Chapter 2

Server establishment, IoT and NLP implementation

The motive of this project is to develop a full fledged server with support to latest technologies developing in the market. That is -

1. Provide IoT support
2. Provide cloud support
3. Provide machine leaning and artificial intelligence support

It combines technologies of various fields into a one unique machine. Use of various programming languages is take like -

1. Python
2. C ++
3. C
4. Java
5. Bash scripts/Terminal scripting
6. MySQL query language
7. HTML

IoT helps in interfacing the virtual world with the physical world, while generating enormous amount of data which can be stored inside the cloud server and proper and easy LAN and WAN access of that data is made possible only with the help of cloud server. While the generated data need to be processed to fetch some meaning and value to the business which is supported with the help of technology called machine leaning and artificial intelligence.

Python is used to create bash scripts and implement the technology of computer vision in this project while C and C++ is used to program the micro controllers.

HTML, JAVA, MySQL and others are used to develop the web interface to provide user support and make the server capable of data protection, access and user credential protection.

The motive of this project is to make a wast scope idea that have its implementation in day to day life by providing and easy mind controlled appliance that allow user to control the power of appliance just by thinking by using EEG, computer vision and IoT.

Second motive of this project is to allow speech controlled environment for have, provide security and real time surveillance to each and every home and provisioning a secure way of data access i.e a self hosted cloud in a cheap yet affective way.

Chapter 2.1 Server Side Utilities

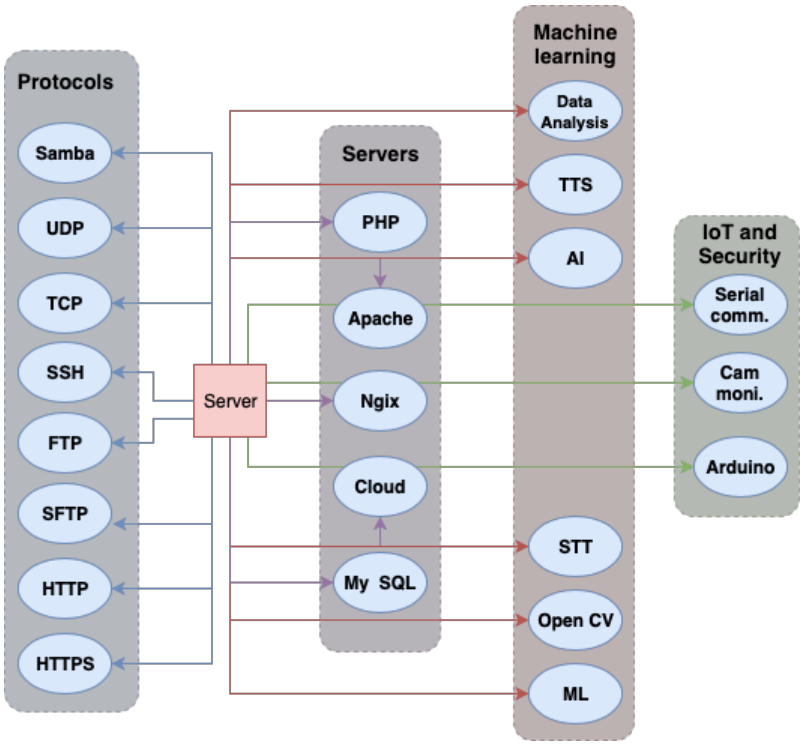


Figure 1.1 Utilities and facilities running on server

Server side is consist of 4 different parts i.e

1. Protocols
2. Servers technologies
3. Machine Learning and AI based technologies
4. IoT and Security Technologies

Protocol – These are the set of rules that are used to implement various technologies for the management and controlling the working of various services inside the server side. Like MIME for email service activation as soon as the server starts, a cloud server scripts begin to run that takes screen short of the screen and send the created screen capture to defined email using MIME protocol. SSH is used to login remotely and etc.

Server Technologies – This contains various different technologies from different vendors which when combined provides services of website hosting, database management, controlling and file access.

Machine Learning and AI based technologies – These contains technologies related to artificial intelligence and machine learning. From speech to text conversion, to text to speech conversion, detection of objects, human faces, posture and movement is managed by this field only, it also provides data analysis and visualisation for the generated data.

IoT and Security Technologies – These contains using the latest technologies like IoT i.e internet of things. It helps in automated heat management of the system, remotely controlling devices and appliances, to authentication using RFIDs, It uses microcontrollers like raspberry pi and arduino with several sensors connected to it.

As shown in figure 1.1, left side shows 8 protocols running:

1. Samba
2. UDP
3. TCP
4. SSH
5. FTP
6. SFTP
7. HTTP
8. HTTPS

Samba – It is used to create a NAS storage at the LAN level to support a simple file accessing system using Samba protocol to the mounted drives attached to the storage.

UDP – It is a fast data transfer protocol, that sends data without expecting a acknowledgment hence making the process fast, and is used mostly for vide streaming where few lost package does not destroy the integrity of the data.

TCP – On the other hand TCP waits for the acknowledgement after the transfer of package from one user to another and is used to transfer data like documents, and data whose integrity is crucial. It is slow but no data is lost as the lost data is sent again if no acknowledgment is recieved.

SSH – Secure shell is used to remotely login into the system and access the system using the shell i.e terminal in Linux and OSX and powershell in windows. It uses port 22.

FTP – File transfer protocol as the name suggest is used to transfer file between two remotely located but interconnected over network systems. Samba is freely available, unlike other SMB/ CIFS implementations, and allows for interoperability between Linux/Unix servers and Windows-based clients. File Transfer Protocol (FTP) is a standard network protocol used to exchange and manipulate files over a TCP/IP based network, such as the Internet.

SFTP – Shell file transfer protocol, uses SSH protocol for the transfer of file between two remote systems.

HTTP – Is an old protocol used to display websites, locally port 80 is open and is allowed by the firewall to make and accept request on the internet.

HTTPS – Is a secure HTTP protocol, Instead of HyperText Transfer Protocol (HTTP), the website uses HyperText Transfer Protocol Secure (HTTPS). Using HTTPS, the computers agree on a "code" between them, and then they scramble the messages using that "code" so that no one in between can read them. This keeps your information safe from hackers. They use the "code" on a Secure Sockets Layer (SSL), sometimes called Transport Layer Security (TLS) to send the information back and forth.

MIME – MIME (Multi-Purpose Internet Mail Extensions) is an extension of the original Internet e-mailprotocol that lets people use the protocol to exchange different kinds of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol, the Simple Mail Transport Protocol (SMTP). In 1991, Nathan Borenstein of Bellcore proposed to the IETF that SMTP be extended so that Internet (but mainly Web) clients and servers could recognize and handle other kinds of data than ASCII text. As a result, new file types were added to "mail" as a supported Internet Protocol file type.

Figure 1.1, shows 5 servers running:

1. PHP
2. Apache
3. Ngix
4. Cloud
5. My SQL

PHP – PHP is a general-purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites or elsewhere.

Apache – Apache is the most widely used web server software. Developed and maintained by Apache Software Foundation, Apache is an open source software available for free. It runs on 67% of all webservers in the world. It is fast, reliable, and secure.

Nginx – The NGINX web server can act as a very capable software load balancer, in addition to its more traditional roles serving static content over HTTP and dynamic content using FastCGI handlers for scripts. Because NGINX uses a non-threaded, event-driven architecture, it is able to outperform web servers like Apache.

Cloud – The cloud server is based on Google Driver type technology which is called nextCloud a fork that developed from ownCloud is used for easy drag and drop cloud based technologies which support various features like, social networking, video calling, device tracking, office document editing and management, note taking, etc. The cloud server uses PHP, My SQL and Apache server and is hosted on port 80 i.e HTTP and HTTPS is used.

My SQL – It is based on a client-server model. The core of MySQL is MySQL server, which handles all of the database instructions (or commands). MySQL servers available as a separate program for use in a client-server networked environment and as a library that can be embedded (or linked) into separate applications.

Figure 1.1, shows 6 ML and AI facilities running:

1) Data Analysis
2) TTS
3) STT
4) AI
5) Open CV
6) ML

Data Analysis – Data analysis can be performed with the help of pre designed scripts developed in python programming language and combined with the bash shell scripts just requires the input of data as the argument along with the input hence making a robust and easy to maintain and modify with changes in future.

TTS – Text to speech is also performed as the user speaks that spoken language is analysed with the help of speech to text technology and that text is transferred over network to the local server for processing in the voice command script. Full description of the functioning will be provided in further document.

STT – Speech to text, this technology is being used at both client and server. When client provides a predefined command to the machine by running the hello.py script in python language that first asks for the host or server IP address that is running the analysis for the converted speech to text and then the commanded task is performed. Full description of the functioning will be provided in further document.

AI – Artificial in itself is a very wide and vast topic and high level implementation requires better processing power. But due to the limited processing power and working in the ability of the knowledge AI here is used for training the Open CV models hence help in creation of harcascades for face detection and object detection that takes input in form of 10000 test cases consisting of several images.

Open CV – It is a well known image processing library that supports from small objects a like microbes to huge objects like starts and planets. Here the used Open CV library are of volume 3 and volume 4. It is used for object and face detection based on the machine and server trained xml(s) of harrcascades.

ML – Machine leaning technology is used in machine in form of data visualisation for the creation of graphs and predication modles.

Figure 1.1, shows 3 IoT and Security facilities running:

1. Serial Communication
2. Monitoring over webcam
3. Arduino *dev*/ttyACM0

Serial Communication – The server hosting hardware is connected to the small microcontrollers like arduino which are connected to the sensors and the server is directly controlling those microcontrollers using serial ports of USB 2.0. All the data received is stored into data files and serial in opened to send and receive data for commanding and performing tasks in physical world. Serial communication here is a true interface between the software and the physical world.

Monitoring over webcam – The server provides security in form of real time surveillance and monitoring using the usb 2.0 webcam attached to one of its serial port. The webcam surveillance uses the port 8080 of the local host which when exposed to public IP helps in proper monitoring of the surroundings inside the local network or over WAN.

Arduino – *dev*/ttyACM0 is the port to which arduino number one is connected which transfers data related to heat of the system, controlling fans and lights and appliance connected to it using and relay, humidity details and PM 2.5 details.

Chapter 2.2 Server Side open ports

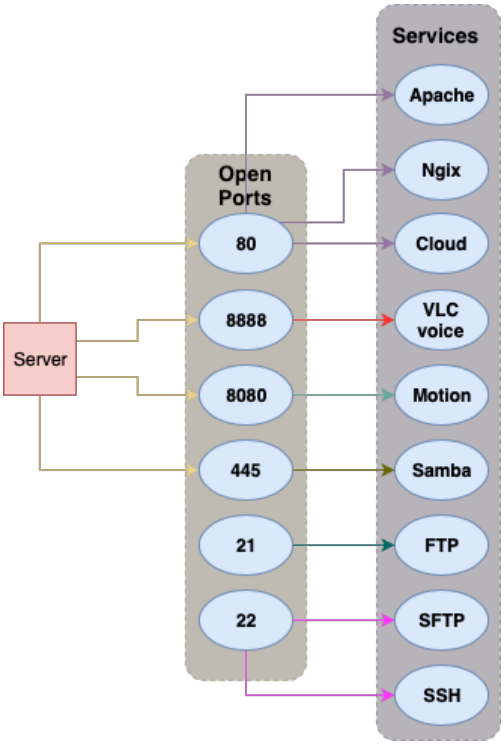


Figure 1.2 open port available on server side

Figure 1.2. shows, server side is consist of 4 different parts i.e

1. 80
2. 8888
3. 8080
4. 445
5. 21
6. 22

Port 80 – It is used mainly for the internet, and all the request from and to internet goes through port 80. most of the firewall allow the request from port 80 and is open by default. It uses protocols like http and https. Here in the project port 80 is used for hosting servers like Apache and Nginx. While the cloud storage hosted on Apache can also be easily accessed over this port 80.

Port 8888 – This port is opened when a script is run accessed using ssh, as soon as the scripts run it open a port on the remote server playing an MP3 file as out.mp3. Which is basically outputting the sound which is being heard by the speakers connected to the servers hardware allowing to hear what is all playing onto and around the server.

Port 8080 – It opens a motion service. When motion service is started using 'sudo service motion start' command onto the server shell, it activates the webcam which can be seen on the remote servers port 8080. Can be accessed using a web browsers as 'Servers.ip.address.inbrowser:8080'.

Port 445 – Samba service for file management is hosted onto the port 445, which gives the server a unique open port that allow the bind drives to the server to be accessed remotely inside the LAN and can be used to access over WAN, if the local host have a public domain attached to it. t is used to create a NAS storage at the LAN level to support a simple file accessing system using Samba protocol to the mounted drives attached to the storage.

Port 21 – Port 21 is famously known for service of FTP. File transfer protocol as the name suggest is used to transfer file between two remotely located but interconnected over network systems. Samba is freely available, unlike other SMB/CIFS implementations, and allows for interoperability between Linux/Unix servers and Windows-based clients. File Transfer Protocol (FTP) is a standard network protocol used to exchange and manipulate files over a TCP/IP based network, such as the Internet.

Port 22 – SSH service uses this port 22 to allow remote login to the server from anywhere and from any client machine inside the LAN or over WAN if the server ip is forwarded to some public domain or Public IP. Secure shell is used to remotely login into the system and access the system using the shell i.e terminal in Linux and OSX and powershell in windows. hell file transfer protocol, uses SSH protocol for the transfer of file between two remote systems.

Chapter 2.3 Voice control

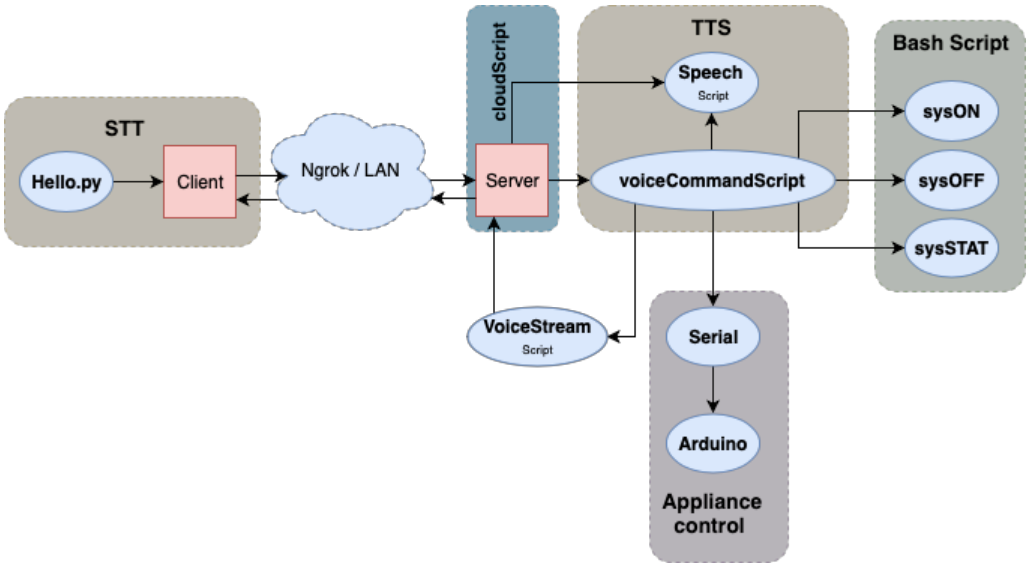


Figure 1.3 Voice control mechanism

Figure 1.3 show the complete flow of the voice control mechanism followed between client and server.

The complete architecture for voice control is divided into 7 components, all the components have sub parts inside it. Those parts are -

1. STT
2. Interconnection (Nrgrok/LAN)
3. CloudScript
4. TTS
5. Bash Scripts
6. Application Control
7. VoiceStream script

STT – Client running the provided python script called hello.py inside the python 3 environment, while being connected to the internet the client can communicate and command the remote server with his speech using the speech to text conversion.

Interconnection (Nrgrok/LAN) – This explains how the internconnection or the communication will establish between the client and the server. It uses two ways of establishing the connection i.e -

1. Locally – LAN – Inside the same network of the router having common gateway and starting class C IP address, which support 254 hosts.
2. Globally – Ngrok – It is a free plan service which helps in forwarding a local machine to a port hosted by the ngrok servers, hence allowing easy access to local machine globally from anywhere without having the need to forward port on the router device.

The ngrok uses commands like:

1. ngrok tcp 22
2. ngrok http 80
3. ngrok start –all

The file for configuration is located at /home/pi/.ngrok2/ngrok.yml and is edited using sudo nano and is as shown in figure 1.4

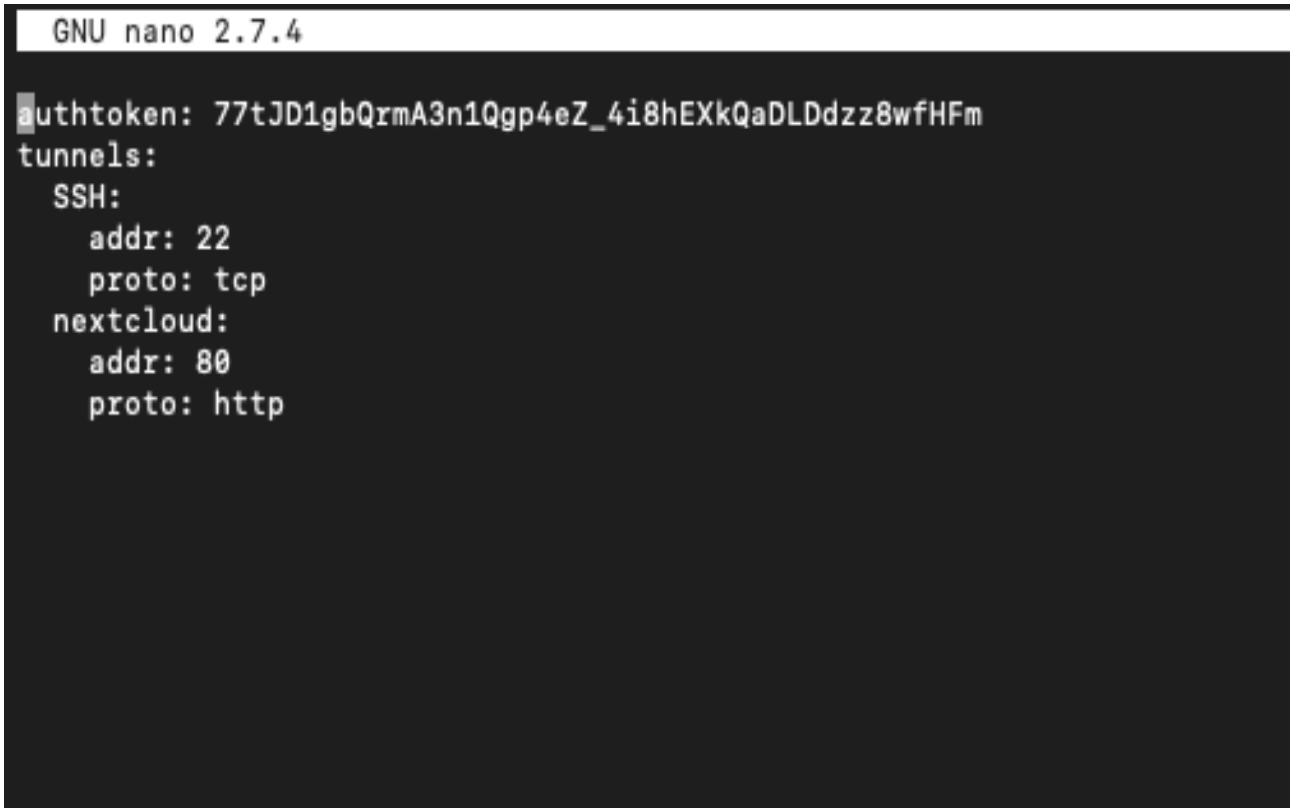


Figure 1.4 ngrok configuration file

CloudScript – This script is saved in bin location and changed mode using 'chmod 755 filename' to executable form so this bash script can easily be executed using a single command called cloudScripts. This command called 'cloudScript' is then saved into the 'bashrc' file, using command 'nano ~/.bashrc'. So this makes the cloudScript to run by default as soon as the terminal opens. Now the terminal is made to open by default as soon as the server starts and login into the operating system.

The cloudScript looks like as shown in figure 1.5:

1. First the script is converted to a bash file so the terminal can know that it is an executable type of file. This is done by using #!/bin/bash
2. a file variable is created with the location of a file in it called cloudScript.temp which is a temporary file that prevents the re-execution of the cloudScript so only one instance creates for a session.
3. Later that file is tested using if-then else condition for the existence of the file.
4. If file do not exist then a file is existed and a line is echoed saying "[WARNING] ngrok Instance created, Please do not delete it.
5. And the programs wait for creation by sleeping for 2 seconds.

6. Then method of parallel exicution of command is used using & p(n) = $(n), where (n) is an integer number matching the number of command and later the execution is killed using **["$?" -gt 1] || exit "$p(n)" .**
7. Screen short is taken using ‘import -window root filenage.png’ and saved in a location.
8. That screen short is then mailed using the script created in python to the specified email address telling the ngrok instance address so that can be accessed using web browser or the terminal to ssh in the remote machine.
9. The mailing script developed in python is shown in figure 1.6(a) and 1.6(b).
10. Another script called RebootScript is created which whose only motive is to remove the png screen capture and the temp file of cloudScript so a new ngrok instance can be created.
11. This file is then made to execute as soon as the os boots and is achieved using ‘sudo update-rc.d NameOfTheScript defaults’ command in the terminal.
12. Saving the file in init.d allow the script to be controlled using a single turn ON and OFF commands:

    1. sudo /etc/init.d/NameOfTheScript start
    2. sudo /etc/init.d/NameOfTheScript start

13. This helps in making the robust and smart and a complete black box which allow a simple power ON and OFF mechanism and the user do not have to worry about what is going inside the server as server manages it self.

```
 GNU nano 2.7.4                                                File: /bin/cloudSc

#! /bin/bash

file="/home/pi/cloudScript.temp"
if [ ! -f "$file" ]
then
        echo '[WARNING] ngrok Instance created, Please do not deleted it!' >/home/pi/cloudScript.temp
        sleep 2s
        ngrok start -all & p1=$!
        sleep 8s
        echo "Now taking Snapshot"
        sleep 2s
        import -window root Network.png & p2=$!
        echo "Now Sending mail....."
        sleep 6s
        mailScriptPython
        wait -n
        [ "$?" -gt 1 ] || exit "$p2"
        wait
else
        echo "File" $file "exist"
fi
```

Figure 1.5 cloudScript

```
GNU nano 2.7.4                    File: mailScriptPython

#! /usr/bin/python

import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders

fromaddr = 'raspberrypiserverinfo@gmail.com'
toaddr = 'rashbits@gmail.com'
password='rashbir england'

s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()
s.ehlo()

s.login(fromaddr,password)

# instance of MIMEMultipart
msg = MIMEMultipart()

# storing the senders email address
msg['From'] = fromaddr

# storing the receivers email address
msg['To'] = toaddr

# storing the subject
msg['Subject'] = 'Raspberry Pi Started'

# string to store the body of the mail
body = 'SSH port details in TCP for port number 22, Cloud Server details in HTTP for port number 80'

# attach the body with the msg instance
msg.attach(MIMEText(body, 'plain'))

# open the file to be sent
filename = 'Network.png'
attachment = open('/home/pi/Network.png', 'rb')

# instance of MIMEBase and named as p
p = MIMEBase('application', 'octet-stream')

# To change the payload into encoded form
p.set_payload((attachment).read())

# encode into base64
encoders.encode_base64(p)

p.add_header('Content-Disposition', "attachment; filename= %s" % filename)

# attach the instance 'p' to instance 'msg'
msg.attach(p)

# Converts the Multipart msg into a string
text = msg.as_string()

# sending the mail

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  ^Y Prev Page  M-\ First Line  M-W WhereIs Next  ^^ Mark Text  M-} Indent Text  M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Linter ^_ Go To Line ^V Next Page  M-/ Last Line  M-] To Bracket  M-A Copy Text  M-{ Unindent Text M-E Redo
```

Figure 1.6(a) mailScriptPython

```
GNU nano 2.7.4                    File: mailScriptPython

# string to store the body of the mail
body = 'SSH port details in TCP for port number 22, Cloud Server details in HTTP for port number 80'

# attach the body with the msg instance
msg.attach(MIMEText(body, 'plain'))

# open the file to be sent
filename = 'Network.png'
attachment = open('/home/pi/Network.png', 'rb')

# instance of MIMEBase and named as p
p = MIMEBase('application', 'octet-stream')

# To change the payload into encoded form
p.set_payload((attachment).read())

# encode into base64
encoders.encode_base64(p)

p.add_header('Content-Disposition', "attachment; filename= %s" % filename)

# attach the instance 'p' to instance 'msg'
msg.attach(p)

# Converts the Multipart msg into a string
text = msg.as_string()

# sending the mail
s.sendmail(fromaddr, toaddr, text)

# terminating the session
s.quit()

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  ^Y Prev Page  M-\ First Line  M-W WhereIs Next  ^^ Mark Text  M-} Indent Text  M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Linter ^_ Go To Line ^V Next Page  M-/ Last Line  M-] To Bracket  M-A Copy Text  M-{ Unindent Text M-E Redo
```

Figure 1.6(b) mailScriptPython

TTS – The text to speech conversition is done and commands are executed as programmed. The script for the command can be sheen in :

1. Figure 1.7(a) i.e voiceCommandScript
2. Figure 1.7(b) i.e Speak script
3. 1.7(c), 1.7(d) and 1.7(e) i.e simple_google_tts script

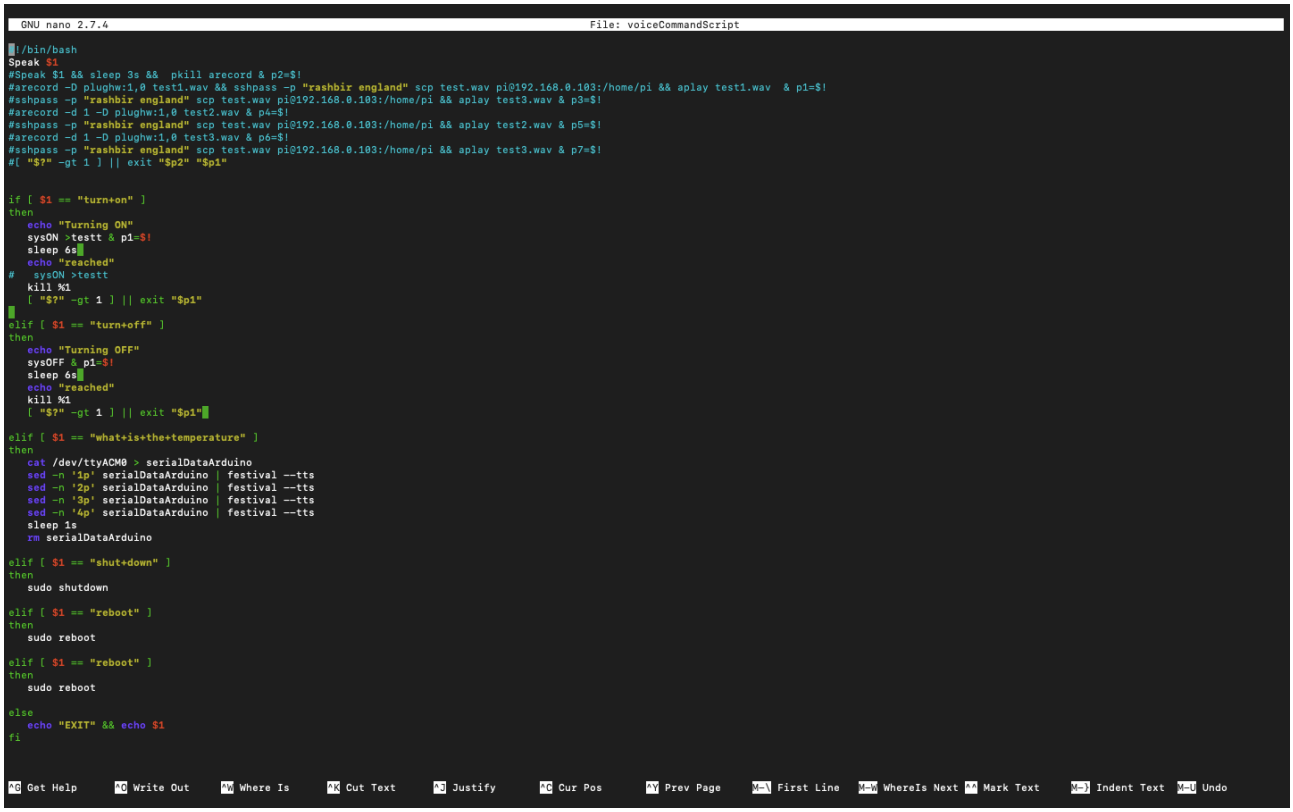


Figure 1.7(a) voiceCommandScript

Figure 1.7(a) Speak

```
 GNU nano 2.7.4                                File: simple_google_tts

#!/bin/bash

# NAME:         Simple Google TTS
# VERSION:      0.1
# AUTHOR:       (c) 2014 - 2016 Glutanimate <https://github.com/Glutanimate/>
# DESCRIPTION:  Wrapper script for Michal Fapso's speak.pl Google TTS script
# DEPENDENCIES: - wrapper: xsel libttspico0 libttspico-utils libttspico-data libnotify-bin
#               - speak.pl: libwww-perl libwww-mechanize-perl libhtml-tree-perl sox libsox-fmt-mp3
#
# LICENSE:      GNU GPLv3 (http://www.gnu.de/documents/gpl-3.0.en.html)
#
# NOTICE:       THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
#               EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
#               PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR
#               IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
#               AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND
#               PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE,
#               YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
#
#               IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY
#               COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS
#               PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL,
#               INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE
#               THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED
#               INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE
#               PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER
#               PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
#
# USAGE:        simple_google_tts [-p|-g|-h] languagecode ['strings'|'file.txt']
#
#               please consult the README or the help output (-h) for more information

############ GLOBVAR/PREP ###############

ScriptPath="$(readlink -f "$0")"
ScriptBase="$(basename "$0")"
ParentPath="${ScriptPath%/*}"
speakpl="$ParentPath/speak.pl"

TOP_PID="$$"
PidFile="/tmp/${0##*/}.pid"


############## SETTINGS #################

Player="play"

#############  DIALOGS  ################

Usage="\
$(basename "$0") [-p|-g|-h] languagecode ['strings'|'file.txt']

    -p:   use offline TTS (pico2wave) instead of Google's TTS system
    -g:   activate gui notifications (via notify-send)
    -h:   display this help section

    Selection of valid language codes: en, es, de...
    Check speak.pl for a list of all valid codes

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  ^Y Prev Page  M-\ First Line  M-W WhereIs Next  ^^ Mark Text  M-} Indent Text  M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Linter ^_ Go To Line ^V Next Page  M-/ Last Line  M-] To Bracket  M-^ Copy Text  M-{ Unindent Text M-E Redo
```

Figure 1.7(c) simple_google_tts

```
 GNU nano 2.7.4                                File: simple_google_tts


    If an instance of the script is already running it will be terminated.

    If you don't provide an input string or input file, $(basename "$0")
    will read from the X selection (current/last highlighted text)\
"

GuiIcon="orca"
GuiTitle="Google TTS script"

MsgErrNoSpeakpl="Error: speak.pl not found. Falling back to offline playback."
MsgErrDeps="Error: missing dependencies. Couldn't find:"
MsgInfoExistInstance="Aborting synthesis and playback of existing script instance"
MsgErrNoLang="Error: No language code provided."
MsgInfoInpXsel="Reading from X selection."
MsgInfoInpFile="Reading from text file."
MsgInfoInpString="Reading from string."
MsgErrInvalidInput="Error: Invalid input (file might not be a text file)."
MsgInfoConnOff="No internet connection."
MsgInfoModePico="Using pico2wave for TTS synthesis."
MsgInfoModeGoogle="Using Google for TTS synthesis."
MsgErrInvalidLang="Error: Offline TTS via pico2wave only supports the .\
following languages: en, de, es, fr, it."
MsgErrInputEmpty="Error: Input empty."
MsgInfoSynthesize="Synthesizing virtual speech."
MsgInfoPlayback="Playing synthesized speech"
MsgInfoSectionEmpty="Skipping empty paragraph"
MsgInfoDone="All sections processed. Waiting for playback to finish."

############## FUNCTIONS #################

check_deps () {
    for i in "$@"; do
      type "$i" > /dev/null 2>&1
      if [[ "$?" != "0" ]]; then
        MissingDeps+=" $i"
      fi
    done
}

check_environment () {
    if [[ ! -f "$speakpl" && "$OptOffline" != "1" ]]; then
      notify "$MsgErrNoSpeakpl"
      OptOffline="1"
    fi
    check_deps sox perl
    if [[ -n "$MissingDeps" ]]; then
      notify "${MsgErrDeps}${MissingDeps}"
      exit 1
    fi
}

check_existing_instance(){
  ExistingPID="$(cat "$PidFile" 2> /dev/null)"
  if [[ -n "$ExistingPID" ]]; then
    rm "$PidFile"
    notify "$MsgInfoExistInstance"
    kill -s TERM "$ExistingPID"

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  ^Y Prev Page  M-\ First Line  M-W WhereIs Next  ^^ Mark Text  M-} Indent Text  M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Linter ^_ Go To Line ^V Next Page  M-/ Last Line  M-] To Bracket  M-^ Copy Text  M-{ Unindent Text M-E Redo
```

Figure 1.7(d) simple_google_tts

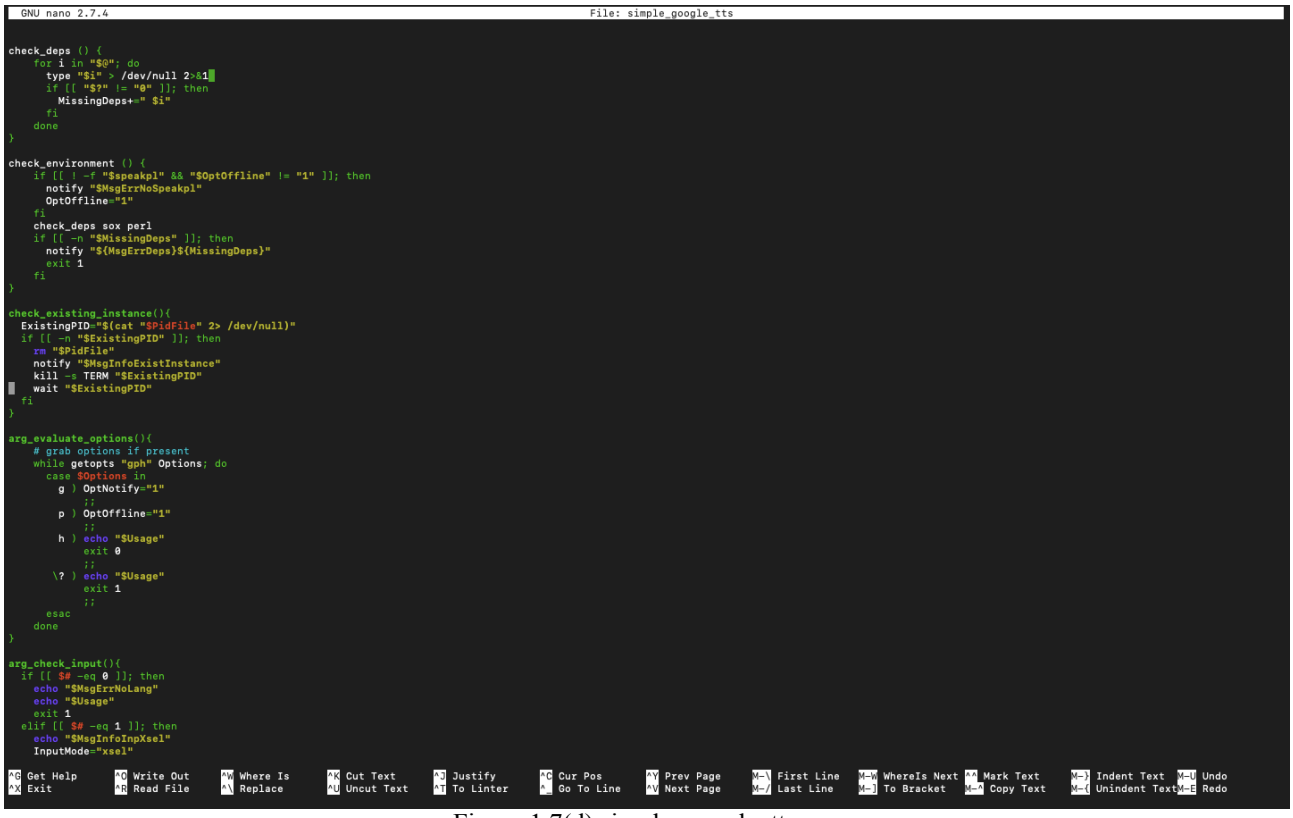


Figure 1.7(d) simple_google_tts

All the scripts combined support the architecture of text to speech conversion executing the commands based on the received text and communication with the client and server.

Bash Scripts – This part is consist of many scripts like:

1. sysON
2. sysOFF
3. sysSTAT
4. and counting

All the scripts in this component are self build and have different intentions, command converted from speech to text is used and that text is matched with the respective script so as the script can easily be executed just with a voice either inside the local area network or outside using the port forwarded by the ngrok in WAN environment.

The sample scripts for:

1. sysON is shown in figure 1.8(a).
2. sysOFF is shown in figure 1.8(b).
3. sysSTAT is shown in figure 1.8(c).

Application Control – The motive of sysON is to communicate with arduino using serial communication and send commands to execute the respective outcome. Hence supporting the IoT technology and using relays to control appliance connected to the high voltage input coming directly from the power socket.

```
  GNU nano 2.7.4

#! /usr/bin/python

import serial
import time
ser = serial.Serial("/dev/ttyACM1", 115200)
ser.write('1')
while(1):
        output = ser.readline()
        print(output)
```

Figure 1.8(a) sysON

```
  GNU nano 2.7.4

#! /usr/bin/python

import serial
ser = serial.Serial("/dev/ttyACM1", 115200)
ser.write('0')
while(1):
        output = ser.readline()
        print(output)
```

Figure 1.8(b) sysOFF

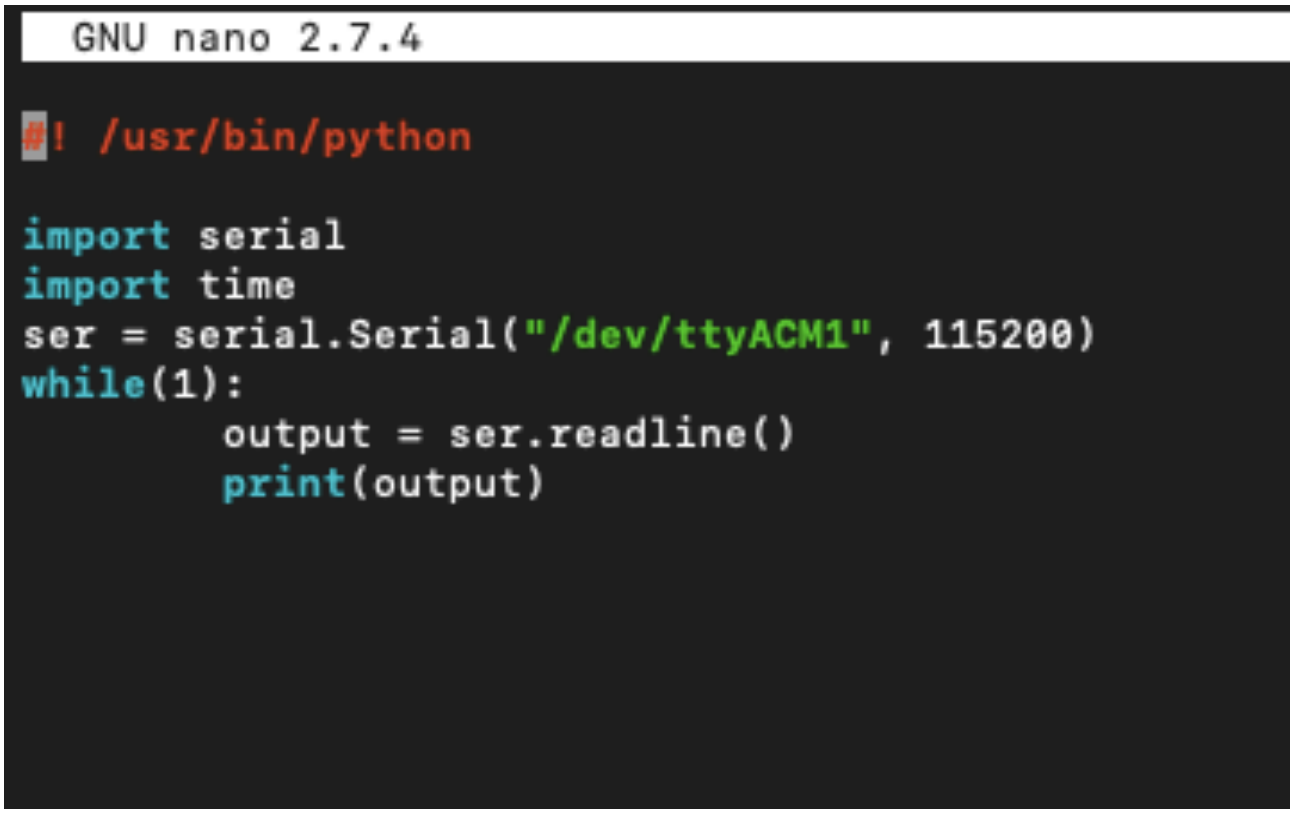


Figure 1.8(c) sysSTAT

VoiceStream script – The motive of this script is to stream the input from the microphone from the server to the target i.e client device and is achieved using commands like ‘arecord’ and ‘aplay’. Which helps is recording the sound and play it directly to the remote client as a live stream in real time, this helps in transfer of the spoken TTS(Text to speech) by the servers speaker which are attached to it using a 3.5MM jack wire.

The command is **‘arecord -D plughw:1,0 -f dat | sshpass -p "rashbir england" ssh -C '+ IP +' aplay -f dat’.**

The script is made executable using same ‘chmode 755 filename’ command and the first line is added with’#!/usr/bin/python3’ which helps the terminal to recognise that the following script requires to be run in python 3 environment.

Hence the python code is used to design the script and os library is imported and os.system(‘terminal command’) is used to execute the terminal command in python environment.

Figure 1.9 shows the voiceScript designed on OS and running on OS.

```
 GNU nano 2.7.4                                                File: voiceStrear

#!/usr/bin/python3

import os

IP = input("Enter the target IP address => ")
os.system('arecord -D plughw:1,0 -f dat | sshpass -p "rashbir england"  ssh -C '+ IP +' aplay -f dat')
```

Figure 1.9 voiceStream script

Microcontroller program – This is the hard coded program uploaded on microcontroller to use its microprocessor to process the hard coded program uploaded to its memory. Here arduino uno is used and the code uploaded is shown in figure 1.10(a) and 1.10(b).

```
pi@raspberrypi: ~   sketch_nov25a | Ardu...   tesingforautomationh...
tesingforautomationhome | Arduino 2:1.0.5+dfsg2-4.1
File Edit Sketch Tools Help
tesingforautomationhome §
#include <dht.h>

dht DHT;
#define DHT11_PIN 7

int incomingBytes = 0;
int fanPin = 5;
int LED = 12;

int manage;

void setup(){
  pinMode(LED, OUTPUT);
  pinMode(fanPin, OUTPUT);
  digitalWrite(fanPin, LOW);
  digitalWrite(LED, HIGH);
  delay(2000);
  digitalWrite(LED, LOW);
  Serial.begin(9600);
}

void loop()
{
  int chk = DHT.read11(DHT11_PIN);
  Serial.print("Temperature = ");
  Serial.print(DHT.temperature);
  Serial.print("  ||  Humidity = ");
  Serial.print(DHT.humidity);

  if(Serial.available() > 0){

    incomingBytes = Serial.read();
    //Serial.println(incomingBytes);
    if(incomingBytes == '1'){
      //Serial.println("  ||  System ON");
   //   Serial.print("Temperature = ");
      // Serial.print(DHT.temperature);
      // Serial.print("  ||  Humidity = ");
      // Serial.print(DHT.humidity);
       digitalWrite(fanPin, HIGH);
       digitalWrite(LED, HIGH);
       manage = 1;
       }

    if(incomingBytes == '0'){
      //Serial.println("  ||  System OFF");
       digitalWrite(fanPin, LOW);
       digitalWrite(LED, LOW);
       manage = 0;
       }
  }

  if (manage == 1){
```

Figure 1.10(a) Arduino Code

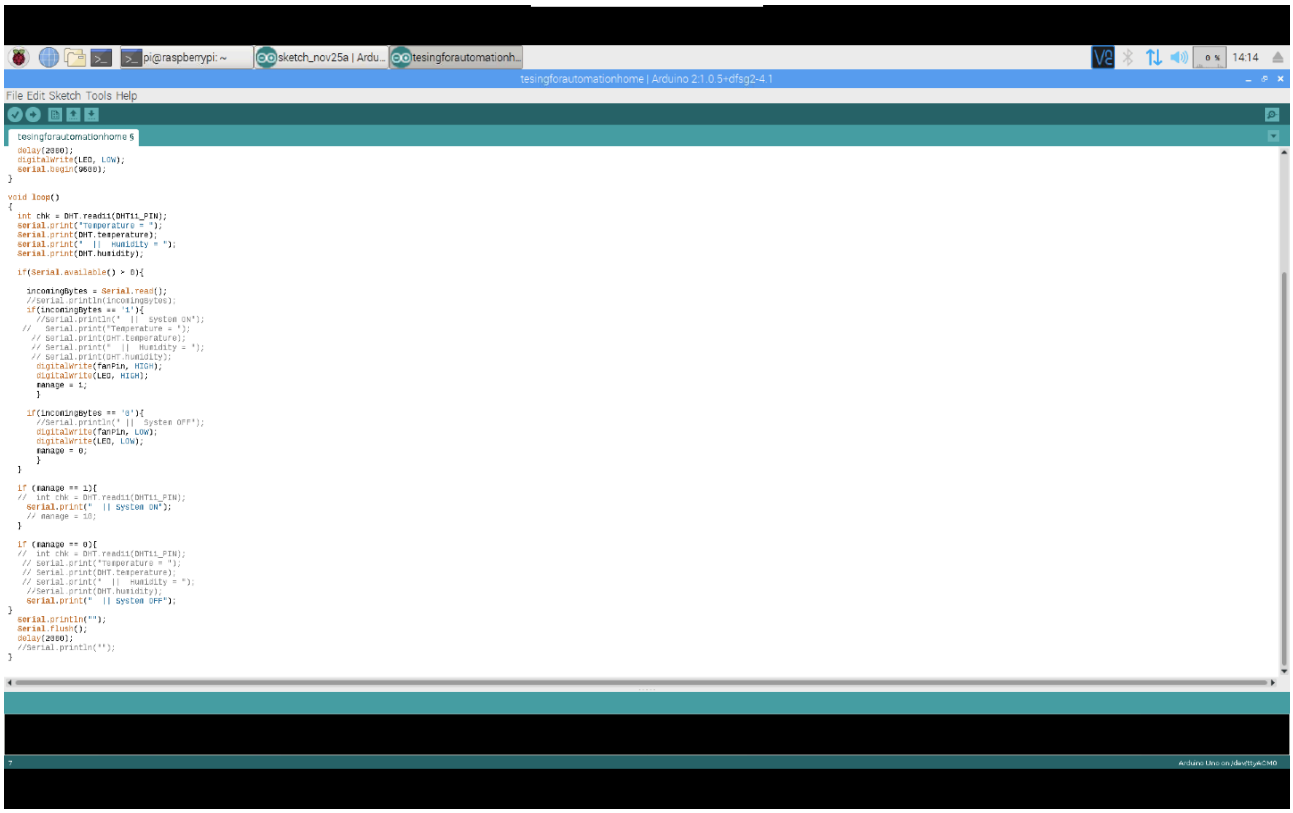


Figure 1.10(b) Ardino Code

Finally figure 1.11 shows the received screen capture in the email that shows the ngrok domain name allocated and ssh port that can be accessed.

The best thing about it is that it always changes, hence providing high security in case of domain leak the server will not be affected as it will reset the domain name. File image here received is called Network.png.

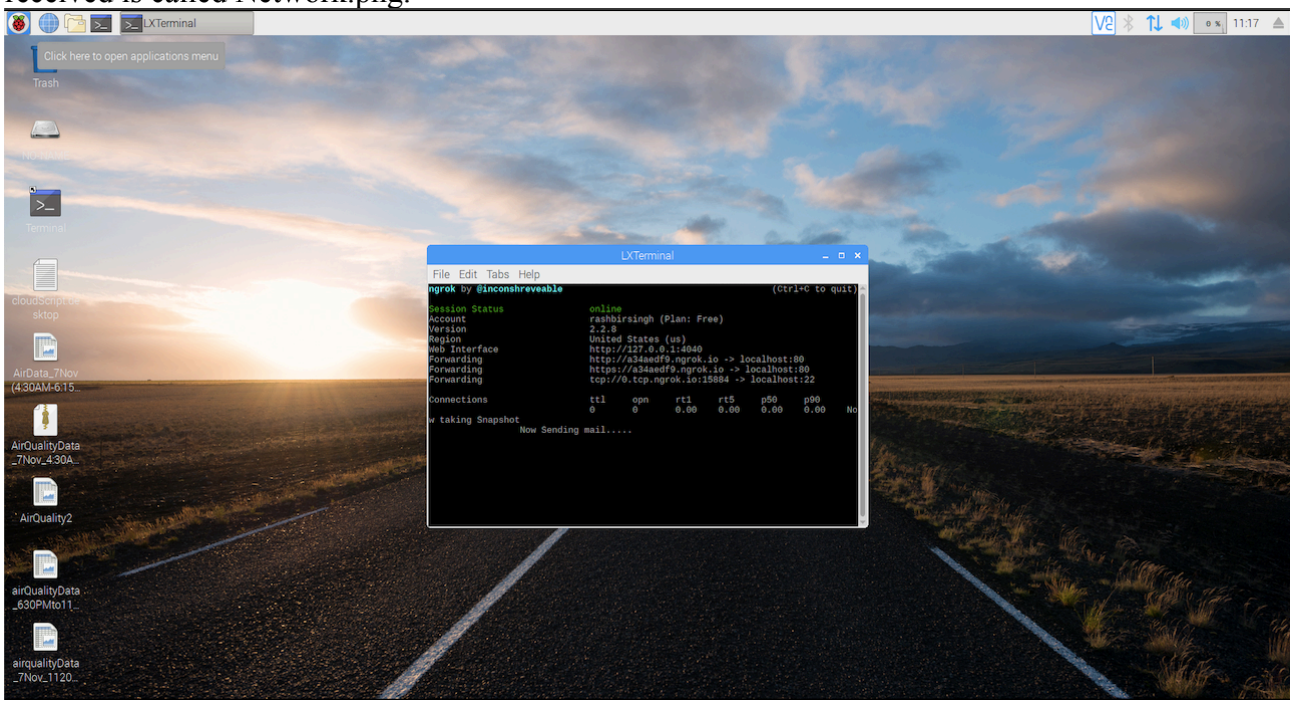


Figure 1.11 Network.png

2.4 Conclusion for chapter 2

The developed system is secure, supports cloud server and IoT. Allow the user to control the environments using speech in LAN or WAN environment but also allow the user to control appliance using EEG i.e controlling the environment just by thinking.

It successfully support computer vision and the developed system is robust and affective. Being running so many technologies and inter compatibility, yet it do not require high knowledge of technology and easy to learn for a non tech person. The developed system can be called an affective black box that supports simple command input and output mechanism.

INTRODUCTION

The three basic needs of the human to survive are air, water and food out of which too air is the most important[10]. But the humans are themselves destroying their survival conditions for present and future by their activities causing pollution of various type one of which is air pollution. The air pollutants could be categorized as pollutant gases (SO_2, NO_2, CO, etc.) and suspended particles (pm10, pm2.5, etc.)[9]. The global burden of disease study estimated 695,000 premature deaths in 2010 due to continued exposure to outdoor particulate matter and ozone pollution for India. By 2030, the expected growth in many of the sectors (industries, residential, transportation, power generation, and construction) will result in an increase in pollution related health impacts for most cities. The available information on urban air pollution, their sources, and the potential of various interventions to control pollution, should help us propose a cleaner path to 2030[1].

Earlier A novel method has been used named Central Limit Theorem and Monte Carlo analysis to directly compare different air standard compliance classification methods by estimating the chronic daily intake of pollutants. This method allows air quality managers to rapidly see how individual classification methods may impact individual population groups, as well as to evaluate different pollutants based on dosage and exposure when complete health impacts are not known. A standard is assumed to be a specific air pollutant, with its threshold concentration value and the averaging time associated with the concentration value. In some cases, pollutants have multiple standards, such as nitrogen dioxide (NO_2), which has a 1-hr average standard (100 ppb, 98th percentile of 1-hr daily maximum concentrations, averaged over 3 years) and an annual average standard (53 ppb, annual mean). The current NAAQS includes 7 air pollutant categories and 12 different standards. [2]

THEORY

An air monitor is assumed to uniformly represent the air quality of the area around it. Probability analyses of combining background concentrations with model-predicted concentrations. *J. Air Waste Manage*recommend that lowest reading in a network area be used for multiple stations. This area could be as small as several hundred square meters in a microscale range of up to 100-m radius from the station, to a middle range (up to 0.5-km radius), neighborhood range (4-km radius), urban range (50-km radius) or regional range (several-hundred-kilometer radius.Siting air monitoring stations is therefore a very important process that incorporates many variables to best represent the local area.[2]

After taking into account potential confounding by other pollutants, we found consistent evidence that the level of PM_{10} is associated with the rate of death from all causes and from cardiovascular and respiratory illnesses. The estimated increase in the relative rate of death from all causes was 0.51 percent (95 percent posterior interval, 0.07 to 0.93 percent) for each increase in the PM_{10} level of 10 µg per cubic meter. The estimated increase in the relative rate of death from cardiovascular and respiratory causes was 0.68 percent (95 percent posterior interval, 0.20 to 1.16 percent) for each increase in the PM_{10} level of 10 µg per cubic meter. There was weaker evidence that increases in ozone levels increased the relative rates of death during the summer, when ozone levels are highest, but not during the winter. Levels of the other pollutants were not significantly related to the mortality rate.

The limitations of our analyses should be considered. Data on levels of $PM_{2.5}$ are not yet available nationally, since a monitoring network for particles in this size range is currently being implemented. We used PM_{10} levels because they have been monitored since 1987; there is variation across the United States in the proportion of PM_{10} mass that is made up of $PM_{2.5}$, so that the PM_{10} level is an imperfect surrogate for the $PM_{2.5}$ level.[3] In addition, for regulatory purposes, PM_{10} levels must only be measured every six days, limiting the extent of available data.
RSPM refers to particulate matter with diameter of less than or equal to 10 micrometres. They are produced from combustion process, vehicles and industrial sources.Dr SumanMor, assistant professor, Department of Environment Studies, PU, said, "RSPM are finer particles which, when inhaled, can cause damage to lungs. Exposure to these can irritate lungs and can cause lung constriction, shortness of breathand cough. It aggravates asthma and other respiratory problems. She added, "People residing in high-risk areas should take precautions such as using face mask while traveling."[4]

.

EXPERIMENTAL SETUP:

Parameter taken for different gases after cleaning data for classification are: For so2-

Range	Parameter
0-40	Low
40-80	Medium
80-120	High
120 above	Very high

For no2

Range	Parameter
0-40	Low
40-80	Medium
80-120	High
120 above	Very high

For rspm

Range	Parameter
0-50	Low
50-100	Medium
100 above	High
200 above	Extreme

METHODOLOGY

Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends . Classification predicts categorical labels, the co-called "classes", while prediction models are used for forecasting continuous valued variables. Basically, both classification and prediction analyses are a two-step process. In the first step, a classification/prediction algorithm is applied on the available data and a decision model is extracted. The mined decision model encapsulates the knowledge lying in the data in a form such as a decision tree, a neural network, a regression model, a support vector machine, etc. This step is usually called model training phase.. This step is usually called model training phase. The second step is the testing phase, which involves the application of the decision model on data for making decisions (predictions). This phase focuses on testing the ability of the decision model to approximate data not used in training. In this respect, both classification and statistical prediction algorithms have been applied for function approximation in several, yet diverse, domains.[3]

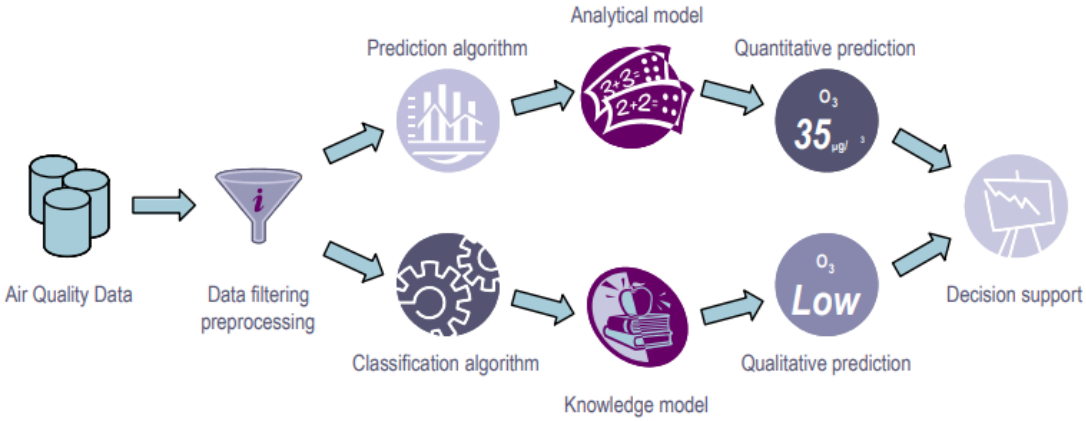
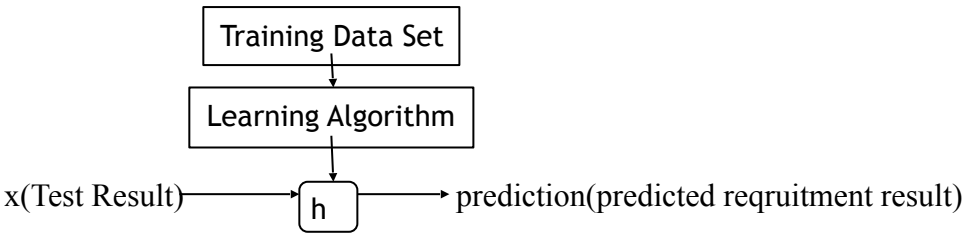


Figure 3. The classical statistical regression algorithms and the classification algorithms approach (upper and lower leg respectively)

The classification problem is much the same as the linear regression problem, with the exception of that the qualities we presently need to predict go up against just few discrete qualities. For instance as we are trying to build applicant classifier for reqruitment selection process then $x^{(i)}$ be some criteria or feature of selection criteria example qualification, apptitude result etc. for an applicant and y=1 or positive/+ve if candidate could be hired and y=0 or negative/-ve otherwise. $x^{(i)}$ and $y^{(i)}$ are also the label for training example.

For any supervised algorithm modal representation is as follow:



where h = hypothesis function to predict the correct value of y

So for the hypothesis for logistic regression is defined as:

$h(x) = g(\Theta^T x)$ (where Θ is the training parameter matix)

$z = \Theta^T x$

$g(z) = 1/(1+e^{-z})$

h(x) in case of logistic regression is known as sigmoid function or logistic function. The range of g(z) and h(x) is interval(0,1).

To find the discrete value of y so that one could clearly classify objects, h(x) value could be treated as:

y=1 if h(x) =>0.5

y=0 if h(x) < 0.5

To measure the accuracy of hypothesis cost function is used. Cost function for Logistic Regression algorithm is :

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \quad \text{if } y = 1$$

$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \quad \text{if } y = 0$$

Cost function makes it sure that J(Θ) is convex(i.e. has single global optima) for classification.

These two cost function cases can be compressed into one to make it simpler.

$$\text{Cost}(h\theta(x), y) = -y\log(h\theta(x)) - (1-y)\log(1-h\theta(x))$$

One can notice that :

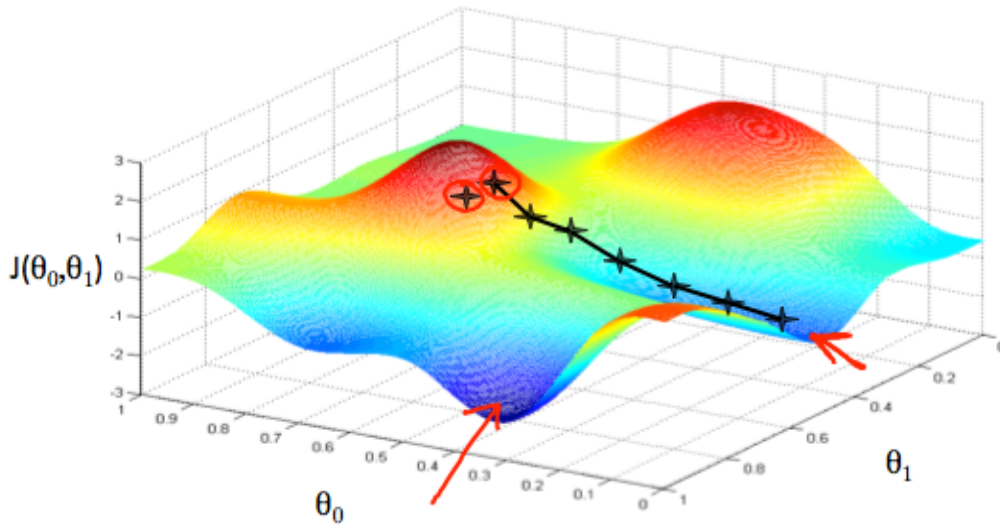
(1-y)log(1-hΘ(x)) = 0 if y=1

and, -ylog(hΘ(x)) =0 if y=0

So, there is no effect on result with simplified computation.

Gradient Descent:

It is used to calculate parameters value(i.e. Θ) for hypothesis function. We need to find parameters values such that they give minimum cost function value.



Graph: Gradient Descent

The gradient Descent algorithm is:

repeat until convergence:

$\Theta_j := \Theta_j - \alpha \, \partial/\partial \, \Theta_j * J(\Theta_0, \Theta_1)$

where j represents feature index.

and α = learning rate

In the graph shown above each point is the value of cost function resulted from specific parameters(Θ_0, Θ_1).The success of gradient descent is achieved when cost function reaches the most bottom point of the graph as that would be the values of parameters(Θ_0, Θ_1) giving minimum/ most convergent cost function value. The direction to move forward at each step is decided on the basis of $\partial/\partial \, \Theta_j * J(\Theta_0, \Theta_1)$ value. Learning rate(α) is the descending or converging rate of parameters towards their minimum value. If α is small that means distance between previous point and current point in graph or previous cost function value and current value.

Before updating values of parameters while calculating gradient descent, all parameter values should be calculated.

The gradient descent for logistic regression/ classification algorithm could be calculated as:

$\Theta := \Theta - \alpha/m * X^T(g(X\Theta) - y^{->})$

where, m = training example

Problem of Overfitting and Underfitting:

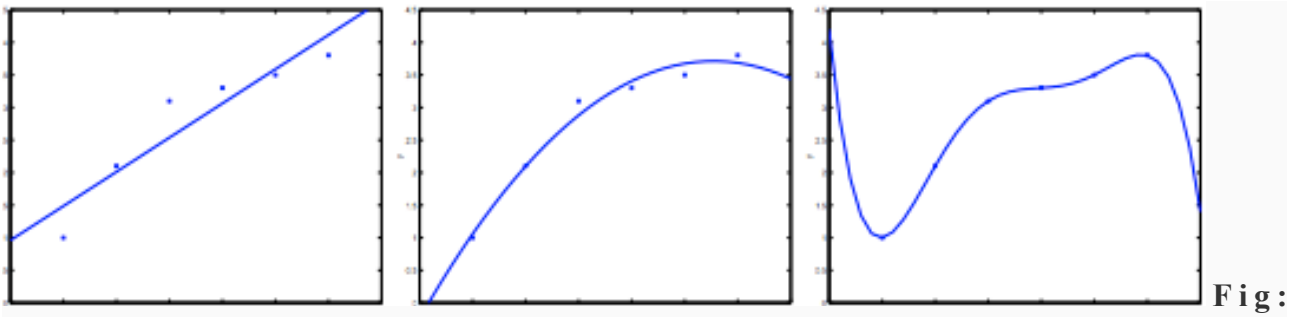


Fig: Underfitting h(x) **Fig: Accurate h(x)** **Fig: Overfitting h(x)**

Let we have to predict y for x € R. As one could observe in left most figure while fitting dataset on $y = \Theta 0+\Theta 1x$ i.e., on straight line, data doesn't fits completely. This problem is known as **Underfitting** or **Biased hypothesis function.** On the other hand in right most figure with $y = \Theta_0 + \Theta_1 x + \Theta_2 x^2 + \Theta_3 x^3 + \Theta_4 x^4 + \Theta_5 x^5$ the model is designed in such a way that y or curve fits the training dataset completely but it will not work great with new data as it is only been designed keeping in mind training dataset. This type of problem is known as **Overfitting** or **Varience hypothesis function.** It is the figure in the middle with $y = \Theta_0 + \Theta_1 x + \Theta_2 x^2$ that demonstrate the best hypothesis function that is neither leaving most of the data points nor covering all, leaving possibility to predict better value of y for some new x.

Reasons for Underfitting:

1. Hypothesis function is too easy or of lower polynomial order.

2. No of features are much smaller than number of training examples i.e, m<<n.

Resolving Underfitting:

Try adding new features to training dataset.

Reasons for Overfitting:

1. No of Training examples are much smaller than number of features i.e., n<<m.

2. Completed hypothesis function creating unnecessary curves and angles while fitting training dataset.

Resolving Overfitting:

1. Use model selection algorithm to select only necessary features.

2. Use regularization method to reduce the magnitude of Θ_j when all features are necessary. It is better to opt regularization over model selection algorithm as complete data is optimized.

DATASET

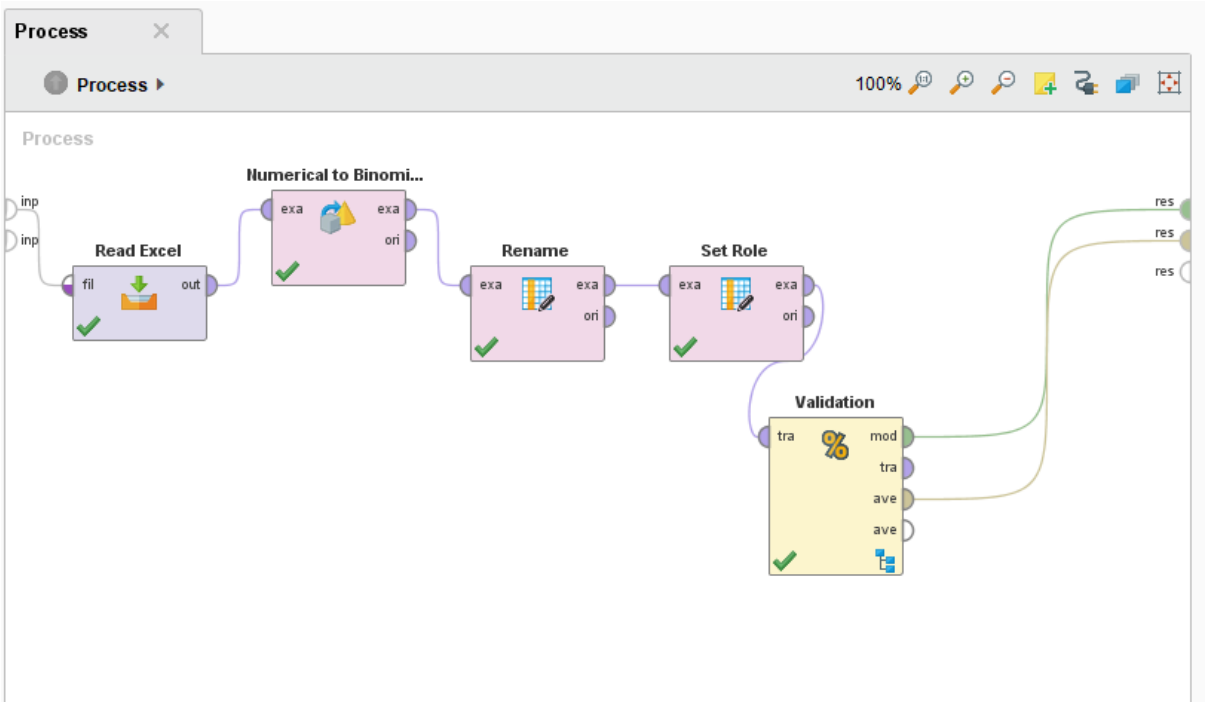


Fig 2.setup for classification in Rapid Minor

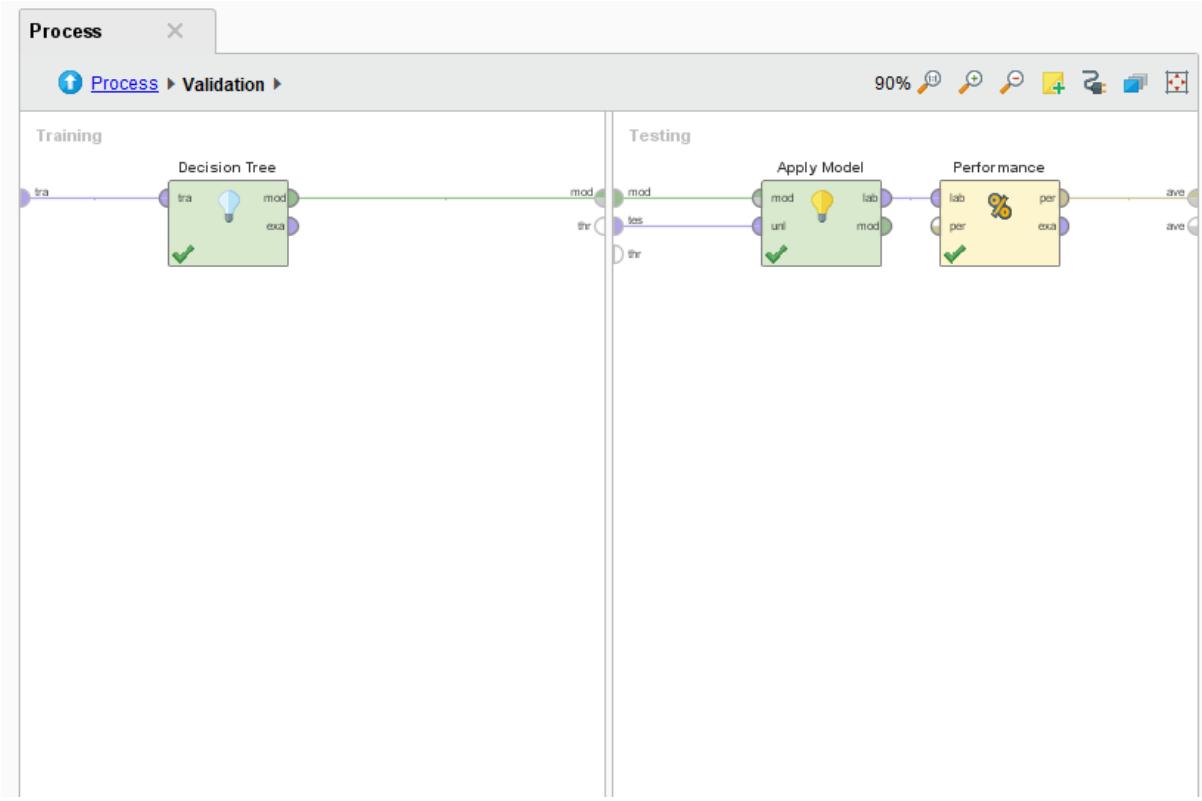


Fig.3 Validation setup for classification in rapid minor

A	B	C	D	E	F	G	H
115	01/01/2015	Andhra Pradesh	Residential, Rural and other Areas	low	low	low	
113	01/01/2015	Andhra Pradesh	Sensitive Area	low	low	low	
106	01/01/2015	Andhra Pradesh	Residential, Rural and other Areas	low	low	low	
104	01/01/2015	Andhra Pradesh	Industrial Area	low	low	low	
103	01/01/2015	Andhra Pradesh	Industrial Area	low	low	low	
102	01/01/2015	Andhra Pradesh	Industrial Area	low	low	low	
99	01/01/2015	Andhra Pradesh	Industrial Area	low	low	low	
91	01/01/2015	Andhra Pradesh	Industrial Area	low	low	low	
88	01/01/2015	Andhra Pradesh	Industrial Area	low	low	medium	
87	01/01/2015	Andhra Pradesh	Residential, Rural and other Areas	low	low	medium	
81	01/01/2015	Andhra Pradesh	Residential, Rural and other Areas	low	low	medium	
77	01/01/2015	Andhra Pradesh	Residential, Rural and other Areas	medium	medium	medium	
109	01/01/2015	Assam	Residential, Rural and other Areas	low	low	low	
92	01/01/2015	Assam	Residential, Rural and other Areas	low	low	low	
84	01/01/2015	Assam	Residential, Rural and other Areas	low	low	medium	
83	01/01/2015	Assam	Residential, Rural and other Areas	low	low	medium	
82	01/01/2015	Assam	Residential, Rural and other Areas	low	low	medium	
80	01/01/2015	Assam	Residential, Rural and other Areas	low	low	medium	
79	01/01/2015	Assam	Residential, Rural and other Areas	low	low	medium	
67	01/01/2015	Assam	Residential, Rural and other Areas	low	low	medium	
54	01/01/2015	Assam	Residential, Rural and other Areas	low	low	medium	
86	01/01/2015	Chandigarh	Residential, Rural and other Areas	low	low	medium	
56	01/01/2015	Chandigarh	Residential, Rural and other Areas	low	low	medium	

Fig. 4 Filtered data set

A	B	C	D	E	F	G	H	I
1	01/01/2015	Uttar Pradesh	Sensitive Area	2	46	399		
2	01/01/2015	Delhi	Industrial Area	4	67	397		
3	01/01/2015	Uttar Pradesh	Sensitive Area	2	37	356		
4	01/01/2015	Rajasthan	Residential, Rural and other Areas	8	19	322		
5	01/01/2015	Uttar Pradesh	Residential, Rural and other Areas	27	51	319		
6	01/01/2015	Uttar Pradesh	Sensitive Area	2	22	300		
7	01/01/2015	Punjab	Residential, Rural and other Areas	10	30	298		
8	01/01/2015	Uttar Pradesh	Residential, Rural and other Areas	7	10	294		
9	01/01/2015	Punjab	Industrial Area	9	28	270		
10	01/01/2015	Uttar Pradesh	Residential, Rural and other Areas	10	30	269		
11	01/01/2015	Rajasthan	Residential, Rural and other Areas	8	55	259		
12	01/01/2015	Rajasthan	Industrial Area	9	39	253		
13	01/01/2015	Uttar Pradesh	Residential, Rural and other Areas	4	48	246		
14	01/01/2015	Rajasthan	Residential, Rural and other Areas	5	26	225		
15	01/01/2015	Himachal Pradesh	Residential, Rural and other Areas	2	19	223		
16	01/01/2015	Rajasthan	Residential, Rural and other Areas	8	25	211		
17	01/01/2015	Rajasthan	Industrial Area	6	24	206		
18	01/01/2015	Madhya Pradesh	Residential, Rural and other Areas	15	31	200		
19	01/01/2015	Uttar Pradesh	Residential, Rural and other Areas	7	43	199		
20	01/01/2015	Punjab	Industrial Area	14	41	190		
21	01/01/2015	Delhi	Residential, Rural and other Areas	4	53	173		
22	01/01/2015	Uttar Pradesh	Industrial Area	24	35	167		
23	01/01/2015	Maharashtra	Residential, Rural and other Areas	15	58	155		

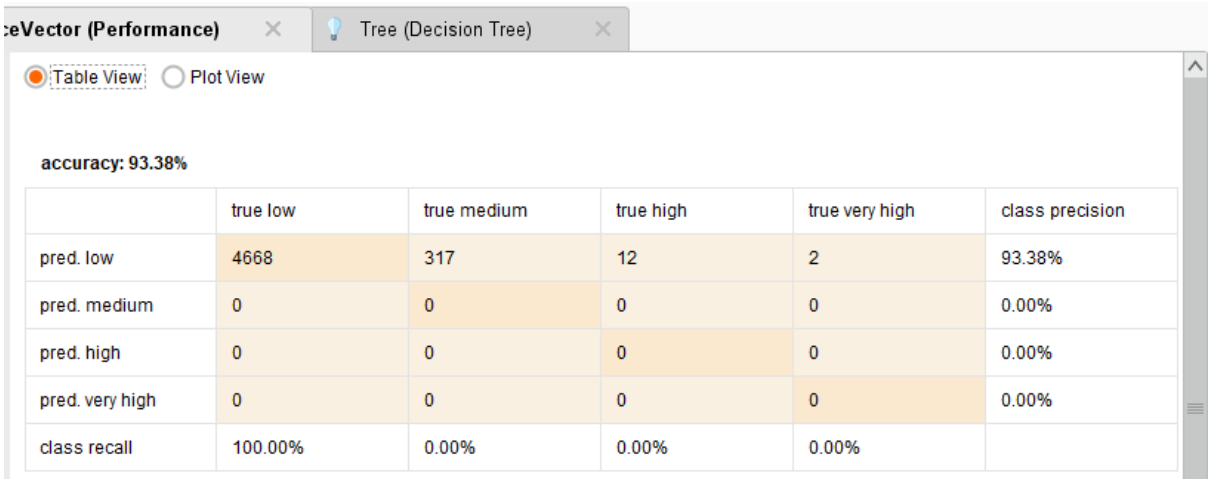
Fig.5 Original data set

	B	C	D	E	F	G	H	I	J
	date	state	area	no2	so2	rspm			
1	01/01/2015	Andhra Pradesh	Residential, Rural and othe	low	low	low			
2	01/01/2015	Andhra Pradesh	Sensitive Area	low	low	low			
3	01/01/2015	Andhra Pradesh	Residential, Rural and othe	low	low	low			
8	01/01/2015	Andhra Pradesh	Industrial Area	low	low	low			
9	01/01/2015	Andhra Pradesh	Industrial Area	low	low	medium			
10	01/01/2015	Andhra Pradesh	Residential, Rural and othe	low	low	medium			
11	01/01/2015	Andhra Pradesh	Residential, Rural and othe	low	low	medium			
12	01/01/2015	Andhra Pradesh	Residential, Rural and othe	medium	medium	medium			
13	01/01/2015	Assam	Residential, Rural and othe	low	low	low			
14	01/01/2015	Assam	Residential, Rural and othe	low	low	low			
15	01/01/2015	Assam	Residential, Rural and othe	low	low	medium			
16	01/01/2015	Assam	Residential, Rural and othe	low	low	medium			
17	01/01/2015	Assam	Residential, Rural and othe	low	low	medium			
18	01/01/2015	Assam	Residential, Rural and othe	low	low	medium			
19	01/01/2015	Assam	Residential, Rural and othe	low	low	medium			
20	01/01/2015	Assam	Residential, Rural and othe	low	low	medium			
21	01/01/2015	Assam	Residential, Rural and othe	low	low	medium			
22	01/01/2015	Chandigarh	Residential, Rural and othe	low	low	medium			
23	01/01/2015	Chandigarh	Residential, Rural and othe	low	low	medium			
24	01/01/2015	Chandigarh	Residential, Rural and othe	low	low	very high			
25	01/01/2015	Delhi	Residential, Rural and othe	low	medium	very high			
26	01/01/2015	Delhi	Industrial Area	low	medium	extreme			

Fig .6 Cleaned data set

RESULT AND CONCLUSION

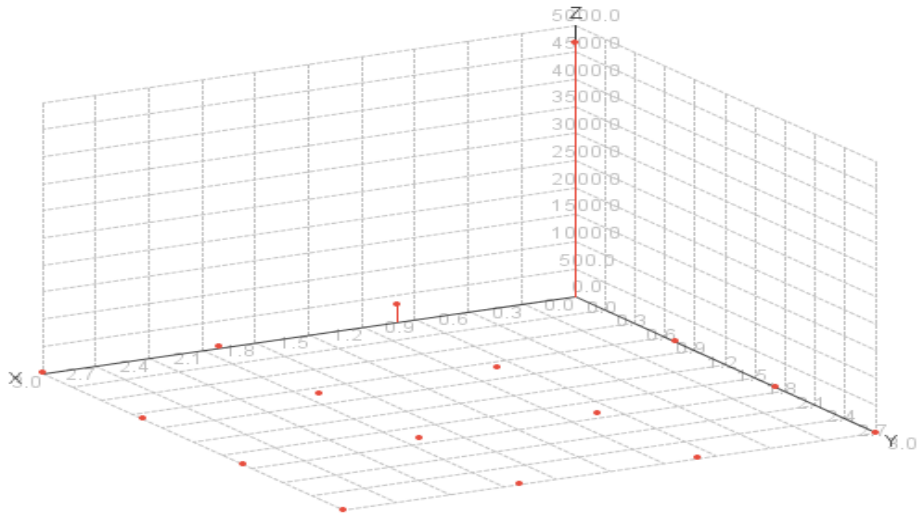
NO2



	true low	true medium	true high	true very high	class precision
pred. low	4668	317	12	2	93.38%
pred. medium	0	0	0	0	0.00%
pred. high	0	0	0	0	0.00%
pred. very high	0	0	0	0	0.00%
class recall	100.00%	0.00%	0.00%	0.00%	

In NO2 it is seen that low data is more, which means amount of No2 in atmosphere in regions of india in 1st six months is low.

Confusion Matrix (x: true class, y: pred. class, z: counters)



SO2

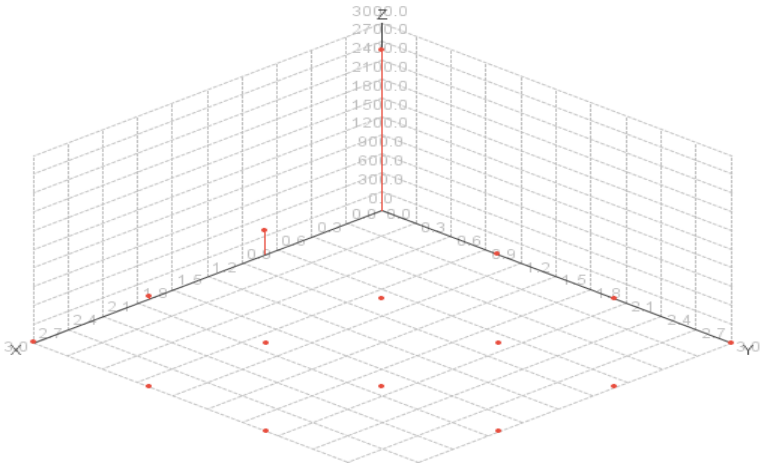
Table View Plot View

accuracy: 85.50%

	true low	true medium	true high	true very high	class precision
pred. low	2565	381	39	15	85.50%
pred. medium	0	0	0	0	0.00%
pred. high	0	0	0	0	0.00%
pred. very high	0	0	0	0	0.00%
class recall	100.00%	0.00%	0.00%	0.00%	

This figure shows that amount of So2 in Regions of india is mostly low and somewhere medium also.

Confusion Matrix (x: true class, y: pred. class, z: counters)

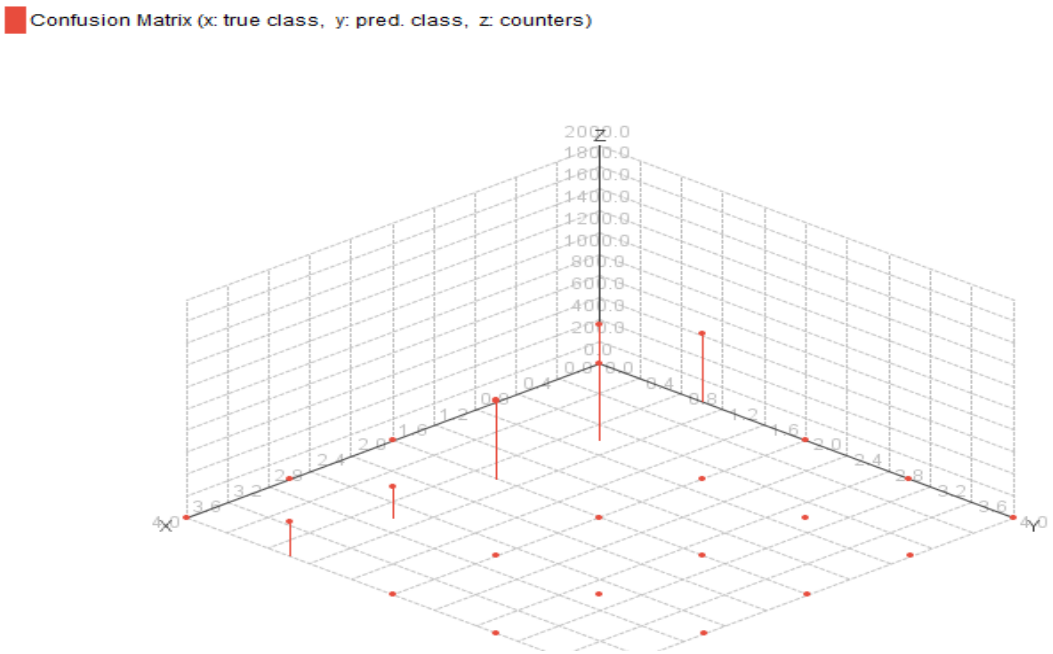


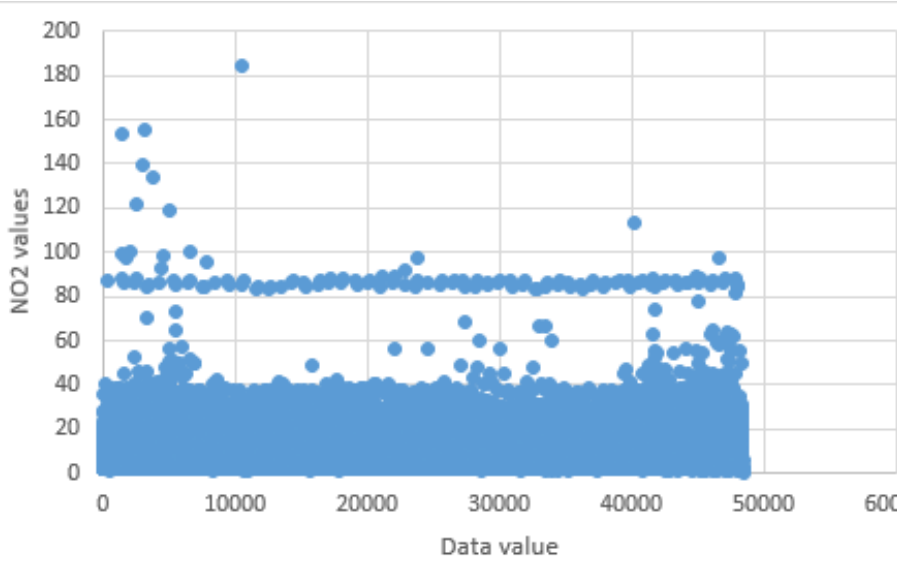
RSPM

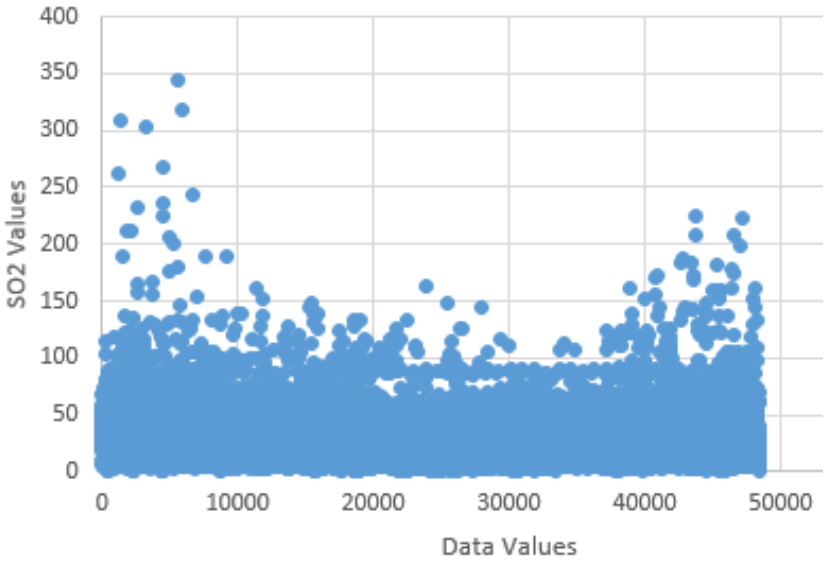
accuracy: 35.13%

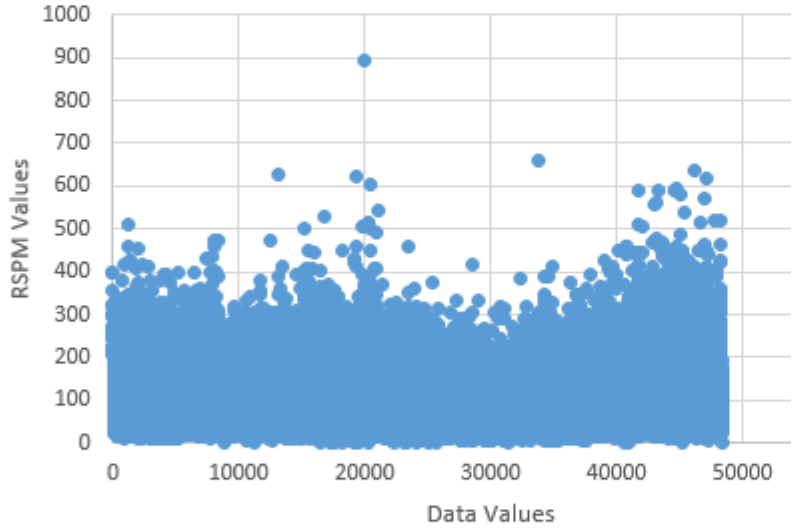
	true low	true medium	true very high	true extreme	true high	class precision
pred. low	0	0	0	0	0	0.00%
pred. medium	625	1054	721	283	317	35.13%
pred. very high	0	0	0	0	0	0.00%
pred. extreme	0	0	0	0	0	0.00%
pred. high	0	0	0	0	0	0.00%
class recall	0.00%	100.00%	0.00%	0.00%	0.00%	

The rspm value for medium prediction is highest and in some regions is low. In very less areas high rspm value is high.









The conclusion derived from these graphs and classification is that the value of rspm and So2 is high in atmosphere.

REFERRENCE

[1] Guttikunda, S.K., Goel, R. and Pant, P., 2014. Nature of air pollution, emission sources, and management in the Indian cities. *Atmospheric environment*, *95*, pp.501-510.

[2] Freeman, B., McBean, E., Gharabaghi, B. and Thé, J., 2017. Evaluation of air quality zone classification methods based on ambient air concentration exposure. *Journal of the Air & Waste Management Association*, *67*(5), pp.550-564.

[3]Athanasiadis, I.N., Karatzas, K.D. and Mitkas, P.A., 2006, August. Classification techniques for air quality forecasting. In *Fifth ECAI Workshop on Binding Environmental Sciences and Artificial Intelligence, 17th European Conf on Artificial Intelligence*

[4]https://indianexpress.com/article/cities/chandigarh/level-of-rspm-in-city-air-above-permissible-limit-shows-data/

[9] Yang, W., Chen, X. and Liao, Q., 2014, November. Evaluation of PM2. 5 and PM10 using normalized first-order absolute sum of high-frequency spectrum. In *Smart Computing (SMARTCOMP), 2014 International Conference on* (pp. 61-65). IEEE.

[10] Chen, L.J., Ho, Y.H., Lee, H.C., Wu, H.C., Liu, H.M., Hsieh, H.H., Huang, Y.T. and Lung, S.C.C., 2017. An Open Framework for Participatory PM2. 5 Monitoring in Smart Cities. *IEEE Access*, *5*, pp.14441-14454.