# A MAJOR PROJECT REPORT

**ON**

# IMAGE COLOURIZATION

**Submitted to**

**Sri Indu College of Engineering & Technology, Hyderabad**

### In partial fulfillment of the requirements for the award of degree of BACHELOR OF TECHNOLOGY

**In**

**“ARTIFICIAL INTELLIGENCE & MACHINE LEARNING”**

**Submitted by**

### A.RISHIKESH [20D41A6604]

**B.DIVYA [20D41A6609]**

**G.CHAKRAPANI [20D41A6620]**

**V.LIKITHA [20D41A6660]**

# Under the esteemed guidance of

### Dr. ADELINE JOHNSON

**(Assistant Professor)**



DEPARTMENT OF

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

# SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution under UGC, Accredited by NBA&NAAC, Affiliated to JNTUH)

Sheriguda, Ibrahimpatnam-501510 (2022-2023)

# SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

**(An Autonomous Institution under UGC, Accredited by NBA&NAAC, Affiliated to JNTUH)**

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



**CERTIFICATE**

Certified that the Major project entitled **“IMAGE COLOURIZATION”** is a bonafide work carried out by **A.RISHIKESH (20D41A6604), B.DIVYA (20D41A6609), G.CHAKRAPANI (20D41A6620), V.LIKITHA (20D41A6660)** in partial fulfillment for the award of Bachelor of Technology in **Artificial Intelligence & Machine Learning** of SICET, Hyderabad for the academic year **2023-2024.**The project has been approved as it satisfies academic requirements in respect of the work prescribed for **IV Year, I- Semester of B. Tech** course.

### INTERNAL GUIDE HEAD OF THE DEPARTMENT

**Dr. Adeline Johnson (G. Uma Maheshwari)**

### DEAN EXTERNAL EXAMINER

**(Dr. K. S. Sadasiva Rao)**



**CONSCIENCE TECHNOLOGIES**

**A Right Platform for all Engineers...**

Certificate

This is to certify that

### A. RISHIKESH [20D41A6604]

**B. DIVYA [20D41A6609]**

### G. CHAKRAPANI [20D41A6620]

### V. LIKITHA [20D41A6660]

Students of final year B.Tech, **Artificial Intelligence & Machine Learning, SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY, HYDERABAD *(****affiliated to J.N.T.University, Hyderabad****),*** have successfully completed their project titled **“IMAGE COLOURIZATION”** in PYTHON DIVISION, CONSCIENCE TECHNOLOGIES (CT), Hyderabad.

#### For CONSCIENCE TECHNOLOGIES

**Project Manager**

C:\Users\user\Desktop\VENKI\Untitled ...2.png

#### K.V. REDDY

#13-18-163/8, 3rd Floor, GPAR Complex, Metro Pillar No.A1565,ROAD NO.5, Kamalanagar,

Beside HDFC bank Line, Chaitanyapuri, Dilsukhnagar, Hyderabad-500060 Email: [consciencetechnologies@gmail.com](mailto:consciencetechnologies@gmail.com) [www.consciencetechnologies.com](http://www.consciencetechnologies.com/) PH:8125484777

# ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We are thankful to Principal **Dr. G. SURESH** garu and Dean CSE & Allied Branches **DR. K. S. SADASIVA RAO** garu for giving us the permission to carry out this project and for providing necessary infrastructure and labs.

We are highly indebted to **G. Uma Maheshwari**, **Professor& Head**, Department of Artificial Intelligence & Machine Learning, for providing valuable guidance at every stage of this project.

We are grateful to our internal project guide **Dr. Adeline Johnson, Assistant Professor** for her constant motivation and guidance given by her during the execution of this project work.

We would like to thank the Teaching & Non-Teaching staff of Department of Artificial Intelligence & Machine Learning for sharing their knowledge with us, last but not least we express our sincere thanks to everyone who helped directly or indirectly for the completion of this project.

### A. RISHIKESH [20D41A6604]

**B. DIVYA [20D41A6609]**

### G. CHAKRAPANI [20D41A6620]

### V. LIKITHA [20D41A6660]

**CONTENTS**

**S NO CHAPTER NAME PAGE NO.**

[Abstract i](#_TOC_250012)

1. [INTRODUCTION](#_TOC_250011)
   1. INTRODUCTION TO PROJECT 01
2. LITURATURE SURVEY 03
3. FEASIBIITY STUDY… 06
4. [SOFTWARE ENVIRONMENT](#_TOC_250010)
   1. PYHTON 08
   2. VIRTUAL ENVIRONMENT AND PACKAGES 22
   3. [ARTIFICIAL INTELLIGENCE 29](#_TOC_250009)
   4. [MACHINE LEARNING 38](#_TOC_250008)
   5. DEEP LEARNING 48
   6. [DJANGO 50](#_TOC_250007)
5. [REQUIREMENTS AND TOOLS 55](#_TOC_250006)
6. [SYSTEM ANALYSIS](#_TOC_250005)
   1. [EXISTING SYSTEM AND DISADVANTAGES 56](#_TOC_250004)
   2. [PROPOSED SYSTEM AND ADVANTAGES 57](#_TOC_250003)
7. RESULT AND ANALYSIS 58
8. OUTPUT SCREENS… 59
9. [CONCLUSION 62](#_TOC_250002)
10. [REFERENCES 63](#_TOC_250001)
11. [FUTURE SCOPE 65](#_TOC_250000)

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig No** | **Name** | **Page No** |
| Fig. 1 | System Architecture diagram | 01 |
| Fig. 2 | Python logo | 08 |
| Fig. 3 | Artificial Intelligence | 29 |
| Fig. 4 | Machine learning | 32 |
| Fig. 5 | Decision Tree | 38 |
| Fig. 6 | Deep learning | 39 |
| Fig. 7 | Home page | 47 |
| Fig. 8 | Analyzing the audio file | 48 |
| Fig. 9 | Menu of Analyzing the audio | 48 |
| Fig. 10 | Prediction of results | 49 |

# ABSTRACT

The detection of the emotions elicited by the speaker while talking. As an example, speech produced in a state of fear, anger, or joy becomes loud and fast, with a higher and wider range in pitch, whereas emotions such as sadness or tiredness generate slow and low- pitched speech. Detection of human emotions through voice- pattern and speech-pattern analysis has many applications such as better assisting human-machine interactions. In particular, we are presenting a classification models of emotions elicited by speeches based on deep neural networks (CNNs), Support Vector Machine (SVM), Multi-layer Perceptron (MLP)Classification based on acoustic features such as Mel Frequency Cepstral Coefficient (MFCC).The models have been trained to classify seven different emotions (neutral, calm, happy, sad, angry, fearful, disgust, surprise). Our evaluation shows that the proposed approach yields accuracies of 86%, 84% and 82% using CNN, MLP and SVM respectively, for 7 emotions using Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset and Toronto Emotional Speech Set (TESS) Datasets.

i

# INTRODUCTION

Human emotions are very difficult to comprehend from a quantitative perspective. Facial expressions are one of the best ways of guessing the emotional state of a person. Speech is another modality that can be used. Speech is a complex signal which contains information about the message, speaker, language and emotions. There are various kinds of emotions which can be articulated using speech. Emotional speech recognition is a system which basically identifies the emotional state of human being from his or her voice; speech is very misleading even for humans to judge the emotion of the speaker.

A major motivation comes from the desire to improve the naturalness and efficiency of human-machine interaction. The reference paper that was chosen has been able to successfully recognize only 4 emotions. The work presented here has classified 7 emotions with an overall good recognition rate. In general, the systems for speech analysis uses various techniques for the extraction of characteristics from the raw signal. The next step in this strategy is to discover the features which discriminate the speech data (to the training labels) and to discard the non-discriminating features. The Emotional profiles (EP) are constructed using SVM with Radial Basis Function (RBF).

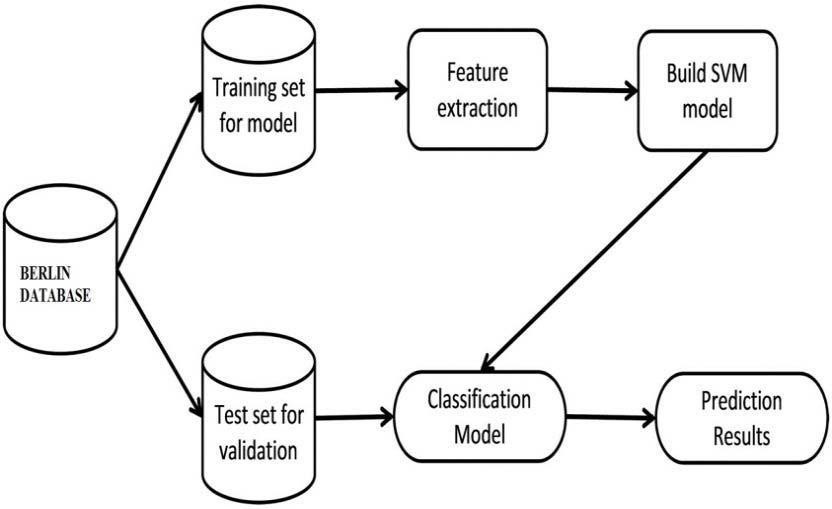


Fig. 1. Block Diagram of the proposed Speech Recognition System

The profiles are created by weighting each of the n-outputs by the distance between the individual point and the hyperplane boundary. The final emotion is selected by classifying the generated profile. This is done by one vs one comparing of each emotion to the existing profile of the emotion.

Fig.1 comprehensively explains the methodology followed in this paper. Emotion recognition is done using two modules. The first module is the feature extraction module and the second is the classifier module. In the feature extraction module, we have used a feature set comprising pitch, prosody and voice quality features. Several classifiers exist for the task of emotion recognition. The different classifiers are SVM, MLP (Multi-Layer Perceptron), HMM(Hidden Markov Model), GMM(Gaussian Mixture Model), ANN(Artificial Neural Networks) etc. The SVM classifier yields good results even from small test samples and hence it is widely used for speech emotional recognition. The SVM classifier is therefore used for the proposed work. Because of the Structural Risk Minimization, SVM classifiers usually have better performance than others.

# LITERATURE SURVEY

An enhanced speech emotion recognition is carried out over six basic emotions of angry, happy, sad, neutral, surprise and fear. Here, as an advanced research methodology, the pre-processing was carried out using PRNN and KNN algorithms while the feature extraction was made using a cascaded structure comprising of MFCC and GLCM In this research study, an enhanced human speech emotion recognition system using a hybrid of PRNN and KNN algorithms is designed. The six basic emotions like neutral, anger, happiness, sadness, surprise and fear over the speech emotions are classified and studied for their accuracy with other previously developed systems.

The database for this study is taken as the emotional speech samples of numbers. A cascaded system of Mel Frequency Cepstral Coefficient (MFCC) and Gray Level Co- occurrence Matrix (GLCM) was used for feature extraction process along with a Wiener filter for filtering the noise in speech. Also, a hybrid of Pattern Recognition Neural Network (PRNN) and KNearest Neighbor (KNN) is used for prediction accuracy of outcomes.

The outcomes are compared with previously developed recognition systems and better efficiency is observed. The obtained results were compared for their accuracy, precision rate and f-Measure with standard algorithms like GMM and HMM and were recognized as a better output than the standard algorithms. H. Zheng and Y. Yang. In order to improve the characterization ability of speech signal and recognition accuracy of speech emotion recognition, a speech emotion recognition model based on improved Deep Belief Network (DBN) is proposed.

The method is to replace the traditional DBN activation function with a Rectified Linear Unit (Relu). And the reconstruction error is used to determine the depth of the DBN network. The short time energy, short-time zero crossing rate, the fundamental frequency, formants and24 dimensional MFCC parameters of emotional speech signal is extracted as the basic features. Using these basic features as input to the DBN, automatic recognition of the 6 emotions, anger, fear, joy, calmness, sadness and surprise can be achieved. Compared with the traditional DBN model and the BP model, a better recognition result is achieved by using the improved DBN discussed

in this paper, and the recognition rate can reach 84.94%. This paper introduces a speech emotion recognition system, and extract the short-term energy, short-time zero crossing rate, gene frequency, first resonance and MFCC. The results show that the eigenvalues selected in this paper are reasonable, and the method of extracting is effective and feasible. The Relu function replaces the sigmoid function of the traditional DBN, and the reconstruction error of the RBM training is improved,

thereby improving the recognition efficiency. Reconstruction error is used to determine the number of hidden layers in DBN model, which makes the model more stable.

Michael Neumann, Ngoc Thang Vu. In this paper, they have shown that incorporating representations generated by an autoencoder that was trained on a large dataset, leads to consistent improvements in recognition accuracy of the presented SER model. Further they presented t-SNE visualizations that reveal the discriminative strength of those representations with regard to low and high arousal. Future work includes experimentation with different variants of autoencoders and investigation in generative adversarial networks for representation learning.

In this paper they present findings on how representation learning on large unlabeled speech corpora can be beneficially utilized for speech emotion recognition (SER). Prior work on representation learning for SER mostly focused on the relatively small emotional speech datasets without making use of additional unlabeled speech data.

They show that integrating representations learnt by an unsupervised autoencoder into a CNN-based emotion classifier improves the recognition accuracy. To gain insights about what those models learn, we analyze visualizations of the different representations using t- distributed neighbor embeddings (t-SNE). We evaluate our approach on IEMOCAP and MSP- IMPROV by means of within and cross-corpus testing.

Valery A. Petrushin. The paper describes two experimental studies on vocal emotion expression and recognition. The first study

deals with a corpus of 700 short utterances expressing five emotions: happiness, anger, sadness, fear, and normal (unemotional) state, which were portrayed by thirty non- professional actors.

After evaluation a part of this corpus was used for extracting features and training backpropagation neural network models. Some statistics of the pitch, the first and second formants, energy and the speaking rate were selected as relevant features using feature selection techniques.

Several neural network recognizers and ensembles of recognizers were created. The recognizers have demonstrated the following accuracy: normal state - 60-75%, happiness

– 60- 70%, anger – 70-80%, sadness – 70-85%, and fear – 35-55%. The total average accuracy is about 70%. The second study uses a corpus of 56 telephone messages of varying length (from 15 to 90 seconds) expressing mostly normal and angry emotions that were recorded by eighteen non-professional actors. These utterances were used for creating recognizers using the methodology developed in the first study.

The recognizers are able to distinguish between two states: “agitation” which includes anger, happiness and fear, and “calm” which includes normal state and sadness with the average accuracy 77%. An ensemble of such recognizers was used as a part of a decision support system for prioritizing voice messages and assigning a proper agent to response the message. In this paper they explored how well people and computers recognize emotions in speech. Several conclusions can be drawn from the above results. First, decoding of emotions in speech is complex process that is influenced by cultural, social, and intellectual characteristics of subjects. People are not perfect in decoding even such manifest emotions as anger and happiness. Second, anger is the most recognizable and easier to portray emotion. It is also the most important emotion for business.

# 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.

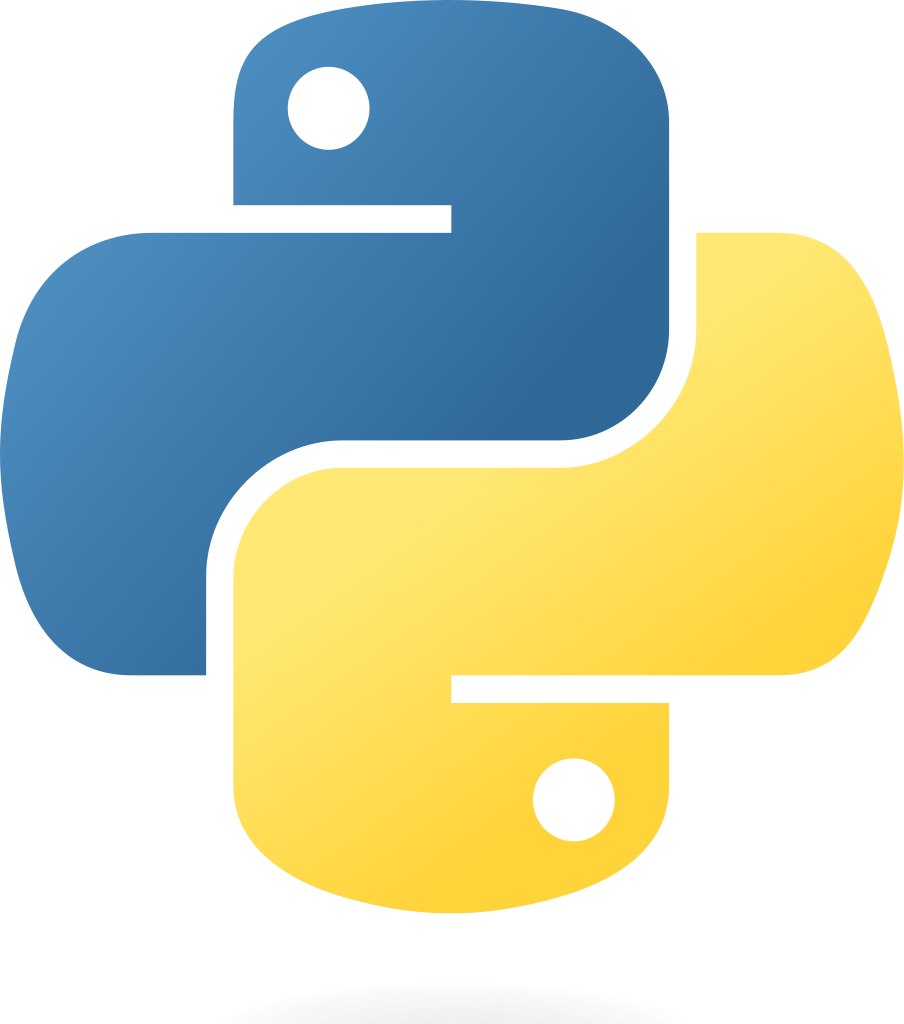
## SOCIAL FEASIBILITY

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.

# SOFTWARE ENVIRONMENT

### Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales.



Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object- oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

### What is Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

### It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

### What can Python do

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

### Why Python?

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi,etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.



* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or afunctional way.

### Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, PyCharm, NetBeans or Eclipse which are particularly useful when managing larger collections of Python files.

#### Advantages of Python:-

1. **Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

#### Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

#### Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

#### Improved Productivity

The language’s simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

#### IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

#### Simple and Easy

When working with Java, you may have to create a class to print **‘Hello World’**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

#### Advantages of Python over Other Languages

* 1. **Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

#### Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

#### The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

* 1. **Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

### Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

### Modules Used in Project

#### TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

#### NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional

container of generic data. Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

#### Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

#### Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

#### Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

#### Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high- level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

#### How to Install Python on Windows and Mac:

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but

this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

**Note:** The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheat sheet here. The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

#### Download the Correct version into the system

**Step 1:** Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: **https://**[**www.python.org.**](http://www.python.org/)

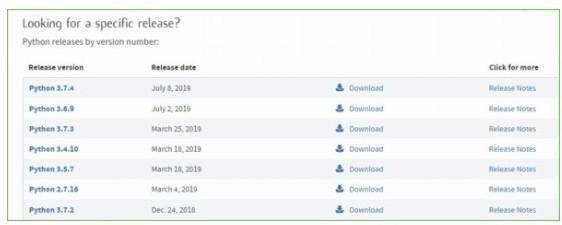


Now, check for the latest and the correct version for your operating system.

**Step 2:** Click on the Download Tab.

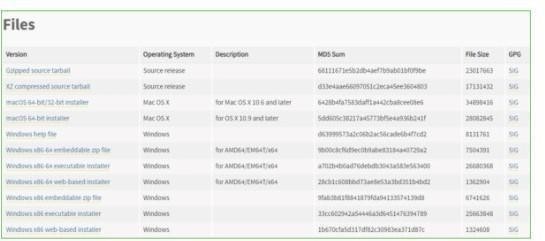


**Step 3:** You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



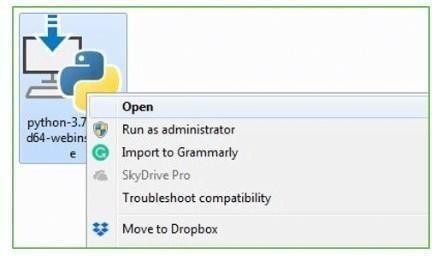
**Step 4:** Scroll down the page until you find the Files option.

**Step 5:** Here you see a different version of python along with the operating system.



#### Installation of Python

**Step 1:** Go to Download and Open the downloaded python version to carry out the installation process.



**Step 2:** Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.



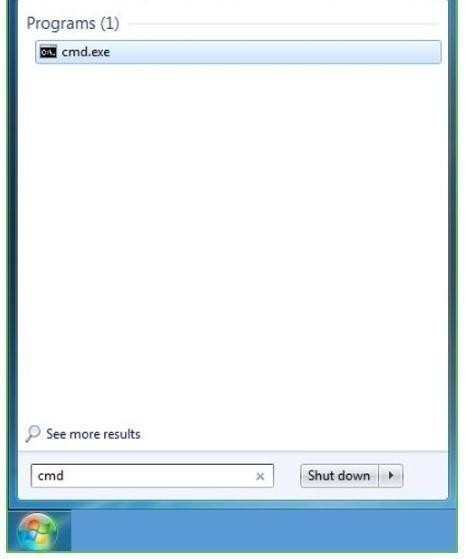
**Step 3:** Click on Install NOW After the installation is successful. Click on Close.

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

**Note:** The installation process might take a couple of minutes.

**Verify the Python Installation Step 1:** Click on Start

**Step 2:** In the Windows Run Command, type “cmd”.



**Step 3:** Open the Command prompt option.

**Step 4:** Let us test whether the python is correctly installed. Type **python –V** and press Enter.



**Step 5:** You will get the answer as 3.7.4

**Note:** If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\Your Name>python --version

To check if you have python installed on a Linux or Mac, then on Linux open the command line or on Mac open the Terminal and type:

python –version

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https:/[/www.python.org/](http://www.python.org/)

Python QuickStart

Python is an interpreted programming language; this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line: C:\Users\Your Name>python helloworld.py

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py print("Hello, World!")

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

C:\Users\Your Name>python helloworld.py

The output should read:

Hello, World!

Congratulations, you have written and executed your first Python program. The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line: C:\Users\Your Name>python

Or, if the "python" command did not work, you can try "py": C:\Users\Your Name>py

From there you can write any python, including our hello world example from earlier in the tutorial:

C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print("Hello, World!")

Which will write "Hello, World!" in the command line: C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print("Hello, World!") Hello, World!

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

exit()

### Virtual Environments and Packages Introduction

Python applications will often use packages and modules that don’t come as part

of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library’s interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A’s environment.

**Creating Virtual Environments**

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

python3 -m venv tutorial-env

This will create the tutorial-env directory if it doesn’t exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you’ve created a virtual environment, you may activate it. On Windows, run:

tutorial-env\Scripts\activate.bat On Unix or MacOS, run:

source tutorial-env/bin/activate

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell’s prompt to show what virtual environment you’re using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

$ source ~/envs/tutorial-env/bin/activate (tutorial-env) $ python

Python 3.5.1 (default, May 6 2016, 10:59:36)

>>> import sys

>>>sys.path

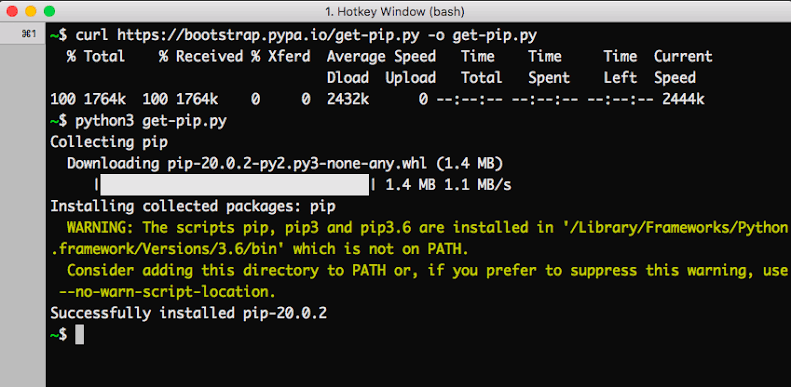
['', '/usr/local/lib/python35.zip', ..., '~/envs/tutorial-env/lib/python3.5/site-packages']

>>>

12.3. Managing Packages with pip

You can install, upgrade, and remove packages using a program called pip. By default, pip will install packages from the Python Package Index,

<https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip’s limited search feature:



(tutorial-env) $ pip search astronomy

skyfield - Elegant astronomy for Python

gary - Galactic astronomy and gravitational dynamics.

novas - The United States Naval Observatory NOVAS astronomy library

astroobs - Provides astronomy ephemeris to plan telescope observations PyAstronomy - A collection of astronomy related tools for Python.

...

pip has a number of subcommands: “search”, “install”, “uninstall”, “freeze”, etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package’s name: (tutorial-env) $ pip install novas

Collecting novas

Downloading novas-3.1.1.3.tar.gz (136kB) Installing collected packages: novas Running setup.py install for novas Successfully installed novas-3.1.1.3

You can also install a specific version of a package by giving the package name followed by == and the version number:

(tutorial-env) $ pip install requests==2.6.0 Collecting requests==2.6.0

Using cached requests-2.6.0-py2.py3-none-any.whl Installing collected packages: requests

Successfully installed requests-2.6.0

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

(tutorial-env) $ pip install --upgrade requests Collecting requests

Installing collected packages: requests

Found existing installation: requests 2.6.0

Uninstalling requests-2.6.0:

Successfully uninstalled requests-2.6.0 Successfully installed requests-2.7.0

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package: (tutorial-env) $ pip show requests

---

Metadata-Version: 2.0 Name: requests Version: 2.7.0

Summary: Python HTTP for Humans. Home-page: [http://python-requests.org](http://python-requests.org/) Author: Kenneth Reitz

Author-email: [me@kennethreitz.com](mailto:me@kennethreitz.com) License: Apache 2.0

Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages Requires:

pip list will display all of the packages installed in the virtual environment: (tutorial-env) $ pip list

novas (3.1.1.3)

numpy (1.9.2)

pip (7.0.3)

requests (2.7.0)

setuptools (16.0)

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

(tutorial-env) $ pip freeze > requirements.txt (tutorial-env) $ cat requirements.txt novas==3.1.1.3

numpy==1.9.2 requests==2.7.0

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

(tutorial-env) $ pip install -r requirements.txt

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))

...

Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

...

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

Installing collected packages: novas , numpy , requests Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you’ve written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

### Cross Platform

Platform. Architecture (executable=sys.executable, bits='', linkage='') Queries the given executable (defaults to the Python interpreter binary) for various architecture information.

Returns a tuple (bits, linkage) which contain information about the bit architecture and the linkage format used for the executable. Both values are returned as strings.

Values that cannot be determined are returned as given by the parameter presets. If bits is given as '', the sizeof(pointer) (or sizeof(long) on Python version < 1.5.2) is used as indicator for the supported pointer size.

The function relies on the system’s file command to do the actual work. This is available on most if not all Unix platforms and some non-Unix platforms and then only if the executable points to the Python interpreter. Reasonable defaults are used when the above needs are not met.

Note On Mac OS X (and perhaps other platforms), executable files may be universal files containing multiple architectures.

To get at the “64-bitness” of the current interpreter, it is more reliable to query the sys.maxsize attribute:

is\_64bits = sys.maxsize> 2\*\*32 platform.machine ()

Returns the machine type, e.g. 'i386'. An empty string is returned if the value cannot be determined.

platform.node ()

Returns the computer’s network name (may not be fully qualified!). An empty string is returned if the value cannot be determined.

platform. Platform(aliased=0, terse=0)

Returns a single string identifying the underlying platform with as much useful information as possible.

The output is intended to be human readable rather than machine parseable. It may look different on different platforms and this is intended.

If aliased is true, the function will use aliases for various platforms that report system names which differ from their common names, for example SunOS will be reported as Solaris. The system\_alias() function is used to implement this.

Setting terse to true causes the function to return only the absolute minimum information needed to identify the platform.

platform.processor()

Returns the (real) processor name, e.g. 'amdk6'.

An empty string is returned if the value cannot be determined. Note that many platforms do not provide this information or simply return the same value as for machine(). NetBSD does this.

platform.python\_build()

Returns a tuple (buildno, builddate) stating the Python build number and date as strings.

platform.python\_compiler()

Returns a string identifying the compiler used for compiling Python. platform.python\_branch()

Returns a string identifying the Python implementation SCM branch. New in version 2.6.

platform.python\_implementation()

Returns a string identifying the Python implementation. Possible return values are: ‘CPython’, ‘IronPython’, ‘Jython’, ‘PyPy’.

New in version 2.6. platform.python\_revision()

Returns a string identifying the Python implementation SCM revision. New in version 2.6.

platform.python\_version()

Returns the Python version as string 'major.minor.patchlevel'.

Note that unlike the Python sys.version, the returned value will always include the patchlevel (it defaults to 0).

platform.python\_version\_tuple()

Returns the Python version as tuple (major, minor, patchlevel) of strings.

Note that unlike the Python sys.version, the returned value will always include the patchlevel (it defaults to '0').

platform.release()

Returns the system’s release, e.g. '2.2.0' or 'NT' An empty string is returned if the value cannot be determined.

platform.system()

Returns the system/OS name, e.g. 'Linux', 'Windows', or 'Java'. An empty string is returned if the value cannot be determined.

platform.system\_alias(system, release, version)

Returns (system, release, version) aliased to common marketing names used for some systems. It also does some reordering of the information in some cases where it would otherwise cause confusion.

platform.version()

Returns the system’s release version, e.g. '#3 on degas'. An empty string is returned if the value cannot be determined.platform.uname()

Fairly portable uname interface. Returns a tuple of strings (system, node, release, version, machine, processor) identifying the underlying platform.

Note that unlike the os.uname() function this also returns possible processor information as additional tuple entry.

Entries which cannot be determined are set to ''.

## Java Platform

platform.java\_ver(release='', vendor='', vminfo=('', '', ''), osinfo=('', '', '')) Version interface for Jython.

Returns a tuple (release, vendor, vminfo, osinfo) with vminfo being a tuple (vm\_name, vm\_release, vm\_vendor) and osinfo being a tuple (os\_name, os\_version, os\_arch). Values which cannot be determined are set to the defaults given as parameters (which all default to '').

Windows Platform

platform.win32\_ver(release='', version='', csd='', ptype='')

Get additional version information from the Windows Registry and return a tuple (release, version, csd, ptype) referring to OS release, version number, CSD level (service pack) and OS type (multi/single processor).

As a hint: ptype is 'Uniprocessor Free' on single processor NT machines and 'Multiprocessor Free' on multi-processor machines. The ‘Free’ refers to the OS version being free of debugging code. It could also state ‘Checked’ which means the OS version uses debugging code, i.e. code that checks arguments, ranges, etc.

Note This function works best with Mark Hammond’s win32all package installed, but also on Python 2.3 and later (support for this was added in Python 2.6). It obviously only runs on Win32 compatible platforms.

### Win95/98 specific

platform.popen(cmd, mode='r', bufsize=None)

Portable popen() interface. Find a working popen implementation preferring win32pipe.popen(). On Windows NT, win32pipe.popen() should work; on Windows 9x it hangs due to bugs in the MS C library.

### Mac OS Platform

platform.mac\_ver(release='', versioninfo=('', '', ''), machine='')

Get Mac OS version information and return it as tuple (release, versioninfo, machine) with versioninfo being a tuple (version, dev\_stage, non\_release\_version).

Entries which cannot be determined are set to ''. All tuple entries are strings.

### Unix Platforms

platform.dist(distname='', version='', id='', supported\_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...))

This is an old version of the functionality now provided by linux\_distribution(). For new code, please use the linux\_distribution().

The only difference between the two is that dist() always returns the short name of the distribution taken from the supported\_dists parameter.

Deprecated since version 2.6.

platform.linux\_distribution(distname='', version='', id='', supported\_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...), full\_distribution\_name=1)

Tries to determine the name of the Linux OS distribution name.

supported\_dists may be given to define the set of Linux distributions to look for. It defaults to a list of currently supported Linux distributions identified by their release file name.

If full\_distribution\_name is true (default), the full distribution read from the OS is returned. Otherwise the short name taken from supported\_dists is used.

Returns a tuple (distname,version,id) which defaults to the args given as parameters. id is the item in parentheses after the version number. It is usually the version codename.

Note This function is deprecated since Python 3.5 and removed in Python 3.8. See alternative like the distro package.

New in version 2.6.

platform.libc\_ver(executable=sys.executable, lib='', version='', chunksize=2048)

Tries to determine the libc version against which the file executable (defaults to the Python interpreter) is linked. Returns a tuple of strings (lib, version) which default to the given parameters in case the lookup fails.

Note that this function has intimate knowledge of how different libc versions add symbols to the executable is probably only usable for executables compiled using gcc. The file is read and scanned in chunks of chunksize bytes.

### Using the Python Interpreter Invoking the Interpreter

The Python interpreter is usually installed as /usr/local/bin/python3.8 on those machines where it is available; putting /usr/local/bin in your Unix shell’s search

path makes it possible to start it by typing the command:

python3.8

to the shell. 1 Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., /usr/local/python is a popular alternative location.)

On Windows machines where you have installed Python from the Microsoft Store, the python3.8 command will be available. If you have the py.exe launcher installed, you can use the py command. See Excursus: Setting environment variables for other ways to launch Python.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn’t work, you can exit the interpreter by typing the following command: quit().

The interpreter’s line-editing features include interactive editing, history substitution and code completion on systems that support the GNU Readline library. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys.

If nothing appears to happen, or if ^P is echoed, command line editing isn’t available; you’ll only be able to use backspace to remove characters from the current line.

The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell’s -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote command in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using python -m module [arg] ..., which executes the source file for module as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

All command line options are described in Command line and environment. Argument Passing

When known to the interpreter, the script name and additional arguments thereafter are turned into a list of strings and assigned to the argv variable in the sys module. You can access this list by executing import sys. The length of the list is at least one; when no script and no arguments are given, sys.argv[0] is an empty string. When the script name is given as '-' (meaning standard input), sys.argv[0] is set to '-'. When -c command is used, sys.argv[0] is set to '-c'. When -m module is used, sys.argv[0] is set to the full name of the located module. Options found after -c command or -m module are not consumed by the Python interpreter’s option processing but left in sys.argv for the command or module to handle.

Interactive Mode

When commands are read from a tty, the interpreter is said to be in interactive mode. In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (>>>); for continuation lines it prompts with the secondary prompt, by default three dots (...). The interpreter prints a welcome

message stating its version number and a copyright notice before printing the first prompt:

$ python3.8

Python 3.8 (default, Sep 16 2015, 09:25:04) [GCC 4.8.2] on linux

Type "help", "copyright", "credits" or "license" for more information.

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

>>>

>>>the\_world\_is\_flat = True

>>>ifthe\_world\_is\_flat:

... print("Be careful not to fall off!")

...

Be careful not to fall off!

For more on interactive mode, see Interactive Mode. **The Interpreter and Its Environment Source Code Encoding**

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the first line of the file. The syntax is as follows:

# -\*- coding: encoding -\*-

where encoding is one of the valid codecs supported by Python.

For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

# -\*- coding: cp1252 -\*-

One exception to the first line rule is when the source code starts with a UNIX “shebang” line. In this case, the encoding declaration should be added as the second line of the file. For example:

#!/usr/bin/env python3 # -\*- coding: cp1252 -\*-

Introduction to Artificial Intelligence

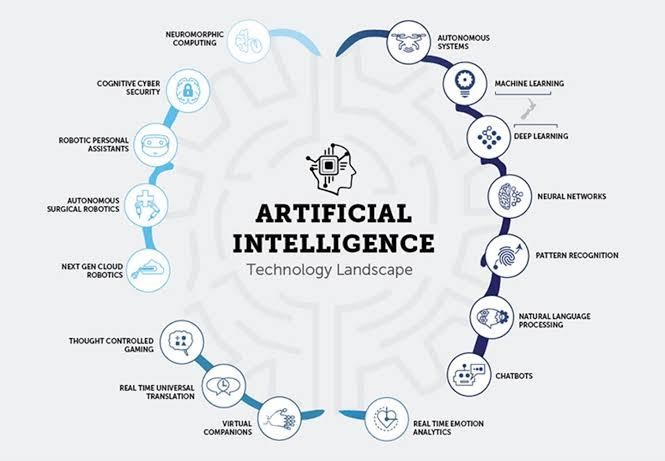
“The science and engineering of making intelligent machines, especially intelligent computer programs”. -John McCarthy-

### ARTIFICIAL INTELLIGENCE

Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs intelligent software systems. The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving.

The intelligence is intangible. It is composed of

* Reasoning
* Learning
* Problem Solving
* Perception
* Linguistic Intelligence



The objectives of AI research are reasoning, knowledge representation, planning, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the general intelligence sector.

Approaches include statistical methods, computational intelligence, and traditional coding AI. During the AI research related to search and mathematical optimization, artificial neural networks and methods based on statistics, probability, and economics, we use many tools. Computer science attracts AI in the field of science, mathematics, psychology, linguistics, philosophy and so on.

### Trending AI Articles:

1. Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Big Data
2. Data Science Simplified Part 1: Principles and Process
3. Getting Started with Building Realtime API Infrastructure
4. AI & NLP Workshop

### Applications of AI

* Gaming − AI plays important role for machine to think of large number of possible positions based on deep knowledge in strategic games. for example, chess, river crossing, N-queens’ problems and etc.

Natural Language Processing − Interact with the computer that understands natural language spoken by humans.

* Expert Systems − Machine or software provide explanation and advice to the users.
* Vision Systems − Systems understand, explain, and describe visual input on the computer.
* Speech Recognition − There are some AI based speech recognition systems have ability to hear and express as sentencesition software reads the text written on paper and recognize the shapes of the letters and convert it into editable text.
* Intelligent Robots − Robots are able to perform the instructions given by a human.

### Major Goals

* + Knowledge reasoning
  + Planning
  + Machine Learning
  + Natural Language Processing
  + Computer Vision
  + Robotics

### IBM Watson



“Watson” is an IBM supercomputer that combines Artificial Intelligence (AI) and complex inquisitive programming for ideal execution as a “question answering” machine. The supercomputer is named for IBM’s founder, Thomas J. Watson.

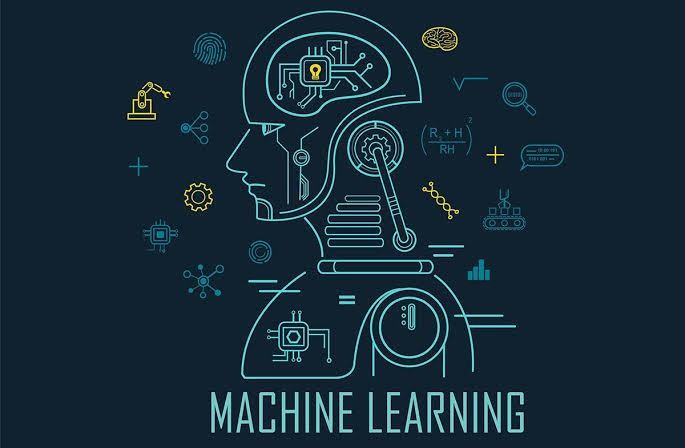
IBM Watson is at the forefront of the new era of computing. At the point when IBM Watson made, IBM communicated that “more than 100 particular techniques are used to inspect perceive sources, find and make theories, find and score affirm, and combination and rank speculations.” recently, the Watson limits have been expanded and the way by which Watson works has been changed to abuse new sending models (Watson on IBM Cloud) and propelled machine learning capacities and upgraded hardware open to architects and authorities. It isn’t any longer completely a request answering figuring system arranged from Q&A joins yet can now ‘see’, ‘hear’, ‘read’, ‘talk’, ‘taste’, ‘translate’, ‘learn’ and ‘endorse’.

### MACHINE LEARNING

**Introduction**

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve.



Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs. Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends.

Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self- driving cars that rely on machine learning to navigate may soon be available to consumers.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

In this tutorial, we’ll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning. We’ll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each.

Additionally, we’ll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

### Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let’s explore these methods in more detail.

### Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able

to “learn” by comparing its actual output with the “taught” outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as fish and images of oceans labeled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as fish and unlabeled ocean images as water.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

### Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.

The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

### Approaches

As a field, machine learning is closely related to computational statistics, so having a background knowledge in statistics is useful for understanding and leveraging machine learning algorithms. For those who may not have studied statistics, it can be helpful to first define correlation and regression, as they are commonly used techniques for investigating the relationship among quantitative variables.

**Correlation** is a measure of association between two variables that are not designated as either dependent or independent. **Regression** at a basic level is used to examine the relationship between one dependent and one independent variable. Because regression statistics can be used to anticipate the dependent variable when the independent variable is known, regression enables prediction capabilities.

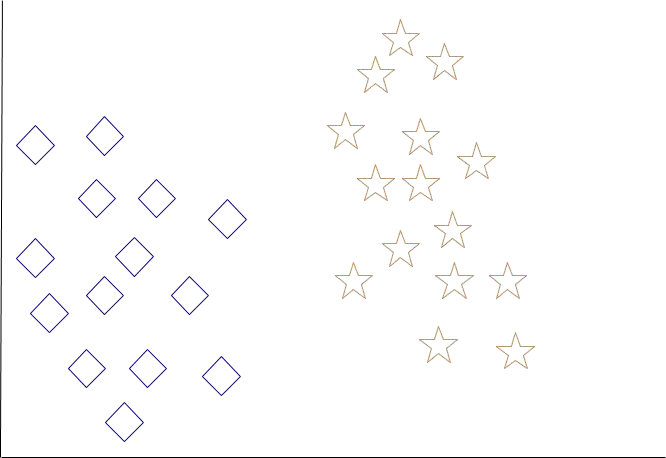
Approaches to machine learning are continuously being developed. For our purposes, we’ll go through a few of the popular approaches that are being used in machine learning at the time of writing.

### K-nearest neighbor

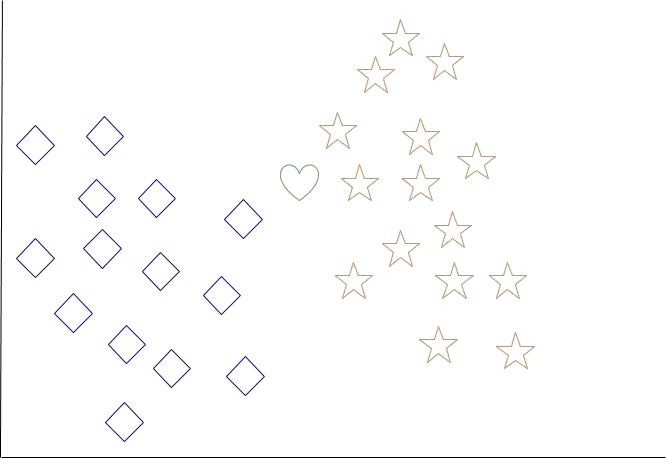
The k-nearest neighbor algorithm is a pattern recognition model that can be used for classification as well as regression. Often abbreviated as k-NN, the **k** in k-nearest neighbor is a positive integer, which is typically small. In either classification or regression, the input will consist of the k closest training examples within a space.

We will focus on k-NN classification. In this method, the output is class membership. This will assign a new object to the class most common among its k nearest neighbors. In the case of k = 1, the object is assigned to the class of the single nearest neighbor.

Let’s look at an example of k-nearest neighbor. In the diagram below, there are blue diamond objects and orange star objects. These belong to two separate classes: the diamond class and the star class.

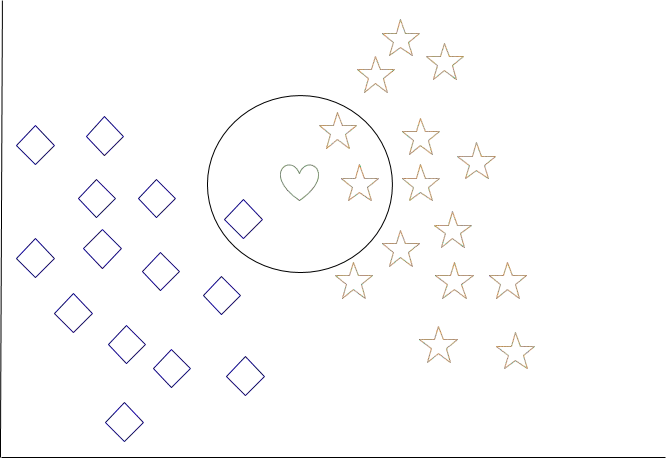


When a new object is added to the space — in this case a green heart — we will want the machine learning algorithm to classify the heart to a certain class.



When we choose k = 3, the algorithm will find the three nearest neighbors of the green heart in order to classify it to either the diamond class or the star class.

In our diagram, the three nearest neighbors of the green heart are one diamond and two stars. Therefore, the algorithm will classify the heart with the star class.



Among the most basic of machine learning algorithms, k-nearest neighbor is considered to be a type of “lazy learning” as generalization beyond the training data does not occur until a query is made to the system.

### Decision Tree Learning

For general use, decision trees are employed to visually represent decisions and show or inform decision making. When working with machine learning and data mining, decision trees are used as a predictive model. These models map observations about data to conclusions about the data’s target value.

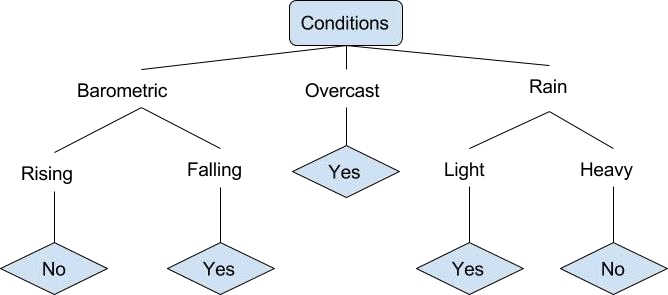
The goal of decision tree learning is to create a model that will predict the value of a target based on input variables.

In the predictive model, the data’s attributes that are determined through observation

are represented by the branches, while the conclusions about the data’s target value are represented in the leaves.

When “learning” a tree, the source data is divided into subsets based on an attribute value test, which is repeated on each of the derived subsets recursively. Once the subset at a node has the equivalent value as its target value has, the recursion process will be complete.

Let’s look at an example of various conditions that can determine whether or not someone should go fishing. This includes weather conditions as well as barometric pressure conditions.



In the simplified decision tree above, an example is classified by sorting it through the tree to the appropriate leaf node. This then returns the classification associated with the particular leaf, which in this case is either a Yes or a No. The tree classifies a day’s conditions based on whether or not it is suitable for going fishing.

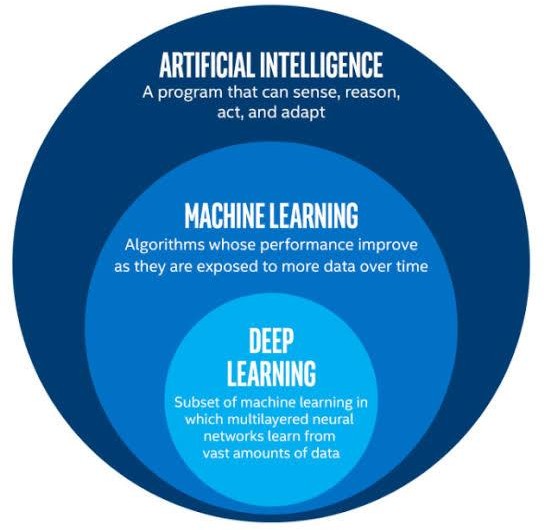
A true classification tree data set would have a lot more features than what is outlined above, but relationships should be straightforward to determine. When working with

decision tree learning, several determinations need to be made, including what features to choose, what conditions to use for splitting, and understanding when the decision tree has reached a clear ending.

### DEEP LEARNING Introduction to Deep Learning

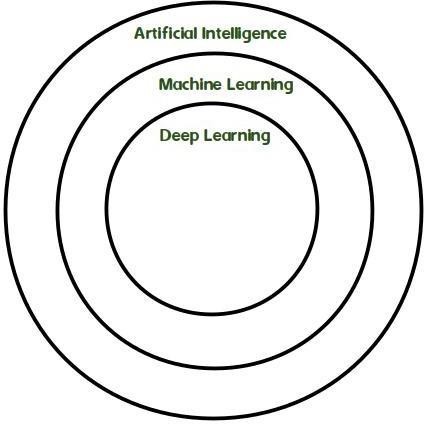
Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don’t need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It’s on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture.

A formal definition of deep learning is- neurons



In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbors. The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

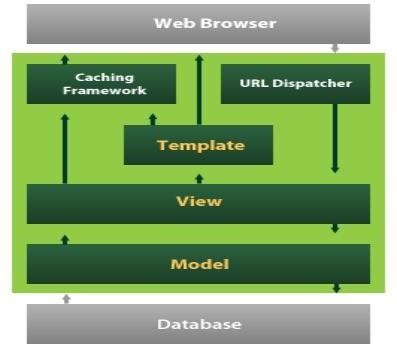
Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.



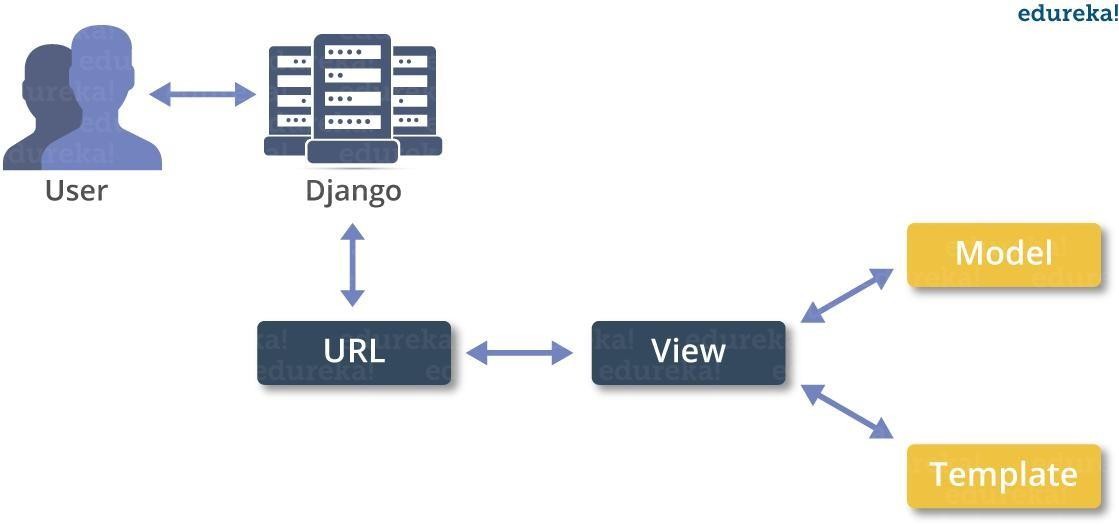
### DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.



Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models



**Create a sample application**

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py −

**YOUR CODE GOES HERE PASTE IT Pedha piki sontaga aythe rayale ga code XD :P**

# REQUIREMENTS AND TOOLS

### Software Requirements:

* + - Operating System: Windows 11
    - Programming languages: Python 3.7
    - Packages Pandas, Streamlit, Tensorflow
    - Audio test files with .mp3 or.wav extension (limit=200mb)

### Hardware Requirements:

* + - Processor: Intel core i5
    - RAM: 8GB
    - Hard disk: 254GB
    - Sound card or speakers

# SYSTEM ANALYSIS

### EXISTING SYSTEM AND DISADVANTAGES

Emotion recognition from speech has been a popular field of study. Several works have explored ways to improve this field. For feature extraction, Ghai et al. chose to take frame sample of the sound signals at 16000 Hz and the selection duration of each frame was 0.25 seconds. In the sample rate was 22050 Hz which were encoded through 16-bit PCM in two-channel.

Different works selected different framesize like 10-20 ms, 0.2 s, 10 ms etc. The most common features for audio speech data are MFCC, entropy and spectral entropy, ZCR (zero crossing rate), pitch, energy, formants etc.

Most of the papers gave emphasis on pitch and energy [1,10- 12,6,13,2]. Most of the papers calculated derivation and statistical features such as mean, standard deviation, etc. to increase accuracy. Previous works proposed different methods to classify emotions from speech such as Support Vector Machine (SVM), Gradient Boosting, K- Nearest Neighbor (KNN), Random Forest and Neural Network. They used different emotional speech databases to build their systems.

### PROPOSED SYSTEM AND ADVANTAGES

The classification model of emotion recognition here proposed is based on a deep learning strategy based on convolutional neural networks (CNN), Support Vector machine (SVM) classifier, MLP Classifier.

The key idea is considering the MFCC commonly referred to as the ”spectrum of a spectrum”, as the only feature to train the model. MFCC is a different interpretation of the Mel-frequency cepstrum (MFC), and it has been demonstrated to be the state of the art of sound formalization in automatic speech recognition task. The MFC coefficients have mainly been used as the consequence of their capability to represent the amplitude spectrum of the sound wavein a compact vectorial form.

As described in, the audio file is divided into frames, usually using a fixed window size, in order to obtain statistically stationary waves. The amplitude spectrum is normalized with a reduction of the ”Mel” frequency scale. This operation is performed for empathizing the frequency more meaningful for a significant reconstruction of the wave as the human auditory system can perceive. Foreach audio file,40 features have been extracted. The feature has been generated by converting each audio file to a floating-point time series. Then, an MFCC sequence has been created from the timeseries.

# RESULTS AND ANALYSIS

The data set was sub divided into a training set and testing set:

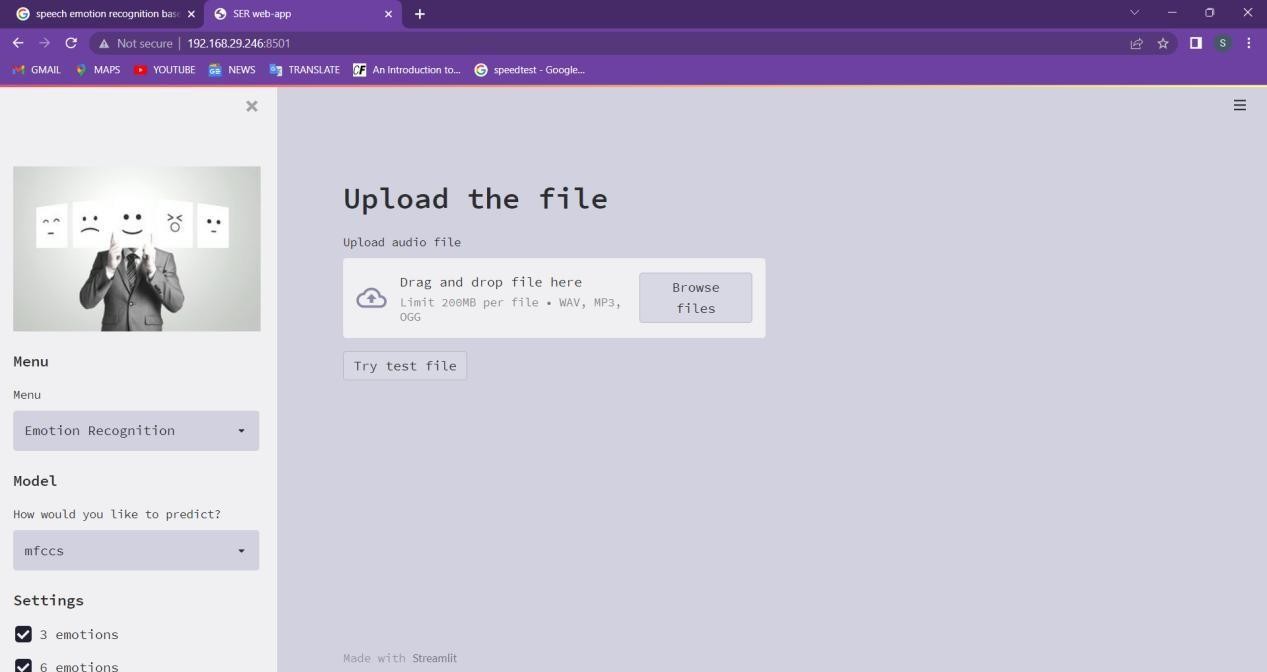
Training Set: 90.093%

Testing Set: 9.906%

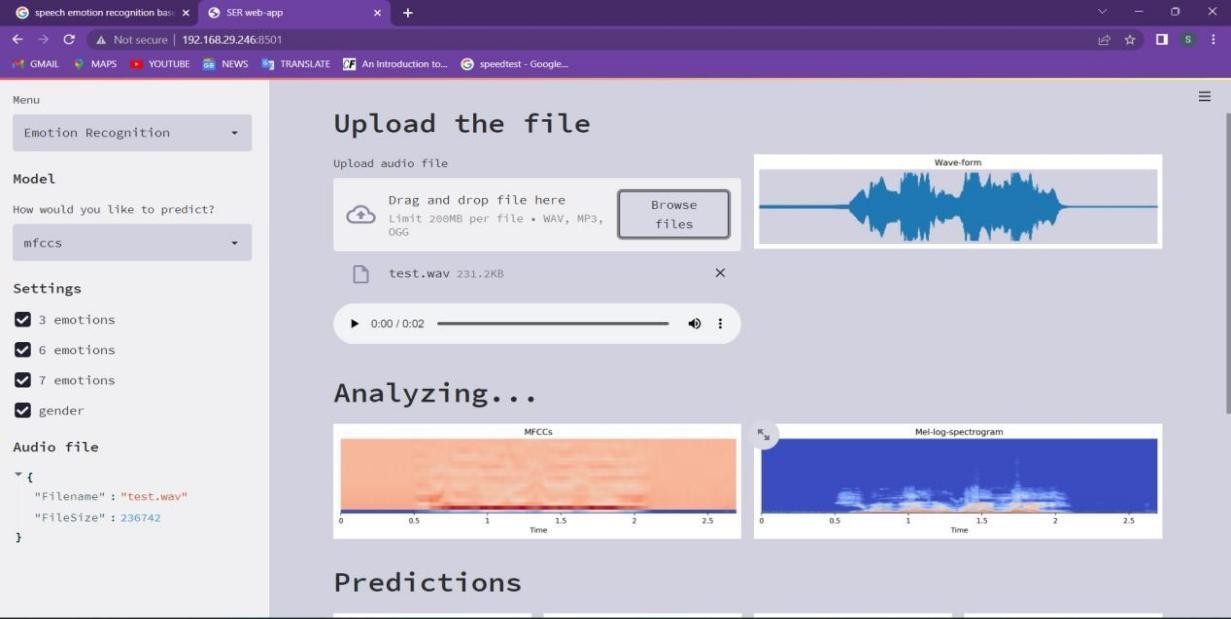
Once the features of each speech signal were extracted, their statistical parameters were calculated. These were then compiled into the feature vector providing us with a training set and a testing set. Each statistical parameter was tested separately and the best results were found using Mean. These labels were then appended to the training and testing matrices. The SVM used was the LibSVM. The SVM was trained using the Training data-set. Once training was complete a model was created and this was used for the testing phase.

A very good recognition rate was obtained for fear, anger and neutral emotions.

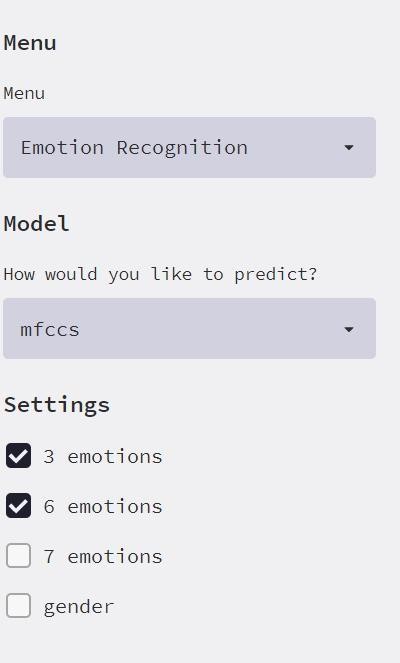
# SCREENSHOTS



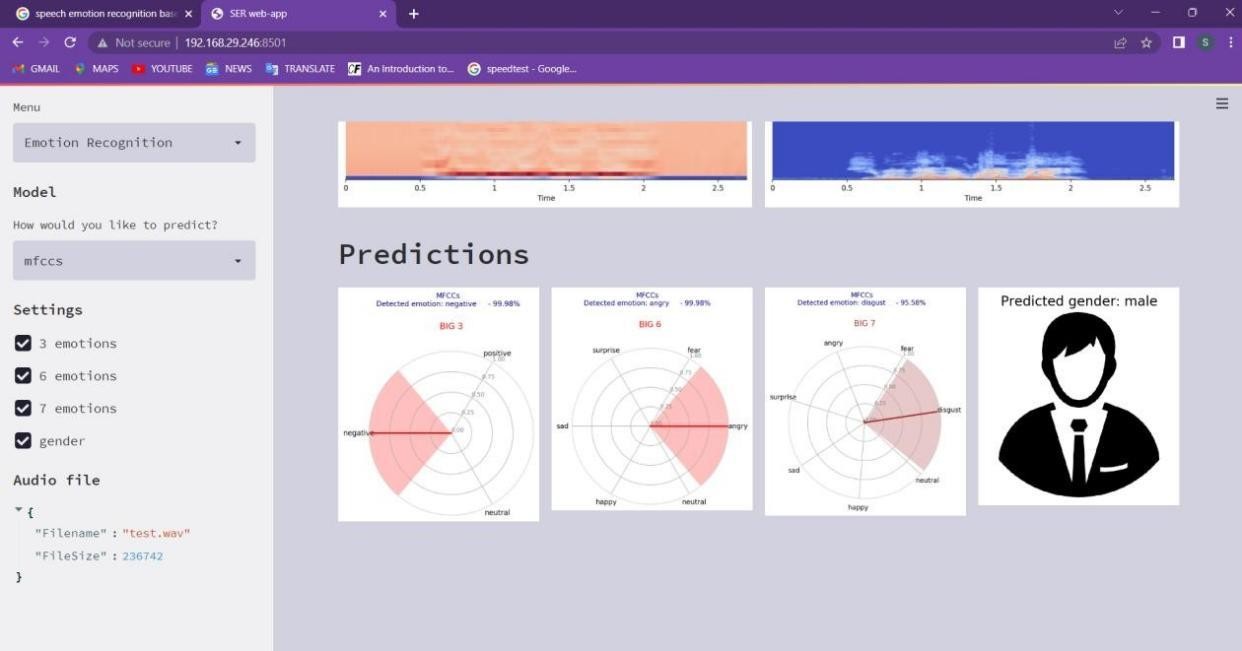
**Fig 8.1 Home page**



**Fig 8.2 Analyzing the audio file**



**Fig 8.3 Menu of analyzing audio**



**Fig 8.4. Predictions after analyzing audio**

# CONCLUSION

Recognizing emotions is automatically and subconsciously performed by humans. It is a vital process for human-to human communication, and thus, to achieve better human machine interaction, emotions need to be considered. The Automated Speech Emotion Recognition is a tough process because of the gap among acoustic characteristics and human emotions, which depends strongly on the discriminative acoustic characteristics extracted for a provided recognition task. Different persons have different emotions and altogether a different way to express it. Speech emotion do have different energies, pitch variations are emphasized if considering different subjects. Therefore, the speech emotion detection is a demanding task in computing vision.Here, the speech emotion recognition is based on the Convolutional Neural Network (CNN) algorithm which uses different modules for the emotion recognition and the classifiers are used to differentiate emotions such as happiness, surprise, anger, neutral state, sadness, etc. Emotional speech recognition importance is growing in several domains. So a solution to this we proposed a system that will do Speech detection and Continuous recognition of emotions from speech.

# REFERENCES

1. M. Neumann and N. T. Vu, "Improving Speech Emotion Recognition with Unsupervised Representation Learning on Unlabeled Speech," ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, United Kingdom, 2019, pp. 7390-7394, doi: 10.1109/ICASSP.2019.8682541.
2. T. Kexin, H. Yongming, Z. Guobao and Z. Lin, "Research on Emergency Parking Instruction Recognition Based on Speech Recognition and Speech Emotion Recognition," 2019 Chinese Automation Congress (CAC), Hangzhou, China, 2019, pp. 2933-2937, doi: 10.1109/CAC48633.2019.8997077.
3. J. Umamaheswari and A. Akila, "An Enhanced Human Speech Emotion Recognition Using Hybrid of PRNN and KNN," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019, pp. 177-183, doi: 10.1109/COMITCon.2019.8862221.
4. F. Chenchah and Z. Lachiri, "Speech emotion recognition in noisy environment," 2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Monastir, 2016, pp. 788-792, doi: 10.1109/ATSIP.2016.7523189.
5. Z. Han and J. Wang, "Speech emotion recognition based on Gaussian kernel nonlinear proximal support vector machine," 2017 Chinese Automation Congress (CAC), Jinan, 2017, pp. 2513-2516, doi: 10.1109/CAC.2017.8243198.
6. E. Bozkurt, E. Erzin, C. E. Erdem and A. T. Erdem, "Use of Line Spectral Frequencies for Emotion Recognition from Speech," 2010 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 3708-3711, doi: 10.1109/ICPR.2010.903.
7. H. Zheng and Y. Yang, "An Improved Speech Emotion Recognition Algorithm Based on Deep Belief Network," 2019 IEEE International Conference on Power,

Intelligent Computing and Systems (ICPICS), Shenyang, China, 2019, pp. 493-497, doi:

10.1109/ICPICS47731.2019.8942482.

1. R. Lotfidereshgi and P. Gournay, "Biologically inspired speech emotion recognition," 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 5135-5139, doi: 10.1109/ICASSP.2017.7953135.
2. O. Lahaie, R. Lefebvre and P. Gournay, "Influence of audio bandwidth on speech emotion recognition by human subjects," 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP).

### FUTURE SCOPE

Today, voice and natural language processing are at the forefront of any human machine interaction environment. The chapter emphasizes the tremendous progress that has taken place in machine learning, statistical data-mining and pattern recognition approaches that can help in making speech interfaces more versatile and pervasive. The growing requirements of speech interfaces also warn against the impediments that may come in the way of successful implementation of acoustically robust natural interfaces. Finally, the chapter underlines the technical advances and research efforts to be undertaken for high performance real-time speech recognition that will completely change the way humans interact with their computing devices.