

A PROJECT REPORT ON
HUMAN EMOTION RECOGNITION
SUBMITTED IN PARTIAL FULFILLMENT OF THE
DEGREE OF
BACHELOR OF SCIENCE (MSD)

BY

B.ASHWITHA- 562151636

S.RAKSHITHA- 562151639

M.NAVYA SREE- 562151643

DEPARTMENT OF COMPUTER SCIENCE



R. B. V.R. R WOMEN'S COLLEGE (AUTONOMOUS)

(Affiliated to Osmania University, Hyderabad)

2022-2023



**RAJA BAHADUR VENKAT RAMA REDDY
WOMEN'S COLLEGE
(AUTONOMOUS)
Reaccredited by NAAC with 'B++' Grade
COLLEGE WITH POTENTIAL FOR EXCELLENCE
NARAYANGUDA, HYDERABAD-27 TS**

CERTIFICATE

This is to certify that the following students of ***B.Sc. MSDs, III
Year, VI Semester***

***B.ASHWITHA-562151636
S. RAKSHITHA-562151639,
M. NAVYA SREE-562151643***

has successfully completed "HUMAN EMOTION RECOGNITION"
in partialfulfilment of the requirements for the award of the
Degree in B.Sc. MSDs for the academic year 2023.

**Signature
Internal Examiner**

**Signature
External Examiner**

**Signature
Head of the
Department**



**RAJA BAHADUR VENKAT RAMA REDDY
WOMEN'S COLLEGE
(AUTONOMOUS)
Reaccredited by NAAC with 'B++' Grade
COLLEGE WITH POTENTIAL FOR EXCELLENCE
NARAYANGUDA, HYDERABAD-27 TS**

CERTIFICATE

This is to certify that the project report entitled **“Human Emotion Recognition”**, being submitted by,, ,B. ASHWITHA, S. RAKSHITHA , M. NAVYA SREE , bearing Roll Number. 562151636,562151639, 562151643 in partial fulfillment of the requirements for the award of the degree of Bachelor's in Science (MSD), is a record of bonafide work carried out by her in academic year 2022-2023. The results are verified and found satisfactory.

Internal Examiner

External Examiner



**RAJA BAHADUR VENKAT RAMA REDDY
WOMEN'S COLLEGE
(AUTONOMOUS)**

**Reaccredited by NAAC with 'B++' Grade
COLLEGE WITH POTENTIAL FOR EXCELLENCE
NARAYANGUDA, HYDERABAD-27 TS**

DATE: 5/05/2023

No objection Certificate

B.ASHWITHA, S. RAKSHITHA, M. NAVYA SREE bearing the Hall ticket number: **562151636, 562151639, 562151643** is the student of B.Sc. (MSD) Final Year and is intending to pursue a project work in your esteemed organization. It will be obliging if you permit our student to take up a project work which is the part of their course curriculum for the partial fulfillment of the B.Sc. degree.

T. Vamshi Mohana Reddy

Head, Department of Computer Science

Declaration

I,B. ASHWITHA, S. RAKSHITHA , M. NAVYA SREE , hereby declare that the project entitled HUMAN EMOTION RECOGNITION under the guidance of T. MADHAVI , Assistant Professor, Department of Computer Science, Hyderabad.

This project is submitted for partial fulfillment “Bachelor’s of Science (MSD) “. This report has not been submitted for any degree or diploma of any university.

ACKNOWLEDGEMENT

It gives immense pleasure to express my deep gratitude to our guide who has been kind enough to guide us in planning of this project, encouragement, timely guidance and invaluable. I am thankful to the staff of R.B.V.R.R Women's College in execution of the project.

I thank Mrs. Vamshi Mohana Madam, Head, Department of Computer Science and all the faculty for their suggestions throughout the project. I also thank all technical staff who helped us to complete our project successfully.

ABSTRACT

In today's world everything depends on human efforts. But artificial intelligence makes human work easier. It will think like a human not exactly depend upon how we are training the algorithm. Face emotion recognition is one of the challenging tasks. Artificial intelligence will predict the emotion of the human without human interaction. Artificial intelligence have different concepts like image processing, deep learning, machine learning. These topics contains different algorithms some of them for classifying the text data and some of them for detecting images.

In this project we predict the face emotions using CNN (Convolution neural network) algorithm. It is more efficient than previous algorithms.

LIST OF CONTENTS

1. INTRODUCTION
2. LITERATURE REVIEW
3. SYSTEM ANALYSIS
4. INPUT AND OUTPUT DESIGN
5. SYSTEM STUDY
6. UML DIAGRAMS
 - 6.1 GOALS
 - 6.2 USE CASE DIAGRAMS
 - 6.3 CLSS DIAGRAMS
 - 6.4 SEQUENCE DIAGRAM
 - 6.5 ACTIVITY DIAGRAM
7. MODULES
8. SOFTWARE ENVIRONMENT
9. SYSTEM TESTING
10. METHODOLOGY
11. SOURCE CODE
12. RESULTS
13. CONCLUSION
14. REFERENCES

INTRODUCTION

Since facial expressions correspond to emotions, they are important markers for human feelings. The majority of the time (roughly 55 percent of the time) , the facial expression is a nonverbal form of expressing sentiment, and it can be positive or negative. as a way of deciding whether or not a person is guilty of a crime Whether a person is telling the truth or not. Current techniques predominantly concentrate on facial investigation while keeping the context intact, resulting in a slew of needless and confusing features that muddle CNN training.

The current paper focuses on four main facial expression classes: displeasure/anger, sadness/unhappiness, smiling/happiness, fear, and surprised/astonished. The FEREC algorithm mentioned in this paper is intended for expressional analysis and classification of a given image into these four basic emotion groups. There are two main approaches to facial expression detection that have been recorded. The first is to differentiate expressions that are identified with an explicit classifier, and the second is to characterise based on the extracted facial highlights. Action units are used as speech markers in the facial action coding scheme (FACS) . Facial muscle shifts were used to differentiate these AUs.

LITERATURE REVIEW

Deep learning in neural networks: An overview

AUTHORS: Juergen Schmidhuber

In recent years, deep artificial neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarises relevant work, much of it from the previous millennium. Shallow and deep learners are distinguished by the depth of their credit assignment paths, which are chains of possibly learnable, causal links between actions and effects. I review deep supervised learning (also recapitulating the history of backpropagation), unsupervised learning, reinforcement learning & evolutionary computation, and indirect search for short programs encoding deep and large networks.

Acoustic modeling using deep belief networks

AUTHORS: A.-R. Mohamed, G. E. Dahl, and G. Hinton

Gaussian mixture models are currently the dominant technique for modeling the emission distribution of hidden Markov models for speech recognition. We show that better phone recognition on the TIMIT dataset can be achieved by replacing Gaussian mixture models by deep neural networks that contain many layers of features and a very large number of parameters. These networks are first pre-trained as a multi-layer generative model of a window of spectral feature vectors without making use of any discriminative information. Once the generative pre-training has designed the features, we perform discriminative fine-tuning using backpropagation to adjust the features slightly to make them better at predicting a probability distribution over the states of monophone hidden Markov models.

A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion

AUTHORS: L. Deng, O. Abdel-Hamid, and D. Yu

We develop and present a novel deep convolutional neural network architecture, where heterogeneous pooling is used to provide constrained frequency-shift invariance in the speech spectrogram while minimizing speech-class confusion induced by such invariance. The design of the pooling layer is guided by domain knowledge about how speech classes would change when formant frequencies are modified. The convolution and heterogeneous-pooling layers are followed by a fully connected multi-layer neural network to form a deep architecture interfaced to an HMM for continuous speech recognition. During training, all layers of this entire deep net are regularized using a variant of the “dropout” technique. Experimental evaluation demonstrates the effectiveness of both heterogeneous pooling and dropout regularization. On the TIMIT phonetic recognition task, we have achieved an 18.7% phone error rate, lowest on this standard task reported in the literature with a single system and with no use of information about speaker identity. Preliminary experiments on large vocabulary speech recognition in a voice search task also show error rate reduction using heterogeneous pooling in the deep convolutional neural network.

Facial expression is the common signal for all humans to convey the mood. There are many attempts to make an automatic facial expression analysis tools as it has application in many fields such as robotics, medicine, driving assist systems, and lie detector . Since the twentieth century, Ekman et al. defined seven basic emotions, irrespective of culture in which a human grows with the seven expressions (anger, feared, happy, sad, contempt , disgust, and surprise). In a recent study on the facial recognition technology (FERET) dataset, Sajid et al. found out the impact of facial asymmetry as a marker of age estimation . Their finding states that right face asymmetry is better compared to the left face asymmetry. Face pose appearance is still a big issue with face detection. Ratyal et al. provided the solution for variability in facial pose appearance. They have used three-dimensional pose invariant approach using subject-specific descriptors . There are many issues like excessive makeup pose and expression which are solved using convolutional networks. Recently, researchers have made extraordinary accomplishment in

facial expression detection , which led to improvements in neuroscience and cognitive science that drive the advancement of research, in the field of facial expression. Also, the development in computer vision and machine learning makes emotion identification much more accurate and accessible to the general population. As a result, facial expression recognition is growing rapidly as a sub-field of image processing. Some of the possible applications are human–computer interaction , psychiatric observations , drunk driver recognition , and the most important is lie detector .

SYSTEM ANALYSIS

EXISTING SYSTEM:

The current system is an automated facial expression recognition system that can recognise seven universal emotions. It's made to be person-independent and only works with static pictures.

Using the Viola-jones algorithm, the machine incorporates a face recognition function. Uses a Multi-layer feed forward multi layer perceptron to extract uniform Gabor features and perform classification.

PROPOSED SYSTEM:

In this paper, we present a facial expression recognition method based on Convolutional Neural Networks (CNN). An picture is fed into our machine, and we use CNN to predict the facial expression mark, which should be one of the following: rage, happiness, fear, sorrow, disgust, or neutral.

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

SOFTWARE REQUIREMENTS:

- **Operating System:** Windows
- **Coding Language:** Python 3.7

TOOLS AND LIBRARIES USED

OpenCV:

For image transformation functions like converting an image to grayscale, we'll use the OpenCV library. It's an open source library with a wide range of algorithm implementations that can be used for a number of image functions. OpenCV supports C++ and Python as programming languages.

Dlib:

Dlib is a popular image-processing library that can be used with Python, C++, and other programming languages. The main purpose of this library is to detect faces, extract features, match features, and so on. Other domains such as machine learning, threading, GUI, and networking are also supported.

Python:

Python is a versatile scripting language that comes in handy when dealing with statistical problems that include machine learning algorithms. It has a number of utility functions that aid in the preprocessing process. Processing is fast and works on nearly all platforms. It's simple to integrate with C++ and other picture libraries, and it comes with built-in functions and libraries for storing and manipulating data of all kinds. It includes the pandas and numpy frameworks, which allow us to manipulate data as required. Numpy arrays, which can hold n-dimensional data, can be used to build a good feature set.

Scikit-learn:

In Python, Scikit-learn is a machine learning library. Matplotlib, numpy, and a variety of machine learning algorithms are included. The API is easy to use and comprehend. It has a number of functions for analysing and plotting results. Many of its feature reduction, feature value, and feature selection functions can be used to create a good feature collection. Its algorithm can be applied to a variety of problems.

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.

3. Create document, report, or other formats that contain information produced by the system.

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.

SYSTEM STUDY

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.

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose 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 Meta-model 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.

GOALS:

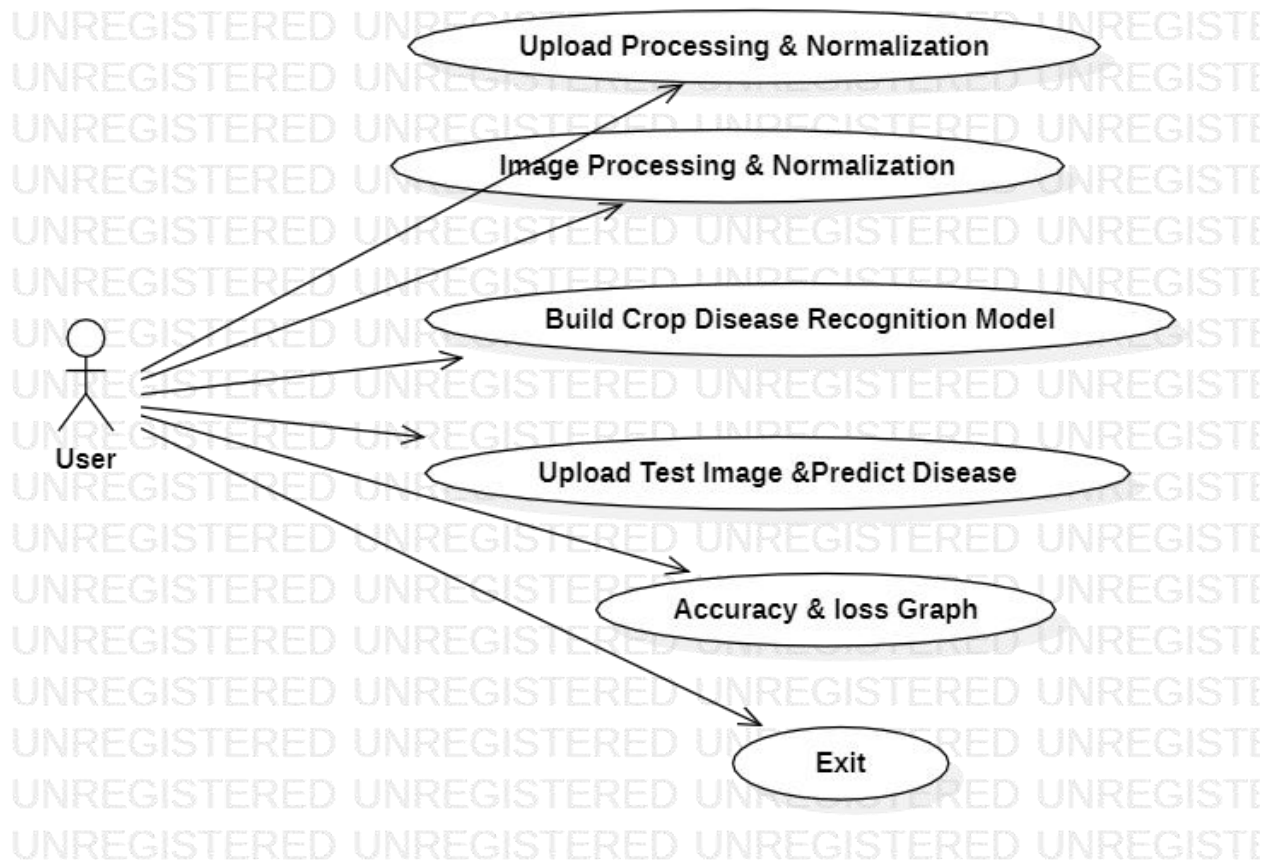
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.

3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

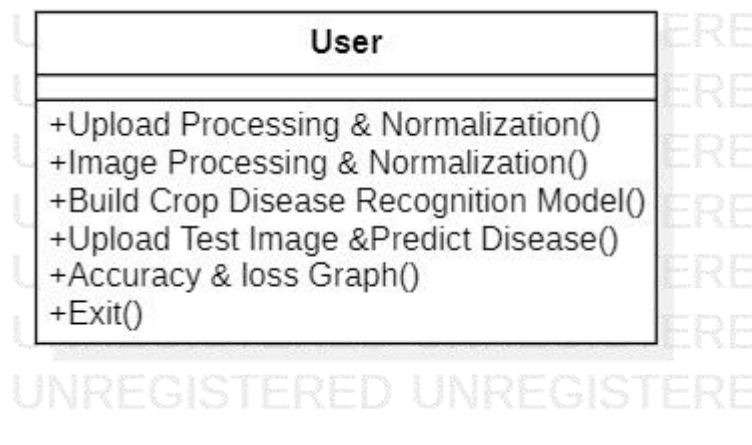
USE CASE DIAGRAM:

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.



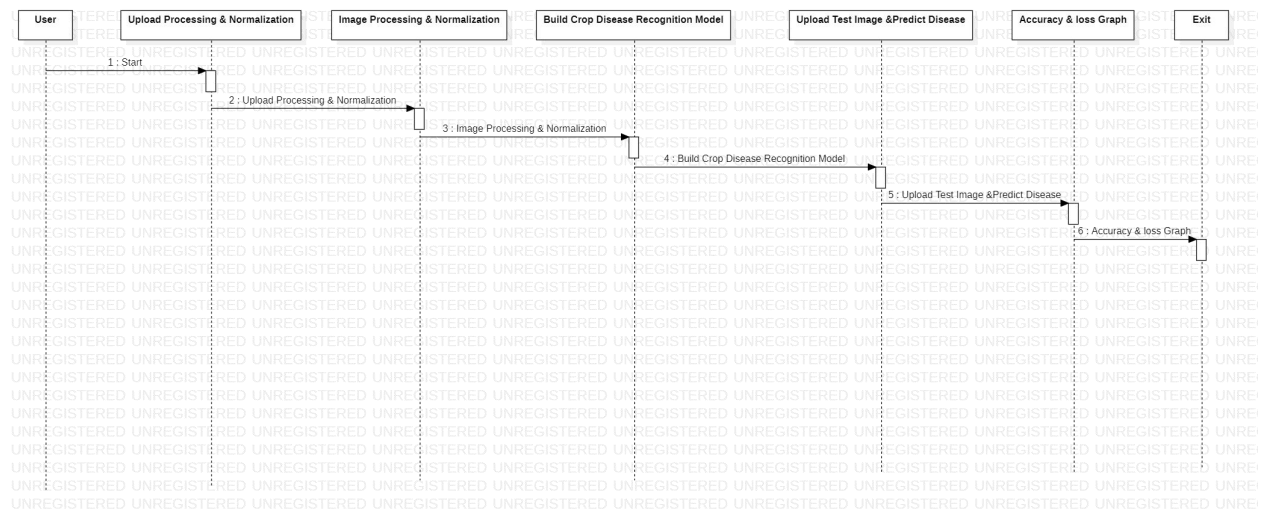
CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



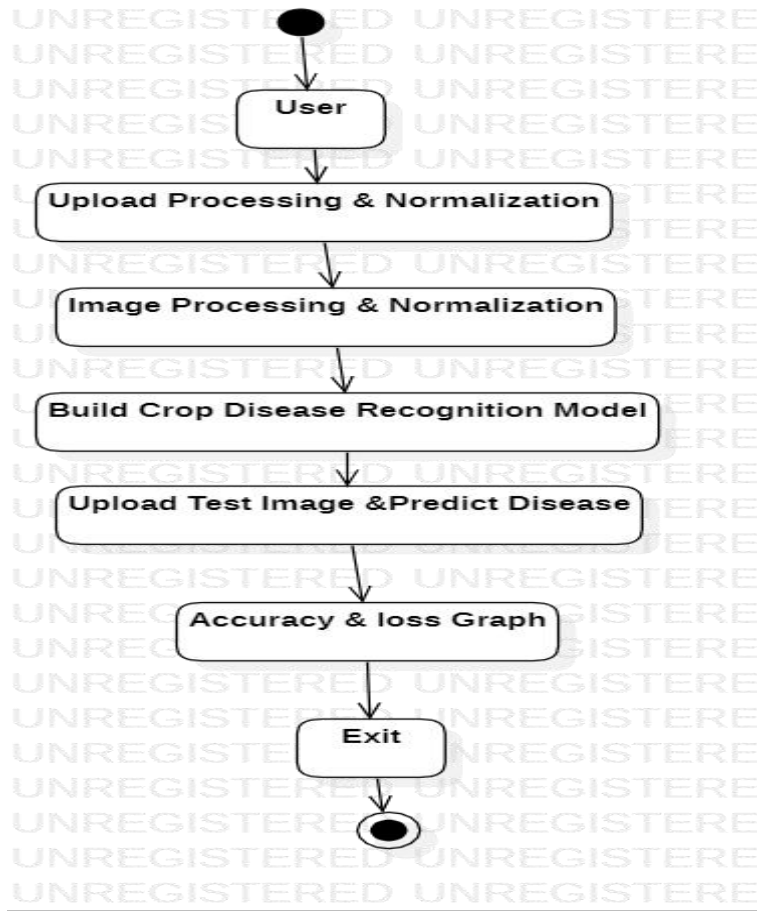
SEQUENCE DIAGRAM:

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.



ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



MODULES

Load Dataset:

Load data set using pandas read_csv() method.

.Split Data Set:

Split the data set to two types. One is train data test and another one is test data set.

Train data set:

Train data set will train our data set using fit method.

Test data set:

Test data set will test the data set using algorithm.

Predict data set:

Predict() method will predict the results.

SOFTWARE ENVIRONMENT

Python Introduction

Python is a general purpose, dynamic, high level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is *easy to learn* yet powerful and versatile scripting language, which makes it attractive for Application Development.

Python's syntax and *dynamic typing* with its interpreted nature make it an ideal language for scripting and rapid application development.

Python supports *multiple programming pattern*, including object-oriented, imperative, and functional or procedural programming styles.

Python is not intended to work in a particular area, such as web programming. That is why it is known as *multipurpose* programming language because it can be used with web, enterprise, 3D CAD, etc.

We don't need to use data types to declare variable because it is *dynamically typed* so we can write `a=10` to assign an integer value in an integer variable.

Python makes the development and debugging *fast* because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

Python 2 vs. Python 3

In most of the programming languages, whenever a new version releases, it supports the features and syntax of the existing version of the language, therefore, it is easier for the projects to switch in the newer version. However, in the case of Python, the two versions Python 2 and Python 3 are very much different from each other.

A list of differences between Python 2 and Python 3 are given below:

1. Python 2 uses **print** as a statement and used as `print "something"` to print some string on the console. On the other hand, Python 3 uses **print** as a function and used as `print("something")` to print something on the console.
2. Python 2 uses the function `raw_input()` to accept the user's input. It returns the string representing the value, which is typed by the user. To convert it into the integer, we need to use the `int()` function in Python. On the other hand, Python 3 uses `input()` function which automatically interpreted the type of input entered by the user. However, we can cast this value to any type by using primitive functions (`int()`, `str()`, etc.).

3. In Python 2, the implicit string type is ASCII, whereas, in Python 3, the implicit string type is Unicode.
4. Python 3 doesn't contain the xrange() function of Python 2. The xrange() is the variant of range() function which returns a xrange object that works similar to Java iterator. The range() returns a list for example the function range(0,3) contains 0, 1, 2.
5. There is also a small change made in Exception handling in Python 3. It defines a keyword **as** which is necessary to be used. We will discuss it in Exception handling section of Python programming tutorial.

[next](#) → ← [prev](#)

Python Features

Python provides lots of features that are listed below.

1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

5) Free and Open Source

Python language is freely available at [official web address](#). The source-code is also available. Therefore it is open source.

6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

9) GUI Programming Support

Graphical user interfaces can be developed using Python.

10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

Python History and Versions

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
- In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- Python 2.0 added new features like: list comprehensions, garbage collection system.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
- *ABC programming language* is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- Python is influenced by following programming languages:
 - ABC language.
 - Modula.

Python programming language is being updated regularly with new features and supports. There are lots of updations in python versions, started from 1994 to current release.

[next](#) → ← [prev](#)

Python Applications

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifying applications areas where python can be applied.

1) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, BeautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web

based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

2) Desktop GUI Applications

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

3) Software Development

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

4) Scientific and Numeric

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

5) Business Applications

Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high level application platform.

6) Console Based Application

We can use Python to develop console based applications. For example: **IPython**.

7) Audio or Video based Applications

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

8) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

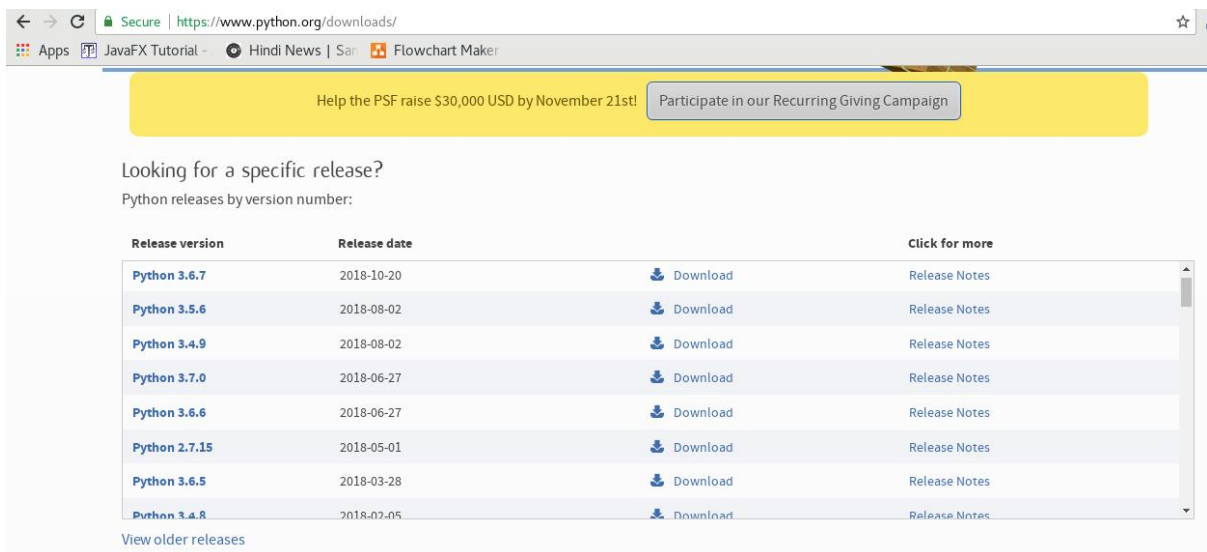
[next](#) → ← [prev](#)

How to Install Python (Environment Set-up)

In this section of the tutorial, we will discuss the installation of python on various operating systems.

Installation on Windows

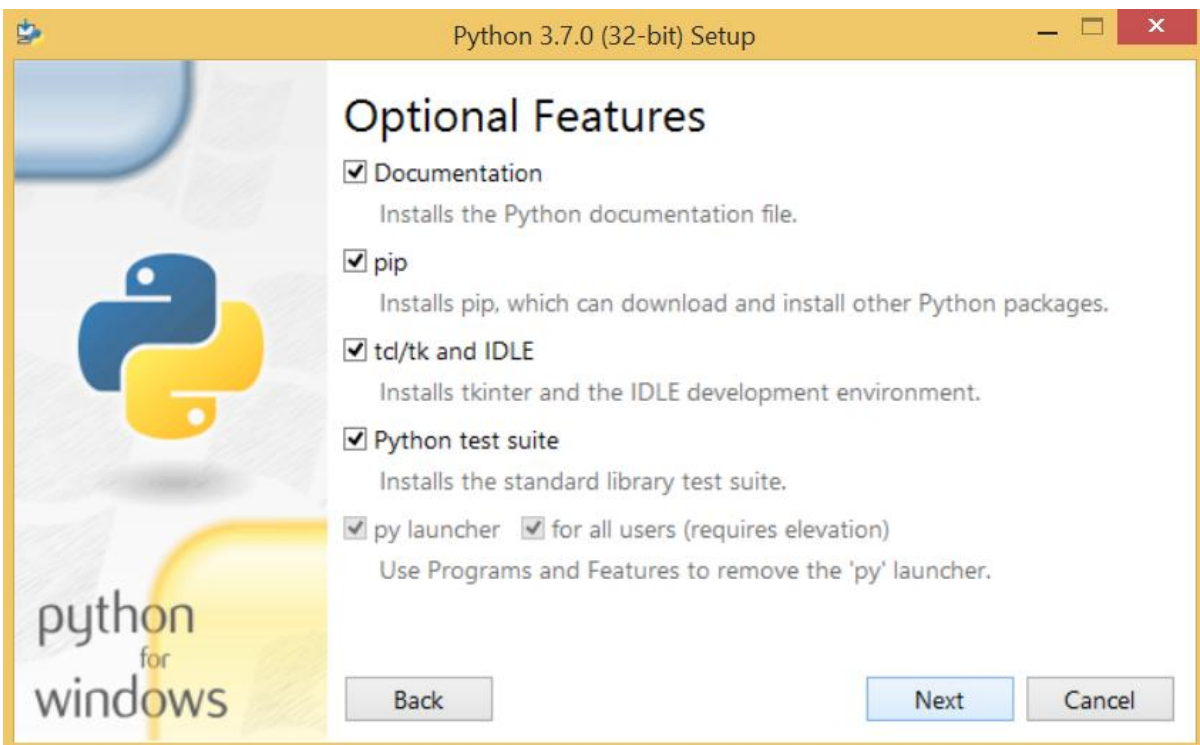
Visit the link <https://www.python.org/downloads/> to download the latest release of Python. In this process, we will install Python 3.6.7 on our Windows operating system.



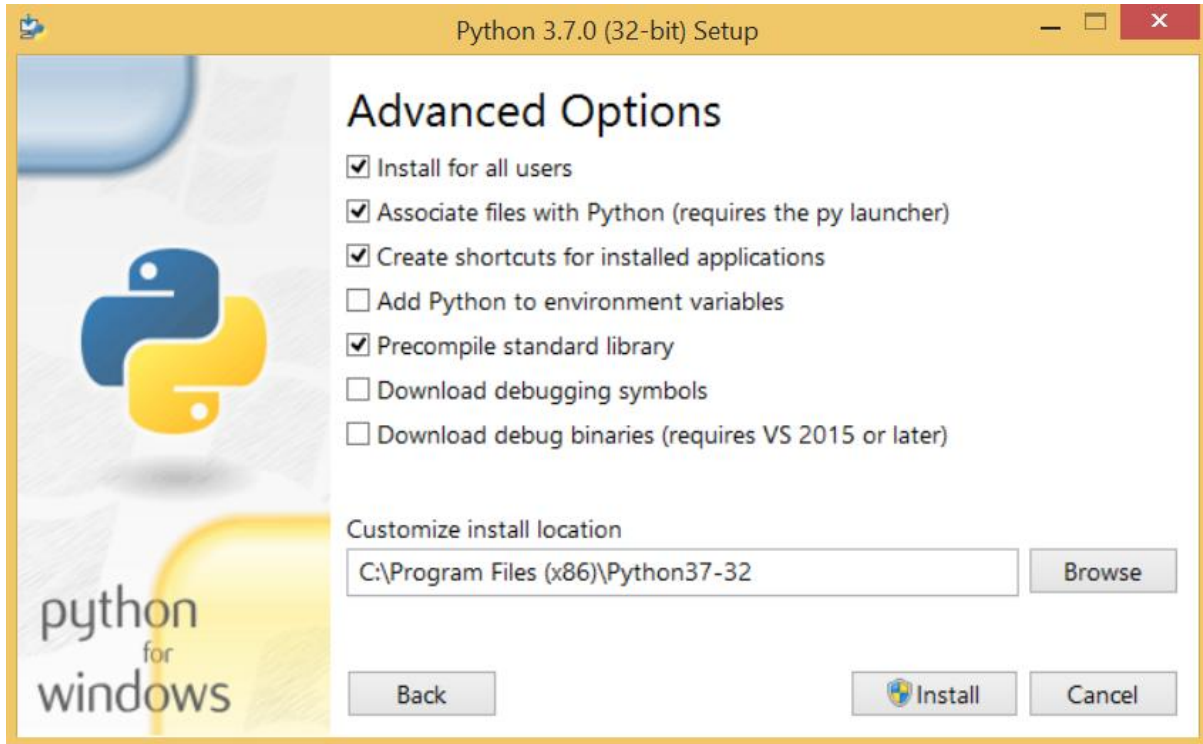
Double-click the executable file which is downloaded; the following window will open. Select Customize installation and proceed.



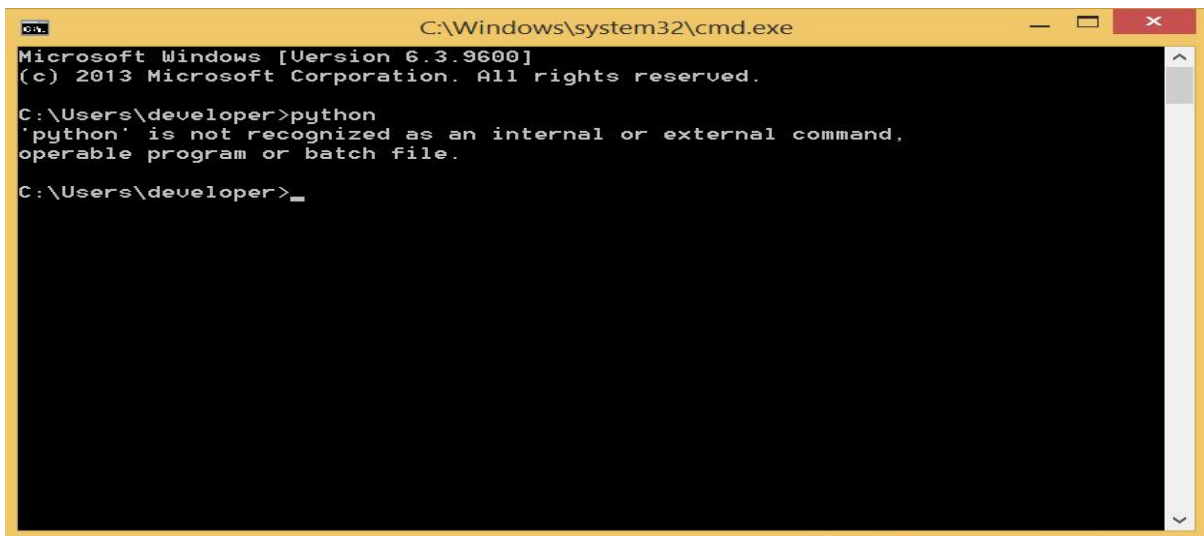
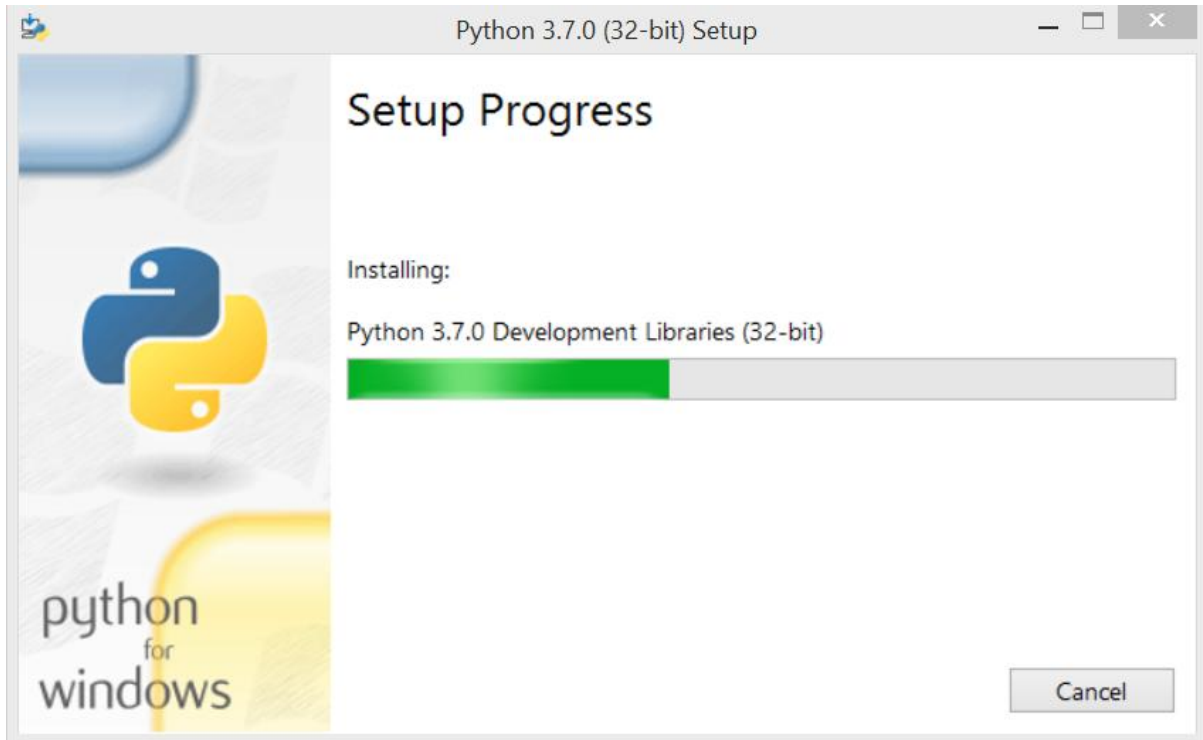
The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.



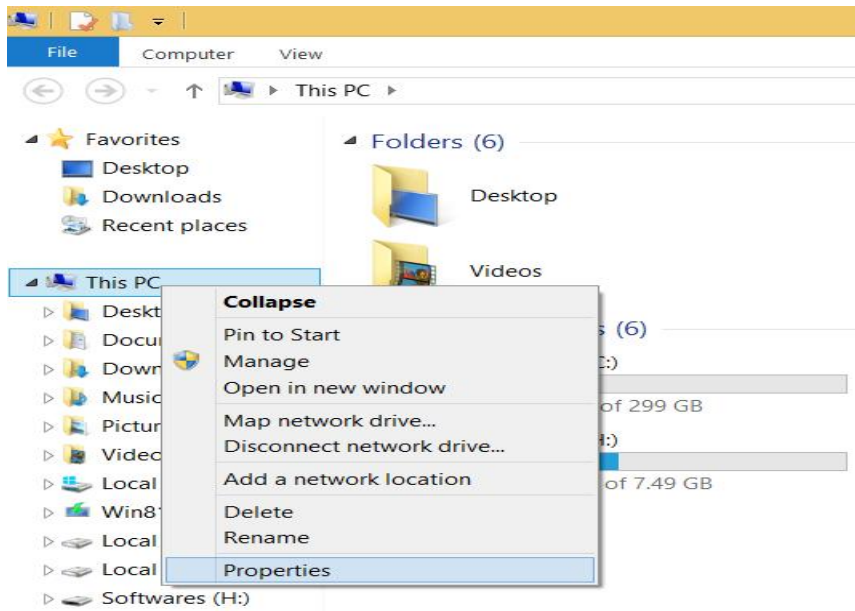
The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, we must notice that the first check-box (install for all users) must be checked.

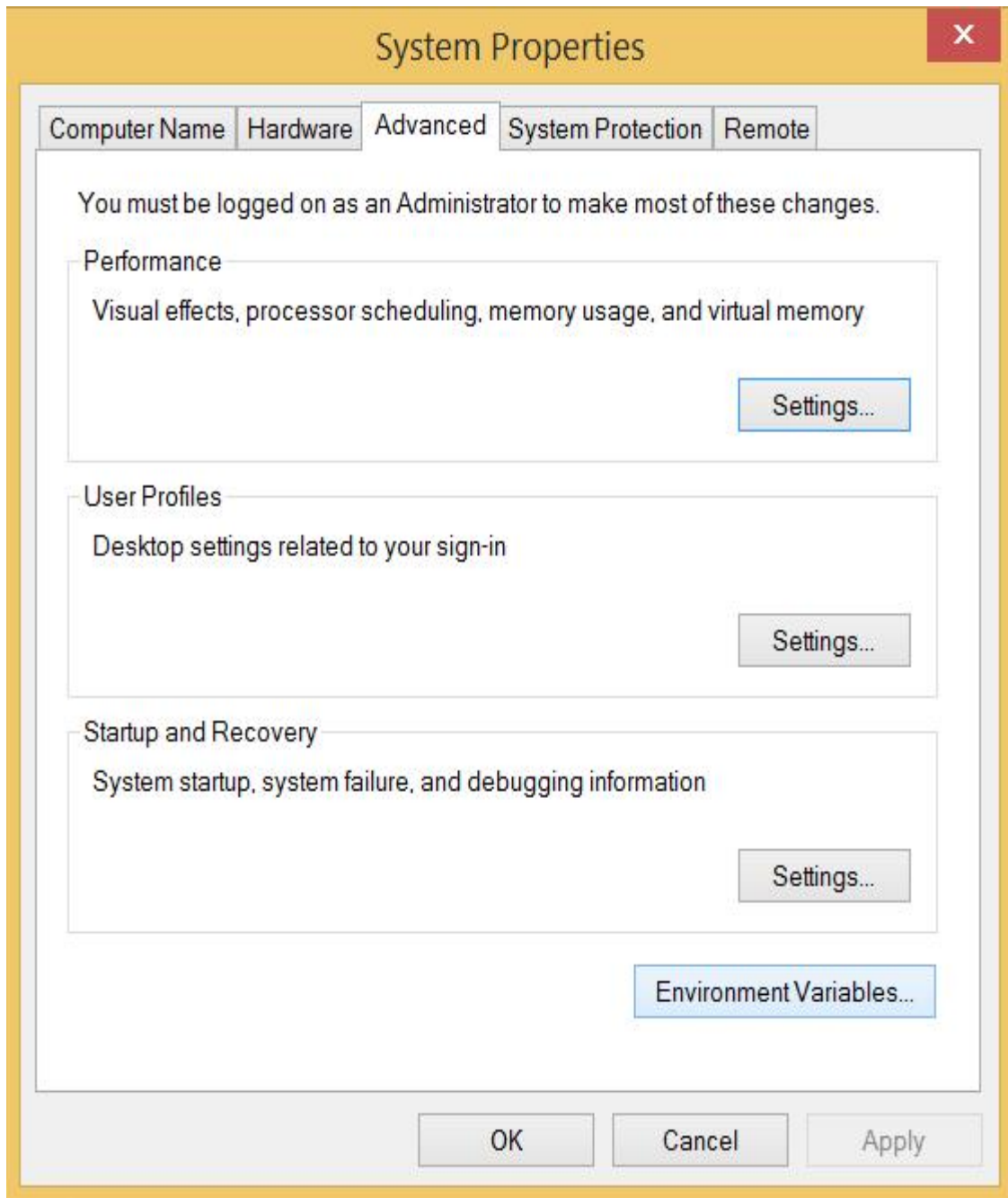


Now, we are ready to install python-3.6.7. Let's install it.

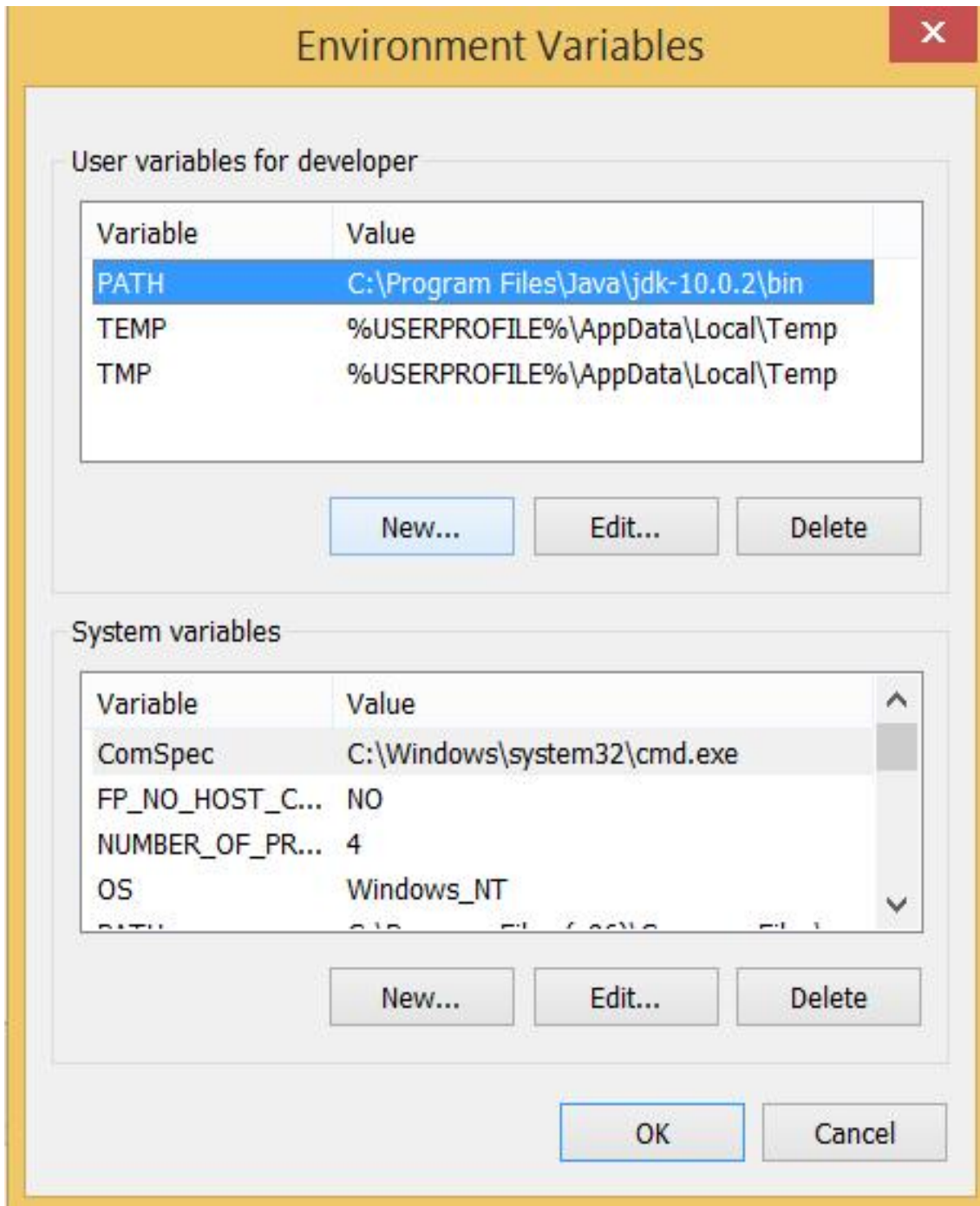


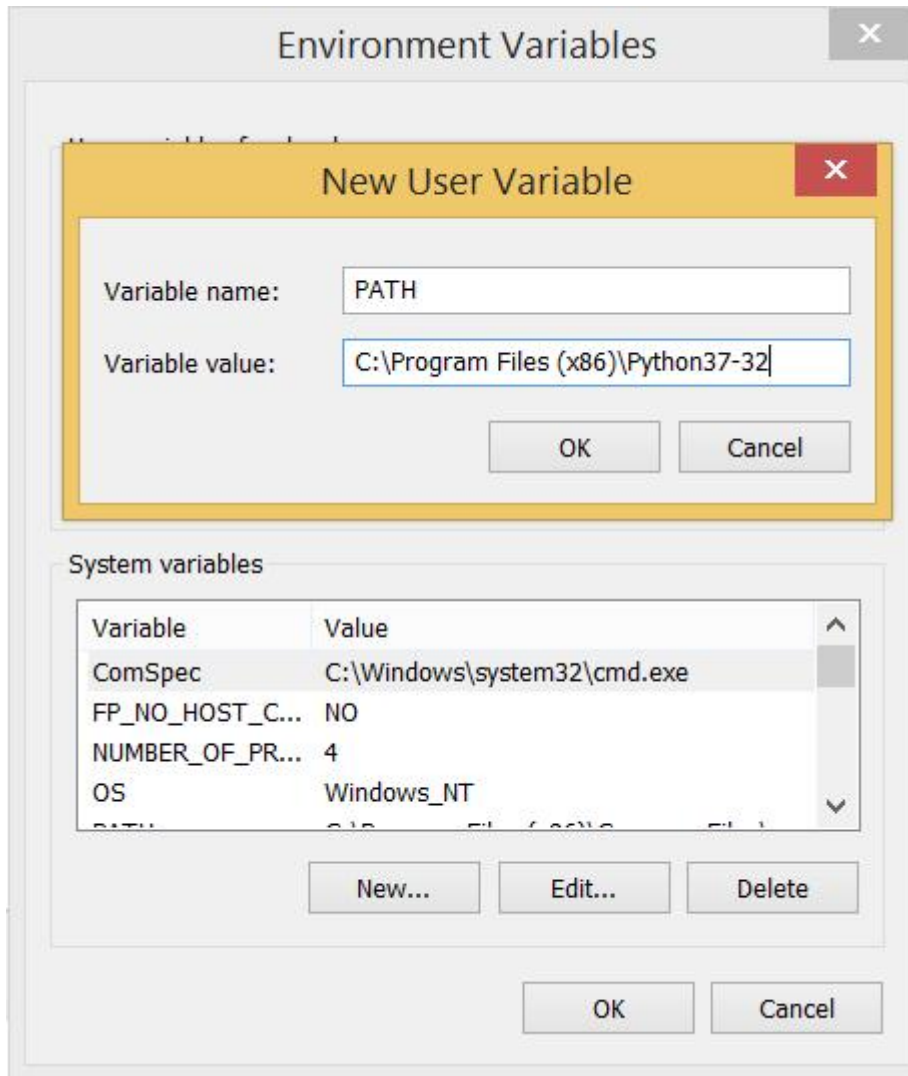
To set the path of python, we need to the right click on "my computer" and go to Properties
→ Advanced → Environment Variables.





Add the new path variable in the user variable section.





[next →](#) [← prev](#)

First Python Program

In this Section, we will discuss the basic syntax of python by using which, we will run a simple program to print hello world on the console.

Python provides us the two ways to run a program:

- Using Interactive interpreter prompt
- Using a script file

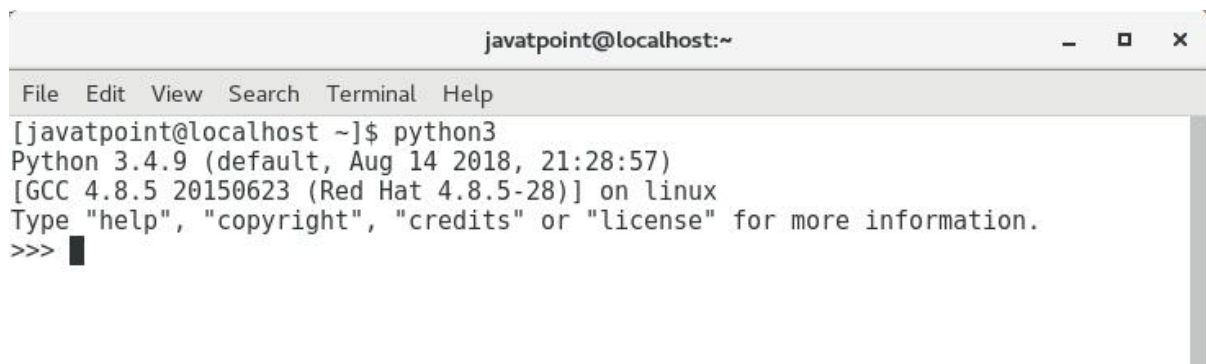
Let's discuss each one of them in detail.

Interactive interpreter prompt

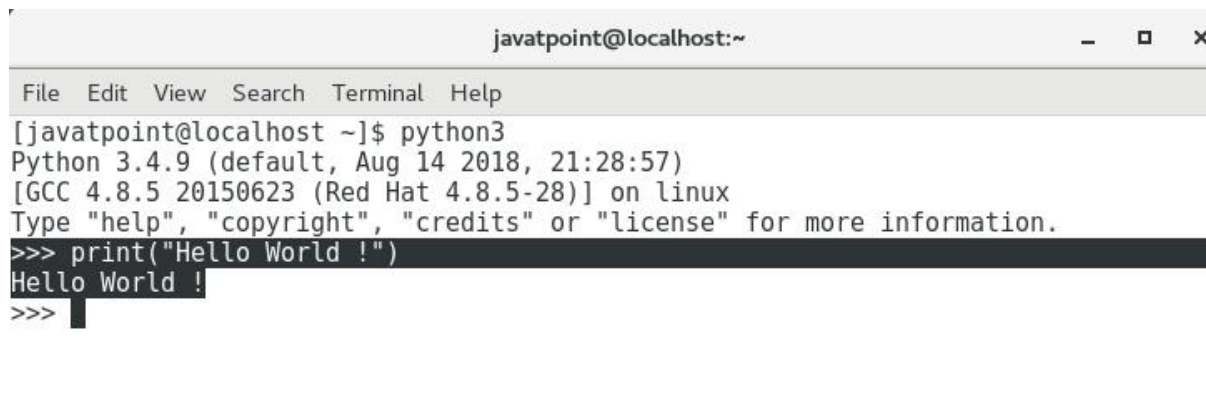
Python provides us the feature to execute the python statement one by one at the interactive prompt. It is preferable in the case where we are concerned about the output of each line of our python program.

To open the interactive mode, open the terminal (or command prompt) and type python (python3 in case if you have python2 and python3 both installed on your system).

It will open the following prompt where we can execute the python statement and check their impact on the console.

A terminal window titled 'javatpoint@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command '[javatpoint@localhost ~]\$ python3' being executed. The output is 'Python 3.4.9 (default, Aug 14 2018, 21:28:57) [GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux' followed by 'Type "help", "copyright", "credits" or "license" for more information.' and the prompt '>>>' with a cursor.

Let's run a python statement to print the traditional hello world on the console. Python3 provides print() function to print some message on the console. We can pass the message as a string into this function. Consider the following image.

A terminal window titled 'javatpoint@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the same initial output as the previous image. Then, the command '>>> print("Hello World !")' is entered and executed. The output 'Hello World !' is printed on the next line, followed by the prompt '>>>' with a cursor.

Here, we get the message **"Hello World !"** printed on the console.

SYSTEM TESTING

INRODUCTION

The purpose of attempting out is to discover errors. Testing is the approach of looking for each capability fault or willing problem in a bit product. It offers a manner to test the capability of additives, subassemblies, assemblies and/or a finished product. It is the tool of workout software program application software program application utility with the motive of making sure that the Software system meets its necessities and purchaser expectancies and does now not fail in an unacceptable way. There are numerous styles of check. Each test type addresses a selected attempting out requirement.

TESTING TYPES

White Box Testing:

White Box Testing (moreover referred to as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software application finding out method wherein the internal shape/ layout/ implementation of the object being tested are idea to the tester. The tester chooses inputs to workout paths thru the code and determines the right outputs. Programming statistics and the implementation understanding is vital. White container locating out is making an attempt out past the character interface and into the nitty-gritty of a tool.

Black Box Testing:

Black Box Testing, furthermore referred to as Behavioral Testing, is a software program software finding out technique wherein the inner shape/ format/ implementation of the object being examined isn't seemed to the tester. These assessments can be practical or non-beneficial, but the truth that commonly

beneficial. This method is known as so due to the reality the software program, within the eyes of the tester, is form of a black box; indoors which one can't see.

Unit Testing:

Unit trying out moreover referred to as Module Testing, focuses verification efforts on the module. The module is examined one after the other and this is done at the programming diploma itself. Unit Test consists of the set of checks finished with the useful resource of an individual programmer earlier than integration of the unit into the tool. Unit take a look at specializes within the smallest unit of software layout-the software program application software program software program software program aspect or module. Using detail diploma layout, important manipulate paths are examined to discover errors inside the boundary of the module. Unit check is white problem oriented and the step can be executed in parallel for a couple of additives.

Functional Testing:

Functional take a look at instances encompass workout the code with ordinary input values for which the predicted outcomes are identified, further to the boundary values

Objective:

The cause is to take unit-examined modules and construct a software program software form that has been dictated via layout.

Performance Testing:

Performance locating out determines the amount of execution time spent in numerous additives of the unit, software application throughput, and response time and device usage of this system unit. It takes region within the direction of all steps inside the finding out way.

Integration Testing:

It is a systematic technique for constructing this tool form at the identical time as on the identical time undertaking assessments to discover errors related to in the interface. It takes the unit tested modules and builds a software program application shape. All the modules are mixed and tested as an entire. Integration of all of the components to shape the complete gadget and an elegant trying out is completed.

Validation Testing:

Validation test succeeds on the equal time because of the truth the software application software capabilities in a way that may be quite predicted thru manner of the client. Software validation is achieved thru a sequence of black hassle locating out which confirms to the necessities. Black problem attempting out is finished at the software program software application interface. The take a look at is designed to discover interface errors, is likewise used to illustrate that software program software utility software talents are operational, enter is nicely common, output are produced and that the integrity of outside records is maintained.

System Testing:

The technique of trying out an included device to affirm that it meets superb requirements. Tests to find out the discrepancies the numerous device and its right reason, current-day-day-day specifications and device documentation.

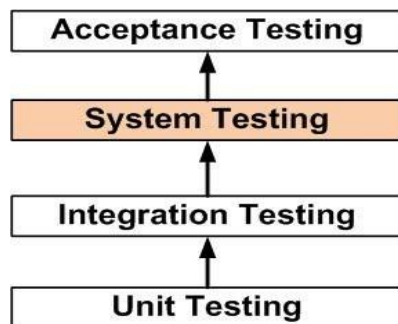


FIG : System Testing

Structure Testing:

It is involved with exercising the inner common experience of a software program and traversing particular execution paths.

Output Testing:

Output of test times in evaluation with the anticipated results created during layout of test instances. Asking the client approximately the layout required with the beneficial useful resource of them exams the output generated or displayed by manner of the usage of the tool beneath consideration. Here, the output format is taken into consideration into have turn out to be, one is on show display and some one in every of a kind one is printed format. The output on the display show is located to be accurate because of the fact the format have grown to be designed in the tool layout segment ordinary with customer dreams. The output comes out because the favored requirements due to the fact the character's hard reproduction.

User beauty Testing:

Final Stage, earlier than coping with over to the consumer it is generally finished through the patron wherein the take a look at times are completed with actual facts. The tool under interest is tested for client recognition and constantly maintaining touch with the viable device person on the time of developing and making changes each time required. It consists of planning and execution of numerous varieties of check and top notch manner to expose that the finished software application software program software application device satisfies the necessities said inside the requirement record.

METHODOLOGY

The most commonly used method of image processing is the convolutional neural network (CNN). CNN differs from a multi-layer perceptron (MLP) in that it has convolutional layers, which are secret layers. A two-level CNN structure underpins the proposed process. The first level, as shown in Fig. 1, is background removal [29], which is used to remove emotions from an image. The traditional CNN network module is used to retrieve the primary expressional vector in this case (EV). The expressional vector (EV) is generated by tracking down significant facial features. Changes in speech are directly linked to EV. The EV is calculated using a simple perceptron unit on a face image with the background removed. We also have a non-convolutional perceptron layer as the final stage in the proposed FERC model. The input data (or image) is received by each convolutional layer, which transforms it before passing it on to the next level. This transformation is a convolution operation. Pattern identification is possible for any of the convolutional layers used. Four filters were used within each convolutional layer. Shapes, edges, textures, and objects, as well as the face, make up the input image fed to the first-part CNN (used for background removal). At the start of the convolutional layer 1, the edge detector, circle detector, and corner detector filters are used. The second-part CNN filter detects facial features such as eyes, ears, mouth, nose, and cheeks after the face has been identified. Layers with 3 3 kernel matrices, such as [0.25, 0.17, 0.9; 0.89, 0.36, 0.63; 0.7, 0.24, 0.82], make up the second-part CNN. These numbers are initially chosen from a range of 0 to 1. Based on the ground truth we had in the supervisory training dataset, these numbers were optimised for EV detection. To maximise filter values, we used minimal error decoding. After supervisory learning has fine-tuned the filter, it is added to the background-removed face (i.e., the output image of the first-part CNN) for detection of different facial components.

Frame extraction from input video.

FERC can work with both image and video input. When the FERC receives video as an input, the difference between respective frames is computed. When the intra-frame difference is zero, the frames are maximally stable. After that, a Canny edge detector was applied to all of these stable frames, and the aggregated number of white pixels was measured. Following a comparison

of the aggregated sums for all stable frames, the frame with the highest aggregated sum is chosen because it includes the most information in terms of edges (more edges more details).

The input to FERC is then chosen from this frame. The decision to use this picture was based on the fact that fuzzy images have few or no edges.

SOURCE CODE

```
# Libraries

from django.shortcuts import render

from django.http import HttpResponseRedirect


import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

import itertools

from sklearn.naive_bayes import MultinomialNB

from sklearn import metrics

from sklearn.linear_model import PassiveAggressiveClassifier

import os


import seaborn as sns

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC
```



```

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix

# Input data files are available in the "../input/" directory.

# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the
input directory

```

```
##### Home #####
```

```

def home(request):

import numpy as np

import argparse

import cv2

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Flatten

from tensorflow.keras.layers import Conv2D

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.preprocessing.image import ImageDataGenerator

```

```
import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

# Define data generators

train_dir = 'E:/DJANGO/face emotion recognition/fakenewsdetect/fakenews/data/train'

val_dir = 'E:/DJANGO/face emotion recognition/fakenewsdetect/fakenews/data/test'


num_train = 28709

num_val = 7178

batch_size = 64

num_epoch = 50


train_datagen = ImageDataGenerator(rescale=1./255)

val_datagen = ImageDataGenerator(rescale=1./255)


train_generator = train_datagen.flow_from_directory(

train_dir,

target_size=(48,48),

batch_size=batch_size,

color_mode="grayscale",

class_mode='categorical')
```

```
validation_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')

# Create the model

model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

```

model.add(Flatten())

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(7, activation='softmax'))

model.load_weights('E:/DJANGO/face emotion recognition/fakenewsdetect/fakenews/model.h5')


# prevents openCL usage and unnecessary logging messages

cv2ocl.setUseOpenCL(False)


# dictionary which assigns each label an emotion (alphabetical order)
emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6:
"Surprised"}


# start the webcam feed

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

#cap = cv2.VideoCapture('E:/DJANGO/face emotion
recognition/fakenewsdetect/fakenews/acharya.MP4')

while True:

# Find haar cascade to draw bounding box around face

ret, frame = cap.read()

if not ret:

break

```

```

facecasc = cv2.CascadeClassifier('E:/DJANGO/face emotion
recognition/fakenewsdetect/fakenews/haar cascade files/haarcascade_frontalface_default.xml')

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = facecasc.detectMultiScale(gray,scaleFactor=1.3, minNeighbors=5)

for (x, y, w, h) in faces:

    cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (0, 255, 0), 2)

    roi_gray = gray[y:y + h, x:x + w]

    cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)

    prediction = model.predict(cropped_img)

    maxindex = int(np.argmax(prediction))

    cv2.putText(frame, emotion_dict[maxindex], (x+20, y-60), cv2.FONT_HERSHEY_SIMPLEX,
    1, (255, 255, 255), 2, cv2.LINE_AA)

cv2.imshow('Video', cv2.resize(frame,(1600,960),interpolation = cv2.INTER_CUBIC))

if cv2.waitKey(1) & 0xFF == ord('q'):

    break

cap.release()

cv2.destroyAllWindows()

return render(request,'index1.html')

def vedio(request):

```

```
import numpy as np

import argparse

import cv2

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Flatten

from tensorflow.keras.layers import Conv2D

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

# Define data generators

train_dir = 'E:/DJANGO/face emotion recognition/fakenewsdetect/fakenews/data/train'

val_dir = 'E:/DJANGO/face emotion recognition/fakenewsdetect/fakenews/data/test'


num_train = 28709

num_val = 7178

batch_size = 64

num_epoch = 50


train_datagen = ImageDataGenerator(rescale=1./255)
```

```
val_datagen = ImageDataGenerator(rescale=1./255)
```

```
train_generator = train_datagen.flow_from_directory(  
    train_dir,  
    target_size=(48,48),  
    batch_size=batch_size,  
    color_mode="grayscale",  
    class_mode='categorical')
```

```
validation_generator = val_datagen.flow_from_directory(  
    val_dir,  
    target_size=(48,48),  
    batch_size=batch_size,  
    color_mode="grayscale",  
    class_mode='categorical')
```

```
# Create the model
```

```
model = Sequential()
```

```
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
```

```
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
```

```

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))


model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))


model.add(Flatten())

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(7, activation='softmax'))

model.load_weights('E:/DJANGO/face emotion recognition/fakenewsdetect/fakenews/model.h5')


# prevents openCL usage and unnecessary logging messages

cv2.ocl.setUseOpenCL(False)


# dictionary which assigns each label an emotion (alphabetical order)

emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6:
"Surprised"}

```



```

# start the webcam feed

#cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

cap = cv2.VideoCapture('E:/DJANGO/face_recognition/face_recognition/fakenewsdetect/fakenews/acharya.MP4')

while True:

    # Find haar cascade to draw bounding box around face

    ret, frame = cap.read()

    if not ret:

        break

    facecasc = cv2.CascadeClassifier('E:/DJANGO/face_recognition/face_recognition/fakenewsdetect/fakenews/haar cascade files/haarcascade_frontalface_default.xml')

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = facecasc.detectMultiScale(gray,scaleFactor=1.3, minNeighbors=5)

    for (x, y, w, h) in faces:

        cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (0, 255, 0), 2)

        roi_gray = gray[y:y + h, x:x + w]

        cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)

        prediction = model.predict(cropped_img)

        maxindex = int(np.argmax(prediction))

```

```
cv2.putText(frame, emotion_dict[maxindex], (x+20, y-60), cv2.FONT_HERSHEY_SIMPLEX,
1, (255, 255, 255), 2, cv2.LINE_AA)
```

```
cv2.imshow('Video', cv2.resize(frame,(1600,960),interpolation = cv2.INTER_CUBIC))
```

```
if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
break
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```

```
return render(request,'index1.html')
```

```
def image(request):
```

```
return render(request,'index1.html')
```

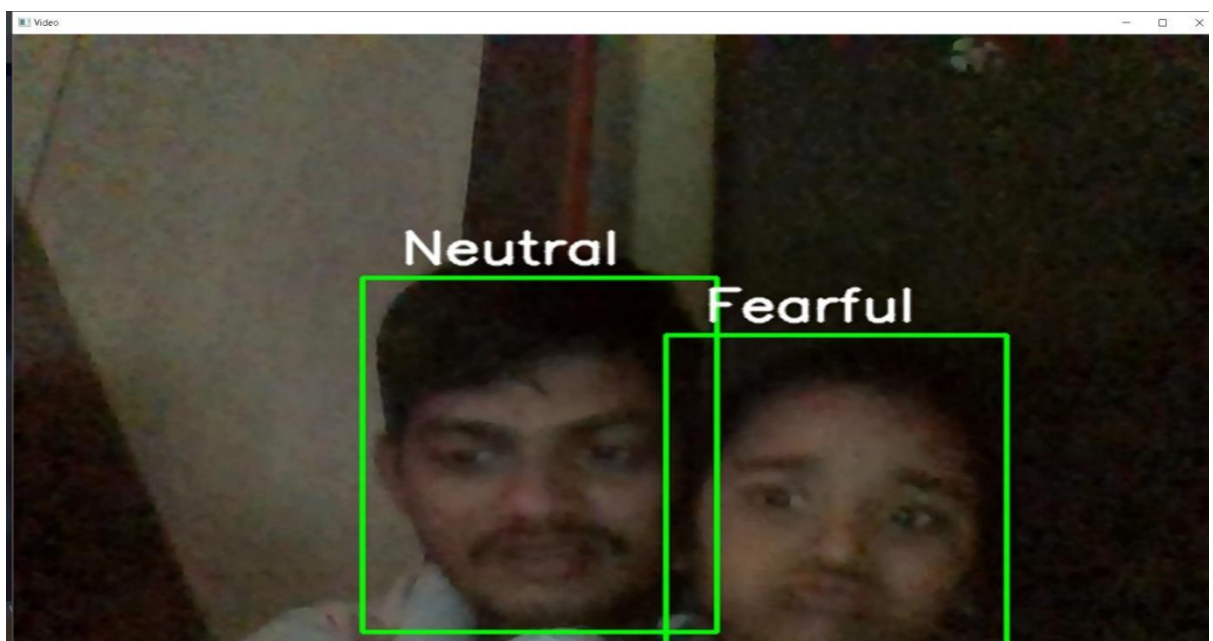
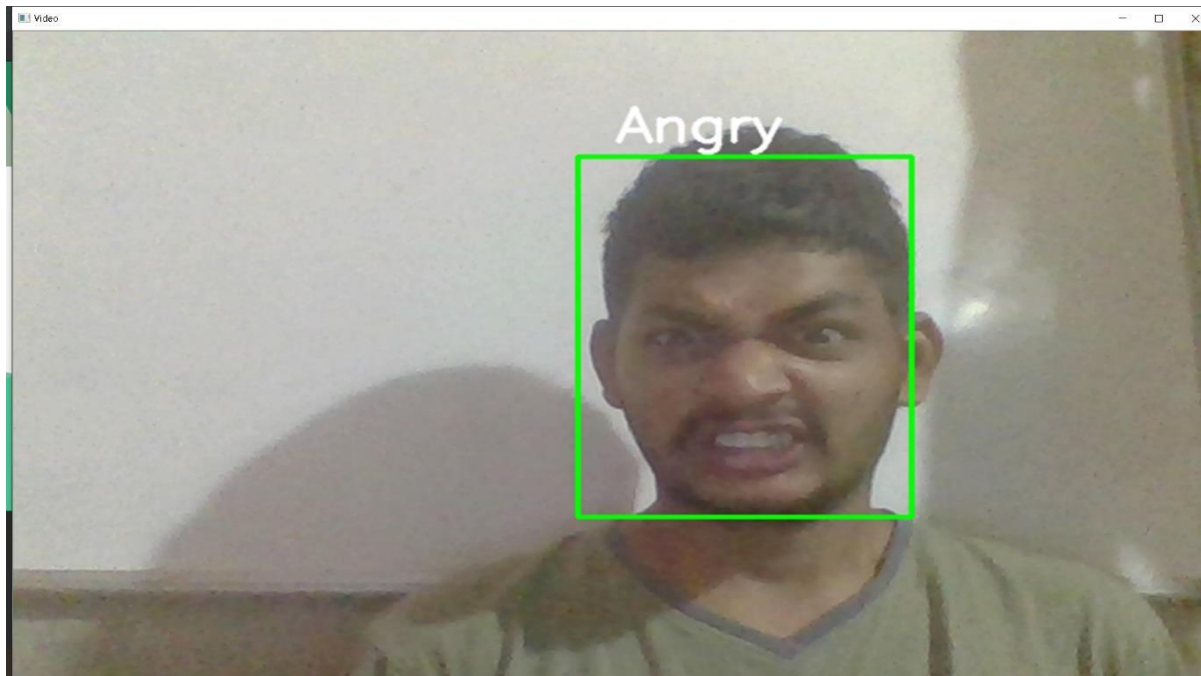
```
def accuracy(request):
```

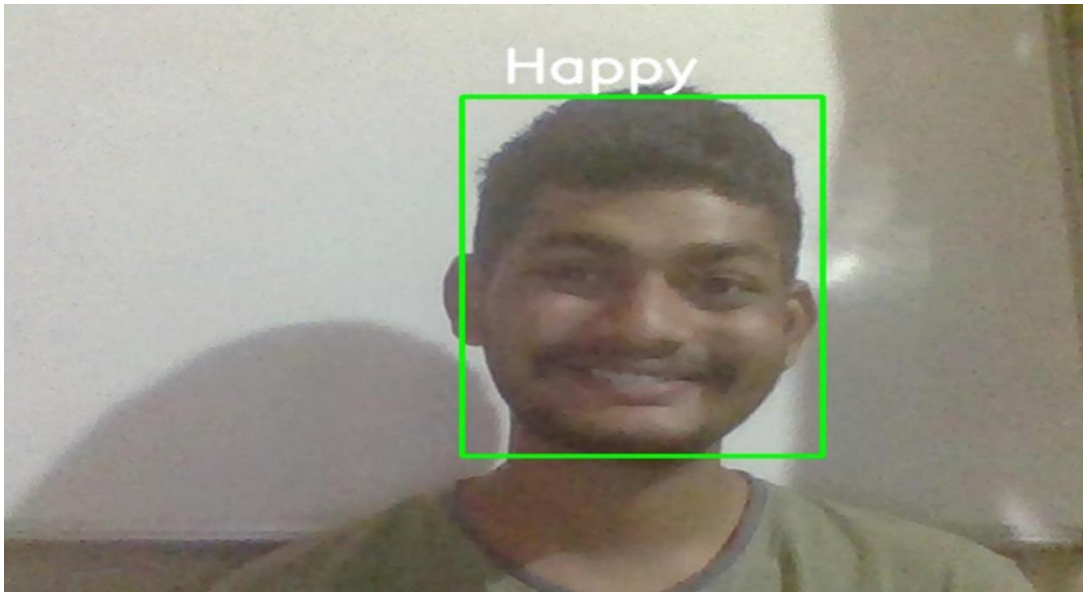
```
return render(request,'index1.html')
```

RESULTS



Emotions Detected Using Live Webcam:





CONCLUSION

We used various post-processing and visualisation techniques to evaluate the output of various CNNs for a facial expression recognition query. The findings showed that deep CNNs are efficient. Human Facial expression recognition can be improved by studying facial characteristics. Furthermore, the hybrid feature sets had no effect on model accuracy, implying that convolutional networks would learn key facial features intrinsically using only raw pixel data.

REFERENCES

1. Cowie, R.; Douglas-Cowie, E.; Tsapatsoulis, N.; Votsis, G.; Kollias, S.; Fellenz, W.; Taylor, J.G. Emotion recognition in human-computer interaction. *IEEE Signal Process. Mag.* **2001**, 18, 32–80.
2. Fragopanagos, N.; Taylor, J.G. Emotion recognition in human–computer interaction. *Neural Netw.* **2005**, 18, 389–405
3. Busso, C.; Deng, Z.; Yildirim, S.; Bulut, M.; Lee, C.M.; Kazemzadeh, A.; Lee, S.; Neumann, U.; Narayanan, S. Analysis of emotion recognition using facial expressions, speech and multimodal information. In *Proceedings of the 6th International Conference on Multimodal Interfaces*, State College, PA, USA, 14–15 October 2004; ACM: New York, NY, USA, 2004; pp. 205–211.
4. El Ayadi, M.; Kamel, M.S.; Karray, F. Survey on speech emotion recognition: Features, classification schemes, and databases. *Pattern Recognit.* **2011**, 44, 572–587.
5. Lin, Y.P.; Wang, C.H.; Jung, T.P.; Wu, T.L.; Jeng, S.K.; Duann, J.R.; Chen, J.H. EEG-based emotion recognition in music listening. *IEEE Trans. Biomed. Eng.* **2010**, 57, 1798–1806.
6. Harms, M.B.; Martin, A.; Wallace, G.L. Facial emotion recognition in autism spectrum disorders: A review of behavioral and neuroimaging studies. *Neuropsychol. Rev.* **2010**, 20, 290–322.
7. Ali, M.; Mosa, A.H.; Al Machot, F.; Kyamakya, K. Emotion recognition involving physiological and speech signals: A comprehensive review. In *Recent Advances in Nonlinear Dynamics and Synchronization*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 287–302.
8. Wu, C.H.; Chuang, Z.J.; Lin, Y.C. Emotion recognition from text using semantic labels and separable mixture models. *ACM Trans. Asian Lang. Inf. Process. TALIP* **2006**, 5, 165–183.
9. Jerritta, S.; Murugappan, M.; Nagarajan, R.; Wan, K. Physiological signals based human emotion recognition: A review. In *Proceedings of the 2011 IEEE 7th International Colloquium on Signal Processing and its Applications*, Penang, Malaysia, 4–6 March 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 410–415.
10. Zheng, W.L.; Zhu, J.Y.; Lu, B.L. Identifying stable patterns over time for emotion recognition from EEG. *IEEE Trans. Affect. Comput.* **2017**, 10, 417–729.

11. Wioleta, S. Using physiological signals for emotion recognition. In Proceedings of the 2013 6th International Conference on Human System Interactions (HSI), Sopot, Poland, 6–8 June 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 556–561.
12. Yoo, G.; Seo, S.; Hong, S.; Kim, H. Emotion extraction based on multi bio-signal using back-propagation neural network. *Multimed. Tools Appl.* **2018**, *77*, 4925–4937.
13. Soleymani, M.; Pantic, M.; Pun, T. Multimodal emotion recognition in response to videos. *IEEE Trans. Affect. Comput.* **2012**, *3*, 211–223.
14. Sim, H.; Lee, W.H.; Kim, J.Y. A Study on Emotion Classification utilizing Bio-Signal (PPG, GSR, RESP). *Adv. Sci. Technol. Lett.* **2015**, *87*, 73–77.
15. Domínguez-Jiménez, J.; Campo-Landines, K.; Martínez-Santos, J.; Delahoz, E.; Contreras-Ortiz, S. A machine learning model for emotion recognition from physiological signals. *Biomed. Signal Process. Control* **2020**.
16. Pinto, J.; Fred, A.; da Silva, H.P. Biosignal-Based Multimodal Emotion Recognition in a Valence-Arousal Affective Framework Applied to Immersive Video Visualization. In Proceedings of the 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany, 23–27 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 3577–3583.
17. Zheng, W.L.; Lu, B.L. Investigating critical frequency bands and channels for EEG-based emotion recognition with deep neural networks. *IEEE Trans. Auton. Ment. Dev.* **2015**, *7*, 162–175.