INDEX

Abstract

AnemiaSense: Leveraging Machine Learning for Precise Anemia Recognition

Anemia is a widespread health condition marked by a deficiency of red blood cells or hemoglobin, affecting a significant portion of the global population. Conventional diagnostic methods often rely on manual interpretation and fixed threshold-based tests, which can lead to inaccuracies and delayed treatment. To address these limitations, this project introduces **AnemiaSense** — a machine learning-based system designed to enhance the accuracy and efficiency of anemia detection and management. The system leverages clinical data, including blood parameters such as hemoglobin, MCV, MCH, and MCHC, to train predictive models capable of identifying anemia with high precision. The development process involved data preprocessing, model evaluation, optimization, and deployment through a user-friendly web interface. The results demonstrate the effectiveness of advanced algorithms, particularly XGBoost, in improving diagnostic reliability. The project demonstrates how machine learning can revolutionize healthcare by providing more accurate and timely anemia management solutions.

Introduction

Anemia is a common health condition that occurs when the body lacks enough healthy red blood cells or hemoglobin to carry adequate oxygen to tissues. It affects millions of people worldwide, leading to fatigue, weakness, and, in severe cases, serious health complications. Early detection and accurate diagnosis are crucial for effective treatment and improved patient outcomes.

Traditional diagnostic methods often rely on manual interpretation of laboratory results, which can be time-consuming and prone to human error. To address these challenges, this project, **AnemiaSense**, leverages machine learning algorithms to enhance the precision, speed, and reliability of anemia detection.

The system analyzes key blood parameters such as hemoglobin, MCH, MCHC, and MCV to predict anemia status. Multiple machine learning models, including Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and XGBoost, are evaluated, with XGBoost emerging as the most effective model. The final solution is integrated into a user-friendly web application, enabling healthcare professionals and patients to easily access accurate predictions and support informed decision-making.

Technology Used

1. Machine Learning Algorithms:

- Logistic Regression
- Random Forest
- Decision Tree
- Gaussian Naive Bayes
- Support Vector Machine (SVM)
- Gradient Boosting Classifier

2. Programming Languages:

- Python

3. Libraries and Tools:

- Pandas, Scikit-learn for data handling and model building
- Matplotlib, Seaborn for data visualization
- Flask for web framework integration

4. Data Sources:

- CSV data files from open-source platforms like Kaggle

5. Web Development:

- HTML, CSS for front-end development
- Flask for back-end integration

About the Project

4.1 Problem Statement

Anemia is often underdiagnosed or mismanaged due to reliance on manual testing methods that are slow, error-prone, and not personalized. These traditional approaches lack the ability to detect early signs or support continuous monitoring, especially in remote areas. **AnemiaSense** aims to solve these issues by using machine learning to deliver accurate, timely, and personalized anemia detection and management, making care more efficient and accessible.

4.2 Project Description

Anemiasense leverages machine learning algorithms to provide precise recognition and management of anemia, a condition characterized by a deficiency of red blood or hemoglobin.

The Anemiasense was completed in the following steps:

Importing of Libraries

The following necessary libraries were imported: -

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

Data Collection & Preparation

Data collection is fundamental to machine learning, providing the raw material for training algorithms and making predictions. This process involves gathering relevant information from various sources such as databases, surveys, sensors, and web scraping. The quality, quantity, and diversity of collected data significantly impact the performance and accuracy of ML models.

Data Collection: There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc. In this project, we have used .csv data. This data is downloaded from kaggle.com.

Read the Dataset: Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas. In pandas, we have a function called read_csv() to read the dataset. As a parameter, we have to give the directory of the CSV file.

Data Preparation: Before we can use our data to teach our machine-learning model, we need to clean it up. That means we have to deal with missing information, like when there's no data for some entries. We also have to figure out what to do with categories, like types of stages and outliers, which are really unusual data points. This activity includes the following steps.

- Handling missing values
- Handling imbalanced data

Exploratory Data Analysis

Exploratory Data Analysis (EDA) refers to the process of analyzing datasets to summarize their main characteristics, often with visual methods. It is a critical step in machine learning, as it helps to understand the structure, patterns, and relationships within the data before applying any algorithms.

Descriptive statistical: Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this described function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max, and percentile values of continuous features.

Visual analysis: Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

- Univariate analysis
- Bivariate analysis
- Multivariate analysis

Splitting data into train and test: The train-test split is used to estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms/Applications. This method is a fast and easy procedure to perform such that we can compare our own machine learning model results to machine results. By default, the Test set is split into 30 % of actual data and the training set is split into 70% of the actual data. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random state.In our project the Test set is split into 20 % of actual data and the training set is split into 80% of the actual data.

Model Building

Model building in machine learning refers to the process of designing, training, and evaluating a machine learning model to solve a specific problem.

Training the model in multiple algorithms:

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project, we are applying Five Regression algorithms. The best model is saved based on its performance.

Logistic Regression Model

A variable named logistic_regression is created and train and test data are passed as the parameters. Inside the function, the Linear Regression algorithm is initialized and training data is passed to the model. fit() function. Test data is predicted with. predict() function and save d in a new variable. For evaluating the model, an accuracy score and classification report are used.

Random forest model

A variable named random_forest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, an accuracy score and classification report are used.

Decision Tree Model

A function named decision_tree_model is created and train and test data are passed as the parameters. Inside the function, the Decision Classifier algorithm is initialized and training data is passed to the model the with .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For even altering the model, accuracy score and classification report are used.

Gaussian Navies Bayes

A variable named NB is created and train and test data are passed as the parameters. Inside the function, the Gaussian Navies Bayes algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, an accuracy score and classification report are used.

Support Vector Machine

A function named SVC is created and train and test data are passed as the parameters. Inside the function, the Support vector machine algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, an accuracy score and classification report are used.

Gradient Boosting Classifier

A function named GBC is created and train and test data are passed as the parameters. Inside the function, the Gradient Boosting algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, an accuracy score and classification report are used.

XGBoost Classifier

A function named XGBoost is created, and the training and testing data are passed as parameters. Inside the function, the XGBoost classifier is initialized using the XGBClassifier() from the xgboost library. The model is trained using the .fit() method on the training dataset. Once trained, the model makes predictions on the test data using the .predict() function, and the predicted values are stored in a new variable. To evaluate the model’s performance, the accuracy score and classification report are generated, providing detailed metrics such as precision, recall, and F1-score.

Testing the model

Here we have tested with the Lasso model algorithm. You can test with all algorithms. With the help of the predict() function.

Performance testing and Hyper Parameter Tunning

Performance testing in machine learning involves evaluating how well a model performs on a given task or dataset. It measures various aspects like accuracy, speed, scalability, and resource usage, ensuring the model meets the desired objectives.Hyperparameter tuning refers to the process of selecting the best set of hyperparameters (e.g., learning rate, number of trees, number of hidden layers) for a machine learning model. Unlike model parameters (which are learned from data), hyperparameters are set before training.

Testing model with multiple evaluation metrics: Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for regression tasks including Accuracy scores and classification report.

Model Deployment

Model Deployment in machine learning refers to the process of integrating a trained machine learning model into a production environment, where it can make real-time predictions or decisions based on incoming data. It involves taking the model that has been trained and tested in a development environment and deploying it so that it can be used by end-users or systems for practical purposes.

Save the Best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future. We have save the XGBoost model.

Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

Building HTML Pages

The selected model was integrated into Flask web application, allowing users to input data and receive prediction. Two HTML pages (index.html and predict.html) were created to provide a user friendly interface.

Build Python code

We created a python file named app.py and saved it in the root directory.

Final deployment

We run the web application using the command python app.py in the virtual environment venv. The web application was deployed and tested, providing a functional tool for prediction tool for predicting anemia and offering insights for personalised management.

Code

app.py file:

```
import numpy as np

import pandas as pd

import pickle

from flask import Flask, request, render_template

#loading the model

model = pickle.load(open('model.pkl', 'rb'))

app=Flask(__name__)

@app.route("/")

def home():

return render_template("index.html")

@app.route("/prediction")

def predictpage():

return render_template("predict.html")

@app.route("/predict", methods = ['POST'])

def predict():

Gender = float(request.form["Gender"])

Hemoglobin = float(request.form["Hemoglobin"])

MCH = float(request.form["MCH"])

MCHC = float(request.form["MCHC"])

MCV = float(request.form["MCV"])
```

```
features_values = np.array([[Gender, Hemoglobin, MCH, MCHC, MCV]])

df = pd.DataFrame(features_values, columns=['Gender', 'Hemoglobin', 'MCH', 'MCHC',
'MCV'])

print(df)

prediction = model.predict(df)

print(prediction[0])

result = prediction[0]

if prediction[0]==0:

result = " you don't have any Anemic Disease"

elif prediction[0]==1:

result = " you have Anemic Disease"

text = "Result: Hence, based on calculation"

return render_template('predict.html', prediction_text = text+str(result))

if __name__=="__main__":

app.run(debug=True, port=5000)
```

index.html file:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Anemia Predictor</title>

<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>
```

```html
<body>

<header>

<nav class="navbar">

<div class="logo">Anemia</div>

<ul class="nav-links">

<li><a href="#home">Home</a></li>

<li><a href="#about">About</a></li>

<li><a href="/prediction">Predict</a></li>

<li><a href="#contact">Contact</a></li>

</ul>

<a href="/prediction" class="btn-check">Check</a>

</nav>

</header>

<!-- Home Section -->

<section id="home" class="home-section">

<div class="overlay">

<h1>Anemia Predictor: Your Health Guardian</h1>

<p>Empowering you with early detection for better management and healthier living</p>

</div>

</section>

<!-- About Section -->

<section id="about" class="about-section">

<div class="about-content">

<div class="about-text">

<h2>About</h2>
```

```html
<p>Anemia: low red blood cells or hemoglobin, causing oxygen shortage. Symptoms:
fatigue, weakness, shortness of breath, pale skin. Causes: nutritional deficiencies, diseases,
blood loss. Treatments vary.</p>

</div>

<div class="cards">

<div class="card">

<h3>Hemoglobin</h3>

<p>Hemoglobin, a protein in red blood cells, transports oxygen and carbon dioxide. Healthy
ranges: Men: 13.2–16.6 g/dL. Women: 11.6–15 g/dL.</p>

</div>

<div class="card">

<h3>MCH</h3>

<p>An MCH value refers to the average quantity of hemoglobin present in a single red blood
cell. The normal range for MCH is between 27.5 and 33.2 picograms (pg).</p>

</div>

<div class="card">

<h3>MCHC</h3>

<p>The mean corpuscular hemoglobin concentration is average concentration of hemoglobin
in red blood cells. The normal range: 33.4–35.5g/dL.</p>

</div>

<div class="card">

<h3>MCV</h3>

<p>MCV stands for mean corpuscular volume. An MCV blood test measures the average
size of red blood cells. The normal range for MCV is between 80 to 100 femtoliter.</p>

</div>

</div>

</div>

</section>
```

```
<!-- Contact Section -->

<footer id="contact" class="contact-section">

<h3>Contact Us</h3>

<p>The Smart Bridge<br>UttarPredesh, Lucknow, India</p>

<p><strong>Phone:</strong> +91 xxxx 5548 55</p>

<p><strong>Email:</strong> info@example.com</p>

</footer>

</body>

</html>
```

Style.css of index.html file:

```
/* General Reset */

* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

body {

line-height: 1.6;

background-color: #f9f9f9;

color: #333;

padding-top: 80px;

}

/* Navbar */

.navbar {
```

```
position: fixed;

top: 0;

left: 0;

width: 100%;

background: white;

z-index: 999;

display: flex;

justify-content: space-between;

align-items: center;

padding: 1rem 2rem;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

}

.logo {

font-size: 1.5rem;

font-weight: bold;

}

.nav-links {

list-style: none;

display: flex;

gap: 1.5rem;

}

.nav-links a {

text-decoration: none;

color: #333;

font-weight: 500;
```

```
transition: color 0.3s;
}
.nav-links a:hover {
color: #e74c3c;
}
.btn-check {
background-color: #e74c3c;
color: white;
padding: 0.5rem 1rem;
border-radius: 4px;
text-decoration: none;
font-weight: 500;
}
/* Home Section */
.home-section {
background: url('../images/blood.jpg') center center/cover no-repeat;
height: 90vh;
display: flex;
align-items: center;
justify-content: center;
text-align: center;
color: white;
position: relative;
}
.home-section .overlay {
```

```
background: rgba(0, 0, 0, 0.5);

padding: 2rem;

border-radius: 8px;

}

.home-section h1 {

font-size: 2.5rem;

margin-bottom: 1rem;

}

/* About Section */

.about-section {

padding: 5rem 2rem;

background: #fff;

min-height: 100vh;

box-sizing: border-box;

}

.about-text {

max-width: 600px;

margin-bottom: 2rem;

font-size: large;

}

.cards {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

gap: 2rem;

position: relative;
```

```
top: 10rem;

}

.card {

background: #f7f7f7;

padding: 1rem;

border-radius: 10px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

transition: transform 0.3s ease;

}

.card:hover {

transform: translateY(-5px);

background: #a8a5a5;

box-shadow: 0 2px 5px rgba(22, 22, 22, 0.1);

border-radius: 15px;

}

.contact-section {

background: url('../images/result-bg.jpg') center center/cover no-repeat;

padding: 4rem 2rem;

color: rgb(20, 1, 1);

text-align: center;

font-size: larger;

}

footer p {

margin: 0.5rem 0;

}
```

Predict.html file:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Predict Anemia</title>

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: linear-gradient(to right, #f5f5f5, #e2e2e2);

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

padding-top: 80px;

}

.navbar {

position: fixed;

top: 0;

left: 0;

width: 100%;

background: white;
z-index: 999;

display: flex;

justify-content: space-between;

align-items: center;

padding: 1rem 2rem;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

}

.logo {

font-size: 1.5rem;

font-weight: bold;

}

.nav-links {

list-style: none;

display: flex;

gap: 1.5rem;

}

.nav-links a {

text-decoration: none;

color: #333;

font-weight: 500;

transition: color 0.3s;

}

.nav-links a:hover {

color: #e74c3c;

}
```

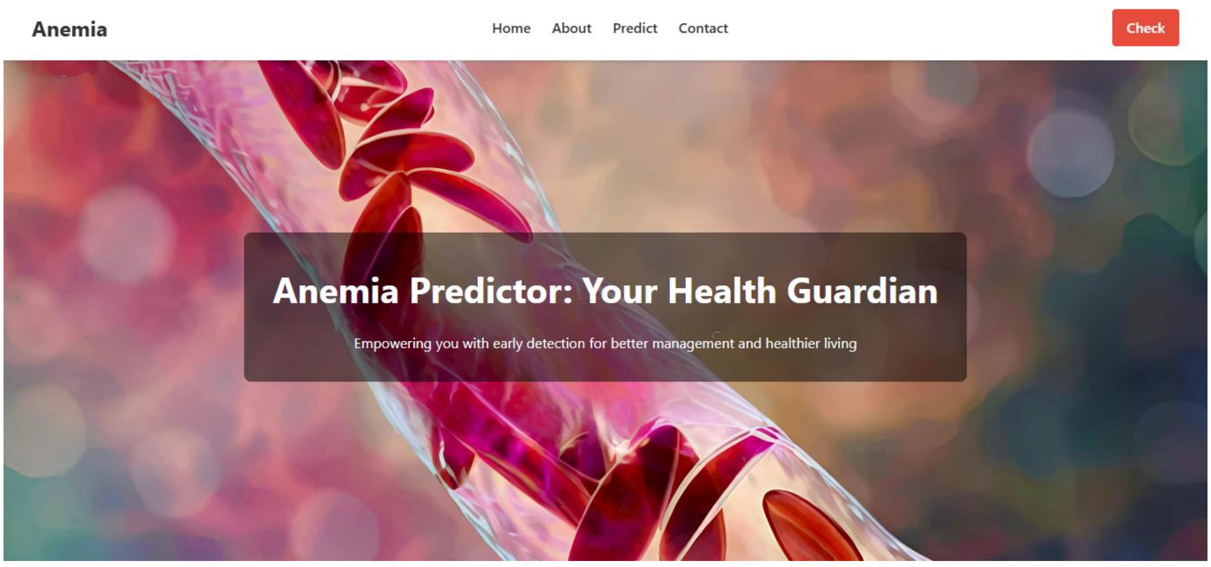
```
.btn-check {

background-color: #e74c3c;

color: white;

padding: 0.7rem 1rem;

border-radius: 5px;

text-decoration: none;

font-weight: 500;

position: relative;

right: 3.5rem;

}

.form-container {

background: white;

padding: 2rem;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0,0,0,0.1);

width: 400px;

}

h2 {

text-align: center;

margin-bottom: 1rem;

}

form {

display: flex;

flex-direction: column;

}
```

```
label {

margin-top: 1rem;

margin-bottom: 0.3rem;

font-weight: bold;

}

input[type="number"] {

padding: 0.5rem;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

margin-top: 1.5rem;

padding: 0.7rem;

background-color: #e74c3c;

color: white;

border: none;

border-radius: 5px;

font-weight: bold;

cursor: pointer;

transition: background-color 0.3s;

}

button:hover {

background-color: #c0392b;

}

.result-box {
```

```
margin-top: 1.5rem;

padding: 1rem;

background-color: #f1f1f1;

border-radius: 8px;

text-align: center;

font-size: 1.1rem;

}

</style>

</head>

<body>

<header>

<nav class="navbar">

<div class="logo">Anemia</div>

<ul class="nav-links">

<li><a href="/">Home</a></li>

<li><a href="/#about">About</a></li>

<li><a href="/prediction">Predict</a></li>

<li><a href="/#contact">Contact</a></li>

</ul>

<a href="/prediction" class="btn-check">Check</a>

</nav>

</header>

<div class="form-container">

<h2>Enter Details to Predict</h2>

<form action="/predict" method="POST">
```

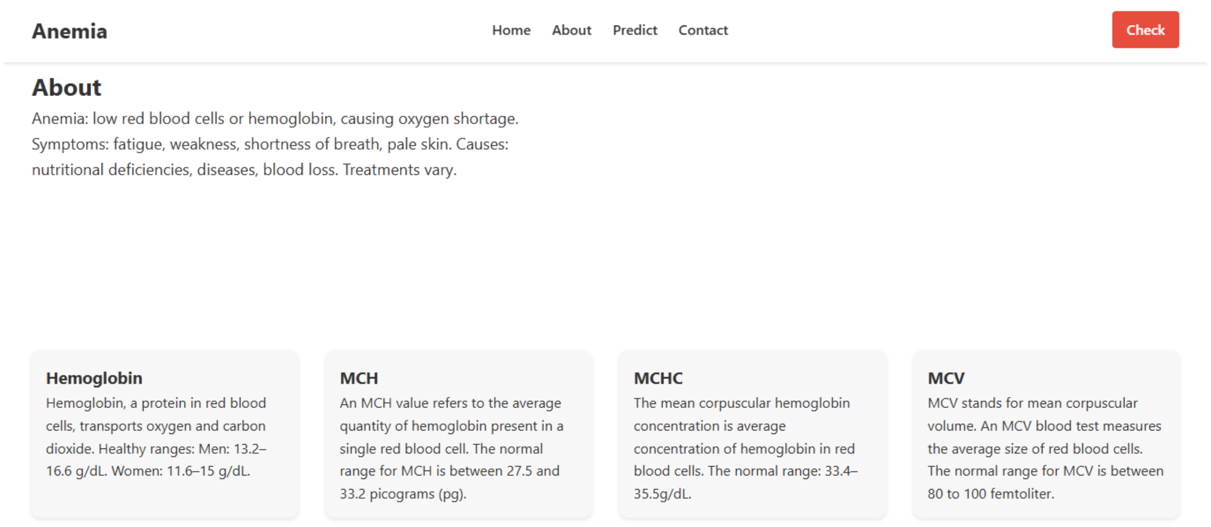
```
<label for="Gender">Gender (0: Female, 1: Male)</label>

<input type="number" step="any" name="Gender" required>

<label for="Hemoglobin">Hemoglobin(Range:7-16)</label>

<input type="number" step="any" name="Hemoglobin" required>

<label for="MCH">MCH(Range:16-30)</label>

<input type="number" step="any" name="MCH" required>

<label for="MCHC">MCHC(Range:28-34)</label>

<input type="number" step="any" name="MCHC" required>

<label for="MCV">MCV(Range:70-100)</label>

<input type="number" step="any" name="MCV" required>

<button type="submit">Predict</button>

</form>

{% if prediction_text %}

<div class="result-box">

{{ prediction_text }}

</div>

{% endif %}

</div>

</body>

</html>
```

Result



Anemia Predictor: Your Health Guardian

Empowering you with early detection for better management and healthier living



About

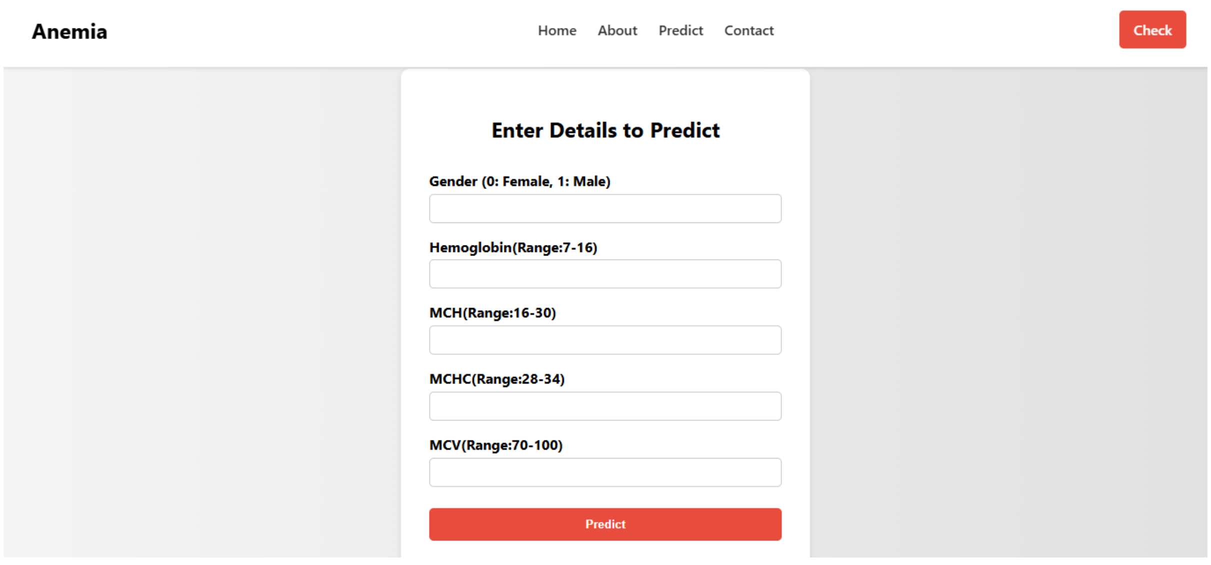
Anemia: low red blood cells or hemoglobin, causing oxygen shortage. Symptoms: fatigue, weakness, shortness of breath, pale skin. Causes: nutritional deficiencies, diseases, blood loss. Treatments vary.

Hemoglobin
Hemoglobin, a protein in red blood cells, transports oxygen and carbon dioxide. Healthy ranges: Men: 13.2–16.6 g/dL. Women: 11.6–15 g/dL.

MCH
An MCH value refers to the average quantity of hemoglobin present in a single red blood cell. The normal range for MCH is between 27.5 and 33.2 picograms (pg).

MCHC
The mean corpuscular hemoglobin concentration is average concentration of hemoglobin in red blood cells. The normal range: 33.4–35.5g/dL.

MCV
MCV stands for mean corpuscular volume. An MCV blood test measures the average size of red blood cells. The normal range for MCV is between 80 to 100 femtoliter.



Enter Details to Predict

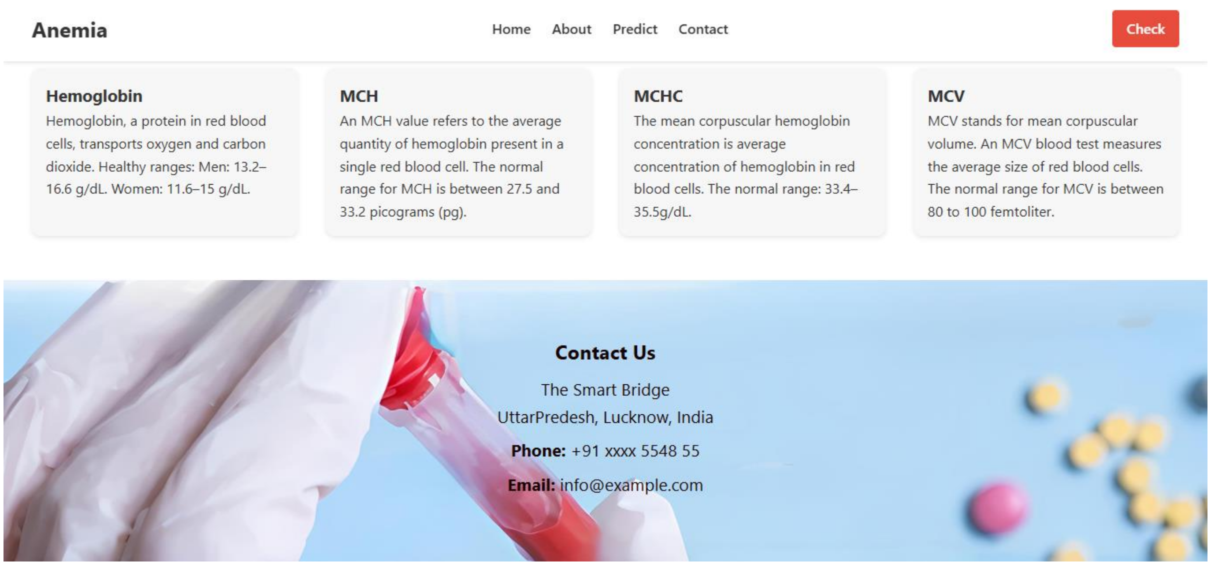
Gender (0: Female, 1: Male)

Hemoglobin(Range:7-16)

MCH(Range:16-30)

MCHC(Range:28-34)

MCV(Range:70-100)



Hemoglobin
Hemoglobin, a protein in red blood cells, transports oxygen and carbon dioxide. Healthy ranges: Men: 13.2–16.6 g/dL. Women: 11.6–15 g/dL.

MCH
An MCH value refers to the average quantity of hemoglobin present in a single red blood cell. The normal range for MCH is between 27.5 and 33.2 picograms (pg).

MCHC
The mean corpuscular hemoglobin concentration is average concentration of hemoglobin in red blood cells. The normal range: 33.4–35.5g/dL.

MCV
MCV stands for mean corpuscular volume. An MCV blood test measures the average size of red blood cells. The normal range for MCV is between 80 to 100 femtoliter.

Contact Us
The Smart Bridge
UttarPredesh, Lucknow, India
Phone: +91 xxxx 5548 55
Email: info@example.com

Conclusion

AnemiaSense demonstrates the potential of machine learning in enhancing healthcare diagnostics by providing precise and timely detection of anemia. Using the XGBoost model, the system achieved high accuracy, making it a reliable tool for identifying anemia at various stages. The integration of this model into a user-friendly web application ensures accessibility for both healthcare professionals and patients, enabling early diagnosis, personalized treatment planning, and effective remote monitoring.

References

1. Kaggle. (n.d.). Anemia Dataset. Retrieved from [https://www.kaggle.com/]
2. Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.
3. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
4. Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90–95.
5. McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference, 51–56.
6. Raschka, S., & Mirjalili, V. (2022). *Machine Learning with PyTorch and Scikit-Learn*. Packt Publishing.