POLYCYSTIC OVARY SYNDROME DETECTION & PREDICTION SYSTEM USING RANDOM FOREST AND CONVOLUTION NEURAL NETWORKS

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF
BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY

By
Batch – 9

K.B.S. Anjali(20JG1A1225) V. Gireeshma (20JG1A1258)

Ruksar Begum (20JG1A1246) D. Yashaswini (20JG1A1206)

Under the esteemed guidance of
Mrs. R. SRIDEVI
Assistant Professor
Department of Information Technology

Department of Information Technology
GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
[Approved by AICTE NEW DELHI, Affiliated to JNTUK Kakinada]
[Accredited by National Board of Accreditation (NBA) for B.Tech. CSE, ECE & IT – Valid from 2019-22 to 2022-25] [Accredited by National Assessment and Accreditation Council [NAAC] with A Grade – valid from 2022-2027] Kommadi,Madhurawada, Visakhapatnam – 530048

2020 – 2024

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN
DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project report titled **“POLYCYSTIC OVARY SYNDROME DETECTION & PREDICTION SYSTEM USING RANDOM FOREST AND CONVOLUTION NEURAL NETWORKS”** is a bonafide work of following IV-II B.Tech students in the Department of Information Technology, Gayatri Vidya Parishad College of Engineering for Women affiliated to JNTU, Kakinada during the academic year 2023- 2024, in fulfillment of the requirement for the award of the degree of Bachelor of Technology of this university.

K.B.S. Anjali(20JG1A1225)	V. Gireeshma (20JG1A1258)
Ruksar Begum (20JG1A1246)	D. Yashaswini (20JG1A1206)

Internal Guide	**Head of the Department**
Mrs. R. Sridevi	Dr. Bhanu Sridhar
Assistant Professor	Associate Professor

External Examiner

ACKNOWLEDGEMENT

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

We feel elated to extend our sincere gratitude to Mrs. R. Sridevi, Assistant Professor for encouragement all the way during analysis of the project. Her annotations, insinuations and criticisms are the key behind the successful completion of the thesis and for providing us all the required facilities.

We express our deep sense of gratitude and thanks to Dr. Bhanu Sridhar, Professor and Head of the Department of Information Technology for his guidance and for expressing valuable and grateful opinions in the project for its development and for providing lab sessions and extra hours to complete the project.

We are also thankful to other teaching faculty and non-teaching staff of the Department of Information Technology for giving valuable suggestions for our project.

We would like to take this opportunity to express our profound sense of gratitude to Vice Principal, Prof. G. Sudheer for allowing us to utilize the college resources thereby facilitating the successful completion of our thesis.

We would like to take the opportunity to express our profound sense of gratitude to the revered Principal, Dr. R.K. Goswami for allowing us to utilize the college resources thereby facilitating the successful completion of our thesis.

VISION & MISSION

INSTITUTE VISION:

- To emerge as an acclaimed center of learning that provides value based technicaleducation for the holistic development of students.

INSTITUTE MISSION:

- Undertake activities that provide value-based knowledge in Science, Engineering & Technology.
- Provide opportunities for learning through Industry – Institute interaction on the state –of – the – art technologies.
- Create a collaborative environment for research, innovation and entrepreneurship toflourish.
- Promote activities that bring in a sense of social responsibility.

DEPARTMENT VISION:

- The Department of Information Technology strives to produce competent professionalswho are technically sound and ethically strong for the IT industry.

DEPARTMENT MISSION:

- Provide quality training that prepares Students to be technically competent for theIndustrial and Societal needs.
- Facilitate an environment that promotes continuous learning to face the challenges in theIT sector. Provide opportunities for learning, leadership and communication skills.

****CONTENTS****

ABSTRACT

Polycystic Ovary Syndrome (PCOS) is a prevalent endocrine disorder causes female fertility, affecting women in their fertilization age, even steady far off the reproductive age. Capitalizing machine learning algorithms and a comprehensive dataset comprising clinical hormonal and imaging parameters. Our model aims for automated detection and prediction of PCOS in individuals. Additionally, we propose a novel detection system that integrates medical imaging analysis, such as ultrasound scans with advanced pattern recognition techniques to enhance the accuracy of PCOS diagnosis. Our hybrid model aims to accurately classify individuals with PCOS and predict its onset in at-risk populations. Through rigorous evaluation, our framework demonstrates promising results, offering a potential tool for clinicians to enhance early diagnosis and personalized treatment strategies for PCOS.

LIST OF FIGURES

LIST OF OUTPUT SCREENS

1. INTRODUCTION

1.1 Problem Definition

Polycystic Ovary Syndrome (PCOS) is a common hormonal disorder affecting women of reproductive age. Early and accurate detection of PCOS is crucial for proper management and treatment. However, diagnosing PCOS can be challenging due to its heterogeneous nature and the involvement of multiple factors.

1.2 Objective

The objective of this project is to develop machine learning model and deep learning model that can effectively predict and detect PCOS based on relevant medical data and imaging data.

1.3 Motivation

The project is driven by the potential of ML and DL techniques to address the complexities and challenges associated with diagnosing Polycystic Ovary Syndrome (PCOS). These advanced algorithms can effectively handle and integrate heterogeneous data sources, including clinical parameters, hormonal levels, demographic information, and medical images. ML models like Random Forests can provide personalized predictions by learning from individual patient data, while Convolutional Neural Networks (CNNs) can excel at analyzing medical images to detect subtle patterns indicative of PCOS. Furthermore, ML models offer interpretability, allowing clinicians to understand the factors contributing to predictions, fostering trust and better decision-making. The scalability and efficiency of these algorithms make them suitable for large-scale screening efforts, potentially reducing the burden on healthcare systems.

1.4 Limitations of the project

The system's performance heavily relies on the quality and quantity of training data, which can be challenging to acquire for PCOS. Data issues like incompleteness, noise, or bias can lead to inaccurate predictions sometimes. Addressing these limitations through robust data collection, responsible deployment, and careful study design is crucial for enhancing the system's reliability and slight variation in the data type of values entering for the report data analysis may mislead in proper detection.

1.5 Organization of Documentation

A concise overview of the rest of the documentation work is explained below. Definition of a few terms which are involved in building this software were reviewed.

Chapter 1: Introduction

Chapter 2: Literature survey describes the primary terms involved in the development of this software. It gives an overview of the existing system and features of the proposed system.

Chapter 3: Analysis deals with detailed analysis of the project.

Chapter 4: Design includes UML diagrams along with explanation of module design

Chapter 5: Contains screenshots of Output Screens with Implementation

Chapter 6: Contains project conclusion and future enhancements.

Chapter 7: References are mentioned along with the resources.

2. LITERATURE SURVEY

2.1 Introduction

What is Polycystic ovary syndrome?

Polycystic ovary syndrome (PCOS) is one of the most common endocrine and metabolic disorders in premenopausal women. Heterogeneous by nature. We can analyze patient data (hormone levels, medical history). It can be done in many ways. By using Machine learning (ML) and deep learning (DL) algorithms are increasingly used to analyze patient data (hormone levels, medical history).

Definition:

Polycystic ovary syndrome (PCOS) is a technique which is defined by a combination of signs and symptoms of androgen excess and ovarian dysfunction in the absence of other specific diagnoses. The aetiology of this syndrome remains largely unknown, but mounting evidence suggests that PCOS might be a complex multigenic disorder with strong epigenetic and environmental influences, including diet and lifestyle factors. PCOS is frequently associated with abdominal adiposity, insulin resistance, obesity, metabolic disorders and cardiovascular risk factors.

How does gender recognition through voice works?

A PCOS (Polycystic Ovary Syndrome) detection system utilizing ultrasound images and clinical reports typically follows these steps:

1.**Data Acquisition:** The system acquires ultrasound images of ovaries from patients suspected of having PCOS. These images are obtained through medical ultrasound examinations performed by healthcare professionals. Additionally, clinical reports containing relevant information such as hormone levels, menstrual history, and physical examination findings are collected.

2.**Image and Report Preprocessing:** The ultrasound images and clinical reports are preprocessed to ensure consistency and suitability for analysis. Image preprocessing may involve resizing, normalization, and noise reduction techniques to enhance image quality. Clinical report preprocessing may include text processing techniques to extract relevant information and convert it into a structured format suitable for analysis.

3.**Feature Extraction**: Features are extracted from both the ultrasound images and clinical reports to capture relevant characteristics indicative of PCOS. For ultrasound images, features may include follicle counts, ovarian volume, and the presence of cysts. Clinical report features may include hormone levels (e.g., testosterone, LH/FSH ratio), menstrual irregularities, and physical examination findings (e.g., hirsutism, acne).
4.**Model Development**: A machine learning or deep learning model is developed to classify patients into PCOS-positive or PCOS-negative categories based on the extracted features. This model may use techniques such as logistic regression, support vector machines (SVM), or neural networks. Ensemble methods combining multiple models may also be employed to improve classification performance.
5.**Training and Validation**: The developed model is trained using a labeled dataset containing both ultrasound images and corresponding clinical reports, along with their PCOS status (positive or negative). The model's performance is evaluated using cross-validation or held-out validation data to ensure generalization to unseen samples.
6.**Model Integration:** The trained model is integrated into a PCOS detection system that takes ultrasound images and clinical reports as input and produces PCOS classification results as output. This system may include a user interface for inputting patient data and viewing classification results.

7.**Inference:** When a patient's ultrasound images and clinical reports are inputted into the system, the integrated model performs inference to classify the patient as PCOS-positive or PCOS-negative based on the extracted features. This classification is based on the model's learned patterns from the training data.

8.**Result Presentation**: The system presents the classification results to healthcare professionals, indicating whether the patient is likely to have PCOS or not. Additionally, the system may provide explanations or visualizations of the features contributing to the classification decision to aid interpretation.

9.**Feedback and Iteration**: The system may incorporate feedback mechanisms to continuously improve its performance. This could involve collecting feedback from healthcare professionals on the accuracy of classification results or periodically retraining the model with updated data to adapt to evolving patterns.

Overall, a PCOS detection system leveraging ultrasound images and clinical reports combines image analysis with clinical data interpretation to assist healthcare professionals in diagnosing and managing PCOS effectively.

2.2 Algorithms

2.2.1 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. As the name suggests, "Random Forest is a classifier that contains a number of decision trees ". It can be used for both Classification and Regression problems in ML.

Random Forest Algorithm: 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." 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 of the model.

Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

- Step-1: Select random K data points from the training set.
- Step-2: Build the decision trees associated with the selected data points (Subsets).
- Step-3: Choose the number N for decision trees that you want to build.
- Step-4: Repeat Step 1 & 2.
- Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

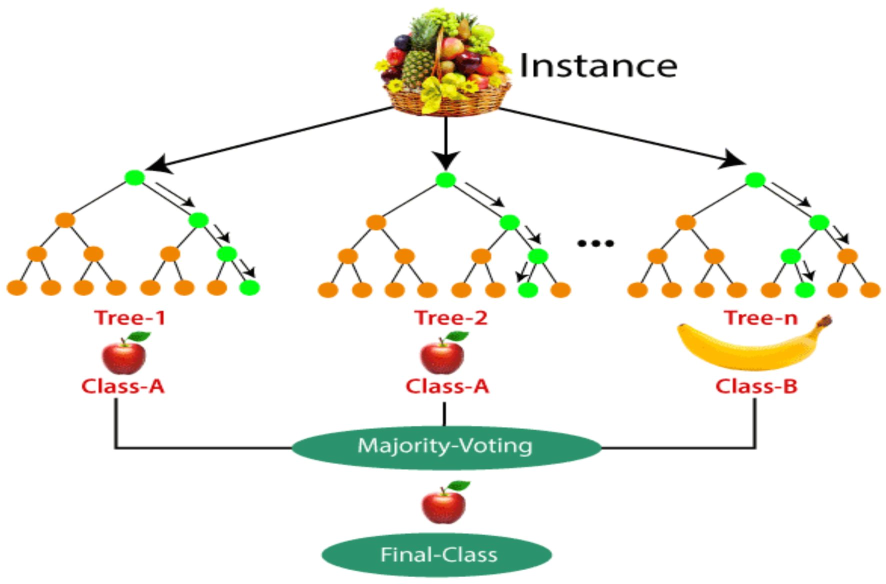


Fig 2.2.1: Random Forest Algorithm

2.2. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data. Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

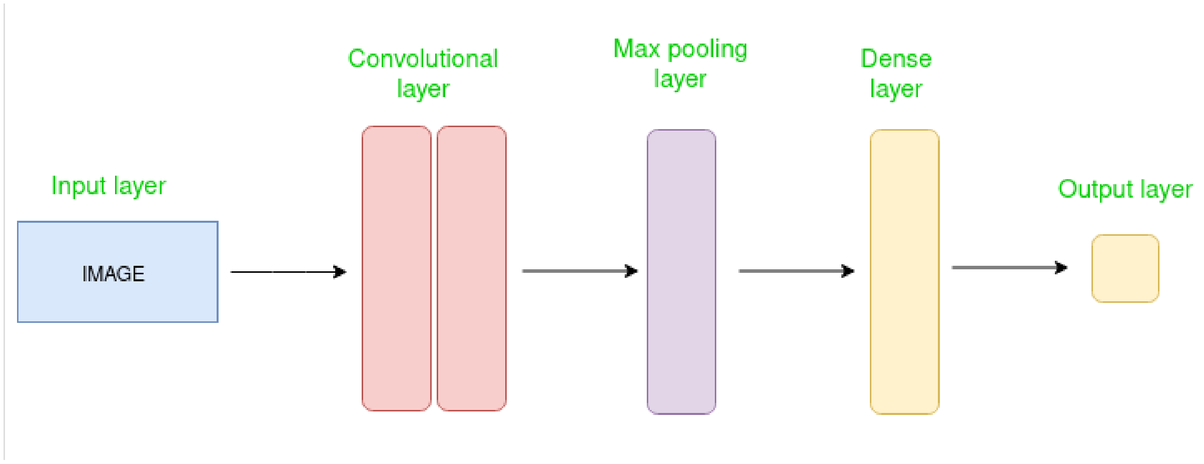
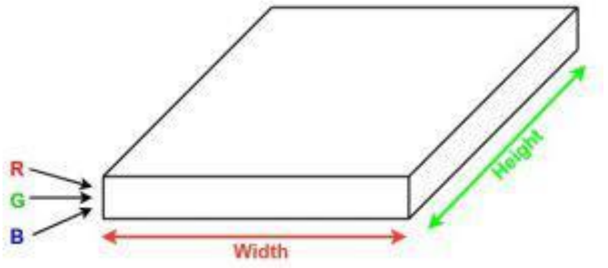


Fig 2.2.2: CNN Algorithm

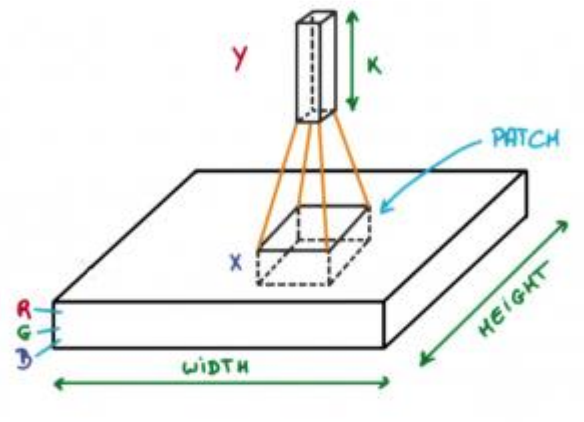
The Convolutional layer applies filters to the input image to extract features, the Pooling layer down samples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

How CNN works?

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels).



Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called **Convolution**. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.



Now let’s talk about a bit of mathematics that is involved in the whole convolution process.

- Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimensions 34x34x3. The possible size of filters can be axax3, where 'a' can be anything like 3, 5, or 7 but smaller as compared to the image dimension.
- During the forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have a value of 2, 3, or even 4 for high-dimensional images) and compute the dot product between the kernel weights and patch from input volume.
- As we slide our filters we will get a 2-D output for each filter and we will stack them together as a result, we will get output volume having a depth equal to the number of filters. The network will learn all the filters.

2.2.3 MobileNet Architecture

MobileNet is TensorFlow's first mobile computer vision model. It uses depth-wise separable convolutions to significantly reduce the number of parameters compared to other networks with regular convolutions and the same depth in the nets. This results in lightweight deep neural networks. MobileNet is a class of convolutional neural network (CNN) that was open-sourced by Google, and therefore, provides an excellent starting point for training classifiers that are insanely small and insanely fast. A depth-wise separable convolution is made from two operations.

i. Depth-wise convolution.

ii. Pointwise convolution.

1. Depth-wise separable convolution

The Depth-wise separable convolution is comprising of two layers, the depth-wise convolution, and the point-wise convolution. Basically the first layer is used to filter the input channels and the second layer is used to combine them to create a new feature.

1.1 Depth-wise convolution

The depth-wise convolutions are used to apply a single filter into each input channel. This is different from a standard convolution in which the filters are applied to all of the input channels.

Let's take a **standard convolution**,
For a standard convolution, the computational cost depends multiplicatively on the number of input and output channels and on the spatial dimensions of the input feature map and convolution kernel. However, this method is only used to filter the input channel.

1.2 Point-wise Convolution

Since the depth-wise convolution is only used to filter the input channel, it does not combine them to produce new features. So an additional layer called pointwise convolution layer is made, which computes a linear combination of the output of depth-wise convolution using a 1 × 1 convolution.

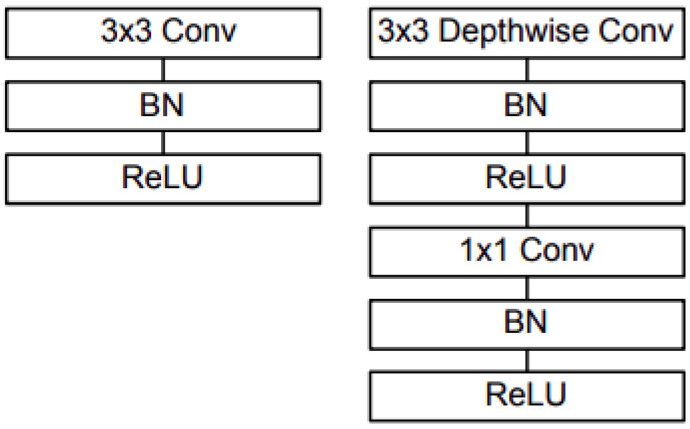


Fig 2.2.3: Left: Standard Convolution followed by batch normalization and RELU. Right: Depth-wise convolution layer and pointwise convolution layer, each followed by batch normalization and RELU.

2.3 Existing System

- Existing systems for PCOS detection and prediction harness the power of both machine learning (ML) and deep learning (DL) methodologies to analyze diverse datasets encompassing medical imaging data, patient demographics, and clinical indicators.

- Current methods of diagnosing PCOS often involve clinical tests and expert consultation, which may lead to delayed detection.
- There is a lack of easily accessible tools for individuals to assess their risk of PCOS early on, contributing to potential health complications.
- Moreover, DL architectures such as Support Vector Machines are employed to extract intricate patterns and features from medical images.

Disadvantages of Existing Systems

- **Manual Diagnosis**: The current diagnostic process for PCOS often relies on manual assessment which can be time-consuming and prone to human error.
- **Limited Accuracy**: Traditional diagnostic methods may lack precision and accuracy, leading to misdiagnosis or delayed diagnosis of PCOS.
- **Costly and Invasive Tests**: Some diagnostic tests for PCOS, such as hormonal assays and transvaginal ultrasound, can be costly and invasive.
- **Lack of Predictive Capability**: The current approach may focus primarily on diagnosing existing symptoms rather than predicting the risk of developing PCOS.

2.4 Proposed System

A proposed system for PCOS detection and prediction using machine learning (ML) and deep learning (DL) techniques involves the development of a comprehensive platform that integrates diverse datasets comprising medical imaging data. The proposed system introduces an innovative approach by machine learning algorithms to analyze user-provided health data for early PCOS detection. The system provides a user-friendly interface that allows the clinicians to assess the risk and help them in proper understanding. Including the preprocessing and feature extraction, relevant information is extracted from the datasets which is then utilized to train ML algorithms i.e Random Forests alongside DL architectures i.e convolutional neural networks (CNNs).

Advantages of Proposed System:

1. **High Accuracy**: Many proposed systems achieve high accuracy in PCOS detection, often outperforming traditional diagnostic methods.
2. **Automation:** ML and DL-based systems automate the process of PCOS detection, reducing the need for manual interpretation of medical images and data.
3. **Early Detection:** By leveraging advanced algorithms, proposed systems can detect PCOS at an earlier stage, enabling timely intervention and treatment.
4. **Scalability:** ML and DL techniques allow for scalability, meaning the systems can handle large volumes of data efficiently.

2.5 Dataset

polycystic-ovary-syndrome-pcos:

This dataset includes 2 csv files, 1 file data contains patient's details with infertility problem and other is without infertility problem which shows us both infected and not infected patient's with PCOS.

Reference:https://www.kaggle.com/datasets/prasoonkottarathil/polycystic-ovary-syndrome-pcos

pcos-detection-using-ultrasound-images:

This dataset included ultrasound pictures of healthy ovaries as well as cystic ovaries. The ultrasound photos that showed infected cysts on the ovary were given the label 'infected' while the ultrasound images that showed a healthy ovary were given the label 'not infected'**.**

Reference:https://www.kaggle.com/datasets/anaghachoudhari/pcos-detection-using-ultrasound-images

Attributes are:

PCOS (Y/N): Describes wheter if a patient is effected with PCOS or NOT.

Age: Describes about the age.

BMI: Body Mass Index (BMI), A numerical measure based on height and weight, used to assess an individual's body composition and potential health risks associated with weight.

Blood Group: Distinguishes the types of blood groups.

Pulse rate(bpm): Measures the Pulse rate.

Hb(g/dl): Hemoglobin (Hb) concentration in grams per deciliter (g/dL): a measure of the amount of oxygen-carrying protein in the blood.

Cycle(R/I): Describes about the menstrual cycle.

Cycle length(days): Describes the cycle length.

Pregnant(Y/N): Tells wheter the patient is pregnant.

I beta-HCG(mIU/mL): Beta-HCG (beta-human chorionic gonadotropin) levels measured in milli-international units per milliliter (mIU/mL).

FSH/LH: Follicle-stimulating hormone (FSH) and luteinizing hormone (LH) levels, key markers of reproductive function, typically measured together.

TSH (mIU/L): Thyroid-stimulating hormone (TSH) levels measured in milli-international units per liter (mIU/L), reflecting thyroid function.

AMH(ng/mL): Anti-Müllerian hormone (AMH) levels measured in nanograms per milliliter (ng/mL), indicating ovarian reserve and fertility potential.

Weight gain(Y/N): Describes the weight of a patient.

Hair loss(Y/N): Describes the hair conditions.

Reg.Exercise(Y/N): Describes about the patient’s fitness.

Follicle No. (L): Number of follicles observed in the left ovary during ultrasound examination, indicating ovarian health and fertility potential.

Follicle No. (R): Number of follicles observed in the right ovary during ultrasound examination, indicating ovarian health and fertility potential.

Endometrium (mm): Thickness of the endometrial lining measured in millimeters, often assessed in relation to menstrual cycle phase or fertility.

3. ANALYSIS

3.1 Introduction

Integrated Development and Learning Environment- Python is a versatile and dynamically typed programming language known for its simplicity, readability, and extensive ecosystem of libraries and frameworks. With its clean and intuitive syntax, Python enables developers to write elegant and efficient code for a wide range of applications, including web development, data analysis, artificial intelligence, scientific computing, automation, and more. Python's interpreted nature allows for rapid development and interactive experimentation, while its cross-platform compatibility ensures that code can run seamlessly on different operating systems. Supported by a vibrant community of developers worldwide, Python continues to be a top choice for both beginners and experienced programmers due to its ease of learning, versatility, and robustness.

3.2 Software Requirement Specification

3.2.1 Software Requirements

a) Operating System

An operating system (OS) is software that manages computer hardware and software resources and provides common services for computer programs. Operating system used is Windows OS with a 64-bit operating system.

b) Programming Language

A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output. Programming languages generally consist of instructions for a computer. Programming languages can be used to create programs that implement specific algorithms.

Programming Language: Python

Python Packages:

- tensorflow
- keras
- csv
- sklearn
- streamlit
- numpy

- dropout

3.2.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, a hardware requirement list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. RAM stands for Random Access Memory. Computer runs smoother and has more RAM. The RAM size is 4GB.

Hard Drives

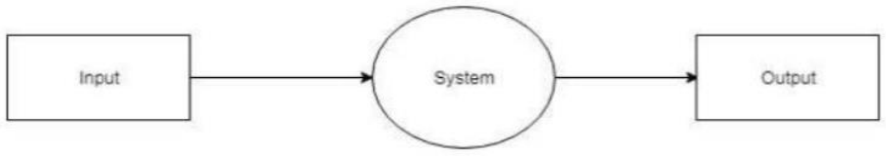
Hard Drives store all of your computer information once your system is turned off. Hard drive of 4GB is minimum required.

Processor

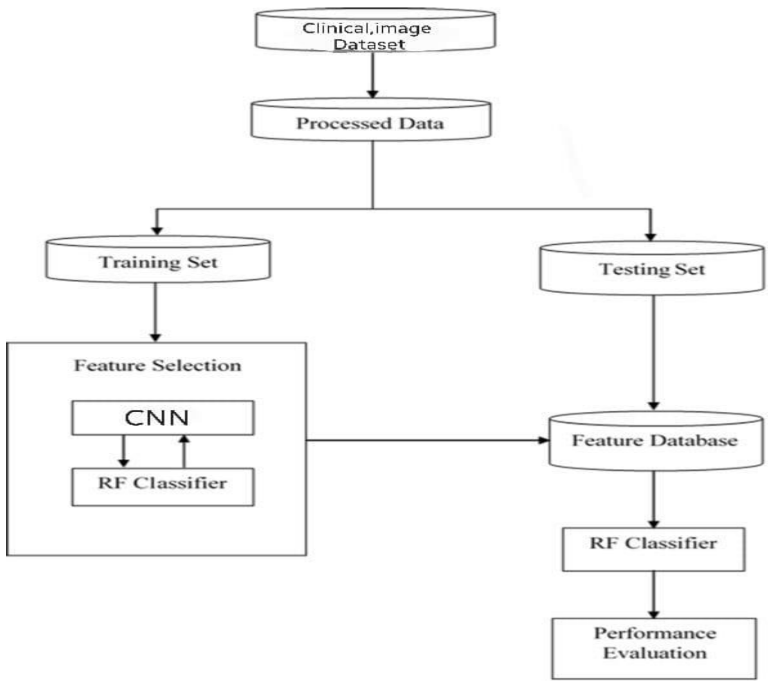
A processor is the logic circuitry that responds to and processes the basic instructions that drive a computer. Processor being used is IntelRCorei5@2.20GHz.

RAM	4GB
Processor	Intel RCorei5
Speed	2.30GHz
Hard Disk	320

3.3 Dataflow Diagram of the Project



Level-0



Level-1

Fig 3.3: Dataflow Diagram

3.4 Architecture of our Project

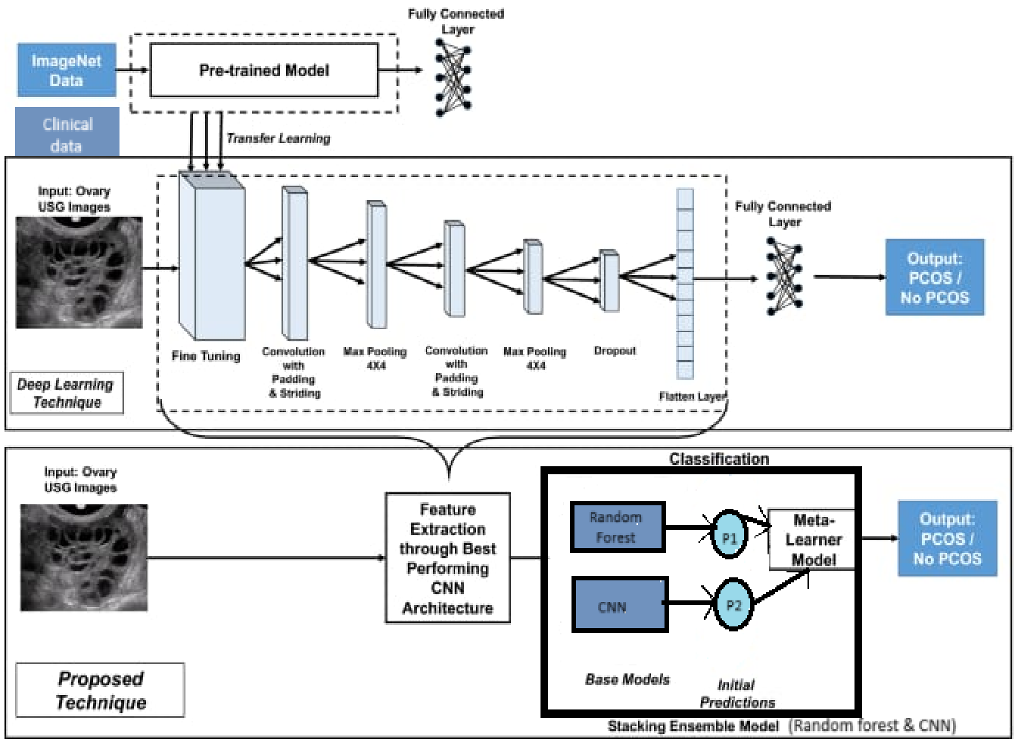


Fig 3.4: Architecture

List of Modules

1) DATA PREPROCESSING

2) FEATURE EXTRACTION

3) TRAINING RANDOM FOREST CLASSIFIER, CONVOLUTION NEURAL NETWORK MODEL

4) TESTING

5) INTERACTIVE WEBPAGE

PERFORMANCE ANALYSIS:

TYPE OF DATASETS	**REPORT DATA**			**IMAGE DATA**	
SYSTEMS	**Existing System**		**Proposed system**	**Existing System**	**Proposed system**
ALGORITHMS	**SVM**	**XGBOOST**	**Random Forest**	**Inception V3**	**Mobile Net**
ACCURACY	**85.2%**	**85.89%**	**91.4%**	**77.9%**	**99.4%**
PRECISION	**88.1%**	**78.5%**	**87.5%**	**78%**	**99.45%**

4. DESIGN

4.1 Introduction

Unified modelling language (UML) is used to represent the software in a graphical format, that is it provides a standard way to visualize how a system is designed Good UMI. design is followed for a better and precise software output.

The UML provides different constructs for specifying, visualizing, constructing and documenting the artifacts of software systems. UML diagrams are used to better understand the system, to maintain the document information about the system and emphasizes on the roles, actors, actions, actions and process. The UML diagrams considered are: Use-case Diagram Class Diagram Sequence Diagram Relationships:

The relationships among different entities in the following diagrams are:

Association: This relationship tells how two entities interact with each other. Dependency: An entity is dependent on another if the change of that is reflecting the other.

Aggregation: It is a special kind of association which exhibits part-of relationship. Generalization: This relationship describes the parent-child relationship among different entities.

Realization: It is a kind of relationship in which one thing specifies the behavior or a responsibility to be carried out, and the other thing carries out that behavior.

4.2 UML Diagrams

UML stands for Unified Modelling Language UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML is different from the other common programming languages like C++, Java, and COBOL, etc. It is designed to enable users to develop an expressive and ready to use visual modelling language In addition, it supports high level development concepts such as frameworks, patterns and I collaborations. UML can be described as a general-purpose visual modelling language to visualize, specify, construct and document software system. But it is not limited within this boundary. To be more precise, UML is a pictorial language used to make software bloe prints UML has a direct relation with object-oriented analysis and design. After some standardization UML becomes an OMG (Object Management Group) standard.

The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting

Goals of UML:

Provide users with a ready-is-se, expressive visual molding language so they can develop and exchange meaningful models. Provide and specialization mechanists is extending the core concepts. Be independent of particular programming languages and development process Provide a formal basis for understanding the modeling language. Encourage the growth of the OO tools market. Support higher-level development concepts such as collaborations, frameworks and components. The system can be software or non-software. So, it must be clear the UML is not just development method

Basic Building Blocks of UML:

The basic building blocks in UML are things and relationships These are mind is different ways following different names to create different types of diagrams in UML there and Eight types of diagrams, below is a list and brief description of the man in-depth descriptions in the document, will focus on the first five diagrams in the form, what can he as the most general, sometimes also referred to as the UML core diagrams.

Components of the UML:

The UML consists of a number of graphical elements that combine so form diagrams. Because it's a language, the UML. has rules for combining these clones. The purpose of the diagrams to present multiple views of the system, and this set of multiples is called Model. A UML. Model of a system is something like scale model of a building. UML model describes what's system is supposed to do. It does not tell how to implement the system.

These are the artifacts of a software-intensive system. The abbreviation for UML is Unified Modelling Language and is being brought of a designed make sure that the exiting ER Diagrams which do not serve the purpose will be replaced by the UML Diagrams where in these language as its own set of Diagram.

Some of the Diagrams that help for Diagrammatic Approach for the Object-oriented Software Engineering are:

- Class diagram
- Use case diagram
- Sequence diagram
- State chart diagram
- Activity diagram

4.2.1 Use-Case Diagram

Use case diagrams capture the behavior of a system and help to emphasize the requirements of the system. It describes the high-level functionalities of the system. It consists of the following artifacts: Actor Use case Secondary Actor and Use case Boundary.

Actor: A role of a user that interacts with the system you're modelling is represented by an actor. A person, an organization, a machine, or another external system all be considered users

Use-Case: A use case is a function that a system does to help the wir achieve their goal A must produce an observable result that is useful to the system's user

Secondary Actor: Actors who are not directly interacting with the system are secondary actors and are placed on the right side of the use case boundary. Use case Boundary:

It is the boundary to separate the use cases that are internal to the system from the actors that are external to the system

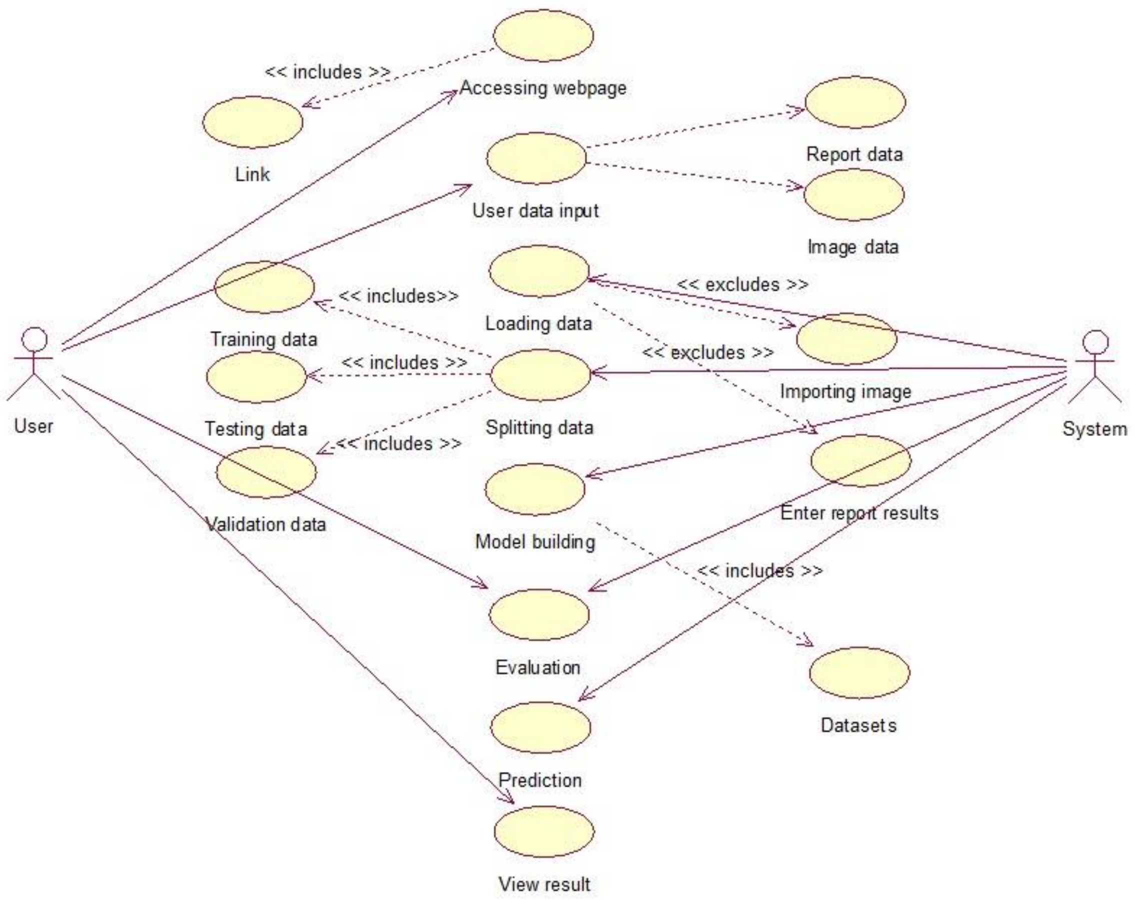


Fig 4.2.1: Use-Case Diagram

4.2.2 Class Diagram

Class diagram is a static overview of the system used to highlight the important aspects as well as executables in the system. These are the only diagrams which can be mapped with the object-oriented systems. These diagrams show the responsibilities of class and relationships among different classes in the system Artifacts of the Class Diagram are

Class: The class represents similar objects and consists of name, responsibilities and attributes

Name: Name of the class is the identity of the class Responsibilities: Responsibilities represents the duties to be done or functionalities that the class exhibits.

Attributes: Attributes represent the properties that a particular class has and the objects of the class will have similar attributes.

Dependency: Dependency is a weaker form of bond which indicates that one class depends on another because it uses it at some point in time. One class depends on another if the independent class is a parameter variable or local variable of a method of the dependent class.

Multiplicity: This association relationship indicates that one of the related classes make reference to the other. The UML representation of an association is a line with an optional arrow head indicating the role of the object(s) in the relationship, and an optional notation at each end indicating the multiplicity of instances of that entity (the number of objects that participate in the association).

Visibility: To Specify the visibility of a class member (i.e., any attribute or method), the sanitation's must be placed before the member's name.

+	Public
/	Derived
#	Protected
-	Private
~	Package

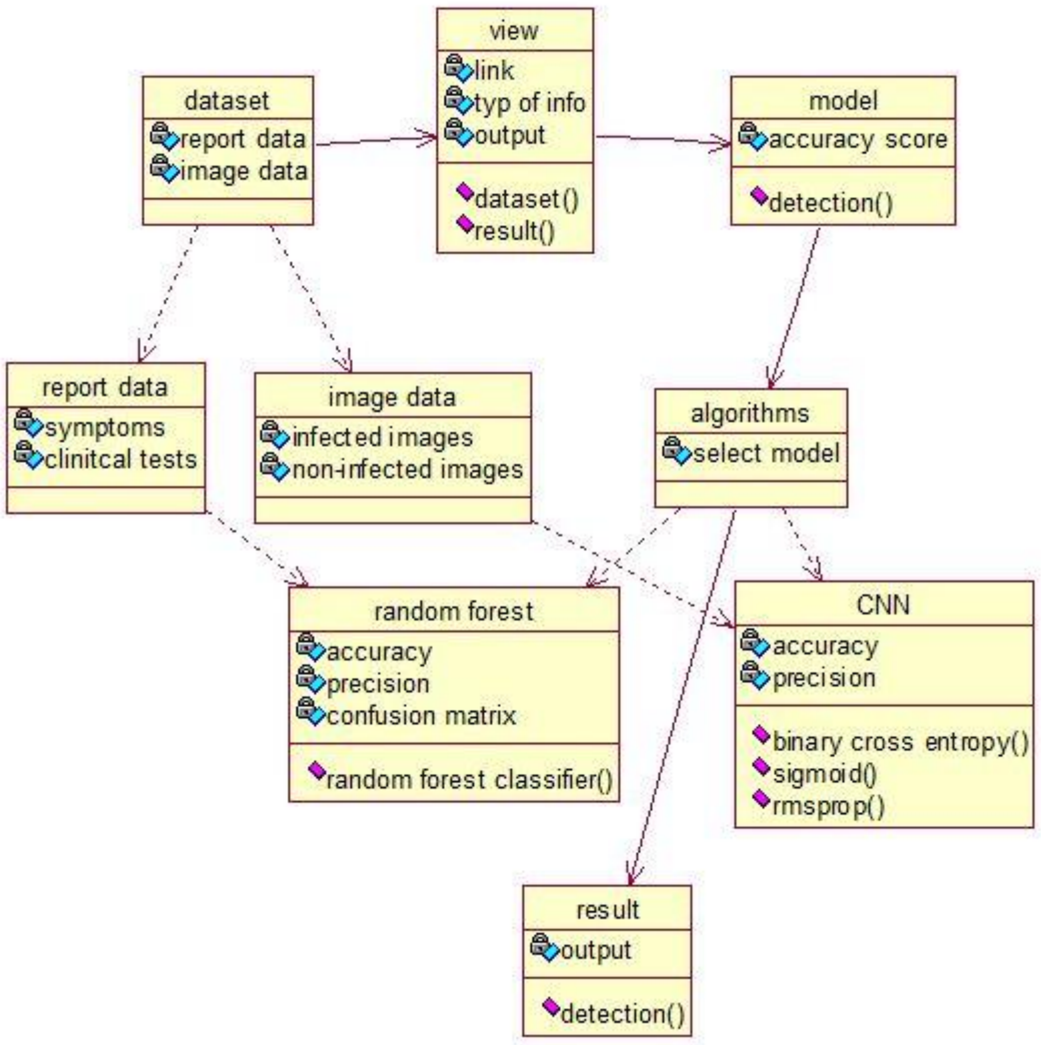


Fig 4.2.2: Class Diagram

4.2.3 Sequence Diagram

The sequence diagram, also known as an event diagram, depicts the flow of messages through the system. It aids in the visualization of a variety of dynamic scenarios. It depicts communication between any two lifelines as a time-ordered series of events, as if these lifelines were present at the same moment. The message flow is represented by a vertical dotted line that extends across the bottom of the page in UML, whereas the lifeline is represented by a vertical bar. It encompasses both iterations and branching. Notations of Sequence Diagram are:

Lifeline: A lifeline represents an individual participant in the sequence diagram. It is at the very top of the diagram.

Actor: An actor is a character who interacts with the subject and plays a part. It isn't covered by the system's capabilities. It depicts a role in which human users interact with external devices or subjects.

Message: Message denotes the interactions between the messages. They are in the sequential ordering the timeline.

Call message: By defining a specific communication between the interaction's lifelines, it shows that the target lifeline has invoked an operation.

Return Message: It specifies a specific communication between the interaction lifelines, which reflect the flow of data from the receiver of the associated caller message.

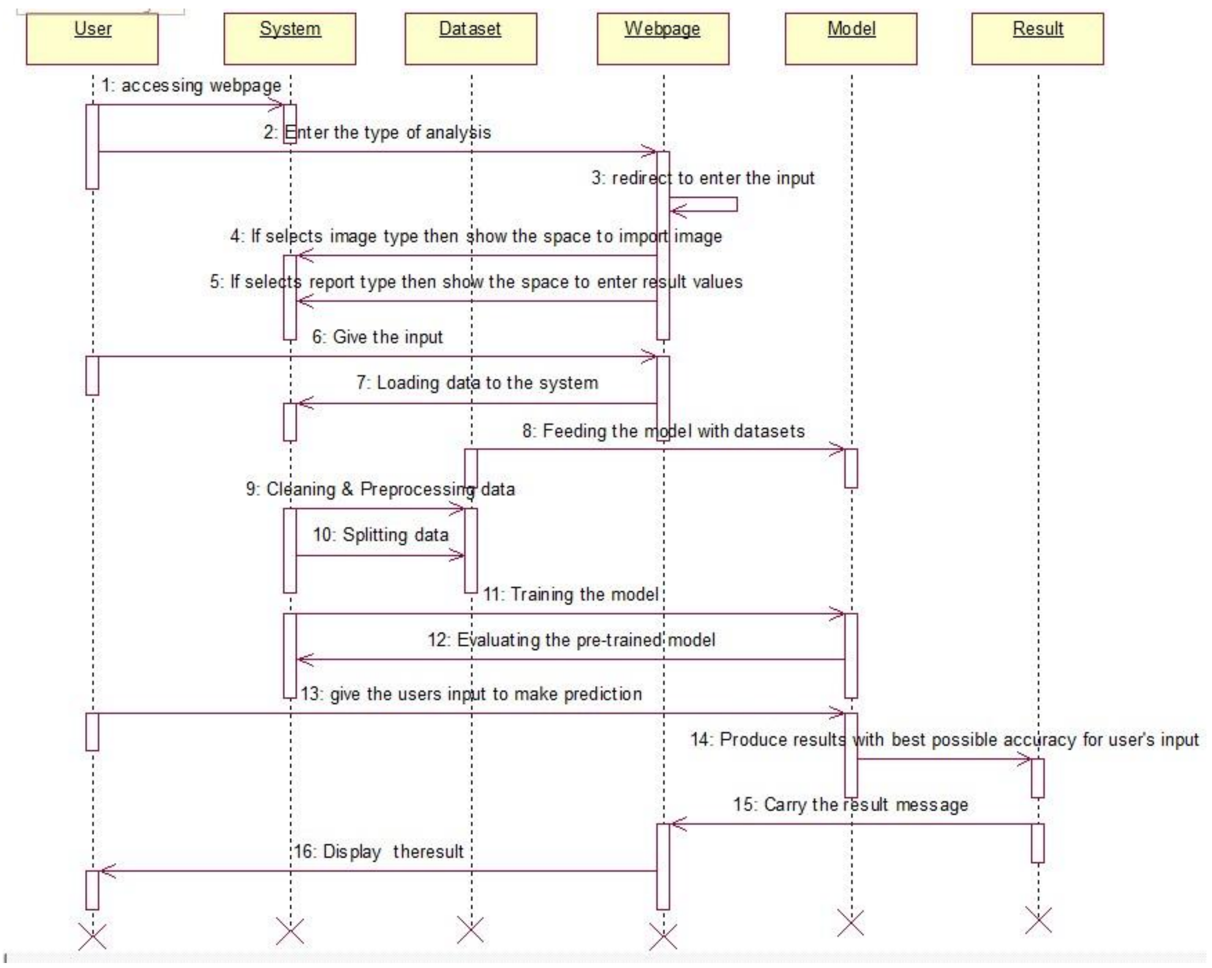

User
System
Dataset
Webpage
Model
Result
1: accessing webpage
2: Enter the type of analysis
3: redirect to enter the input
4: If selects image type then show the space to import image
5: If selects report type then show the space to enter result values
6: Give the input
7: Loading data to the system
8: Feeding the model with datasets
9: Cleaning & Preprocessing data
10: Splitting data
11: Training the model
12: Evaluating the pre-trained model
13: give the users input to make prediction
14: Produce results with best possible accuracy for user's input
15: Carry the result message
16: Display theresult

Fig 4.2.3: Sequence Diagram

4.2.4. State Chart Diagram

At any given time, an object is in particular state. One way to characterize change in a system is to say that its objects change the state in response to events and to time. The UML State Diagram captures these kinds of changes it presents the states an object can be in along with the transitions between the states, and shows the starting point and endpoint of a sequence of state changes.

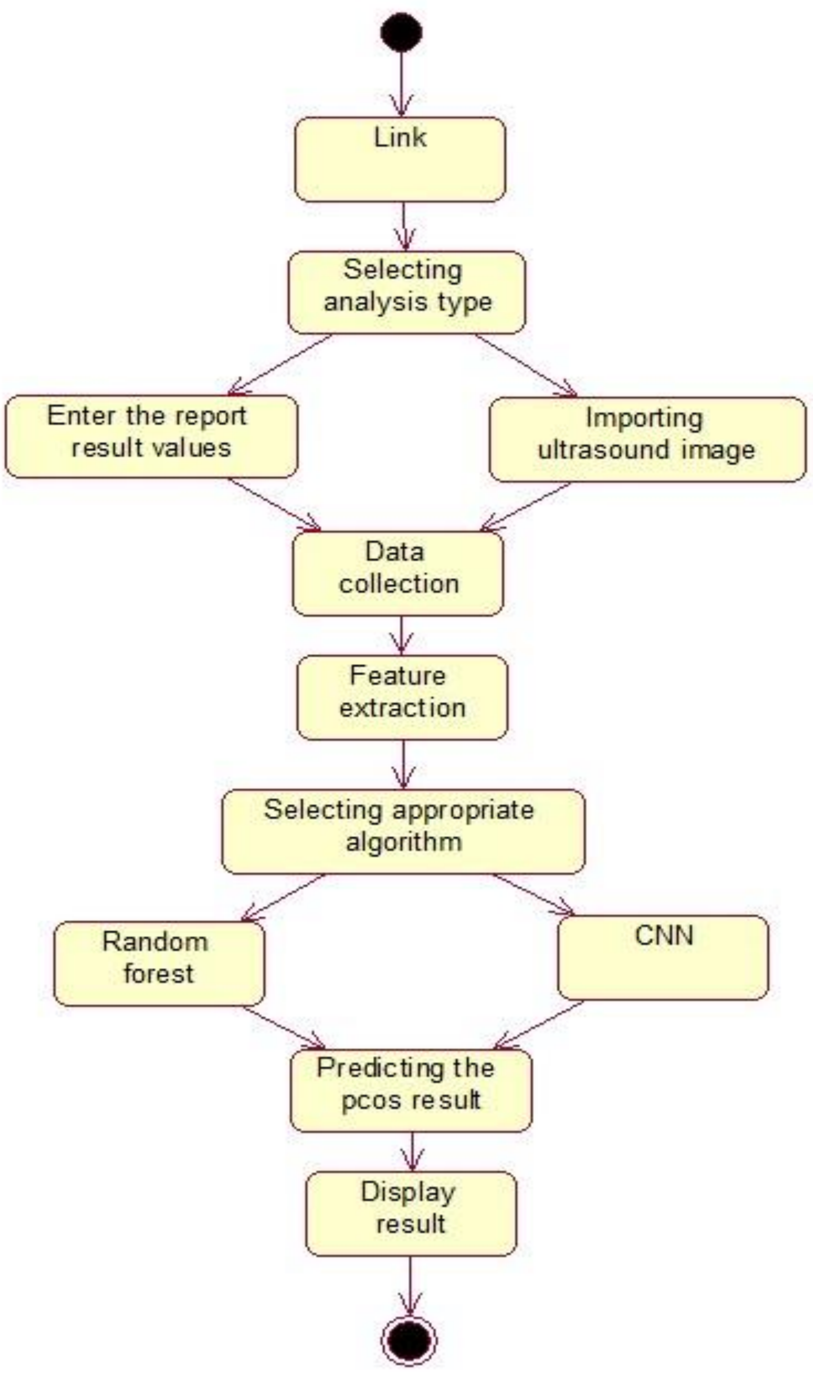

Link
Selecting analysis type
Enter the report result values
Importing ultrasound image
Data collection
Feature extraction
Selecting appropriate algorithm
Random forest
CNN
Predicting the pcos result
Display result

Fig 4.2.4: State Chart Diagram

A Rounded Rectangle represents a state, along with the solid line and arrowhead that represents a transition. The arrowhead points to the state being transitioned into. The solid circle symbolizes the starting point and the bull's eye symbolizes the end point.

4.2.5. Activity Diagram

The Activity Diagram highlights the activities. Each activity is represented by a rounded rectangle narrower and more oval-shaped than the state icon. An arrow represents the transition from one activity to the next. The activity diagram has a starting point represented by a filled-in circle, and an endpoint represented by a bull's eye. Activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagram are used to model workflow or business processes and internal operation. It is a special kind of statement diagram that shows the flow from activity to activity within the system. Activity diagram address the dynamic view of the system. The easiest way to visualize an Activity diagram is to think of a flowchart of a code.

The flowchart issued to depict the business logic flow and the events that cause decisions and actions in the code to take place. An Activity diagram represents the business and operational workflow of a system.

An Activity diagram shows the overall flow of control.

- Initial node
- Activity final node
- Activity
- Flow/edge
- Fork
- Join
- Condition
- Decision
- Merge
- Partition
- Final Flow

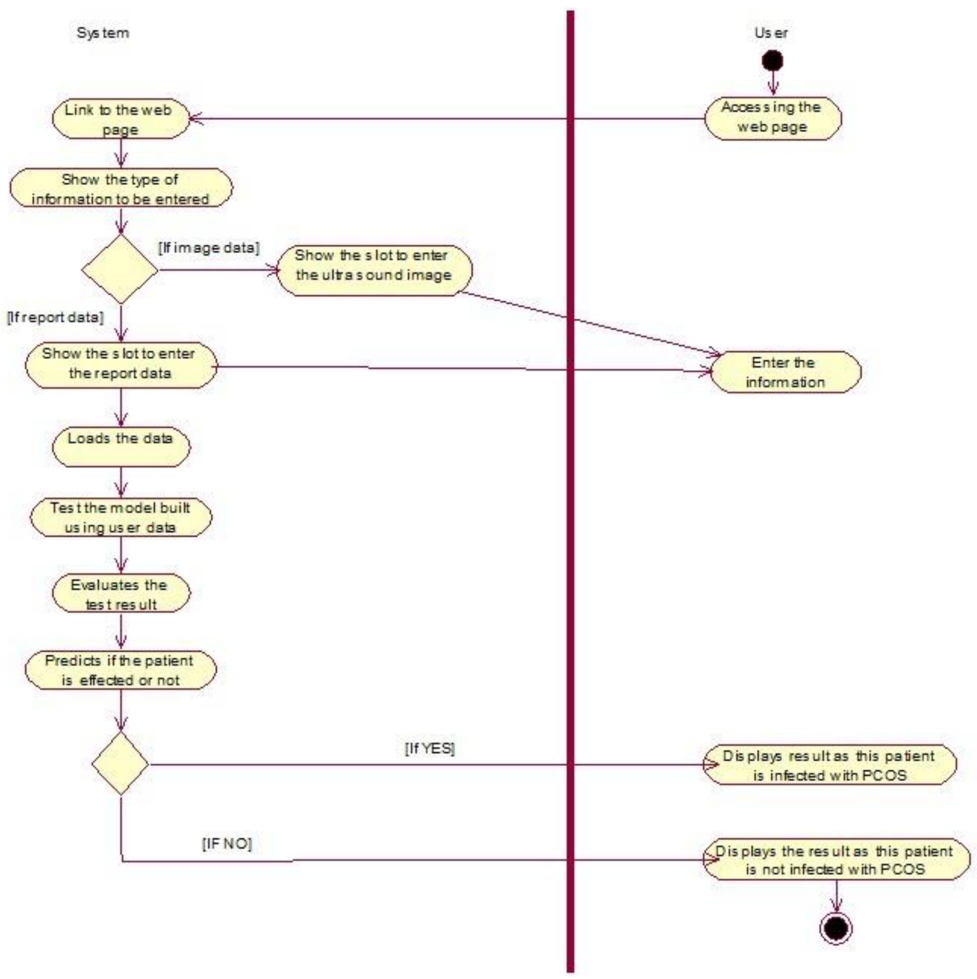

System
User
Accessing the web page
Link to the web page
Show the type of information to be entered
[If image data]
Show the slot to enter the ultra sound image
[If report data]
Show the slot to enter the report data
Enter the information
Loads the data
Test the model built using user data
Evaluates the test result
Predicts if the patient is effected or not
[If YES]
Displays result as this patient is infected with PCOS
[IF NO]
Displays the result as this patient is not infected with PCOS

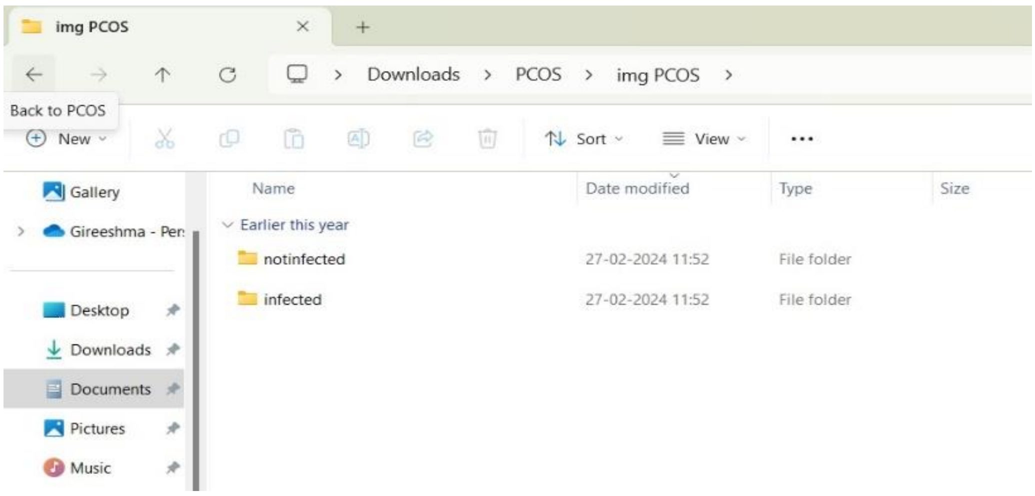
Fig 4.2.5: Activity Diagram

5. IMPLEMENTATION OF MODULES

5.1. Data preprocessing

- By opening the webpage user clicks on the type of data and the user needs to give input of required data. The data will be taken and processed, after processing the result will be displayed.

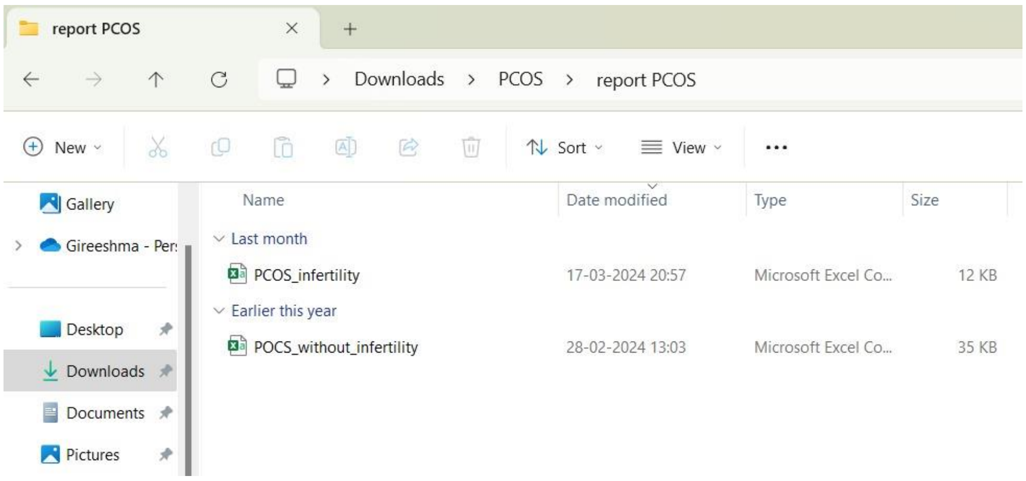
Output:



img PCOS

Downloads > PCOS > img PCOS

Name	Date modified	Type	Size
Earlier this year			
notinfected	27-02-2024 11:52	File folder	
infected	27-02-2024 11:52	File folder	



report PCOS

Downloads > PCOS > report PCOS

Name	Date modified	Type	Size
Last month			
PCOS_infertility	17-03-2024 20:57	Microsoft Excel Co...	12 KB
Earlier this year			
POCS_without_infertility	28-02-2024 13:03	Microsoft Excel Co...	35 KB

Fig 5.1: Data Preprocessing

MODULE-2

5.2. Feature Extraction

Polycystic Ovary Syndrome (PCOS) prediction and detection often involve analyzing various data types such as medical images, hormone levels, clinical symptoms, and genetic markers. Feature extraction plays a crucial role in identifying relevant patterns and characteristics from these data sources. Here's a general approach for feature extraction in PCOS prediction and detection:

1. Medical Imaging:

-Ultrasound: Extract features such as ovarian volume, follicle count, follicle size, and presence of cysts.

- Ultrasound Images: Extract features related to ovarian morphology, such as shape irregularities or structural abnormalities.

-Texture analysis: Analyze the texture patterns within the ovarian tissue to extract relevant features.

2.Hormonal Data:

-Testosterone levels: Extract statistical features such as mean, median, standard deviation, and skewness.

-Luteinizing hormone (LH) and follicle-stimulating hormone (FSH) levels: Extract features related to their ratio, peaks, and fluctuations.

-Insulin levels: Extract features indicating insulin resistance, such as fasting insulin levels and HOMA-IR index.

3.Clinical Symptoms:

-Menstrual cycle irregularities: Extract features related to cycle length, frequency of menstruation, and presence of amenorrhea.

-Hirsutism: Extract features from standardized scoring systems like the Ferriman-Gallwey score.

-Acne and alopecia: Quantify severity and frequency of these symptoms.

4.Genetic Markers:

-Single Nucleotide Polymorphisms (SNPs): Extract features related to specific genetic variations associated with PCOS, such as those in genes involved in hormone regulation or insulin signaling pathways.

-Epigenetic markers: Extract features from methylation patterns associated with PCOS-related genes.

5.Machine Learning-based Feature Selection:

- Utilize algorithms such as Principal Component Analysis (PCA), Recursive Feature Elimination (RFE), or feature importance from ensemble models to select the most informative features from high-dimensional datasets.

6.Integration and Fusion:

- Combine features from different modalities (e.g., imaging, hormonal, clinical, genetic) to capture comprehensive information about PCOS.
- Use techniques like feature concatenation or fusion to merge features from diverse sources into a single feature vector for analysis.

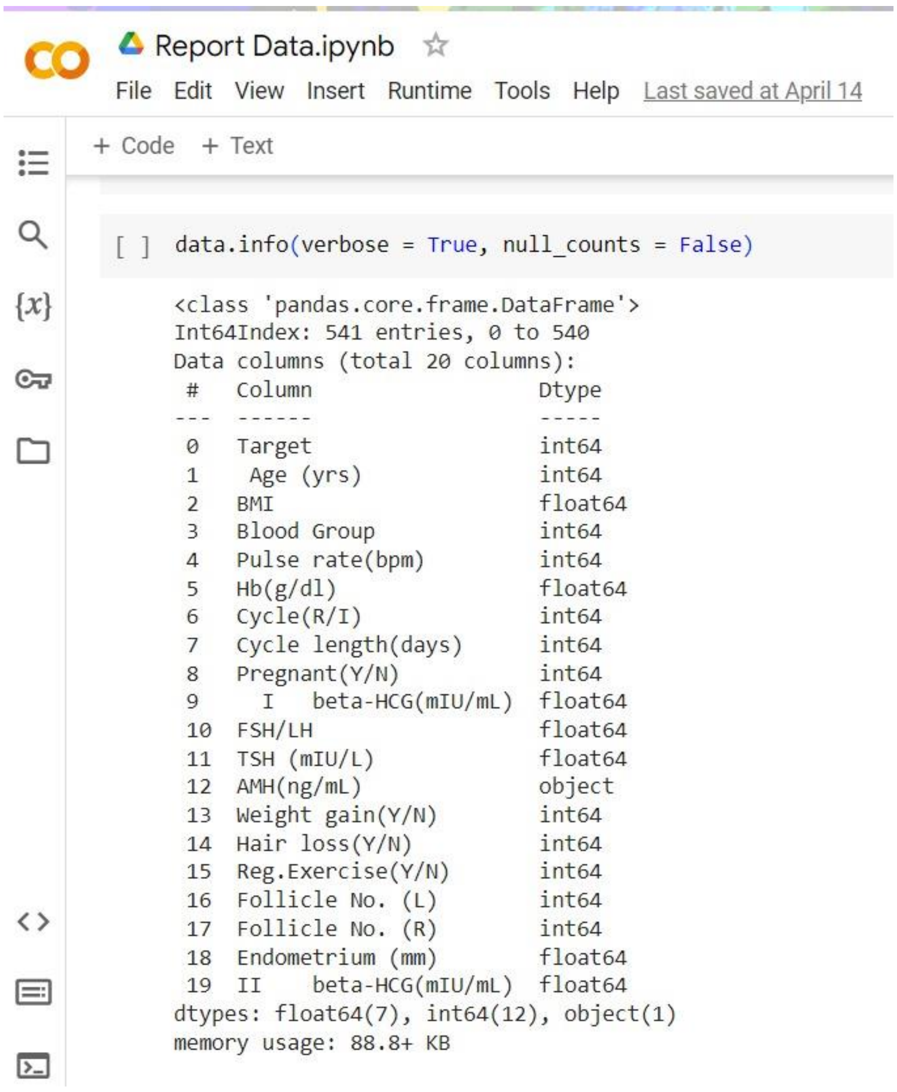
7.Normalization and Preprocessing:

- Normalize features to ensure that they are on a similar scale, preventing any particular feature from dominating the analysis due to its magnitude.
- Handle missing data appropriately through imputation techniques or exclusion based on the extent of missingness.

8.Validation and Model Training:

- Validate the extracted features using appropriate statistical tests and visualization techniques.
- Train machine learning models (e.g., logistic regression, random forest, support vector machines) using the extracted features for PCOS prediction and detection.

By systematically extracting and analyzing relevant features from multiple data sources, you can improve the accuracy and reliability of PCOS prediction and detection models. Additionally, ongoing research and advancements in feature extraction techniques may further enhance the understanding and diagnosis of PCOS.

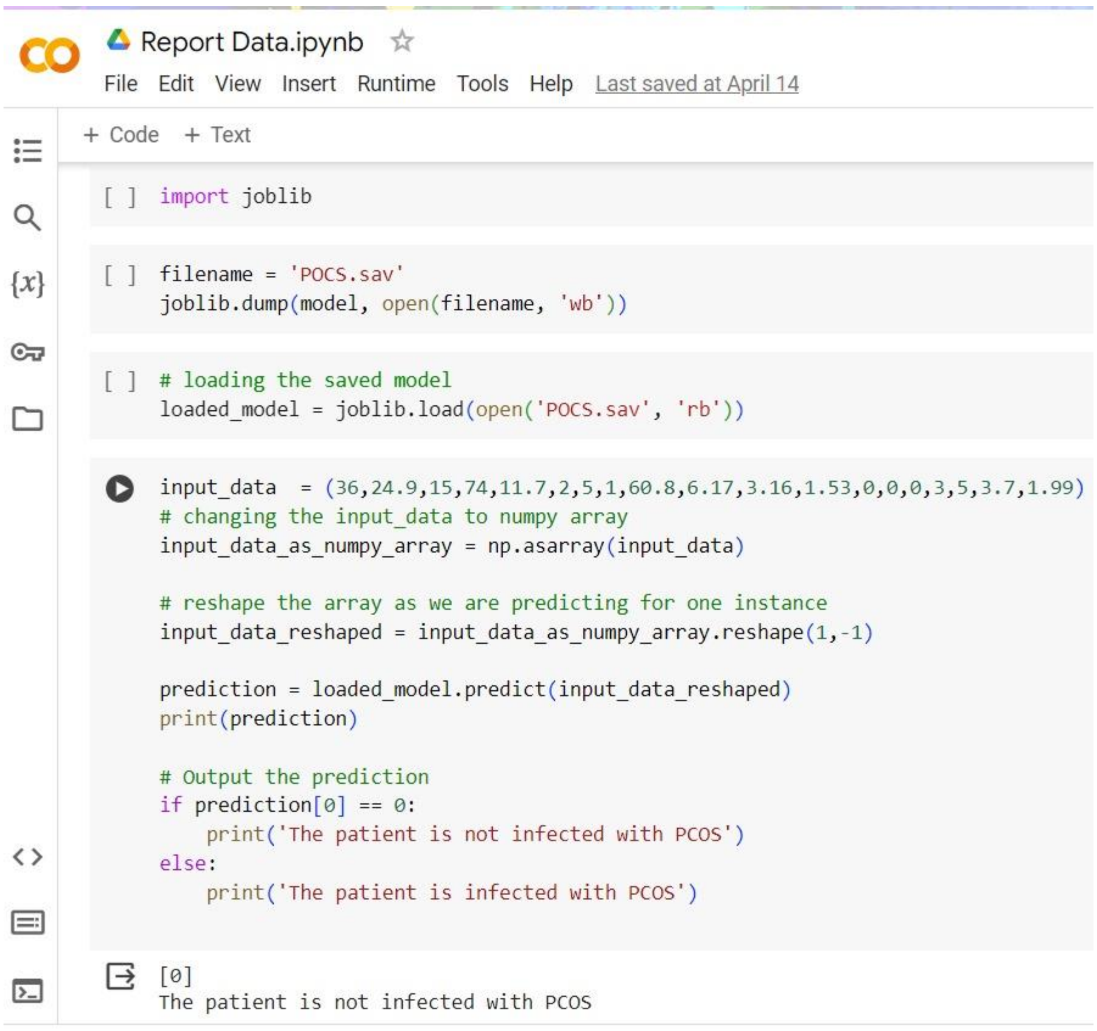

Report Data.ipynb
File Edit View Insert Runtime Tools Help Last saved at April 14
+ Code + Text

```
[ ]  data.info(verbose = True, null_counts = False)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 541 entries, 0 to 540
Data columns (total 20 columns):
 #   Column                  Dtype
---  ------                  -----
 0   Target                  int64
 1    Age (yrs)              int64
 2   BMI                     float64
 3   Blood Group             int64
 4   Pulse rate(bpm)         int64
 5   Hb(g/dl)                float64
 6   Cycle(R/I)              int64
 7   Cycle length(days)      int64
 8   Pregnant(Y/N)           int64
 9     I   beta-HCG(mIU/mL)  float64
 10  FSH/LH                  float64
 11  TSH (mIU/L)             float64
 12  AMH(ng/mL)              object
 13  Weight gain(Y/N)        int64
 14  Hair loss(Y/N)          int64
 15  Reg.Exercise(Y/N)       int64
 16  Follicle No. (L)        int64
 17  Follicle No. (R)        int64
 18  Endometrium (mm)        float64
 19  II    beta-HCG(mIU/mL)  float64
dtypes: float64(7), int64(12), object(1)
memory usage: 88.8+ KB
```

OUTPUT:

In this step extracted features will be stored and used for further process. The features such as Age, BMI, Blood Group, Pulse rate, HB(g/dl), Cycle, FSH and Follicle counts of both the left and right ovaries will be used. By using these the process of feature extraction process will be done.


Report Data.ipynb
File Edit View Insert Runtime Tools Help Last saved at April 14
+ Code + Text

```
import joblib
```

```
filename = 'POCS.sav'
joblib.dump(model, open(filename, 'wb'))
```

```
# loading the saved model
loaded_model = joblib.load(open('POCS.sav', 'rb'))
```

```
input_data  = (36,24.9,15,74,11.7,2,5,1,60.8,6.17,3.16,1.53,0,0,0,3,5,3.7,1.99)
# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = loaded_model.predict(input_data_reshaped)
print(prediction)

# Output the prediction
if prediction[0] == 0:
    print('The patient is not infected with PCOS')
else:
    print('The patient is infected with PCOS')
```

```
[0]
The patient is not infected with PCOS
```

Fig 5.2: Feature Extraction

MODULE-3

5.3. Training the model using Random Forest and Mobile Net Architectures

Mobile Net Architecture:

Training a MobileNet architecture for PCOS prediction and detection follows a similar process to training other convolutional neural networks (CNNs), but with some considerations specific to the MobileNet architecture. MobileNet is known for its efficiency and suitability for mobile and embedded devices due to its lightweight design. Here's how you would train a MobileNet model for PCOS prediction and detection:

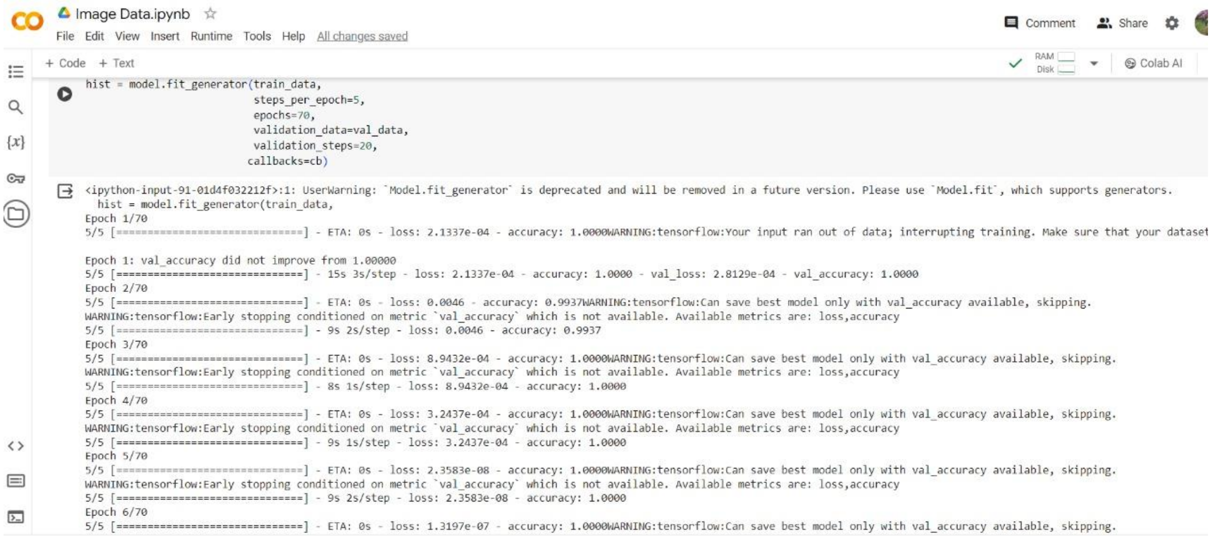
- Collect a dataset of medical images relevant to PCOS diagnosis, such as ovarian ultrasound images.

- Annotate the images with labels indicating the presence or absence of PCOS.
- Preprocess the images by resizing them to fit the input size expected by the MobileNet architecture (typically 224x224 pixels) and normalizing pixel values.
- Split the dataset into training, validation, and testing sets as described earlier.
- Choose a MobileNet variant suitable for the task. MobileNetV1, MobileNetV2, and MobileNetV3 are common choices, each offering different trade-offs between speed and accuracy.
- Customize the MobileNet architecture to fit the input image size and the number of output classes (binary classification for PCOS prediction).
- Initialize the MobileNet model with random weights.
- Feed batches of preprocessed training images into the model and perform forward propagation to obtain predictions.
- Compute the loss between the predicted outputs and the ground truth labels using a suitable loss function (e.g., binary cross-entropy).
- Backpropagate the loss through the network to update the model parameters using an optimization algorithm such as stochastic gradient descent (SGD) or Adam.
- Iterate the training process over multiple epochs, monitoring performance on the validation set to prevent overfitting.
- Evaluate the trained MobileNet model on the held-out testing set to assess its performance.
- Calculate evaluation metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC).
- Visualize the model's predictions and analyze any misclassifications.
- Fine-tune the MobileNet model by adjusting hyperparameters such as learning rate, batch size, and regularization strength to optimize performance further.
- Explore techniques like transfer learning, where the MobileNet model is initialized with weights pretrained on a large dataset (e.g., ImageNet) and then fine-tuned on the

PCOS dataset to leverage learned features.

By following these steps and leveraging the efficiency of the MobileNet architecture, you can train a lightweight yet effective model for PCOS prediction and detection, suitable for deployment in resource-constrained environments or mobile applications.

OUTPUT:



```
hist = model.fit_generator(train_data,
                           steps_per_epoch=5,
                           epochs=70,
                           validation_data=val_data,
                           validation_steps=20,
                          callbacks=cb)
```

```
<ipython-input-91-01d4f032212f>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  hist = model.fit_generator(train_data,
Epoch 1/70
5/5 [==============================] - ETA: 0s - loss: 2.1337e-04 - accuracy: 1.0000WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset

Epoch 1: val_accuracy did not improve from 1.00000
5/5 [==============================] - 15s 3s/step - loss: 2.1337e-04 - accuracy: 1.0000 - val_loss: 2.8129e-04 - val_accuracy: 1.0000
Epoch 2/70
5/5 [==============================] - ETA: 0s - loss: 0.0046 - accuracy: 0.9937WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_accuracy` which is not available. Available metrics are: loss,accuracy
5/5 [==============================] - 9s 2s/step - loss: 0.0046 - accuracy: 0.9937
Epoch 3/70
5/5 [==============================] - ETA: 0s - loss: 8.9432e-04 - accuracy: 1.0000WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_accuracy` which is not available. Available metrics are: loss,accuracy
5/5 [==============================] - 8s 1s/step - loss: 8.9432e-04 - accuracy: 1.0000
Epoch 4/70
5/5 [==============================] - ETA: 0s - loss: 3.2437e-04 - accuracy: 1.0000WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_accuracy` which is not available. Available metrics are: loss,accuracy
5/5 [==============================] - 9s 1s/step - loss: 3.2437e-04 - accuracy: 1.0000
Epoch 5/70
5/5 [==============================] - ETA: 0s - loss: 2.3583e-08 - accuracy: 1.0000WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_accuracy` which is not available. Available metrics are: loss,accuracy
5/5 [==============================] - 9s 2s/step - loss: 2.3583e-08 - accuracy: 1.0000
Epoch 6/70
5/5 [==============================] - ETA: 0s - loss: 1.3197e-07 - accuracy: 1.0000WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
```

Fig 5.3: Algorithm Accuracy

<u>Module-4</u>

5.4. Interactive Webpage

Streamlit: Streamlit is an open-source Python library that allows you to create interactive web applications quickly and easily, without the need for extensive web development knowledge. It is designed to make the process of building data-driven web applications more accessible to data scientists, analysts, and developers.

Some of the features of Streamlit include:

- Simple Python Syntax
- Rapid Prototyping
- Widgets for Interactivity
- Data Visualization
- Custom Components
- Sharing and Deployment

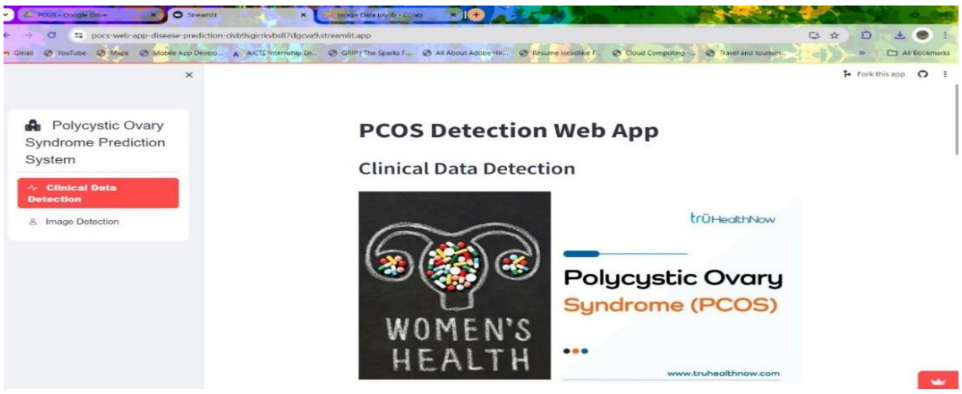
Easy integration with other libraries and tools, including databases and authentication systems.

Streamlit is a popular choice for web development because of its simplicity, flexibility, and scalability. It allows developers to create web applications quickly and easily, and its modular design allows for easy customization and extension. Additionally, Flask has a large and active community of developers who contribute plugins and extensions that can be used to add additional functionality to Flask applications.

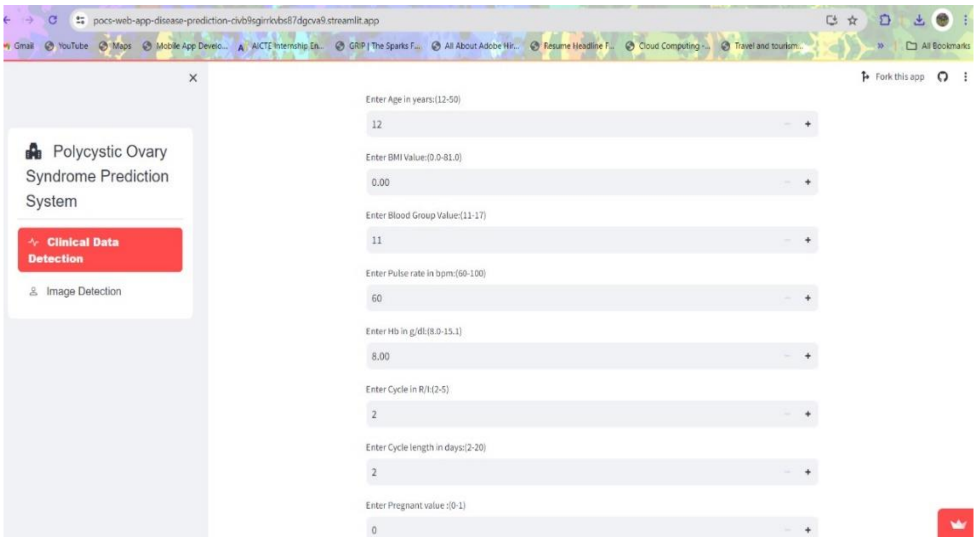
- We used Streamlit to connect and create a real-time interactive webpage.
- With Streamlit, you can create interactive elements such as sliders, buttons, and text inputs directly in your Python script. You can then use these elements to control and visualize data, making it easy to prototype and showcase your data-driven projects.
- Streamlit provides a single function-based API, which makes it easy to learn and use. You can simply write Python code to define your app's layout and functionality, and Streamlit takes care of rendering it into a web app that can be accessed through a browser.

Overall, Streamlit is a powerful tool for turning your data science and machine learning projects into interactive web applications with minimal effort.

Interactive Webpage for Report Data:



- The user will select the type of data to be entered.



Enter Age in years:(12-50)
12

Enter BMI Value:(0.0-81.0)
0.00

Enter Blood Group Value:(11-17)
11

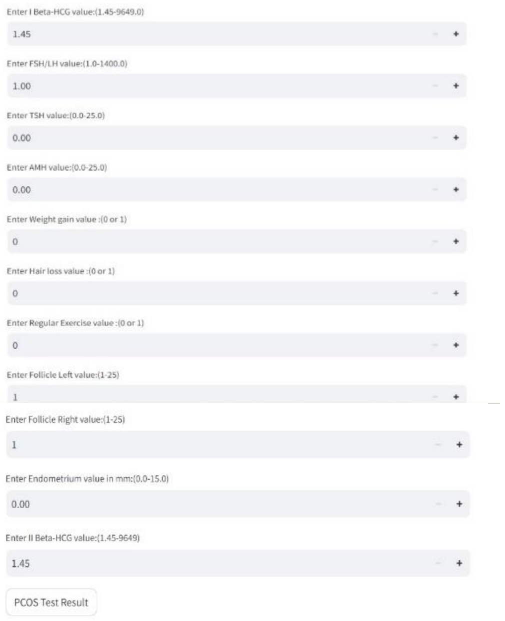
Enter Pulse rate in bpm:(60-100)
60

Enter Hb in g/dl:(8.0-15.1)
8.00

Enter Cycle in R/I:(2-5)
2

Enter Cycle length in days:(2-20)
2

Enter Pregnant value :(0-1)
0



Enter I Beta-HCG value:(1.45-9649.0)
1.45

Enter FSH/LH value:(1.0-1400.0)
1.00

Enter TSH value:(0.0-25.0)
0.00

Enter AMH value:(0.0-25.0)
0.00

Enter Weight gain value :(0 or 1)
0

Enter Hair loss value :(0 or 1)
0

Enter Regular Exercise value :(0 or 1)
0

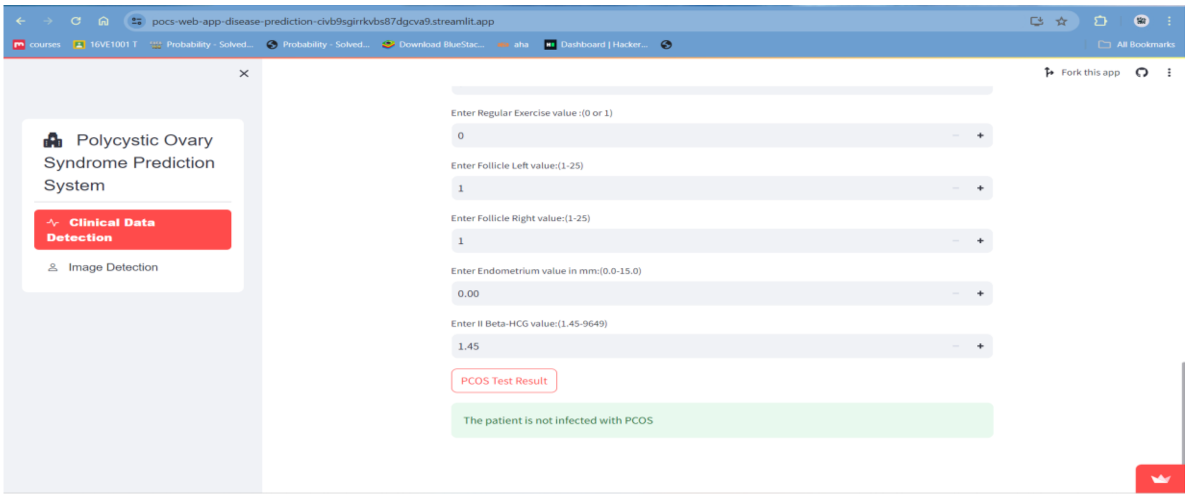
Enter Follicle Left value:(1-25)
1

Enter Follicle Right value:(1-25)
1

Enter Endometrium value in mm:(0.0-15.0)
0.00

Enter II Beta-HCG value:(1.45-9649)
1.45

Output:



Enter Regular Exercise value :(0 or 1)
0

Enter Follicle Left value:(1-25)
1

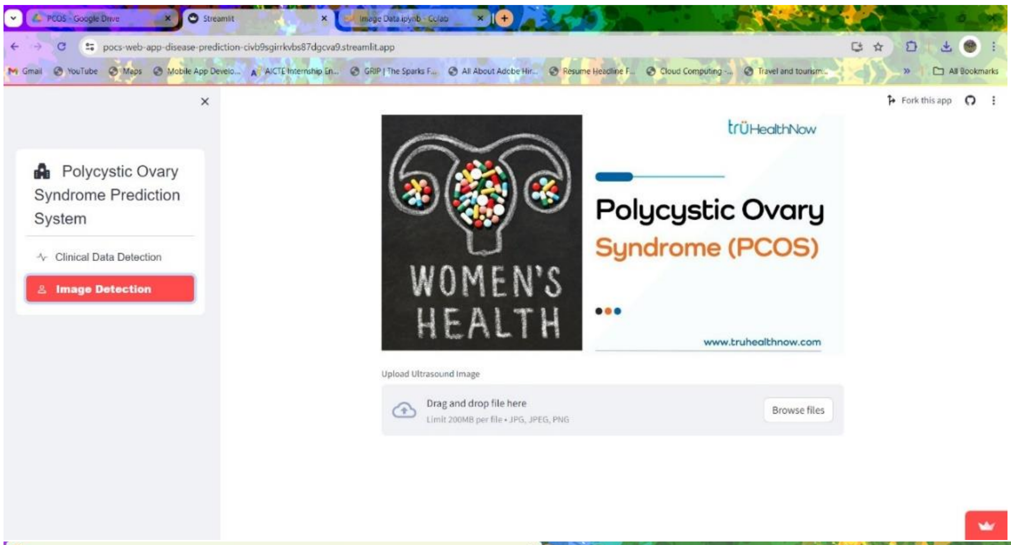
Enter Follicle Right value:(1-25)
1

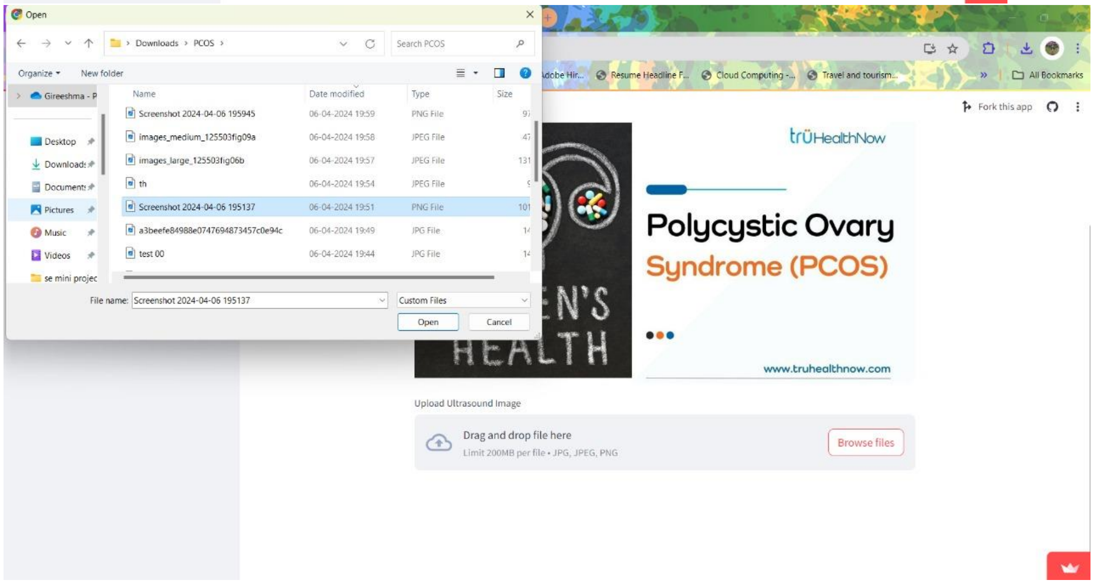
Enter Endometrium value in mm:(0.0-15.0)
0.00

Enter II Beta-HCG value:(1.45-9649)
1.45

The patient is not infected with PCOS

Interactive Webpage for Image Data:


pocs-web-app-disease-prediction-civb9sgirrkvbs87dgcva9.streamlit.app
Polycystic Ovary Syndrome Prediction System
Clinical Data Detection
Image Detection
Fork this app
trüHealthNow
Polycystic Ovary Syndrome (PCOS)
WOMEN'S HEALTH
www.truhealthnow.com
Upload Ultrasound Image
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files

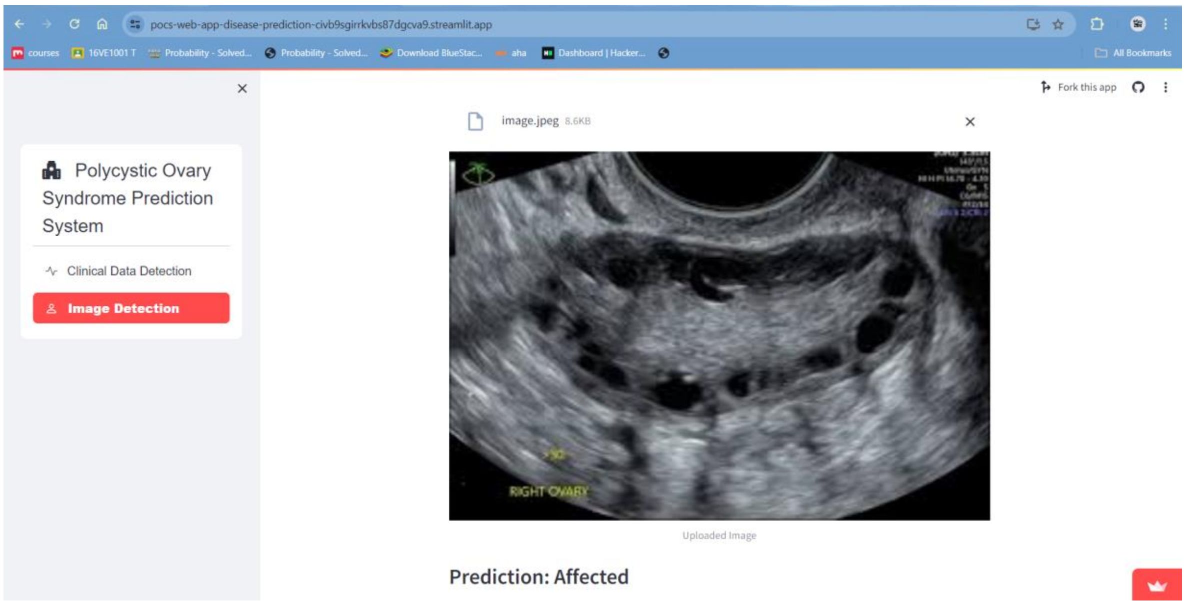


Name	Date modified	Type	Size
Screenshot 2024-04-06 195945	06-04-2024 19:59	PNG File	97
images_medium_125503fig09a	06-04-2024 19:58	JPEG File	47
images_large_125503fig06b	06-04-2024 19:57	JPEG File	131
th	06-04-2024 19:54	JPEG File	9
Screenshot 2024-04-06 195137	06-04-2024 19:51	PNG File	101
a3beefe84988e0747694873457c0e94c	06-04-2024 19:49	JPG File	14
test 00	06-04-2024 19:44	JPG File	14

File name: Screenshot 2024-04-06 195137 — Custom Files — Open — Cancel

- The images will be given.

Output:


pocs-web-app-disease-prediction-civb9sgirrkvbs87dgcva9.streamlit.app
Fork this app
Polycystic Ovary Syndrome Prediction System
Clinical Data Detection
Image Detection
image.jpeg 8.6KB
RIGHT OVARY
Uploaded Image

Prediction: Affected

Fig 5.4: Interactive Webpage

<u>Module -5</u>

5.5. Testing

<u>INTRODUCTION:</u>

Testing is a crucial part of software development that involves the process of evaluating and verifying software products or applications to ensure that they meet the intended requirements and perform as expected. The goal of testing is to identify defects or errors in the software and ensure that it meets the required quality standards, user expectations, and business needs.

Software testing involves executing software components, modules, or entire systems under varying conditions and scenarios to identify any potential issues or bugs. It can be done at different stages of the software development life cycle, including unit testing, integration testing, system testing, and acceptance testing.

Testing helps to improve the quality, reliability, and performance of software systems while reducing risks and ensuring compliance with relevant standards and guidelines. It can also help to increase customer satisfaction and loyalty by ensuring that the software meets the needs and expectations of end-users.

Effective testing requires careful planning, execution, and analysis of test results, as well as the use of appropriate testing methodologies, tools, and techniques. Testing is an ongoing process that requires continuous improvement and adaptation to changing requirements and environments.

Goals and Objectives:

The main goals and objectives of testing are:

Identify defects or errors: The primary objective of testing is to identify defects or errors in the software system. Testing helps to detect issues early in the development cycle, which can save time and money.

Ensure software quality: Testing ensures that the software meets the required quality standards and user expectations. This includes verifying that the software meets functional requirements, performs as expected, and is reliable and secure.

Improve software performance: Testing helps to identify performance bottlenecks and other issues that can impact the software's overall performance. This can lead to improvements in the software's speed, responsiveness, and scalability.

Verify compliance: Testing helps to ensure that the software complies with relevant industry standards, regulations, and guidelines. This includes ensuring that the software meets accessibility and usability requirements, as well as security and data privacy standards.

Reduce risks: Testing helps to reduce the risks associated with software development, including the risk of system failures, data breaches, and other security issues. This can help to protect the company's reputation and prevent financial losses.

Improve customer satisfaction: Testing helps to ensure that the software meets the needs and expectations of end-users. This can lead to increased customer satisfaction, loyalty, and positive reviews.

Overall, the goals and objectives of testing are to improve the quality, reliability, and performance of software systems while reducing risks and ensuring compliance with relevant standards and guidelines.

Design of Testcases and Scenarios:

Levels of Testing

In order to uncover the error's present in the different phases we have the concept of levels of testing.

The basic steps involved in testing are:

UNIT TESTING:

Unit testing is a type of software testing in which individual units or components of software are tested in isolation from the rest of the system to ensure that they function correctly and meet the intended requirements. The purpose of unit testing is to validate that each unit or component of the software works as intended and can be integrated into the larger system without issues.

During unit testing, each unit is tested independently using specialized test cases that focus on a specific function or behaviour of the unit. The tests are usually automated and repeatable to ensure consistency and efficiency.

INTEGRATION TESTING:

Integration testing is a type of software testing that focuses on testing the interfaces and interactions between different modules or components of a software system to ensure that they function correctly and work together as intended. The purpose of integration testing is to detect defects or issues in the system's integration early in the development process and to verify that the individual components work correctly when integrated into the larger system.

Integration testing involves testing different modules or components of a system in various combinations to identify any issues that arise when they interact with each other. The tests are designed to verify that the interfaces between the modules are working correctly and that the system functions as intended when all components are integrated.

There are two main approaches to integration testing: bottom-up integration and top-down integration. In bottom-up integration, the testing starts with individual units or components, and the tests are gradually integrated upward until the entire system is tested. In top-down integration, testing starts with the main system components, and the tests are gradually integrated downward until all the lower-level components are tested.

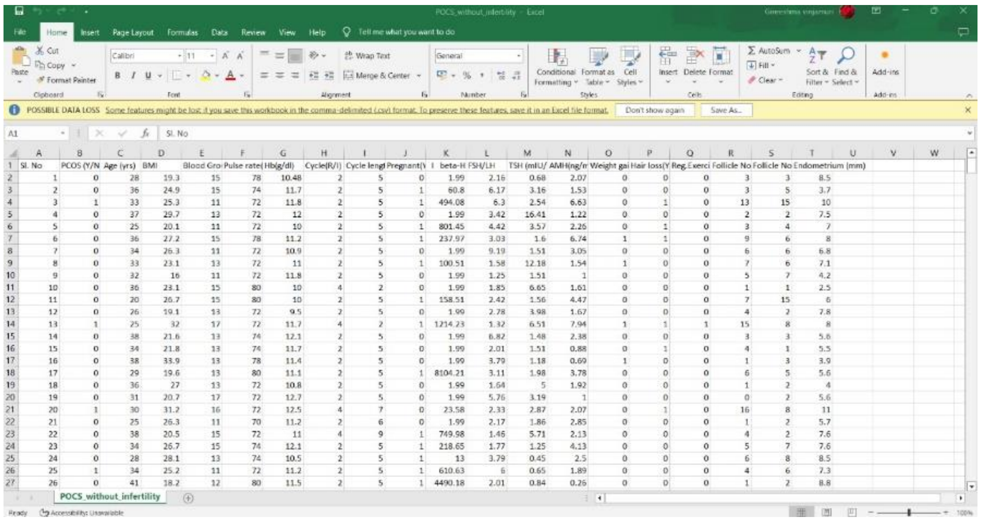
SYSTEM TESTING:

System testing is a type of software testing that focuses on testing the entire software system as whole to ensure that it meets the intended requirements and functions correctly. The purpose of system testing is to validate that the software system meets the user's requirements, business needs, and quality standards.

System testing involves testing the software system in its entirety, including its interfaces, interactions with external systems, performance, security, and other aspects of the system's behavior. The tests are designed to verify that the system functions correctly and performs as intended under various scenarios and conditions.

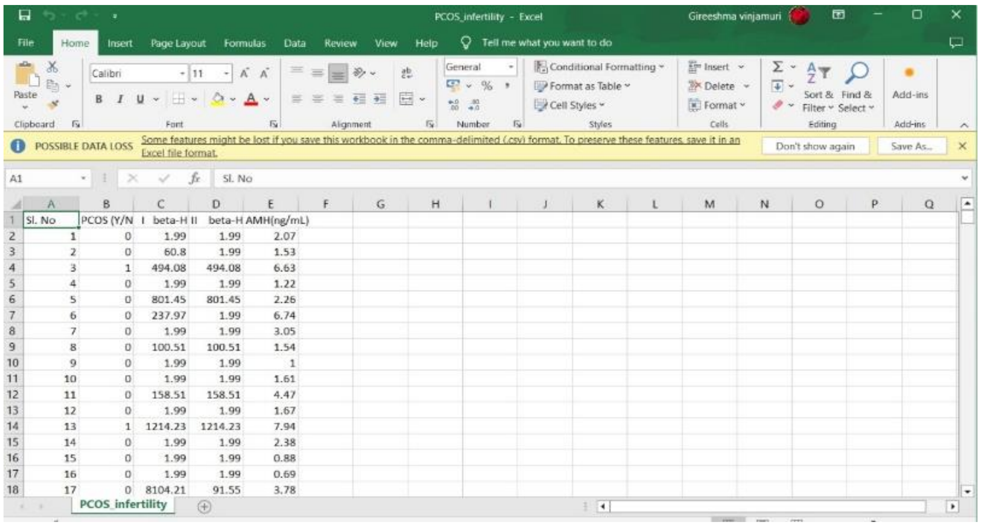
5.5.1 VALIDATION

- We have tested and validated 170 samples of affected and non-affected patients.
- The samples are tested on Random Forest Model and CNN model.
- We have taken around 15% for testing and 15% for validation.



Sl. No	PCOS (Y/N	Age (yrs)	BMI	Blood Gro	Pulse rate(	Hb(g/dl)	Cycle(R/I)	Cycle lengt	Pregnant(Y	I beta-H	FSH/LH	TSH (mIU/	AMH(ng/m	Weight gai	Hair loss(Y	Reg.Exerci	Follicle No	Follicle No	Endometrium (mm)
1	0	28	19.3	15	78	10.48	2	5	0	1.99	2.16	0.68	2.07	0	0	0	3	3	8.5
2	0	36	24.9	15	74	11.7	2	5	1	60.8	6.17	3.16	1.53	0	0	0	3	5	3.7
3	1	33	25.3	11	72	11.8	2	5	1	494.08	6.3	2.54	6.63	0	1	0	13	15	10
4	0	37	29.7	13	72	12	2	5	0	1.99	3.42	16.41	1.22	0	0	0	2	2	7.5
5	0	25	20.1	11	72	10	2	5	1	801.45	4.42	3.57	2.26	0	1	0	3	4	7
6	0	36	27.2	15	78	11.2	2	5	1	237.97	3.03	1.6	6.74	1	1	0	9	6	8
7	0	34	26.3	11	72	10.9	2	5	0	1.99	9.19	1.51	3.05	0	0	0	6	6	6.8
8	0	33	23.1	13	72	11	2	5	1	100.51	1.58	12.18	1.54	1	0	0	7	6	7.1
9	0	32	16	11	72	11.8	2	5	0	1.99	1.25	1.51	1	0	0	0	5	7	4.2
10	0	36	23.1	15	80	10	4	2	0	1.99	1.85	6.65	1.61	0	0	0	1	1	2.5
11	0	20	26.7	15	80	10	2	5	1	158.51	2.42	1.56	4.47	0	0	0	7	15	6
12	0	26	19.1	13	72	9.5	2	5	0	1.99	2.78	3.98	1.67	0	0	0	4	2	7.8
13	1	25	32	17	72	11.7	4	2	1	1214.23	1.32	6.51	7.94	1	1	1	15	8	8
14	0	38	21.6	13	74	12.1	2	5	0	1.99	6.82	1.48	2.38	0	0	0	3	3	5.8
15	0	34	21.8	13	74	11.7	2	5	0	1.99	2.01	1.51	0.88	0	1	0	4	1	5.5
16	0	38	33.9	13	78	11.4	2	5	0	1.99	3.79	1.18	0.69	1	0	0	1	3	3.9
17	0	29	19.6	13	80	11.1	2	5	1	8104.21	3.11	1.98	3.78	0	0	0	6	5	5.6
18	0	36	27	13	72	10.8	2	5	0	1.99	1.64	5	1.92	0	0	0	1	2	4
19	0	31	20.7	17	72	12.7	2	5	0	1.99	5.76	3.19	1	0	0	0	0	2	5.6
20	1	30	31.2	16	72	12.5	4	7	0	23.58	2.33	2.87	2.07	0	1	0	16	8	11
21	0	25	26.3	11	70	11.2	2	6	0	1.99	2.17	1.86	2.85	0	0	0	1	2	5.7
22	0	38	20.5	15	72	11	4	9	1	749.98	1.46	5.71	2.13	0	0	0	4	2	7.6
23	0	34	26.7	15	74	12.1	2	5	1	218.65	1.77	1.25	4.13	0	0	0	5	7	7.6
24	0	28	28.1	13	74	10.5	2	5	1	13	3.79	0.45	2.5	0	0	0	6	8	8.5
25	1	34	25.2	11	72	11.2	2	5	1	610.63	6	0.65	1.89	0	0	0	4	6	7.3
26	0	41	18.2	12	80	11.5	2	5	1	4490.18	2.01	0.84	0.26	0	0	0	1	2	8.8

POCS_without_infertility



Sl. No	PCOS (Y/N	I beta-H	II beta-H	AMH(ng/mL)
1	0	1.99	1.99	2.07
2	0	60.8	1.99	1.53
3	1	494.08	494.08	6.63
4	0	1.99	1.99	1.22
5	0	801.45	801.45	2.26
6	0	237.97	1.99	6.74
7	0	1.99	1.99	3.05
8	0	100.51	100.51	1.54
9	0	1.99	1.99	1
10	0	1.99	1.99	1.61
11	0	158.51	158.51	4.47
12	0	1.99	1.99	1.67
13	1	1214.23	1214.23	7.94
14	0	1.99	1.99	2.38
15	0	1.99	1.99	0.88
16	0	1.99	1.99	0.69
17	0	8104.21	91.55	3.78

PCOS_infertility

Fig 5.5: Real Time Dataset

6. REFERENCES

[1]file:///C:/Users/ANJALI/Downloads/A_Review_on_the_Detection_Techniques_of_Polycystic_240209_110158.pdf

[2] R. M. Dewi, Adiwijaya, U. N. Wisesty, and Jondri, "Classification of polycystic ovary based on ultrasound images using competitive neural network," Journal of Physics: Conference Series, vol. 971, p. 012005, Mar 2018.

[3] A. Denny, A. Raj, A. Ashok, C. M. Ram, and R. George, "i-hope: Detection and prediction system for polycystic ovary syndrome (PCOS) using machine learning techniques," in TENCON 2019-2019 IEEE Region 10 Conference (TENCON). IEEE, 2019, pp. 673–678.

[4] S. Chandrasekaran, "Metabolic syndrome in women with polycystic ovary syndrome," The Obstetrician and Gynecologist , vol. 20, no. 4, pp. 245–252, 2018.

[5] M. Sumathi, P. Chitra, R. Sakthi Prabha, and K. Srilatha, ''Study and detection of PCOS related diseases using CNN,'' IOP Conf. Ser., Mater. Sci. Eng., vol. 1070, Nov. 2021, Art. no. 012062.

[6] R. M. Dewi, Adiwijaya, U. N. Wisesty, and Jondri, "Classification of polycystic ovary based on ultrasound images using competitive neural network," Journal of Physics: Conference Series, vol. 971, p. 012005, Mar 2018.

[7] A. Denny, A. Raj, A. Ashok, C. M. Ram, and R. George, "i-hope: Detection and prediction system for polycystic ovary syndrome (PCOS) using machine learning techniques," in TENCON 2019-2019 IEEE Region 10 Conference (TENCON). IEEE, 2019, pp. 673–678.

[8] S. Chandrasekaran, "Metabolic syndrome in women with polycystic ovary syndrome," The Obstetrician and Gynecologist , vol. 20, no. 4, pp. 245–252, 2018.

[9] M. Sumathi, P. Chitra, R. Sakthi Prabha, and K. Srilatha, ''Study and detection of PCOS related diseases using CNN,'' IOP Conf. Ser., Mater. Sci. Eng., vol. 1070, Nov. 2021, Art. no. 012062.

[10] S. A. Bhat and R. Gupta. (2021). National College of Ireland Project Submission Sheet School of Computing. Accessed: May 16, 2022. [Online].Available: http://norma.ncirl.ie/5137/1/shakoorahmadbhat.pdf

[11] C. Bento. (Sep. 21, 2021). Multilayer Perceptron Explained With aReal-Life Example and Python Code Towards Data Science. Accessed: May 16, 2022. [Online].Available: https://towardsdatascience.com/multilayer-perceptron-explained-with-areal-life-example-and-python-code-sentiment-analysis-cb408ee93141

[12] https://www.kaggle.com/datasets/anaghachoudhari/pcos-detection-using-ultrasound-images

[13] https://www.kaggle.com/datasets/prasoonkottarathil/polycystic-ovary-syndrome-pcos

[14] https://www.mdpi.com/2075-4418/13/8/1506

Polycystic Ovary Syndrome (PCOS) Detection Machine Learning Model Based on Various Feature Selection Methods by R.A. Nasreddine, J. Nasreddine, and M.O. Diab. This paper discusses various feature selection methods and machine learning models for PCOS detection[1].

[15] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9556522/

An extended machine learning technique for polycystic ovary syndrome (PCOS) diagnosis using ultrasound images by S.A. Siddique and M.N. Islam. This study proposes a technique for diagnosing PCOS by automatically segmenting cysts and follicle regions from ultrasound images using machine learning classifiers[2].

[16] https://www.labellerr.com/blog/empowering-diagnosis-detecting-pcos-using-deep-learning-in-ultrasound-images/

Empowering Diagnosis: Detecting PCOS Using Deep Learning in Ultrasound Images by Labelerr. This article provides a detailed guide on building a PCOS detection model using deep learning techniques for ultrasound images[3].

[17] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8828568/

Deep Learning Algorithm for Automated Detection of Polycystic Ovary Syndrome (PCOS) Based on Ultrasound Images by Y. Wang, Y. Shen, R. Fan, et al. This study proposes a deep learning algorithm for automated PCOS detection based on ultrasound images[4].

[18]https://www.frontiersin.org/journals/endocrinology/articles/10.3389/fendo.2024.1298628/full

[19] https://www.frontiersin.org/journals/endocrinology/articles/10.3389/fendo.2021.789878/full

The paper "Deep Learning Algorithm for Automated Detection of Polycystic Ovary Syndrome Using Scleral Images" presents a deep learning algorithm for the auxiliary detection of PCOS using scleral images. The algorithm utilizes Resnet18 as the feature extraction network and achieves good AUC, accuracy, and other indicators. The paper also uses Grad-CAM, a popular CNN explanation tool, to highlight the features that are important for PCOS detection.

[20]https://www.frontiersin.org/journals/endocrinology/articles/10.3389/fendo.2024.1298628/full

[21] https://ieeexplore.ieee.org/iel7/6287639/10005208/10214584.pdf

Thara, A., et al. (2022). A Review on the Detection Techniques of Polycystic Ovary Syndrome Using Machine Learning. IEEE Access, 10, 79717-79727.

[22] https://www.mdpi.com/2075-4418/13/8/1506

Nasreddine, R.A., et al. (2022). Polycystic Ovary Syndrome Detection Machine Learning Model Based on Optimized Feature Selection and Explainable Artificial Intelligence. Diagnostics, 13(8), 1506.

[23] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9556522/

Siddique, S.A., & Islam, M.N. (2022). An extended machine learning technique for polycystic ovary syndrome detection using ovary ultrasound image. Journal of Healthcare Engineering, 2022, 1-12.

[24] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8828568/

Wang, Y., et al. (2022). Deep Learning Algorithm for Automated Detection of Polycystic Ovary Syndrome Based on Ultrasound Images. Computational and Mathematical Methods in Medicine, 2022, 1-11.

[25] https://www.sciencedirect.com/science/article/pii/S2405844023017255

Khanna, V.V., et al. (2023). Exploring the dominant features and data-driven detection of polycystic ovary syndrome through modified stacking ensemble machine learning technique. Journal of Ambient Intelligence and Humanized Computing, 1-15.

[26] https://www.nature.com/articles/s41598-022-21724-0

The study "An extended machine learning technique for polycystic ovary syndrome detection using ovary ultrasound image" compares the performance of deep learning models and traditional machine learning models for PCOS detection using ultrasound images. The study finds that deep learning models achieve higher accuracy but take longer execution time. To overcome this, the study proposes a hybrid model that combines the feature extraction phase of deep learning models with traditional machine learning models. The hybrid model achieves higher accuracy in a shorter amount of time.

[27] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8828568/

The paper "Deep Learning Algorithm for Automated Detection of Polycystic Ovary Syndrome" presents a deep learning architecture for PCOS detection based on CNN. The architecture is composed of feature extraction and classification. The feature extraction phase uses a pre-trained CNN model to extract features from ultrasound images. The classification phase uses a fully connected neural network to classify the extracted features. The paper also discusses the limitations of the study, including the need for a larger and more diverse dataset.

7.APPENDIX

Code for Report Data:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

from google.colab import drive
drive.mount('/content/drive')

d1=pd.read_csv("/content/drive/MyDrive/PCOS/PCOS_infertility.csv")
d2= pd.read_csv("/content/drive/MyDrive/PCOS/POCS_without_infertility.csv")

d1
d2

# Merging the two files as per patient file no.
# The files were sorted into two based on patients with infertility and without infertility

data = pd.merge(d2,d1, on='Sl. No', suffixes={'','_wo'},how='left')

data.columns

#Dropping the repeated features after merging.

data =data.drop([ 'PCOS (Y/N)_wo',' I   beta-HCG(mIU/mL)_wo','AMH(ng/mL)_wo',],
axis=1)
```

```
#'Unnamed: 41', 'Weight (Kg)' ,'Height(Cm) ' ,

# Changing the title of the properties.

data = data.rename(columns = {"PCOS (Y/N)":"Target"})

data.head()

# Dropping unnecessary features.

data = data.drop(["Sl. No"],axis = 1)

data.info(verbose = True, null_counts = False)

# Having a look at the data with dtype objects.

data["AMH(ng/mL)"].head()

data["II    beta-HCG(mIU/mL)"].head()

# Dealing with categorical values.
# In this database the type objects are numeric values saved as strings.
# So I am just converting it into a numeric value.

data["AMH(ng/mL)"] = pd.to_numeric(data["AMH(ng/mL)"], errors='coerce')

data['BMI'] = pd.to_numeric(data['BMI'], errors='coerce')

data['FSH/LH'] = pd.to_numeric(data['FSH/LH'], errors='coerce')

data["II    beta-HCG(mIU/mL)"] = pd.to_numeric(data["II    beta-HCG(mIU/mL)"],
errors='coerce')
```

```
# Dealing with missing values.
# Filling NA values with the median of that feature.

data['AMH(ng/mL)'].fillna(data['AMH(ng/mL)'].median(),inplace=True)

data.dropna(subset=['BMI'], inplace=True)

data.dropna(subset=['FSH/LH'], inplace=True)

#data['Marraige Status (Yrs)'].fillna(data['Marraige Status (Yrs)'].median(),inplace=True)

data['II    beta-HCG(mIU/mL)'].fillna(data['II    beta-
HCG(mIU/mL)'].median(),inplace=True)

#data['Fast food (Y/N)'].fillna(data['Fast food (Y/N)'].median(),inplace=True)

# Clearing up the extra space in the column names.

data.columns = [col.strip() for col in data.columns]

data

#Assiging the features (X)and target(y).

X= data.drop("Target",axis = 1)
y=data["Target"]

#Splitting the data into test and training sets.

X_train,X_test, y_train, y_test = train_test_split(X,y, test_size=0.3)

#model = XGBRFClassifier(n_estimators=100, random_state=42)
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Results for Random Forest Classifier:")
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-score: {f1}")
print(f"ROC-AUC: {roc_auc}")
print("\nConfusion Matrix:")
print(conf_matrix)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d")
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix - Random Forest Classifier')
plt.show()

data.head(26)
```

Predicting System:

```
    # Convert input data to a numpy array
#input_data =
(34,25.23634033,11,72,20,11.2,2,5,12.0,1,0,5.34,0.89,6.0,38,32,0.842105263,0.65,1.89,11.46,21
.5,0.98,100,0,1,0,0,1,1,0,120,70,4,6,18,17,7.3,610.63,610.63)
```

```
input_data  = (34,25.2,11,72,11.2,2,5,1,610.63,6.0,0.65,1.89,0,0,0,4,6,7.3,610.63)
input_data_as_numpy_array = np.asarray(input_data)

# Reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)

# Make prediction using the classifier
prediction = model.predict(input_data_reshaped)

# Output the prediction
if prediction[0] == 0:
    print('The patient is not infected with PCOS')
else:
    print('The patient is infected with PCOS')
```

Saving the trained model:

```
import joblib

filename = 'POCS.sav'
joblib.dump(model, open(filename, 'wb'))

# loading the saved model
loaded_model = joblib.load(open('POCS.sav', 'rb'))

input_data  = (36,24.9,15,74,11.7,2,5,1,60.8,6.17,3.16,1.53,0,0,0,3,5,3.7,1.99)
# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = loaded_model.predict(input_data_reshaped)
print(prediction)

# Output the prediction
if prediction[0] == 0:
    print('The patient is not infected with PCOS')
else:
    print('The patient is infected with PCOS')
```

Code for Image Data:

```
import numpy as np

import matplotlib.pyplot as plt

import os

import math

import shutil

import glob
import zipfile

from google.colab import drive
drive.mount('/content/drive')

ROOT_DIR = '/content/drive/MyDrive/PCOS/PCOS'
number_of_images = {}

for dir in os.listdir(ROOT_DIR):
   number_of_images[dir] = len(os.listdir(os.path.join(ROOT_DIR,dir)))
   print("",dir,"" ,number_of_images[dir])

from keras.preprocessing.image import ImageDataGenerator
from keras.applications.mobilenet import preprocess_input

def preprocessingImage1(path):
  image_data = ImageDataGenerator(zoom_range=0.2,shear_range=0.2,preprocessing_function=
preprocess_input,horizontal_flip=True)
  image =
image_data.flow_from_directory(directory=path,target_size=(224,224),batch_size=32,class_mode='bi
nary')
  return image

def preprocessionfImage2(path):
  """
  Input :path
  Output : preprocessed Image
  """
  image_data  = ImageDataGenerator(preprocessing_function= preprocess_input )
  image =
image_data.flow_from_directory(directory=path,target_size=(224,224),batch_size=32,class_mode='bi
nary')
return image
```

We will split the data such that 70% for training 15 % for vailadation 15 % for testing

```
def datafolder(path,split):
  if not os.path.exists("./"+path):
    os.mkdir("./"+path)

    for dir in os.listdir(ROOT_DIR):
      os.makedirs("./"+path+"/"+dir)
      for img in np.random.choice(a=os.listdir(os.path.join(ROOT_DIR,dir)),
                              size=(math.floor(split * number_of_images[dir])-5),replace=False):

          O = os.path.join(ROOT_DIR,dir,img)
          D = os.path.join("./"+path,dir)
          shutil.copy(O,D)
          os.remove(O)

  else:
     print("Folder already exist")

datafolder("train",0.7)

datafolder("val",0.15)

datafolder("test",0.15)

path ='/content/train'
train_data = preprocessingImage1(path)

path ='/content/test'
test_data = preprocessionfImage2(path)

path = '/content/val'
val_data = preprocessionfImage2(path)
```

Model Block

```
import numpy as np
import matplotlib.pyplot as plt
from keras.layers import Flatten,Dense
from keras.models import Model,load_model
from keras.applications.mobilenet import MobileNet
import keras

base_model = MobileNet(input_shape=(224,224,3),include_top=False)

for layer in base_model.layers:
  layer.trainable = False

x= Flatten()(base_model.output)
x= Dense(units=1,activation='sigmoid')(x)
```

```
model = Model(base_model.input,x)

model.compile(optimizer='rmsprop',loss=keras.losses.binary_crossentropy,metrics=['accuracy'])


from keras.callbacks import ModelCheckpoint,EarlyStopping

mc =
ModelCheckpoint(filepath="bestmodel.h5",monitor='val_accuracy',verbose=1,save_best_only=True)

#Early check points
es = EarlyStopping(monitor="val_accuracy",min_delta=0.01,patience=5,verbose=1)

cb = [mc,es]

hist = model.fit_generator(train_data,
                           steps_per_epoch=5,
                           epochs=70,
                           validation_data=val_data,
                           validation_steps=20,
                          callbacks=cb)

model = load_model("/content/bestmodel.h5")

acc = model.evaluate_generator(train_data)[1]
print(f"our model accuracy is  {acc * 100} %")

acc = model.evaluate_generator(test_data)[1]
print(f"our model accuracy is  {acc * 100} %")


precision = model.evaluate_generator(test_data)[1]
print(precision)

recall= model.evaluate_generator(test_data)

f1score= 2*((precision*recall)/(precision+recall))
h = hist.history
h.keys()

train_data.class_indices

test_data.class_indices

val_data.class_indices

#now get some random images and predict the class
from keras.preprocessing import image
import tensorflow as tf

def predictimage(path):

     img = tf.keras.utils.load_img(path,target_size=(224,224))
```

```
    i = tf.keras.utils.img_to_array(img)/255
    input_arr= np.array([i])
    input_arr.shape

    pred =model.predict(input_arr)
    if pred == 1:
       print("Not Affected")
    else :
      print("Affected")
    #display image
    plt.imshow(input_arr[0],vmin=0, vmax=255)
    plt.title("given Image")
    plt.show()

#It is the infected image sample
predictimage("/content/drive/MyDrive/PCOS/PCOS/infected/img7.jpg")

#It is the not affected sample images
predictimage("/content/drive/MyDrive/PCOS/PCOS/notinfected/img_0_100.jpg")

#not infected image
predictimage("/content/drive/MyDrive/PCOS/PCOS/notinfected/img3.jpg")

predictimage("/content/drive/MyDrive/PCOS/PCOS/infected/img3.jpg")
```

Code for Interactive WebPage:

```
import streamlit as st
import numpy as np
from PIL import Image
import tensorflow as tf
from tensorflow.keras.models import load_model
from streamlit_option_menu import option_menu
import zipfile
import joblib

# Function to extract and load the trained image detection model
def load_image_model(zip_file_path):
    try:
        # Extract the model from the zip file
        with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
            zip_ref.extractall("temp_image_model")

        # Load the model
        model = load_model("temp_image_model/pcos_detection_model.keras")

        return model

    except Exception as e:
        st.error(f"Error loading image detection model: {e}")
        return None

# Load the trained image detection model
image_model = load_image_model("pcos_detection_model.zip")
```

```
# Function to load the saved clinical data detection model
def load_clinical_model(model_path):
    try:
        model = joblib.load(open(model_path, 'rb'))
        return model

    except Exception as e:
        st.error(f"Error loading clinical data detection model: {e}")
        return None

# Load the saved clinical data detection model
clinical_model_path = 'POCS.sav'
clinical_model = load_clinical_model(clinical_model_path)

# Function for clinical data prediction
def pcos_prediction(input_data):
    try:
        input_data_as_numpy_array = np.asarray(input_data, dtype=np.float64)
        input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)
        prediction = clinical_model.predict(input_data_reshaped)
        return 'The patient is not infected with PCOS' if prediction[0] == 0 else 'The patient is infected with
PCOS'

    except Exception as e:
        st.error(f"Error predicting clinical data: {e}")
        return None

# Function to predict image
def predict_image(path):
    try:
        img = Image.open(path)
        img = img.resize((224, 224))  # Resize image to match model input size
        img_array = tf.keras.preprocessing.image.img_to_array(img) / 255
        input_arr = np.array([img_array])
        pred = image_model.predict(input_arr)
        return "Affected" if pred[0][0] < 0.5 else "Not Affected"

    except Exception as e:
        st.error(f"Error predicting image: {e}")
        return None

def main():
    st.title('PCOS Detection Web App')
    with st.sidebar:
        selected = option_menu('Polycystic Ovary Syndrome Prediction System',

                              ['Clinical Data Detection',
                               'Image Detection'],
                              menu_icon='hospital-fill',
                              icons=['activity', 'person'],
                              default_index=0)


    if selected == "Clinical Data Detection":
        st.header("Clinical Data Detection")
        image = Image.open('pocs image.png')
        st.image(image)
        # Add input fields for clinical data
        age = st.number_input('Enter Age in years:(12-50)',12,50)
        bmi = st.number_input('Enter BMI Value:(0.0-81.0)',0.0,81.0)
        bloodgroup = st.number_input('Enter Blood Group Value:(11-17)',11,17)
        pulserate = st.number_input('Enter Pulse rate in bpm:(60-100)',60,100)
        hb = st.number_input('Enter Hb in g/dl:(8.0-15.1)',8.0,15.1)
        cycle = st.number_input('Enter Cycle in R/I:(2-5)',2,5)
        cyclelength = st.number_input('Enter Cycle length in days:(2-20)',2,20)
        pregnant = st.number_input('Enter Pregnant value :(0-1)',0,1)
        ibetahcg = st.number_input('Enter I Beta-HCG value:(1.45-9649.0)',1.45,9649.0)
        fshlh = st.number_input('Enter FSH/LH value:(1.0-1400.0)',1.0,1400.0)
        tsh = st.number_input('Enter TSH value:(0.0-25.0)',0.0,25.0)
        amh = st.number_input('Enter AMH value:(0.0-25.0)',0.0,25.0)
        weightgain = st.number_input('Enter Weight gain value :(0 or 1)',0,1)
        hairloss = st.number_input('Enter Hair loss value :(0 or 1)',0,1)
        regexercise = st.number_input('Enter Regular Exercise value :(0 or 1)',0,1)
        follicleleft = st.number_input('Enter Follicle Left value:(1-25)',1,25)
        follicleright = st.number_input('Enter Follicle Right value:(1-25)',1,25)
        endometrium = st.number_input('Enter Endometrium value in mm:(0.0-15.0)',0.0,15.0)
        iibetahcg = st.number_input('Enter II Beta-HCG value:(1.45-9649)',1.45,9649.0)

        if st.button('PCOS Test Result'):
            if clinical_model:
                diagnosis = pcos_prediction([age, bmi, bloodgroup, pulserate, hb, cycle, cyclelength, pregnant,
ibetahcg, fshlh, tsh, amh, weightgain, hairloss, regexercise, follicleleft, follicleright, endometrium,
iibetahcg])
                if diagnosis:
                    st.success(diagnosis)
            else:
                st.error("Error: Clinical model not loaded.")

    elif selected == "Image Detection":
        st.header("Image Detection")
        image = Image.open('pocs image.png')
        st.image(image)
        uploaded_file = st.file_uploader("Upload Ultrasound Image", type=["jpg", "jpeg", "png"])

        if uploaded_file is not None:
            # Display the uploaded image
            st.image(uploaded_file, caption="Uploaded Image", use_column_width=True)

            # Make prediction on the uploaded image
            if image_model:
                prediction = predict_image(uploaded_file)
                if prediction:
                    st.markdown(f"<h3>Prediction: {prediction}</h3>", unsafe_allow_html=True)
            else:
                st.error("Error: Image detection model not loaded.")

if __name__ == "__main__":
    main()
```