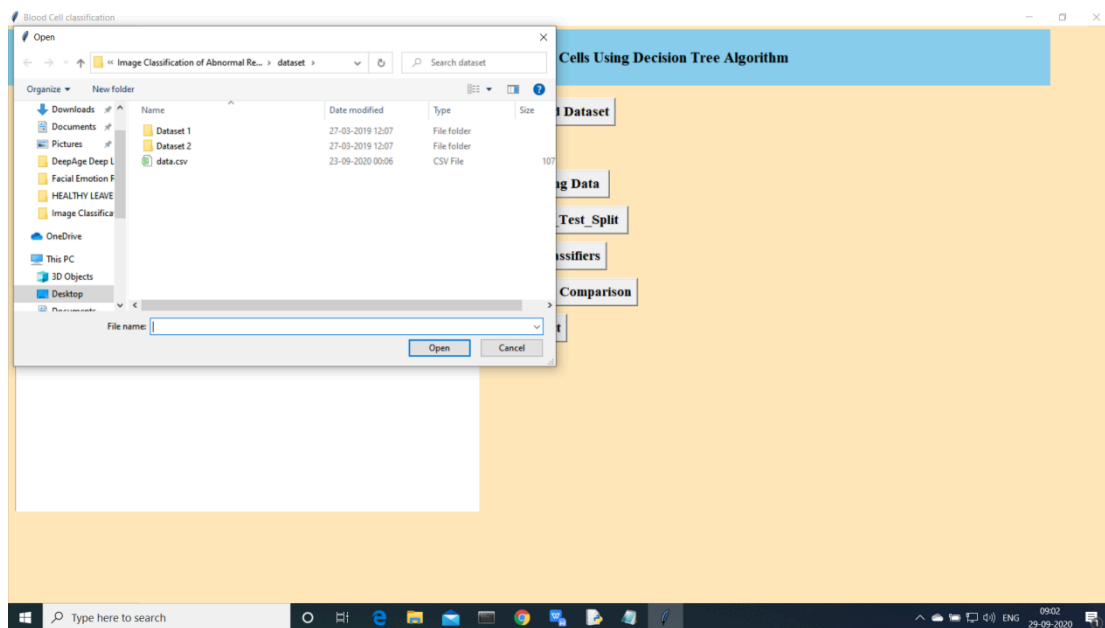
Execution Steps:

1. Open anaconda Prompt
2. Conda activate tf
3. Goto Project Directory
4. Python featureextract.py
5. Python final.py

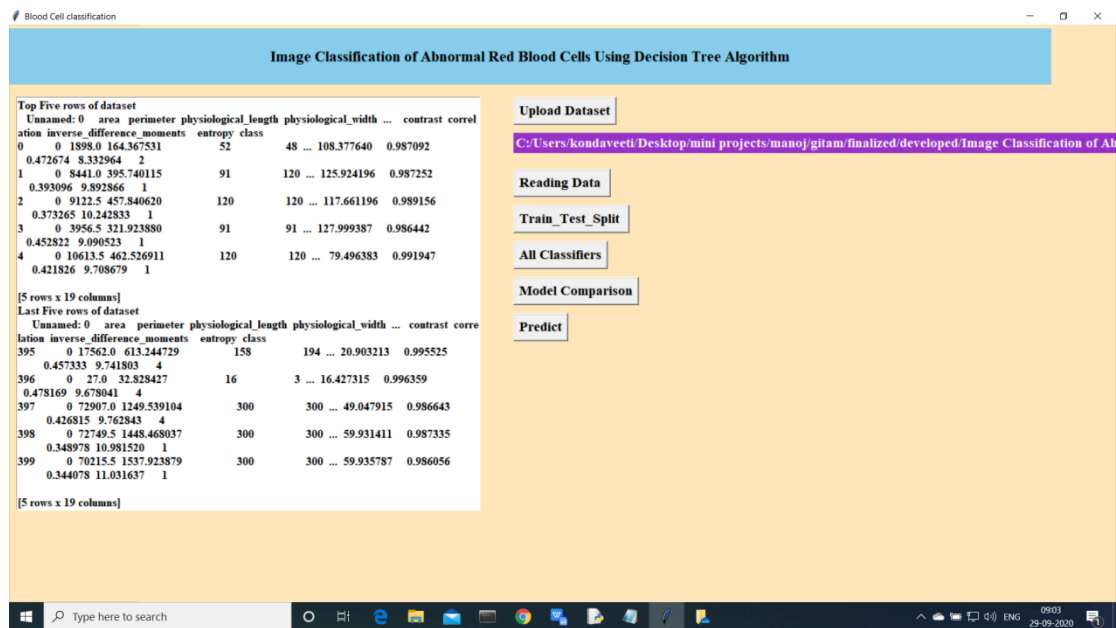


Above screen will be opened.

6. Now click on “Upload data ”

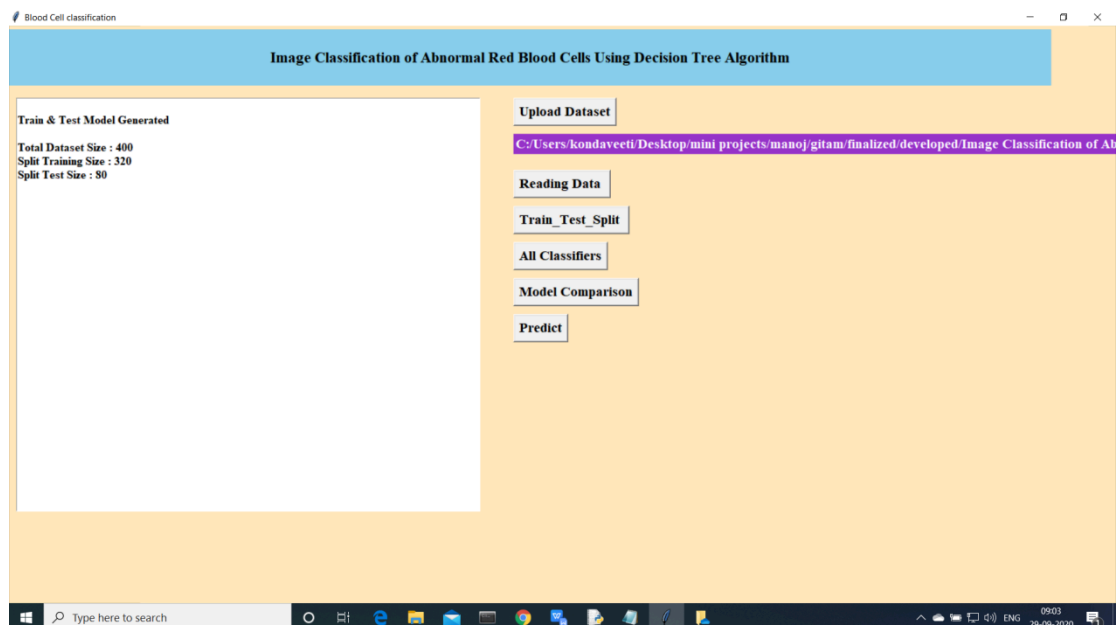


7. Import the data and Preprocee



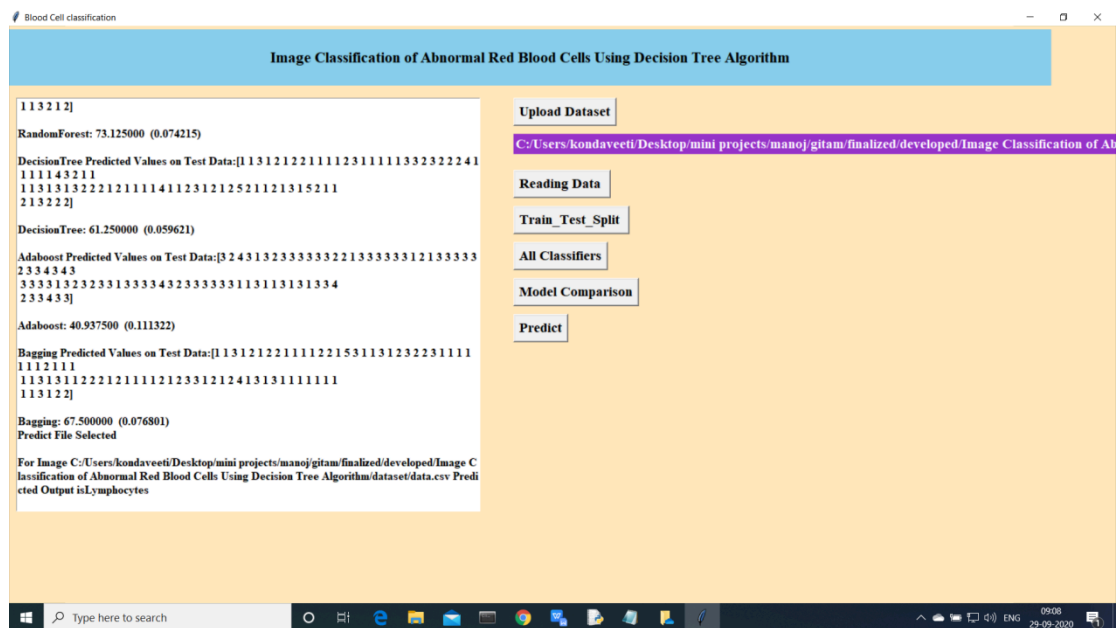
Upload the data and read the basic data information will be shown on the screen

8. Now click on “Train and Test model”. split the data into train and test and train will be used for training and to test the performance we are using test data



9. Now click on “Run Algorithms”. Mentioned algorithms will be run on the data

11. Predict:



Extension Random Forest algorithm is performed well compared other ml algorithms

8. CONCLUSION

- Random forest can be used as a classification for the abnormal red blood cells found in the blood. The system was able to classify the abnormal red blood cells using Random forest based on the data gathered from the 40 images that were composed of 400 sample cells. Errors in classifications were the result of small differences between the attributes used. Abnormal red blood cells like lymphocytes and eosinophils almost have the same parameters and attributes resulting in difficulty in classifying the two abnormal red blood cells. The average reliability rate is 89.31%. The average error rate of 10.69% was encountered mostly from node H. The irregularity of the central pallor of the codocyte was hardly recognized and detected that caused error in classification of abnormal red blood cells. The error rate for detection was often caused by the quality of the blood slides that was acquired from the hospitals. Likewise, the system was also found effective in classification of abnormal red blood cells.

FUTUREWORK

- Future work can be developing the algorithm better segmented techniques. So there is a scope of improvement in the techniques.

8.REFERENCES

- [1] Mohammad Syahputra Et. Al (2017). Abnormal Classification on the Shape of Red Blood Cells Using Radial Basis Function Network.
- [2] Pooja Tukaram Dalvi and Nagaraj Vernekar (2016). Computer Aided Detection of Abnormal Red Blood Cells.
- [3] Vishwas Sharma Et. Al (2010). Detection of Sickle Cell Anemia and Thalassaemia Causing Abnormalities in Thin Smear of Human Blood Sample Using Image Processing.
- [4] Ramin Soltanzadeh and Hossein Rabbani (2018). Classification of Three Types of Red Blood Cells in Peripheral Blood Smear Based on Morphology.
- [5] Jameela Ali Akrimi ET. Al (2014). Classification Red Blood Cells Using Support Vector Machine.
- [6] Pranati Rakshit, Et. Al (2013). Detection of Abnormal Findings in Human RBC in Diagnosing G-6-P-D Deficiency Haemolytic Anaemia Using Image Processing.
- [7] Ahmad Sabry Mohamad ET. Al (2017). Sickle Cell Disease Verification via Sobel Edge Algorithms for Image Processing.
- [8] Krishna Kumar Jha, Et. Al (2014). Detection of abnormal blood cells on the basis of nucleus shape and counting of WBC.
- [9] Nicoleta Safca, Et. Al (2018). Image Processing Techniques to Identify Red Blood Cells.
- [10] Hajara Abdulkarim Aliyu, Et. Al (2018). Red Blood Cell Classification: Deep Learning Architecture Versus Support Vector Machine.
- [11] S.F. Bikhet, Et. Al (2000) Segmentation and Classification of White Blood Cell
- [12] Sarach Tantikitti, Et. Al (2015). Image Processing for Detection of Dengue Virus Based on White Blood Cell Classification and Decision Tree
- [13] Vasundhara Acharya Et. Al (2017). Identification and Red Blood Cell Classification using Computer Aided System to Diagnose Blood Disorder

- [14] Hajara Abdulkarim Aliyu Et. Al (2018). Red Blood Cell Classification :
Deep Learning Architecture versus Support Vector Machine
- [15] Pooja Tukaram Dalvi Et. Al (2016). Computer aided detection of
abnormal red blood cells