

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology Degree in
Information Technology

By

Name: IMMANUEL STANLEY S
(REG.NO:37120031)



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119. TAMIL NADU

AUGUST 2020



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" Grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119.

Phone: 044 - 2450 3150 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344

www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of IMMANUEL STANLEY S (Reg. No. 37120031) who underwent the professional training in "INTRANET VIDEO CONFERENCING" under our supervision from July 2020 to August 2020.

Internal Guide

DR.S. GOWRI, M.E., Ph.D.,

Head of the Department

Dr. R.SUBHASHINI, M.E.,Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **IMMANUEL STANLEY S (Reg no:37120031)** hereby declare that the Professional Training Report on **“INTRANET VIDEO CONFERENCING”** done by me under the guidance of guide **Dr.S.GOWRI** , Department of Information Technology at Sathyabama Institute of Science and technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Information Technology.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

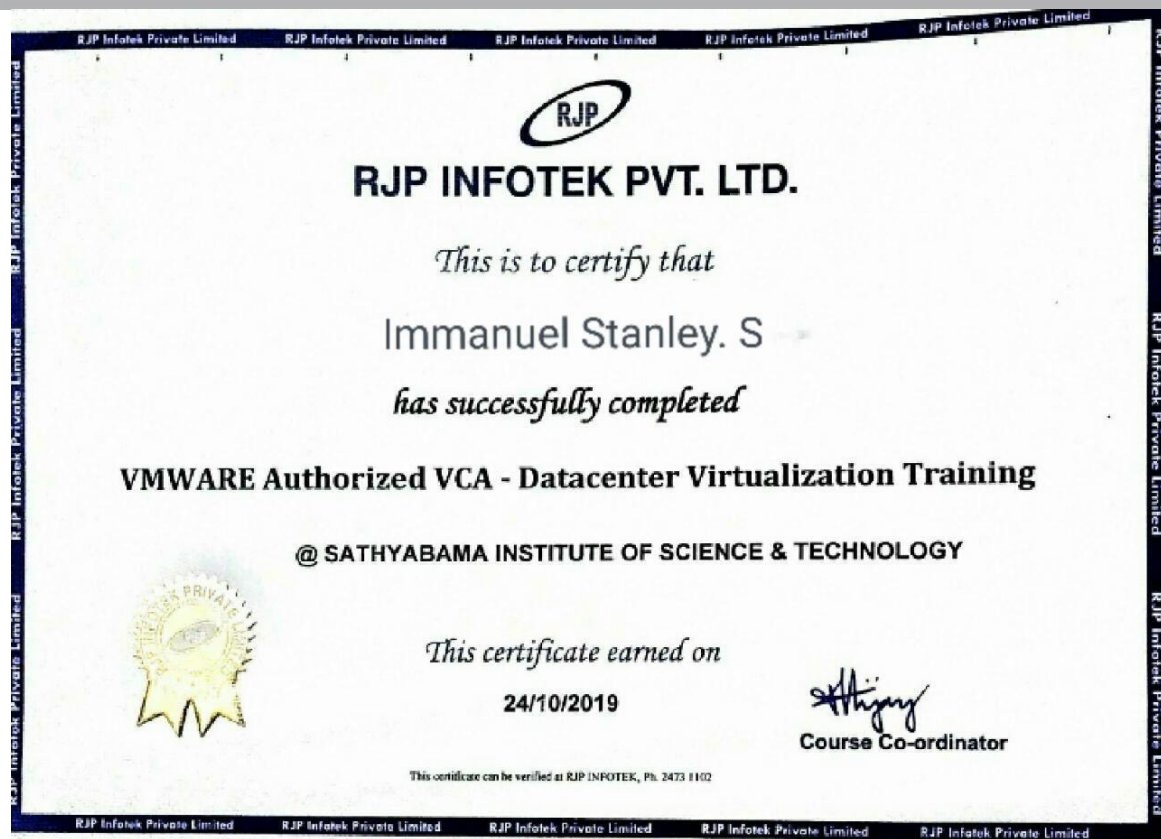
ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.Sasikala M.E.,Ph.d, Dean**, School of Computing, **Dr. Subashini M.E.,Ph.D.**, Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S.Gowri M.E.,Ph.D.**, for this valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.



ABSTRACT

INTRANET VIDEO CONFERENCING (One-many)

This project aims to provide video conference call to all the systems connected in the same network by providing their individual ip addresses. This project uses socket programming to connect between server and client and provides high reliable and quality video call service through the use of in-built python modules. This package uses modules such as OpenCV, NumPy, Threading, Socket, IO modules. Videoconferencing boosts productivity, saves time, reduces travel expenses, and overall promotes collaboration. The advantage of video conferencing is the ability to facilitate all of those benefits without requiring constant travel for face-to-face communication.

SERVER:

It receives the client images and process it into a video and continues this process until one of them terminates.

CLIENT:

It sends the client images and receives the server images and process them into a video and wait for termination.

TABLE OF CONTENTS

CH NO	TITLE NAME	PAGE NO
	ABSTRACT	
	LIST OF FIGURES	
	LIST OF ABBREVIATIONS	
1	INTRODUCTION	
	1.1 INTRODUCTION ON INTRANET VIDEO CONFERENCING	
2	LITERATURE SURVEY	
	2.1 PROPOSED SYSTEM	
	2.2 PROCESSING METHODS	
	2.2.1 OPEN CV	
	2.2.2 PYTHON NUMPY	
	2.2.3 THREADING	
3	SYSTEM REQUIREMENTS	
	3.1 SOFTWARE AND HARDWARE REQUIREMENTS	
	3.2 MATPLOTLIB	
	3.3 PANDAS	
	3.4 SKLEARN	
4	SCREENSHOTS	
5	CONCLUSION	
	REFERENCE	
	APPENDIX	
	A. SOURCE CODE	

LIST OF FIGURES

FIG.NO	FIGURE NAME	PAGE NO
2.1	MESSAGE PASSING	

LIST OF ABBREVIATIONS

LR	Logistic Regression
----	---------------------

1.INTRODUCTION

Video conferencing is a technology that allows users in different locations to hold face-to-face meetings without having to move to a single location together. Video conferencing is a technology-enabled type of meeting where two or more people, in different geographic locations, conduct live visual conferences through the internet for the purpose of communicating and collaborating. To provide video conference call to all the systems connected in the same network by providing their individual ip addresses. This project uses socket programming to connect between server and client and provides high reliable and quality video call service through the use of in-built python modules. Intranet Video Conferencing Video conferencing in a private network i.e. Intranet video conferencing is a special requirement of organizations. TrueConf Server is a secure solution that allows you to conduct video conferences in a corporate (private) network without an Internet connection to enable intranet video conferencing within it.

We'll then implement two Python scripts:

- A *client* that will capture frames from a simple webcam
- And a *server* that will take the input frames and run object detection on them

Will be using Raspberry Pis as our clients to demonstrate how cheaper hardware can be used to build a distributed network of cameras capable of piping frames to a more powerful machine for additional processing. In this package uses modules such as OpenCV, NumPy, Threading, Socket, IO modules. Videoconferencing boosts productivity, saves time, reduces travel expenses, and overall promotes collaboration. The advantage of video conferencing is the ability to facilitate all of those benefits without requiring constant travel for face-to-face communication.

2. LITERATURE SURVEY

2.1 PROPOSED SYSTEM:

In Intranet Video Conferencing there are a number of reasons why you may want to stream frames from a video stream over a network with OpenCV. To start, you could be building a security application that requires all frames to be sent to a central hub for additional processing and logging. Or, your client machine may be highly resource constrained (such as a Raspberry Pi) and lack the necessary computational horsepower required to run computationally expensive algorithms (such as deep neural networks, for example). In these cases, you need a method to take input frames captured from a webcam with OpenCV and then pipe them over the network to another system. There are a variety of methods to accomplish this task (discussed in the introduction of the post), but today we are going to specifically focus on **message passing**.

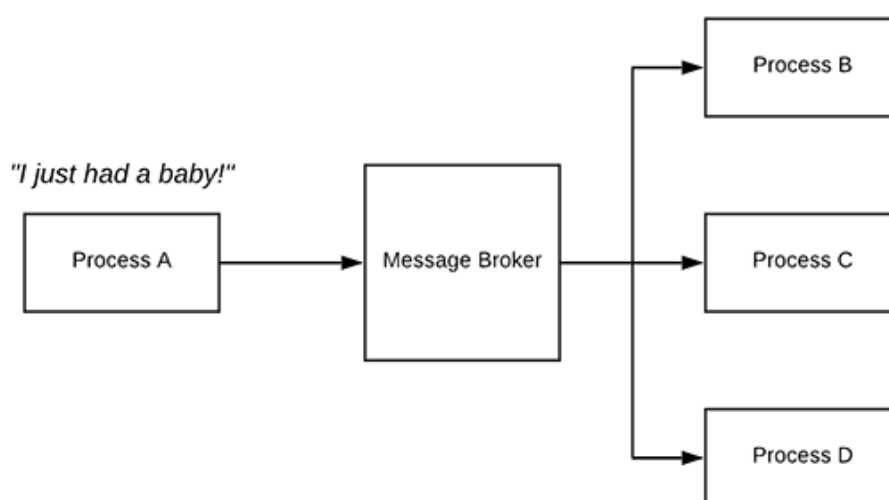


FIG 2.1 MESSAGE PASSING

Message passing is a programming paradigm/concept typically used in multiprocessing, distributed, and/or concurrent applications. Using message passing, one process can communicate with one or more other processes, typically using a *message broker*. Whenever a process wants to communicate with another process, including *all* other processes, it must first send its request to the message broker. The

message broker receives the request and then handles sending the message to the other process(es).

2.2 PROCESSING METHODS

2.2.1 OPEN CV:

OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge. Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project. OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day. OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development. OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language.

OpenCV-Python:

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. Python is a general purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability. Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it is easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

2.2.2 PYTHON NUMPY:

Python Numpy:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

Arrays in Numpy:

Array in Numpy is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In Numpy, number of dimensions of the array is called rank of the array. A tuple of integers giving the size of the array along each dimension is known as shape of the array. An array class in Numpy is called as ndarray. Elements in Numpy arrays are accessed by using square brackets and can be initialized by using nested Python Lists. In a numpy array, indexing or accessing the array index can be done in multiple ways. To print a range of an array, slicing is done. Slicing of an array is defining a range in a new array which is used to print a range of elements from the original array. Since, sliced array holds a range of elements of the original array, modifying content with the help of sliced array modifies the original array content.

2.2.3 THREADING:

A thread is a separate flow of execution. This means that your program will have two things happening at once. But for most Python 3 implementations the different threads do not actually execute at the same time: they merely appear to. It's tempting to think of threading as having two (or more) different processors running on your program, each one doing an independent task at the same time. That's almost right.

The threads may be running on different processors, but they will only be running one at a time. Getting multiple tasks running simultaneously requires a non-standard implementation of Python, writing some of your code in a different language, or using multiprocessing which comes with some extra overhead.

Because of the way CPython implementation of Python works, threading may not speed up all tasks. This is due to interactions with the GIL that essentially limit one Python thread to run at a time. Tasks that spend much of their time waiting for external events are generally good candidates for threading. Problems that require heavy CPU computation and spend little time waiting for external events might not run faster at all. This is true for code written in Python and running on the standard CPython implementation. If your threads are written in C they have the ability to release the GIL and run concurrently. If you are running on a different Python implementation, check with the documentation too see how it handles threads. If you are running a standard Python implementation, writing in only Python, and have a CPU-bound problem, you should check out the multiprocessing module instead. Architecting your program to use threading can also provide gains in design clarity. Most of the examples you'll learn about in this tutorial are not necessarily going to run faster because they use threads. Using threading in them helps to make the design cleaner and easier to reason about.

3.SYSTEM REQUIREMENTS

3.1 SOFTWARE AND HARDWARE REQUIREMENTS

Software Requirements

- Operating System: Windows10 and other versions.
- Language: Python 3.8.5

Hardware requirements

- Ram: 8 GB and more
- Processor: Any Intel Processor
- Hard Disk: 6 GB and more
- Speed: 1 GHZ and more

3.2 MATPLOTLIB

Humans are very visual creatures: we understand things better when we see things visualized. However, the step to presenting analyses, results or insights can be a bottleneck: you might not even know where to start or you might have already a right format in mind, but then questions like “Is this the right way to visualize the insights that I want to bring to my audience?” will have definitely come across your mind.

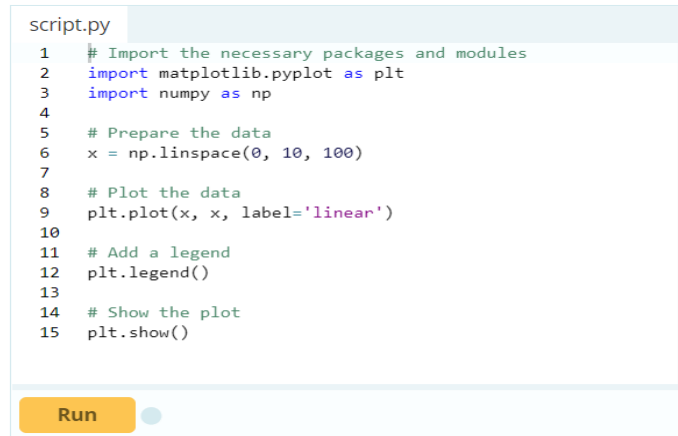
When you’re working with the Python plotting library Matplotlib, the first step to answering the above questions is by building up knowledge on topics like:

- The **Anatomy of Matplotlib plot**: what is a subplot? What are the Axes? What exactly is a figure?
- **Plot Creation**: which could raise questions about what module you exactly need to import (pylab or pyplot?), how you exactly should go about initializing the figure and the Axes of your plot, how to use matplotlib in Jupyter notebooks, etc.
- **Plotting Routines**: from simple ways to plot your data to more advanced ways of visualizing your data.
- **Basic Plot customization**, with a focus on plot legends and text, titles, axes labels and plot layout.
- **Saving, Showing and Clearing your plots**: show the plot, save one or more figures to, for example, pdf files, clear the axes, clear the figure or close the plot, etc.

(To practice matplotlib interactively, try the free Matplotlib chapter at the start of this Intermediate Python course or see DataCamp’s Viewing 3D Volumetric Data With Matplotlib tutorial to learn how to work with matplotlib’s event handler API.)

What does the Matplotlib looks like?

At first sight, it will seem that there are quite some components to consider when you start plotting with this Python data visualization library. You'll probably agree with me that it's confusing and sometimes even discouraging seeing the amount of code that is necessary for some plots, not knowing where to start yourself and which components you should use. Luckily, this library is very flexible and has a lot of handy, built-in defaults that will help you out tremendously. As such, you don't need much to get started: you need to make the necessary imports, prepare some data, and you can start plotting with the help of the `plot()` function! When you're ready, don't forget to show your plot using the `show()` function.



```
script.py
1  # Import the necessary packages and modules
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  # Prepare the data
6  x = np.linspace(0, 10, 100)
7
8  # Plot the data
9  plt.plot(x, x, label='linear')
10
11 # Add a legend
12 plt.legend()
13
14 # Show the plot
15 plt.show()
```

Figure 3.1. matplotlib plot

3.3 PANDAS

pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python. Additionally, it has the broader goal of becoming **the most powerful and flexible open source data analysis / manipulation tool available in any language**. It is already well on its way toward this goal. Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

3.4 SKLEARN

Simple and efficient tools for predictive data analysis. Accessible to everybody, and reusable in various contexts. Built on NumPy, SciPy, and matplotlib. Open source,

commercially usable - BSD license. Used in: Classification, Regression, Clustering, Dimensionality reduction, Model selection, pre-processing

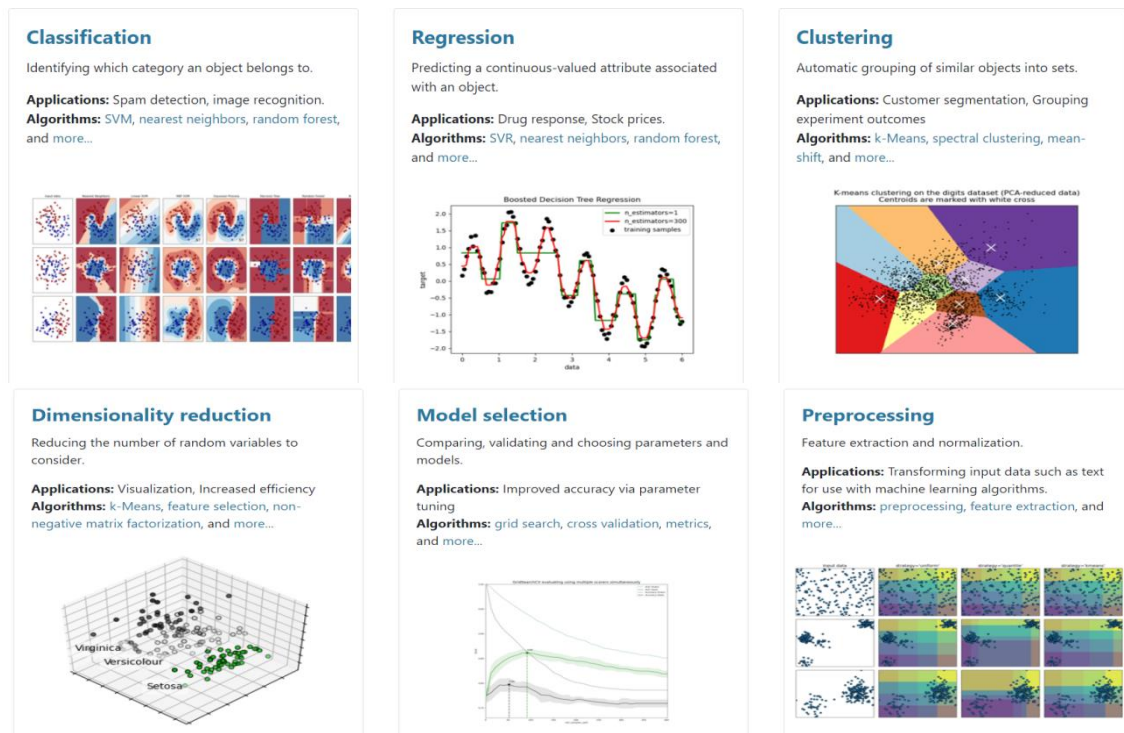
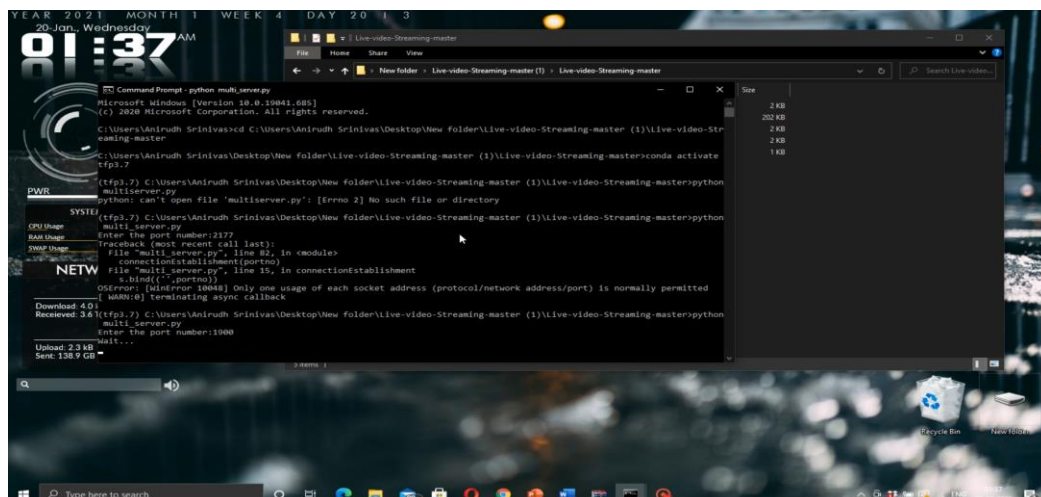


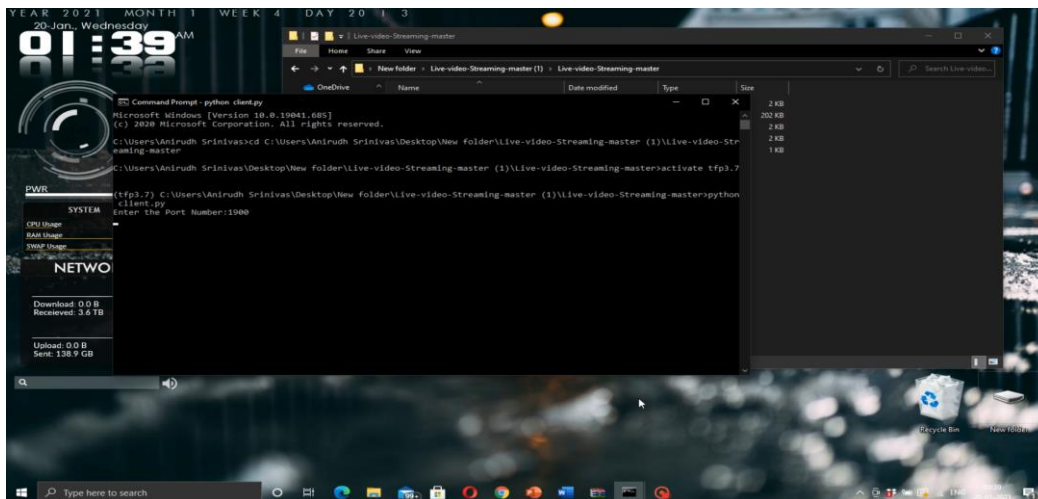
Figure 3.2. sklearn

4 . IMPLEMENTATION

RESULT AND DISCUSSION:

SCREENSHOTS:





5.CONCULSION

Video conferencing could lead the way for a dual approach, giving students more responsibility for their learning, working in groups, and doing educational tasks; all of which would benefit conventional teaching, but video conferencing provides an opportunity to implement them. It does not replace the use of print or other methods used in the conceptualization process. It can be used to encourage construction and its true use lies in encouraging dialogue and increasing the scope for dialogue. With the advancement and ease of availability of high speed and cheap internet connections, it is expected that video conferencing will increasingly become popular thus, leading to more interest and use of distance learning. its application in distance learning was presented as an effective way of delivering subject matter in classrooms.

REFERENCE

- [1]. Sami Andberg (2008). Post Graduate Thesis: Video Conferencing in Distance Learning. Department of Computer Science, University of Helsinki.
- [2]. Rachel Roberts (2009). Video Conferencing in Distance Learning: A New Zealand School's Perspective. Journal of Distance Learning ©Distance Education Association of New Zealand, Vol. 13, pp. 91 – 107.
- [3]. Dr. Lynne (2007). Video Conferencing in Higher Education”, Institute of Computer Based Learning, Heriot Watt University Edinburgh.
- [4]. JNT Association (2007). Introduction to Video Conferencing. <http://www.ja.net/vtas>
© The JNT Association,

- [5]. Alan D. Greenberg (2009). Mapping the Latest Research into Video-Based Distance Education. Wainhouse Research LLC, USA,
- [6]. Graeme Byrne and Lorraine Staehr (2002). International Internet Based Video Conferencing in Distance Education: A Low-Cost Option. InSITE – Where Parallels Intersect, pp. 187 – 194.
- [7]. Chris McCuller (2010). Videoconferencing and Distance Learning. Valdosta State University Whitepaper.
- [8]. Polycom inc. (2010). The Top Five Benefits of Video Conferencing. Polycom, inc., Polycom Worldwide Headquarters,
- [9]. Donald Gillies (2008). Student Perspectives on Videoconferencing in Teacher Education at a Distance. Distance Education, Vol. 29, No. 1, pp. 107 – 118,
- [10]. Alan Greenberg (2004). Navigating the Sea of Research on Videoconferencing-Based Distance Education. Wainhouse Research, LLC,

APPENDIX

A. SOURCE CODE

CLIENT SIDE:

```
import sys
sys.path.append('/usr/local/lib/python2.7/site-packages')

import cv2
import io
import numpy
```

```

from PIL import Image
import socket

cap=cv2.VideoCapture(0)

def receive(sock):

    totrec = 0
    msgArray = bytearray()
    c = sock.recv(8)
    c=c.decode()
    length=int(c)
    while totrec<length :
        chunk = sock.recv(length-totrec)
        msgArray.extend(chunk)
        totrec += len(chunk)
    show_frd(msgArray)

def send2(framestring,sock):

    length =len(framestring)
    lengthstr=str(length).zfill(8)
    sent = sock.send((lengthstr).encode())
    sock.sendall(framestring)

def show_me(im_b):

    p=io.BytesIO(im_b)
    pi=Image.open(p)
    img = cv2.cvtColor(numpy.array(pi), cv2.COLOR_RGB2BGR)
    cv2.imshow('Me',img)
    cv2.waitKey(1)
    if cv2.waitKey(1) & 0xFF == ord('q'):

```

```

        cap.release()
        cv2.destroyAllWindows()

def show_frd(im_b):

    p=io.BytesIO(im_b)
    pi=Image.open(p)
    img = cv2.cvtColor(numpy.array(pi),cv2.COLOR_RGB2BGR)
    cv2.imshow('Friend',img)
    cv2.waitKey(1)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()

def send1(client):

    ret,img = cap.read()
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    pimg = Image.fromarray(img)
    b = io.BytesIO()
    pimg.save(b,'jpeg')
    image_b = b.getvalue()
    show_me(image_b)
    send2(image_b,client)

def connectionEstablishment(ip,portno):

    s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    s.connect((ip,portno))
    print("Connection Established with :",ip,"at port number:",portno)
    print("Wait...")
    return s

portno = int(input("Enter the Port Number:"))

```

```
ip = '10.1.82.126'  
s = connectionEstablishment(ip,portno)
```

```
while True:
```

```
    send1(s)  
    receive(s)
```

SERVER SIDE:

```
import sys  
sys.path.append('/usr/local/lib/python2.7/site-packages')  
import cv2  
import io  
import numpy  
from PIL import Image  
import socket  
import threading  
cap=cv2.VideoCapture(0)
```

```
def connectionEstablishment(portno):
```

```
    s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)  
    s.bind(('',portno))  
    s.listen(5)  
    print("Wait...")  
    while True:  
        (client,(ip,port))=s.accept()  
        print("Connection Established with :",ip,"at port number:",port)  
        threading.Thread(target = sendAndReceive,args = (client,ip)).start()
```

```
def sendAndReceive(client,ip):
```

```

while True:
    receive(client,ip)
    send1(client,ip)

def receive(sock,ip):

    totrec = 0
    msgArray = bytearray()
    c = sock.recv(8)
    c = c.decode()
    length = int(c)
    while totrec < length:
        chunk = sock.recv(length - totrec)
        msgArray.extend(chunk)
        totrec+=len(chunk)
    show_frd(msgArray,ip)

def send2(framestring,sock):

    length =len(framestring)
    lengthstr = str(length).zfill(8)
    sent = sock.send((lengthstr).encode())
    sock.sendall(framestring)

def show_me(im_b):

    p = io.BytesIO(im_b)
    pi = Image.open(p)
    img = cv2.cvtColor(numpy.array(pi), cv2.COLOR_RGB2BGR)
    cv2.imshow('Me',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()

```

```
def show_frd(image_b,ip):

    p = io.BytesIO(image_b)
    pi = Image.open(p)
    img = cv2.cvtColor(numpy.array(pi), cv2.COLOR_RGB2BGR)
    cv2.imshow(ip,img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
```

```
def send1(client,ip):

    ret,img = cap.read()
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    pimg = Image.fromarray(img)
    b = io.BytesIO()
    pimg.save(b,'jpeg')
    image_b = b.getvalue()
    show_me(image_b)
    send2(image_b,client)
```

```
portno=int(input("Enter the port number:"))
connectionEstablishment(portno)
```