

جامعة سيدي محمد بن عبد الله بفاس  
ⵜⴰⵎⴰⵎⴰⵔⵜ ⴰⵎⴰⵏⴰ ⴰⵏⴰⵔⴰⵏ ⴰⵏⴰⵔⴰⵏ ⴰⵏⴰⵔⴰⵏ ⴰⵏⴰⵔⴰⵏ  
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH DE FES  
المدرسة الوطنية للعلوم التطبيقية - فاس  
ⵜⴰⵎⴰⵎⴰⵔⵜ ⴰⵎⴰⵏⴰ ⴰⵏⴰⵔⴰⵏ ⴰⵏⴰⵔⴰⵏ ⴰⵏⴰⵔⴰⵏ  
Ecole Nationale des Sciences Appliquées de Fès



---

# ***Research Internship Report***

---

## **Project Title: Health AI Disease Detection**

Submitted by:  
**Hassane Skikri**

Under the Guidance of  
**Safae Elhajbenali**

Project Duration:

**1 month**

Major: Computer Science at National  
School of Applied science in Fez

Promotion: 2023/2014



# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to several individuals and organizations for their support and contributions to my research internship project.

First and foremost, I extend my heartfelt thanks to my supervisor, Professor Safae Elhajbenali, for her invaluable guidance, continuous support, and encouragement throughout the duration of my internship. Her expertise and insights were instrumental in shaping the direction and outcomes of this project.

I also wish to acknowledge Krish Naik and his YouTube channel [Krish Naik \(@krishnaik06\)](#), which provided a wealth of information and tutorials on data science. The content on his channel significantly enhanced my understanding of various data science concepts and techniques, which were crucial to the success of my project.

Additionally, I am grateful for the [AI for Medical Diagnosis](#) course on Coursera, which offered comprehensive and valuable insights into the application of artificial intelligence in medical diagnosis. The knowledge gained from this course was pivotal in developing the AI models used in this project.



# Abstract

This project aims to revolutionize disease diagnosis using AI technology. Our primary objective is to develop a platform that swiftly and accurately detects diseases, addressing issues such as delayed diagnosis, limited access to healthcare, high costs, and misdiagnosis. We initially focus on six major diseases: Brain Tumor, Alzheimer's, Covid-19, pneumonia, breast cancer, and diabetes. The platform leverages ensemble modeling for traditional problems and Convolutional Neural Networks (CNNs) for image-based diagnosis. By utilizing medical images and data files, we train AI models to ensure precise detection. This AI platform is designed to provide rapid and reliable results, facilitating quicker patient treatment, reducing healthcare costs, and enhancing the accessibility of medical diagnostics. Future plans include expanding the platform to diagnose a broader range of diseases.



# Table of Content

## Table of Contents

1.	Introduction .....	1
2.	Key Concepts and Techniques in Artificial Intelligence .....	2
2.1	What is artificial intelligence (AI)? .....	3
2.2	The History of AI .....	3
2.3	Machine Learning? .....	4
2.4	Ensemble Learning? .....	4
2.5	Deep Learning? .....	5
2	What is XGBoost? .....	5
3	Convolutional Neural Networks (CNN) .....	6
4	Tools and Methods Used in Health AI disease detection .....	9
5.1	Programming Languages .....	9
5.2	Libraries and Frameworks .....	9
5.3	Development Environment .....	10
5.4	Version Control .....	10
5.5	Frontend Development .....	11
5.6.	Data Collection and Preprocessing .....	11
5.7.	Modeling .....	12
5.8.	Evaluation .....	12
5	Project Development Roadmap .....	12
5.2	Create a virtual environment and install the dependencies .....	12
6.2	Create the notebooks and download the models then save them in google drive .....	13
5.3	cerate the index.html .....	13
6.3	create the html file for every disease .....	16
6.4	Style the Application .....	20
6.5	create some animation using js .....	22
7.	Create the flask application .....	22
7.1	import the libraries: .....	22
7.2	Creating Custom Transformers for Diabetes detection .....	23
7.3.	Flask App Configuration .....	24
7.4	Utility Functions .....	25
7.4	Define the Home and prediction Routes .....	25
7.5	Handle Image-Based Disease Prediction .....	26
7.6	Handle Breast Cancer Prediction .....	27

7.7 Handle Diabetes Prediction.....	28
7.8 Running the Flask app.....	29
8. Project UI.....	29
8.1 Homepage.....	30
8.2 Brain Tumor.....	31
8.3 Covid 19.....	32
8.4 pneumonia.....	33
8.5 Alzheimer.....	34
8.6 diabetes.....	35
9.Conclusion.....	36
References.....	37



# List of Figure

Figure 1:The difference between (AI), (ML), (DL)..... 2

Figure 2: Flow chart of XGBoost (ResearchGate)..... 6

Figure 3: CNN architecture (theclickreader) ..... 8



# 1. Introduction

Artificial intelligence (AI) is revolutionizing healthcare by improving patient care, reducing costs, and enhancing efficiency. In recent years, AI technology has advanced rapidly, providing healthcare organizations with numerous tools and strategies. To make the best use of these technologies, it's important to understand how AI works, including how it processes, analyzes, and delivers data. This knowledge ensures that AI solutions are accurate, reliable, and worth the investment.

Healthcare organizations must be well-informed about AI to choose the right products or hire skilled data scientists to develop AI systems in-house. Deep learning, a powerful branch of AI, has quickly become a major player in healthcare. It can analyze vast amounts of data with unprecedented speed and accuracy.

One type of deep learning, called Convolutional Neural Networks (CNNs), is particularly effective at analyzing medical images. CNNs are designed to process large images efficiently, making them ideal for examining medical scans like MRIs and X-rays. According to experts at Stanford University, CNNs are built specifically for image processing, enabling them to work faster and more accurately than other AI methods.

CNNs have demonstrated exceptional effectiveness in analyzing medical images, often matching or surpassing human doctors in identifying critical features. This high level of accuracy makes CNNs invaluable in clinical settings, where they assist doctors in making better diagnoses. By enhancing the speed and accuracy of medical image analysis, CNNs significantly improve patient care.

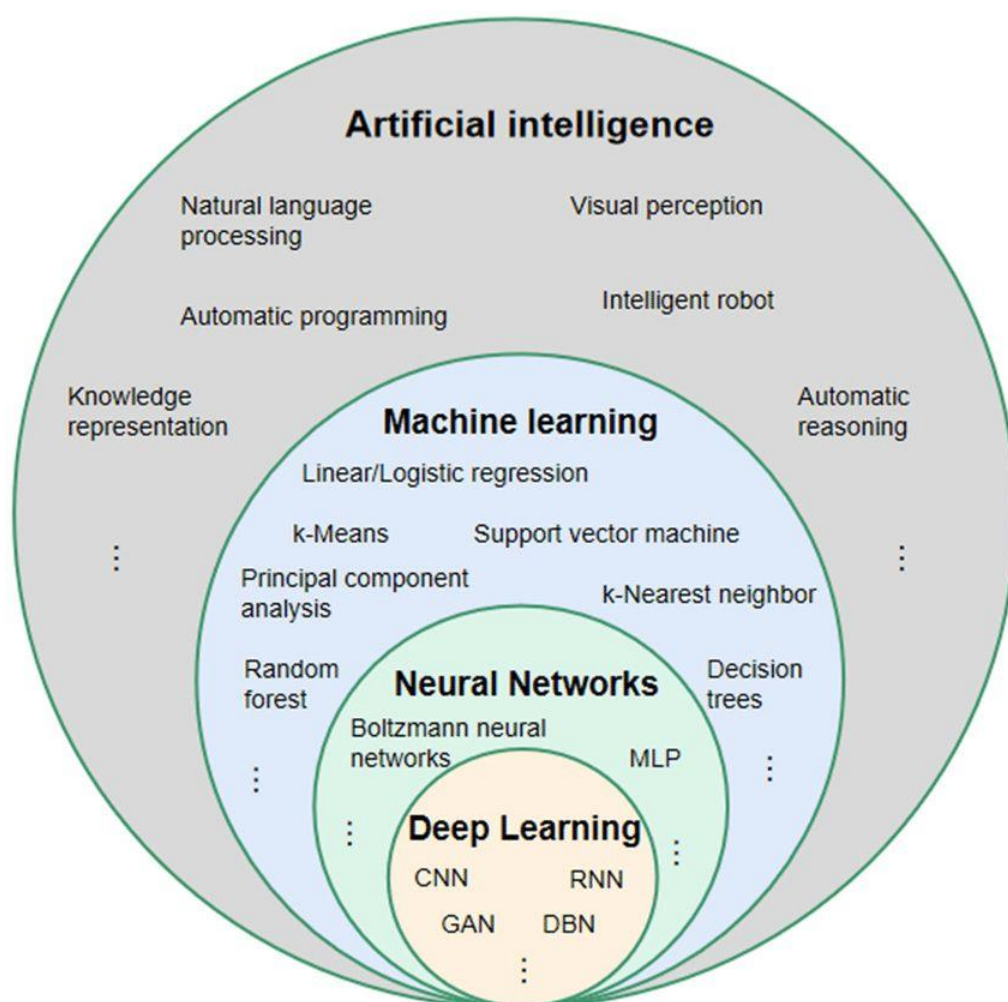
Another crucial AI technique in healthcare is ensemble learning. Ensemble learning involves combining multiple machine learning models to improve the overall performance and accuracy of predictions. This approach leverages the strengths of different models to create a more robust and reliable system.

Ensemble learning can be applied to various types of data, including medical records, patient histories, and diagnostic tests. By using multiple models, ensemble learning reduces the risk of errors and increases the chances of accurate predictions. This is particularly important in healthcare, where accurate diagnoses can have a significant impact on patient outcomes.

For example, in diagnosing diseases like diabetes, an ensemble learning approach might combine several models, each focusing on different aspects of patient data. One model could analyze blood sugar levels, another could look at family history, and yet another could consider lifestyle factors. By integrating the results from these models, the ensemble learning system can provide a more comprehensive and accurate diagnosis.

In summary, AI, particularly deep learning and ensemble learning, is transforming healthcare. These technologies offer new ways to improve diagnosis accuracy, reduce costs, and provide better care. As AI continues to evolve, healthcare organizations need to stay informed about these advancements to fully realize their potential. By integrating advanced AI techniques like CNNs and ensemble learning, healthcare can become more efficient, accurate, and accessible, ultimately benefiting patients worldwide.

## 2. Key Concepts and Techniques in Artificial Intelligence



**Figure 1: The difference between (AI), (ML), (DL).**  
(LinkedIn - Eric Vyacheslav)

## 2.1 What is artificial intelligence (AI)?

Artificial Intelligence (AI) is a branch of computer science that aims to create systems capable of performing tasks that typically require human intelligence. These tasks include problem-solving, learning, understanding natural language, recognizing patterns, and making decisions. AI systems leverage various techniques, including machine learning, neural networks, deep learning, and Natural language processing (NLP) to process and analyze data, learn from it, and apply that knowledge to achieve specific goals.

AI is widely applied across industries: in healthcare for diagnostics and personalized treatment plans; in Finance for fraud detection and tailored banking services; in Retail to enhance customer experiences with personalized recommendations; in Manufacturing to optimize production processes and quality control; and in Transportation through autonomous vehicles and traffic management systems.

AI is transforming how we live and work by automating routine tasks, enhancing decision-making processes, and enabling new capabilities that were previously thought to be exclusive to human intelligence

## 2.2 The History of AI

**1956:** John McCarthy coins the term "artificial intelligence" at the first AI conference at Dartmouth College. Allen Newell, J.C. Shaw, and Herbert Simon create the Logic Theorist, the first AI software program.

**1967:** Frank Rosenblatt builds the Mark 1 Perceptron, the first neural network computer that learns through trial and error. Marvin Minsky and Seymour Papert publish "Perceptions," which initially halts neural network research.

**1980s:** Neural networks using backpropagation become popular in AI applications.

**1995:** Stuart Russell and Peter Norvig publish "Artificial Intelligence: A Modern Approach," a leading AI textbook.

**1997:** IBM's Deep Blue defeats world chess champion Garry Kasparov.

**2004:** John McCarthy publishes "What Is Artificial Intelligence?" defining AI.

**2011:** IBM Watson wins against Jeopardy! champions Ken Jennings and Brad Rutter.

**2015:** Baidu's Minwa supercomputer uses a convolutional neural network to categorize images more accurately than humans.

**2016:** DeepMind's AlphaGo, using a deep neural network, beats world champion Go player Lee Sodol. Google buys DeepMind for \$400 million.

**2023:** Large language models like ChatGPT revolutionize AI, significantly enhancing its performance and potential for enterprise use.

## 2.3 Machine Learning?

Machine Learning (ML) is a branch of artificial intelligence (AI) that focuses on developing systems that can learn from and make decisions based on data. Rather than being explicitly programmed to perform a task, these systems learn from patterns and insights derived from data. ML can be divided into three main types: supervised learning, unsupervised learning, and reinforcement learning.

**Supervised Learning:** This type involves training a model on a labeled dataset, which means the data comes with input-output pairs. The model learns to predict the output from the input data. Examples include linear regression, logistic regression, and support vector machines.

**Unsupervised Learning:** In unsupervised learning, the model is given data without labeled responses. The goal is to uncover hidden patterns or intrinsic structures in the input data. Techniques include clustering (e.g., k-means) and dimensionality reduction (e.g., principal component analysis).

**Reinforcement Learning:** This involves training an agent to make a sequence of decisions by rewarding it for good actions and penalizing it for bad ones. The agent learns to maximize cumulative rewards through trial and error. Applications include robotics, gaming, and autonomous vehicles.

## 2.4 Ensemble Learning?

Ensemble Learning is a technique that involves combining multiple machine learning models to produce a better performing model. The key idea is to leverage the strengths and compensate for the weaknesses of different models.

**Random Forest:** An ensemble method that builds multiple decision trees during training and merges their outputs for improved accuracy and robustness. It is particularly useful for classification and regression tasks.

**XGBoost:** An optimized implementation of gradient boosting that builds a series of trees, where each tree corrects the errors of the previous ones. XGBoost is known for its high performance and efficiency.

**Benefits of Ensemble Learning:** By combining different models, ensemble learning reduces the risk of overfitting and improves the generalizability of the model. It is particularly useful in scenarios where the accuracy of predictions is crucial, such as in healthcare diagnostics.

## 2.5 Deep Learning?

Deep Learning is a subset of ML that uses neural networks with many layers (hence "deep") to analyse various types of data. Deep learning models, particularly those with multiple layers, are highly effective at capturing complex patterns in data.

**Artificial Neural Networks (ANNs):** These are inspired by the human brain and consist of interconnected layers of nodes (neurons). ANNs are used for tasks like pattern recognition and data classification.

**Deep Neural Networks (DNNs):** These networks have multiple hidden layers between the input and output layers, allowing them to model more complex relationships. DNNs are used in applications such as speech recognition and image processing.

**Convolutional Neural Networks (CNNs):** A type of DNN designed specifically for processing structured grid data like images. CNNs use convolutional layers that apply filters to the input data, making them highly effective for image recognition tasks.

Our project will focus on XGBoost and CNN, so let's explain them clearly.

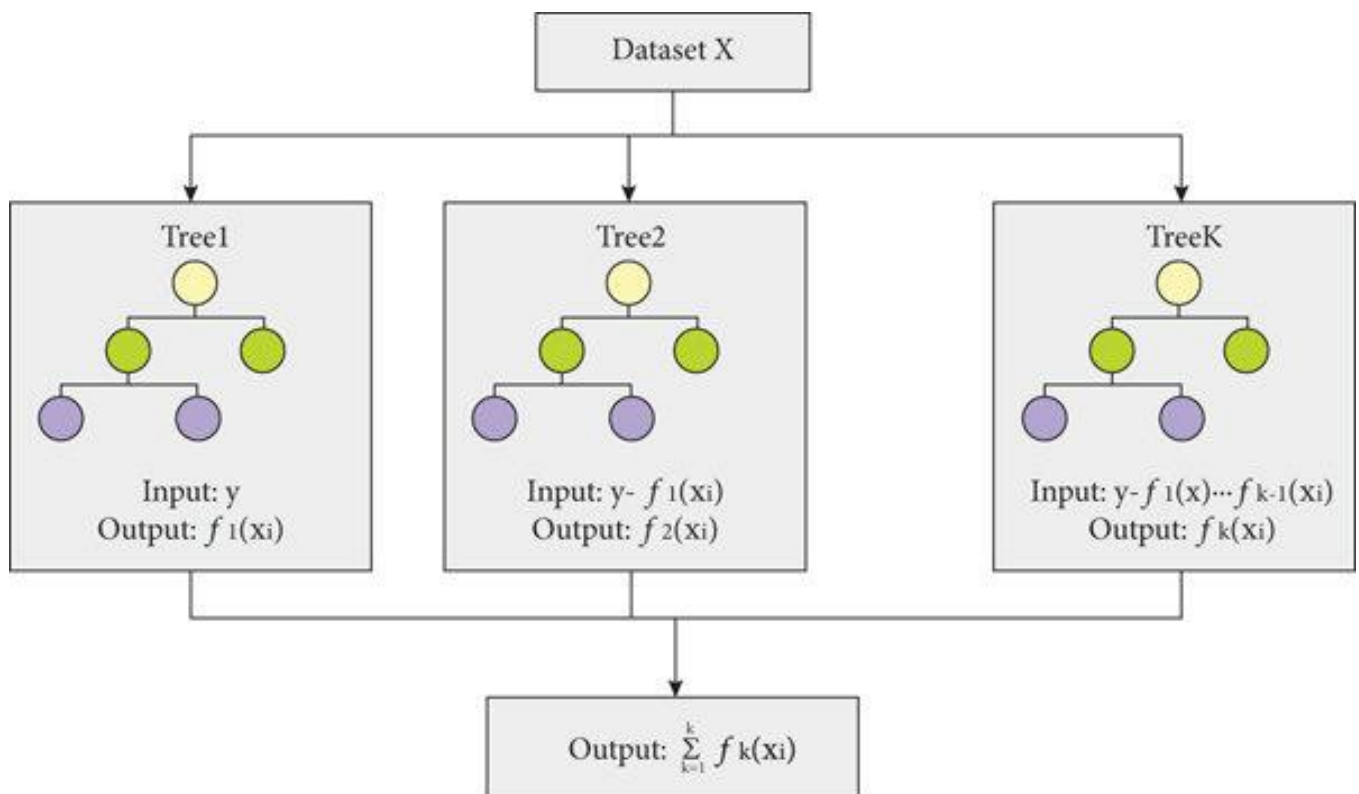
## 2 What is XGBoost?

Boosting is an ensemble modelling, technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now. XGBoost works as Newton-Raphson in function space unlike gradient boosting that works as gradient descent in function space, a second order Taylor approximation is used in the loss function to make the connection to Newton Raphson method.



**Figure 2: Flow chart of XGBoost (ResearchGate)**

### 3 Convolutional Neural Networks (CNN)

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of artificial neural network (ANN), most commonly applied to analyze visual imagery. CNNs are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses known as feature maps[2]. Counter-intuitively,

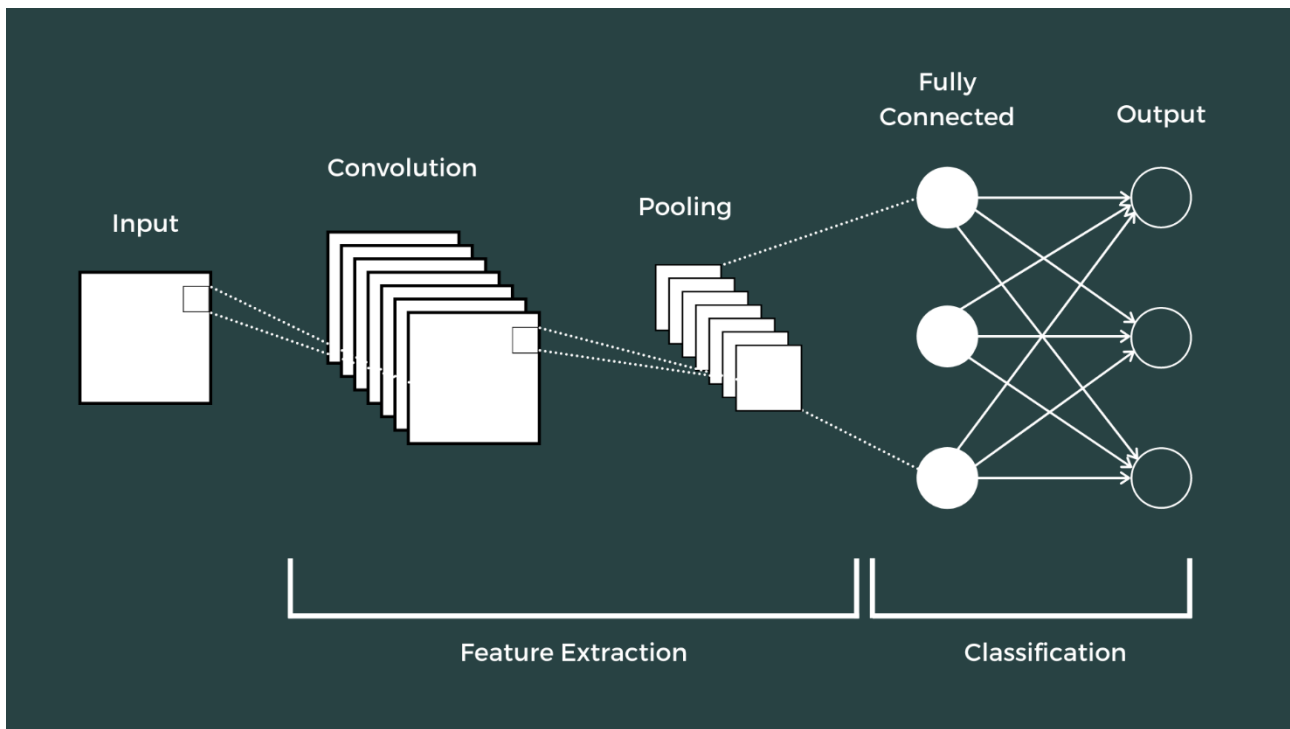


most convolutional neural networks are not invariant to translation, due to the downsampling operation they apply to the input[3]. They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain–computer interfaces, and financial time series.

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

- The construction of a convolutional neural network is a multi-layered feed forward neural network, made by assembling many unseen layers on top of each other in a particular order.
- It is the sequential design that gives permission to CNN to learn hierarchical attributes.
- In CNN, some of them followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers.
- The pre-processing needed in a ConvNet is kindred to that of the related pattern of neurons in the human brain and was motivated by the organization of the Visual Cortex.

A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function. A few distinct types of layers are commonly used.



**Figure 3: CNN architecture (theclickreader)**

- ✓ **Convolutional layer:** The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the filter entries and the input, producing a 2 dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input
- ✓ **Pooling layer:** Another important concept of CNNs is pooling, which is a form of non linear down-sampling. There are several non-linear functions to implement pooling, where max pooling is the most common. It partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum.
- ✓ **ReLU layer:** ReLU is the abbreviation of rectified linear unit, which applies the non saturating activation function. It effectively removes negative values from an activation map by setting them to zero. It introduces nonlinearities to the decision function and in the overall network without affecting the receptive fields of the convolution layers.
- ✓ **Fully connected layer:** After several convolutional and max pooling layers, the final classification is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non convolutional) artificial neural networks. Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset (vector addition of a learned or fixed bias term).

- ✓ **Loss layer:** The "loss layer", or "loss function", specifies how training penalizes the deviation between the predicted output of the network, and the true data labels (during supervised learning). Various loss functions can be used, depending on the specific task. The Softmax loss function is used for predicting a single class of K mutually exclusive classes.

Hyperparameters are various settings that are used to control the learning process. CNNs use more hyperparameters than a standard multilayer perceptron (MLP).

- Kernel Size
- Padding
- Stride
- Number of filters
- Filter size
- Pooling type and size

## 4 Tools and Methods Used in Health AI disease detection

### 5.1 Programming Languages

- **Python:** Python is a high-level, interpreted programming language known for its simplicity and readability. With its extensive ecosystem of libraries and frameworks, Python was the primary language used in our project for data preprocessing, model development, and implementation of AI algorithms, making it a preferred choice for both data science and machine learning tasks.

### 5.2 Libraries and Frameworks

- **TensorFlow:** an open-source machine learning framework developed by Google, provides comprehensive tools for building and deploying machine learning models, including deep learning. In our project, TensorFlow was crucial for creating, training, and evaluating convolutional neural networks (CNNs) for image-based disease detection, allowing efficient handling of complex neural network architectures.
- **Keras:** a high-level neural network API running on top of TensorFlow, simplifies the process of building and training neural networks. It was utilized in our project to design and train deep learning models due to its user-friendly interface, enabling rapid experimentation and model iteration.
- **Scikit-Learn:** is a powerful Python library for machine learning that provides simple and efficient tools for data mining and analysis. It was employed in our project for traditional machine learning tasks, including the implementation of the XGBoost algorithm and other ensemble learning techniques for disease prediction.
- **Pandas:** is a Python library for data manipulation and analysis, offering data structures and functions necessary for handling structured data. In our project, Pandas was used extensively for data preprocessing tasks such as handling missing values, data

normalization, and feature engineering, ensuring the data was clean and suitable for modeling.

- **NumPy:** is a library that supports large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. It was used for numerical computations and handling array operations, fundamental for machine learning and deep learning workflows in our project.
- **Matplotlib and Seaborn:** are Python libraries for data visualization, providing a wide range of plotting functions and statistical graphics. These libraries were used in our project to visualize data distributions, model performance, and other important metrics, aiding in the interpretation and presentation of results.
- **OpenCV:** (Open-Source Computer Vision Library) provides a common infrastructure for computer vision applications. It was used in our project for image preprocessing tasks, such as resizing and normalizing medical images, which were crucial steps before feeding the images into the CNN models.
- **Flask:** is a lightweight web framework for Python, allowing developers to create web applications with ease. In our project, Flask was used for the backend development, serving as the glue to integrate the machine learning models with a web interface, enabling users to interact with the AI platform for disease detection.

### 5.3 Development Environment

- **Jupyter Notebook:** Jupyter Notebook is an open-source web application that allows creating and sharing documents containing live code, equations, visualizations, and narrative text. It was used in our project for interactive development and experimentation with machine learning models, facilitating iterative testing and debugging.
- **Visual Studio Code:** (VS Code) is a source-code editor featuring support for debugging, embedded Git control, syntax highlighting, and intelligent code completion. It was used in our project for coding and managing project files, providing a robust development environment.
- **Kaggle:** is a renowned platform for data science and machine learning, offering a wide range of datasets, computational resources, and a collaborative environment. Kaggle was the primary source for our datasets, providing well-organized and labeled data essential for training our models. The platform's notebooks provided a robust and user-friendly environment for writing and executing Python code, facilitating data exploration, preprocessing, and visualization. Using Kaggle's powerful GPUs and TPUs, we trained complex models such as Convolutional Neural Networks (CNNs) and XGBoost for image classification and tabular data prediction, respectively.

### 5.4 Version Control

- **Git and GitHub:** Git is a distributed version control system, and GitHub is a web-based platform using Git for version control and collaboration. These tools were used in our project for version control, project management, and collaboration, enabling efficient tracking of changes.

## 5.5 Frontend Development

TML, CSS, and JavaScript are fundamental technologies for web development. HTML (HyperText Markup Language) is used for structuring the content on the web, CSS (Cascading Style Sheets) for styling and layout, and JavaScript for creating dynamic and interactive user interfaces. In our project, these technologies were employed to develop the frontend of the web application, providing a user-friendly interface for users to upload medical images, view results, and interact with the AI platform for disease detection.

## 5.6. Data Collection and Preprocessing

In our project, we utilized both structured and unstructured data, including image datasets and tabular data, to build and train our models for disease detection. This involved collecting various datasets from Kaggle, which provided high-quality data for our analysis and modeling. Below is an overview of the datasets used:

➤ **Brain Tumor:** [Brain Tumor MRI Dataset](#)

This dataset contains MRI images categorized into three classes: glioma, meningioma, and no tumor. These images are essential for training convolutional neural networks (CNNs) to detect and classify brain tumors accurately.

➤ **Alzheimer's:** [Alzheimer's Dataset \(4 Class of Images\)](#)

This dataset includes images of four different stages of Alzheimer's disease: Mild Demented, Moderate Demented, Non-Demented, and Very Mild Demented. The data is used to train models to detect and classify the stages of Alzheimer's from MRI scans.

➤ **Covid-19:** [Covid-19 Radiography Database](#)

The dataset comprises chest X-ray images of patients with COVID-19, along with images of normal lungs and those affected by viral pneumonia. This data helps in training CNNs to differentiate between COVID-19, viral pneumonia, and normal lung conditions.

➤ **Pneumonia:** [Chest X-ray Images \(Pneumonia\)](#)

This dataset consists of chest X-ray images labeled as normal or pneumonia. It is used to develop models that can accurately detect pneumonia in X-ray images.

➤ **Breast Cancer:** [Breast Cancer Dataset](#)

This dataset includes features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. It contains both benign and malignant cases, used to train models for predicting breast cancer.

➤ **Diabetes:** [Diabetes Data Set](#)

This dataset contains medical information such as glucose level, blood pressure, BMI, and other variables to predict the likelihood of diabetes. It is used to train ensemble learning models like XGBoost for diabetes prediction.

Data preprocessing tasks included data cleaning (handling missing values, removing duplicates), data normalization (scaling numerical features), and image preprocessing (resizing, normalizing, and augmenting images) to ensure the data was ready for training machine learning models.

## 5.7. Modeling

This step involved selecting, training, and evaluating machine learning algorithms to create predictive models. Techniques used in our project included ensemble learning (combining multiple weak learners), gradient boosting (XGBoost algorithm), and convolutional neural networks (CNNs) for image-based disease detection. These methods were chosen for their effectiveness in handling both structured and unstructured data, providing accurate predictions for disease diagnosis.

## 5.8. Evaluation

Evaluation involved assessing the quality of the models and determining if they met the initial project objectives. In our project, models were evaluated using various metrics and tools to ensure they provided accurate and reliable predictions. This included the use of ROC curves and other diagnostic tools to determine the optimal classification model, ensuring the models' robustness and applicability in real-world scenarios.

# 5 Project Development Roadmap

## 5.2 Create a virtual environment and install the dependencies

The goal of setting a Virtual environment is to manage dependencies and isolate the project environment.

- First to create it just we need to enter this command:

```
python -m venv. Healthai
```

- Then we need to activate it and install the requirement for our project in this VM:

```
.healthai\Scripts\activate
```

- It is a good practice to upgrade the pip :

```
python.exe -m pip install --upgrade pip
```

- After this create the requirement.txt file that hold all the dependencies that I need for this project.

```
Flask
```

```
numpy
```

```
pandas
```

```
tensorflow-cpu
```

*keras*

*scikit-learn*

*gunicorn*

- Then to help you ignore the files that you don't want to track in your repository like the .healthai directory we need to create a file with the name of .gitignore
- To install all this libraries in our Virtual environment:  
*pip install -r .\requirements.txt*

## 6.2 Create the notebooks and download the models then save them in google drive

- [Diabetes Disease detection notebook](#)
- [Breast Cancer Disease detection notebook](#)
- [Alzheimer Disease Detection notebook](#)
- [Brain tumor disease detection notebook](#)
- [Pneumonia disease detection notebook](#)
- [Covid19 disease detection notebook](#)

All the models are available here: [google drive healthai models](#)

## 5.3 cerate the index.html

```
6 <!DOCTYPE html>
7 <html lang="en">
8   <head>
9     <meta charset="UTF-8">
10    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
11    <title>HealthAI Intelligent Disease Detection</title>
12    <link rel="stylesheet" href="{ url_for('static',
filename='css/global.css') }}">
13    <link rel="stylesheet" href="{ url_for('static',
filename='css/index.css') }}">
14  </head>
15  <body>
16    <header>
17      <nav>
18        <ul>
19          <li><a href="{
url_for('index') }}">Main</a></li>
20          <li><a href="{ url_for('disease_page',
disease='BreastCan') }}">BreastCan</a></li>
```

```

21         <li><a href="{ { url_for('disease_page',
disease='diabetes') }}">Diabetes</a></li>
22         <li><a href="{ { url_for('disease_page',
disease='alzheimer') }}">Alzheimer</a></li>
23         <li><a href="{ { url_for('disease_page',
disease='brain_tumor') }}">brain tumor</a></li>
24         <li><a href="{ { url_for('disease_page',
disease='covid19') }}">Covid 19</a></li>
25         <li><a href="{ { url_for('disease_page',
disease='pneumonia') }}">pneumonia</a></li>
26     </ul>
27 </nav>
28 </header>
29 <main>
30     <section id="main" class="hero">
31         <h1>HealthAi intelligent disease Detection</h1>
32         <p>Get predictions for various diseases using AI
models.</p>
33     </section>
34     <section class="cards">
35
36         <!-- TODO: Card1: -->
37         <div id="BreastCan" class="card">
38             
39             <h2>Breast Cancer</h2>
40             <p>Breast cancer is the most common cancer
amongst women in the world. It accounts
41                 for 25% of all cancer cases, and affected
over 2.1 Million people in 2015 alone.
42                 It starts when cells in the breast begin to
grow out of control. These cells usually
43                 form tumors that can be seen via X-ray or
felt as lumps in the breast area
44             </p>
45             <a href="{ { url_for('disease_page',
disease='BreastCan') }}">Predict</a>
46         </div>
47
48         <!-- TODO: Card2: -->
49         <div id="Diabetes" class="card">
50             
51             <h2>Diabetes</h2>
52             <p>Diabetes is a chronic disease that occurs when
the body cannot properly use and store
53                 glucose and sugar. It is a metabolic disorder
that affects the body's ability to process
54                 blood sugar. It is a common disease that
affects millions of people worldwide. It can lead

```



```

55         to serious health complications if not
managed properly.
56     </p>
57     <a href="{ { url_for('disease_page',
disease='diabetes') }}">Predict</a>
58
59 </div>
60
61 <!-- TODO: Card3: -->
62 <div id="Alzheimer" class="card">
63     
64     <h2>Alzheimer's Disease</h2>
65     <p>Alzheimer's disease is a progressive
neurological disorder that leads to memory loss,
66         cognitive decline, and changes in behavior.
It is the most common cause of dementia,
67         affecting a person's ability to think,
remember, and carry out daily activities.
68     </p>
69     <a href="{ { url_for('disease_page',
disease='alzheimer') }}">Predict</a>
70 </div>
71
72 <!-- TODO: Card4: -->
73 <div id="brain_tumor" class="card">
74     
75     <h2>Brain Tumor</h2>
76     <p>A brain tumor is a mass of abnormal cells that
grows in the brain. It can be either cancerous or
77         noncancerous. The tumor may cause an increase
in pressure inside the skull, resulting in potential
78         brain damage and life-threatening
consequences.</p>
79     <a href="{ { url_for('disease_page',
disease='brain_tumor') }}">Predict</a>
80
81 </div>
82
83 <!-- TODO: Card5: -->
84 <div id="Covid19" class="card">
85     
86     <h2>Covid 19</h2>
87     <p> Covid 19 is caused by the severe acute
respiratory

```

```

88         syndrome coronavirus 2 (SARS-CoV-2). It was
first identified in December 2019 in Wuhan, China,
89         and has since led to a global pandemic. The
disease is primarily spread through respiratory
90         droplets, and symptoms can range from mild
(or no symptoms) to severe respiratory illness.
91         </p>
92         <a href="{{ url_for('disease_page',
disease='covid19') }}">Predict</a>
93
94     </div>
95
96     <!-- TODO: Card6: -->
97     <div id="pneumonia" class="card">
98         
99         <h2>Pneumonia</h2>
100        <p>Pneumonia is an inflammatory condition of the
lung affecting primarily the small air sacs known
101        as alveoli.Symptoms typically include some
combination of productive or dry cough, chest pain, fever
102        and difficulty breathing. The severity of the
condition is variable.</p>
103        <a href="{{ url_for('disease_page',
disease='pneumonia') }}">Predict</a>
104    </div>
105    </section>
106    </main>
107    <footer>
108        <p>&copy; 2024 Health AI Platform. All rights
reserved.</p>
109    </footer>
110    <script src="{{ url_for('static', filename='js/scripts.js')
}}"></script>
111    </body>
112</html>

```

## 6.3 create the html file for every disease

Let's list two examples:

For covid19:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">

```

```

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Covid19 disease detection</title>
    <link rel="stylesheet" href="{ { url_for('static',
filename='css/global.css') }}">
    <link rel="stylesheet" href="{ { url_for('static',
filename='css/prediction.css') }}">

</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="{ { url_for('index') }}">Main</a></li>
                <li><a href="{ { url_for('disease_page',
disease='BreastCan') }}">BreastCan</a></li>
                <li><a href="{ { url_for('disease_page',
disease='diabetes') }}">Diabetes</a></li>
                <li><a href="{ { url_for('disease_page',
disease='alzheimer') }}">Alzheimer</a></li>
                <li><a href="{ { url_for('disease_page',
disease='brain_tumor') }}">brain tumor</a></li>
                <li><a href="{ { url_for('disease_page',
disease='covid19') }}">Covid 19</a></li>
                <li><a href="{ { url_for('disease_page',
disease='pneumonia') }}">pneumonia</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <div class="container">
            <section class="hero">
                <h1>Covid19 disease Prediction</h1>
            </section>
            <form action="" method="post" enctype="multipart/form-data"
id="image-form">
                
                <h2>Upload your image</h2>
                <input type="file" name="image" id="image"
accept="image/*" required>
                <button type="submit">Predict</button>
            </form>
            {% if prediction_result %}
            <div class="prediction">
                <h2>Prediction Result:</h2>
                <div class="result">
                    

```

```

        <p>Prediction: {{ prediction_result.prediction }}</p>
    </div>
</div>
{% endif %}
</div>

</main>
<footer>
    <p>© 2024 Covid19 Disease Detection</p>
</footer>

</body>
</html>

```

And for diabetes:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Diabetes disease detection</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/global.css')}}">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/prediction.css')}}">

</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="{{ url_for('index')}}">Main</a></li>
                <li><a href="{{ url_for('disease_page',
disease='BreastCan')}}">BreastCan</a></li>
                <li><a href="{{ url_for('disease_page',
disease='diabetes')}}">Diabetes</a></li>
                <li><a href="{{ url_for('disease_page',
disease='alzheimer')}}">Alzheimer</a></li>
                <li><a href="{{ url_for('disease_page',
disease='brain_tumor')}}">brain tumor</a></li>
                <li><a href="{{ url_for('disease_page',
disease='covid19')}}">Covid 19</a></li>
                <li><a href="{{ url_for('disease_page',
disease='pneumonia')}}">pneumonia</a></li>
            </ul>
        </nav>
    </header>

```

```

<main>
  <div class="container">
    <section class="hero">
      <h1>Diabetes disease Prediction</h1>
    </section>
    <form action="{{ url_for('disease_page', disease='diabetes') }}" method="post" enctype="multipart/form-data" id="normal_prediction">
      
      <div class="grid-container">
        <div class="grid-item">
          <label for="pregnancies">Pregnancies:</label>
          <input type="number" id="pregnancies"
name="pregnancies" required>
        </div>
        <div class="grid-item">
          <label for="glucose">Glucose:</label>
          <input type="number" id="glucose" name="glucose"
required>
        </div>
        <div class="grid-item">
          <label for="bloodPressure">Blood
Pressure:</label>
          <input type="number" id="bloodPressure"
name="bloodPressure" required>
        </div>
        <div class="grid-item">
          <label for="skinThickness">Skin
Thickness:</label>
          <input type="number" id="skinThickness"
name="skinThickness" required>
        </div>
        <div class="grid-item">
          <label for="insulin">Insulin:</label>
          <input type="number" id="insulin" name="insulin"
required>
        </div>
        <div class="grid-item">
          <label for="bmi">BMI:</label>
          <input type="number" step="0.1" id="bmi"
name="bmi" required>
        </div>
        <div class="grid-item">
          <label for="diabetesPedigreeFunction">Diabetes
Pedigree Function:</label>
          <input type="number" step="0.01"
id="diabetesPedigreeFunction" name="diabetesPedigreeFunction" required>
        </div>
        <div class="grid-item">
          <label for="age">Age:</label>

```

```

        <input type="number" id="age" name="age"
required>
    </div>
</div>
<button type="submit">Predict</button>
</form>
{% if prediction_result %}
<div class="prediction">
    <div class="input_data">
        <h3>Input Data:</h3>
        <ul class="inputs">
            <li>Pregnancies: {{
prediction_result.input_data.Pregnancies }}</li>
            <li>Glucose: {{
prediction_result.input_data.Glucose }}</li>
            <li>Blood Pressure: {{
prediction_result.input_data.BloodPressure }}</li>
            <li>Skin Thickness: {{
prediction_result.input_data.SkinThickness }}</li>
            <li>Insulin: {{
prediction_result.input_data.Insulin }}</li>
            <li>BMI: {{ prediction_result.input_data.BMI
}}</li>
            <li>Diabetes Pedigree Function: {{
prediction_result.input_data.DiabetesPedigreeFunction }}</li>
            <li>Age: {{ prediction_result.input_data.Age
}}</li>
        </ul>
    </div>
    <h2>Prediction Result: </h2>
    <div id="prediction-result">
        <p>Prediction: {{ prediction_result.prediction }}</p>
    </div>
</div>
{% endif %}
</div>
</main>
<footer>
    <p>© 2024 Diabetes Disease Detection</p>
</footer>
</body>
</html>

```

## 6.4 Style the Application

This is just a peace of code :

```

/*TODO: the global styles for the website*/
:root{

```

```

--primary-color: #18181b;
--secondary-color: rgb(39, 39, 42);
--title-color: rgb(244, 244, 245);
--subtitle-color: rgb(161, 161, 170);
--hover-color: #333333;
--border: rgba(63, 63, 70, 0.4);
--blue: rgb(93,188,252);
}

body{
  font-family: 'Arial', sans-serif;
  background-color: var(--primary-color);
  color: var(--subtitle-color);
  margin: 0;
  padding: 0;
}

/* TODO: the header part */
header{
  font-weight: bold;
  color: var(--title-color);
  padding: 1em 0; /*1em = 16px*/
  margin:0;
}

header nav{
  justify-content: center;
  display: flex;
}

header nav ul{
  list-style: none;
  display: flex; /*to make it horizontal*/
  justify-content: center;
  margin: 0;
  padding: 1em;
  background: var(--secondary-color);
  border-radius: 50px;
}

header nav ul li{
  margin: 0 1em
}

header nav ul li a{
  text-decoration:none;
  color: var(--title-color);
  font-size: 1.2em; /*1.2em = 19.2px*/
  padding: 0.5em 0.5em;
  transition: background 0.3s; /*for smooth transition*/
}

```

```

}

header nav ul li a:hover{
  background:var(--hover-color);
  color: var(--blue);
  border-radius: 10px;
}

/*TODO:hero section */

.hero{
  text-align:center;
  padding: 2em 0;
}

.hero h1{
  color: var(--title-color);
  font-size:3em;
  margin: 0.5em 0;
}

```

## 6.5 create some animation using js

```

document.addEventListener("DOMContentLoaded",function(){

  //TODO: animate the hero title
  const heroTitle = document.querySelector('.hero');
  heroTitle.style.opacity = 0;
  heroTitle.style.transform = 'translateY(-50px)';

  setTimeout(() => {
    heroTitle.style.transition = 'opacity 1s ease-out, transform 1s ease-out';
    heroTitle.style.opacity = 1;
    heroTitle.style.transform = 'translateY(0)';
  },100);
});

```

## 7.Create the flask application

### 7.1 import the libraries:

```

# TODO:_____importing the
libraries_____

```



```

from flask import Flask, request, redirect, url_for, render_template
from numpy import array, nan
from pandas import DataFrame
import os
import joblib, pickle
from werkzeug.utils import secure_filename
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import img_to_array, load_img
# TODO: _____creating and
configuring the flask app_____

```

## 7.2 Creating Custom Transformers for Diabetes detection

This step involves creating custom transformers using the BaseEstimator and TransformerMixin classes from sklearn to preprocess data related to BMI, Insulin, and Glucose levels. These transformers will categorize and transform the input data before making predictions.

```

from sklearn.base import BaseEstimator, TransformerMixin
from pandas import Series, to_numeric

# TODO: Creating Custom Transformers
class BMITransformer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        NewBMI = Series(["Underweight", "Normal", "Overweight", "Obesity 1",
"Obesity 2", "Obesity 3"], dtype="category")
        X['NewBMI'] = "Normal"
        X['BMI'] = to_numeric(X['BMI'], errors='coerce') # Convert BMI to numeric,
setting errors to NaN
        X.loc[X["BMI"] < 18.5, "NewBMI"] = NewBMI[0]
        X.loc[(X["BMI"] >= 18.5) & (X["BMI"] <= 24.9), "NewBMI"] = NewBMI[1]
        X.loc[(X["BMI"] > 24.9) & (X["BMI"] <= 29.9), "NewBMI"] = NewBMI[2]
        X.loc[(X["BMI"] > 29.9) & (X["BMI"] <= 34.9), "NewBMI"] = NewBMI[3]
        X.loc[(X["BMI"] > 34.9) & (X["BMI"] <= 39.9), "NewBMI"] = NewBMI[4]
        X.loc[X["BMI"] > 39.9, "NewBMI"] = NewBMI[5]
        return X

class InsulinTransformer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        def set_insulin(row):

```

```

        if 16 <= row["Insulin"] <= 166:
            return "Normal"
        else:
            return "Abnormal"
    X["NewInsulinScore"] = X.apply(set_insulin, axis=1)
    return X

class GlucoseTransformer(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.NewGlucose = Series(["Low", "Normal", "Overweight", "Secret", "High"],
dtype="category")

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        X["NewGlucose"] = "Normal"
        X.loc[X["Glucose"] <= 70, "NewGlucose"] = self.NewGlucose[0]
        X.loc[(X["Glucose"] > 70) & (X["Glucose"] <= 99), "NewGlucose"] =
self.NewGlucose[1]
        X.loc[(X["Glucose"] > 99) & (X["Glucose"] <= 126), "NewGlucose"] =
self.NewGlucose[2]
        X.loc[X["Glucose"] > 126, "NewGlucose"] = self.NewGlucose[3]
        return X

```

### 7.3.Flask App Configuration

Set up the Flask app, configure it, and load the necessary machine learning models.

```

UPLOAD_FOLDER = 'static/uploads/'
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}

app = Flask(__name__)
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 0
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Load models
models = {
    'alzheimer': load_model('Models/alzheimer_model.keras'),
    'brain_tumor': load_model('Models/Brain_tumor_model.keras'),
    'breast_cancer': pickle.load(open('Models/breast_cancer_model.pkl', 'rb')),
    'covid19': load_model('Models/covid_model.h5', compile=False),
    'diabetes': joblib.load('Models/diabetes_model.pkl'),
    'pneumonia': load_model('Models/pneumia_model.keras')
}

model = models['covid19']
# Re-compile the model with the correct settings
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

```

## 7.4 Utility Functions

Define utility functions to check if the file type is allowed and to transform the image for predictions.

```
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

def transform_image(file_path, disease, height=150, width=150,
color_mode='grayscale'):
    img = load_img(file_path, target_size=(height, width), color_mode=color_mode)
    img_array = img_to_array(img)
    img_array = expand_dims(img_array, axis=0)

    model = models[disease]
    prediction = model.predict(img_array)
    result = argmax(prediction, axis=1)[0]
    return result
```

## 7.4 Define the Home and prediction Routes

Render the homepage of the web application, and Handle disease-specific prediction requests based on the type of disease selected.

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/<disease>', methods=['GET', 'POST'])
def disease_page(disease):
    prediction_result = None
    if disease in ['alzheimer', 'brain_tumor', 'covid19', 'pneumonia', 'BreastCan',
'diabetes']:
        if request.method == 'POST':
            if disease in ['alzheimer', 'brain_tumor', 'covid19', 'pneumonia']:
                prediction_result = handle_image_disease(request, disease)
            elif disease == 'BreastCan':
                prediction_result = handle_breast_cancer(request)
            elif disease == 'diabetes':
                prediction_result = handle_diabetes(request)
        return render_template(f'{disease}.html',
prediction_result=prediction_result)
    return redirect(url_for('index'))
```

## 7.5 Handle Image-Based Disease Prediction

Process the uploaded image, categorize it, and make predictions for image-based diseases.

```
def handle_image_disease(request, disease):
    if 'image' not in request.files:
        return {'error': 'No file part'}
    file = request.files['image']
    if file.filename == '':
        return {'error': 'No selected file'}
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(file_path)

        if disease == 'alzheimer':
            result = transform_image(file_path, 'alzheimer', 176, 208, 'rgb')
            alzheimer_classes = {
                0: 'MildDemented',
                1: 'ModerateDemented',
                2: 'NonDemented',
                3: 'VeryMildDemented'
            }
            result = alzheimer_classes.get(result, 'Unknown')
        elif disease == 'brain_tumor':
            result = transform_image(file_path, 'brain_tumor', 150, 150, 'rgb')
            brain_classes = {
                0: 'glioma',
                1: 'meningioma',
                2: 'no tumor',
                3: 'pituitary'
            }
            result = brain_classes.get(result, 'Unknown')
        elif disease == 'covid19':
            result = transform_image(file_path, 'covid19', 70, 70, 'rgb')
            covid_classes = {
                0: 'Normal',
                1: 'COVID',
                2: 'Lung Opacity',
                3: 'Viral Pneumonia'
            }
            result = covid_classes.get(result, 'Unknown')
        elif disease == 'pneumonia':
            result = transform_image(file_path, 'pneumonia', 150, 150)
            pneumonia_classes = {
                0: 'Pneumonia',
                1: 'Normal'
            }
            result = pneumonia_classes.get(result, 'Unknown')
```

```

        categorized_folder_path = os.path.join(app.config['UPLOAD_FOLDER'],
disease, result)
        if not os.path.exists(categorized_folder_path):
            os.makedirs(categorized_folder_path)

        unique_filename = filename
        counter = 1
        while os.path.exists(os.path.join(categorized_folder_path,
unique_filename)):
            name, ext = os.path.splitext(filename)
            unique_filename = f"{name}_{counter}{ext}"
            counter += 1

        categorized_file_path = os.path.join(categorized_folder_path,
unique_filename)
        os.rename(file_path, categorized_file_path)

        return {'prediction': result, 'file_path':
categorized_file_path.replace('\\', '/')}
    return {'error': 'File not allowed'}

```

## 7.6 Handle Breast Cancer Prediction

Process form data and make predictions for breast cancer.

```

def handle_breast_cancer(request):
    features = [
        request.form.get('radius_mean'),
        request.form.get('texture_mean'),
        request.form.get('perimeter_mean'),
        request.form.get('area_mean'),
        request.form.get('smoothness_mean'),
        request.form.get('compactness_mean'),
        request.form.get('concavity_mean'),
        request.form.get('concave_points_mean'),
        request.form.get('symmetry_mean'),
        request.form.get('radius_se'),
        request.form.get('perimeter_se'),
        request.form.get('area_se'),
        request.form.get('compactness_se'),
        request.form.get('concavity_se'),
        request.form.get('concave_points_se'),
        request.form.get('radius_worst'),
        request.form.get('texture_worst'),
        request.form.get('perimeter_worst'),
        request.form.get('area_worst'),
        request.form.get('smoothness_worst'),
        request.form.get('compactness_worst'),
        request.form.get('concavity_worst'),

```

```

        request.form.get('concave_points_worst'),
        request.form.get('symmetry_worst'),
        request.form.get('fractal_dimension_worst')
    ]

    features = array([float(feature) if feature else nan for feature in
features]).reshape(1, -1)
    feature_names = [
        'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
'smoothness_mean',
        'compactness_mean', 'concavity_mean', 'concave_points_mean',
'symmetry_mean',
        'radius_se', 'perimeter_se', 'area_se', 'compactness_se', 'concavity_se',
'concave_points_se', 'radius_worst', 'texture_worst', 'perimeter_worst',
'area_worst', 'smoothness_worst', 'compactness_worst', 'concavity_worst',
'concave_points_worst', 'symmetry_worst', 'fractal_dimension_worst'
    ]
    input_data = DataFrame(features, columns=feature_names)

    model = models['breast_cancer']
    prediction = model.predict(input_data)
    breast_cancer_classes = {
        0: 'Benign',
        1: 'Malignant'
    }
    prediction = breast_cancer_classes.get(int(prediction[0]), 'Unknown')

    return {'prediction': prediction, 'input_data':
input_data.to_dict(orient='records')[0]}

```

## 7.7 Handle Diabetes Prediction

```

def handle_diabetes(request):
    features = [
        request.form.get('pregnancies'),
        request.form.get('glucose'),
        request.form.get('bloodPressure'),
        request.form.get('skinThickness'),
        request.form.get('insulin'),
        request.form.get('bmi'),
        request.form.get('diabetesPedigreeFunction'),
        request.form.get('age')
    ]
    features = array([float(feature) if feature else nan for feature in
features]).reshape(1, -1)
    feature_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
    input_data = DataFrame(features, columns=feature_names)
    input_data = BMITransformer().transform(input_data)
    input_data = InsulinTransformer().transform(input_data)
    input_data = GlucoseTransformer().transform(input_data)

```

```

model = models['diabetes']
prediction = model.predict(input_data)
diabetes_classes = {
    0: 'No Diabetes',
    1: 'Diabetes'
}
prediction = diabetes_classes.get(int(prediction[0]), 'Unknown')
return {'prediction': prediction, 'input_data':
input_data.to_dict(orient='records')[0]}

```

## 7.8 Running the Flask app

```

if __name__ == '__main__':
    app.run(debug=True)

```

## 8. Project UI

The user interface is the point at which human users interact with a computer, website or application. The goal of effective UI is to make the user's experience easy and intuitive, requiring minimum effort on the user's part to receive the maximum desired outcome. UI is created in layers of interaction that appeal to the human senses (sight, touch, auditory and more). They include both input devices like a keyboard, mouse, trackpad, microphone, touch screen, fingerprint scanner, e-pen and camera, and output devices like monitors, speakers and printers.

User interface is important to meet user expectations and support the effective functionality of your site. A well-executed user interface facilitates effective interaction between the user and the program, app or machine through contrasting visuals, clean design and responsiveness.


User Interface of the project is designed using HTML, CSS, and JavaScript.

## 8.1 Homepage

[Main](#)[BreastCan](#)[Diabetes](#)[Alzheimer](#)[brain tumor](#)[Covid 19](#)[pneumonia](#)

# HealthAi intelligent disease Detection


Get predictions for various diseases using AI models.



### Breast Cancer

Breast cancer is the most common cancer amongst women in the world. It accounts for 25% of all cancer cases, and affected over 2.1 Million people in 2015 alone. It starts when cells in the breast begin to grow out of control. These cells usually form tumors that can be seen via X-ray or felt as lumps in the breast area


Predict



### Diabetes

Diabetes is a chronic disease that occurs when the body cannot properly use and store glucose and sugar. It is a metabolic disorder that affects the body's ability to process blood sugar. It is a common disease that affects millions of people worldwide. It can lead to serious health complications if not managed properly.


Predict



### Alzheimer's Disease

Alzheimer's disease is a progressive neurological disorder that leads to memory loss, cognitive decline, and changes in behavior. It is the most common cause of dementia, affecting a person's ability to think, remember, and carry out daily activities.


Predict



### Brain Tumor

A brain tumor is a mass of abnormal cells that grows in the brain. It can be either cancerous or noncancerous. The tumor may cause an increase in pressure inside the skull, resulting in potential brain damage and life-threatening consequences.


Predict



### Covid 19

Covid 19 is caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). It was first identified in December 2019 in Wuhan, China, and has since led to a global pandemic. The disease is primarily spread through respiratory droplets, and symptoms can range from mild (or no symptoms) to severe respiratory illness.

Predict



### Pneumonia

Pneumonia is an inflammatory condition of the lung affecting primarily the small air sacs known as alveoli. Symptoms typically include some combination of productive or dry cough, chest pain, fever and difficulty breathing. The severity of the condition is variable.


Predict

© 2024 Health AI Platform. All rights reserved.



## 8.2 Brain Tumor

### Brain Tumor disease Prediction




**Upload your image**

Choose File

no\_tumor.jpg

Predict

### Prediction Result:



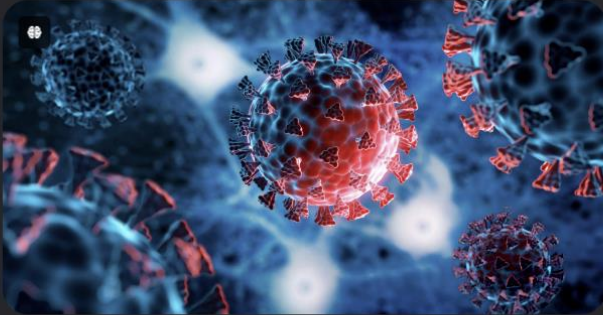
**Prediction: no tumor**

© 2024 Brain tumor Disease Detection

## 8.3 Covid 19

[Main](#) [BreastCan](#) [Diabetes](#) [Alzheimer](#) [brain tumor](#) [Covid 19](#) [pneumonia](#)


# Covid19 disease Prediction



Upload your image

Normal.png

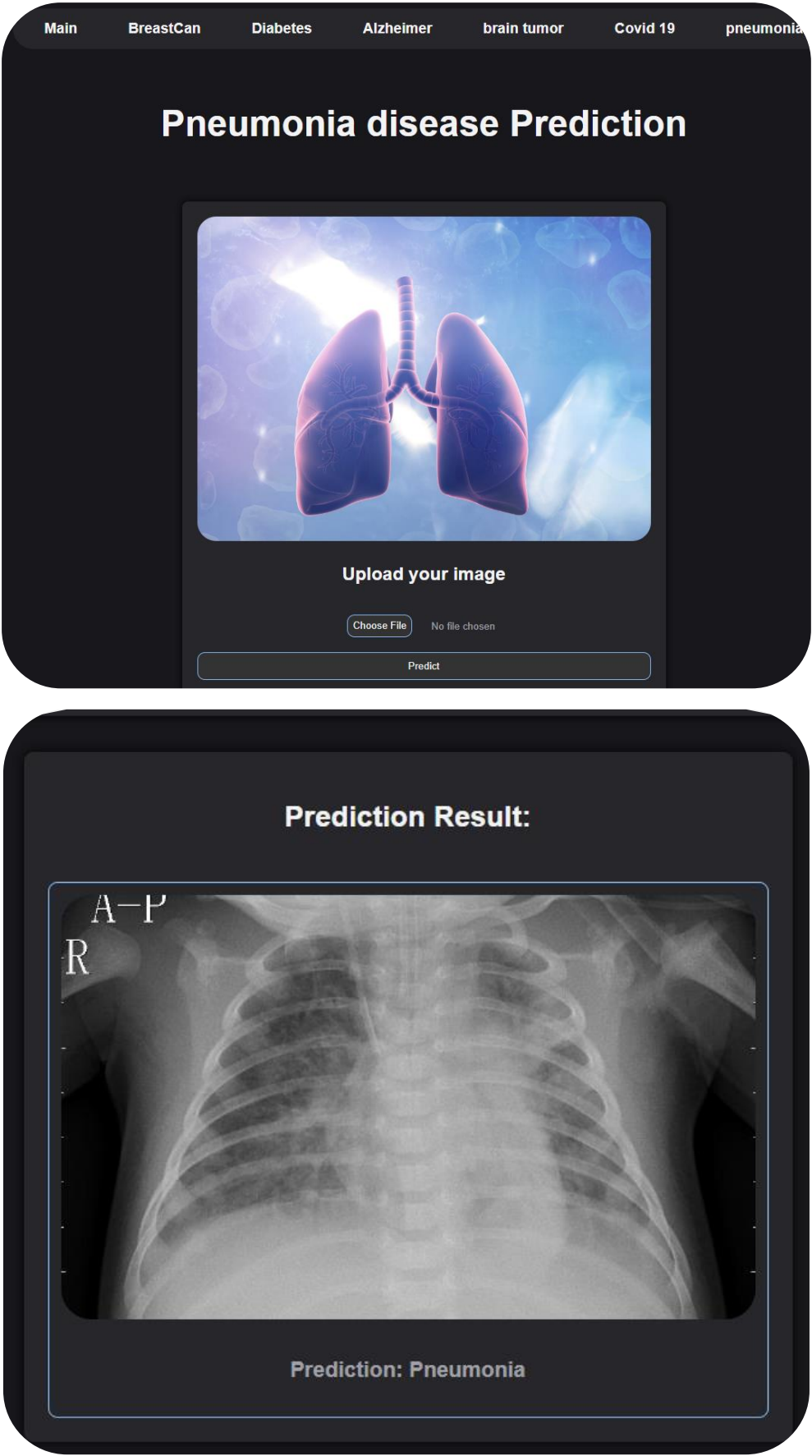
### Prediction Result:



Prediction: Normal

© 2024 Covid19 Disease Detection


8.4 pneumonia



## 8.5 Alzheimer

[Main](#) [BreastCan](#) [Diabetes](#) [Alzheimer](#) [brain tumor](#) [Covid 19](#) [pneumonia](#)

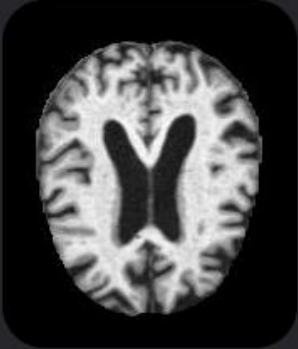
### Alzheimer's disease Prediction



Upload your image

mildDemented.jpg

### Prediction Result:



Prediction: MildDemented

© 2024 Alzheimer Disease Detection

# Diabetes disease Prediction



Pregnancies:

2

Glucose:

232

Blood Pressure:

3

Skin Thickness:

23

Insulin:

32

BMI:

23

Diabetes Pedigree Function:

23

Age:

50

Predict

**Input Data:**

Pregnancies: 1.0

Glucose: 2.0

Blood Pressure: 32.0

Skin Thickness: 32.0

Insulin: 23.0

BMI: 23.0

Diabetes Pedigree Function: 23.0

Age: 23.0

**Prediction Result:**

Prediction: No Diabetes

© 2024 Diabetes Disease Detection

## 9.Conclusion

In this project, we set out to use the power of machine learning and deep learning to help improve the way we diagnose diseases. By focusing on advanced algorithms like XGBoost and Convolutional Neural Networks (CNNs), we created models that can accurately identify conditions such as Alzheimer's, brain tumors, breast cancer, Covid-19, pneumonia, and diabetes.

We started by collecting and preparing various datasets from trusted sources, ensuring they were ready for model training. Using tools like TensorFlow and Keras, we built deep learning models that excel at analyzing medical images, while traditional machine learning methods like XGBoost proved effective for handling structured data, especially in predicting diabetes.

We didn't just stop at building these models—we integrated them into a user-friendly web platform using Flask. This allows anyone to input data and get instant predictions, making our work not only a technical achievement but also a practical tool that could potentially assist in real-world medical settings.

Overall, our project highlights how AI can play a crucial role in healthcare by providing accurate and efficient diagnostic tools. As we continue to refine and expand these models, there's great potential for even more impactful applications in the future, helping to improve patient care and outcomes.

## References

- [1] - Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach*.
- [2] - IBM. (n.d.). Artificial Intelligence. Retrieved from [IBM](#)
- [3] - McCarthy, J. (2004). What Is Artificial Intelligence?
- [4] - "XGBoost Documentation." Available online: [here](#)
- [5] - Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.
- [6] – Analytics Vidhya. What is XGBoost Algorithm. Retrieved from [Analytics Vidhya](#)
- [7] - LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning, 521(7553), 436-444
- [8] - He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778)
- [9]- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. [Available online](#)
- [10] – Coursera. Deep Learning Specialization by Andrew Ng. Retrieved from [Coursera](#)
- [11] - Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9)
- [12]- Brain Tumor MRI Dataset: <https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/data>
- [13] - Alzheimer's Dataset (4 Class of Images): <https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images/data>
- [14] - Covid-19 Radiography Database: <https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database/versions/5>
- [15] - Chest X-ray Images (Pneumonia): <https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia>
- [16] - Breast Cancer Dataset: <https://www.kaggle.com/datasets/yasserh/breast-cancer-dataset>
- [17] - Diabetes Data Set: <https://www.kaggle.com/datasets/mathchi/diabetes-data-set>
- [18] - TensorFlow Documentation: <https://www.tensorflow.org/>
- [19] - Keras Documentation: <https://keras.io/>
- [20] - Scikit-Learn Documentation: <https://scikit-learn.org/>
- [21] -Pandas Documentation: <https://pandas.pydata.org/>
- [22] - NumPy Documentation: <https://numpy.org/>
- [23] - OpenCV Documentation: <https://opencv.org/>
- [24] - Matplotlib Documentation: <https://matplotlib.org/>
- [25] - Seaborn Documentation: <https://seaborn.pydata.org/>
- [26] -Flask Documentation: <https://flask.palletsprojects.com/en/3.0.x/>