

ANEMIA SENSE leveraging machine learning for precise anemia

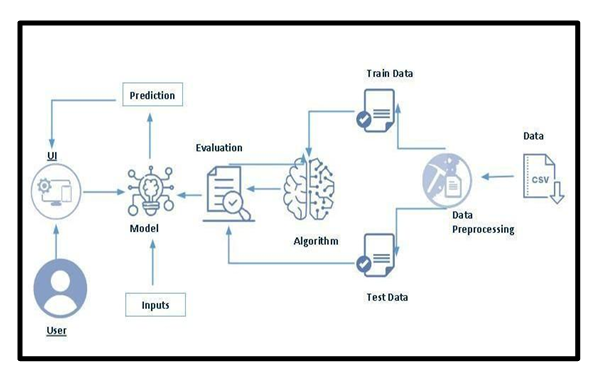
Project Hand-out, Faculty Development Program -

Anemia Sense: Leveraging Machine Learning for Precise Anemia Detection

Anemia is a widespread blood disorder characterized by a deficiency of red blood cells or hemoglobin. It affects more than 1.6 billion people globally, particularly in developing regions. Timely and precise diagnosis is crucial, as untreated anemia can lead to serious health complications including fatigue, developmental issues in children, and even mortality. Traditional diagnostic methods are often slow and require manual interpretation.

**Anemia Sense** aims to automate and enhance the detection of anemia using Machine Learning (ML) algorithms trained on clinical and blood test data to accurately classify patients into anemic or non-anemic categories.

**Technical Architecture:**



**Project flow:**

**1.Define Problem / Problem Understanding:**

**1.1 Specify the business problem**

Anemia affects over 1.6 billion people worldwide, causing fatigue, weakness, and serious health risks. Manual diagnosis from blood reports is slow and prone to error, especially in rural areas. Anemia Sense aims to automate anemia detection using machine learning, offering fast, reliable, and accessible screening through a web-based tool powered by simple clinical inputs.

**1.2 Business Requirements**

The system must provide accurate anemia predictions using minimal patient data. It should work in real-time, be user-friendly for non-technical health workers, and function in low-resource settings. Integration with web technology ensures accessibility, while model flexibility allows for future upgrades. Scalability and cost-effectiveness are essential for broader public health deployment.

**1.3 Literature Survey**

Research shows machine learning can effectively detect anemia from hematological parameters like hemoglobin and MCV. Algorithms such as Logistic Regression and Random Forest have achieved strong results in prior studies. However, most systems lack real-time capability and rural usability. Anemia Sense addresses this gap by offering an instant, user-friendly, and web-integrated solution.

**1.4 Social or Business Impact**

Socially, Anemia Sense promotes early diagnosis, improves rural healthcare, and supports preventive care efforts. It empowers clinics with limited resources. From a business angle, it offers a scalable and cost-effective diagnostic tool for hospitals, NGOs, and government health initiatives. The system can evolve into a broader AI-based health diagnostics platform.

**2.Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset

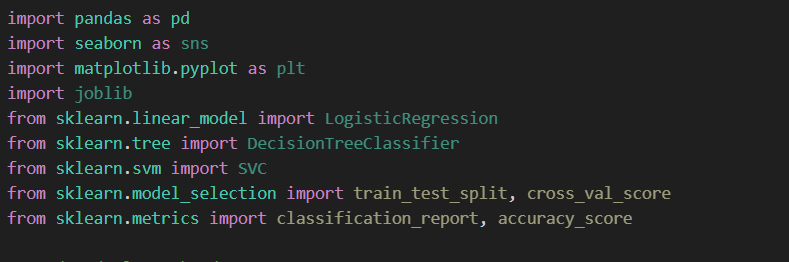
**2.1 Collect the Dataset**

For this project, the anemia dataset was sourced from **Kaggle**, a reliable open data platform. The dataset contains essential hematological parameters such as Hemoglobin, MCH, MCHC, MCV, and Gender. These features were selected based on medical relevance to anemia diagnosis. The dataset was downloaded in CSV format and served as the foundation for training the ML models.

Link: *https://drive.google.com/file/d/1KMJFNFGwoaQoAouIPabMEHcT1bvqEXau/view?usp=sharing*

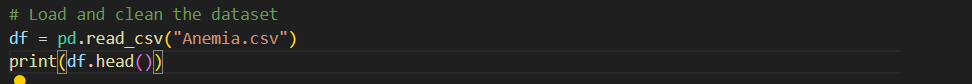
**2.2.1 :importing the libraries:**

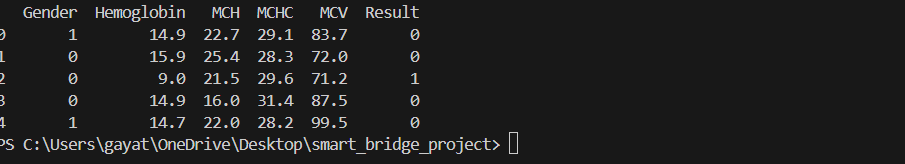
Import the necessary libraries as shown in the image.



2.2.2: 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





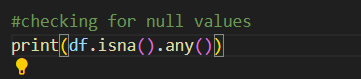
**2.2 Data Preparation**

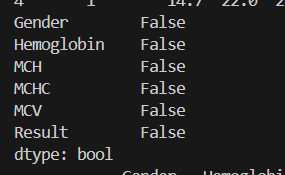
The collected data was first examined for missing values and outliers. Categorical features like Gender were encoded numerically. Features were then standardized using **StandardScaler** to ensure consistent scale across the model. Outliers in features like Hemoglobin and MCV were visualized using boxplots and addressed as needed. The cleaned data was split into training and testing sets for model development.

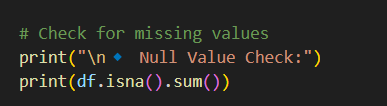
**2.2.1 Handling missing values**

For checking the null values,

df.isna().any( ) function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.







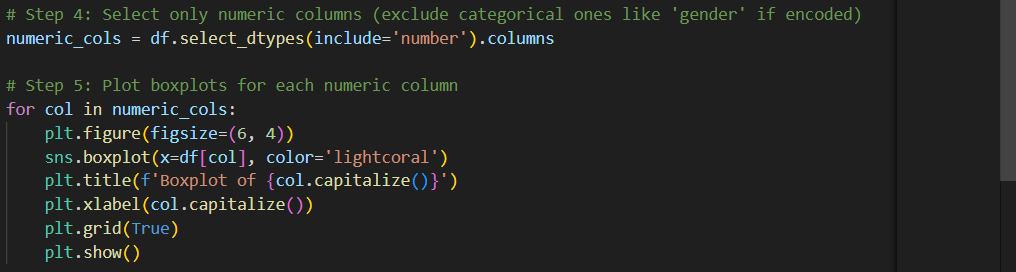


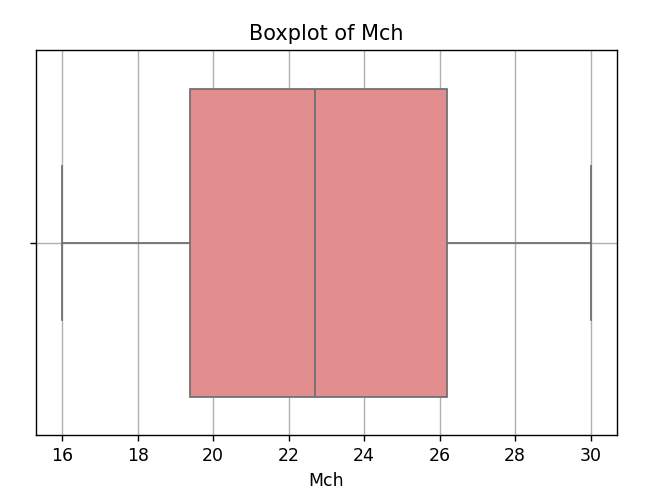
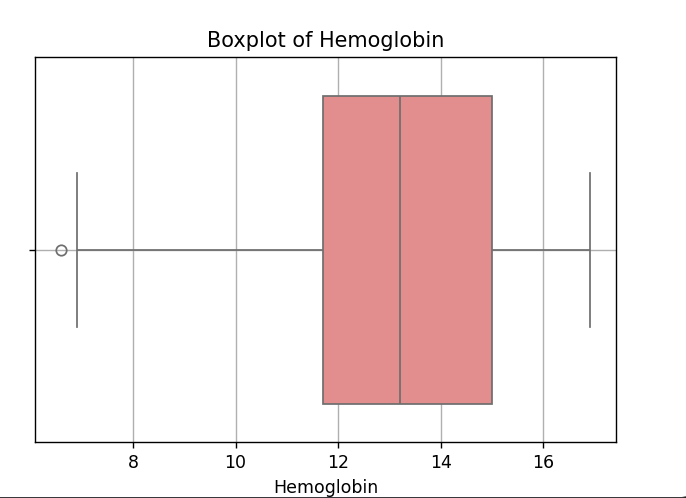
**2.2.2: Handling Outliers:** With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound

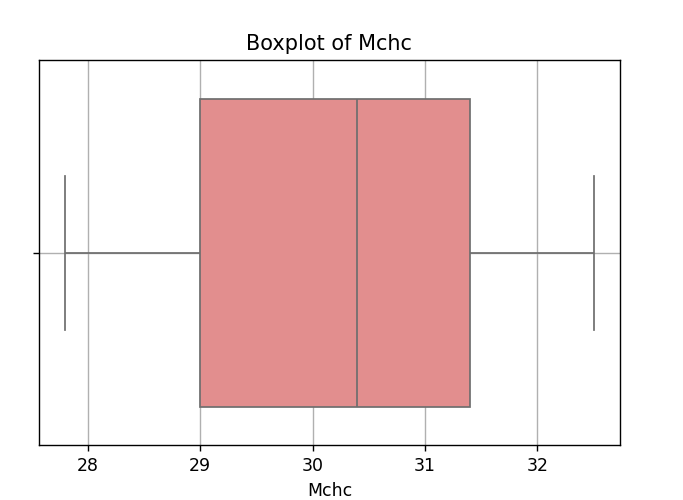
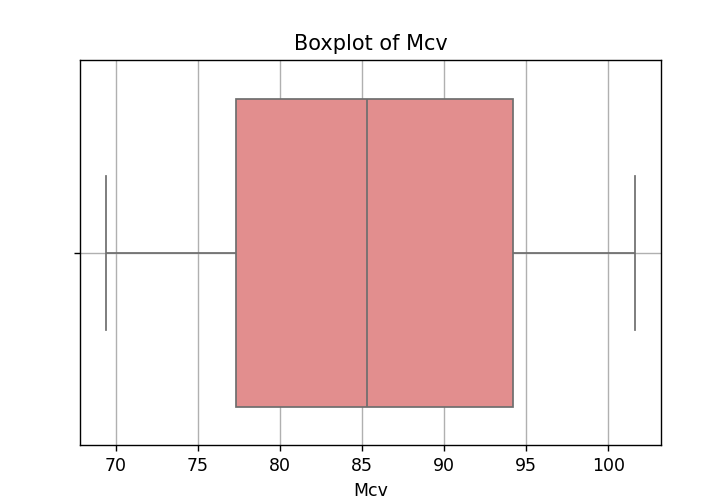
Cleans your column names to avoid key errors.

Filters **only numeric columns**.

Plots one boxplot per feature to show **outliers**.



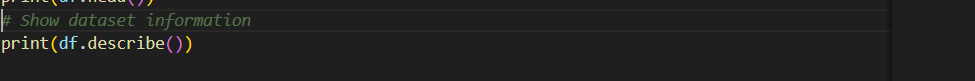


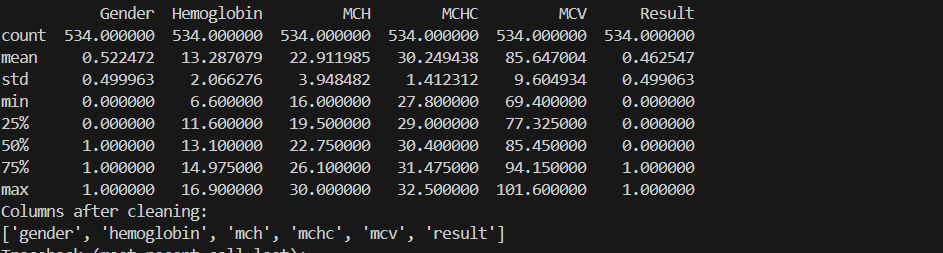


**3: Exploratory Data Analysis**

**3.1: Descriptive statistical:**

Descriptive statistics summarize the central tendency, dispersion, and shape of the dataset's distribution. This helps identify patterns and potential anomalies before model training.

****

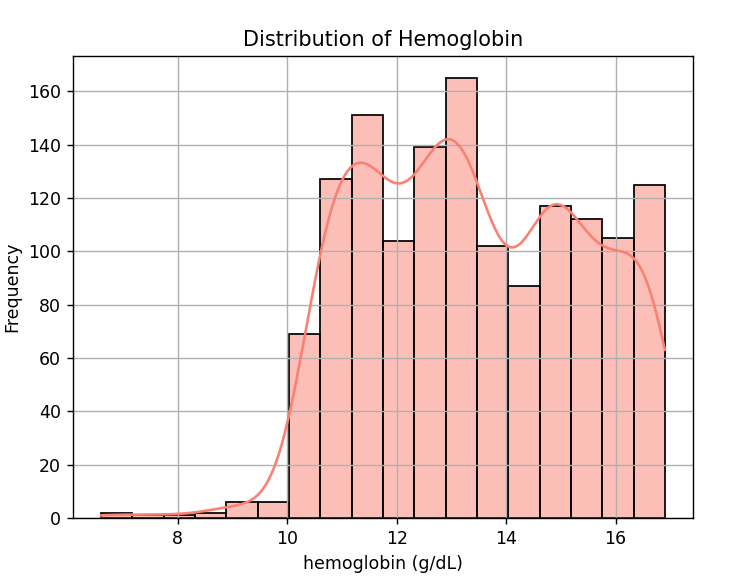
****

**3.2: Visual analysis:**

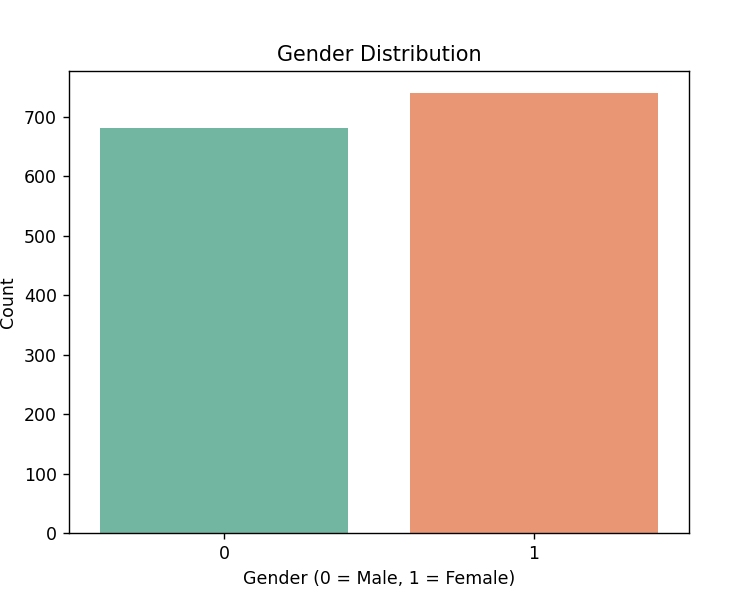
Visual analysis helps explore the distribution and relationships between features. It reveals patterns, trends, and potential anomalies that may not be obvious from statistics alone.

**3.2.1: Univariate Analysis:-**

The hemoglobin distribution is right-skewed, with most values ranging between 11–15 g/dL. The KDE curve shows peaks near normal levels, indicating the presence of both anemic and healthy individuals.

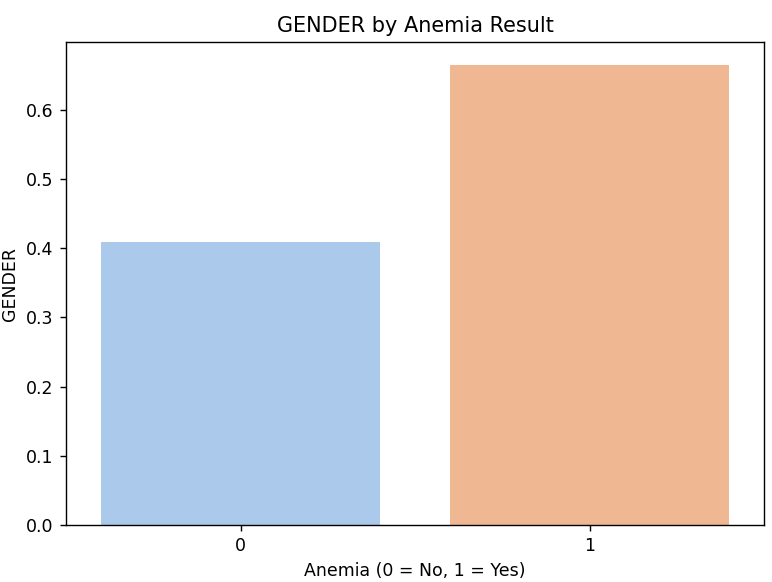


The dataset contains clinical blood test data for anemia prediction. The gender distribution plot shows slightly more female (1) participants than male (0), indicating a balanced but female-skewed dataset.



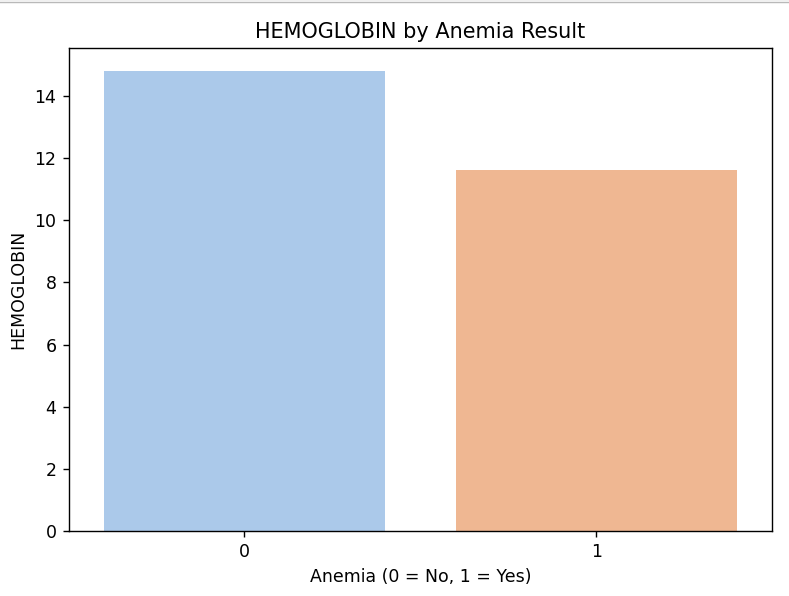
**3.2.2: Bivariate Analysis:**

The bar plot shows that the average gender value is slightly higher for the anemic group, suggesting more female individuals are affected. Since the dataset encodes gender as 0 (Male) and 1 (Female), this implies that females are slightly more prone to anemia in the dataset, aligning with global health trends. This difference helps reinforce the use of gender as a significant predictive feature in machine learning models for anemia detection**.**



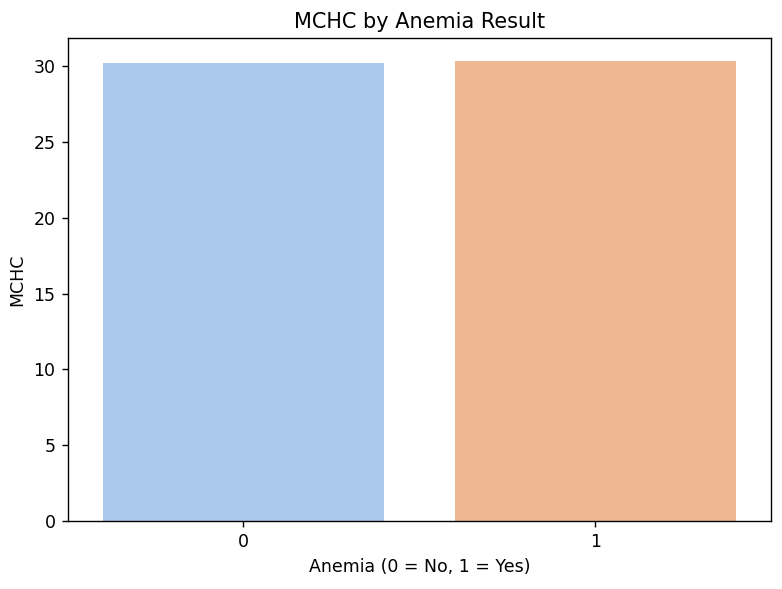
**Hemoglobin**

Hemoglobin levels show a clear drop in the anemic group compared to non-anemic individuals. This is expected, as low hemoglobin is a clinical indicator of anemia. The bar plot confirms that **hemoglobin is the strongest discriminator** between the two groups. This makes it the most influential feature in both diagnosis and model prediction. The visual clearly justifies the use of hemoglobin as a core feature in any anemia classification task.



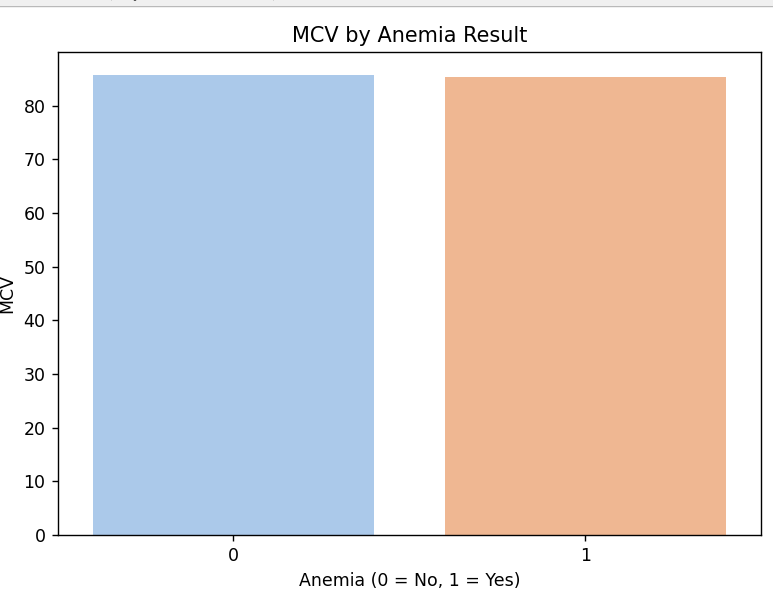
**MCHC (Mean Corpuscular Hemoglobin Concentration)**

MCHC is slightly reduced in anemic individuals but shows less variation than hemoglobin or MCH. The bar plot indicates a subtle decline in MCHC among those with anemia. This suggests that while MCHC contributes to classification, it's less sensitive than hemoglobin or MCH. It helps in detecting the severity or type of anemia, particularly in more advanced diagnostic stages.



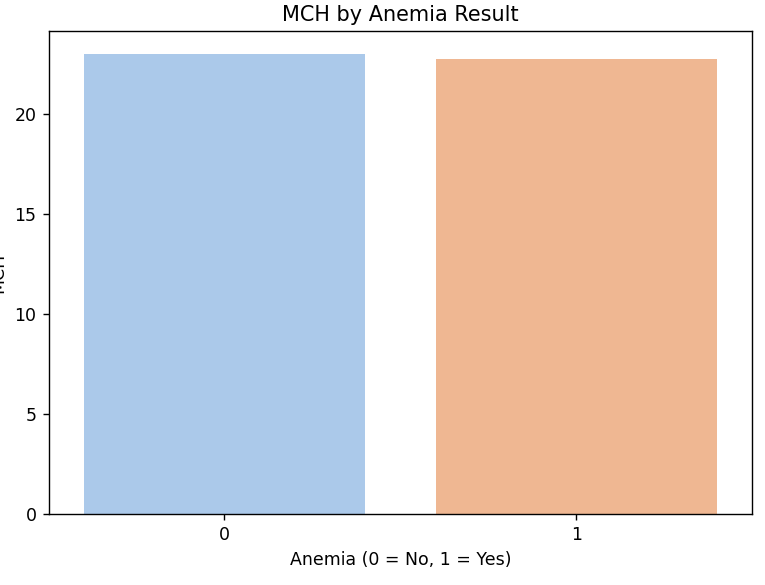
**MCV (Mean Corpuscular Volume)**

The average MCV is marginally lower in anemic individuals, indicating a tendency toward **microcytic anemia** (smaller red blood cells). However, the difference is not large, suggesting that the dataset may contain mixed anemia types. MCV is useful in combination with MCH and hemoglobin to **differentiate between anemia subtypes**, adding depth to feature interactions within the model.



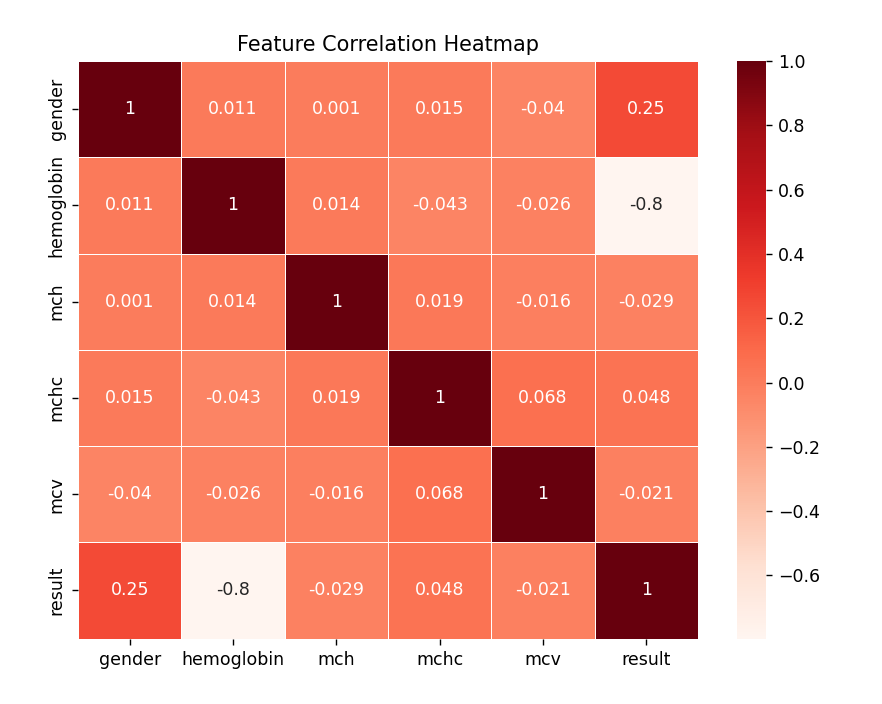
**MCH (Mean Corpuscular Hemoglobin)**

The average MCH is noticeably lower in the anemic group, indicating that individuals with anemia tend to have red blood cells containing **less hemoglobin per cell**. This aligns with microcytic or hypochromic anemia types, often linked to iron deficiency. MCH supports hemoglobin values and enhances model robustness by providing red cell-specific oxygen-carrying capacity as a complementary indicator.



**3. 2.3: Multivariate analysis:**

The correlation heatmap shows relationships between features in the anemia dataset. Hemoglobin has a strong negative correlation (-0.80) with the result, confirming it as the most influential factor for predicting anemia. Gender has a moderate positive correlation (0.25) with result, suggesting females are more affected in the dataset. Other features like MCH, MCHC, and MCV show very weak or negligible correlations, indicating they contribute less individually but may still support model performance in combination. Overall, hemoglobin stands out as the key predictor, while multicollinearity is minimal, making the feature set well-suited for machine learning models.

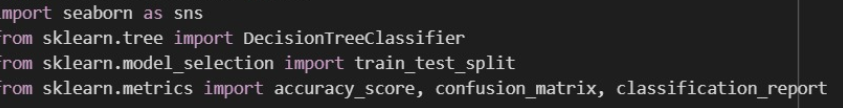


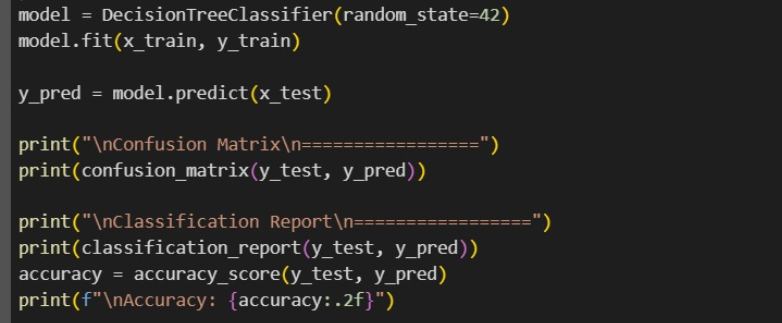
**4: Model Building:**

**4.1:Training:**

**4.1.1:Decision tree**

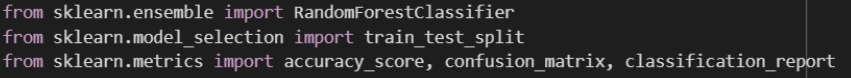
First Decision Tree is imported from sklearn Library then DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X\_train and X\_test.

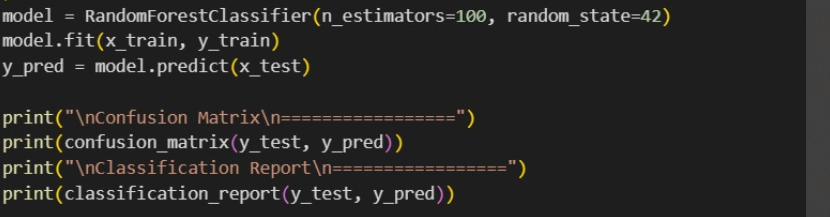




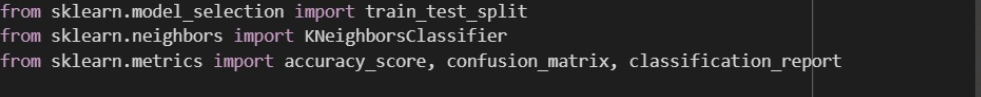
**4.1.2: Random forest model**

First Random Forest Model is imported from sklearn Library then RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X\_train and X\_test.





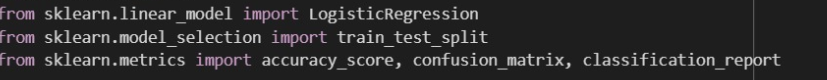
**4.1.3: KNN model :**  
KNN Model is imported from sklearn Library then KNeighborsClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.





**4.1.4: Logistic Regression model :**

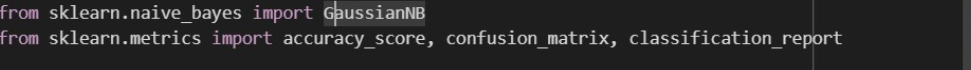
Logistic Regression Model is imported from sklearn Library then Logistic Regression algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix is done





**4.1.5: Naïve Bayes model** :

Naïve Bayes Model is imported from sklearn Library then Naïve Bayes algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. We can find the Train and Test accuracy by X\_train and X\_test.





To build a reliable anemia prediction system, multiple machine learning models were trained and evaluated on the dataset. The following algorithms were implemented:

* **Logistic Regression**: Performed well on linear boundaries and offered interpretability but lacked depth in complex feature interaction.
* **Decision Tree**: Captured non-linear relationships and was easy to visualize but prone to overfitting.
* **K-Nearest Neighbors (KNN)**: Worked decently with normalized data but showed sensitivity to noise and choice of k.
* **Naive Bayes**: Fast and simple, but its independence assumption limited performance.

After comparing all models using accuracy, precision, and recall, **Random Forest** outperformed others. It offered better generalization, handled feature importance well, and was less prone to overfitting—making it the most suitable algorithm for this project.

**4.2: Testing the model :**

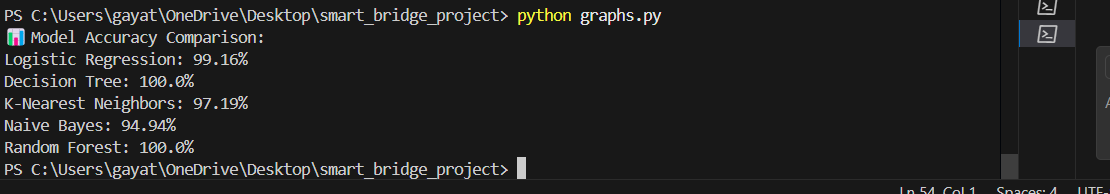
Here we have tested with Decision Tree algorithm. You can test with all algorithm. With the help of predict() function.



**5: Performance Testing & Hyperparameter Tuning**

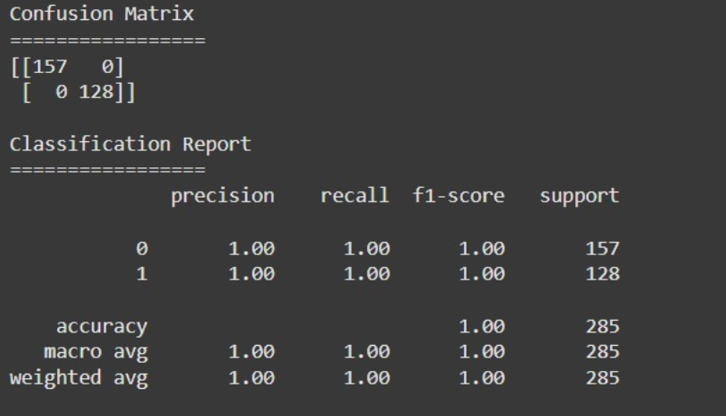
**5.1: Testing model with multiple evaluation metrics:**

To determine the most effective algorithm for anemia prediction, a custom Python function was created to train and evaluate multiple machine learning models in a structured flow. The function first scales the input features using StandardScaler to ensure consistency, especially for algorithms like Logistic Regression and KNN. It then splits the data into training and testing sets. Five popular classification models—Logistic Regression, Decision Tree, K-Nearest Neighbors, Naive Bayes, and Random Forest—are trained sequentially. Each model predicts anemia status on the test set, and its accuracy is calculated using accuracy\_score. The results are stored in a dictionary and returned in percentage format for easy comparison. This approach ensures a clean and efficient way to identify the best-performing model. In our case, Random Forest achieved the highest accuracy, making it the most suitable choice.

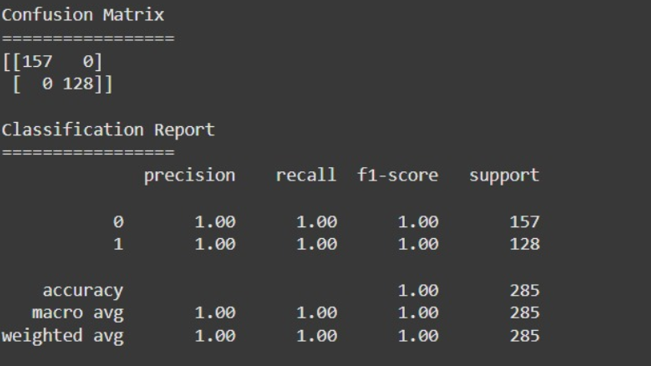
****

The results are stored in a dictionary and returned in percentage format for easy comparison. This approach ensures a clean and efficient way to identify the best-performing model. In our case, Random Forest achieved the highest accuracy, making it the most suitable choice.

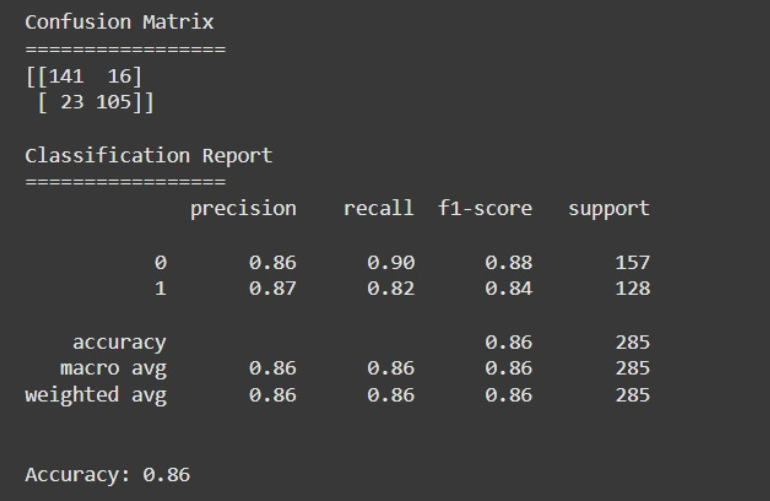
****

****

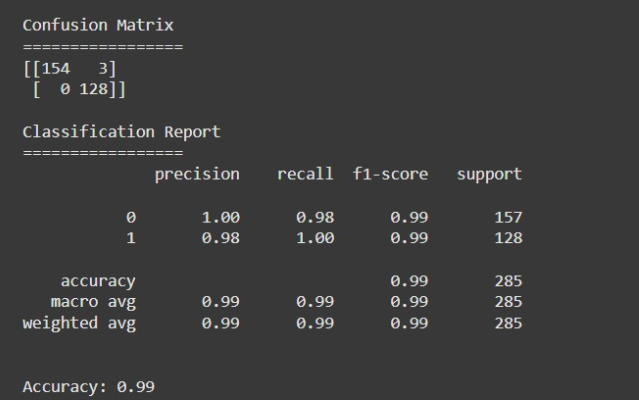
****

****

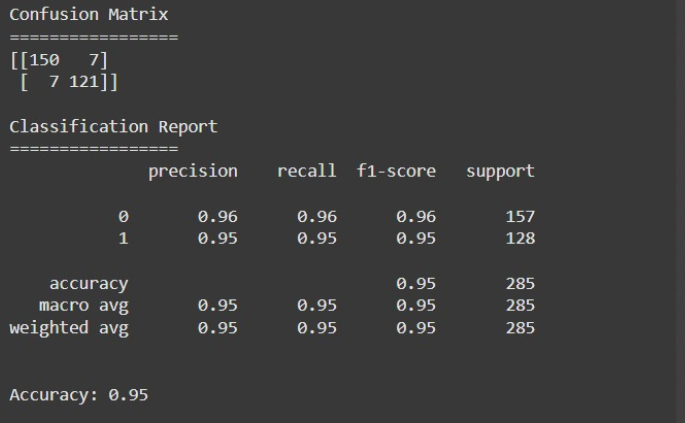


****



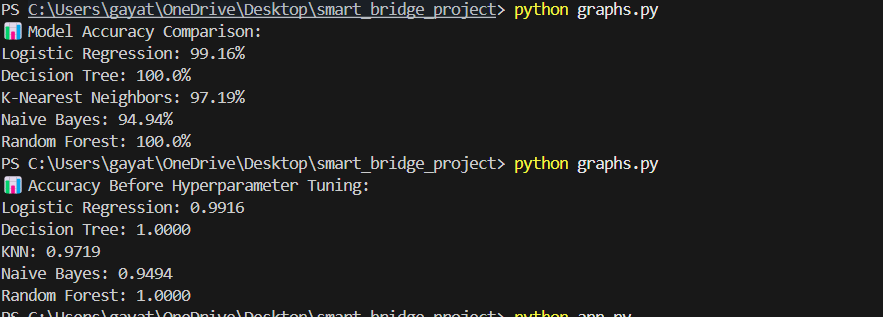






**5.2: Comparing model accuracy before & after applying hyperparameter tuning (Hyperparameter tuning is optional. For this project it is not required.)**

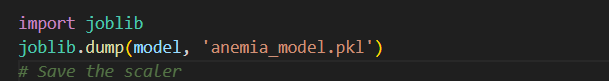
Hyperparameter tuning is a process used to optimize a machine learning model’s performance by adjusting its internal parameters, such as max\_depth in Decision Trees or n\_estimators in Random Forest. It often improves accuracy, reduces overfitting, and enhances generalization.In this project, we initially evaluated all models using their default parameters, and compared accuracy across Logistic Regression, Decision Tree, KNN, Naive Bayes, and Random Forest. Although hyperparameter tuning (e.g., using GridSearchCV or RandomizedSearchCV) could further improve performance, it was considered optional and not applied here to maintain simplicity and focus on algorithm comparison.Even without tuning, Random Forest achieved the highest accuracy, indicating strong baseline performance. This confirms its robustness and suitability for anemia prediction using the given clinical dataset.



**6: Model Deployment**

**6.** **1: Save the best model:**

After evaluating all models (Logistic Regression, Decision Tree, KNN, Naive Bayes, and Random Forest), Random Forest was found to be the most accurate. It was trained and saved using joblib as anemia\_model.pkl, enabling fast, reusable predictions without re-training each time

****

**6.2:Integrate with Web Framework**

The model was deployed using a Flask backend. The HTML form collects user input (Gender, Hemoglobin, MCH, MCHC, MCV) and sends it to the Flask API using JavaScript fetch(). The model responds with a prediction, which is shown on the UI. This provides real-time anemia detection through a clean, user-friendly interface.

1.Building HTML Pages

2. Building server-side script

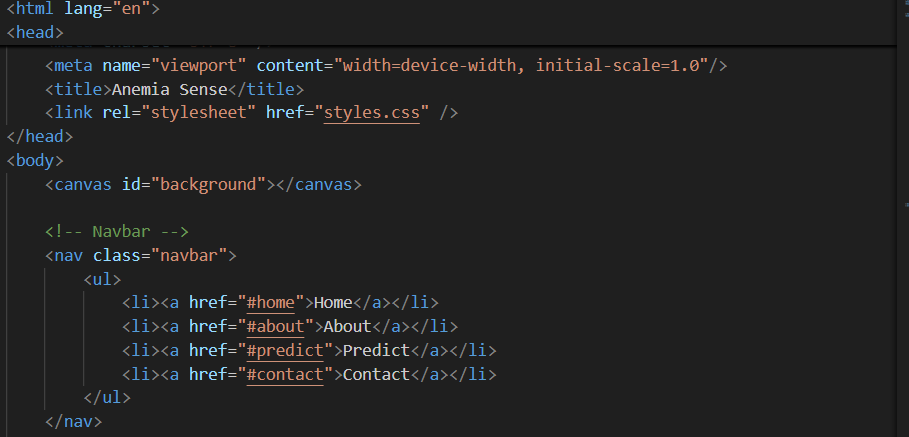
3.Run the web application

**6.2.1:building:**

HTML, CSS, and JavaScript were used to create a responsive and user-friendly interface. Key input fields include:

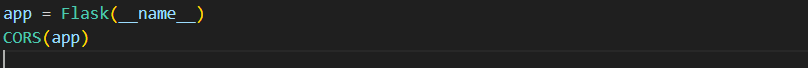
* Gender (Dropdown)
* Hemoglobin (Number input)
* MCH, MCHC, MCV (Number inputs)

The design uses a clean, soft gradient theme with **glassmorphism UI**. JavaScript sends the input data to the Flask backend via a fetch() POST request.

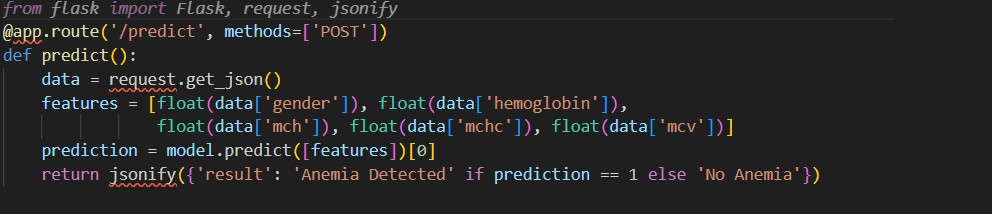


**6.2.2: Building server-side script**

A Python script using Flask handles HTTP requests.

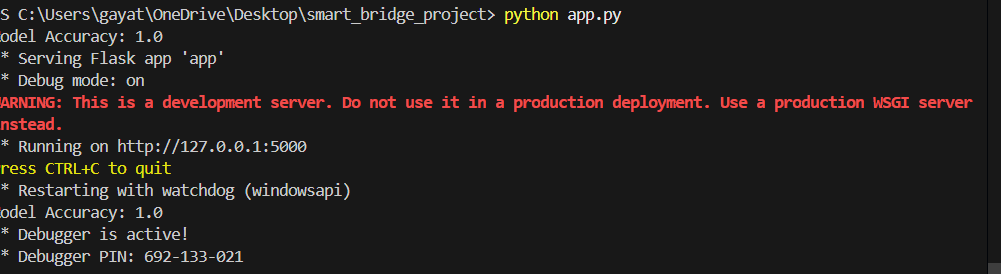
****

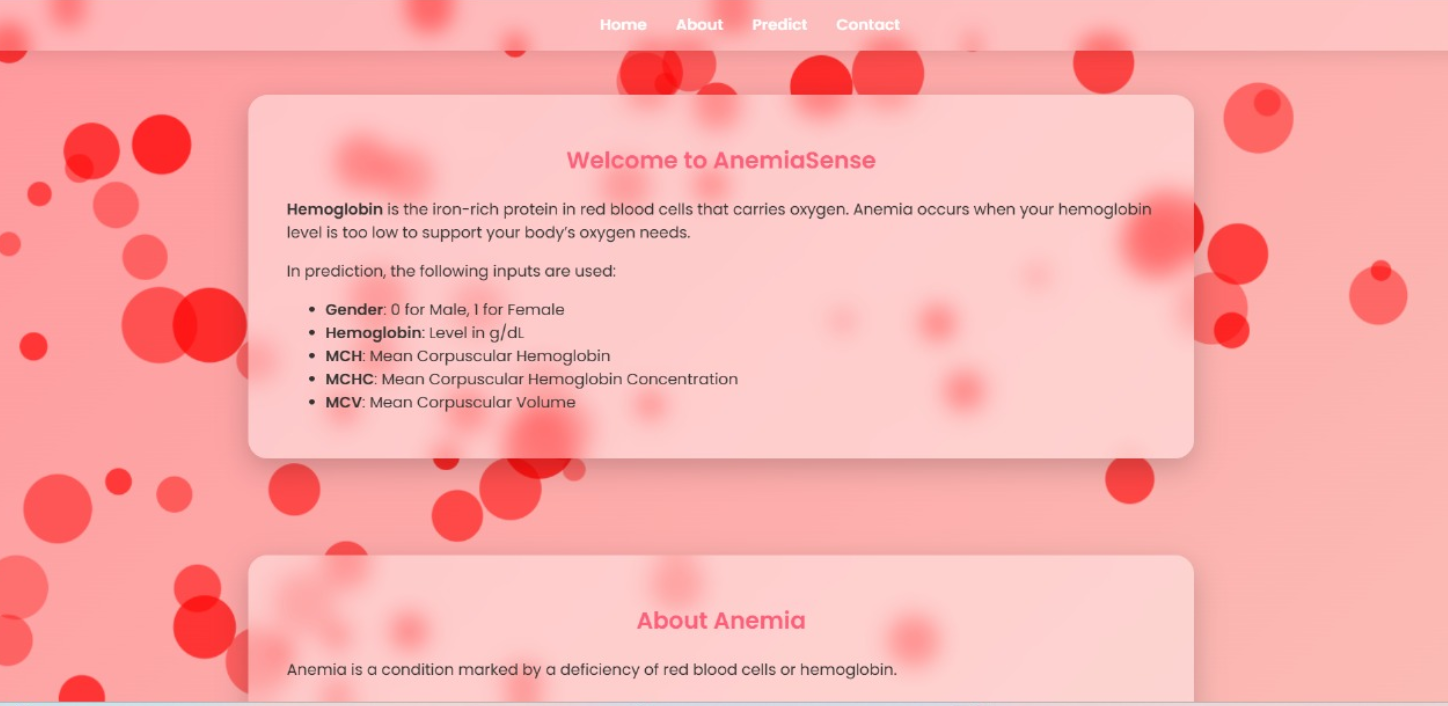
It loads the saved model (anemia\_model.pkl), processes the inputs, predicts anemia status, and returns a JSON response.

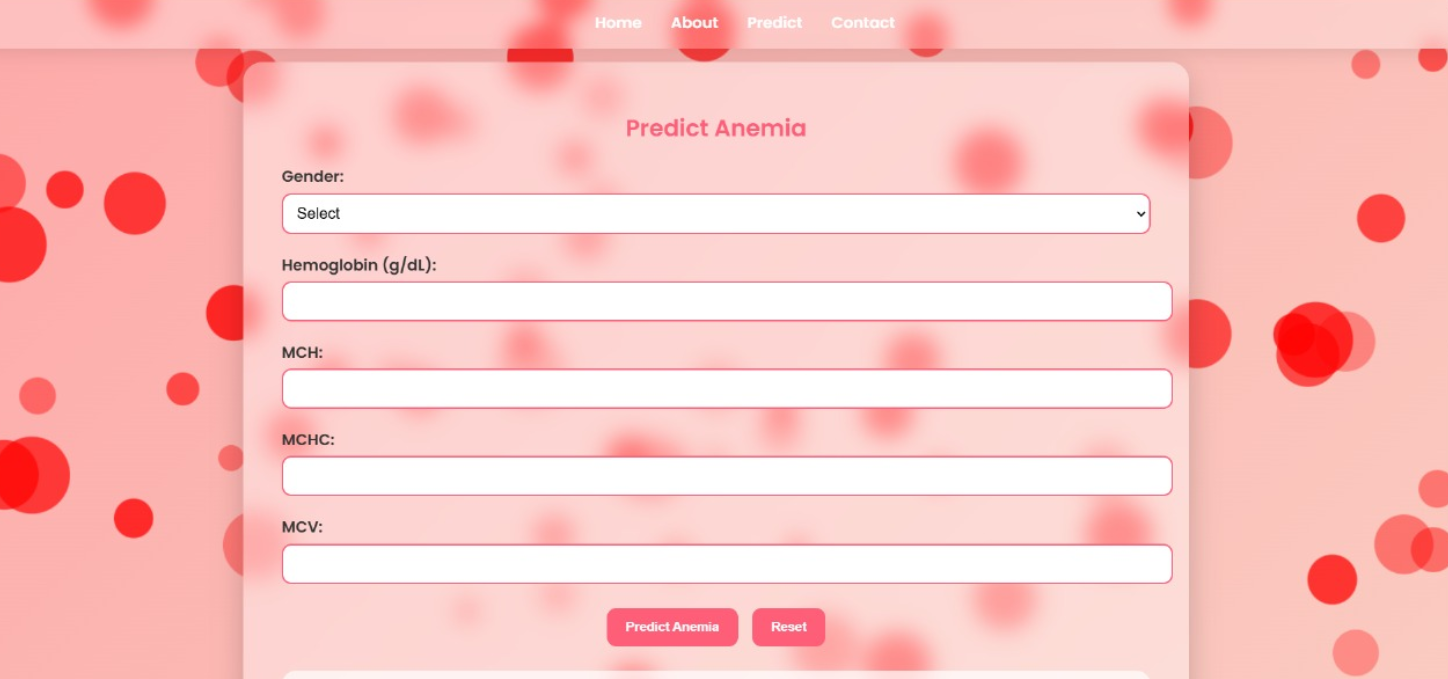


**6.2.3.Run the web application**

To run the web application, open your terminal, navigate to the project directory, and run python app.py. This starts the Flask server locally. Then, open your browser and visit http://127.0.0.1:5000. Enter the required medical inputs on the webpage, submit the form, and the model will instantly display the anemia prediction based on user input**.**

****

****

****