VISUALIZING AND PREDICTING HEART DISEASES WITH AN INTERACTIVE DASHBOARD

PROJECT REPORT

Submitted by

TEAM ID : PNT2022TMID12921

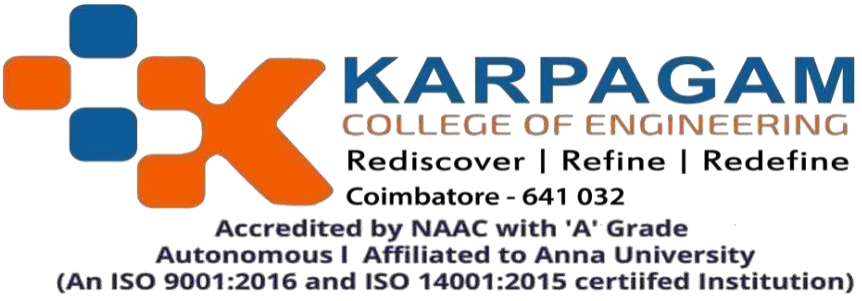
JANAKA LAKSHMI R	**19L318**
KEERTHANA N	**19L322**
SONA G	**19L342**
SUBIKSHA S	**19L345**

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING



NOVEMBER 2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

S.NO	ACRONYMS	EXPANSION
1	**DSS**	Decision Support System
2	**ML**	Machine Learning
3	**KEEL**	Knowledge Extraction Evolutionary Learning
4	**WHO**	World Health Organization
5	**IDE**	Integrated Development Environment
6	**CSV**	Comma Separated Values
7	**CNN**	Convolution Neural Network
8	**KNN**	K-Nearest Neighbor
9	**CART**	Classification and Regression Tree algorithm
10	**GUI**	Graphical User Interface

ABSTRACT

Heart disease, alternatively known as cardiovascular disease, encases various conditions that impact the heart and is the primary basis of death worldwide over the span of the past few decades. It associates many risk factors in heart disease and a need of the time to get accurate, reliable, and sensible approaches to make an early diagnosis to achieve prompt management of the disease. This Project presents various attributes related to heart disease, and the model on basis of supervised learning algorithms as, Naive Bayes, decision tree, K-nearest neighbor, and random forest algorithm. It uses the existing dataset from the Cleveland database of UCI repository of heart disease patients. The dataset comprises 303 instances and 76 attributes. Of these 76 attributes, only 14 attributes are considered for testing, important to substantiate the performance of different algorithms. Thisproject aims to envision the probability of developing heart disease in the patients. The Naive Bayes results portray that the highest accuracy score is achieved with K-nearest neighbor. This project describes cardiovascular disease and its pervasiveness. It explains the advantage of decision support systems for the prediction of heart disease. Medical diagnosis can be improved by the use of computer-based systems and algorithms taking decisions at the appropriate stages.Such systems are called decision support systems (DSSs). Thus, Intelligence alsoplays a role here.

CHAPTER 1

INTRODUCTION

According to the WHO (World Health Organization), every year 12 million deaths occur worldwide due to heart disease. Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of data analysis. The load of cardiovascular disease is rapidly increasing all over the world from the pastfew years. Many researches have been conducted in attempt to pinpoint the most influential factors of heart disease as well as accurately predict the overall risk. Heart Disease is even highlighted as a silent killer which leads to the death of the person without obvious symptoms. The early diagnosis of heart disease plays a vital role in making decisions on lifestyle changes in high-risk patients and in turn reduces the complications.

Machine learning proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the health care industry. This project aims to predict future heart disease by analyzing data of patients which classifies whether they have heart disease or not using machine-learning algorithm. Machine Learning techniques canbe a boon in this regard. Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources, classifying them under suitable headings & finally analyzing to extract the desired data we can say that this technique can be very well adapted to do the prediction of heart disease.

MOTIVATION FOR THE WORK

The main motivation of doing this research is to present a heart disease prediction model for the prediction of occurrence of heart disease. Further, this research work is aimed towards identifying the best classification algorithm for identifying the possibility of heart disease in a patient. This work is justified by performing a comparative study and analysis using three classification algorithms namely Naïve Bayes, Decision Tree, and Random Forest are used at different levels of evaluations. Although these are commonly used machine learning algorithms, the heart disease prediction is a vital task involving highest possible accuracy. Hence, the three algorithms are evaluated at numerous levels and types of evaluation strategies. This will provide researchers and medical practitioners to establish a better.

PROBLEM STATEMENT

The major challenge in heart disease is its detection. There are instruments available which can predict heart disease but either it is expensive or are not efficient to calculate chance of heart disease in human. Early detection of cardiac diseases can decrease the mortality rate and overall complications. However, it is not possible to monitor patients every day in all cases accurately and consultation of a patient for 24 hours by a doctor is not available since it requires more sapience, time and expertise. Since we have a good amount of data in today's world, we can use various machine learning algorithms to analyze the data for hidden patterns. The hidden patterns can be used for health diagnosis in medicinaldata.

OBJECTIVE

- The Primary Objective of this Project work regarding Heart disease Prediction. It is one of the most significant causes of mortality in the world today.
- The Primary goal is to Prediction the heart disease using Machine learning (ML) has

been shown to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry.

We propose a novel method KNN algorithms that aims at finding significant features by applying machine learning techniques resulting in improving the accuracyin the prediction of heart disease.

SCOPE OF THE PROJECT

Diagnosing Heart Disease starts with attaining a medical history of the patient for identifying the occurrence of disease. Threat factors to be identified throughtesting the presence of heart disease are confirmed. The healthcare provider would select the testing modality, which offers diagnosis if heart disease is present, decide the impairment.

CHAPTER 2

LITERATURE SURVEY

[1]. Purushottam T,

"Efficient Heart Disease Prediction System" published in 2019

Methodology:

They used Cleveland dataset and preprocessing of data is performed before using classification algorithms. The Knowledge Extraction is done based on Evolutionary Learning (KEEL), an open-source data mining tool that fills the missing values in the data set. A decision tree follows top-down order. For each actual node selected by hill-climbing algorithm anode is selected by a test at each level. The parameters and their values used are confidence.

[2]. Santhana Krishnan. J,

"Prediction of Heart Disease Using Machine Learning Algorithms" Published in 2007

Methodology:

Tree and Naive Bayes algorithm for prediction of heart disease. In decision tree algorithm the tree is built using certain conditions which gives True or False decisions. The algorithms like SVM, KNN are results based on vertical or horizontal split conditions depends on dependent variables. But decision tree for a tree like structure having root node, leaves and branches base on the decision made in each of tree Decision tree also help in the understating the importance of the attributes in the dataset.They have also used Cleveland data set. Dataset splits in 70% training and 30% testingby using some methods. This algorithm gives 91% accuracy. The second algorithm is Naive Bayes, which is used for classification.

[3]. Sonam Nikhar e,

"Prediction of Heart Disease Using Machine Learning Algorithms" Published in 2010

Methodology:

Their research gives point to point explanation of Naïve Bayes and decision tree classifier that are used especially in the prediction of heart disease.

[4]. Aditi Gavhane et al,

"Prediction of Heart Disease Using Machine Learning", Published in 2014

Methodology:

Testing of dataset is performed by using neural network algorithm multi-layer perceptron. In this algorithm there will be one input layer and one output layer and one or more layers are hidden layers between these two input and output layers. Through hidden layers each input node is connected to output layer. This connection is assigned with some random weights. The other input is called bias which is assigned with weight based on requirement the connection between the nodes can be feed forwarded or feedback.

[5]. Avinash Golande et al,

"Heart Disease Prediction Using Effective Machine Learning Techniques" Published in 2019

Methodology:

In which few data mining techniques are used that support the doctors to differentiate the heart disease. Usually utilized methodologies are k-nearest neighbour, Decisiontree and Naïve Bayes. Other unique characterization-based strategies utilized are packing calculation, Part thickness, consecutive negligiblestreamlining and neural systems, straight Kernel self-arranging guide and SVM (Bolster Vector Machine).

[6]. Lakshmana Rao et al,

"Machine Learning Techniques for Heart Disease Prediction" Published in 2008

Methodology:

In which the contributing elements for heart disease are more. So, it is difficult to distinguish heart disease. To find the seriousness of the heart disease among people different neural systems and data mining techniques are used.

[7]. Abhay Kishore V,

"Heart Attack Prediction Using Deep Learning" Published in 2004

Methodology:

In which heart attack prediction system by using Deep learning techniques and to predict the probable aspects of heart related infections of the patient Recurrent Neural System is used. This model uses deep learning and data mining to give the best precise model and least blunders. This paper acts as strong reference model for another type of heart attack prediction models

[8]. Senthil Kumar Mohan et al,

"Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques" Published in 2016

Methodology:

Their main objective is to improve exactness in cardiovascular problems. The algorithms used are KNN, LR, SVM, NN to produce an improved exhibition level with a precision level of 88.7% through the prediction model for heart disease with hybrid random forest with linear model (HRFLM).

CHAPTER 3

SYSTEM ANALYSIS

PROBLEM DEFINITION

Heart is one of the significant organs present in the human body. It is a muscle that is made with 4 chambers divided by valves and two divisions. Every half consists of single chamber termed as atrium and one called ventricle. The atria (plural term of atrium) gather blood, and ventricles contract to move blood from heart. The right half of the heart pumps oxygen, i.e., blood, which has minimum amount of oxygen to lungs where blood cells obtain maximum oxygen. Next, fresh oxygenated blood transfers from lungs to the left atrium and left ventricle. Left ventricle pumps the new oxygenated blood to alternate organs and other tissues of the body. The oxygenated blood promotes more energy and health for human body According to National Heart, Lung, and Blood Institute (2008), heart disease is common for a widerange of diseases, anomalies, and limitations which influence the heart as well as the blood vessels. It is prone to males and females in US and crosses millions in Americans where they have myocardial infarctions.

ADVANTAGES

Diagnosing Heart Disease starts with attaining a medical history of the patient for identifying the occurrence of disease. Threat factors to be identified through testing the presence of heart disease are confirmed. The healthcare provider would select the testing modality, which offers diagnosis if heart disease is present, decide the impairment.

SYSTEM SPECIFICATION

HARDWARE REQUIREMENTS

The hardware requirements for executing this model are:

- RAM – 4 GB
- Operating System – Windows 10
- Processor – Intel(R) Core (TM) i3
- Processor speed – 3.60 GHz

SOFTWARE REQUIREMENTS

The programming language used to develop this application is

- Python and the IDE used is Jupyter Notebook.
- Programming Language – Python
- Python IDE – Jupyter Notebook
- Deep Learning Framework – Tensor flow.

CHAPTER 4

SYSTEM ARCHITECTURE

Overall Architecture

This research aims to foresee the odds of having heart disease as probable cause of computerized prediction of heart disease that is helpful in the medical field for clinicians and patients. To accomplish the aim, we have discussed the use of various machine learning algorithms on the data set and dataset analysis is mentioned in this research paper. This paper additionally depicts which attributes contribute more than the others to anticipation of higher precision. This may spare the expense of different trialsof a patient, as all the attributes may not contribute such a substantial amount to expect the outcome.

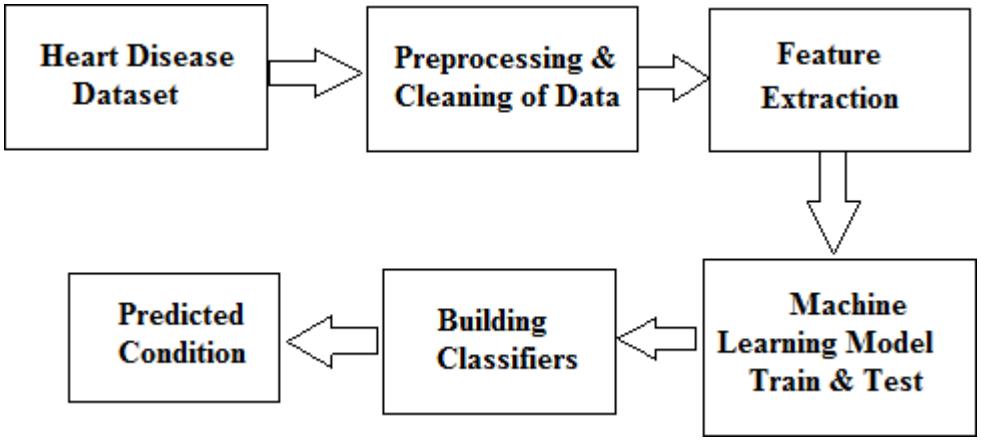


Fig 4.1.1 Overall proposed architecture

DATASET DESCRIPTION

The dataset is publicly available on the Kaggle Website at [4] which is from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. The dataset comprises 303 instances and 76 attributes. Of these 76 attributes, only 14 attributes are considered for testing, important to substantiate the performance of different algorithms. The attributes include: age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting, sugar blood, resting electrocardiographic results, maximum heart rate, exercise induced

angina, ST depression induced by exercise, slope of the peak exercise, number of major vessels, and target ranging from0 to 2, where 0 is absence of heart disease. The data set is in csv (Comma Separated Value) format which is further prepared to data frame as supported by panda's library in python.

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4235	0	48	2.0	1	20.0	NaN	0	0	0	248.0	131.0	72.0	22.00	84.0	86.0
4236	0	44	1.0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19.16	86.0	NaN
4237	0	52	2.0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.47	80.0	107.0
4238	1	40	3.0	0	0.0	0.0	0	1	0	185.0	141.0	98.0	25.60	67.0	72.0
4239	0	39	3.0	1	30.0	0.0	0	0	0	196.0	133.0	86.0	20.91	85.0	80.0

4240 rows × 16 columns

Fig 4.2.1 Dataset configuration

The education data is irrelevant to the heart disease of an individual, so it is dropped. Further with this dataset pre-processing and experiments are then carried out.

Proposed Technique

- We are Using KNN Algorithm for Heart Disease Prediction

Expected Advantages

- Diagnosing the heart disease correctly & providing effective treatment to patients will define the quality of service
- This Project aims to foresee the odds of having heart disease as probable cause of computerized prediction of heart disease that is helpful in the medical field for clinicians and Patients

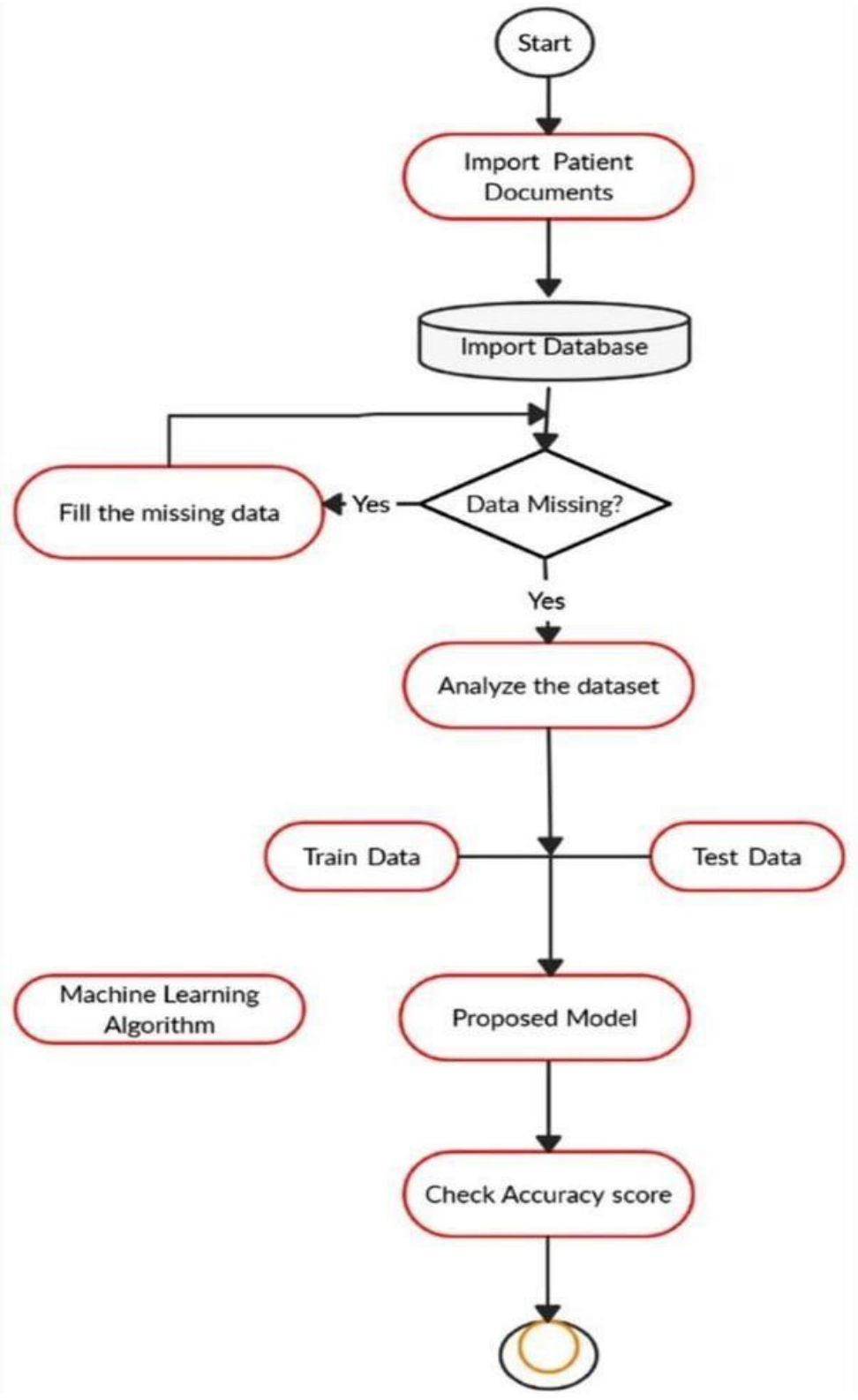


Fig 4.2.2 Flow chart for proposed system

CHAPTER 5

SYSTEM IMPLEMENTATION

HARDWARE AND SOFTWARE SPECIFICATION

HARDWARE REQUIREMENTS:

- Hard Disk: 500GB and Above
- RAM: 4GB and Above
- Processor: I 3 and Above

SOFTWARE REQUIREMENTS:

- Operating System: Windows 7, 8, 10 (64 bit)
- Software: Python 3.7
- Tools: Anaconda (Jupyter Notebook IDE), Teachable Machine, COLAB

TECHNOLOGIES USED:

- Python
- Deep learning
- Tensor Flow

Introduction to Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

- Web development (server-side)
- Software development
- Mathematics
- System Scripting

Uses of Python:

- 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.

Characteristics of 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 a functional way.

Python 2 Vs Python 3:

- 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.
- Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been back ported to Python 2. But in general, they remain not quite compatible.
- Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End of Life date of January 1, 2020 has been established for Python 2, after which time it will no longer be maintained.
- Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python's Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.
- 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.

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.

Python is Interpreted:

- Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.
- This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.
- One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.
- In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.
- For all its syntactic simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.
- Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.

- Python accomplishes what many programming languages don’t: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

TENSOR FLOW

Commonly used software for ML tasks is TF. It is widely adopted as it provides an interface to express common ML algorithms and executable code of the models. Models created in TF can be ported to heterogeneous systems with little or no change with devices ranging from mobile phones to distributed servers. TF was created by and is maintained by Google, and is used internally within the company for ML purposes. TF expresses computations as a stateful data flow graph enabling easy scaling of ANN training with parallelization and replication. As the model described in this paper is to be trained on computational servers and later ported to mobile devices, TF is highly suitable

TENSOR FLOW MOBILE

During design, Google developed TF to be able to run on heterogeneous systems, including mobile devices. This was due to the problems of sending data back and forth between devices and data centers when computations could be executed on the device instead. TFM enabled developers to create interactive applications without the need of network round-trip delays for ML computations. As ML tasks are computationally expensive, model optimization is used to improve performance. The minimum hardware requirements of TFM in terms of Random-Access Memory (RAM) size and CPU speed are low, and the primary bottleneck is the calculation speed of the computations as the desired latency for mobile applications is low. For example, a mobile device with hardware capable of running 10 Giga Floating Point Operations Per Second (FLOPS) is limited to run a 5 GFLOPS model in 2 FPS, which might impede desired application performance.

TENSORFLOW FRAMEWORK:

Tensor flow is an open-source software library.

Tensor flow was originally developed by researchers and engineers.

It is working on the Google Brain Team within Google's Machine Intelligence research organization the purposes of conducting machine learning and deep neural networks research.

It is an open source framework to run deep learning and other statistical and predictive analytics workloads.

It is a python library that supports many classification and regression algorithms and more generally deep learning.

Tensor Flow 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 is Google Brain's second-generation system.

Version 1.0.0 was released on February 11, While the reference implementation runs on single devices, Tensor Flow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units).
Tensor Flow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

The name Tensor Flow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

OPENCV:

1. It is a cross-platform library using which we can develop real-time computer vision applications.
2. It mainly focuses on image processing, video capture and analysis including feature like face detection and object detection. Currently Open CV supports a wide variety of programming languages like C++, Python, Java etc. and is available on different platforms including Windows, Linux, OS X, Android, iOS etc.
3. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations. Open CV-Python is the Python API of Open CV.
4. It combines the best qualities of Open CV C++ API and Python language.
5. OpenCV (Open-Source Computer Vision Library) is an opensource computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.
6. The library has more than 2500optimized algorithms, which includes a comprehensive set of both classic and state-of -the-art computer vision and machine learning algorithms.
7. Algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images

from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

NUMPY:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Num array into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific and engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer and maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier.

An implementation of a matrix package was completed by Jim Fulton, then generalized by Jim Hugunin to become Numeric also variously called Numerical Python extensions or NumPy Hugunin, a graduate student at Massachusetts Institute of Technology (MIT) joined the Corporation for National Research Initiatives (CNRI) to work on J Python in 1997 leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to take over as maintainer. In early 2005, NumPy developer Travis Oliphant wanted to unify the community around a single array package and ported num-array's features to Numeric, releasing the result as NumPy 1.0 in 2006. This new project was part of SciPy. To avoid installing the large SciPy package just to get an array object, this new package was separated and called NumPy.

MATPLOT:

Mat plot is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, WX Python, Qt, or GTK+. There is also a procedural "Pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and further joined by Thomas Caswell.

IPYTHON

What exactly is Python? You may be wondering about that. You may be referring to this book because you wish to learn editing but are not familiar with editing languages. Alternatively, you may be familiar with programming languages such as C, C ++, C #, or Java and wish to learn more about Python language and how it compares to these "big word" languages.

PYTHON CONCEPTS

You can skip to the next chapter if you are not interested in how and why Python. In this chapter, I will try to explain why I think Python is one of the best programming languages available and why it is such a great place to start.

Python was developed into an easy-to-use programming language. It uses English words instead of punctuation, and has fewer syntax than other languages. Python is a highly developed, translated, interactive, and object-oriented language.

Python translated - Interpreter processing Python during launch. Before using your software, you do not need to install it. This is similar to PERL and PHP editing languages.

Python interactive - To write your own applications, you can sit in Python Prompt and communicate directly with the interpreter.

Python Object-Oriented - Python supports the Object-Oriented program style or method, encoding the code within objects.

Python is a language for beginners - Python is an excellent language for beginners, as it allows for the creation of a variety of programs, from simple text applications to web browsers and games.

Python Features

Python features include -

1. Easy-to-learn - Python includes a small number of keywords, precise structure, and well-defined syntax. T This allows the student to learn the language faster
2. Easy to read - Python code is clearly defined and visible to the naked eye.
3. Easy-to-maintain - Python source code is easy to maintain.
4. Standard General Library - Python's bulk library is very portable and shortcut compatible with UNIX, Windows, and Macintosh.
5. Interaction mode - Python supports interaction mode that allows interaction testing and correction of captions errors.
6. Portable - Python works on a variety of computer systems and has the same user interface for all.
7. Extensible - Low-level modules can be added to Python interpreter. These modules allow system developers to improve the efficiency of their tools either by installing or customizing them.
8. Details - All major commercial information is provided by Python ways of meeting.

9. GUI Programming - Python assists with the creation and installation of a user interface for images of various program phones, libraries, and applications, including Windows MFC, Macintosh, and Unix's X Window.
10. Scalable - Major projects benefit from Python building and support, while Shell writing is not

Aside from the characteristics stated above, Python offers a long list of useful features, some of which are described below.

- It supports OOP as well as functional and structured programming methodologies.
- It can be used as a scripting language or compiled into Byte-code for large-scale application development.
- It allows dynamic type verification and provides very high-level dynamic data types.
- Automatic garbage pickup is supported by IT.

ADVANTAGES/BENEFITS OF PYTHON:

The diverse application of the Python language is a result of the combination of features which

give this language an edge over others. Some of the benefits of programming in Python include:

1. Presence of Third-Party Modules:

The Python Package Index (PyPI) contains numerous third-party modules that make Python capable of interacting with most of the other languages and platforms.

2. Extensive Support Libraries:

Python provides a large standard library which includes areas like internet protocols, string

operations, web services tools and operating system interfaces. Many high use programming tasks have already been scripted into the standard library which reduces length of code to be written significantly.

3. Open Source and Community Development:

Python language is developed under an OSI-approved opensource license, which makes it free to use and distribute, including for commercial purposes. Further, its development is driven by the community which collaborates for its code through hosting conferences and mailing lists, and provides for its numerous modules.

4. Learning Ease and Support Available:

Python offers excellent readability and uncluttered simple-to-learn syntax which helps beginners to utilize this programming language. The code style guidelines, PEP 8, provide a set of rules to facilitate the formatting of code. Additionally, the wide base of users and active developers has resulted in a rich internet resource bank to encourage development and the continued adoption of the language.

5. User-friendly Data Structures:

Python has built-in list and dictionary data structures which can be used to construct fast runtime data structures. Further, Python also provides the option of dynamic high-level data typing which reduces the length of support code that is needed.

6. Productivity and Speed:

Python has Clean object-oriented design, provides enhanced process control capabilities, and possesses strong integration and text processing capabilities and its own unit testing framework, all of which contribute to the increase in its speed and productivity. Python is considered a viable option for building complex multi-protocol network applications. 10 As can be seen from the above-mentioned points, Python offers a number of

advantages for software development. As upgrading of the language continues, its loyalist base could grow as well.

Python has five standard data types:

- Numbers
- String
- List
- Tuple
- Dictionary

PYTHON NUMBERS

Numeric values are stored in number data types. When you give a number a value, it becomes a number object.

PYTHON STRINGS

In this python uses a string is defined as a collection set of characters enclosed in quotation marks. Python allows you to use any number of quotes in pairs. The slice operator ([] and [:]) can be used to extract subsets of strings, with indexes starting at 0 at the start of the string and working their way to -1 at the end.

PYTHON LISTS

The most diverse types of Python data are lists. Items are separated by commas and placed in square brackets in the list ([]). Lists are similar to C-order in some ways. Listings can be for many types of data, which is one difference between them.

The slide operator ([] and [:]) can be used to retrieve values stored in the list, the indicators start with 0 at the beginning of the list and work their way to the end of the list. The concatenation operator of the list is a plus sign (+), while the repeater is an asterisk (*).

PYTHON TUPLES

A cone is a type of data type similar to a sequence of items. Cone is a set of values separated by commas. The pods, unlike the list, are surrounded by parentheses. Lists are placed in parentheses ([]), and the elements and sizes can be changed, but the lumps are wrapped in brackets (()) and cannot be sorted. Powders are the same as reading lists only.

PYTHON DICTIONARY

Python dictionaries in Python are a way of a hash table. They are similar to Perl's combination schemes or hashes, and are made up of two key numbers. The dictionary key can be any type of Python, but numbers and strings are very common. Prices, on the other hand, can be anything you choose Python. Curly braces () surround dictionaries, and square braces ([) are used to assign and access values.

Different modes in python

Python Normal and interactive are the two basic Python modes. The scripted and completed.py files are executed in the Python interpreter in the regular manner.

Interactive mode is a command line shell that provides instant response for each statement while simultaneously running previously provided statements in active memory.

The feed programme is assessed in part and whole as fresh lines are fed into the interpreter.

PANDAS

Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data

and time series. It is mainly popular for importing and analysing data much easier. Pandas is fast and it has high-performance & productivity for users.

Pandas is a python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real-world** data analysis in Python. Additionally, it has the broader goal of becoming **the most powerful and flexible opensource data analysis/manipulation tool available in any language**. It is already well on its way toward this goal.

Explore data analysis with Python. Pandas Data Frames make manipulating your data easy, from selecting or replacing columns and indices to reshaping your data.

Pandas is well suited for many different kinds of data:

Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet Ordered and unordered (not necessarily fixed-frequency) time series data.

Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels Any other form of observational / statistical data sets. The data need not be labeled at all to be placed into a Pandas data structure.

KERAS

KERAS is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages.

It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with

image and text data easier to simplify the coding necessary for writing deep neural network code.

The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras is a minimalist Python library for deep learning that can run on top of Theano or Tensor Flow.

It was developed to make implementing deep learning models as fast and easy as possible for research and development.

FOUR PRINCIPLES:

- Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.
- Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.
- Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.
- Python: No separate model files with custom file formats. Everything is native Python. Keras is designed for minimalism and modularity allowing you to very quickly define deep learning models and run them on top of a Theano or TensorFlow backend.

CHAPTER 6
METHODOLOGY

Naive Bayes Classifier Algorithm

Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basisof the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

K-Nearest Neighbor (KNN) Algorithm

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. KNN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using KNN algorithm. KNN algorithm can be used for Regression as wellas for Classification but mostly it is used for the Classification problems. K-NN is a **non- parametric algorithm**, which means it does not make any assumption on underlying data. It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Decision tree Algorithm

Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed onthe basis of features of the given dataset. It is a graphical representation for getting all the possiblesolutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**

Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problemsin ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance ofthe model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, andit predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

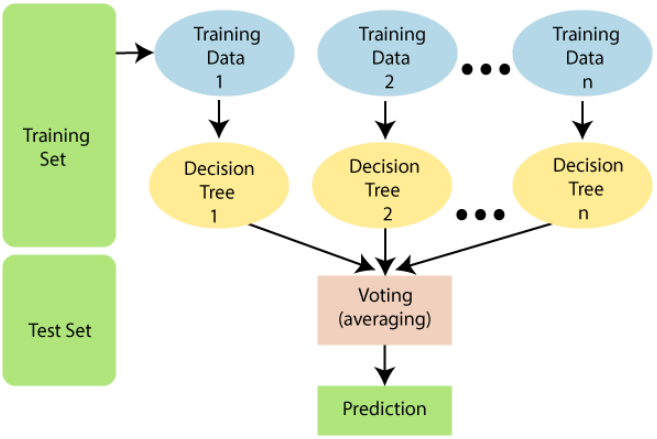


Fig 6.4.1 Block diagram for Random Forest algorithm

CHAPTER 7

CONCLUSION AND FUTURE WORK

The early prognosis of cardiovascular diseases can aid in making decisions on lifestyle changes in high-risk patients and in turn reduce the complications, which can be a great milestone in the field of medicine. This project resolved the feature selection i.e. decision tree and randomforest algorithm and successfully predict the heart disease, with 85% accuracy. The model used was K-nearest algorithm. Further for its enhancement, we can train on models and predict the types of cardiovascular diseases providing recommendations to the users, and also use more enhanced models. The proposed system is GUI-based, user-friendly, scalable, reliable and an expandable system. The proposed working model can also help in reducing treatment costs by providing Initial diagnostics in time. The model can also serve the purpose of training tool for medical students and will be a soft diagnostic tool available for physicians and cardiologists. General physicians can utilize this tool for initial diagnosis of cardio-patients. There are many possible improvements that could be explored to improve the scalability and accuracy of this prediction system. As we have developed a generalized system, in future we can use this systemfor the analysis of different data sets. The performance of the health's diagnosis can be improved significantly by handling numerous class labels in the prediction process, so identification and selection of significant attributes for better diagnosis of heart disease are very challenging tasks for future research.

REFERENCES

[1] Senthil Kumar Mohan et al, "Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques" Published in 2016

[2] Avinash Golande et al, "Heart Disease Prediction Using Effective Machine Learning Techniques" Published in 2019

[3] Lakshmana Rao et al, "Machine Learning Techniques for Heart Disease Prediction" Published in 2008

[4] Aditi Gavhane et al, "Prediction of Heart Disease Using Machine Learning", Published in 2014

[5] Sonam Nikhar e, "Prediction of Heart Disease Using Machine Learning Algorithms" Published in 2010

[6] Santhana Krishnan. J, "Prediction of Heart Disease Using Machine Learning Algorithms" Published in 2007

[7] Purushottam T, "Efficient Heart Disease Prediction System" published in 2019

[8] Abhay Kishore V, "Heart Attack Prediction Using Deep Learning" Published in 2004

APPENDIX

```
import tensorflow as tf
import numpy as np
import pandas as pd
from random import shuffle
import os
import cv2
from tensorflow.python.keras.models import Model, Sequential
from tensorflow.python.keras import layers
from tensorflow.python.keras.layers import Activation, Convolution2D, Dropout,
Conv2D
from tensorflow.python.keras.layers import AveragePooling2D, BatchNormalization
from tensorflow.python.keras.layers import GlobalAveragePooling2D
from tensorflow.python.keras.layers import Input, MaxPooling2D, SeparableConv2D
from tensorflow.python.keras.callbacks import CSVLogger, ModelCheckpoint,
EarlyStopping
from tensorflow.python.keras.callbacks import ReduceLROnPlateau, TensorBoard
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.python.keras.regularizers import l2
# Parameters
batch_size = 32
num_epochs = 10000
input_shape = (64, 64, 1)
validation_split = .2
verbose = 1
num_classes = 7
```

```
patience = 50
base_path = 'training_output/'
# Data generator
data_generator = ImageDataGenerator(
featurewise_center=False,
featurewise_std_normalization=False,
rotation_range=10,
width_shift_range=0.1,
height_shift_range=0.1,
zoom_range=.1,
horizontal_flip=True)
class DataManager(object):
def__init__(self, dataset_name='fer2013', dataset_path=None, image_size=(48, 48)):
self.dataset_name = dataset_name
self.image_size = image_size

self.dataset_path = 'datasets/fer2013/fer2013.csv'
def get_data(self):
data = self._load_fer2013()
return data
def _load_fer2013(self):
data = pd.read_csv(self.dataset_path)
pixels = data['pixels'].tolist()
width, height = 48, 48
faces = []
for pixel_sequence in pixels:
face = [int(pixel) for pixel in pixel_sequence.split(' ')]
face = np.asarray(face).reshape(width, height)
face = cv2.resize(face.astype('uint8'), self.image_size)
```

```
faces.append(face.astype('float32'))
faces = np.asarray(faces)
faces = np.expand_dims(faces, -1)
emotions = pd.get_dummies(data['emotion']).as_matrix()
return faces, emotions
def split_data(x, y, validation_split=.2):
num_samples = len(x)
num_train_samples = int((1 - validation_split)*num_samples)
train_x = x[:num_train_samples]
train_y = y[:num_train_samples]
val_x = x[num_train_samples:]
val_y = y[num_train_samples:]
train_data = (train_x, train_y)
val_data = (val_x, val_y)
return train_data, val_data
def preprocess_input(x):
x = x.astype('float32')
x = x / 255.0
#- data_format mode=tf: will scale pixels between -1 and 1, sample-wise.
x = x - 0.5
x = x * 2.0
return x
def myModel(input_shape, num_classes):
# Base Modulr
img_input = Input(input_shape)
x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=l2(0.01),

use_bias=False) (img_input)
x = BatchNormalization()(x)
```

```
x = Activation('relu')(x)
x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=l2(0.01), use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
# Residual Module 1
residual = Conv2D(16, (1, 1), strides=(2, 2), padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)
x = SeparableConv2D(16, (3, 3), padding='same', kernel_regularizer=l2(0.01),
use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(16, (3, 3), padding='same', kernel_regularizer=l2(0.01),
use_bias=False)(x)
x = BatchNormalization()(x)
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])
# Residual Module 2
residual = Conv2D(32, (1, 1), strides=(2, 2), padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)
x = SeparableConv2D(32, (3, 3), padding='same', kernel_regularizer=l2(0.01),
use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(32, (3, 3), padding='same', kernel_regularizer=l2(0.01),
use_bias=False)(x)
x = BatchNormalization()(x)
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])
# Residual Module 3
```

```
residual = Conv2D(64, (1, 1), strides=(2, 2), padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)
x = SeparableConv2D(64, (3, 3), padding='same', kernel_regularizer=l2(0.01),
use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(64, (3, 3), padding='same', kernel_regularizer=l2(0.01),
use_bias=False)(x)

x = BatchNormalization()(x)
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])
# Residual Module 4
residual = Conv2D(128, (1, 1), strides=(2, 2), padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)
x = SeparableConv2D(128, (3, 3), padding='same', kernel_regularizer=l2(0.01),
use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(128, (3, 3), padding='same', kernel_regularizer=l2(0.01),
use_bias=False)(x)
x = BatchNormalization()(x)
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])
#Output Module
x = Conv2D(num_classes, (3, 3), padding='same')(x)
x = GlobalAveragePooling2D()(x)
output = Activation('softmax',name='predictions')(x)
model = Model(img_input, output)
```

```
return model
# Model parameters/compilation
model = myModel(input_shape, num_classes)
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.summary()
datasets = ['fer2013']
for dataset_name in datasets:
print('Training dataset:', dataset_name)
# Callbacks
log_file_path = base_path + dataset_name + '_emotion_training.log'
csv_logger = CSVLogger(log_file_path, append=False)
early_stop = EarlyStopping('val_loss', patience=patience)
reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1, patience=int(patience/4),
verbose=1)
trained_models_path = base_path + dataset_name + '_myModel'

model_names = trained_models_path + '. {epoch:02d}-{val_acc:.2f}. hdf5'
model_checkpoint = ModelCheckpoint(model_names, 'val_loss', verbose=1,
save_best_only=True)
tbCallback= TensorBoard(log_dir='./Graph', histogram_freq=1, write_graph=True,
write_images=False)
callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr, tbCallback]
# Loading dataset
data_loader = DataManager(dataset_name, image_size=input_shape[:2])
faces, emotions = data_loader.get_data()
faces = preprocess_input(faces)
num_samples, num_classes = emotions.shape
train_data, val_data = split_data(faces, emotions, validation_split)
```

```
train_faces, train_emotions = train_data
model.fit_generator(data_generator.flow(train_faces, train_emotions,
batch_size),
steps_per_epoch=len(train_faces) / batch_size,
epochs=num_epochs, verbose=1, callbacks=callbacks,

validation_data=val_data)
```