RECOGNIZATION OF CARDIOVASCULAR DISEASES IN ECG IMAGES USING CNN MODEL.

A PROJECT REPORT

Submitted by

KAMESH.K - 422419104016

KARANRAJ.M - 422419104017

MURALIDHARAN.P - 422419104023

PRAVEENKUMAR.R – 422419104030

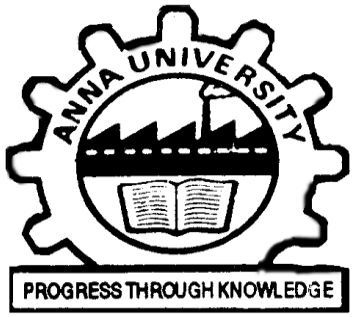
In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



UNIVERSITY COLLEGE OF ENGINEERING TINDIVANAM

ANNA UNIVERSITY: CHENNAI 600 025

JUNE 2022

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**RECOGNIZATION OF CARDIOVASCULAR DISEASES IN ECG IMAGES USING CNN MODEL**" is the bonafide work Of "**KAMESH.K (422419104016),KARANRAJ.M(422419104017),MURALIDHARAN.P (422419104023), PRAVEENKUMAR.R(422419104030)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. L. Jegatha Deborah, M.E., Ph.D,	Ms.Suganya.K, M.E
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Assistant Professor,	Teaching Fellow
Department of CSE,	Department of CSE,
University College of Engineering,	University College of Enginering
Tindivanam,	Tindivanam,
Melpakkam – 604 001.	Melpakkam – 604 001.

Submitted for the University Examination held on _____

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We express our grateful thanks to the **Dean Dr.P. Thamizhazhagan M.E., Ph.D.,** for his constant inspiration and encouragement throughout the project.

For mostly, we pay our grateful acknowledgement and extend our sincere gratitude to **Dr. L. Jegatha Deborah M.E., Ph.D.,** HOD/CSE. We extent our sincere thanks to **Ms.K. Suganya,M.E.** Project Internal Guide for better guidance and constant encouragement in completing this project work successfully.

We like to extend our whole hearty thanks to our Project Coordinator **Mrs.R. Karthika,M.E., Ph.D.,**for her continual support during the entire period schedule.

We thank all our teaching and non-teaching faculty members of the Department and also our fellow friends for helping us in providing valuable suggestions and timely ideas for the successful completion of the project. Last but not the least we extent our thanks to our family members who have been a great source of inspiration and strength to us during the course of this project work. We sincerely thank all of them.

ABSTRACT

Cardiovascular disease (CVD) is the most common class of chronic and life-threatening diseases and, therefore, considered to be one of the main causes of mortality. The proposed new neural architecture based on the recent popularity of convolutional neural networks (CNN) was a solution for the development of automatic heart disease diagnosis systems using electrocardiogram (ECG) signals. The above mentioned proposed CNN model were used as feature extraction tools for traditional machine learning algorithms. A new lightweight deep learning CNN architecture is proposed for cardiovascular diseases prediction using 12 lead-based ECG images. Proposed CNN model were used as feature extractors to apply the extracted features to the conventional machine learning methods. The new approach based on deep learning and, in particular, on a CNN network guaranteed excellent performance in automatic recognition and, therefore, prevention of cardiovascular diseases.

KEYWORDS: Cardiovascular, deep learning, ECG images, feature extraction, machine learning, transfer learning.

திட்டப்பணி சுருக்கம்

இருதய நோய்கள் (இதய நோய்கள்) உலகளவில் மரணத்திற்கு முக்கிய காரணமாகும். முன்னரே அவர்கள் கணித்து வகைப்படுத்தலாம்; அதிக உயிர்களை காப்பாற்ற முடியும். ஆழமான கற்றல் நுட்பங்களின் சக்தி நான்கு முக்கிய இதய அசாதாரணங்களைக் கணிக்கப் பயன்படுத்தப்பட்டது: அசாதாரண இதயத் துடிப்பு, மாரடைப்பு, மாரடைப்பின் வரலாறு மற்றும் இதய நோயாளிகளின் பொது ECG பட தரவுத்தொகுப்பைப் பயன்படுத்தி சாதாரண நபர் வகுப்புகள். ஒரு புதிய கன்வல்யூஷனல் நியூரல் நெட்வொர்க் (CNN) கட்டமைப்பு இதய அசாதாரண கணிப்புக்கு முன்மொழியப்பட்டது. மேற்கூறிய முன் பயிற்சி பெற்ற மாதிரிகள் மற்றும் எங்கள் முன்மொழியப்பட்ட CNN மாதிரி ஆகியவை பாரம்பரிய இயந்திர கற்றல் வழிமுறைகளுக்கான அம்சம் பிரித்தெடுக்கும் கருவிகளாகப் பயன்படுத்தப்பட்டன.

முக்கிய வார்த்தைகள்: கார்டியோவாஸ்குலர், ஆழ்ந்த கற்றல், ஈசிஜி படங்கள், அம்சம் பிரித்தெடுத்தல், இயந்திர கற்றல், பரிமாற்ற கற்றல்.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

For many years, doctors have been aware that cardiovascular diseases constitute a class of diseases considered to be one of the main causes of death. Cardiovascular diseases occur in the form of myocardial infarction (MI). Myocardial infarction, commonly referred to as heart attack, stands for the failure of heart muscles to contract for a fairly long period of time. . Using appropriate treatment within an hour of the start of the heart attack, the mortality risk of the person who suffers from a heart attack in progress can be reduced. When a heart condition occurs, the first diagnostic check consists of an electrocardiogram (ECG), which, therefore, is the main diagnostic tool for cardiovascular disease (CVD). The electrocardiograph detects the electrical activity of the heart during the test time, which is then represented on a graphic diagram that reflects cyclical electrophysiological events in the cardiac muscle [2]. By conducting a careful analysis of the ECG trace, doctors can diagnose a probable myocardial infarction. It is important, however, to underline that the sensitivity and specificity of manual detection of acute myocardial infarction are 91% and 51%, respectively . Developing a computer-aided system to automatically detect MI would help the cardiologists make better decisions. Hence, lately, various studies have been conducted on automatic MI detection. Given the nonlinearity of the heart anomaly classification, techniques based on neural networks have recently been adopted. In a precedent study, the authors proposed a training technique based

CHAPTER 2

PROBLEM STATEMENT

- The accuracy has significantly improved after applying the proposed method as a feature extraction tool for traditional machine learning algorithms.
- The machine learning methods with features selection and outliers' detection achieved accuracy rates as: RF is 80.3%, LR is 83.31%, K-NN is 84.86%, SVM is 83.29%, DT is 82.33%, and XGBoost is 71.4%.

2.2 SCOPE OF THE PROJECT

It has broad applications in many areas, such as

- IoT healthcare environment.
- A tool for clinicians in the medical field to detect cardiac diseases from ECG images.
- Heart- rate monitoring.

2.2 OBJECTIVE

• To perform convolution neural network(CNN) for ECG images and improve accuracy using CNN classification Algorithm to predict four major cardiac abnormalities.

CHAPTER 3

LITERATURE SURVEY

3.1 Title: Personal Heart Health Monitoring Based on 1D Convolutional Neural Network.

Publish Year: 2021

Author: Antonella Nannavecchia, Francesco Girardi , Pio Raffaele Fina , Michele Scalera and Giovanni Dimauro.

A system able to automatically detect the suspect of cardiac pathologies in ECG signals from personal monitoring devices, with the aim to alert the patient to send the ECG to the medical specialist for a correct diagnosis and a proper therapy.

MERITS:

Correct diagnosis can be carried on which specific tests and intervention medical staff.

DEMERITS:

The network showed an accuracy of 89.51% and a recall of 91.09% for normal and 87.79% for anomalous segments.

3.2 Title: A comparative study of classification and prediction of CardioVascular Diseases (CVD) using Machine Learning and Deep Learning techniques .

Publish Year: 2021.

Author: M. Swathy , K. Saruladha.

Cardio-Vascular Diseases (CVD) are found to be rampant in the populace leading to fatal death. The statistics of a recent survey reports that the mortality rate is expanding due to obesity, cholesterol, high blood pressure and usage of tobacco among the people. The results of these predictions will help the clinicians in decision making and early diagnosis, which would reduce the risk of patients becoming fatal.

MERITS:

The various Classification, Data Mining, Machine Learning, Deep Learning models that are used for prediction of the Cardio-Vascular diseases.

DEMERITS:

SVM has a highest accuracy of 90.5% and Logistic Regression has the lowest of 73.9%.

3.3 Title: Automatic ECG Diagnosis Using Convolutional Neural Network.

Publish Year: 2020.

Author: Roberta Avanzato and Francesco Beritelli.

Proposed an automated heart disease recognition technique based on recent and innovative CNN networks. The proposed technique had high accuracy and had low complexity of implementation. This approach harnessed the potential of deep learning to capture the typical characteristics of given heart disease in the ECG signal domain.

MERITS:

Contrasting various methods in the "Discussion" section, we could affirm that the method applied in the present paper yielded considerably better performances than those of the state-of-the-art.

DEMERITS:

The authors claimed an average accuracy of 96.56% and 87.66%for decision tree (DT) and R-peak (RP).

3.4 Title: HeartID: A Multiresolution Convolutional Neural Network for ECG-Based Biometric Human Identification in Smart Health Applications.

Publish Year: 2017.

Author: Qingxue Zhang1, Dian Zhou1 and Xuan Zeng.

A novel multiresolution convolutional neural network for biometric human identification applications. Focusing on existing challenges, we have introduced blind signal processing and automatic feature learning techniques to effectively lower the algorithm engineering effort and also highly enhance the generalization ability of the algorithm .

MERITS:

Evaluate the proposed algorithm extensively on eight ECG datasets and illustrate the advantages over state-of-the-art works.

DEMERITS:

The identification rate for these four abnormal datasets, which is from 86.6% to 93.9%.

3.5 Title: Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks.

Publish Year: 2016 .

Author: Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj, Fellow.

A fast and accurate patientspecific electrocardiogram (ECG) classification and monitoring system. Methods: An adaptive implementation of 1- D convolutional neural networks (CNNs) is inherently used to fuse the two major blocks of the ECG classification into a single learning body .

MERITS:

Due to its simple and parameter invariant nature, the proposed system is highly generic, and, thus, applicable to any ECG dataset.

DEMERITS:

Data can be utilized and organized is a problem yet to be solved.

CHAPTER 4

REQUIREMENT ANALYSIS

Requirements are the basic needs or constraints which are required to develop a system. Requirements are broadly classified as user requirements and the system requirements. These requirements can be collected while designing the system. The two major classifications are software and hardware requirements.

- Functional Requirements
- Non-Functional Requirements

4.1 FUNCTIONAL REQUIREMENTS

Functional requirements specify which output file should be produced from the given file by describing the relationship between the input and output of the system, for each functional requirement a detailed description of all data input and their source and the range of valid input must be specified.

4.1.1 HARDWARE REQUIREMENTS

- Processor - Intel Core
- Speed -10 Ghz
- Ram - 4gb
- Hard Disk - 1000gb
- Key Board - Standard Windows Keyboard
- Mouse - Two Or Three Button

4.1.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 7/8,10 Or Linux
- Coding Language : Python

CHAPTER 5

5.1 SYSTEM ARCHITECTURE

Convolutional neural network is a special kind of artificial neural network developedfor image classification in which the model normally processes two-dimensional spatialinput data representing an image's pixels, in a process called feature learning. The samemodel can be used for one-dimensional sequence of data, such as an analysis of time seriesdata, signal data, or natural language processing. The architecture of the model is describedin. The electrocardiogram signal is a time series data sequence that represents electricalimpulses from the myocardium. Thus, we propose a 1D convolutional neural networkconsisting of four convolutional blocks and one output block for the analysis of ECG signaldata, in order to automatically identify normal and anomalous ECG recordings.The convolutional block, as represented in Figure 3, consists of a 1D convolutionallayer, a batch normalization layer, a 1D max pooling layer, and a rectified linear unit (ReLU)layer, while the output contains a 1D average pooling, a flatten layer, a dense layer, and aSoftmax layer. We chose a four convolutional blocks architecture, since it showed a righttradeoff between computational efficiency and results accuracy.The 1D convolutional layer creates a convolution kernel that is convolved with theinput layer over a single dimension to produce a tensor of output. The kernel size wasset to 80 in the first layer and decreased to 4 in the subsequent layers, in order to reducecomputational costs. The batch normalization standardized the input and it wasapplied after each convolutional layer

and before the level of pooling, in order to improveperformance and stabilize the learning process of the deep neural network. The outputof the batch normalization layer was downsampled by means of a 1D max pooling layerwith a pool size of 4.

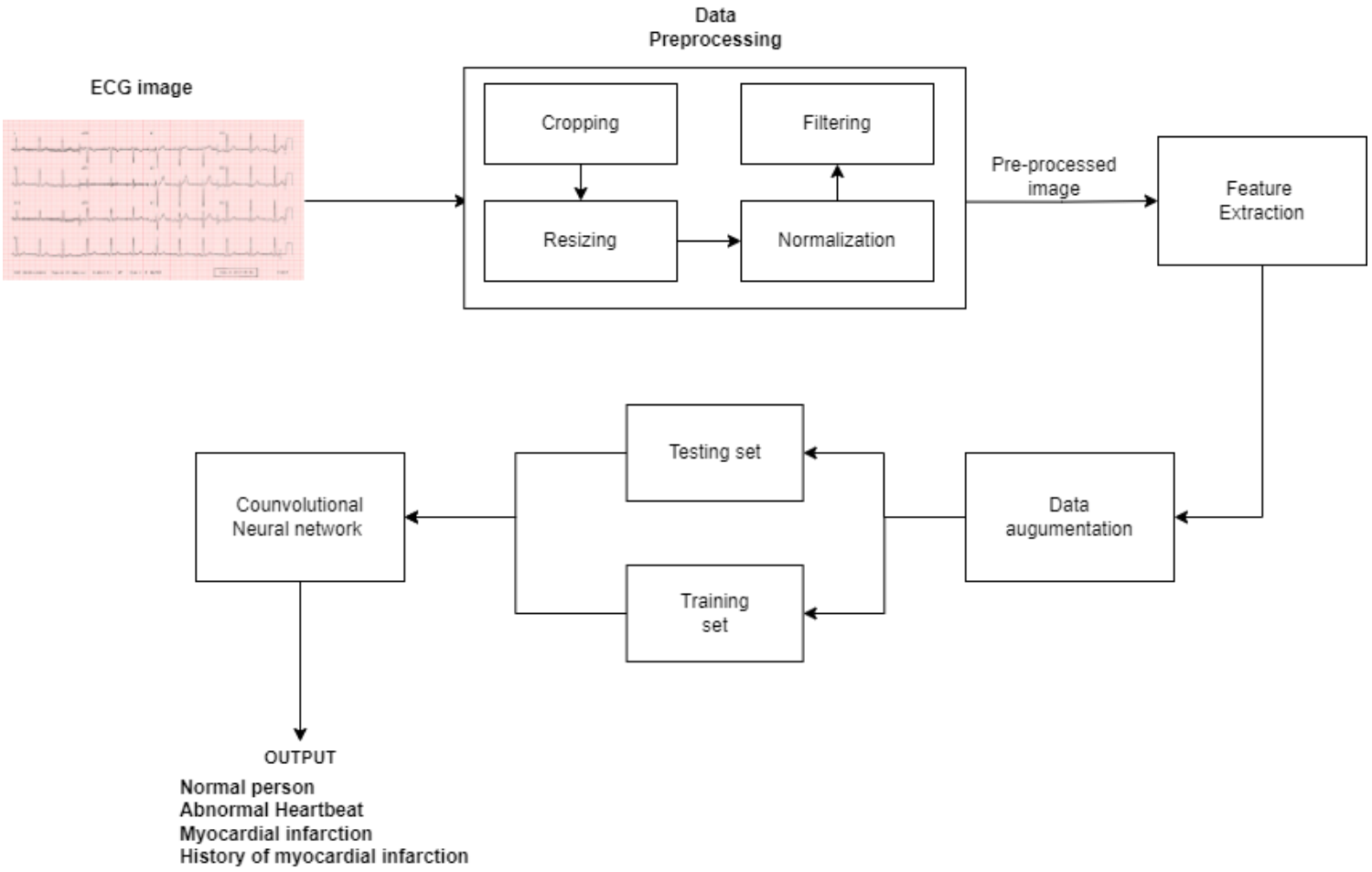


FIGURE 5.1 SYSTEM ARCHITECTURE

5.2 MODULE DESCRIPTION

5.2.1 MODULE 1

5.2.1.1 PREPROCESSING :

The ECG images in the dataset contain a header and a footer information that have no relation to the features we need. Therefore, we have applied cropping for all images to focus on the valuable features Image.

5.2.1.2 RESIZING :

Image interpolation occurs when you resize or distort your image from one pixel grid to another.

5.2.1.3DATA LABELLING :

The process of identifying raw data (images, text files, videos, etc.) and adding one or more meaningful and informative labels to provide context so that a machine learning model can learn from it.

Fundamental Image Processing Steps :

5.2.1.4 IMAGE RESTORATION

Image restoration is the process of improving the appearance of an image. However, unlike image enhancement, image restoration is done using certain mathematical or probabilistic models.

5.2.1.5 COLOR IMAGE PROCESSING

Color image processing includes a number of color modeling techniques in a digital domain. This step has gained prominence due to the significant use of digital images over the internet.

5.2.2 MODULE 2

5.2.2.1DATA AUGMENTATION :

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data.

5.2.2.2 TECHNIQUES :

5.2.2.3 IMAGE ROTATION

One of the most commonly used augmentation techniques is image rotation. Even if you rotate the image, the information on the image remains the same. A cat is a cat, even if you see it from a different angle.

Hence, we can use this technique to increase the size of our training data by creating multiple images rotated at different angles.

5.2.2.4 IMAGE SHIFTING

The next image augmentation technique is image shifting. By shifting the images, we can change the position of the objects in the image and hence give more variety to the model. Which eventually can result in a more generalized model.

Image shift is a geometric transformation that maps the position of every object in the image to the new location of the final output image. So if an object is at position x,y in the original image, it gets shifted to new position X, Y in the new image as shown below. Where dx and dy are the respective shifts along with the different directions.

5.2.2.5 IMAGE FLIPPING

The next technique is Image flipping. Flipping can be considered as an extension of rotation. it allows us to flip the image in the Left-Right direction as well as the Up-Down direction.

The leftmost image here is the original image of the training set and the remaining two images are the flipped images.

5.2.2.6 IMAGE NOISING

Another popularly used image augmentation technique is, Image Noising where we add noise to the image. This technique allows our model to learn how to separate the signal from the noise in the image. This also makes our model more robust to changes in the image.

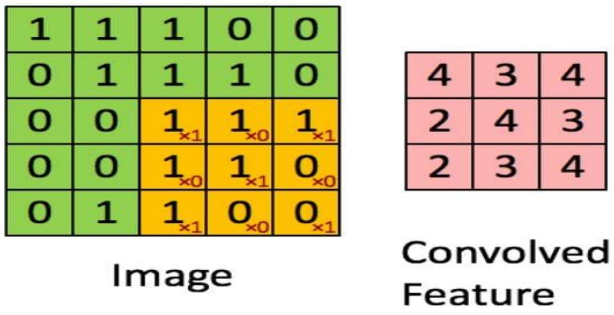
5.2.2.7 IMAGE BLURRING

Let's understand one more augmentation technique, Image blurring. Images come from different sources and hence the quality of images will not be the same from each source. Some images might be of very high quality and others must be very bad. In such scenarios we can blur the original images, this will make our model more robust to the quality of the image being used in the test data.So these are some commonly used image augmentation techniques.

5.2.3 MODULE 3

5.2.3.1 CONVOLUTIONAL LAYER:

A convolution layer transforms the input image in order to extract features from it. In this transformation, the image is convolved with a kernel



5.2.3.2 **POOLING LAYER:**

As identical to the recognized layer "convolutional," the foremost aim of the Pooling layer is essential to decrease the spatial size of the Convolved Feature. So, in short words, it works for decreasing the required computational power for the processing of data by the method of dimensionality reduction. Moreover, it is also beneficial for the extraction of the dominant features, which are basically rotational as well as positional invariant, so the maintenance of the process effectively is needed.

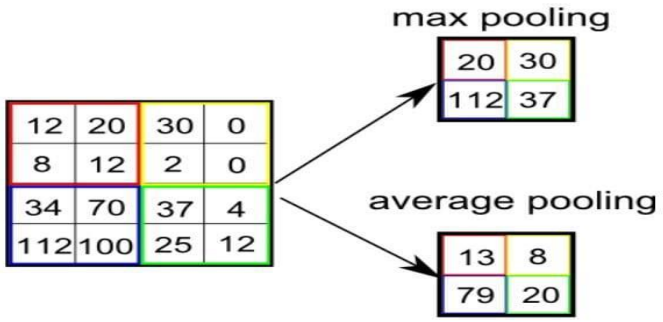
There are mainly two different types of Pooling which are as follows:

5.2.3.3 MAX POOLING:

The Max Pooling basically provides the maximum value within the covered image by the Kernel.

5.2.3.4 Average Pooling:

The Average Pooling provides and returns the average value within the covered image by the Kernel.



The other functionality of Max Pooling is also noise-suppressing, as it works on discarding those activations which contain noisy activation. And on the other side, the Average Pooling simply works on the mechanism of noise-suppressing by dimensionality reduction. So, in short words, we can conclude that Max Pooling works more efficiently than Average Pooling.

The Convolutional Layer, altogether with the Pooling layer, makes the "i-th layer" of the Convolutional Neural Network. Entirely reliant on the image intricacies, the layer counts might be rise-up for the objective of capturing the details of the detailed level, but also needs to have more computational power. After analyzing the above-described information about the process, we can easily execute the model for understanding the features. Moreover, here we are about to get the output and then provide it as an input for the regular Neural Network for further classification reasons.

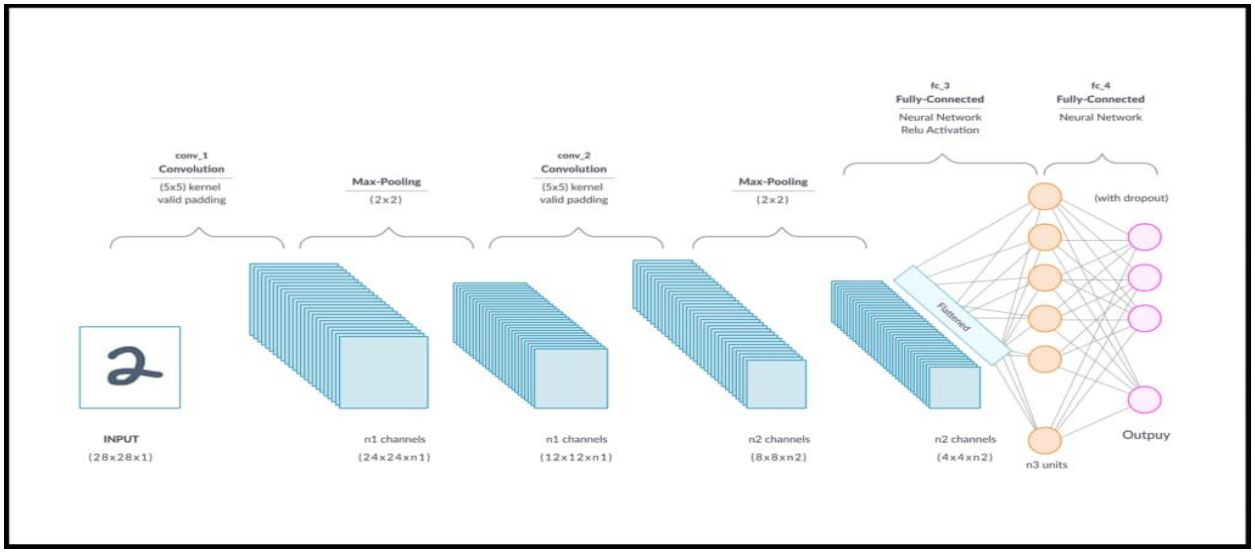
The other functionality of Max Pooling is also noise-suppressing, as it works on discarding those activations which contain noisy activation. And on the other side, the Average Pooling simply works on the mechanism of noise-suppressing by dimensionality reduction. So, in short words, we can conclude that Max Pooling works more efficiently than Average Pooling.

The Convolutional Layer, altogether with the Pooling layer, makes the "i-th layer" of the Convolutional Neural Network. Entirely reliant on the image intricacies, the layer counts might be rise-up for the objective of capturing the details of the detailed level, but also needs to have more computational power. After analyzing the above-described information about the process, we can easily execute the model for understanding the features. Moreover, here we are about to get the

output and then provide it as an input for the regular Neural Network for further classification reasons.

5.2.3.5 **FULLY CONNECTED LAYER:**

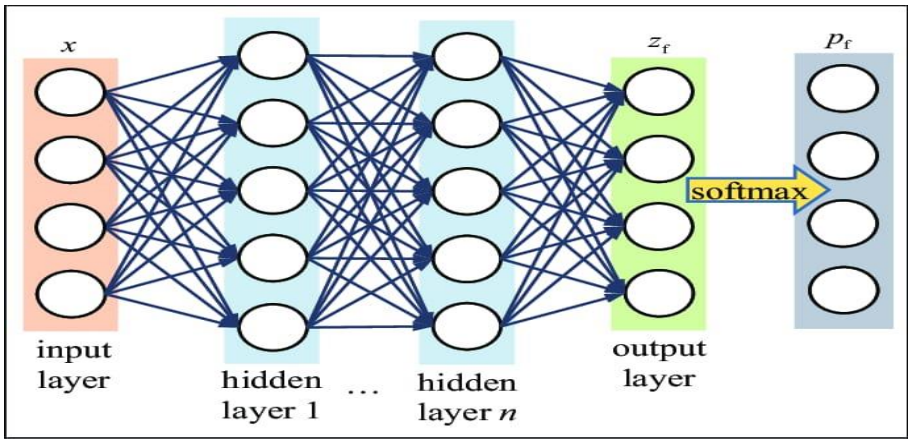
Fully connected layer, hence its name. In our model, the fully connected layer contains 16 neurons. Each neuron in a fully connected layer is connected to each neuron in the previous layer.



5.2.3.6 SOFTMAX LAYER:

Typically the final output layer in a neural network that performs multi-class classification

Softmax extends this idea into a multi-class world. That is, Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise would.


x
z_f
p_f
softmax
input layer
hidden layer 1
...
hidden layer n
output layer

5.3 ALGORITHM :

Convolutional Neural Networks

The CNN model for feature engineering consists of four layers including the embedding layer, 1D convolutional layer, max-pooling layer, and flatten layer. The embedding layer takes 11 features of the CVD dataset with 20,000 vocabulary size and 300 output dimensions with an input length of 11. The embedding layer is followed by the 1D convolutional layer with 5000 filters, 2 × 2 kernel size, and the rectifier linear unit (ReLU) activation function. To map the important features from 1D convolution output, a max-pooling layer with a 2 × 2 pool size is used.

Suppose that the CVD dataset X consists of a tuple set (f si , tci), where i is the index of the tuple, f s is the feature set, and tc is the target class column; then, the embedding layer can be used to convert the training set into the required input format as

$$EL = embedding\_layer(Vs, Os, I)$$

$$EOs = EL(f\ s) \quad (1)$$

The third parameter is the input length I, which represents the number of features in the original dataset which are 11. The embedding layer processes the input data and generates the output for further processing by the CNN model. The output dimensions of the embedding layer are EOs = (None, 11, 300):

$$1D - Convs = CNN(F, Ks, AF) \leftarrow Eos \quad (2)$$

The $1D - Convs$ is extracted from the output of the embedding layer. We used 5000 filters, i.e., F = 5000, in CNN layers and a kernel size of Ks = 2 × 2. A rectified linear unit (ReLU) activation function AF is used to set all negative values to zero in the $1D - Convs$ output matrix while others remain

constant:

$$f(x) = \max(0, E)s \quad (3)$$

The max-pooling layer is used to map the significant features from the CNN. A pool size of 2 × 2 is used for the feature map. Here, Fmap are features after max-pooling, Ps = 2 is the pooling window size, and S − 2 is the stride:

$$Cf = Fmap = [I - Ps]\, S \,\% + 1 \quad (4)$$

In the end, a Flatten layer is used to convert 3D data into 1D because machine learning models work on 1D data. In this way, a total of 25,000 features are obtained to train machine learning models. The below equation shows the features for the training of learning models:

CHAPTER 6

UML DIAGRAMS

6.1 USE CASE DIAGRAMS

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. A use case diagram at its simplest is a representation of a user‟s interaction with the system that show the relationship between the user andthe different use cases in which the user is involved. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It is represented using ellipse or circles. Use case diagram can be a good communication to stakeholders.

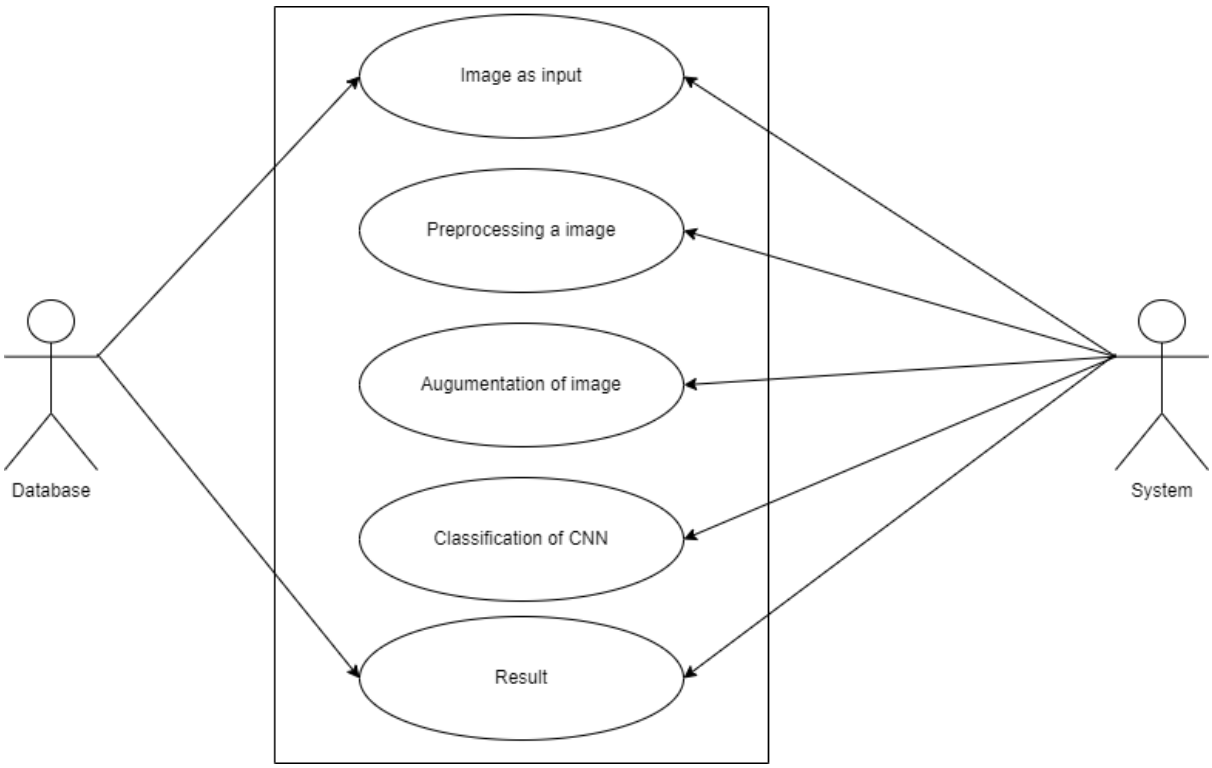


FIGURE 6.1 USE CASE DIAGRA

6.2 SEQUENCE DIAGRAM

In sequence diagram step by step sequence of steps is shown. In above diagram first preprocess all train data and test data. Then by applying the train data Train the machine and build the module and at the last apply machine learning algorithm on it. For testing purpose apply the test data on module and see the classification either fake or real.

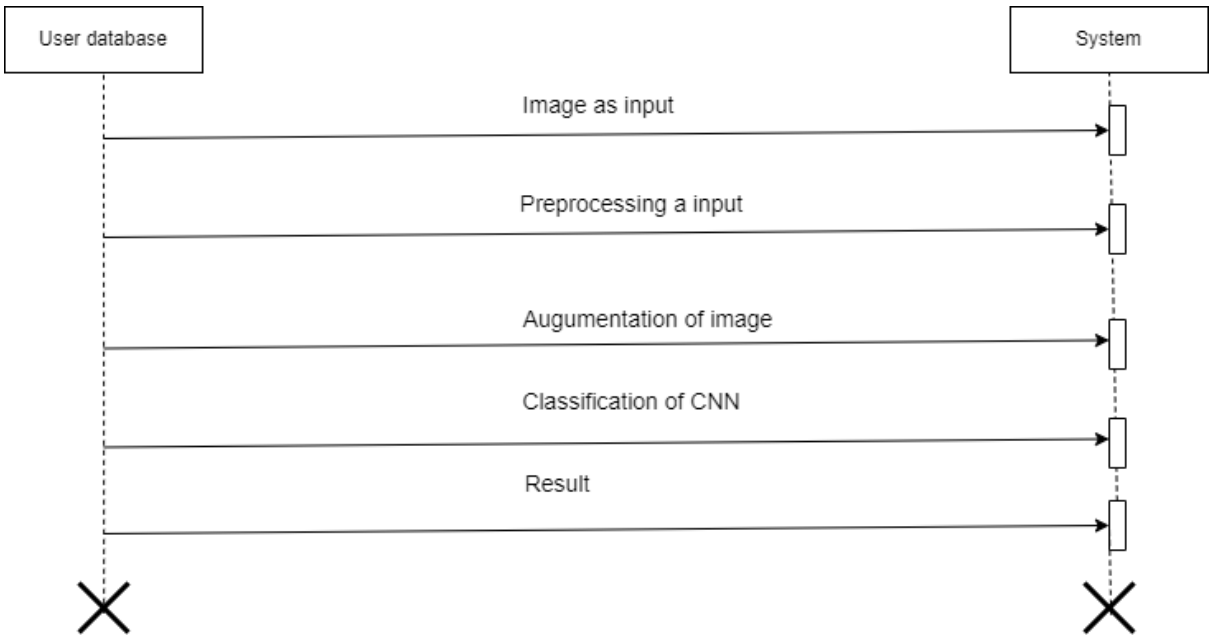


FIGURE 6.2 SEQUENCE DIAGRAM

6.3 COLLABORATIVE DIAGRAMS

A collaboration diagram, also called a communication diagram or interaction diagram. A sophisticated modelling tool can easily convert a collaboration diagram into a sequence diagram and vice versa. A collaboration diagram resembles a flowchart that portrays the roles, functionality and behaviour of individual objects as well as the overall operation of the system in real time.

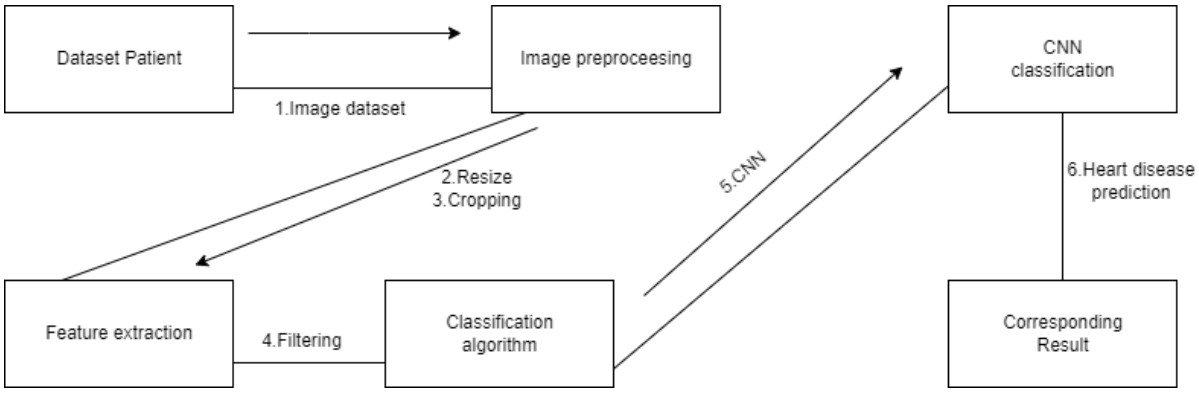


FIGURE 6.3COLLABORATIVE DIAGRAM

6.4 COMPONENT DIAGRAM

A component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems. They are represented using rectangles. A component diagram displays the structural relationship components of the software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. In our project, there are four components which interact with the main system component using interfaces.

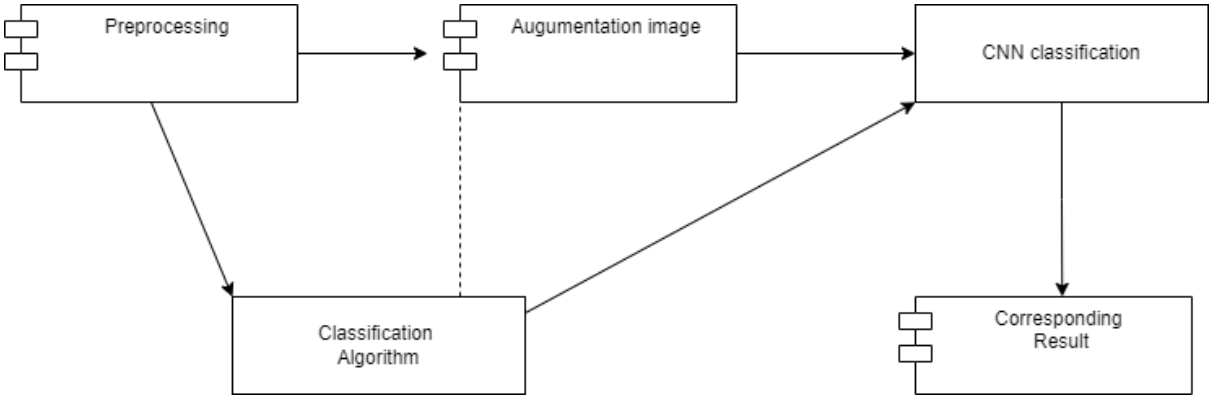


FIGURE 6.4 COMPONENT DIAGRAM

6.5 DEPLOYMENT DIAGRAM

Deployment diagram is a structure diagram which shows the execution architecture of the system, includes nodes such as deployment of software artifacts or software execution to deployment targets and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of the system. Deployment diagrams help model the hardware topology of the system compared to other UML diagram types which mostly outline the logical components of a system.

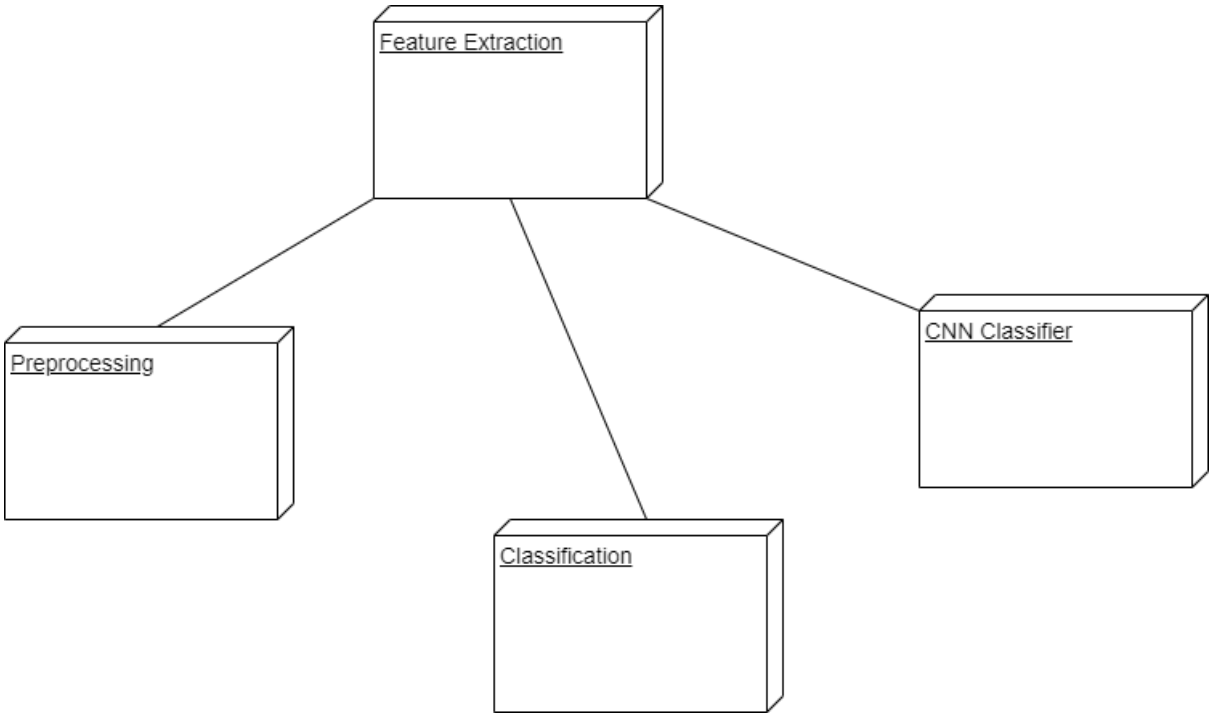


FIGURE 6.5 DEPLOYMENT DIAGRAM

6.6 STATE CHART DIAGRAM

State chart diagram model the dynamic behaviour of individual classes or may other kind of object. They show the sequences of states that an object goes through, the events that cause transition from one state to another. It resembles a flowchart in which the initial state is represented by a large black dot and subsequent states are portrayed as boxes with rounded corners. There may be one or two horizontal lines through a box, dividing it into stacked sections. In that case, the upper section contains the name of the state, the middle section contains the state variables and the lower section contains the action performed in that state. If there are no horizontal lines through a box, only the name of the state is written inside it. External straight lines, each with an arrow at one end, connect various pairs of boxes. These lines define the transitions between states. The final state is portrayed as a large black dot with a circle around it.

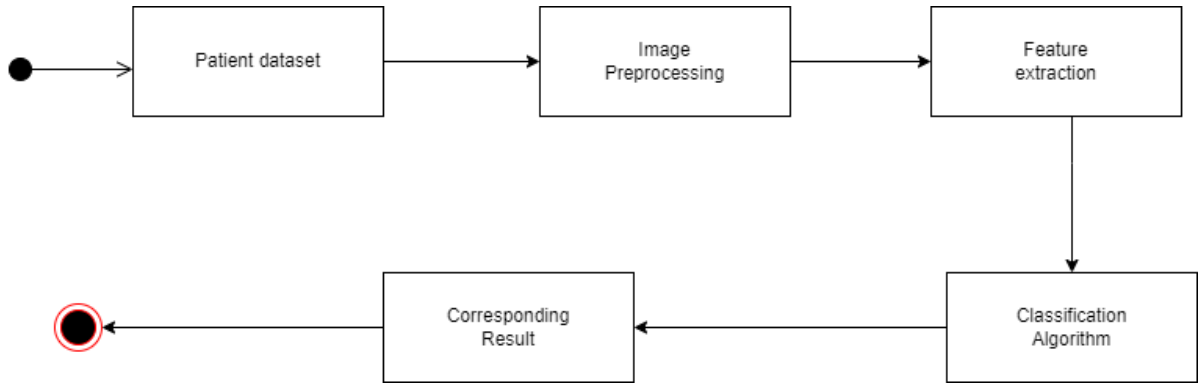


FIGURE 6.6 STATE CHART DIAGRAM

6.7 ACTIVITY DIAGRAM

Activity Diagram shows the active flow of the system. In above diagram theflow of our project is shown actually how the data flow

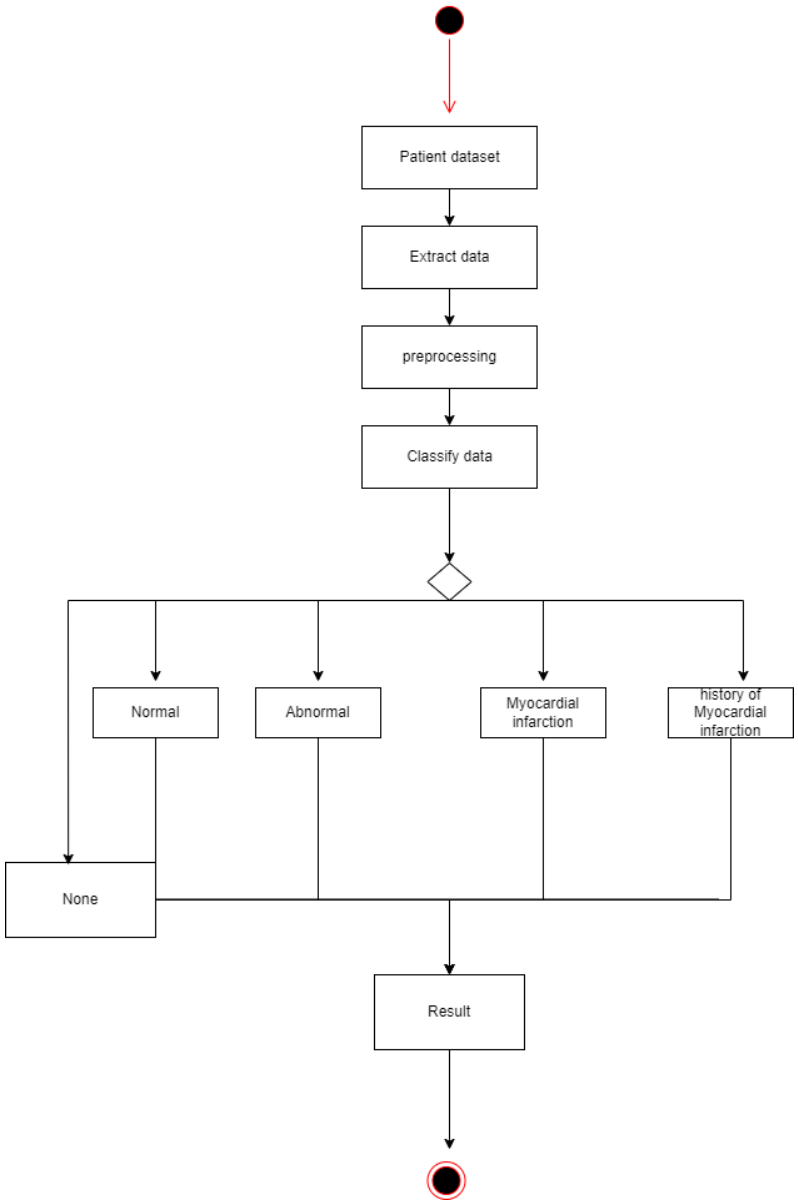


FIGURE 6.7 ACTIVITY DIAGRAM

6.8 PACKAGE DIAGRAM

Package diagram is UML structure diagram which shows packages and dependencies between the packages. A package is a UML construct that enables you to organize model elements, such as use cases or classes, into groups. Packages can be depicted as file folders and can be applied on any UML diagram. Initially, the images are taken from the user.

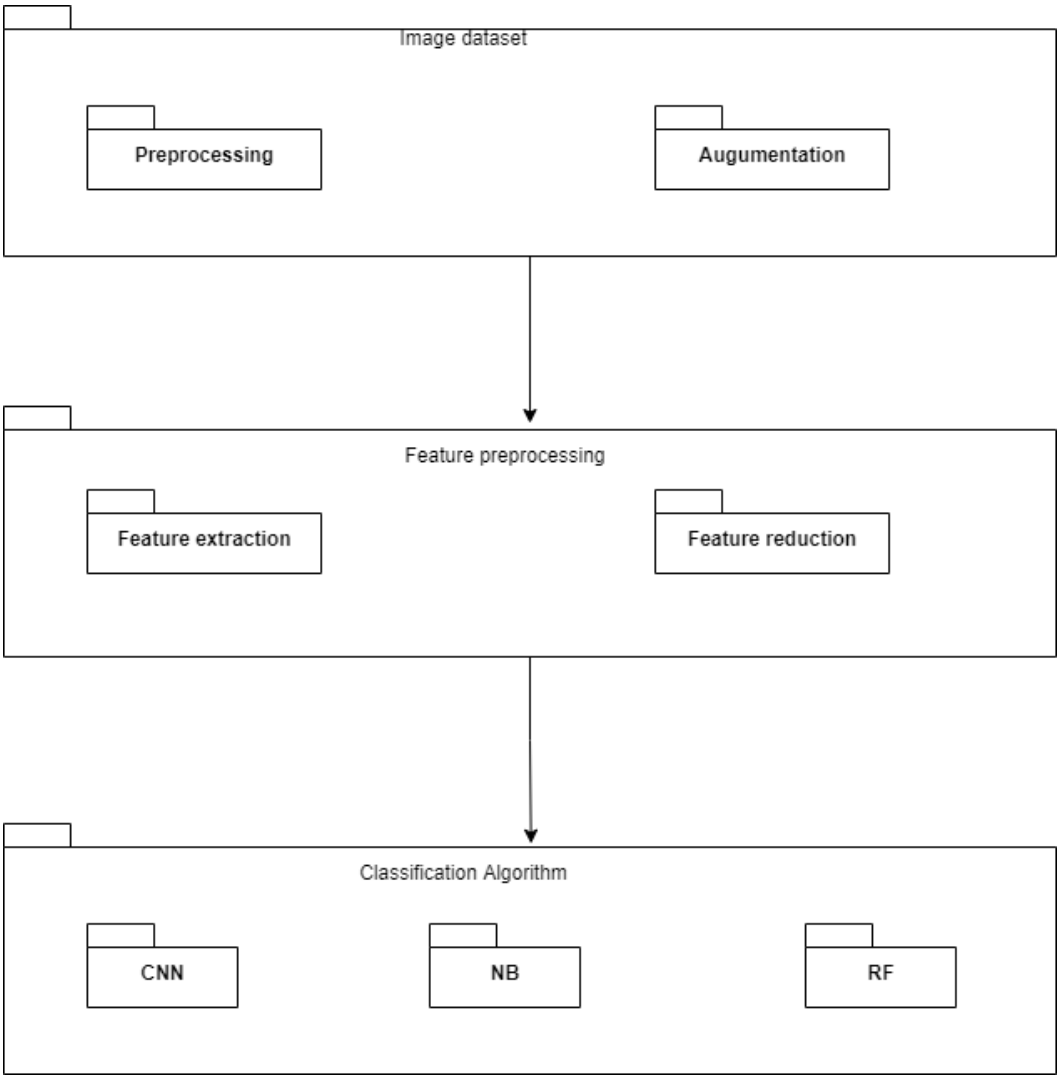


FIGURE 6.8 PACKAGE DIAGRAM

6.9 DATA FLOW DIAGRAM:

A data flow diagram (DFD) is a graphical representation of the „flow" of a data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data flow diagram can also be used for the visualization of data processing . DFD is designed to show how a system is divided onto smaller portions and to highlight the flow of data between those parts. DFD is an important technique for modeling a systems high level detail by showing how input data is transformed to output results through a sequence of functional transformations. DFD consists of four major components: entities, processes, data stores and dataflow. A context diagram is a top level (also known as "Level 0") data flow diagram. It only contains one process node ("Process 0") that generalizes the function of the entire system in relationship to external entities. DFD diagrams can be made in several nested layers. It is the mostbasic form of DFD. There is only one process in the system and all the data flows either into or out of this process. Context level DFD demonstrates the interactions between the process and external entities. They do not contain data stores. When drawing context level DFD we must first identify the process, all the eternal entities and all the dataflow**.**

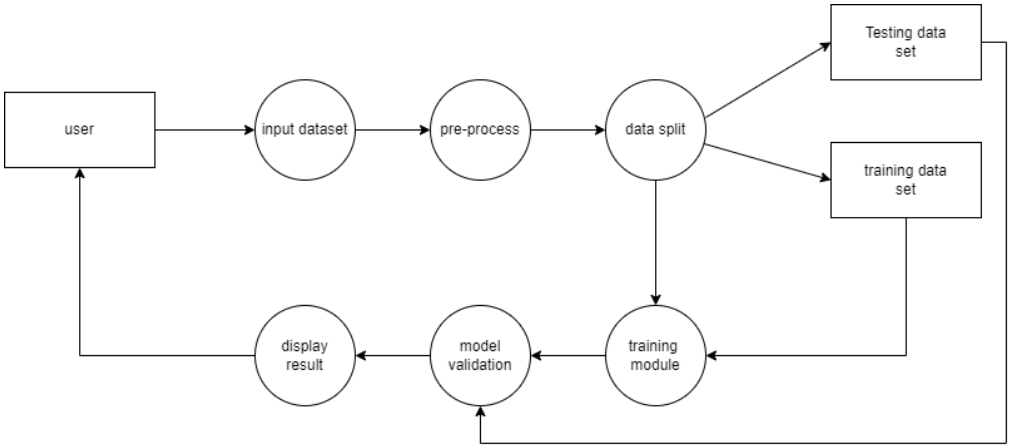


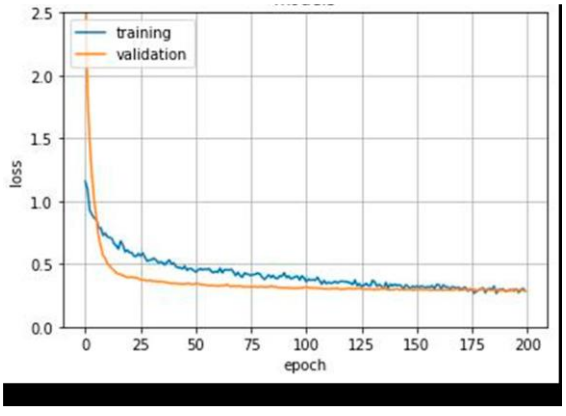
FIGURE 6.9 OVERALL DATA FLOW DIAGRAM

CHAPTER 7

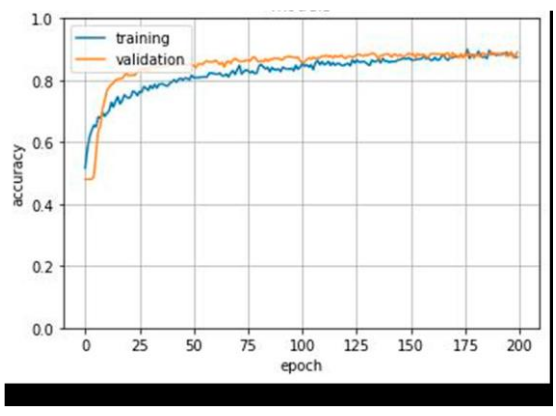
IMPLEMENTATION:

id		Age	gender	height	weight	ap_hi	ap_lo	cholesterol	Gluc	smoke		alco	active	cardio	
0		18393	2	168	62	110	80	1	1	0		0	1	0	
1		20228	1	156	85	140	90	3	1	0		0	1	1	
2		18857	1	165	64	130	70	3	1	0		0	0	1	
3		17623	2	169	82	150	100	1	1	0		0	1	1	
4		17474	1	156	56	100	60	1	1	0		0	0	0	
8		21914	1	151	67	120	80	2	2	0		0	0	0	
9		22113	1	157	93	130	80	3	1	0		0	1	0	
12		22584	2	178	95	130	90	3	3	0		0	1	1	
13		17668	1	158	71	110	70	1	1	0		0	1	0	
14		19834	1	164	68	110	60	1	1	0		0	0	0	
15		22530	1	169	80	120	80	1	1	0		0	1	0	
16		18815	2	173	60	120	80	1	1	0		0	1	0	
18		14791	2	165	60	120	80	1	1	0		0	0	0	
21		19809	1	158	78	110	70	1	1	0		0	1	0	

FIGURE 7. DATABASE OF PATIENTS



(a) Training and Validation loss



(b) Training and Validation Accuracy

CHAPTER 8

8.1 CONCLUSION

We proposed a lightweight CNN-based model to classify the four major cardiac abnormalities: abnormal heartbeat, myocardial infarction, history of myocardial infarction, and normal person classes using public ECG images dataset of cardiac patients. According to the results of the experiments, the proposed CNN model achieves remarkable results in cardiovascular disease classification and can also be used as a feature extraction tool for the traditional machine learning classifiers. Thus, the proposed CNN model can be used as an assistance tool for clinicians in the medical field to detect cardiac diseases from ECG images and bypass the manual process that leads to inaccurate and time-consuming results.

8.2 FUTURE WORK

Optimization techniques can be used to obtain optimized values for the hyperparameters of the proposed CNN model. The proposed model can also be used for predicting other types of problems. Since, the proposed model belongs to the family of low-scale deep learning methods in terms of the number of layers, parameters, and depth. Therefore, a study on using the proposed model in the Industrial Internet of Things (IIoT) domain for classification purposes can be explored.

REFERENCE:

[1]. Nannavecchia, A.; Girardi, F.; Fina, P.R.; Scalera, M.; Dimauro, G. Personal Heart Health Monitoring Based on 1D Convolutional Neural Network. J. Imaging 2021, 7, 26. https://doi.org/10.3390/ jimaging7020026 .

[2]. S. Kiranyaz, T. Ince and M. Gabbouj, "Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks," IEEE Transactions on Biomedical Engineering, vol. 63, no. 3, pp. 664-675, 2016. **https://doi.org/10.1109/TBME.2015.2468589**.

[3]. Q. Zhang, D. Zhou and X. Zeng, "HeartID: A Multiresolution Convolutional Neural Network for ECG-Based Biometric Human Identification in Smart Health Applications," IEEE Access, vol. 5, pp. 11805-11816, 2017. **https://doi.org/10.1109/ACCESS.2017.2707460**..

[4]. R. Avanzato and F. Beritelli, "Automatic ECG diagnosis using convolutional neural network," Electronics, vol. 9, no. 6, p. 951, 2020. **https://doi.org/10.3390/electronics9060951**..

[5]. M. Swathy and K. Saruladha, "A comparative study of classification and prediction of Cardio-Vascular Diseases (CVD) using Machine Learning and Deep Learning techniques," ICT Express, 2021. **https://doi.org/10.1016/j.icte.2021.08.021**.

[6] A. Rath, D. Mishra, G. Panda and S. C. Satapathy, "Heart disease detection using deep learning methods from imbalanced ECG samples," Biomedical Signal Processing and Control, vol. 68, no. 102820, 2021. **https://doi.org/10.1016/j.bspc.2021.102820**.

[7] A. Mincholé and B. Rodriguez, "Artificial intelligence for the electrocardiogram," Nature Medicine, vol. 25, no. 1, pp. 22-23, 2019. https://doi.org/10.1038/s41591-018-0306-1.

[8] A. Isin and S. Ozdalili, "Cardiac arrhythmia detection using deep learning," Procedia Computer Science, vol. 120, pp. 268-275, 2017. **https://doi.org/10.1016/j.procs.2017.11.238**.

[9] H. Bleijendaal, L. A. Ramos, R. R. Lopes, T. E. Verstraelen, S. W. E. Baalman, M. D. Oudkerk Pool, F. V. Y. Tjong, F. M. MelgarejoMeseguer, J. Gimeno-Blanes, J. R. Gimeno-Blanes, A. S. Amin, M. M. Winter, H. A. Marquering, W. E. M. Kok, A. H. Zwinderman, A. A. M. Wilde and Y. M. Pinto, "Computer versus Cardiologist: Is a machine learning algorithm able to outperform an expert in diagnosing phospholamban (PLN) p.Arg14del mutation on ECG?," Heart rhythm., vol. 18, no. 1, pp. 79-87, 2020. https://doi.org/10.1016/j.hrthm.2020.08.021.

[10] U. R. Acharya, H. Fujita, O. S. Lih, M. Adam, J. H. Tan and C. K. Chua, "Automated detection of coronary artery disease using different durations of ECG segments with convolutional neural network," Knowledge-Based Systems, vol. 132, pp. 62-71, 2017. https://doi.org/10.1016/j.knosys.2017.06.003.

[11] M. Kantardzic, Data Mining: Concepts, Models, Methods, and Algorithms, 3 ed., John Wiley & Sons, Inc., 2020.

[12] S. García, J. Luengo and F. Herrera, Data Preprocessing in Data Mining, 1 ed., Springer, 2015. [13] G. Dougherty, Pattern Recognition and Classification: An Introduction, Springer, 2013.

[14] A. Subasi, Practical Machine Learning for Data Analysis Using Python, Academic Press, 2020.

[15] J. Soni, U. Ansari, D. Sharma and S. Soni, "Predictive data mining for medical diagnosis: An overview of heart disease prediction," International Journal of Computer Applications, vol. 17, no. 8, pp. 43-48, 2011.

APPENDIX :


```
{
 "nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "name": "Heart Disease Detection Using Python and Machine Learning.ipynb",
   "provenance": [],
   "collapsed_sections": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
  }
 },
 "cells": [
  {
   "cell_type": "code",
   "metadata": {
    "id": "qxm3ljVHD73Q"
   },
   "source": [
    "#This program classifies the person as having a cardiovuscular disease or not"
   ],
   "execution_count": null,
   "outputs": []
```

```
},
{
 "cell_type": "code",
 "metadata": {
  "id": "4o7OcdHACu1u",
  "outputId": "acc90fc1-b311-47d0-b573-d7367a6702b5",
  "colab": {
   "base_uri": "https://localhost:8080/",
   "height": 71
  }
 },
 "source": [
  "import pandas as pd\n",
  "import numpy as np\n",
  "import seaborn as sns"
 ],
 "execution_count": null,
 "outputs": [
  {
   "output_type": "stream",
   "text": [
    "/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.\n",
    "  import pandas.util.testing as tm\n"
   ],
   "name": "stderr"
  }
 ]
```

```
},
{
 "cell_type": "code",
 "metadata": {
  "id": "ekMD_XETMNuq",
  "outputId": "4e01b8e2-228a-46b3-f2ef-b1a49dfad0c2",
  "colab": {
   "resources": {
    "ok": true,
    "headers": [
     [
      "content-type",
      "application/javascript"
     ]
    ],
    "status": 200,
    "status_text": ""
   }
  },
  "base_uri": "https://localhost:8080/",
  "height": 72
 }
},
"source": [
 "from google.colab import files\n",
 "uploaded=files.upload()"
],
"execution_count": null,
```

```
 "outputs": [
  {
   "output_type": "display_data",
   "data": {
    "text/html": [
     "\n",
     "      <input type=\"file\" id=\"files-3ae72c81-ebd5-4b7c-b2ae-4f06f2a18d45\" name=\"files[]\" multiple disabled\n",
     "      style=\"border:none\" />\n",
     "      <output id=\"result-3ae72c81-ebd5-4b7c-b2ae-4f06f2a18d45\">\n",
     "      Upload widget is only available when the cell has been executed in the\n",
     "      current browser session. Please rerun this cell to enable.\n",
     "      </output>\n",
     "      <script src=\"/nbextensions/google.colab/files.js\"></script> "
    ],
    "text/plain": [
     "<IPython.core.display.HTML object>"
    ]
   },
   "metadata": {
    "tags": []
   }
  },
  {
   "output_type": "stream",
   "text": [
    "Saving cardio_train1.xlsx to cardio_train1.xlsx\n"
```

```
      ],
      "name": "stdout"
    }
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "cT93hzV4MTh1",
    "outputId": "5e6fb0f3-7848-4cb9-d76f-2ec640c36634",
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 204
    }
  },
  "source": [
    "df=pd.read_excel('cardio_train1.xlsx')\n",
    "df.head()"
  ],
  "execution_count": null,
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/html": [
          "<div>\n",
          "<style scoped>\n",
          "    .dataframe tbody tr th:only-of-type {\n",
```

```
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>id</th>\n",
"      <th>age</th>\n",
"      <th>gender</th>\n",
"      <th>height</th>\n",
"      <th>weight</th>\n",
"      <th>ap_hi</th>\n",
"      <th>ap_lo</th>\n",
"      <th>cholesterol</th>\n",
"      <th>gluc</th>\n",
"      <th>smoke</th>\n",
"      <th>alco</th>\n",
"      <th>active</th>\n",
"      <th>cardio</th>\n",
```

```
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>0</td>\n",
"      <td>18393</td>\n",
"      <td>2</td>\n",
"      <td>168</td>\n",
"      <td>62.0</td>\n",
"      <td>110</td>\n",
"      <td>80</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>1</td>\n",
"      <td>20228</td>\n",
"      <td>1</td>\n",
"      <td>156</td>\n",
"      <td>85.0</td>\n",
"      <td>140</td>\n",
"      <td>90</td>\n",
```

```
"      <td>3</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>2</td>\n",
"      <td>18857</td>\n",
"      <td>1</td>\n",
"      <td>165</td>\n",
"      <td>64.0</td>\n",
"      <td>130</td>\n",
"      <td>70</td>\n",
"      <td>3</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>3</td>\n",
"      <td>17623</td>\n",
"      <td>2</td>\n",
```

```
          "</table>\n",
          "</div>"
         ],
         "text/plain": [
          "    id    age  gender  height  weight  ...  gluc  smoke  alco  active  cardio\n",
          "0   0  18393       2     168    62.0  ...     1      0     0       1       0\n",
          "1   1  20228       1     156    85.0  ...     1      0     0       1       1\n",
          "2   2  18857       1     165    64.0  ...     1      0     0       0       1\n",
          "3   3  17623       2     169    82.0  ...     1      0     0       1       1\n",
          "4   4  17474       1     156    56.0  ...     1      0     0       0       0\n",
          "\n",
          "[5 rows x 13 columns]"
         ]
        },
        "metadata": {
         "tags": []
        },
        "execution_count": 15
       }
      ]
     },
     {
      "cell_type": "code",
      "metadata": {
       "id": "MvWXdTlrMlEt",
       "outputId": "bf928ccc-7ca1-493e-d108-29489e78d146",
       "colab": {
        "base_uri": "https://localhost:8080/",
```

```
      "height": 34
     }
    },
    "source": [
     "df.shape"
    ],
    "execution_count": null,
    "outputs": [
     {
      "output_type": "execute_result",
      "data": {
       "text/plain": [
        "(70000, 13)"
       ]
      },
      "metadata": {
       "tags": []
      },
      "execution_count": 16
     }
    ]
   },
   {
    "cell_type": "code",
    "metadata": {
     "id": "tvTVJNyZNkNu",
     "outputId": "128fd231-f551-4120-f52d-fc0562fcb24e",
     "colab": {
```

```
    "base_uri": "https://localhost:8080/",
    "height": 255
  }
},
"source": [
  "#Count the emply or null values in each column\n",
  "df.isna().sum()"
],
"execution_count": null,
"outputs": [
  {
    "output_type": "execute_result",
    "data": {
      "text/plain": [
        "id             0\n",
        "age            0\n",
        "gender         0\n",
        "height         0\n",
        "weight         0\n",
        "ap_hi          0\n",
        "ap_lo          0\n",
        "cholesterol    0\n",
        "gluc           0\n",
        "smoke          0\n",
        "alco           0\n",
        "active         0\n",
        "cardio         0\n",
        "dtype: int64"
```

```
        ]
       },
       "metadata": {
         "tags": []
       },
       "execution_count": 17
     }
   ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "vq8QIrUBaAzC",
    "outputId": "fa6a9fba-b7b9-4be5-c3ec-5a6dd0118e6d",
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 34
    }
  },
  "source": [
    "#Another way to chech for null or missing values\n",
    "df.isnull().values.any()"
  ],
  "execution_count": null,
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
```

```
        "text/plain": [
          "False"
        ]
      },
      "metadata": {
        "tags": []
      },
      "execution_count": 20
    }
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "7hIsI99JaLu3",
    "outputId": "cbab973e-5da8-4c86-d268-a21d5bb3f77e",
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 317
    }
  },
  "source": [
    "#View some basic statistics\n",
    "df.describe()"
  ],
  "execution_count": null,
  "outputs": [
    {
```

```
"output_type": "execute_result",
"data": {
  "text/html": [
    "<div>\n",
    "<style scoped>\n",
    "    .dataframe tbody tr th:only-of-type {\n",
    "        vertical-align: middle;\n",
    "    }\n",
    "\n",
    "    .dataframe tbody tr th {\n",
    "        vertical-align: top;\n",
    "    }\n",
    "\n",
    "    .dataframe thead th {\n",
    "        text-align: right;\n",
    "    }\n",
    "</style>\n",
    "<table border=\"1\" class=\"dataframe\">\n",
    "  <thead>\n",
    "    <tr style=\"text-align: right;\">\n",
    "      <th></th>\n",
    "      <th>id</th>\n",
    "      <th>age</th>\n",
    "      <th>gender</th>\n",
    "      <th>height</th>\n",
    "      <th>weight</th>\n",
    "      <th>ap_hi</th>\n",
    "      <th>ap_lo</th>\n",
```

```
"      <th>cholesterol</th>\n",
"      <th>gluc</th>\n",
"      <th>smoke</th>\n",
"      <th>alco</th>\n",
"      <th>active</th>\n",
"      <th>cardio</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>count</th>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"      <td>70000.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>mean</th>\n",
"      <td>49972.419900</td>\n",
```

```
"      <td>19468.865814</td>\n",
"      <td>1.349571</td>\n",
"      <td>164.359229</td>\n",
"      <td>74.205690</td>\n",
"      <td>128.817286</td>\n",
"      <td>96.630414</td>\n",
"      <td>1.366871</td>\n",
"      <td>1.226457</td>\n",
"      <td>0.088129</td>\n",
"      <td>0.053771</td>\n",
"      <td>0.803729</td>\n",
"      <td>0.499700</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>std</th>\n",
"      <td>28851.302323</td>\n",
"      <td>2467.251667</td>\n",
"      <td>0.476838</td>\n",
"      <td>8.210126</td>\n",
"      <td>14.395757</td>\n",
"      <td>154.011419</td>\n",
"      <td>188.472530</td>\n",
"      <td>0.680250</td>\n",
"      <td>0.572270</td>\n",
"      <td>0.283484</td>\n",
"      <td>0.225568</td>\n",
"      <td>0.397179</td>\n",
"      <td>0.500003</td>\n",
```

```
        ],
        "text/plain": [
          "              id          age  ...       active       cardio\n",
          "count   70000.000000  70000.000000  ...  70000.000000  70000.000000\n",
          "mean    49972.419900  19468.865814  ...      0.803729      0.499700\n",
          "std     28851.302323   2467.251667  ...      0.397179      0.500003\n",
          "min         0.000000  10798.000000  ...      0.000000      0.000000\n",
          "25%     25006.750000  17664.000000  ...      1.000000      0.000000\n",
          "50%     50001.500000  19703.000000  ...      1.000000      0.000000\n",
          "75%     74889.250000  21327.000000  ...      1.000000      1.000000\n",
          "max     99999.000000  23713.000000  ...      1.000000      1.000000\n",
          "\n",
          "[8 rows x 13 columns]"
        ]
      },
      "metadata": {
        "tags": []
      },
      "execution_count": 21
    }
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "60-Uy-vVaXmp",
    "outputId": "0c56035a-4ad9-4e8b-cdaf-51d59daf0363",
    "colab": {
```

```
      "base_uri": "https://localhost:8080/",
      "height": 68
     }
    },
    "source": [
     "#Get a count of the number of patients with cardiovuscular disease or without\n",
     "df['cardio'].value_counts()"
    ],
    "execution_count": null,
    "outputs": [
     {
      "output_type": "execute_result",
      "data": {
       "text/plain": [
        "0    35021\n",
        "1    34979\n",
        "Name: cardio, dtype: int64"
       ]
      },
      "metadata": {
       "tags": []
      },
      "execution_count": 23
     }
    ]
   },
   {
    "cell_type": "code",
```

```
"metadata": {
  "id": "wpAPqStta5-F",
  "outputId": "6c928a6d-0be5-42fb-f045-51fe764ec7d6",
  "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 296
  }
},
"source": [
  "#Visualize the count\n",
  "sns.countplot(df['cardio'])"
],
"execution_count": null,
"outputs": [
  {
    "output_type": "execute_result",
    "data": {
      "text/plain": [
        "<matplotlib.axes._subplots.AxesSubplot at 0x7fd298efbb38>"
      ]
    },
    "metadata": {
      "tags": []
    },
    "execution_count": 24
  },
  {
    "output_type": "display_data",
```

```
   "data": {
  "\n",
  "#Visualizing the data\n",
  "sns.countplot(x='years', hue='cardio', data=df, palette='colorblind', edgecolor=sns.color_palette('dark', n_colors=1))"
 ],
 "execution_count": null,
 "outputs": [
  {
   "output_type": "execute_result",
   "data": {
    "text/plain": [
     "<matplotlib.axes._subplots.AxesSubplot at 0x7fd296aa6da0>"
    ]
   },
   "metadata": {
    "tags": []
   },
   "execution_count": 25
  },
  {
   "output_type": "display_data",
   "data": {
    "image/png":
    "text/plain": [
     "<Figure size 432x288 with 1 Axes>"
    ]
   },
   "metadata": {
```

```
        "tags": [],
        "needs_background": "light"
      }
    }
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "7QOHIaNVcPvL",
    "outputId": "c9ce5495-5db3-4c97-935b-50772905d59a",
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 483
    }
  },
  "source": [
    "#Get correlation of the columns\n",
    "df.corr()"
  ],
  "execution_count": null,
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/html": [
          "<div>\n",
          "<style scoped>\n",
```

```
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>id</th>\n",
"      <th>age</th>\n",
"      <th>gender</th>\n",
"      <th>height</th>\n",
"      <th>weight</th>\n",
"      <th>ap_hi</th>\n",
"      <th>ap_lo</th>\n",
"      <th>cholesterol</th>\n",
"      <th>gluc</th>\n",
"      <th>smoke</th>\n",
"      <th>alco</th>\n",
"      <th>active</th>\n",
```

```
"      <th>cardio</th>\n",
"      <th>years</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>gluc</th>\n",
"      <td>0.002467</td>\n",
"      <td>0.098703</td>\n",
"      <td>-0.020491</td>\n",
"      <td>-0.018595</td>\n",
"      <td>0.106857</td>\n",
"      <td>0.011841</td>\n",
"      <td>0.010806</td>\n",
"      <td>0.451578</td>\n",
"      <td>1.000000</td>\n",
"      <td>-0.004756</td>\n",
"      <td>0.011246</td>\n",
"      <td>-0.006770</td>\n",
"      <td>0.089307</td>\n",
"      <td>0.098596</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>smoke</th>\n",
"      <td>-0.003699</td>\n",
"      <td>-0.047633</td>\n",
"      <td>0.338135</td>\n",
"      <td>0.187989</td>\n",
```

```
"      <td>0.067780</td>\n",
"      <td>-0.000922</td>\n",
"      <td>0.005186</td>\n",
"      <td>0.010354</td>\n",
"      <td>-0.004756</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.340094</td>\n",
"      <td>0.025858</td>\n",
"      <td>-0.015486</td>\n",
"      <td>-0.047884</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>alco</th>\n",
"      <td>0.001210</td>\n",
"      <td>-0.029723</td>\n",
"      <td>0.170966</td>\n",
"      <td>0.094419</td>\n",
"      <td>0.067113</td>\n",
"      <td>0.001408</td>\n",
"      <td>0.010601</td>\n",
"      <td>0.035760</td>\n",
"      <td>0.011246</td>\n",
"      <td>0.340094</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.025476</td>\n",
"      <td>-0.007330</td>\n",
"      <td>-0.029918</td>\n",
"    </tr>\n",
```

```
  "    <tr>\n",
  "      <td>-0.007330</td>\n",
  "      <td>-0.035653</td>\n",
  "      <td>1.000000</td>\n",
  "      <td>0.237749</td>\n",
  "    </tr>\n",
  "    <tr>\n",
  "      <th>years</th>\n",
  "      <td>0.003050</td>\n",
  "      <td>0.999090</td>\n",
  "      <td>-0.023017</td>\n",
  "      <td>-0.081456</td>\n",
  "      <td>0.053661</td>\n",
  "      <td>0.020793</td>\n",
  "      <td>0.017754</td>\n",
  "      <td>0.154386</td>\n",
  "      <td>0.098596</td>\n",
  "      <td>-0.047884</td>\n",
  "      <td>-0.029918</td>\n",
  "      <td>-0.009819</td>\n",
  "      <td>0.237749</td>\n",
  "      <td>1.000000</td>\n",
  "    </tr>\n",
  "  </tbody>\n",
  "</table>\n",
  "</div>"
 ],
"text/plain": [
```

```
            "                id      age    gender ...  active    cardio     years\n",
            "id              1.000000  0.003457  0.003502 ...  0.003755  0.003799  0.003050\n",
            "age             0.003457  1.000000 -0.022811 ... -0.009927  0.238159  0.999090\n",
            "gender          0.003502 -0.022811  1.000000 ...  0.005866  0.008109 -0.023017\n",
            "height         -0.003038 -0.081515  0.499033 ... -0.006570 -0.010821 -0.081456\n",
            "weight         -0.001830  0.053684  0.155406 ... -0.016867  0.181660  0.053661\n",
            "ap_hi           0.003356  0.020764  0.006005 ... -0.000033  0.054475  0.020793\n",
            "ap_lo          -0.002529  0.017647  0.015254 ...  0.004780  0.065719  0.017754\n",
            "cholesterol     0.006106  0.154424 -0.035821 ...  0.009911  0.221147  0.154386\n",
            "gluc            0.002467  0.098703 -0.020491 ... -0.006770  0.089307  0.098596\n",
            "smoke          -0.003699 -0.047633  0.338135 ...  0.025858 -0.015486 -0.047884\n",
            "alco            0.001210 -0.029723  0.170966 ...  0.025476 -0.007330 -0.029918\n",
            "active          0.003755 -0.009927  0.005866 ...  1.000000 -0.035653 -0.009819\n",
            "cardio          0.003799  0.238159  0.008109 ... -0.035653  1.000000  0.237749\n",
            "years           0.003050  0.999090 -0.023017 ... -0.009819  0.237749  1.000000\n",
            "\n",
            "[14 rows x 14 columns]"
          ]
        },
```

```
      "metadata": {
        "tags": []
      },
      "execution_count": 26
    }
  ]
},
{
  "cell_type": "code",
  "metadata": {
    "id": "RU4lhqnmctWr",
    "outputId": "9795abfd-9966-4e29-9308-40f48ade87c5",
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 494
    }
  },
  "source": [
    "#Visualize the data\n",
    "import matplotlib.pyplot as plt\n",
    "plt.figure(figsize=(7,7))\n",
    "sns.heatmap(df.corr(), annot=True, fmt='.0%')"
  ],
  "execution_count": null,
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
```

```
        "text/plain": [
          "<matplotlib.axes._subplots.AxesSubplot at 0x7fd299d1e518>"
        ]
       },
       "metadata": {
        "tags": []
       },
       "execution_count": 27
      },
      {
       "output_type": "display_data",
       "data": {
        "image/png":
+d+NO7pjZSuWASG4ff1/2PxkzXiPyRQtUYSvShVNVauHWyf2xmxlb3K5K
F/xG+
        "text/plain": [
          "<Figure size 504x504 with 2 Axes>"
        ]
       },
       "metadata": {
        "tags": [],
        "needs_background": "light"
       }
      }
     ]
    },
    {
     "cell_type": "code",
     "metadata": {
```

```
    "id": "PfCa9j4H_VJg"
   },
   "source": [
    "#Remove years and id\n",
    "df=df.drop('years', axis=1)\n",
    "df=df.drop('id', axis=1)"
   ],
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "metadata": {
    "id": "8FSIMX68EuWh"
   },
   "source": [
    "#Split data into feature data and target data\n",
    "X=df.iloc[:,:-1].values\n",
    "Y=df.iloc[:,-1].values"
   ],
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "metadata": {
    "id": "7v-Evbg5FNxs"
   },
```

```
    "outputId": "ebee833f-3860-484a-8df9-78445df5916d",
    "colab": {
     "base_uri": "https://localhost:8080/",
     "height": 153
    }
   },
   "source": [
    "#Use random forest classifier\n",
    "from sklearn.ensemble import RandomForestClassifier\n",

"forest=RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=1)\n",
    "forest.fit(X_train,Y_train)"
   ],
   "execution_count": null,
   "outputs": [
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,\n",
       "                       criterion='entropy', max_depth=None, max_features='auto',\n",
       "                       max_leaf_nodes=None, max_samples=None,\n",
       "                       min_impurity_decrease=0.0, min_impurity_split=None,\n",
       "                       min_samples_leaf=1, min_samples_split=2,\n",
       "                       min_weight_fraction_leaf=0.0, n_estimators=10,\n",
```

```
          "                          n_jobs=None, oob_score=False, random_state=1, verbose=0,\n",
          "                  warm_start=False)"
         ]
        },
        "metadata": {
         "tags": []
        },
        "execution_count": 53
       }
     ]
    },
    {
     "cell_type": "code",
     "metadata": {
      "id": "CkaPVQwmGjSd",
      "outputId": "1d9e2816-72f4-4df8-ae96-9da958ccaf29",
      "colab": {
       "base_uri": "https://localhost:8080/",
       "height": 34
      }
     },
     "source": [
      "#Testing model accuracy\n",
      "model=forest\n",
      "model.score(X_train,Y_train)"
     ],
     "execution_count": null,
     "outputs": [
```

```
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "0.979904761904762"
      ]
     },
     "metadata": {
      "tags": []
     },
     "execution_count": 54
    }
   ]
  },
  {
   "cell_type": "code",
   "metadata": {
    "id": "4WM1h6uEGu_F"
   },
   "source": [
    "#Testing model accuracy on the test data\n",
    "from sklearn.metrics import confusion_matrix\n",
    "cm=confusion_matrix(Y_test, model.predict(X_test))\n",
    "TN=cm[0][0]\n",
    "TP=cm[1][1]\n",
    "FN=cm[1][0]\n",
    "FP=cm[0][1]"
   ],
```

```
  "execution_count": null,
  "outputs": []
},
{
  "cell_type": "code",
  "metadata": {
    "id": "TMXzAAPHHNEc",
    "outputId": "3fbf9484-fb33-4660-c0e9-0239a4336501",
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 68
    }
  },
  "source": [
    "print(cm)\n",
    "\n",
    "print('Model accuracy={}'.format((TP+TN)/(TP+TN+FN+FP)))"
  ],
  "execution_count": null,
  "outputs": [
    {
      "output_type": "stream",
      "text": [
        "[[6589 2020]\n",
        " [3349 5542]]\n",
        "Model accuracy=0.6932\n"
      ],
      "name": "stdout"
```

```
      }
     ]
    },
    {
     "cell_type": "code",
     "metadata": {
      "id": "aS-NhBQaH3JT"
     },
     "source": [
      ""
     ],
     "execution_count": null,
     "outputs": []
    }
  ]
}
```