WEB APPLICATION DEVELOPMENT

A industrial internship report submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING SUBMITTED BY

MOOLA MANOJ KUMAR
221910305028

Under the guidance of

MR. RAJ MOHAMMED

Assistant Professor

Department of Computer Science and Engineering GITAM School of Technology

GITAM Deemed to be University

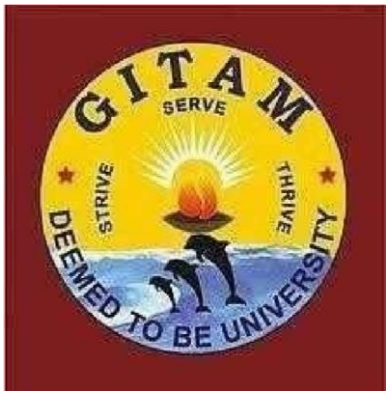
Hyderabad Campus -502329

August - 2022

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT (GITAM)

(Declared as Deemed-to-be-University u/s 3 of UGC Act 1956)

HYDERABAD CAMPUS



DECLARATION

I hereby declare that the industrial internship report entitled **"WEB APPLICATION DEVELOPMENT"** is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of

"Bachelor of Technology" in Computer Science and Engineering. The work had not been submitted to any other college or university for the award of any degree or diploma.

Place-HYDERABAD MOOLA MANOJ KUMAR

Date 221910305028

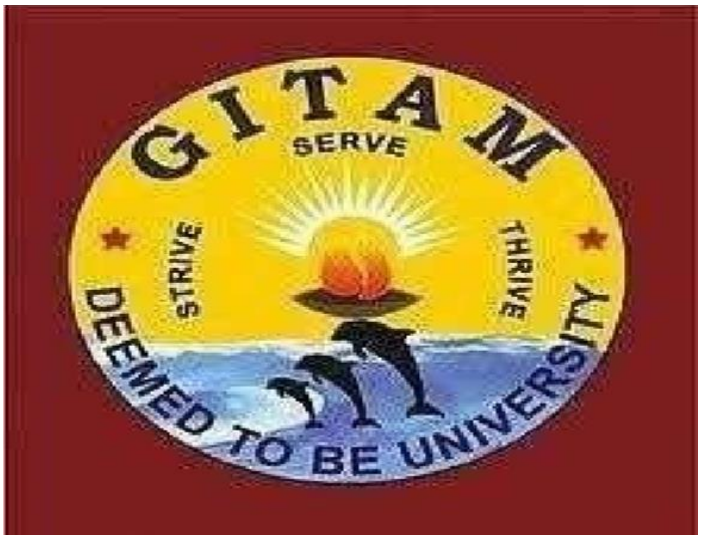
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(DEEMED TO BE UNIVERSITY)

HYDERABAD CAMPUS


GITAM
SERVE
STRIVE
THRIVE
DEEMED TO BE UNIVERSITY

MANAC
Infotech (P) Limited

Development • Training • Staffing • Recruiting

Certificate of Participation

This is to certify that

Mr./Miss Moola Manoj Kumar

has successfully completed Internship in

Web Application Development.

From 24th May 2022 to 02nd July 2022.

Cert No: MIPL/22-22/148897

Director

1-2-524/3/E/301-305, Sagar View Complex, Gaganmahal, Hyderabad - 500029.

ABSTRACT

The new Coronavirus disease (COVID-19) has seriously affected the world. By the endof November 2020, the global number of new coronavirus cases had already exceeded 60 millionand the number of deaths 1,410,378 according to information from the World Health Organization(WHO). To limit the spread of the disease, mandatory face-mask rules are now becoming common inpublic settings around the world. Additionally, many public service providers require customersto wear face-masks in accordance with predefined rules (e.g., covering both mouth and nose) whenusing public services. These developments inspired research into automatic (computer-vision-based)techniques for facemask detection that can help monitor public behavior and contribute towardsconstraining the COVID-19 pandemic. Although existing research in this area resulted in efficienttechniques for face-mask detection, these usually operate under the assumption that modern facedetectors provide perfect detection performance (even for masked faces) and that the main goal ofthe techniques is to detect the presence of face-masks only. In this study, we revisit these commonassumptions and explore the following research questions:(i) How well do existing face detectorsperform with masked-face images? (ii) Is it possible to detect a proper (regulation-compliant)placement of facial masks? and iii) How useful are existing face-mask detection techniques for monitoring applications during the COVID-19 pandemic? To answer these and related questionswe conduct a comprehensive experimental evaluation of several recent face detectors for theirperformance with masked-face images. Furthermore, we investigate the usefulness of multipleoff-the-shelf deep-learning models for recognizing correct face-mask placement. Finally, we designa complete pipeline for recognizing whether face-masks are worn correctly or not and comparethe performance of the pipeline with standard face-mask detection models from the literature. Tofacilitate the study, we compile a large dataset of facial images from the publicly available MAFA andWider Face datasets and annotate it with compliant and non-compliant labels. The annotation dataset,called Face-MaskLabel Dataset (FMLD), is made publicly available to the research community.

CONTENTS

LIST OF FIGURES

ALGORITHM & IMPLEMENTATION

ALGORITHAM MODEL FLOW DIAGREAM

LIST OF FIGURES

LIST OF SCREENSHOTS

Fig No **Name** **Page No**

CLICK ON PYTHON FILE

WRITE FILE NAME

WRITE THE CODE

FACE DETECTION WITHOUT MASK

FACE MASK DETECTION

CHAPTER 1 INTRODUCTION

ABOUT PROJECT

1.1 Introduction :

According to the World Health Organization (WHO)'s official Situation Report, coronavirus disease 2019 (COVID-19) has globally infected over 20 million people causing over 0.7 million deaths. Individuals with COVID-19 have had a wide scope of symptoms reported – going from mellow manifestations to serious illness. Respiratory problems like shortness of breath or difficulty in breathing is one of them. Elder people having lung disease can possess serious complications from COVID-19 illness as they appear to be at higher risk. Some common human

coronaviruses that infect public around the world are 229E, HKU1, OC43, and NL63. Before debilitating individuals, viruses like 2019-nCoV, SARS-CoV, and MERSCoV infect animals and evolve to human coronaviruses. Persons having respiratory problems can expose anyone (who is in close contact with them) to infective beads. Surroundings of a tainted individual can cause contact transmission as droplets carrying virus may withal arrive on his adjacent surfaces.

To curb certain respiratory viral ailments, including COVID-19, wearing a clinical mask is very necessary. The public should be aware of whether to put on the mask for source control or aversion of COVID-19. Potential points of interest of the utilization of masks lie in reducing vulnerability of risk from a noxious individual during the "pre-symptomatic" period and stigmatization of discrete persons putting on masks to restraint the spread of virus. WHO stresses on prioritizing medical masks and respirators for health care assistants. Therefore, face mask detection has become a crucial task in present global society.

Face mask detection involves in detecting the location of the face and then determining whether it has a mask on it or not. The issue is proximately cognate to general object detection to detect the classes of objects. Face identification categorically deals with distinguishing a specific group of entities i.e. Face. It has numerous applications, such as autonomous driving, education, surveillance, and so on. This paper presents a simplified approach to serve the above purpose using the basic Machine Learning (ML) packages such as OpenCV and Scikit-Learn.

Existing System:

COVID-19 pandemic has rapidly affected our day-to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. In the near future, many public service providers will ask the customers to wear masks correctly to avail of their services. Therefore, face mask detection has become a crucial task to help global society. This paper presents a simplified approach to achieve this purpose using some basic Machine Learning packages like OpenCV and Scikit-Learn. The proposed method detects the face from the image correctly and then identifies if it has a mask on it or not. As a surveillance task performer, it can also detect a face along with a mask in motion. The method attains accuracy up to 95.77% and 94.58% respectively on two different datasets. We explore optimized values of parameters using

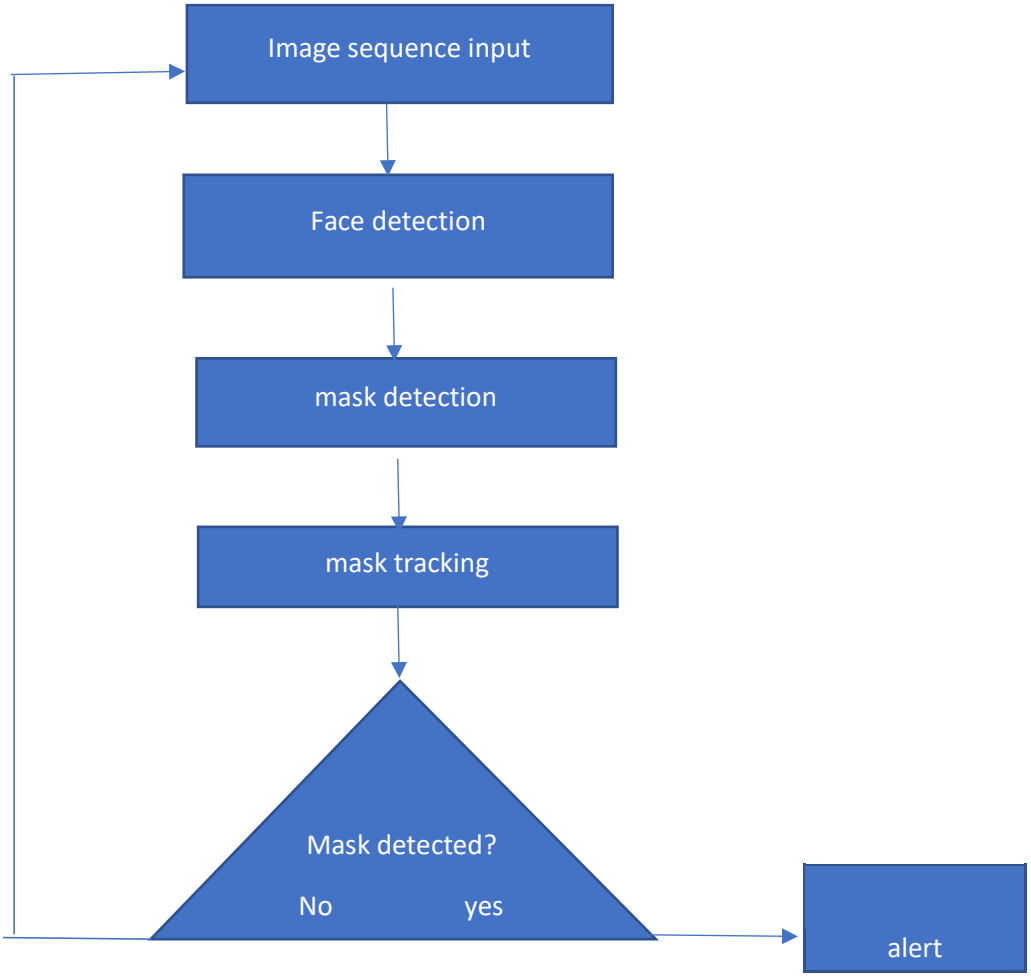
the Sequential Convolutional Neural Network model to detect the presence of masks correctly without causing over-fitting.

Proposed system:

In face detection method, a face is detected from an image that has several attributes in it. According to, research into face detection requires expression recognition, face tracking, and pose estimation. Given a solitary image, the challenge is to identify the face from the picture. Face detection is a difficult errand because the faces change in size, shape, colour, etc and they are not immutable. It becomes a laborious job for opaque image impeded by some other thing not confronting camera, and so forth. Authors in think occlusive face detection comes with two major challenges: 1) unavailability of sizably voluminous datasets containing both masked and unmasked faces, and 2) exclusion of facial expression in the covered area. Algorithm and the dictionaries trained on an immensely colossal pool of masked faces, synthesized mundane faces, several mislaid expressions can be recuperated and the ascendancy of facial cues can be mitigated to great extent. According to the work reported in, convolutional neural network (CNNs) in computer vision comes with a strict constraint regarding the size of the input image. The prevalent practice reconfigures the images before fitting them into the network to surmount the inhibition.

Here the main challenge of the task is to detect the face from the image correctly and then identify if it has a mask on it or not. In order to perform surveillance tasks, the proposed method should also detect a face along with a mask in motion.

Methodology:



WORKING :

Face mask is monitored through out using a video or web camera. In order to detect the mask the first step is to detect the face using the set of frames taken by the camera. Then the location of the mask is detected and it continuously monitored. The captured image is sent to the processor for image processing. It converts the received image to digital signal using Open CV. The digital signal is alert will be transimitted.

HARDWARE REQUIREMENTS:

- ✞ System : Intel I3 2.4 GHz. Or Advanced
- ✞ Hard Disk : 200 GB +
- ✞ Monitor : 14’ Colour Monitor. Or Advanced
- ✞ Mouse : Optical Mouse.
- ✞ Ram : 4GB +
- ✞ GRAPHICS CARD : 2GB +

SOFTWARE REQUIREMENTS:

- ✞ Operating system : Windows / linux
- ✞ Coding Language : Python /anaconda
- ✞ Front-End : Python.

OVERVIEW:

We proposed an automated smart framework for screening persons who are not using a face mask in this paper. In the smart city, all public places are monitored by cameras. The cameras are used to capture images from public places; then these images are feed into a system that identifies if any person without face mask appears in the image. If any person without a face mask is detected.

Image Preprocessing:

The images captured by the CCTV cameras required preprocessing before going to the next step. In the preprocessing step, the image is transformed into a grayscale image because the RGB color image contains so much redundant information that is not necessary for face mask detection. RGB color image stored 24 bit for each pixel of the image. On the other hand, the grayscale image stored 8 bit for each pixel and it contained sufficient information for classification. Then, we reshaped the images into (64×64) shape to maintain uniformity of the input images to the architecture. Then, the images are normalized and after normalization, the value of a pixel resides in the range from 0 to 1. Normalization helped the learning algorithm to learn faster and captured necessary features from the images.

Deep Learning Architecture:

The deep learning architecture learns various important nonlinear features from the given samples. Then, this learned architecture is used to predict previously unseen samples. To train our deep learning architecture, we collected images from different sources. The architecture of the learning technique highly depends on CNN. All the aspects of deep learning architecture are described below.

Two datasets have been used for experimenting the current method. Dataset consists of images with people wearing face masks and the images with people who do not wear face masks.

OpenCV:

OpenCV (Open Source Computer Vision Library), an open-source computer vision and ML software library, is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, expel red eyes from pictures taken utilizing flash, find comparative pictures from an image database, perceive landscape and set up markers to overlay it with increased reality and so forth. The proposed method makes use of these features of OpenCV in resizing and color conversion of data images.

Data Visualization:

Data visualization is the process of transforming abstract data to meaningful representations using knowledge communication and insight discovery through encodings. It is helpful to study a particular pattern in the dataset.

The total number of images in the dataset is visualized in both categories – ‘with mask’ and ‘without mask’.

Conversion of RGB image to Gray image:

Modern descriptor-based image recognition systems regularly work on grayscale images, without elaborating the method used to convert from color-to-grayscale. This is because the

color-to-grayscale method is of little consequence when using robust descriptors. Introducing nonessential information could increase the size of training data required to achieve good performance. As grayscale rationalizes the algorithm and diminishes the computational requisites, it is utilized for extracting descriptors instead of working on color images instantaneously.

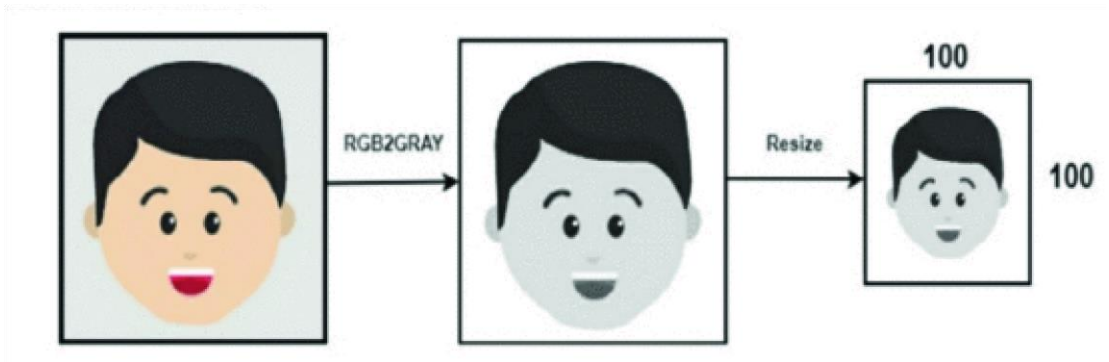


Fig : colour to gray scale image.

Image Reshaping:

The input during relegation of an image is a three-dimensional tensor, where each channel has a prominent unique pixel. All the images must have identically tantamount size corresponding to 3D feature tensor. However, neither images are customarily coextensive nor their corresponding feature tensors. Most CNNs can only accept fine-tuned images. This engenders several problems throughout data collection and implementation of model. However, reconfiguring the input images before augmenting them into the network can help to surmount this constraint.

The images are normalized to converge the pixel range between 0 and 1.As, the final layer of the neural network has 2 outputs – with mask and without mask i.e. it has categorical representation, the data is converted to categorical labels.

Training of Model

a) Building the model using CNN architecture

CNN has become ascendant in miscellaneous computer vision tasks. The current method makes use of Sequential CNN.

The Convolution layer learns from 200 filters. As the model should be aware of the shape of the input expected, the first layer in the model needs to be provided with information about input shape. Following layers can perform instinctive shape reckoning. In this case, *input_shape* is specified as *data.shape[1:]* which returns the dimensions of the data array from index 1. Default padding is "valid" where the spatial dimensions are sanctioned to truncate and the input volume is non-zero padded. The activation parameter to the Conv2D class.. It represents an approximately linear function that possesses all the assets of linear models that can easily be optimized with gradient-descent methods. Considering the performance and generalization in deep learning, it is better compared to other activation functions.

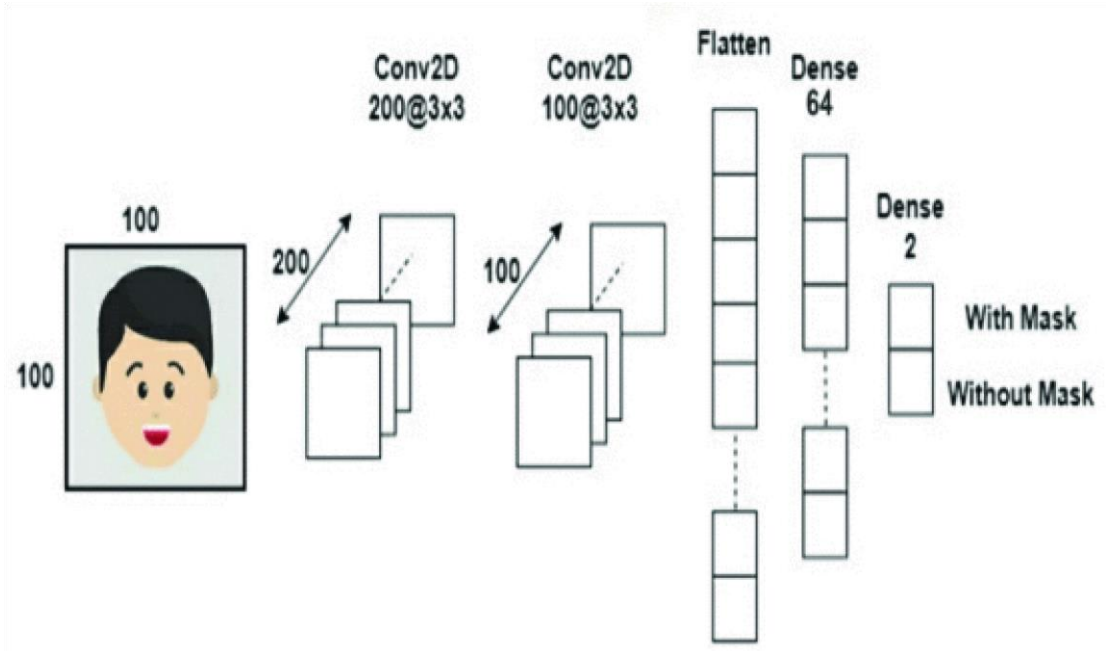


Fig : Train model image.

b) Splitting the data and training the CNN model:

After setting the blueprint to analyze the data, the model needs to be trained using a specific dataset and then to be tested against a different dataset. A proper model and optimized train_test_split help to produce accurate results while making a prediction. The test_size is set to 0.1 i.e. 90% data of the dataset undergoes training and the rest 10% goes for testing purposes.

The validation loss is monitored using ModelCheckpoint. Next, the images in the training set and the test set are fitted to the Sequential model. Here, 20% of the training data is used as validation data. The model is trained for 20 epochs (iterations) which maintains a trade-off between accuracy and chances of overfitting.

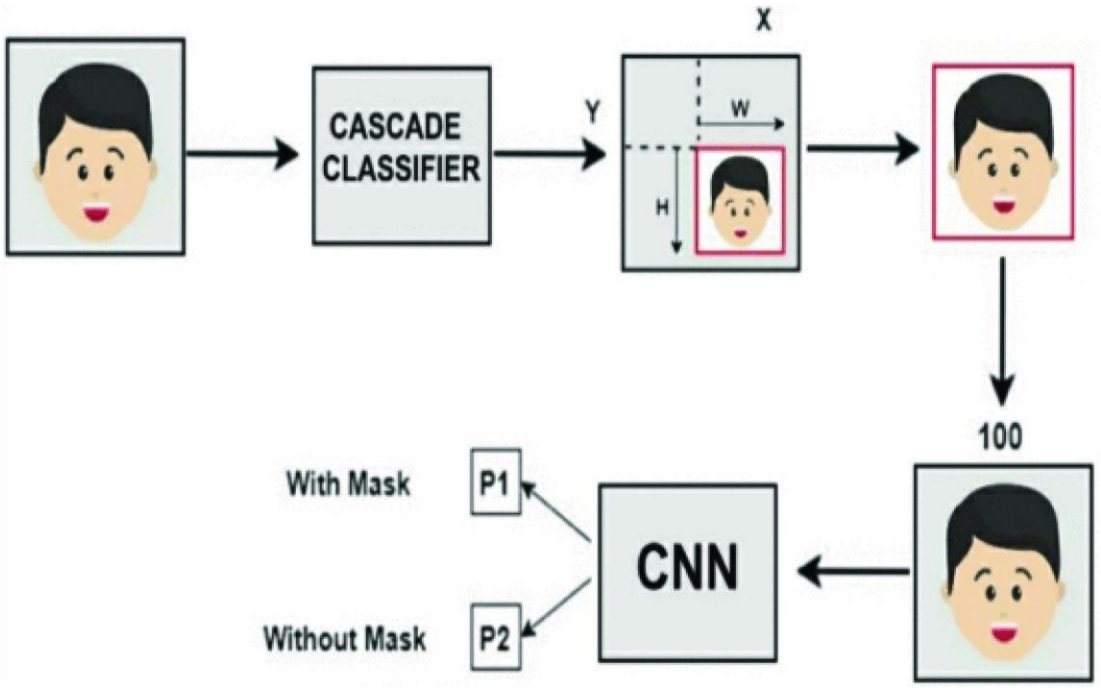


Fig. depicts visual representation of the proposed model.

CHAPTER 2

LITERATURE SURVEY

In earlier days face detection models are implemented using edge,line and centre near features and patterns are recognized from those feature. These approaches are used to find binary patterns locally. These approaches are very effective to deal with Gray-scale images and the computation effort required also very less[1] European Journal of Molecular & Clinical Medicine ISSN 2515-8260 Volume 07, Issue 06, 2020 662 AdaBoost is a regression based classifier which is going to fit regression function on original data set even some miss classified objects waits also adjusted during back propagation to optimize the results[2] Viola Jones Detector proposed an real time object model used to detect different classes of objects. It uses 24x24 base window size to evaluate any image with edge ,line and four rectangular features. Harr-like features are like convolutions to check weather given feature is available in the image or not[3]. This model fail to work in when image brightness varies even it exhibits poor performance when images are in different orientations. Convolution networks are mainly used for classification problems there are various kinds of CNN architectures such as VGG-16 this architecture consists of 2 convolution layers with input size224 kernel (64,3x3) followed by max pool with size 2x2 then again two convolution layers followed by max pool then three convolution layers with max pool again three convolution layers and max pool and three fully connected layers final FC is soft max this architecture works fine when compared to AlexNet[4,7]. Google Net architecture fundamentally using inception method by constructing small convolution layers to reduce number of parameters it having around 22 layers with convolution and max pooling etc it can able to work effectively over Alexnet it can able to bring down 60 million features in Alex net to 4 million features[8]. In this paper Deepnueral networks which adopts residual learning to train the models more deeper around 152 layers are used in this which is 8 times more than VGGnet with minimum complexity. This approach achieved relatively better performance in object detection over COCO data set[9] In this paper UNet and SEnet are used to perform segmentation of heart ventricular segmentation . This model is arrange the weights in such a way like more weights are given to useful features and less weights are given to unimportant features.[10]Support vector machines are used to perform classification on objects which is going to build an equation for constructing line and classifies the objects based on the values mapping to this line. Semantic segmentation method was used to detect facial mask in this paper they have used VGG net for training and FCN is used to semantically segment the faces available in the image[11] performed experiments on multi

parsing human data sets and achieved higher accuracy. In this paper medical image processing was done. They have taken human brain images and are trained by using FCN to identify tumours very effectively in this paper rather than using 2D segmentation for detecting tumour we have used 3D segmentation[12]. Tumuluru,Lakshmi Ramaniet. al.[13], used CNN model to detect human face which is used efficiently in security related applications. In this paper they have collected various facial features such as mouth, nose, eyes stored as facial template and used it for detecting difference between faces. Malathi, J. et. al., [14] mainly focused on identifying forgery images used in different places like in social media, and other publicity required places. In this paper various techniques are proposed to find out features of a forgery image like image spicing,copy move attack which can be handled by using correlation analysis to find duplicate features.Patelet. al., [15]proposed a model to find out the quality of the ironore by extracting the features from sample material in the mining industry. It is very important to asses the quality of the ore . SVR support vector regressor used for online measure of the quality of the ore. In this process they have extracted 280 features extracted for object identification, SFFS was model was developed using SVR. Object detection become the important area in the field of image investigation there are various techniques are there for image analysis in [15]. In this paper author introduced a European Journal of Molecular & Clinical Medicine ISSN 2515-8260 Volume 07, Issue 06, 2020 663 wavelet based nueral network for feature extraction and learning which is working efficiently in object detection[. Satapathy, Sandeep Kumar, et al.,[16]proposed a model to detect number plate which is very important problem helping police to chase many criminal cases. Authors used OCR based approach to detect characters in the number plate and they are stored and processed to client server based model for collecting the details of the owner.Pathaket.al.,[17] proposed multi dimension biometric authentication system which will work effectively in low lightening conditions here accuracy was improved by using entropy based CNN. Medical plant detection becoming very important problem which will help ordinary people to detect spices [18].In this paper authors proposed a model using CNN. It was trained with medical leaf images and it can able to detect medical plants more accurately. Human pose detection is one of the important research are nowadays drawing lot of attention [19].In this paper author proposed a model which is going to detect traditional dance based on human pose. To achieve this, they have used CNN and various steps in traditional dance was trained and model learned from it and effectively able to detect traditional dance. Ravi, Sunitha, et al. [20], sine language detection was implemented by training a CNN model which can able to detect signs in the real world video. Which is very much useful in the driver less cars also. Even it is useful in sign language in machine translation. Joint angular displacement approach was used through CNN for further enhanceing the capabilities of CNN

to capture 3D motion sign language in real time which can be applied to many real time applications in these days[21].Patel, Ashok Kumar et. al.,[22]proposed a model to find out the quality of the ironore by extracting the features from sample material in the mining industry. It is very important to asses the quality of the ore . SVR support vector regressor used for online measure of the quality of the ore. In this process they have extracted 280 features extracted for object identification, SFFS was model was developed using SVR

CHAPTER 3

AI & ML

3.1 ARTIFICIAL INTELLIGENCE

Artificial Intelligence is the future of the world. It is expanding rapidly in every industry vertical. Hence, there is a bright future in Artificial Intelligence. AI is a technique that enables machines to mimic human behavior. Artificial Intelligence is the theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making and translation between languages. AI is the simulation of human intelligence done by machines programmed by us. The machines need to learn how to reason and do some self-correction as needed along the way. Artificial Intelligence is accomplished by studying how human brain thinks, learns, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems. So, Let's continue this Artificial Intelligence Tutorial and understand it's importance. Today, a few applications of artificial intelligence seem to bring us closer to the future. The most convincing pieces of evidence are self-driving cars, Google Translate, and Sophia (humanoid robots). In this Artificial Intelligence tutorial, we shall be covering Machine Learning, Deep Learning, neural networks, real-life applications of Artificial Intelligence, Python and various packages available in it, TensorFlow, Keras, multilayer perceptron, convolution neural networks, recurrent neural networks, long short-term memory, opencv, and much more.

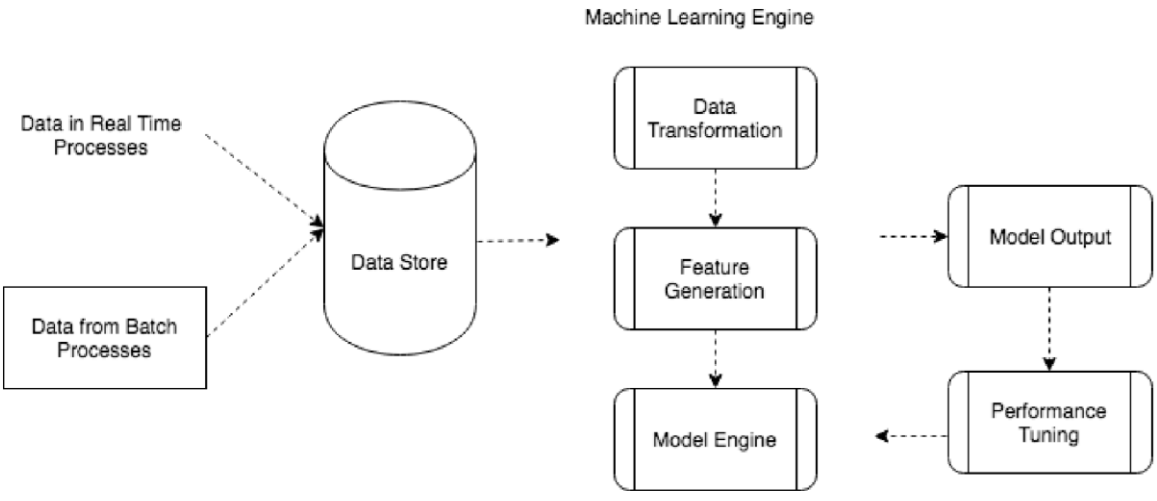


Fig:- Block diagram flow of architecture for Machine learning systems

3.2 WHAT ARE THE GOALS OF ARTIFICIAL INTELLIGENCE

Creativity and ideas never end as they are limitless. Likewise, there are a lot more things to create, improve, implement, and invent in the field of Artificial Intelligence.

Evidently, AI is far from reaching its saturation level of creating new things.

In short, here are the goals of Artificial Intelligence:

Create machines that can replicate human beings

Improve machine efficiency and accuracy

Develop tools to help people solve real-world problems, e.g., robotics for people with disabilities, auto-driving cars to avoid accidents caused by human error, etc.

3.3 APPLICATIONS OF ARTIFICIAL INTELLIGENCE

Artificial intelligence machines have the ability to make decisions and when exposed to large amounts of real-world data, they try to learn and improve themselves. To illustrate this, here are some practical applications of artificial intelligence:

Self-Driving Cars: Tesla's famous self-driving cars is a magnificent real-life application of Artificial Intelligence. These cars have in-built iot sensors for image recognition, forehead collision, spot monitoring, and many more complex mechanisms that allow them to navigate and work in real life.

Google Translate: Google Translate is another great application of Artificial Intelligence. It helps us translate sentences formed in one language to another. It can also translate the entire text on websites, which is possible only because of Artificial Intelligence.

Amazon's Alexa: Alexa includes a speech recognition system that listens to our voice commands and gives answers. It recognizes our voice and then interprets it as a series of commands and returns the results to us. It uses AVS (Alexa Voice Service), which Amazon provides for free of cost.

Google Maps: Today, without Google Maps, it is impossible to survive in the city. With Google Maps, we can travel from one place to another without any difficulty. All we have to do is open Google Maps and enter our location. Then, its navigation will lead us with the most optimized path to our destination. This is also one of the wonderful applications of artificial intelligence

3.4 SUBSETS OF ARTIFICIAL INTELLIGENCE

Artificial Intelligence is an umbrella term. There are two subsets of Artificial Intelligence: Machine Learning and Deep Learning.

3.5 APPLICATIONS OF MACHINES LEARNING

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

Emotion analysis

Sentiment analysis

Error detection and prevention

Weather forecasting and prediction

Stock market analysis and forecasting

Speech synthesis

Speech recognition

Customer segmentation

Object recognition

Fraud detection

Fraud prevention

Recommendation of products to customer in online shopping

3.6 TYPES OF MACHINE LEARNING:

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

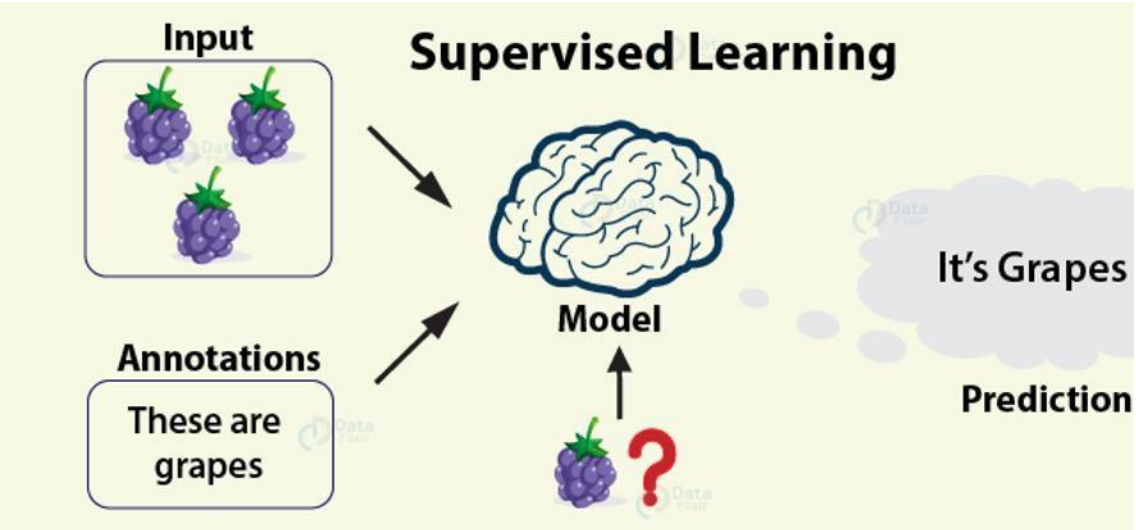
Supervised Learning

Unsupervised Learning

Reinforcement Learning

3.6.1 Supervised Learning

Supervised Learning is the most popular paradigm for performing machine learning operations. It is widely used for data where there is a precise mapping between input-output data. The dataset, in this case, is labeled, meaning that the algorithm identifies the features explicitly and carries out predictions or classification accordingly. As the training period progresses, the algorithm is able to identify the relationships between the two variables such that we can predict a new outcome.



Resulting Supervised learning algorithms are task-oriented. As we provide it with more and more examples, it is able to learn more properly so that it can undertake the task and yield us the output more accurately. Some of the algorithms that come under supervised learning are as follows –

Linear Regression

In linear regression, we measure the linear relationship between two or more than two variables. Based on this relationship, we perform predictions that follow this linear pattern.

Random Forest

Random Forests are an ensemble learning method that is for performing classification, regression as well as other tasks through the construction of decision trees and providing the output as a class which is the mode or mean of the underlying individual trees.

Gradient Boosting

Gradient Boosting is an ensemble learning method that is a collection of several weak decision trees which results in a powerful classifier.

Support Vector Machine:

Svms are powerful classifiers that are used for classifying the binary dataset into two classes with the help of hyperplanes.

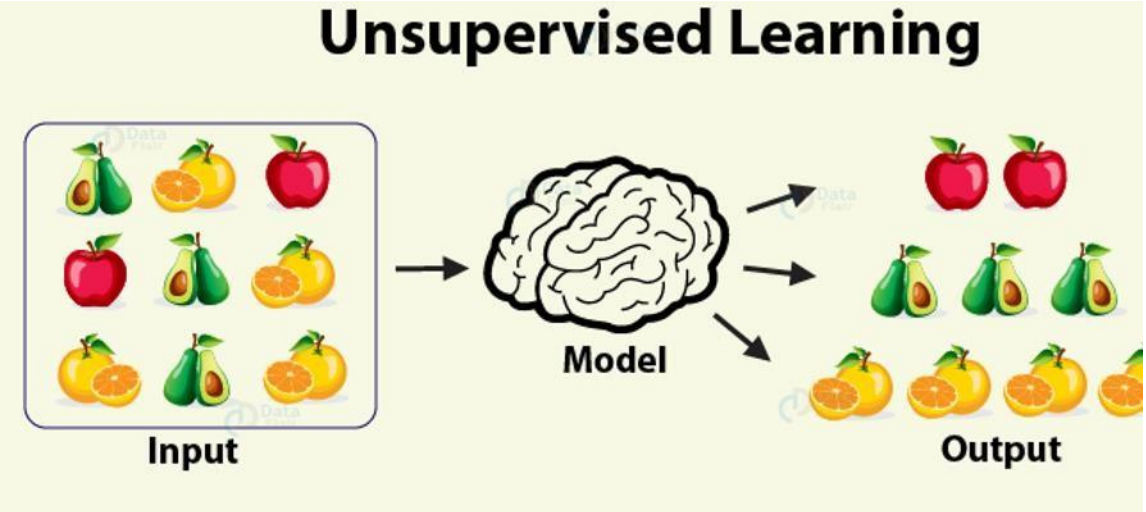
Logistic Regression:

It makes use of a bell-shaped S curve that is generated with the help of logit function to categorize the data into their respective classes.

3.6.2 Unsupervised Learning:

In the case of unsupervised learning algorithm, the data is not explicitly labeled into different classes, that is, there are no labels. The model is able to learn from the data by finding implicit patterns. Unsupervised Learning algorithms identify the data based on their densities, structures, similar segments, and other similar features. Unsupervised Learning Algorithms are based on Hebbian Learning. Cluster analysis is one of the most widely used techniques in supervised

learning. Let us look at some of the important algorithms that come under Unsupervised Learning.



Clustering

Clustering, also known as cluster analysis, is a technique of grouping similar sets of objects in the same group that is different from the objects in other group. Some of the essential clustering techniques are as follows –

K-means

The aim of the k-means clustering algorithm is to partition the n observations in the data into k clusters such that each observation belongs to the cluster with the nearest mean. This serves as the prototype of the cluster.

DBSCAN

This is a clustering method that groups the data based on the density. It groups together the points that are given in the space and marks the outliers in the low-density region.

Hierarchical clustering

In this form of clustering, a hierarchy of clusters is built.

Anomaly Detection

Anomaly Detection techniques detect outliers in the unlabeled data under an assumption that most of the data examples are normal by observing the instances that fit the remainder of the data set.

Autoencoders

Autoencoders are a type of Neural Networks that are used in Unsupervised Learning for representation learning. They are used in denoising and dimensionality reduction.

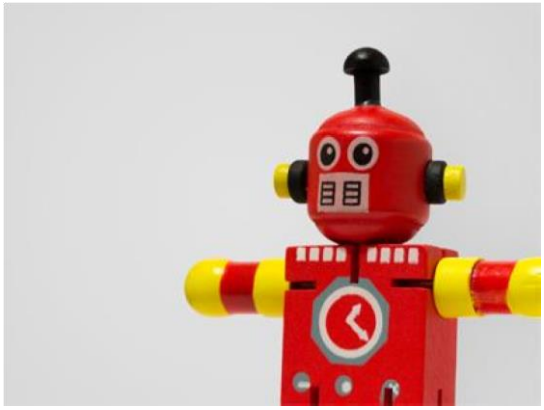
Deep Belief Network

It is a generative graphical model which is also a class of neural network designed for unsupervised learning. It is different from the supervised type of neural networks in the sense that it probabilistically reconstructs its inputs to act as feature detectors.

Principal Component Analysis

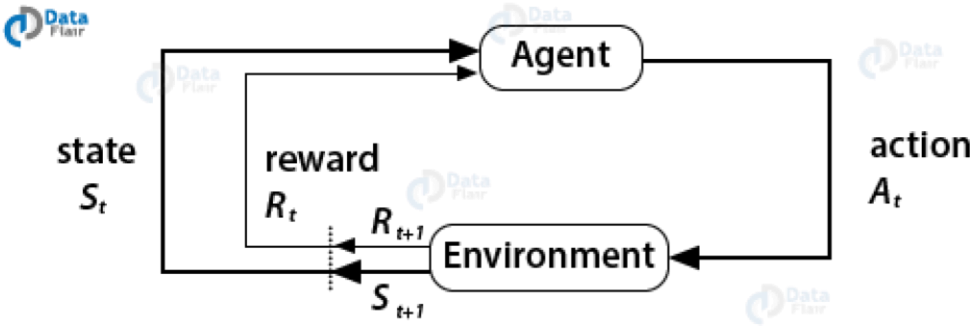
It is a class of unsupervised learning paradigm which is used for reducing the dimensions of the data.

3.6.3 Reinforcement Learning:



Reinforcement Learning covers more area of Artificial Intelligence which allows machines to interact with their dynamic environment in order to reach their goals. With this, machines and software agents are able to evaluate the ideal behavior in a specific context. With the help of this reward feedback, agents are able to learn the behavior and improve it in the longer run.

This simple feedback reward is known as a reinforcement signal.



The agent in the environment is required to take actions that are based on the current state. This type of learning is different from Supervised Learning in the sense that the training data in the former has output mapping provided such that the model is capable of learning the correct answer. Whereas, in the case of reinforcement learning, there is no answer key provided to the agent when they have to perform a particular task. When there is no training dataset, it learns from its own experience.

Based on learning ability

In the learning process, the following are some methods that are based on learning ability −

Batch Learning

In many cases, we have end-to-end Machine Learning systems in which we need to train the model in one go by using whole available training data. Such kind of learning method or algorithm is called Batch or Offline learning. It is called Batch or Offline learning because it is a one-time procedure and the model will be trained with data in one single batch. The following are the main steps of Batch learning methods −

Step 1 − First, we need to collect all the training data for start training the model.

Step 2 − Now, start the training of model by providing whole training data in one go.

Step 3 − Next, stop learning/training process once you got satisfactory results/performance.

Step 4 − Finally, deploy this trained model into production. Here, it will predict the output for new data sample.

Online Learning

It is completely opposite to the batch or offline learning methods. In these learning methods, the training data is supplied in multiple incremental batches, called mini-batches, to the algorithm. Followings are the main steps of Online learning methods −

Step 1 − First, we need to collect all the training data for starting training of the model.

Step 2 − Now, start the training of model by providing a mini-batch of training data to the algorithm.

Step 3 − Next, we need to provide the mini-batches of training data in multiple increments to the algorithm.
Step 4 − As it will not stop like batch learning hence after providing whole training data in mini-batches, provide new data samples also to it.

Step 5 − Finally, it will keep learning over a period of time based on the new data samples.

Based on Generalization Approach

In the learning process, followings are some methods that are based on generalization approaches −

Instance based Learning

Instance based learning method is one of the useful methods that build the ML models by doing generalization based on the input data. It is opposite to the previously studied learning methods in the way that this kind of learning involves ML systems as well as methods that uses the raw data points themselves to draw the outcomes for newer data samples without building an explicit model on training data.

In simple words, instance-based learning basically starts working by looking at the input data points and then using a similarity metric, it will generalize and predict the new data points.

Model based Learning

In Model based learning methods, an iterative process takes place on the ML models that are built based on various model parameters, called hyperparameters and in which input data is used to extract the features. In this learning, hyperparameters are optimized based on various model validation techniques. That is why we can say that Model based learning methods uses more traditional ML approach towards generalization.

3.2 REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

3.2.1 REQUIREMENT SPECIFICATION

Functional Requirements

- ✦ Graphical User interface with the User.

3.2.2 SYSTEM SPECIFICATION:

HARDWARE REQUIREMENTS:

- ✞ System : Intel I3 2.4 GHz. Or Advanced
- ✞ Hard Disk : 200 GB +
- ✞ Monitor : 14’ Colour Monitor. Or Advanced
- ✞ Mouse : Optical Mouse.
- ✞ Ram : 4GB +
- ✞ GRAPHICS CARD : 2GB +

SOFTWARE REQUIREMENTS:

- ✞ Operating system : Windows / linux
- ✞ Coding Language : Python /anaconda
- ✞ Front-End : Python.

3.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and

to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.4 SDLC

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.

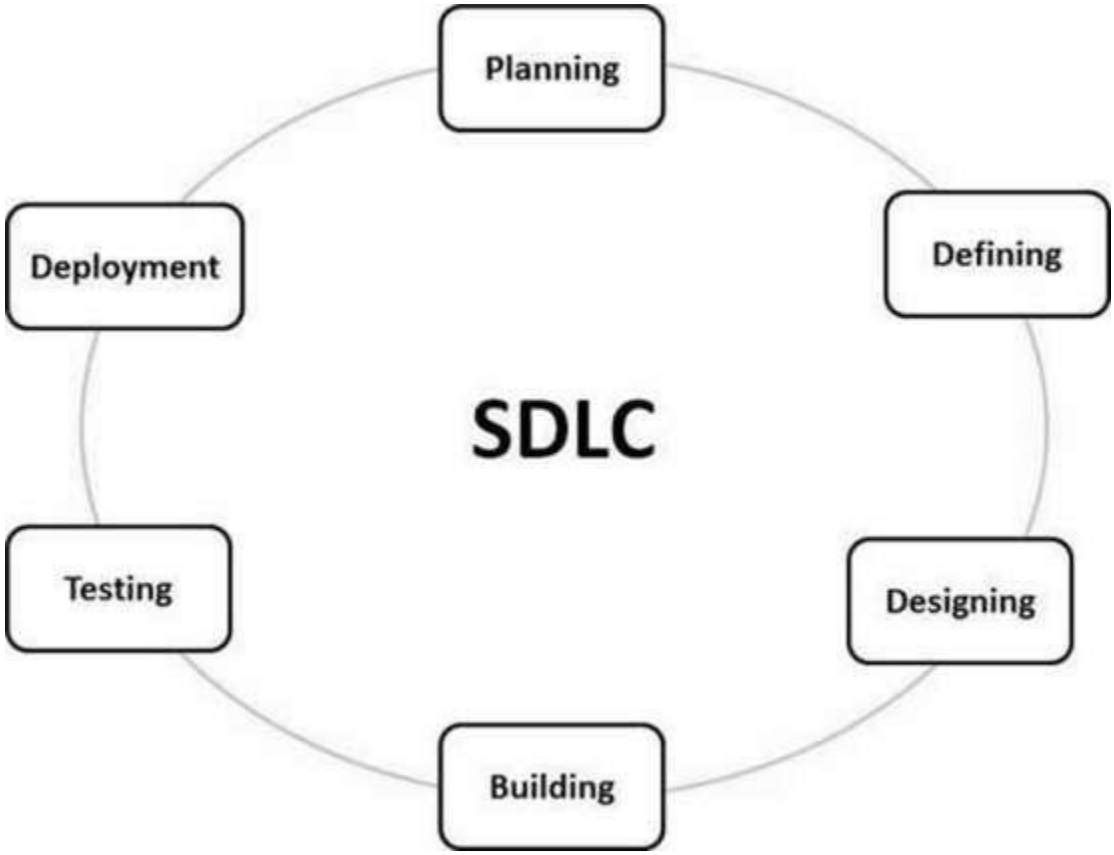


Fig.1 : Software Development Life Cycle(SDLC) representation

A typical Software Development Life Cycle consists of the following stages −

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through

an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

3.4.1. OBJECT ORIENTED SYSTEM DEVELOPMENT

In Object-Oriented Development, we apply object orientation across every system development activity such as requirement analysis, design, programming, testing, and maintenance. For instance, an object oriented analysis (OOA) will usually have the following steps:

- Identifying the system functionality
- Identifying the involved objects
- Recognizing the object classes
- Analysing the objects to fulfil the system functionality

Object-Oriented Programming (OOP) is based on object oriented features like Encapsulation, Polymorphism, Inheritance and Abstraction. These features are usually referred to as the OOPs concepts. We will see more about analysis, design, testing and maintenance later.

3.4.2 OBJECT BASICS

An object is a real-world element in an object–oriented environment that may have a physical or a conceptual existence. Each object has −

- Identity that distinguishes it from other objects in the system.

- State that determines the characteristic properties of an object as well as the values of the properties that the object holds.

- Behavior that represents externally visible activities performed by an object in terms of changes in its state.

Objects can be modelled according to the needs of the application. An object may have a physical existence, like a customer, a car, etc.; or an intangible conceptual existence, like a project, a process, etc.

3.5 UNIFIED APPROACH

The Unified Approach (UA) is a methodology for software development that is proposed by the author Ali Bahrami (1999). The UA, based on methodologies by Booch, Rumbaugh, and Jacobson, tries to combine the best practices, processed, and guidelines along with the Object

Management Group's unified modeling language. The unified modeling language (UML) is a set of notations and conventions used to describe and model an application. However, the UML does not specify a methodology or what steps to follow to develop an application; that would be the task of the UA. The heart of the UA is Jacobson's use case. The use case represents a typical interaction between a user and a computer system to capture the users' goals and needs. In its simplest usage, you capture a use case by talking to typical users and discussing the various ways they might want to use the system. The use cases are entered into all other activities of the UA.

The UA establishes a unifying and unitary framework around their works by utilizing the UML to describe the model and document the software development process. The idea behind the UA is not to introduce yet another methodology. The main motivation here is to combine the best practices, processes, methodologies, and guidelines along with UML notations and diagrams for better understanding of object-oriented concepts and system development.

The unified approach to software development revolves around (but is not limited to) to the following processes and concepts. The processes are:

1. Use-case driven development
2. Object-oriented analysis

3. Object-oriented design

4. Incremental development and prototyping

5. Continuous testing

The UA allows iterative development by allowing you to go back and forth between the design and the modeling or analysis phases. It makes backtracking very easy and departs from the linear waterfall process, which allows no form of back tracking.

I .Object-Oriented Analysis

Analysis is the process of extracting the needs of a system and what the system must do to satisfy the users' requirements. The goal of object-oriented analysis is to first understand the domain of the problem and the system's responsibilities by understanding how the users use or will use the system. It concentrates on describing what the system does rather than how it does it. Separating the behavior of a system from the way it is implemented require viewing the system from the user's perspective rather than that of the machine. OOA process consists of the following steps:

1. Identify the Actors.

2. Develop a simple business process model using UML Activity diagram.

3. Develop the Use Case.

4. Develop interaction diagrams.

5. Identify classes.

II .Object-Oriented Design

Booch, provides the most comprehensive object-oriented design method. Ironically, since it is so comprehensive, the method can be somewhat imposing to learn and especially tricky to figure out where to start. Rumbaugh et al.'s and Jacobson et al.'s high-level models provide good avenues for getting started. UA combines these by utilizing Jacobson et al.'s analysis and interaction diagrams, Booch's object diagrams, and Rumbaugh et al.'s domain models. Furthermore, by following Jacobson et al.'s life cycle model, we can produce designs that are traceable across requirements, analysis, design, coding, and testing. OOD Process consists of:

1. Designing classes, their attributes, methods, associations, structures and protocols, apply design axioms.

2. Design the Access Layer

3. Design and prototype User interface

4. User Satisfaction and Usability Tests based on the Usage/Use Cases 5. Iterated and refine the design

III .Iterative Development and Continuous Testing

You must iterate and reiterate until, eventually, you are satisfied with the system. Sine testing often uncovers design weaknesses or at least provides additional information you will want to use, repeat the entire process, taking what you have learned and reworking your design or moving on the prototyping and retesting. Continue this refining cycle through the development process until you are satisfied with the results. During this iterative process, your prototypes will be incrementally transformed into the actual application. The UA encourages the integration of testing plans from day 1 of the project. Usage scenarios can become test scenarios; therefore, use case will drive the usability testing. Usability testing is the process in which the functionality of software is measured.

4 SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE :

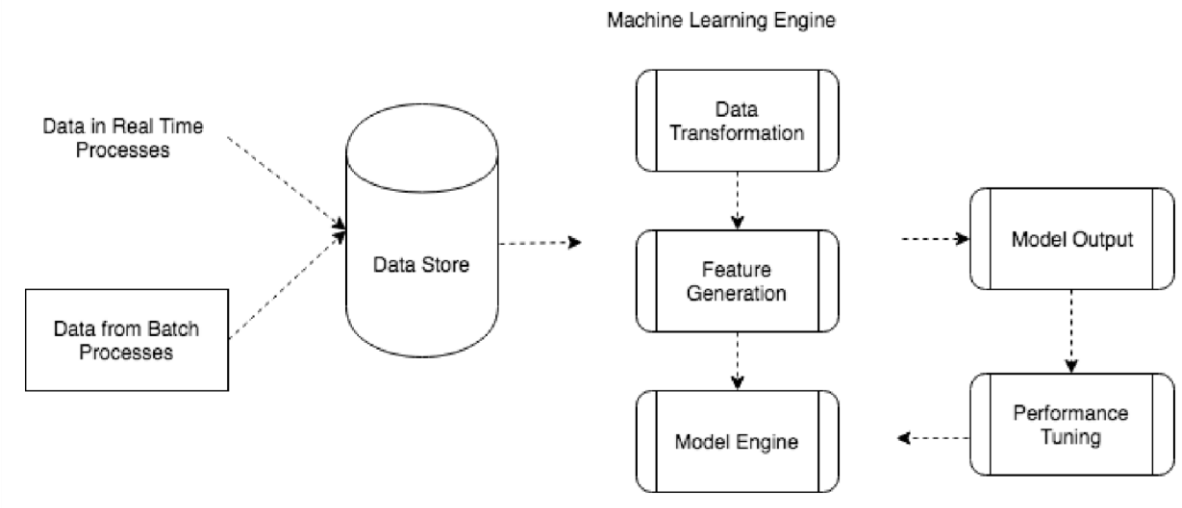


Fig. 2 : System architecture

4.2 Data flow diagrams

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

Data-flow diagrams can be regarded as inverted Petri nets, because places in such networks correspond to the semantics of data memories. Analogously, the semantics of transitions from Petri nets and data flows and functions from data-flow diagrams should be considered equivalent.

4.3 UML concepts

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

OMG is continuously making efforts to create a truly industry standard.

- UML stands for **Unified Modeling Language**.
- UML is different from the other common programming languages such as C++, Java, COBOL, etc.
- UML is a pictorial language used to make software blueprints.
- UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.
- Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.

4.3.1 Goals of UML

A picture is worth a thousand words, this idiom absolutely fits describing UML. Objectoriented concepts were introduced much earlier than UML. At that point of time, there were no standard methodologies to organize and consolidate the object-oriented development. It was then that UML came into picture.

There are a number of goals for developing UML but the most important is to define some general purpose modeling language, which all modelers can use and it also needs to be made simple to understand and use.

UML diagrams are not only made for developers but also for business users, common people, and anybody interested to understand the system. The system can be a software or non-software system. Thus it must be clear that UML is not a development method rather it accompanies with processes to make it a successful system.

In conclusion, the goal of UML can be defined as a simple modeling mechanism to model all possible practical systems in today's complex environment.

4.3.2 A Conceptual Model of UML

To understand the conceptual model of UML, first we need to clarify what is a conceptual model? and why a conceptual model is required?

- A conceptual model can be defined as a model which is made of concepts and their relationships.

- A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other.

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. The conceptual model of UML can be mastered by learning the following three major elements −

- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML.

4.3.3 UML diagrams

A UML diagram is a diagram based on the UML (**Unified Modeling Language**) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system. UML stands for Unified Modeling Language. UML is a standardized generalpurpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Metamodel and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

1. ARCHITECTURE DIAGRAM

In software engineering, a architecture 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.

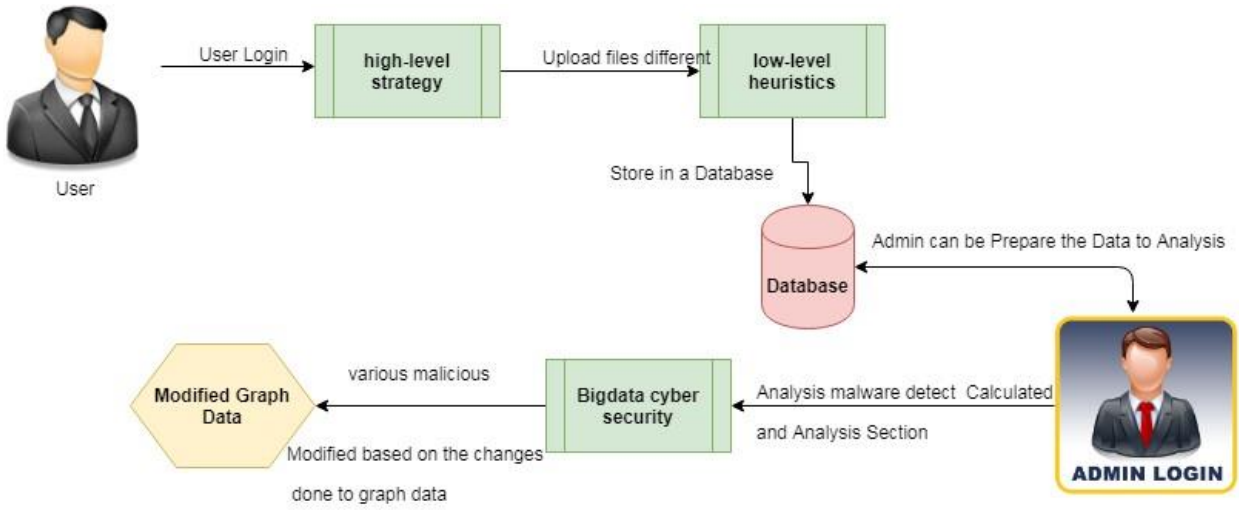
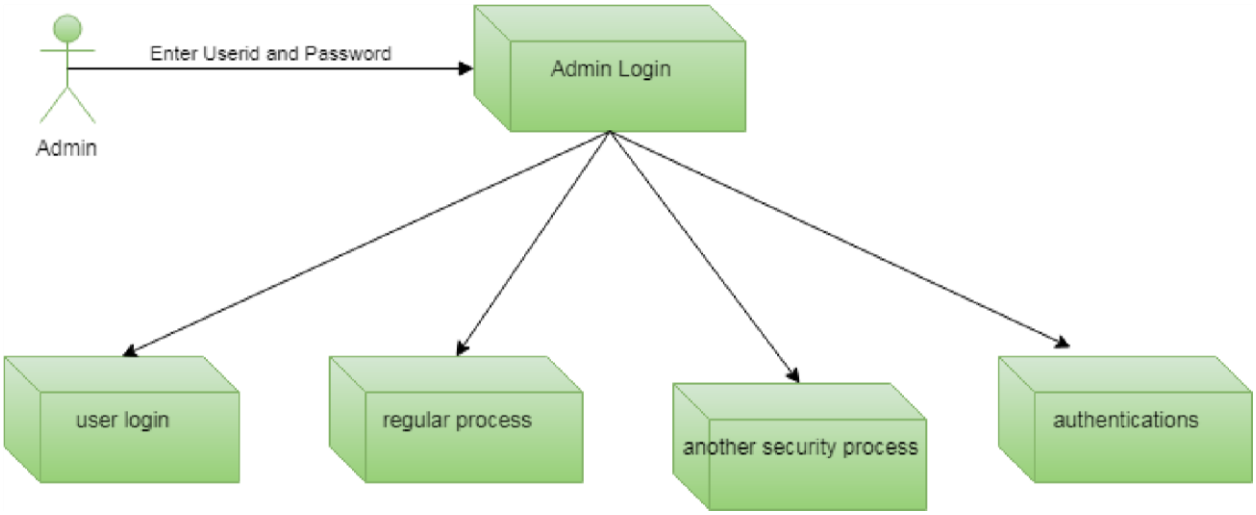


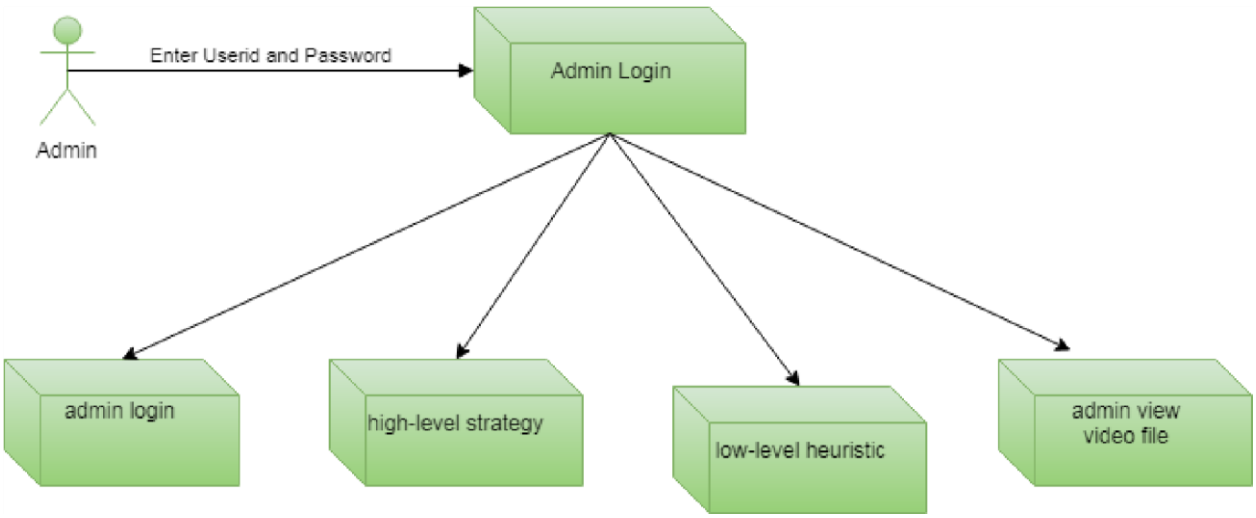
Fig. 4 Architecture diagram

2 COMPONENT DIAGRAM:

The purpose of a **component diagram** is to show the relationship between different **components** in a system. For the purpose of **UML** 2.0, the term "**component**" refers to a module of classes that represent independent systems or subsystems with the ability to interface with the rest of the system. **a.User**



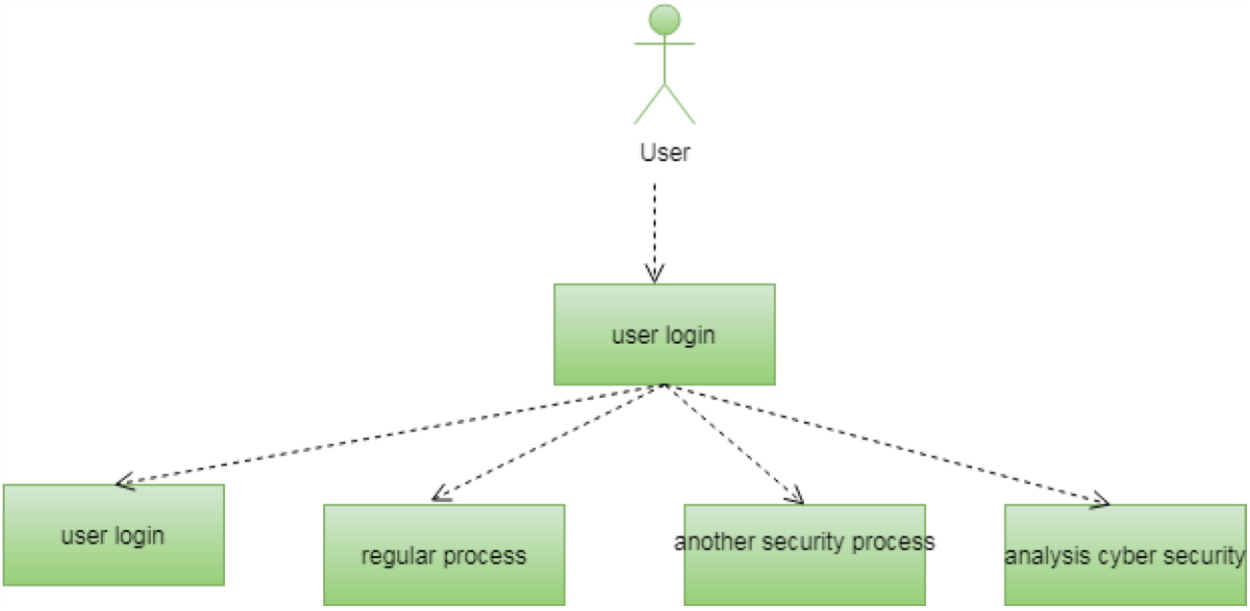
b.Admin:



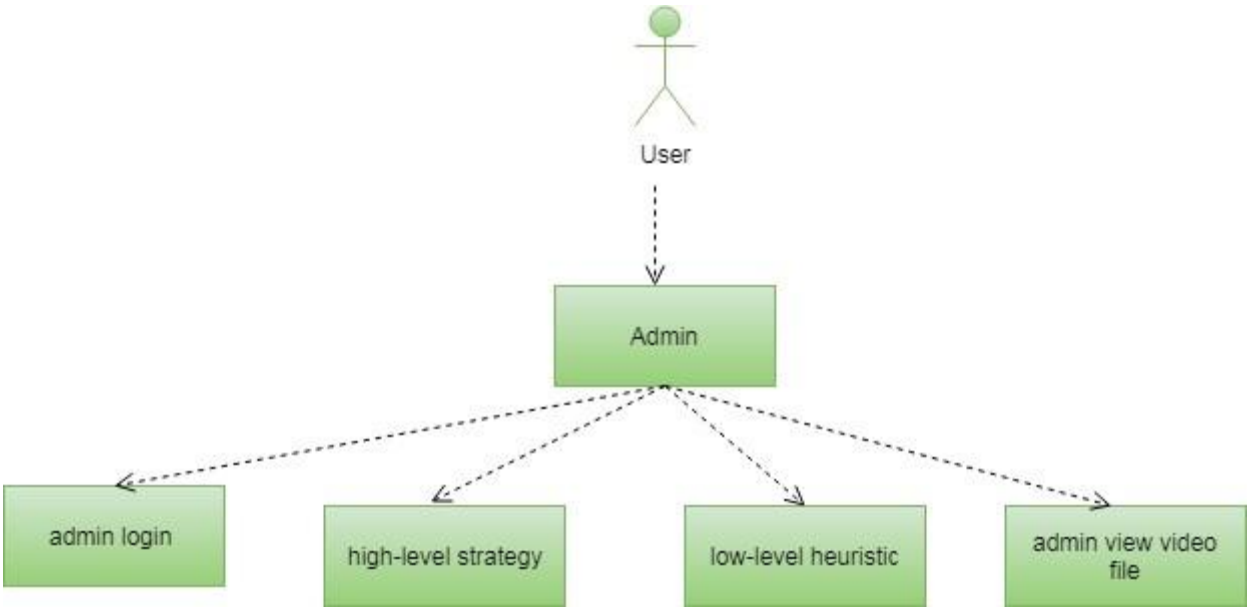
4.USE CASE DIAGRAM

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.

a.User:

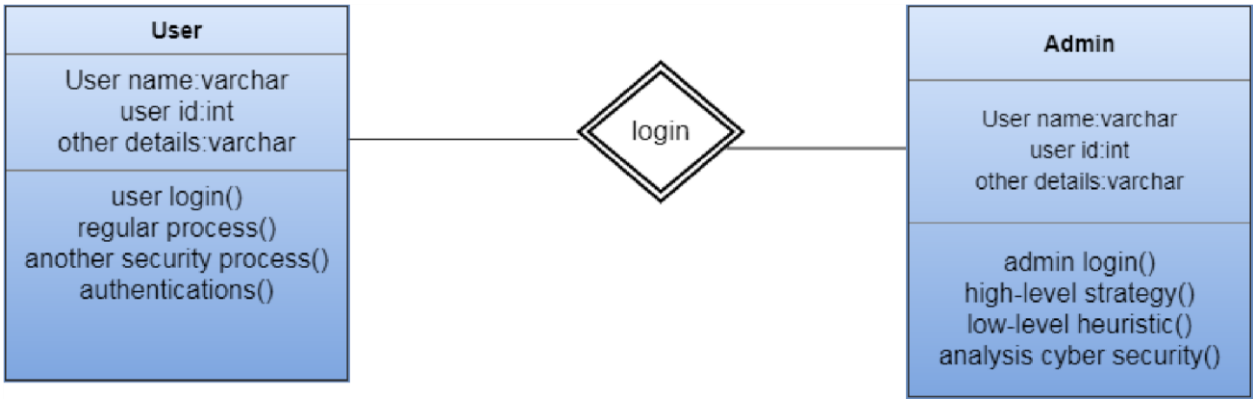


b.Admin:



5.CLASS DIAGRAM:

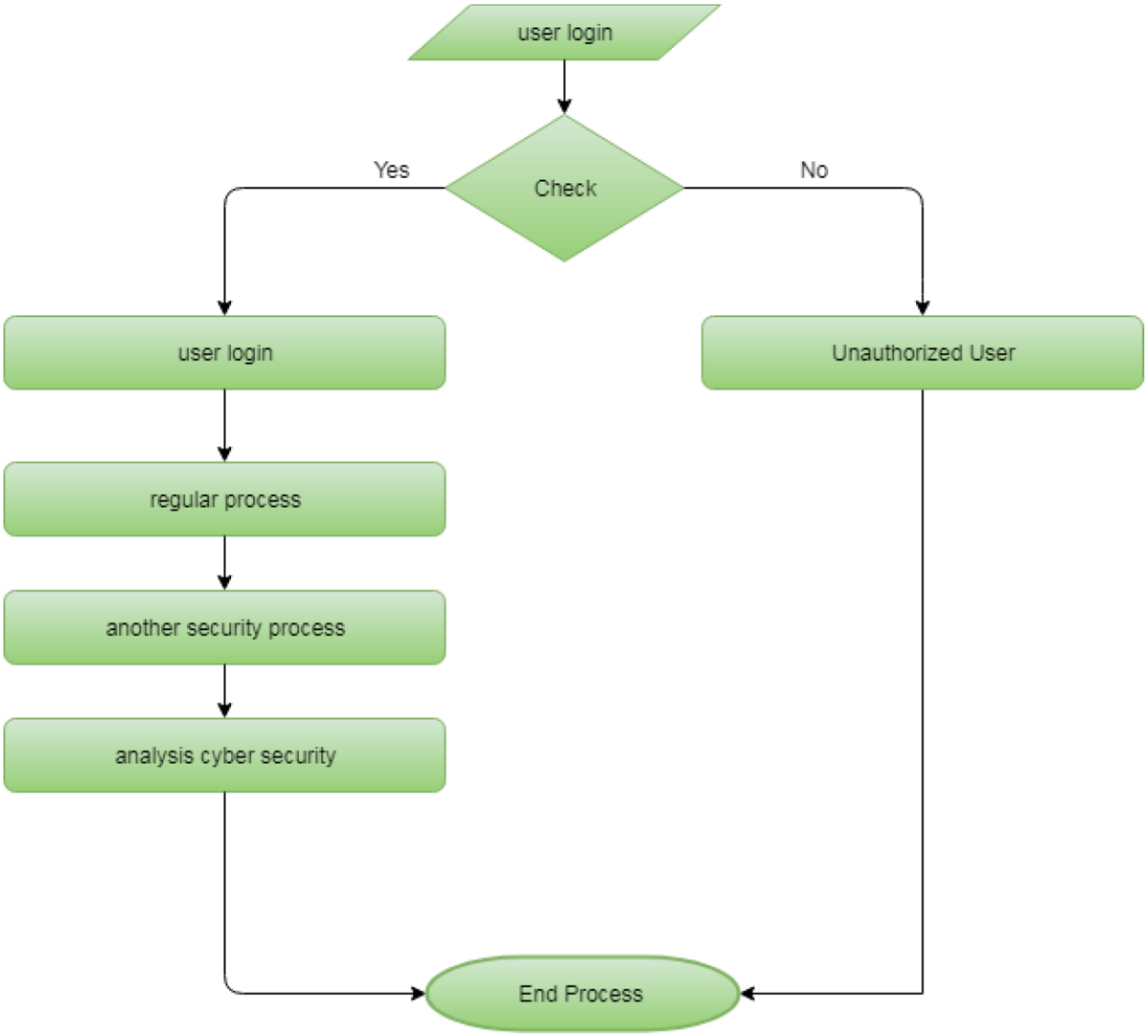
Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. **a.user**



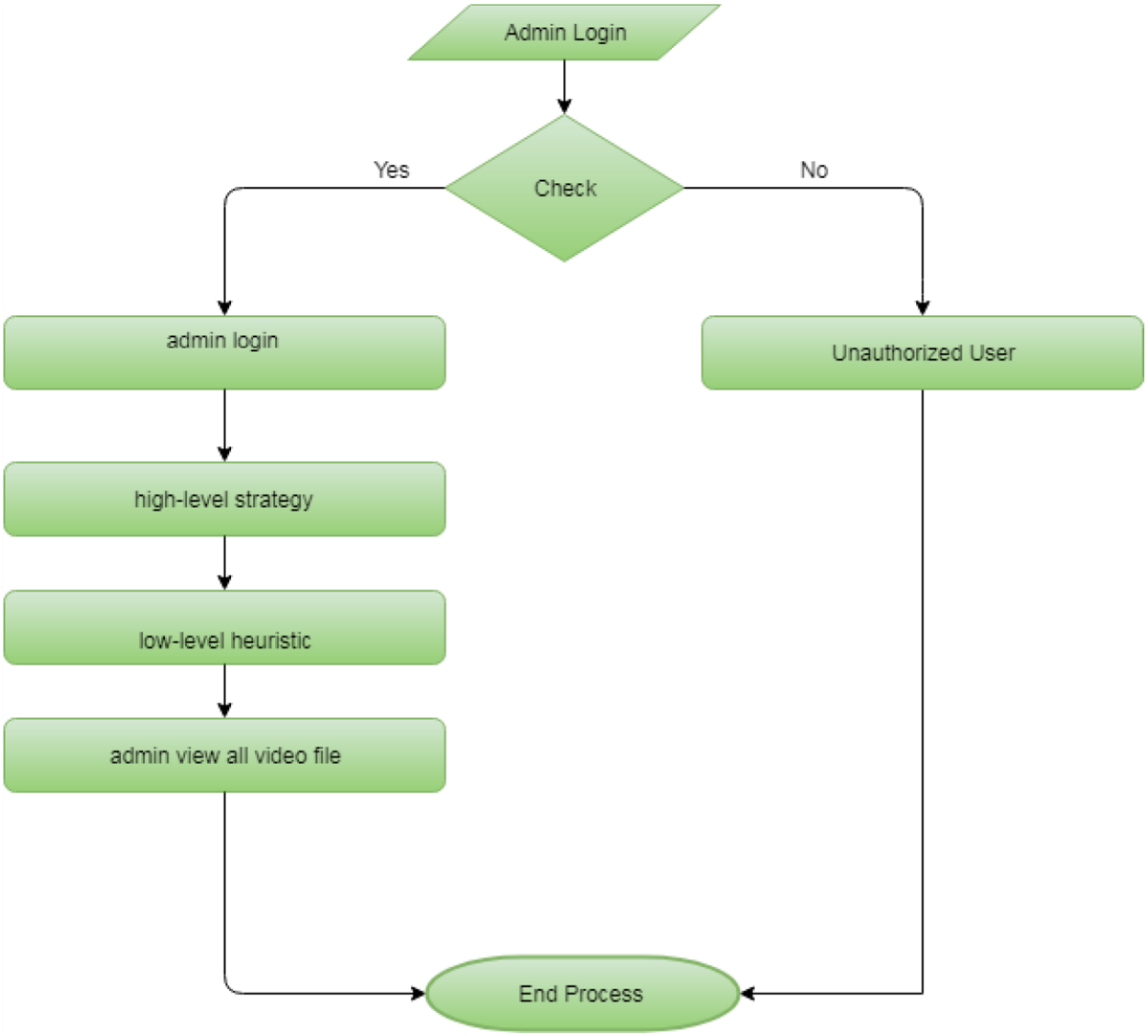
6.DATA FLOW DIAGRAM

A **data-flow diagram** is a way of representing a **flow** of **data** through a process or a system (usually an information system). The **DFD** also provides information about the outputs and inputs of each entity and the process itself. ... When using **UML**, the activity **diagram** typically takes over the role of the **data-flow diagram**.

a.User:


user login
Check
Yes
No
user login
Unauthorized User
regular process
another security process
analysis cyber security
End Process

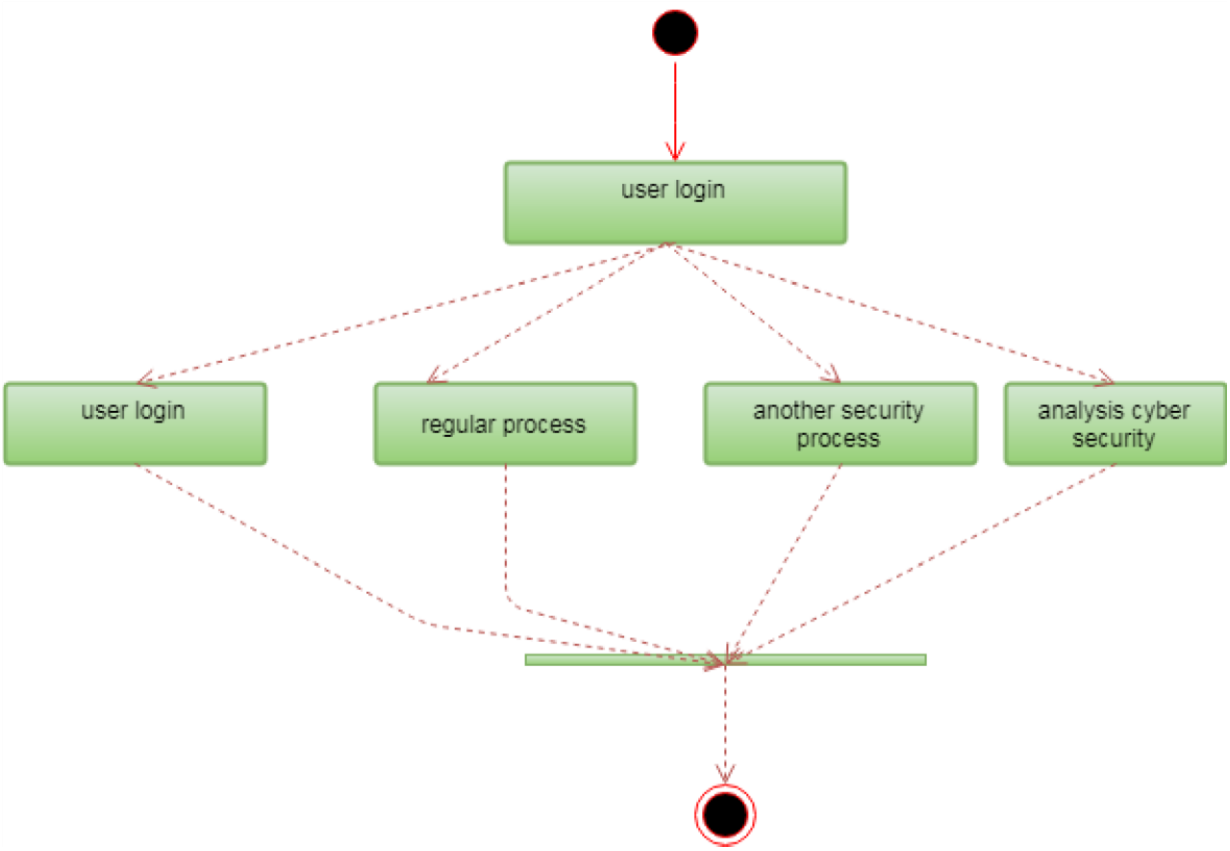
b.Admin:



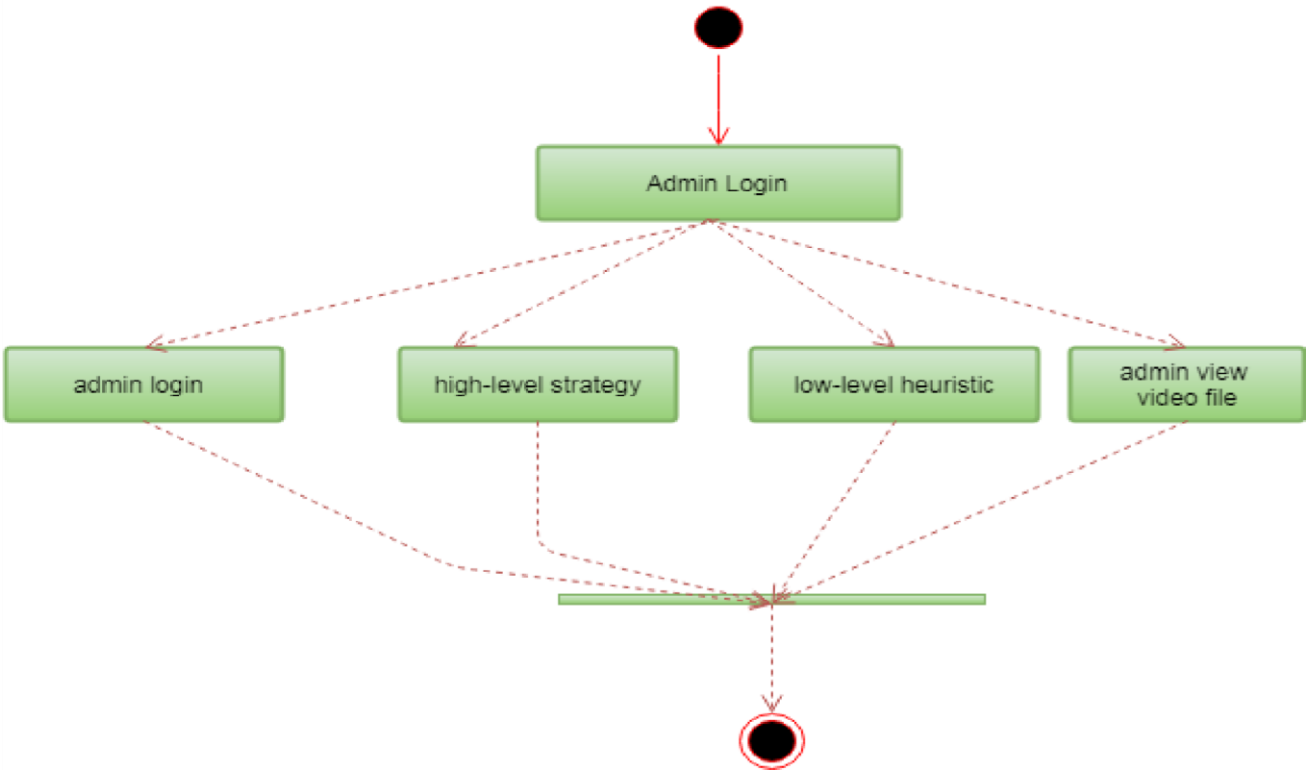
7.ACTIVITY DIAGRAM

Activity diagram is another important **diagram in UML** to describe the dynamic aspects of the system. **Activity diagram** is basically a flowchart to represent the flow from one **activity** to another **activity**. The **activity** can be described as an operation of the system

a.User:


user login
user login
regular process
another security process
analysis cyber security

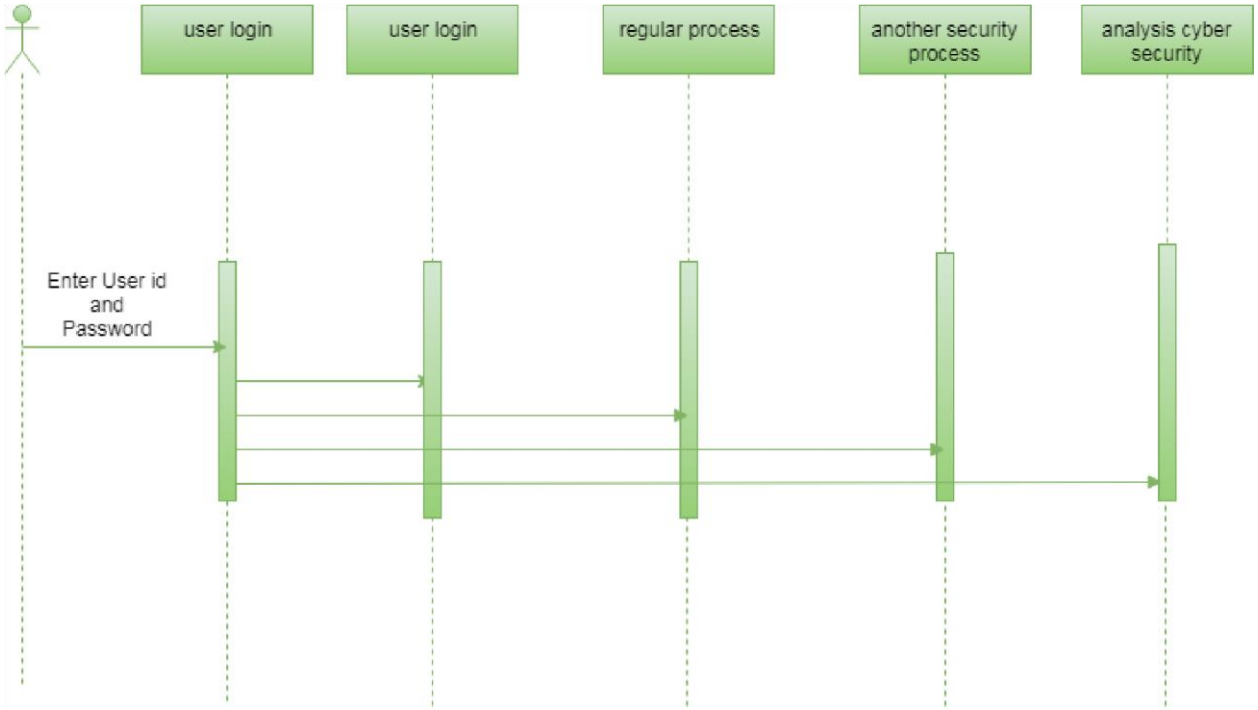
b.Admin:



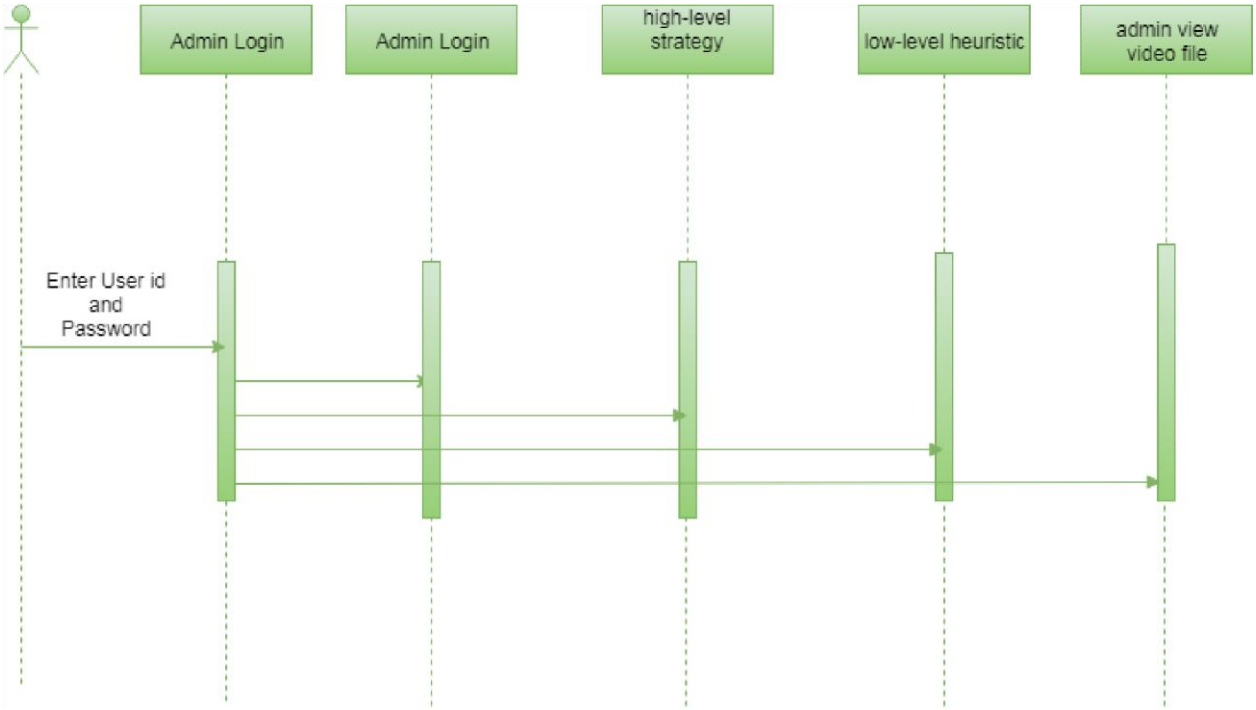
8.SEQUENCE DIAGRAM

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

a.User:


user login
user login
regular process
another security process
analysis cyber security
Enter User id and Password

b.Admin:

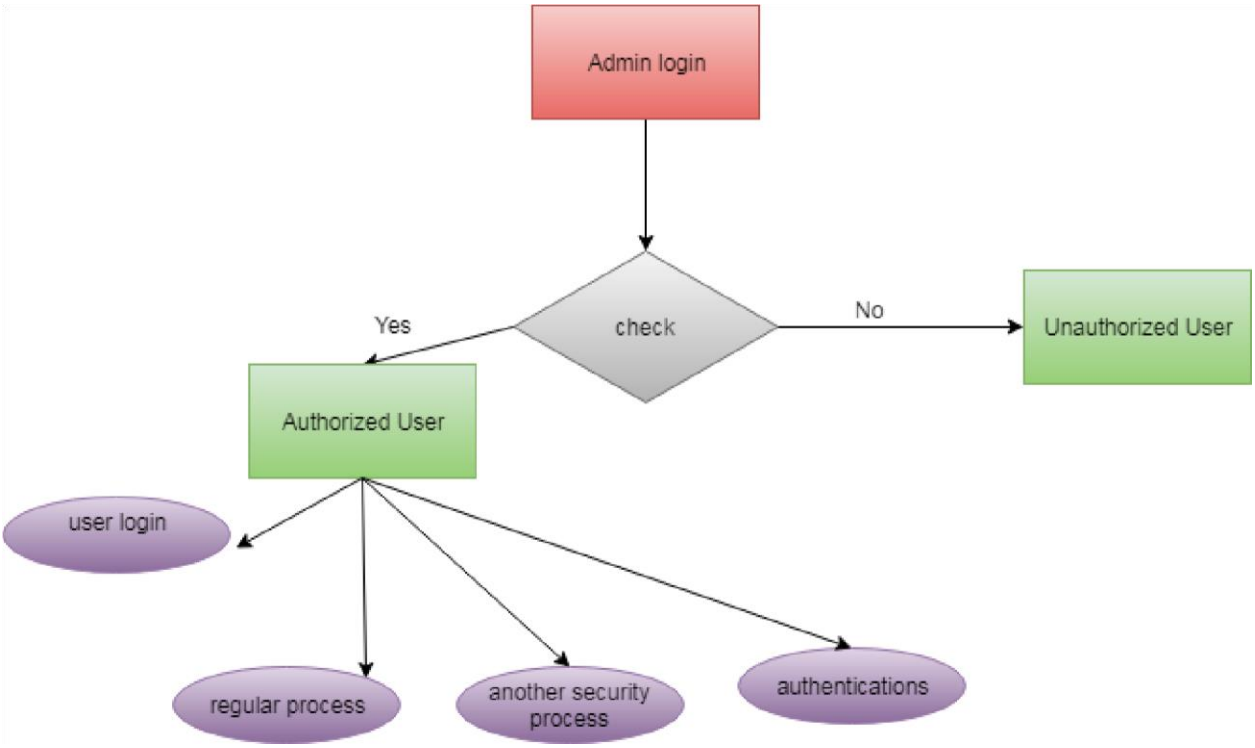


4.4 E-R DIAGRAM

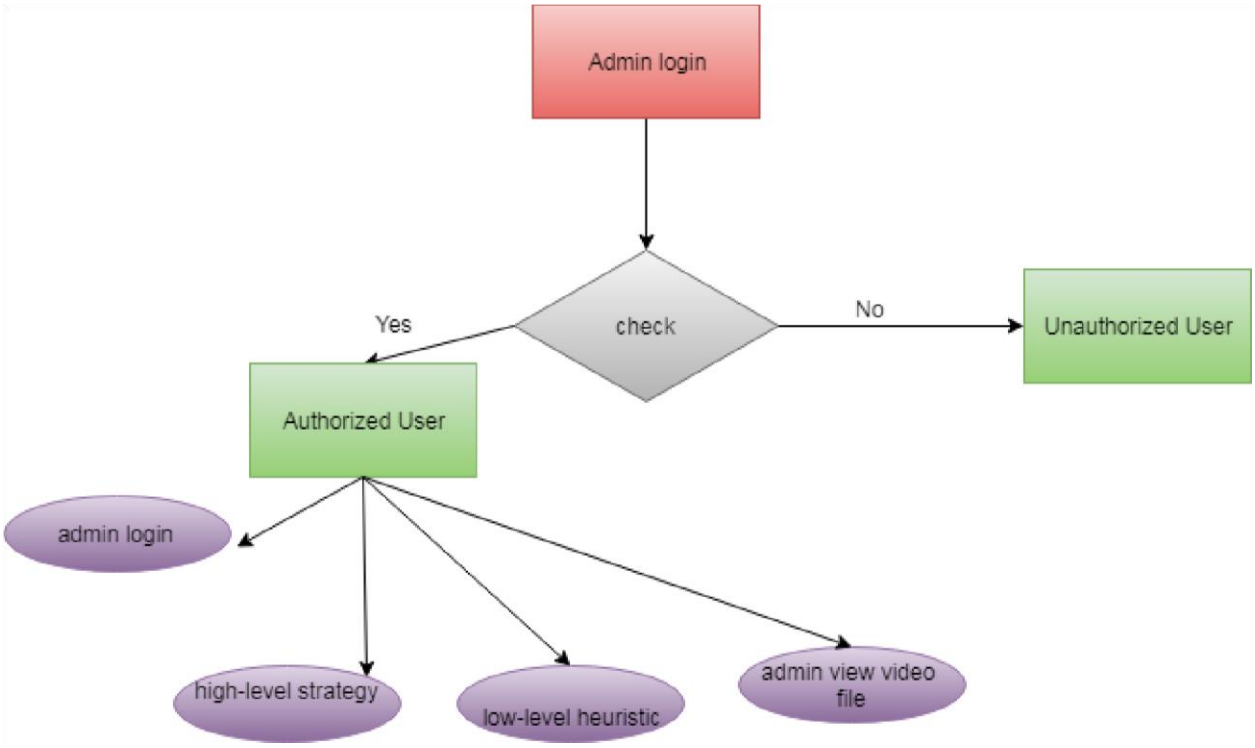
An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

ER DIAGRAM OF MODEL

a.User:



b.Admin



4.5 INPUT AND OUTPUT DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 4

SOFTWARE REQUIREMENTS & ALGORITHMS

4.1 INTRODUCTION TO PYTHON

Python is a widely used high-level programming language, created by Guido Van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using white space indentation to delimit code blocks rather than curly brackets or keywords) and syntax that allows the programmer to express concepts in fewer lines of code that might be used in programming languages such as C++ or Java. It provides constructs that enable to clear programming on both small and large scales.

Python features a dynamic type system & automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. The reference implementation of Python is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

Programming languages have been for ages, and every decade sees the launch of a new language sweeping the developers off their feet. Below are the major features and applications due to which people choose Python as their first programming language:

- Python's popularity and high salary.
- Python is used in Data Science.
- Python's scripting and automation.
- Python used with Big Data.
- Python supports Testing.
- Computer Graphics in Python.
- Python used in Artificial Intelligence.
- Python in Web Development.
- Python is portable and extensible.
- Python is simple and easy to learn.

4.2 HISTORY OF PYTHON:

Python was conceived in the late 1980's and its implementation began in December 1989 by Guido van Rossum at Centrum Wiskunde and Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system.

Guido Van Rossum remains Python's principal author. His continuing central role in Python's development is reflected in the title 'Benevolent Dictator for Life (BDFL)' given to him by the Python community.

Python 2.0 was released on 16 October 2000 and had many major new features, including cycle-detecting garbage collector. With this release, the development process became more transparent.

Python 3.0 (initially called Python 3000 or py3k) was released on 3 December 2008 after a long testing period. It is a major version of the language that is not backward compatible with previous versions. However, many of its major features have been backported to the backward compatible Python 2.6.x and 2.7.x version series.

Python 2.7's end-of-life date (a.k.a. EOL, sunset date) was initially set at 2015, and then postponed to 2020 out of concern that a large body of existing code could not easily be forward ported to Python 3.x.

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time rather, operations on an object may fail, signifying that the given object is not a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows the programmer to define their own types using classes, which are most often used for Object-Oriented Programming (OOP). New instances of classes are constructed by calling the class.

4.3 FEATURES OF PYTHON:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. It 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 library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows the interactive testing and debugging of code snippets.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** We can add low-level modules to the Python interpreter. These modules enable the programmers to add 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.
- 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 the large applications.
- It provides very high-level dynamic datatypes and supports dynamic type checking.
- It supports automatic garbage collection.

It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

4.4 INSTALLING OF PYTHON:

DOWNLOADING OF PYTHON:

To download Python we have to follow the below steps:

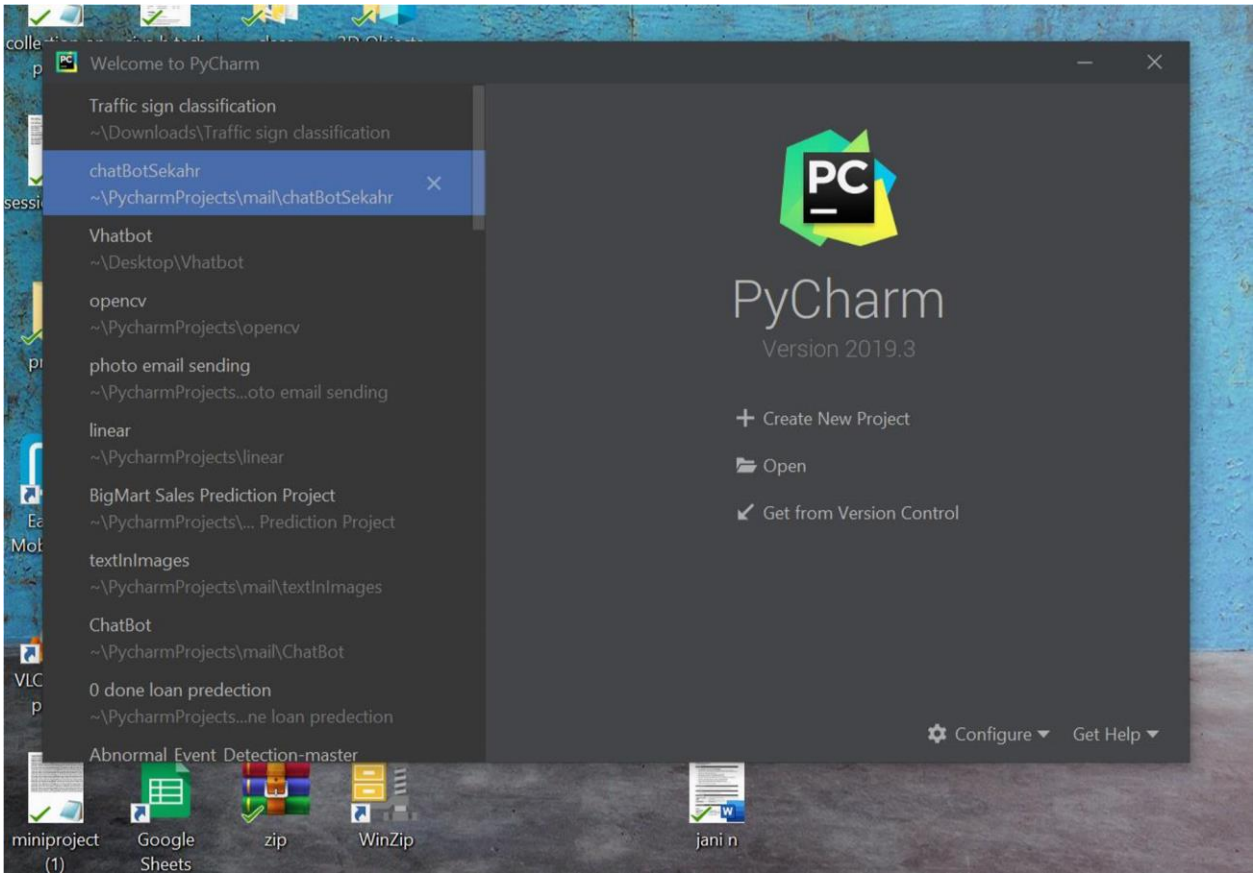
- Go to www.python.org
- Go to the downloads tab.
- Click on the Python version you required to download according to the OS you are using.
- Double click on the downloaded Python version.

- Select the path for Python.
- Click on install now.
- Click on yes.
- Click on close once it was done.
-

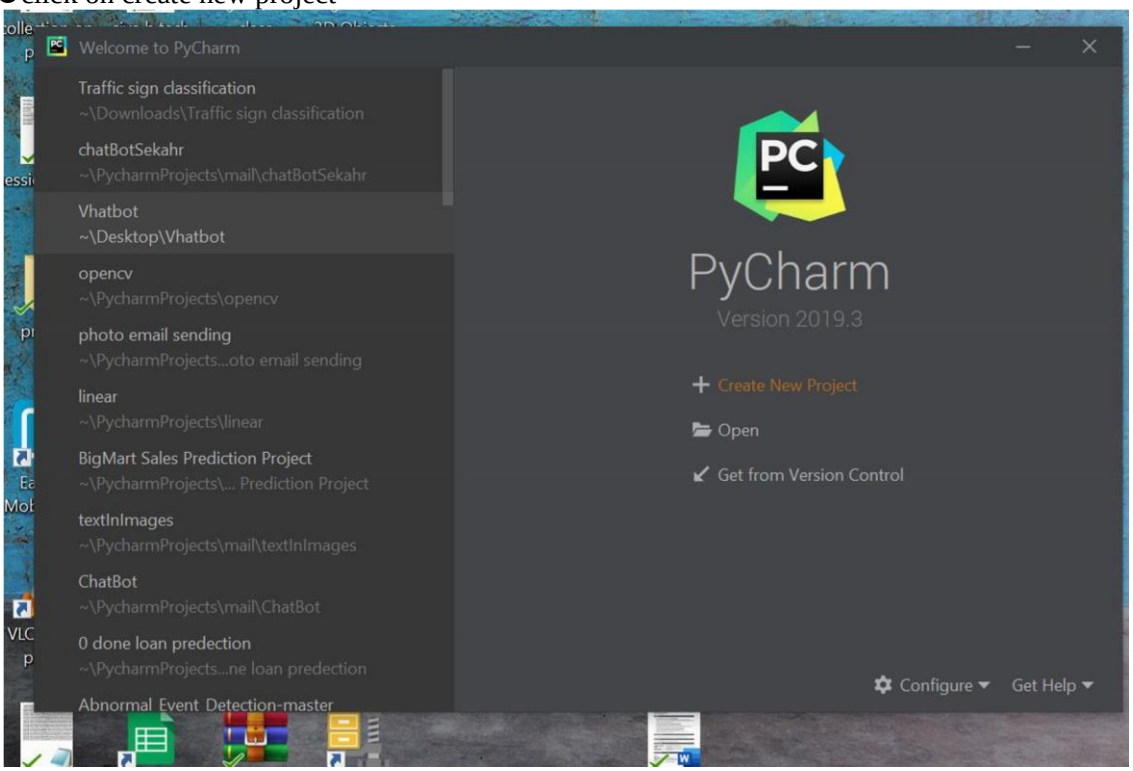
4.5 INSTALLING OF PYTHON:

Python distribution is available for a wide variety of platforms. You need to download only the binary code which is applicable for your platform (OS) and install Python. If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Here is a quick overview of installing Python on Windows platform.

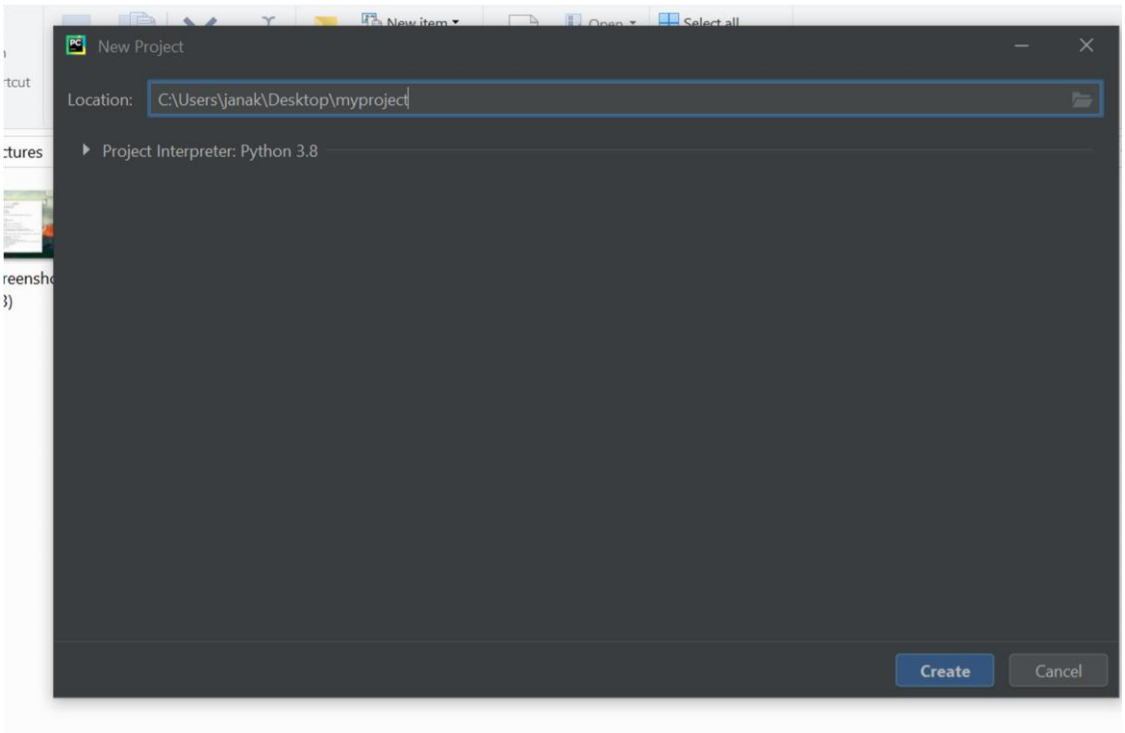
4.6 PROJECT CREATION STEPS



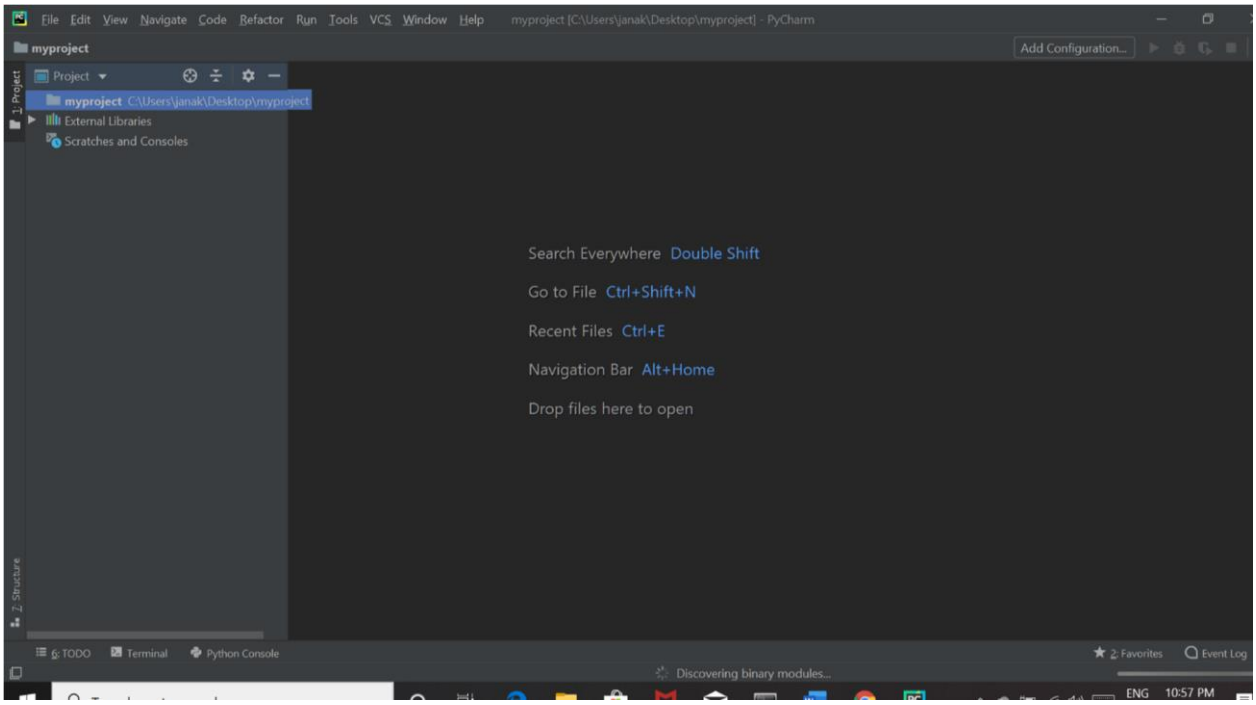
❼click on create new project



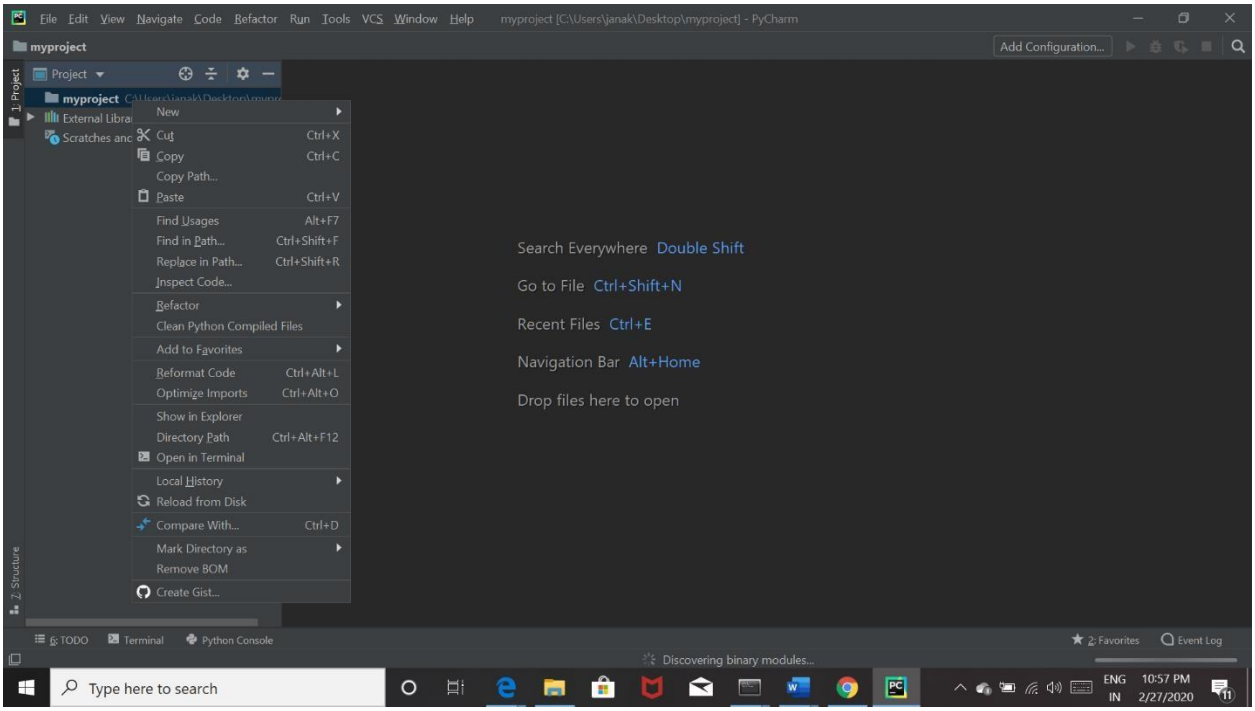
❼Write project name



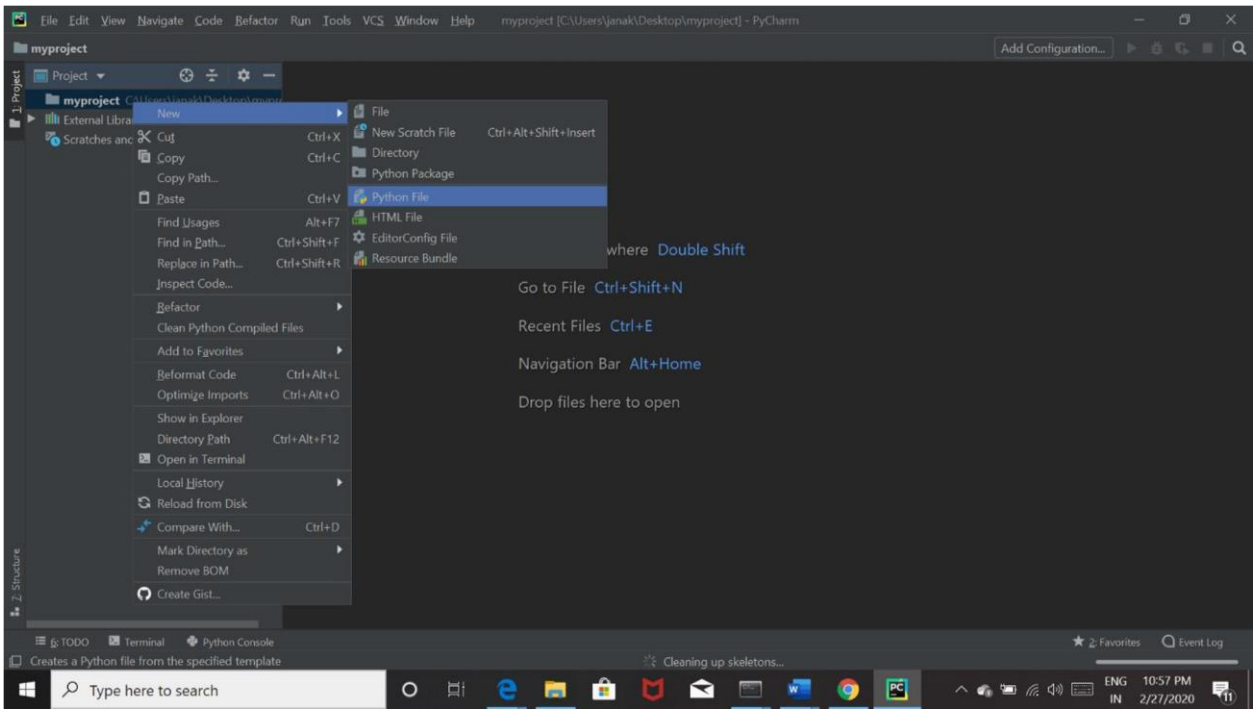
❼Right Click on project name



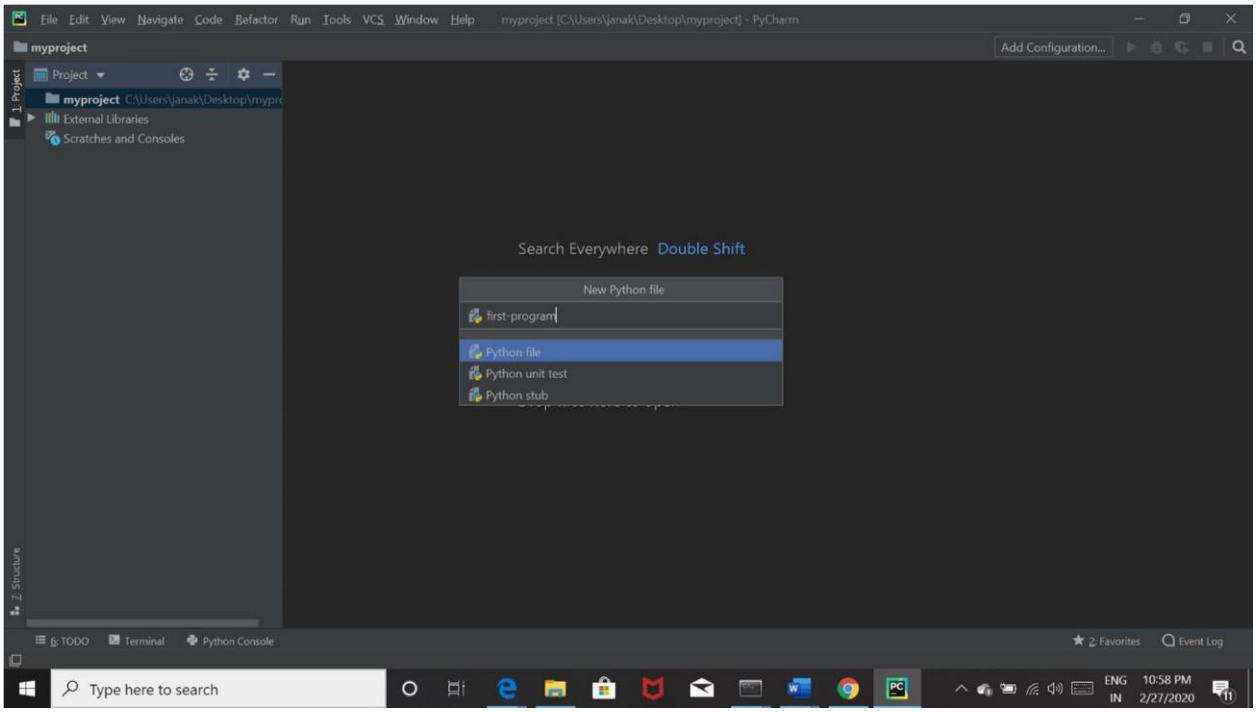
❾ Click on new



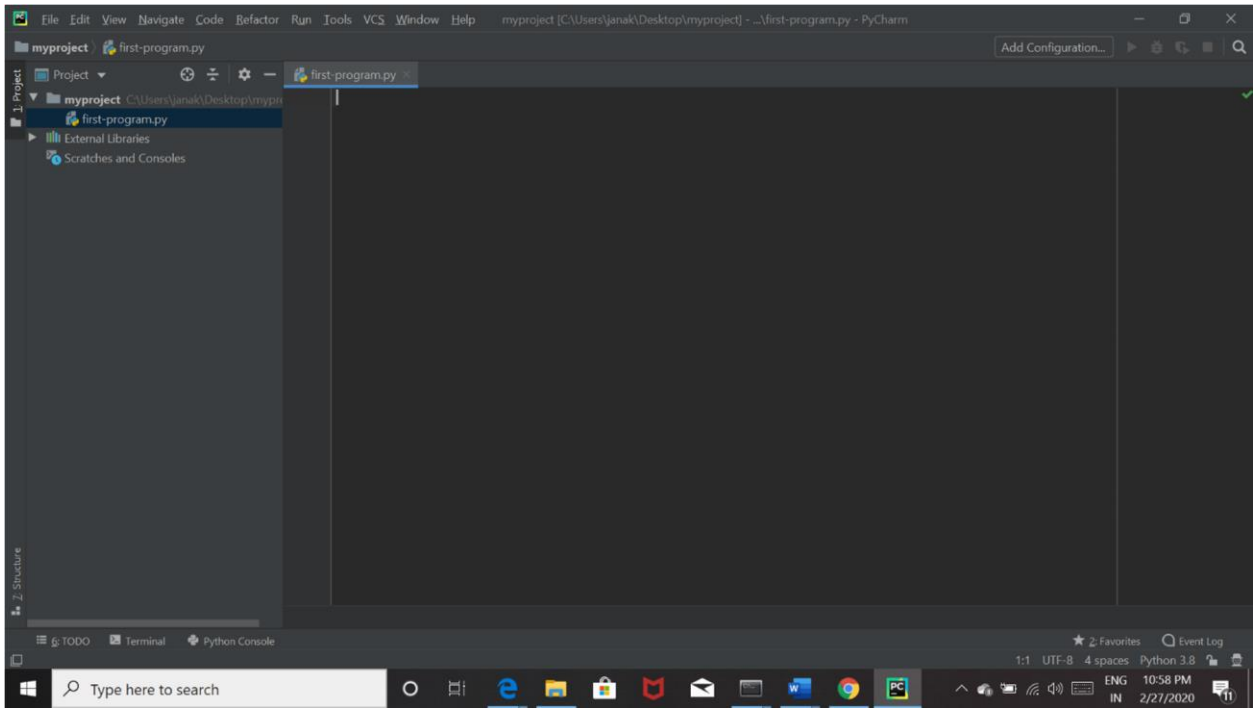
❼click on python file



❼write file name



-❼write the code



OpenCV :

OpenCV is an open source computer vision library accessible in python coding language to code for visionary capabilities of our smart pc. OpenCV was expected for computational capability and having a high focus on ongoing picture location and distinguishing proof. OpenCV is coded with streamlined C and can take work with multicore processors. If we need progressively programmed improvement

utilizing Intel models [Intel], you can purchase Intel's Integrated Performance Primitives (IPP) libraries [IPP]. These comprise of low-level schedules in different algorithmic regions which are streamlined. OpenCV consequently utilizes the IPP library, at runtime if that library is introduced. alarm beeps and the speed of the vehicle is automatically reduced.

The Computer's Vision PC's vision is the change of information from a still, or camcorder into either a depiction or another choice. Each and every such changes are performed to achieve a particular target. A Computer gains a cross section of numbers from a camera or from the circle, and it's just as simple as that. For the most part, there is no worked in example acknowledgment or programmed control of center and gap, no cross-relationship with long periods of experience. Generally, vision frameworks are still reasonably gullible. The Origin of OpenCV OpenCV left an Intel Research action proposed to drive CPU-raised applications.Toward this end, Intel moved various endeavors that included constant beam following and moreover 3D show dividers. One of the product engineers working for Intel at the time was visiting schools. He saw that several top school social affairs, like the MIT Media Lab, used to have well-made similarly as inside open PC vision frameworks—code which was supplied starting with one understudy then onto the next and which gave each resulting understudy an important establishment while building up his own vision application. Rather than rehashing the fundamental capacities from starting, another understudy may begin by adding to that which preceded .

OpenCV Structure and Content:

OpenCV left an Intel Research movement planned to drive CPU-raised applications.Toward this end, Intel pushed various endeavors that included continuous beam following and moreover 3D show dividers. One of the product engineers working for Intel at the time was visiting schools. He saw that two or three top school social events, like the MIT Media Lab, used to have well-made similarly as inside open PC vision foundations

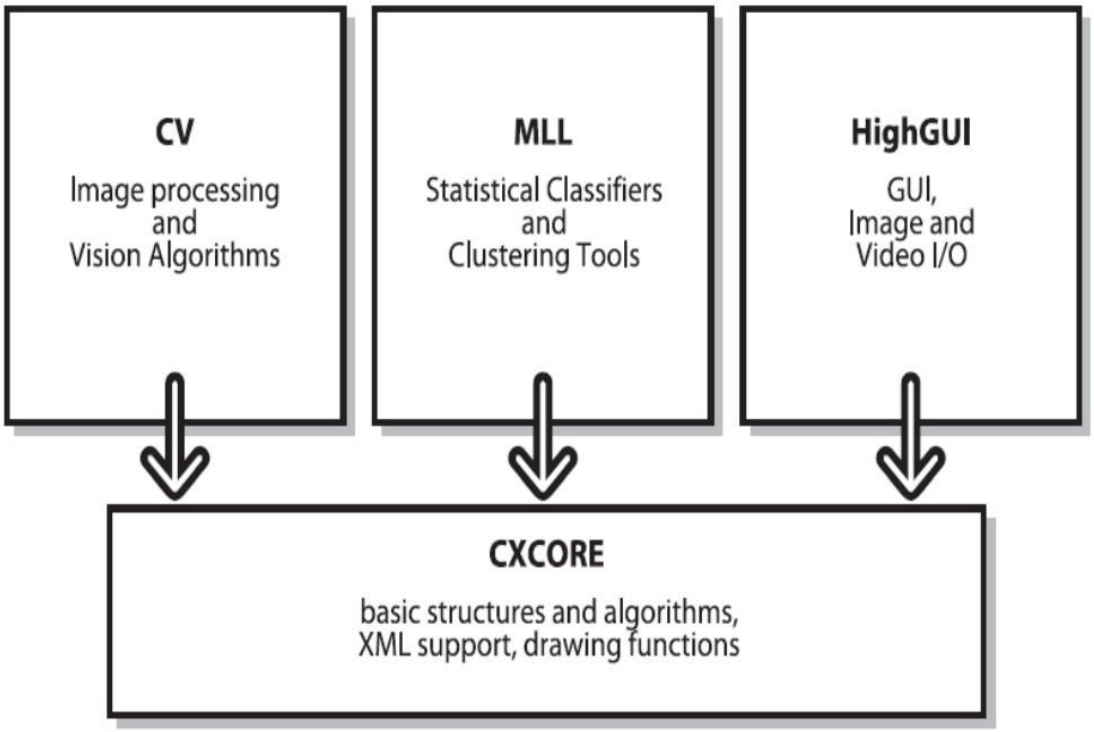


Fig: Parts of OpenCV

Why Open CV?

Specific :OpenCV was planned for picture handling. Each capacity and information structure has been arranged in perspective on an Image Processing application. Then, Matlab, is very conventional. You can get almost everything on the planet by methods for tool compartments. It may be money related tool stash or then again concentrated DNA tool compartments

- **Speedy** : Matlab is just excessively moderate. Matlab itself depended on Java. Similarly Java depended on C. So when we run a Matlab program, our PC gets caught up with attempting to translate and assemble all that convoluted Matlab code. At that point it is transformed into Java, lastly executes the code. In case we use C/C++, we don't waste such time. We direct give machine language code to the PC, and it gets executed. So in the end we get more picture taking care of, and not additionally interpreting. Ensuing to doing some constant picture handling with both Matlab and OpenCV, we typically got low speeds, a point of confinement of around 4-5 outlines arranged each second with Matlab. With OpenCV in any case, we get genuine persistent dealing with at around 30 outlines being handled every second. 22Beyond any confusion we give the prize for speediness – a progressively enigmatic language to handle, yet it's unquestionably of true worth . We can complete a lot more work, as calculate some extremely perplexing arithmetic on pictures utilizing C and still pull off adequatespeeds for your application.

- **Efficient**:

Matlab utilizes just an excessive amount of system assets. With OpenCV, we can pull off as pitiful as 10mb RAM for a constant application. Notwithstanding the way that with the present PCs, the RAM factor is surely not a noteworthy thing to be worried over. In any case, our tiredness identification framework is to be used inside a vehicle in a way that is non-meddlesome and little; so a low handling necessity is vital. Subsequently we shall perceive as to how OpenCV is superior for a real-time drowsiness detection system

CHAPTER 5

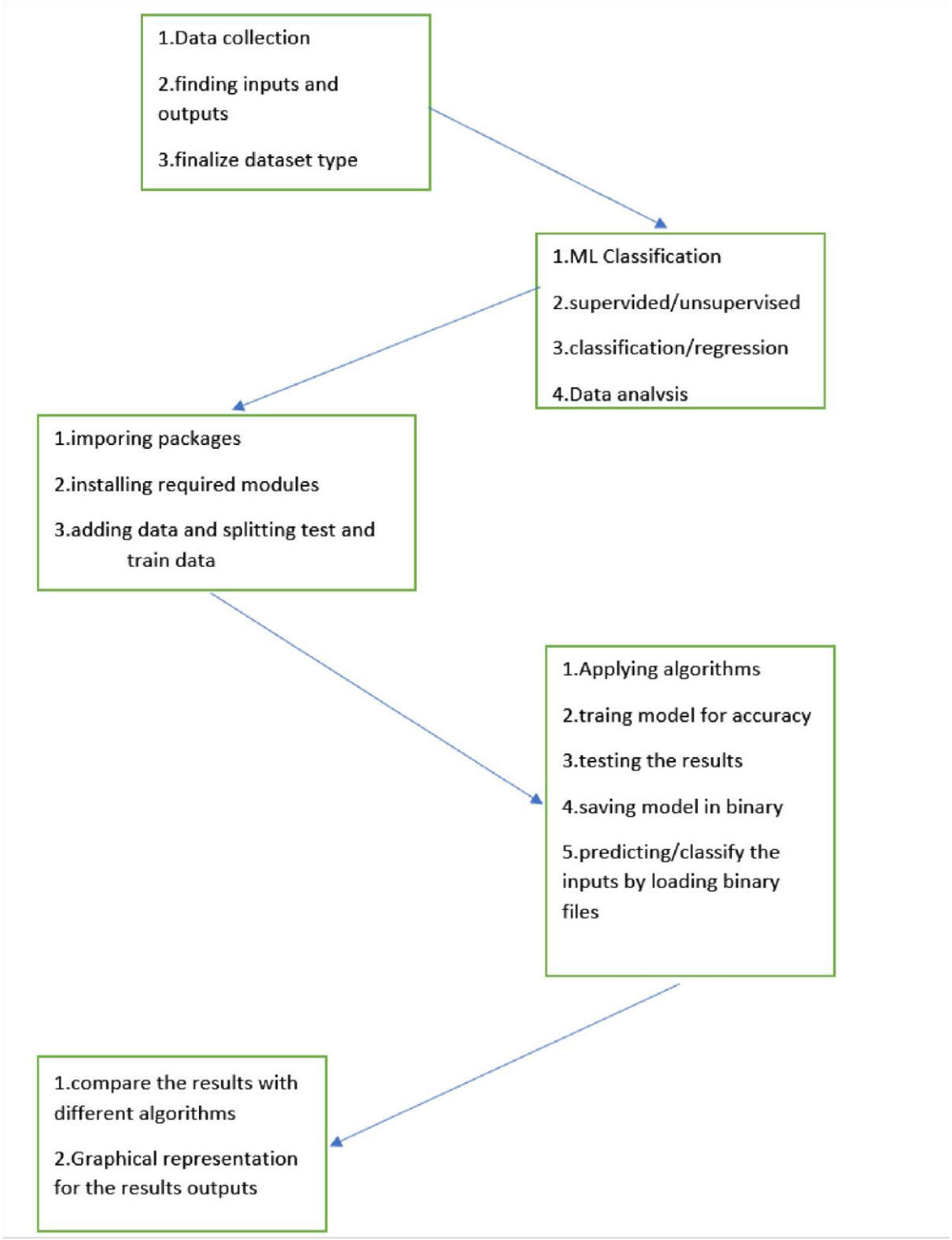
SYSTEM DESIGN & TESTING

5.1 INTRODUCTION

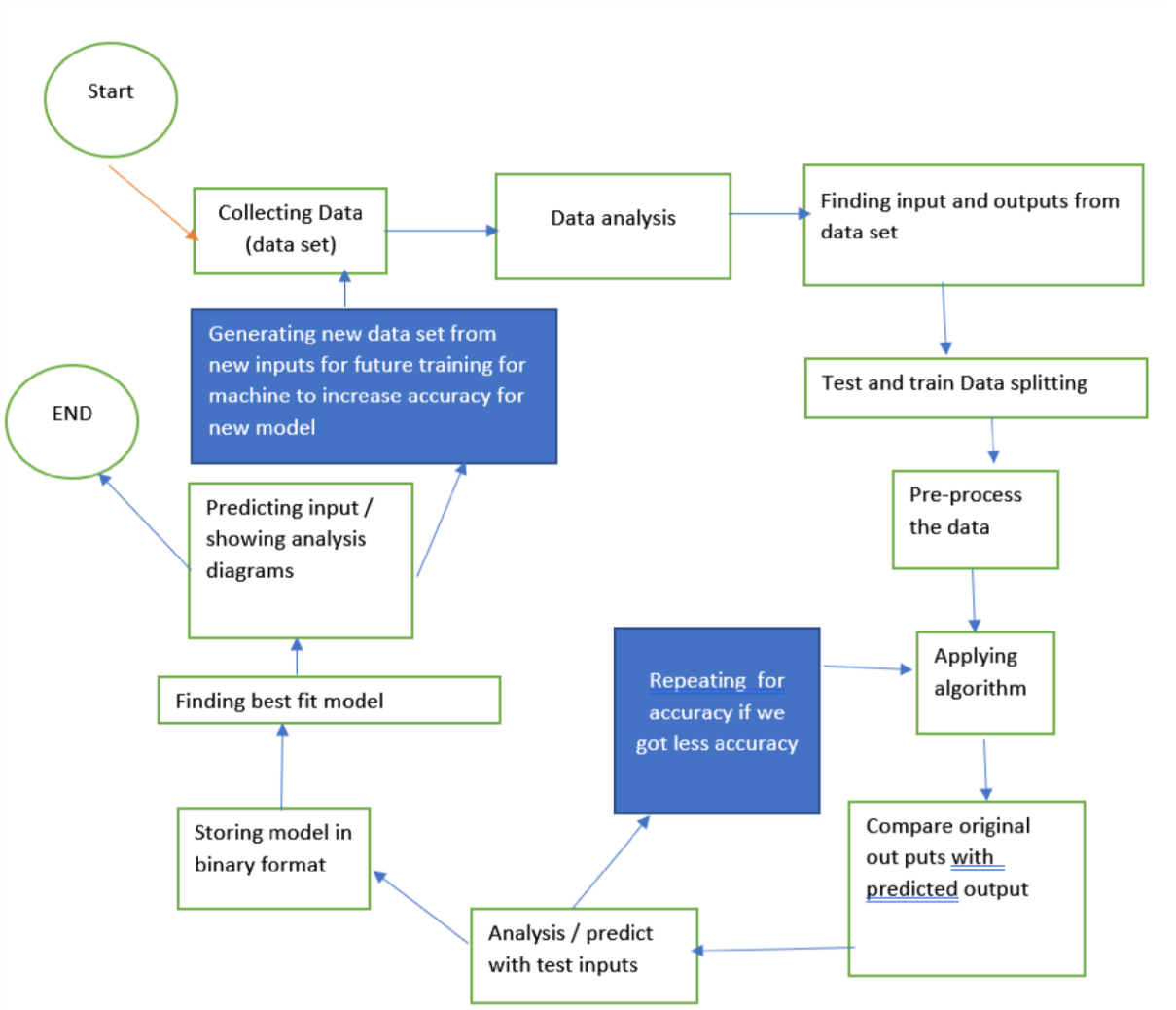
The purpose of the design phase is to plan a solution of the problem specified by the requirement document. It is the process of defining software methods, functions, objects and overall structure and interaction of your code so that the resulting functionality will satisfy your users requirements. It allows you to do the best abstraction, to understand the requirements better and meet them better. This prevents redundancy and increases reusability. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed design takes us towards how to satisfy the needs. The design of a system is

perhaps the most critical factor affecting the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design. System Design also called top-level design sign aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. During detailed design, the internal logic of each of the modules specification in system design is decided. During this phase, the details of the data are usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented. In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules. During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.

5.2 ALGORITHM & IMPLEMENTATION



5.3 ALGORITHAM MODEL FLOW DIAGREAM



5.4 Machine Learning Testing

First of all, what are we trying to achieve when performing ML testing, as well as any software testing whatsoever?

- Quality assurance is required to make sure that the software system works according to the requirements. Were all the features implemented as agreed? Does the program behave as expected? All the parameters that you test the program against should be stated in the technical specification document.

- Moreover, software testing has the power to point out all the defects and flaws during development. You don't want your clients to encounter bugs after the software is released and come to you waving their fists. Different kinds of testing allow us to catch bugs that are visible only during runtime.

However, in machine learning, a programmer usually inputs the data and the desired behaviour, and the logic is elaborated by the machine. This is especially true for deep learning. Therefore, the purpose of machine learning testing is, first of all, to ensure that this learned logic will remain consistent, no matter how many times we call the program.

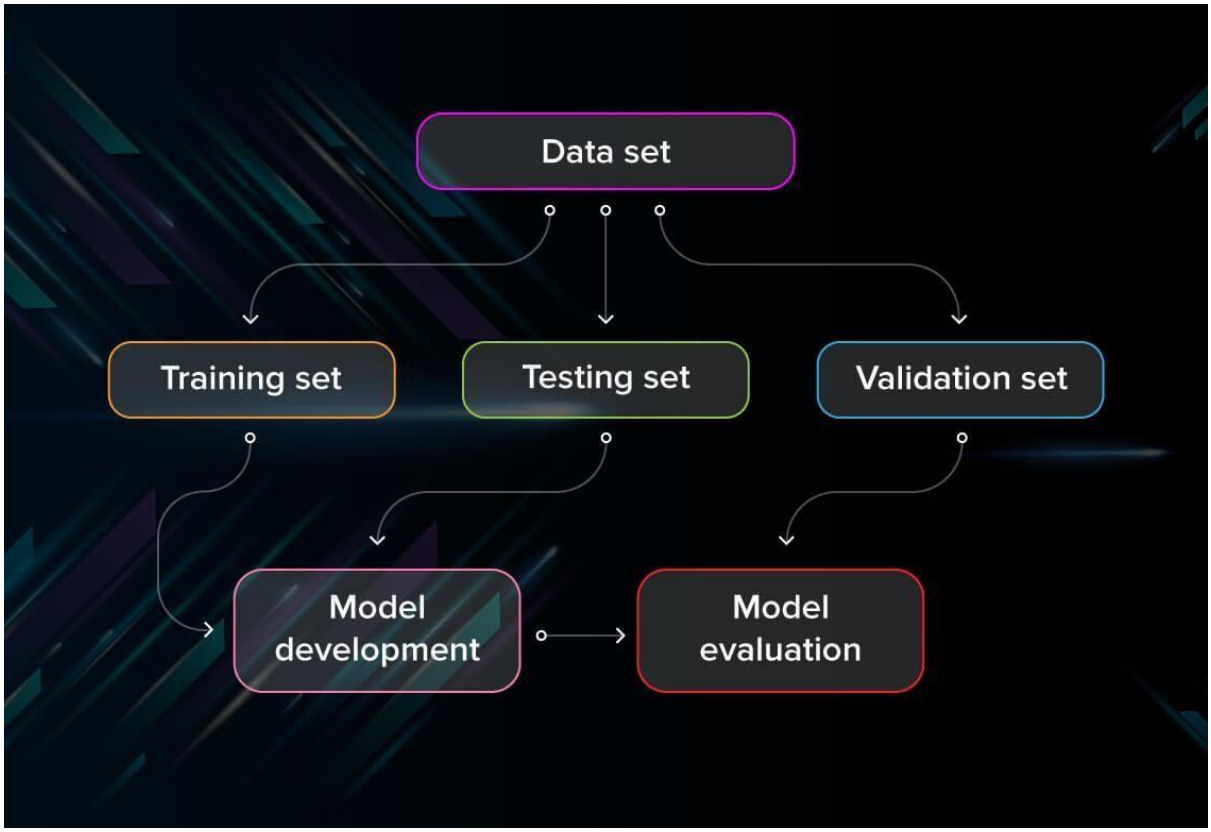
5.5 Model evaluation in machine learning testing

Usually, software testing includes:

- Unit tests. The program is broken down into blocks, and each element (unit) is tested separately.
- Regression tests. They cover already tested software to see if it doesn't suddenly break.
- Integration tests. This type of testing observes how multiple components of the program work together.

Moreover, there are certain rules that people follow: don't merge the code before it passes all the tests, always test newly introduced blocks of code, when fixing bugs, write a test that captures the bug.

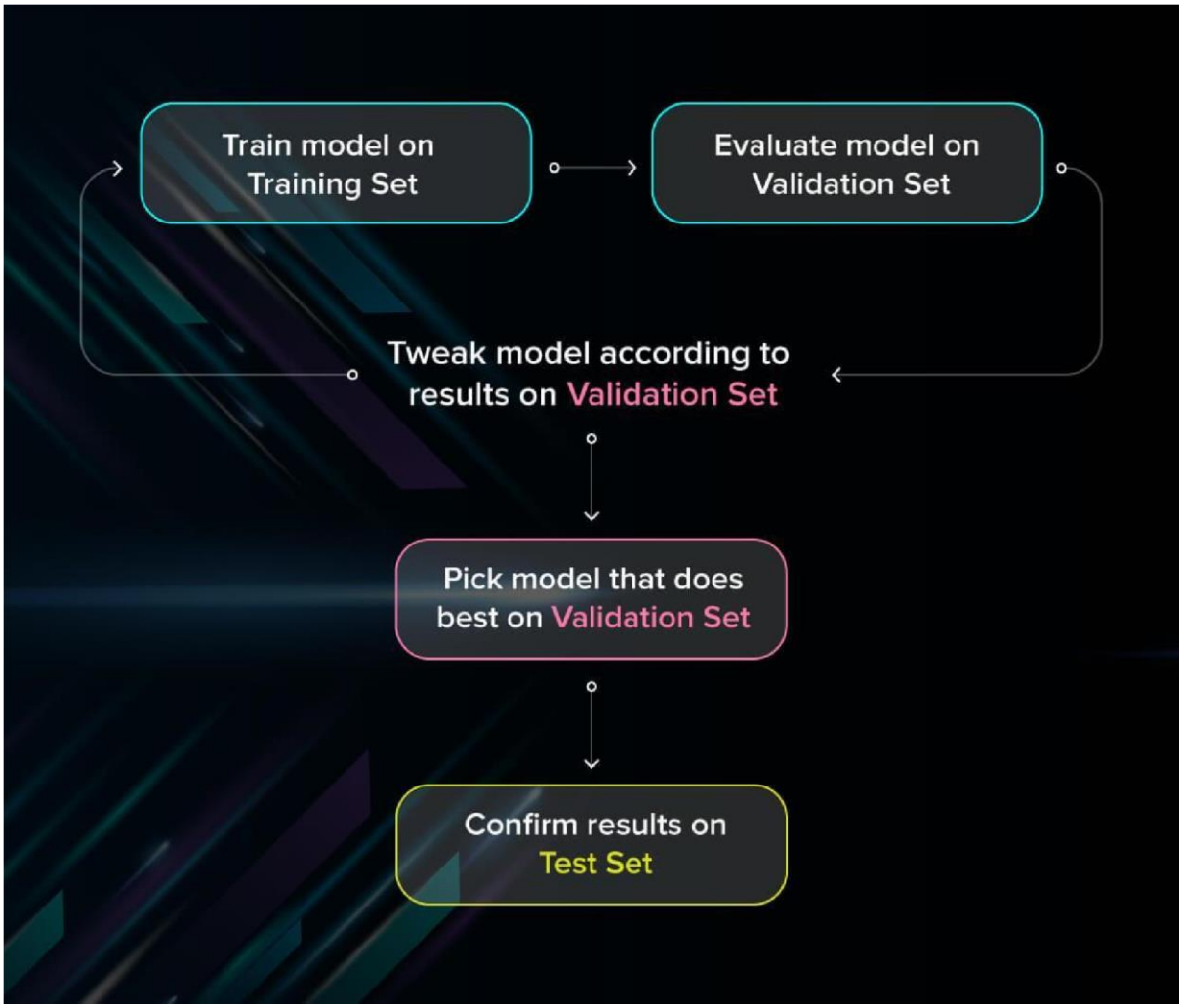
Machine learning adds up more actions to your to-do list. You still need to follow ML's best practices. Moreover, every ML model needs not only to be tested but evaluated. Your model should generalize well. This is not what we usually understand by testing, but evaluation is needed to make sure that the performance is satisfactory.



First of all, you split the database into three non-overlapping sets. You use a training set to train the model. Then, to evaluate the performance of the model, you use two sets of data:

- **Validation set.** Having only a training set and a testing set is not enough if you do many rounds of hyperparameter-tuning (which is always). And that can result in overfitting. To avoid that, you can select a small validation data set to evaluate a model. Only after you get maximum accuracy on the validation set, you make the testing set come into the game.

- **Test set (or holdout set).** Your model might fit the training dataset perfectly well. But where are the guarantees that it will do equally well in real-life? In order to assure that, you select samples for a testing set from your training set — examples that the machine hasn't seen before. It is important to remain unbiased during selection and draw samples at random. Also, you should not use the same set many times to avoid training on your test data. Your test set should be large enough to provide statistically meaningful results and be representative of the data set as a whole.



But just as test sets, validation sets "wear out" when used repeatedly. The more times you use the same data to make decisions about hyperparameter settings or other model improvements, the less confident you are that the model will generalize well on new, unseen data. So it is a good idea to collect more data to 'freshen up' the test set and validation set.

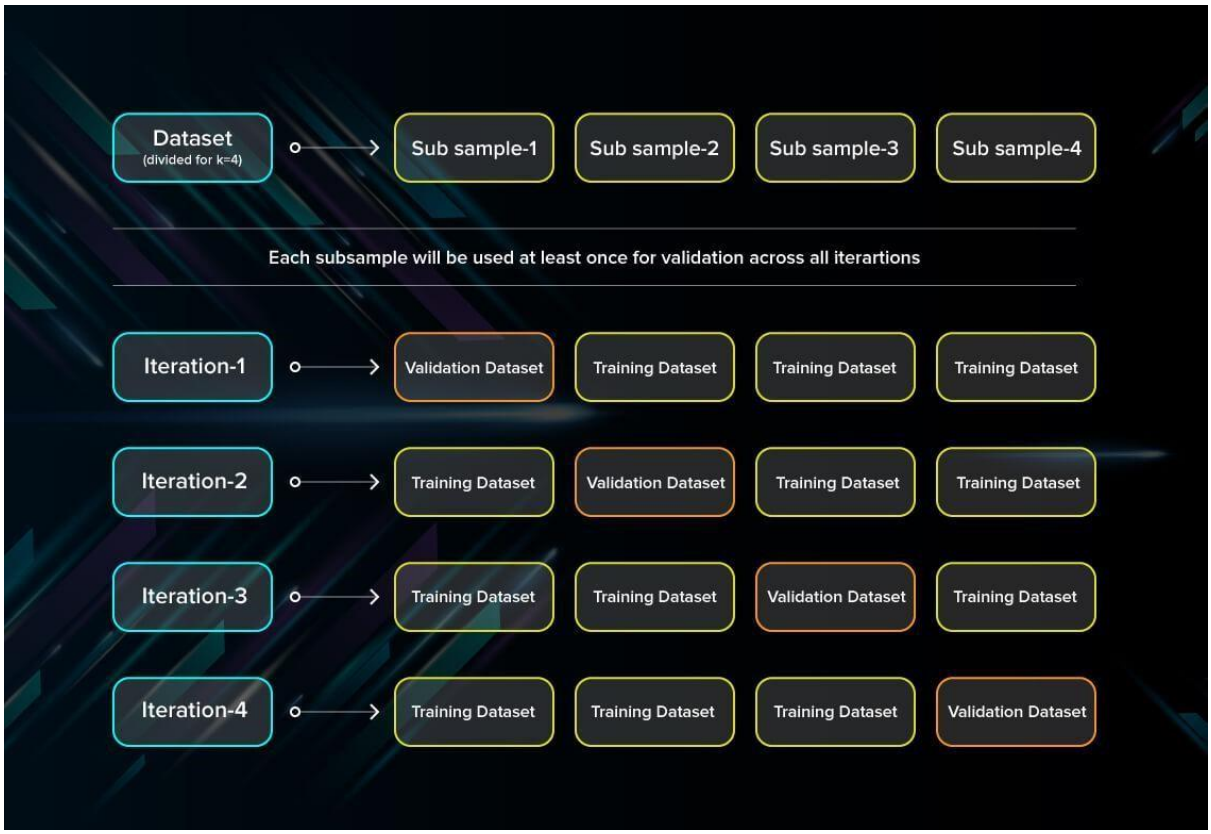
Cross-validation

Cross-validation is a model evaluation technique that can be performed even on a limited dataset. The training set is divided into small subsets, and the model is trained and validated on each of these samples.

k-fold cross-validation

The most common cross-validation method is called k-fold cross-validation. To use it, you need to divide the dataset into kk subsets (also called folds) and use them kk times. For example, by breaking the dataset into 10 subsets, you will perform a 10-fold cross-validation.

Each subset must be used as the validation set at least once.



This method is useful to test the skill of the machine learning model on unseen data. It is so popular because it is simple to apply, works well even with relatively small datasets, and the results you get are generally quite accurate. If you want to learn more about how to crossvalidate the model, check out a more detailed explanation on Medium.

Leave-one-out cross-validation

In this method, we train the model on all the data samples in the set except for one data point that is used to test the model. By repeating this process iteratively, each time leaving a different data point as a testing set, you get to test the performance for all the data.

The benefit of the method is low bias since all the data points are used. However, it also leads to higher variation in testing because we are testing the model against just one data point each time.

5.6 Evaluate models using metrics

Evaluating the performance of the model using different metrics is integral to every data science project. Here is what you have to keep an eye on:

Accuracy

Accuracy is a metric for how much of the predictions the model makes are true. The higher the accuracy is, the better. However, it is not the only important metric when you estimate the performance.

Accuracy≡True Positives+True NegativesTrue Positives+False Positives+True Negatives+Fa lse NegativesAccuracy≡True Positives+True NegativesTrue Positives+False Positives+True Negatives+False Negatives

Loss

Loss describes the percentage of bad predictions. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater.

Precision

The precision metric marks how often the model is correct when identifying positive results.
For example, how often the model diagnoses cancer to patients who really have cancer.

Precision≡True PositivesTrue Positives+False PositivesPrecision≡True PositivesTrue Positiv es+False Positives

Recall

This metric measures the number of correct predictions, divided by the number of results that should have been predicted correctly. It refers to the percentage of total relevant results correctly classified by your algorithm.

Recall≡True PositivesTrue Positives+False NegativesRecall≡True PositivesTrue Positives+F alse Negatives

Confusion matrix

A confusion matrix is an N×NN×N square table, where NN is the number of classes that the model needs to classify. Usually, this method is applied to classification where each column represents a label. For example, if you need to categorize fruits into three categories: oranges, apples, and bananas, you draw a 3×33×3 table. One axis will be the actual label, and the other will be the predicted one.

Pre-train tests

This type of test is performed early on and allows you to catch bugs before running the model. They do not need training parameters to be run. An example of a pre-train test is a program that checks whether there are any labels missing in your training and validation datasets.

Post-train tests

These tests are performed on a trained model and check whether it performs correctly. They allow us to investigate the logic behind the algorithm and see whether there are any bugs there. There are three types of tests that report the behavior of the program:

- Invariance tests. Using invariance tests, we can check how much we can change the input without it affecting the performance of the model. We can pair up input examples and check for consistency in predictions. For example, if we run a pattern recognition

model on two different photos of red apples, we expect that the result will not change much.

- Directional expectation tests. Unlike invariance tests, directional expectation tests are needed to check how perturbations in input will change the behavior of the model. For example, when building a regression model that estimates the prices of houses and takes square meters as one of the parameters, we want to see that adding extra space makes the price go up.
- Minimum functionality tests. These tests enable us to test the components of the program separately just like traditional unit tests. For example, you can assess the model on specific cases found in your data.

CHAPTER 6

Sample code

```
# import the necessary packages
```

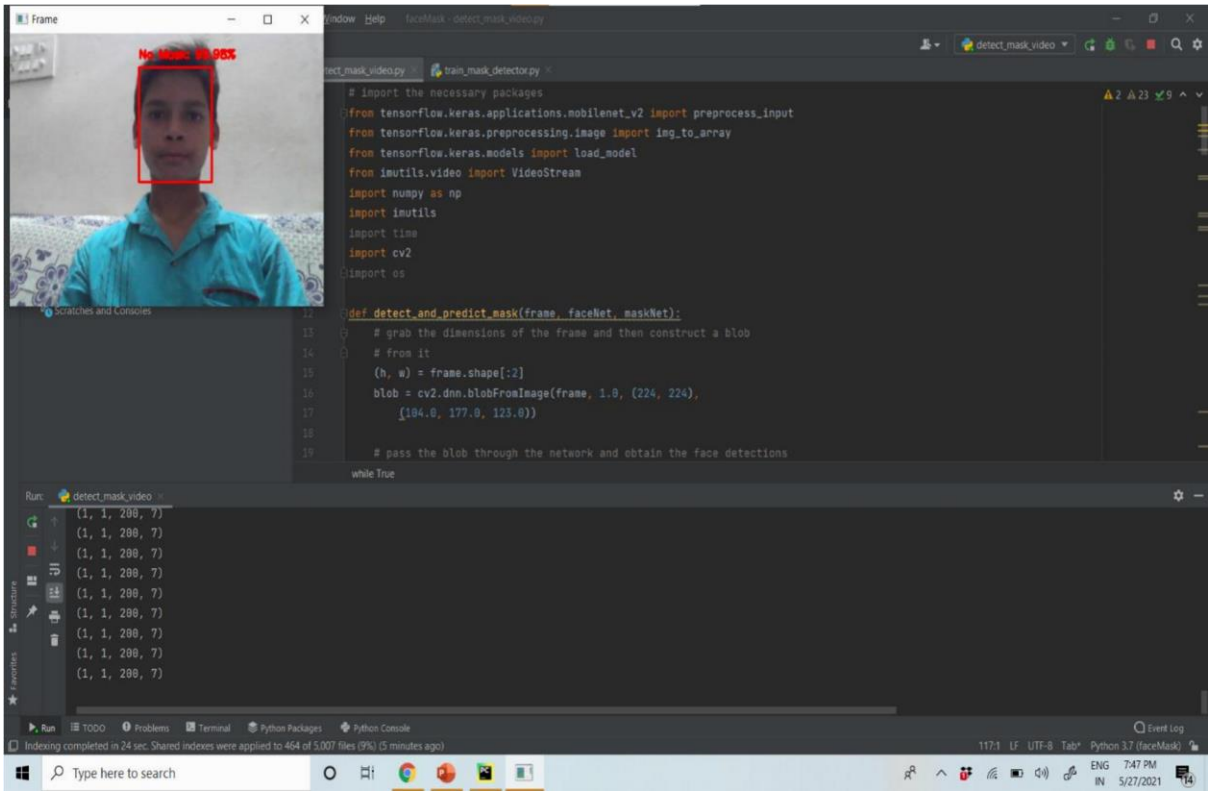
```
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array from
tensorflow.keras.models import load_model from imutils.video import
VideoStream import numpy as np import imutils import time import cv2
import os
 def detect_and_predict_mask(frame, faceNet,
maskNet):
   # grab the dimensions of the frame and then construct a blob
   # from it
   (h, w) = frame.shape[:2]
   blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
(104.0, 177.0, 123.0))

   # pass the blob through the network and obtain the face detections
faceNet.setInput(blob)    detections = faceNet.forward()
print(detections.shape)

   # initialize our list of faces, their corresponding locations,
# and the list of predictions from our face mask network    faces
= []    locs = []    preds = []

   # loop over the detections    for i in
range(0, detections.shape[2]):
      # extract the confidence (i.e., probability) associated with
      # the detection
      confidence = detections[0, 0, i, 2]

      # filter out weak detections by ensuring the confidence is
      # greater than the minimum confidence
if confidence > 0.5:
         # compute the (x, y)-coordinates of the bounding box for
         # the object
         box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
(startX, startY, endX, endY) = box.astype("int")
         # ensure the bounding boxes fall within the dimensions of
         # the frame
         (startX, startY) = (max(0, startX), max(0, startY))
(endX, endY) = (min(w - 1, endX), min(h - 1, endY))
         # extract the face ROI, convert it from BGR to RGB channel
# ordering, resize it to 224x224, and preprocess it          face =
frame[startY:endY, startX:endX]          face = cv2.cvtColor(face,
cv2.COLOR_BGR2RGB)          face = cv2.resize(face, (224, 224))
face = img_to_array(face)          face = preprocess_input(face)

         # add the face and bounding boxes to their respective
         # lists
         faces.append(face)
         locs.append((startX, startY, endX, endY))

   # only make a predictions if at least one face was detected
if len(faces) > 0:
      # for faster inference we'll make batch predictions on *all*
# faces at the same time rather than one-by-one predictions
      # in the above `for` loop
```

```
      faces = np.array(faces, dtype="float32")
preds = maskNet.predict(faces, batch_size=32)

   # return a 2-tuple of the face locations and their corresponding
   # locations
   return (locs, preds)

# load our serialized face detector model from disk prototxtPath =
r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk maskNet =
load_model("mask_detector.model")

# initialize the video stream
print("[INFO] starting video stream...") vs =
VideoStream(src=0).start()

# loop over the frames from the video stream while
True:
   # grab the frame from the threaded video stream and resize it
   # to have a maximum width of 400 pixels
frame = vs.read()
   frame = imutils.resize(frame, width=400)

   # detect faces in the frame and determine if they are wearing a
# face mask or not
   (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
   # loop over the detected face locations and their corresponding
   # locations    for (box, pred) in
zip(locs, preds):
      # unpack the bounding box and predictions
      (startX, startY, endX, endY) = box
      (mask, withoutMask) = pred
      # determine the class label and color we'll use to draw
      # the bounding box and text
      label = "Mask" if mask > withoutMask else "No Mask"
color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
      # include the probability in the label
      label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
      # display the label and bounding box rectangle on the output
      # frame
      cv2.putText(frame, label, (startX, startY - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
      cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

   # show the output frame
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF

   # if the `q` key was pressed, break from the loop
if key == ord("q"):
      break
```

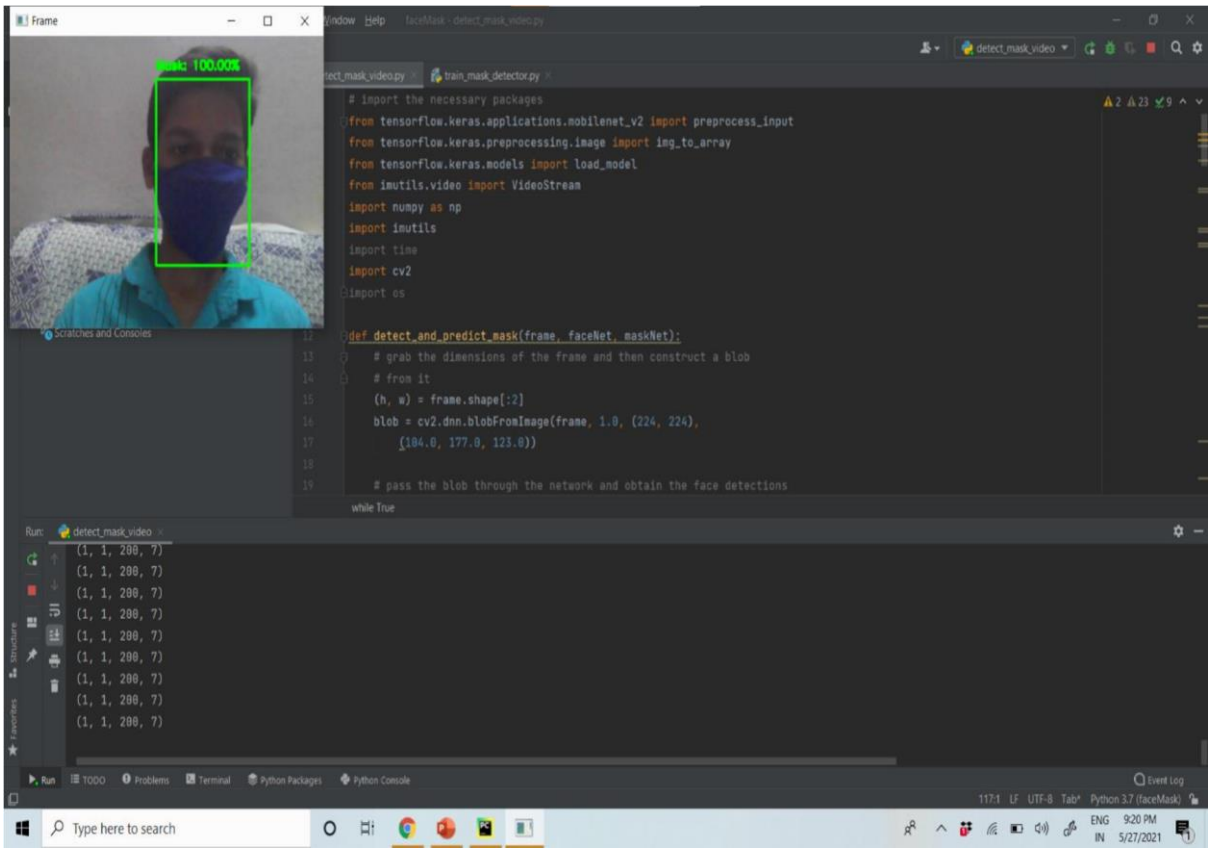
```
# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

OUTPUT SCREENS

FACE DETECTION WITHOUT MASK



FACE MASK DETECTION



CHAPTER 7

CONCLUSION

In this paper, we studied the problem of face-mask detection relevant in the scope of monitoring applications for the COVID-19 pandemic. We introduced a novel (annotation) dataset for studying face-mask detection problems and conducted an experimental study that looked at:

(i) the performance of existing face detectors with masked-face images.

(ii) the feasibility of recognition techniques aiming at the detection of properly worn facemasks.

(iii) the usefulness of existing face-mask detection models for monitoring applications in the fight against COVID-19. Our results showed that all tested detection models significantly deteriorate in performance when trying to detect masked faces compared to the performance observed with faces without masks. The most stable here was the RetinaFace model that also includes a generative component in the detection procedure. Furthermore, we observed that it is possible to design efficient techniques for recognizing faces with properly placed masked and that the selection of model architecture plays only a limited role in the final recognition performance. Finally, we demonstrated that existing models for face-masked detection have only limited value for real-life applications, as they only detect the presence of facial masks in the images, but not how these masks are placed. Because the tested models work well and in real time, we plan to integrate the best performing approaches into a real-world monitoring system. We also plan to extend our analysis to other datasets that contain a wider range of mask types. Since measures to contain the spread of

COVID-19 infections go in the direction that a certain group of people must use a certain type of mask, it would also make sense to design a classifier that can differentiate between different types of face-masks.