# Anemia Sense: Leveraging Machine Learning For Precise Anemia Recognitions

**Anemia Sense**

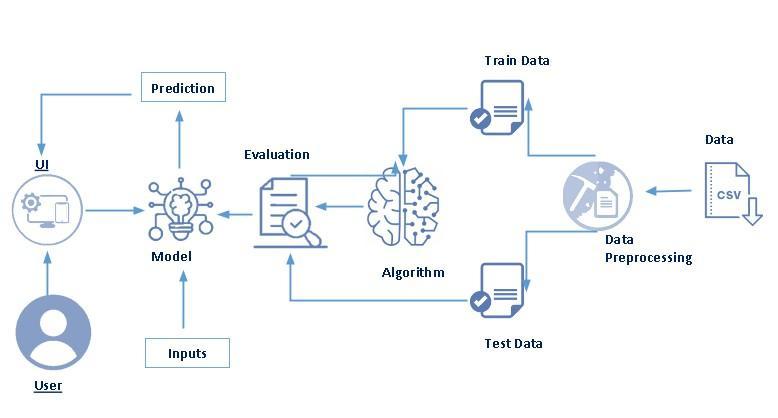
Anemia Sense is a healthcare innovation project that utilizes machine learning for the accurate detection and management of anemia — a condition characterized by insufficient hemoglobin or red blood cells. By analyzing blood parameters and patient-specific data, Anemia Sense offers early diagnosis, personalized treatment plans, and continuous remote monitoring.

This platform is designed to:

- Detect anemia early using blood data

- Easy UI for immediate implementation

**Technical Architecture:**



**Project Flow :**

- The user inputs clinical blood data via a web interface

- The integrated ML model processes the data

- The prediction result (anemia type or severity) is displayed on the UI

**To accomplish this, we follow project stages which are designed in a particular order for a proper outcome**

**Project Stages**

- Define Problem / Problem Understanding

- Data Collection & Preparation

- Exploratory Data Analysis

- Model Building

- Performance Testing & Hyperparameter Tuning

- Model Deployment

- Project Demonstration & Documentation

**Prior Knowledge:**

To effectively understand and implement this project, participants should possess prior knowledge in the following key areas:

- ML Concepts: Supervised and Unsupervised Learning - A fundamental understanding of machine learning paradigms, particularly supervised learning (for tasks such as classification and regression) and unsupervised learning (for tasks like clustering and dimensionality reduction), is essential. This includes knowing when and how to apply different learning approaches.

- Decision Tree, Random Forest, KNN, Logistic Regression, SVM: Familiarity with the theoretical underpinnings and practical application of these common classification algorithms is crucial. This involves understanding their strengths, weaknesses, and appropriate use cases in predictive modeling.

- Evaluation Metrics: Accuracy, Precision, Recall, F1-Score: A strong grasp of model evaluation metrics is necessary to assess the performance of machine learning models accurately. Understanding how to interpret these metrics (especially in the context of imbalanced datasets) is vital for selecting the best-performing model.

- Flask Web Development: Basic knowledge of Flask, a Python micro-framework, is required for integrating the trained machine learning model into a web application. This includes understanding routing, templating (Jinja2), handling HTTP requests (GET/POST), and deploying simple Flask applications.

- Data Processing & Visualization in Python: Proficiency in Python libraries such as Pandas for data manipulation and NumPy for numerical operations is critical. Additionally, experience with data visualization libraries like Matplotlib and Seaborn for exploratory data analysis and presenting insights is highly beneficial.

**Project Structure:**

● Flask application (app.py)

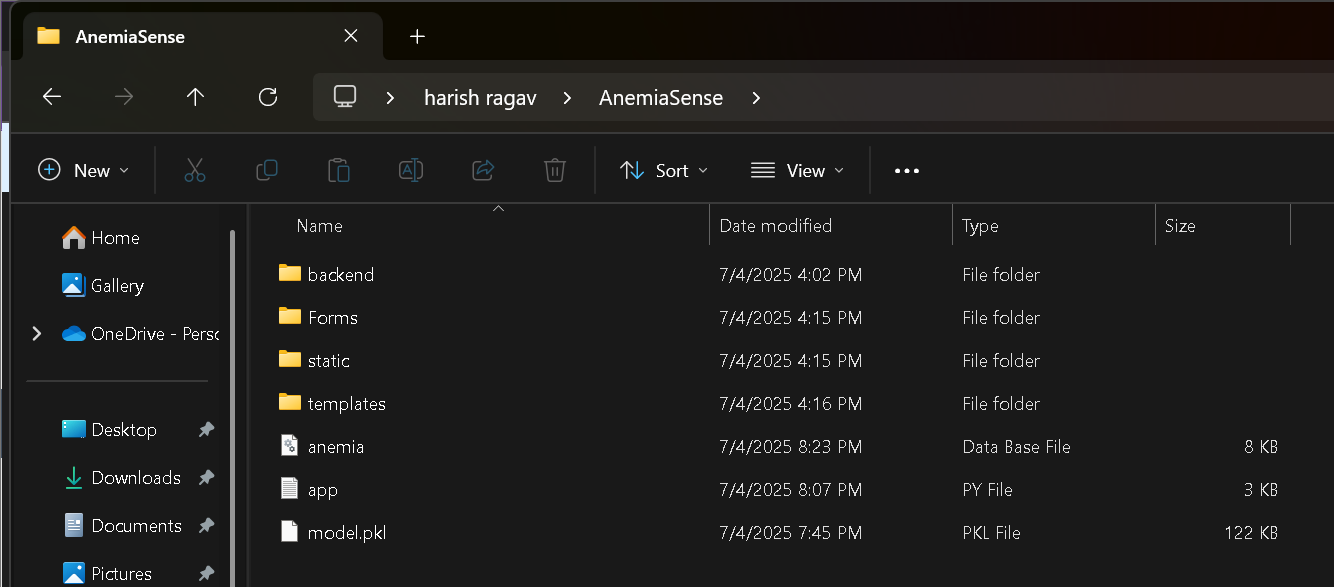
● HTML templates folder (templates)

● Trained model (model.pkl)

● Dataset folder (AnemiaSense)

● Jupyter Notebook for model development (Anemia\_Sense.ipynb)

**Create a folder that includes**



**Milestone-1: Problem Understanding**

**Activity-1: Specify the Business Problem**

Anemia is a global health concern affecting a significant portion of the population, especially in low-resource settings. It is characterized by a deficiency of red blood cells or hemoglobin, leading to fatigue, weakness, and other health complications. Traditional diagnostic methods are often manual, time-consuming, and require in-person clinical visits, which may delay timely diagnosis and treatment.

The primary business problem addressed by the Anemia Sense project is the **lack of fast, accessible, and data-driven systems for precise anemia prediction**. Many existing approaches rely solely on hemoglobin levels or rule-based systems that fail to account for the broader clinical context. There is a growing need for an intelligent system that can accurately predict anemia using multiple hematological features, reducing the burden on healthcare systems and improving patient outcomes.

Anemia Sense leverages machine learning to provide accurate, early-stage predictions based on key blood parameters such as Hemoglobin, MCV, MCH, and MCHC. By doing so, it helps healthcare providers identify at-risk individuals promptly and enables effective prioritization of further diagnostic procedures.

**Activity-2: Business Requirements**

To address the specified business problem, Anemia Sense must fulfill the following requirements:

* **High Prediction Accuracy**: The system must deliver reliable and consistent predictions across varied datasets to assist healthcare professionals.
* **Feature-Based Classification**: Predictions should be based on a combination of hematological markers, not just hemoglobin.
* **Simple UI for Input and Results**: A user-friendly web interface should allow data entry and display predictions without requiring technical expertise.
* **Rapid Prediction**: The model should be optimized for speed, allowing real-time inference.
* **Scalability**: The solution should be extendable to integrate additional features (e.g., genetic data, lifestyle indicators) in the future.

**Activity-3: Literature Survey**

Recent studies and published research in the field of medical diagnostics emphasize the growing potential of machine learning in healthcare. In the context of anemia detection, various algorithms have been applied to classify anemia types or predict its presence using parameters such as MCV, MCH, RDW, and hemoglobin levels. However, many models remain either overly simplistic or lack generalizability across populations.

Several academic works highlight the need for better model interpretability and integration into clinical workflows. While rule-based expert systems exist, they often fail to handle exceptions or learn from new data. Anemia Sense addresses these limitations by using flexible, trainable machine learning models that can adapt to new data and offer better predictive performance.

**Activity-4: Social or Business Impact**

**Social Impact:**

* Empowers early identification of anemia in underserved populations.
* Reduces the dependency on lab-based testing where infrastructure is limited.
* Contributes to better management of chronic diseases where anemia is a comorbidity.

**Business Impact:**

* Reduces diagnostic delays and associated costs in clinics and hospitals.
* Enables integration into telemedicine platforms for scalable use.
* Offers opportunities for healthcare startups to expand into AI-driven diagnostics.

**Milestone-2: Data Collection & Preparation**

**Activity-1: Dataset Collection**

The dataset used in the Anemia Sense project is sourced from structured clinical data, containing essential hematological indicators.

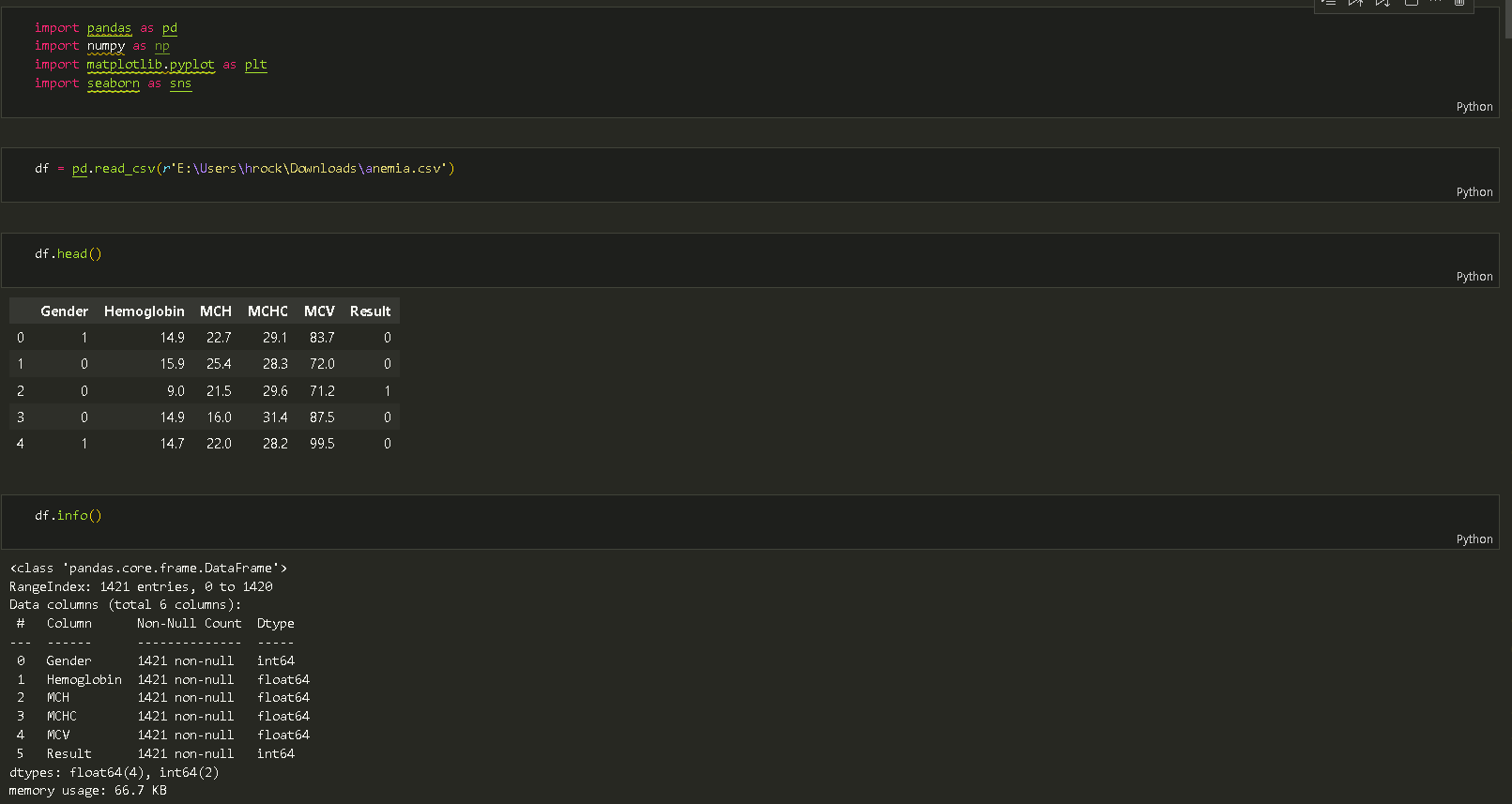
**Data Source:**

* Manually curated dataset from healthcare records (or research sources like Kaggle/UCI if applicable)

**Features:**

* **Gender**: Encoded as 0 (Female) and 1 (Male)
* **Hemoglobin**: Concentration of hemoglobin in g/dL
* **MCH**: Mean Corpuscular Hemoglobin, indicating the average mass of hemoglobin per red blood cell (pg)
* **MCHC**: Mean Corpuscular Hemoglobin Concentration (g/dL)
* **MCV**: Mean Corpuscular Volume (fL)
* **Result**: Binary classification — 0 for non-anemic and 1 for anemic

**Activity 1.1: Importing Libraries and reading Dataset:**



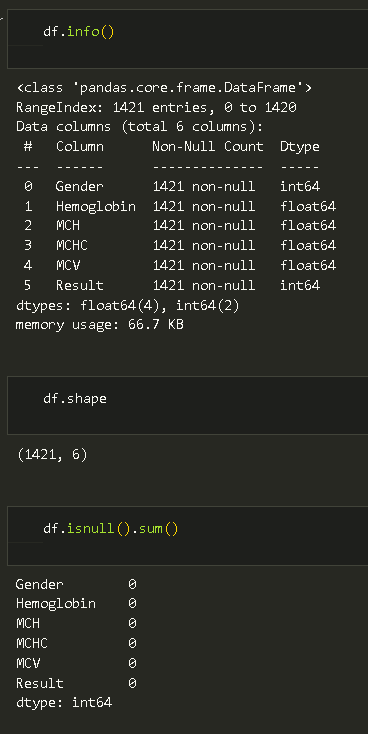
**Activity-2: Data Preparation**

Once collected, raw data often contains inconsistencies, missing values, and noise that can negatively impact model performance. This activity involves a series of pre-processing steps to clean and transform the data into a suitable format for machine learning.

● Handling Missing Values:

○ The dataset is thoroughly checked for any missing or null values using functions like df.isna().any() and .sum().

○ If missing values are identified, appropriate imputation techniques (e.g., mean, median, mode imputation, or more advanced methods) are applied to ensure data completeness. For this project, if no null values are found, this step is skipped.



**Handling Outliers:**

- Outliers, which are data points significantly different from other observations, are identified using visualization techniques such as box plots.Mathematical formulas, such as the Interquartile Range (IQR) method (where Upper Bound = Q3 + 1.5 \* IQR and Lower Bound = Q1 - 1.5 \* IQR), are used to define boundaries for outlier detection.

- Transformation techniques, like log transformation, are applied to mitigate the impact of outliers and normalize data distribution, improving model stability.

**Encoding Categorical Variables:**

-Machine learning models typically require numerical input. Categorical features (e.g., soil type, crop variety) are converted into numerical representations using encoding techniques.

- LabelEncoder from the sklearn.preprocessing library is utilized to transform categorical features into numerical ones based on their unique values.

**Scaling Numerical Features:**

-Features with different scales can disproportionately influence machine learning algorithms. Scaling ensures all numerical features contribute equally to the model.

-StandardScaler is applied to transform the data to have a mean of 0 and a standard deviation of 1, following the formula: Xscaled =(X−Xmean )/Xstd .

**Balancing the Dataset using SMOTE:**

**-In real-world agricultural datasets, certain outcomes (e.g., pest outbreaks) might be rare, leading to imbalanced classes. Imbalanced data can cause models to be biased towards the majority class.**

**-The Synthetic Minority Over-sampling Technique (SMOTE) is employed to oversample the minority class, creating synthetic samples to balance the dataset and improve the model's ability to predict rare events accurately.**

**Splitting into Training and Testing Datasets:**

-The prepared dataset is divided into two subsets: a training set and a testing set.

-The target variable (e.g., yield prediction, crop recommendation) is separated into y, while the remaining features form x.

-train\_test\_split() from sklearn.model\_selection is used, with parameters for test\_size (e.g., 20-30% for testing) and random\_state for reproducibility. This split ensures the model is evaluated on unseen data.

**Milestone-3: Exploratory Data Analysis**

**Activity-1: Descriptive Statistics**

Exploratory Data Analysis (EDA) is a crucial step to understand the underlying structure of the data, identify patterns, detect anomalies, and test hypotheses. It involves both statistical and visual methods.



**Activity-2: Visual Analysis**

- Univariate: Histograms, Countplots for categorical values (e.g., Gender)

- Bivariate: Boxplots (e.g., MCV vs Anemia type), Scatterplots

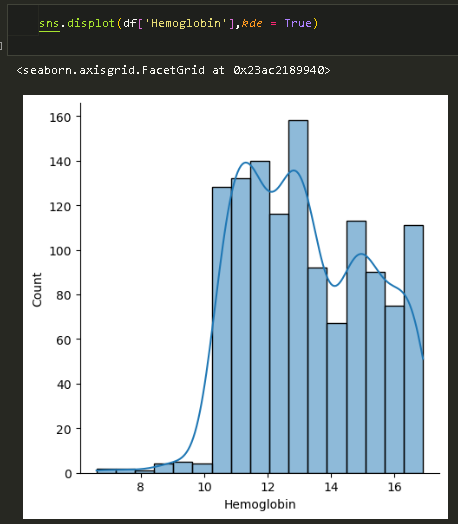
- Multivariate: Correlation heatmap between all blood metrics

**Activity2.1: Univariate:**

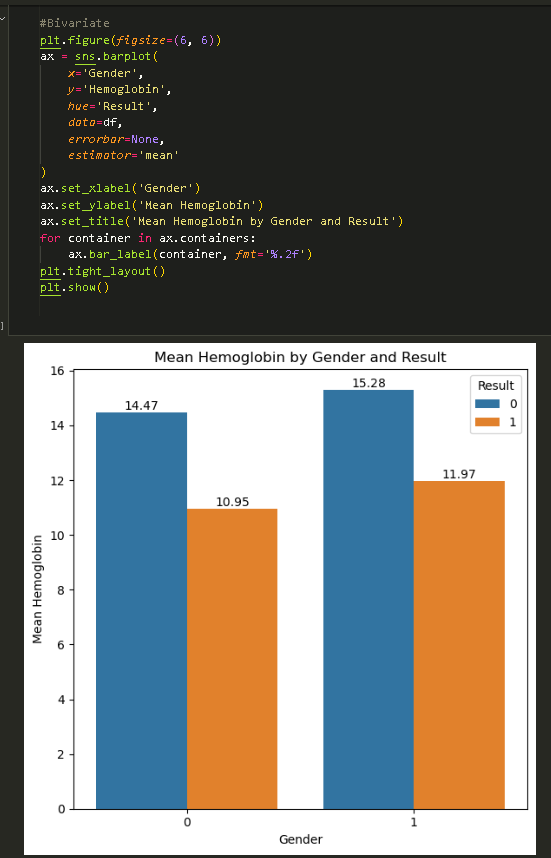
**1.CountPlot**

****

**2.Histograms:**

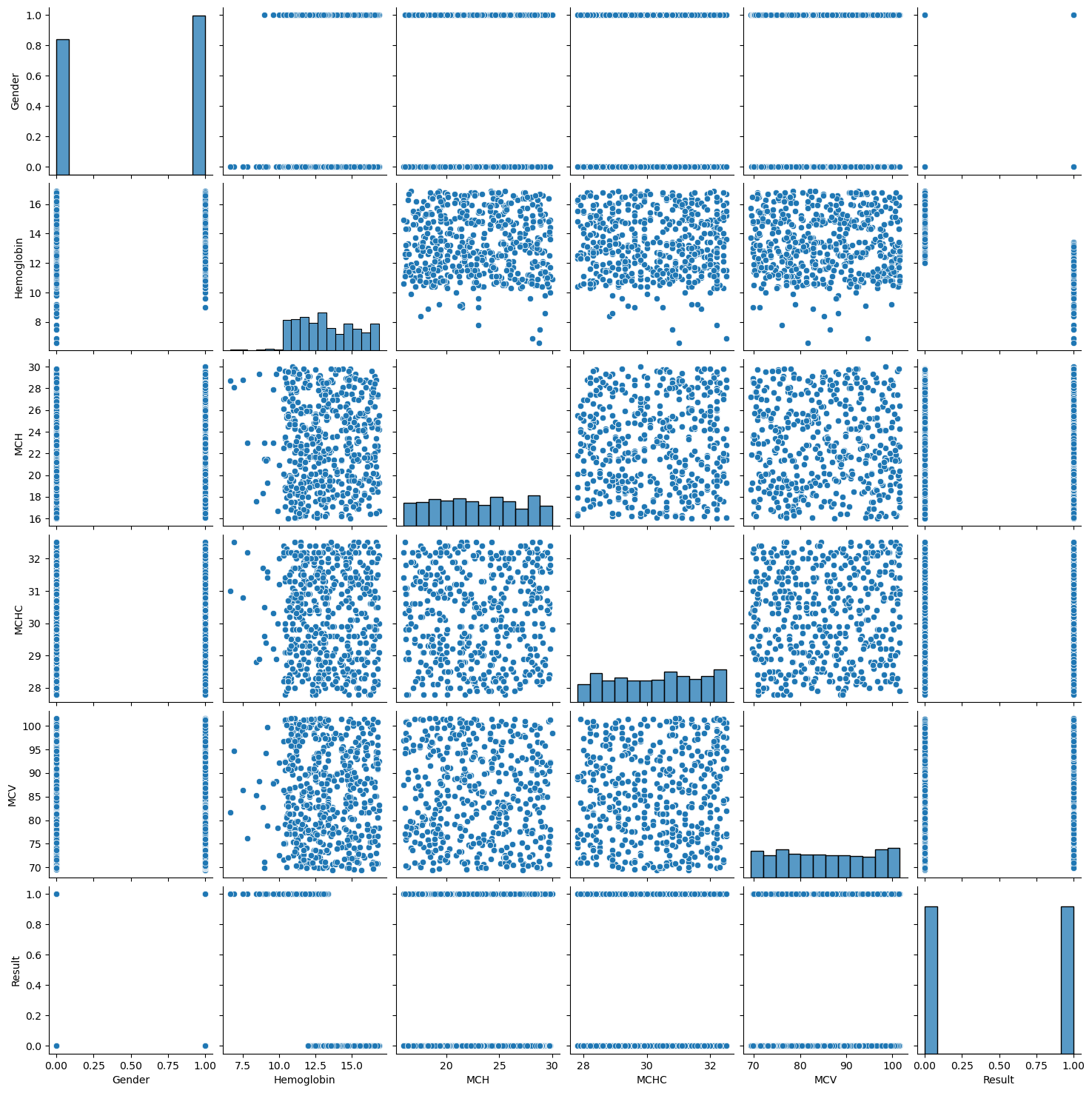
****

**Activity2.2: Bivariate:**

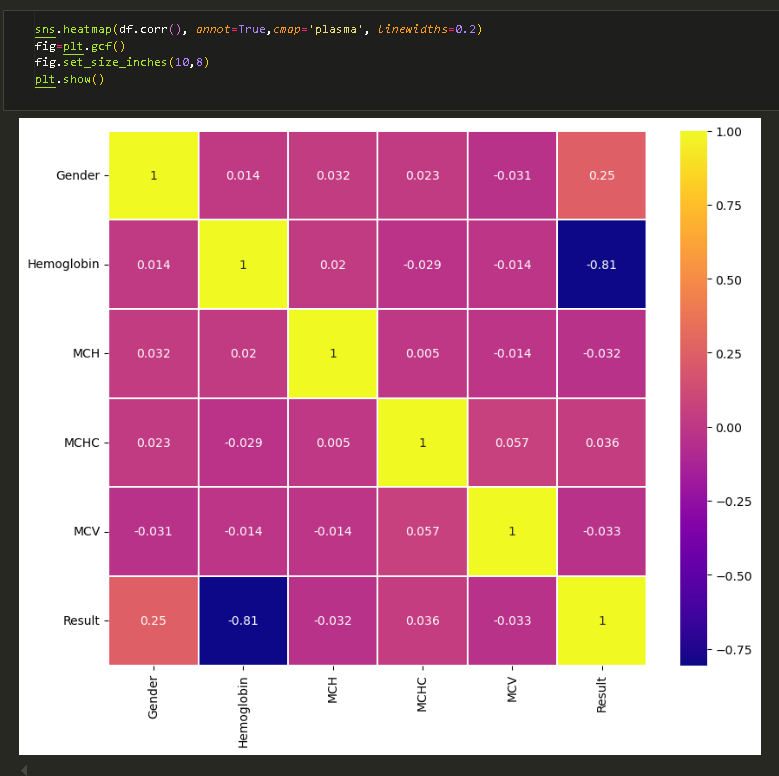


**Activity 2.3: Multivariate analysis**

**1.Pair-plot:**

****

**2.Heat Map:**

****

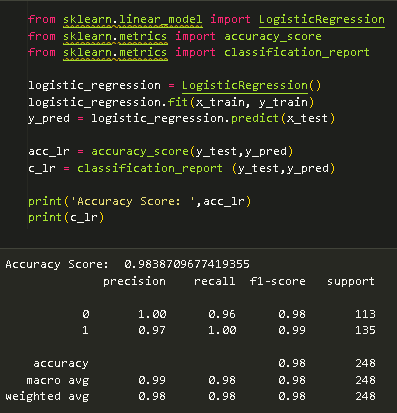
**Milestone-4: Model Building**

**Activity-1: 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.

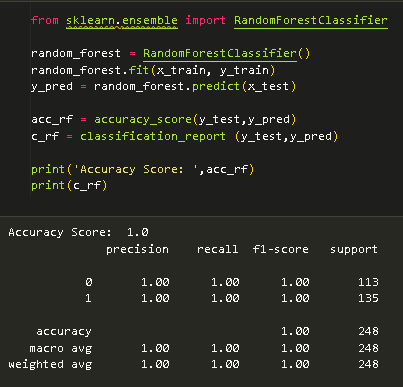
**Logistic Regression Model:**

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.



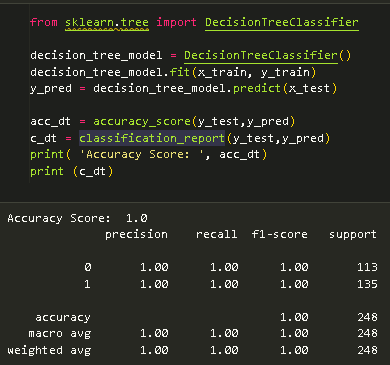
**Random forest model**

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.



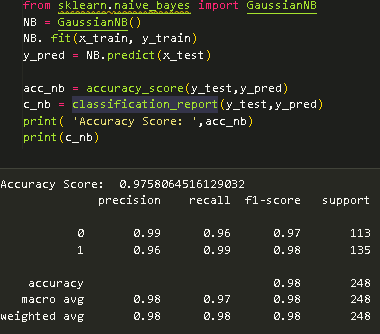
**Decision Tree Model**

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.



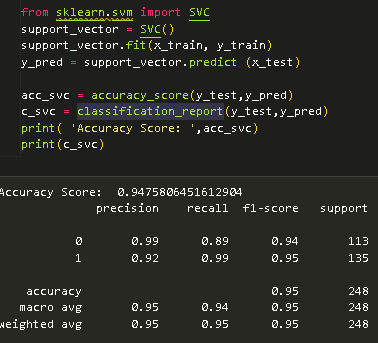
**Gaussian Navies Bayes**

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.



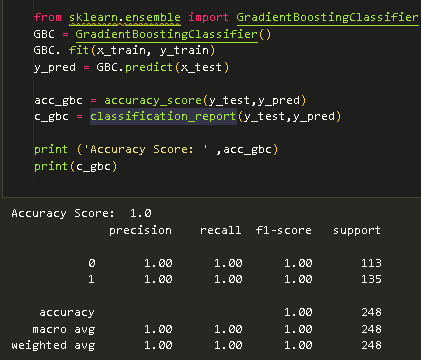
**Support Vector Machine**

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.



**Milestone 5: Performance Testing & Hyperparameter Tuning**

**Activity-1: Compare Models**

After training multiple models, it's critical to systematically compare their performance across various evaluation metrics to identify the most suitable model for the OptiCrop application.

● **Comprehensive Evaluation:** Instead of relying on a single metric, models are tested with multiple evaluation metrics (accuracy, precision, recall, F1-score, and support) to gain a holistic understanding of their strengths and weaknesses.

● **compareModel Function:** A custom function, compareModel, is defined to streamline the comparison process. This function takes the trained models and test data as input, calculates all relevant metrics for each model, and presents them in a structured format (e.g., a DataFrame or a table).

● **Result Analysis:** The output of the compareModel function displays the performance metrics for each algorithm (Decision Tree, Random Forest, KNN, Logistic Regression, SVM).

● **Model Selection:** Based on this comprehensive comparison, the **Random Forest** model is identified as the best performer. It typically offers a strong balance between overall accuracy and robust performance across precision and recall, making it well-suited for agricultural prediction tasks where both false positives and false negatives can have significant implications.

**Milestone-6: Model Deployment**

**Activity-1: Save the Model**

Saving the best-performing model is a crucial step that allows for its reuse without the need for retraining every time the application is run.

● **Purpose:** To persist the trained machine learning model to disk. This is essential for deployment, as it allows the web application to load the model and make predictions efficiently. It also ensures that the exact model used for evaluation is the one deployed.

● **Method:** Python's pickle module is commonly used for serializing and deserializing Python objects. The best model (identified in Milestone 5, likely Gradient Boost Classifier) is saved to a .pkl file (e.g, model.pkl).



**Activity-2: Integrate with Web Framework**

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

This Section has the following tasks:

● Building HTML Pages

● Building a server-side script

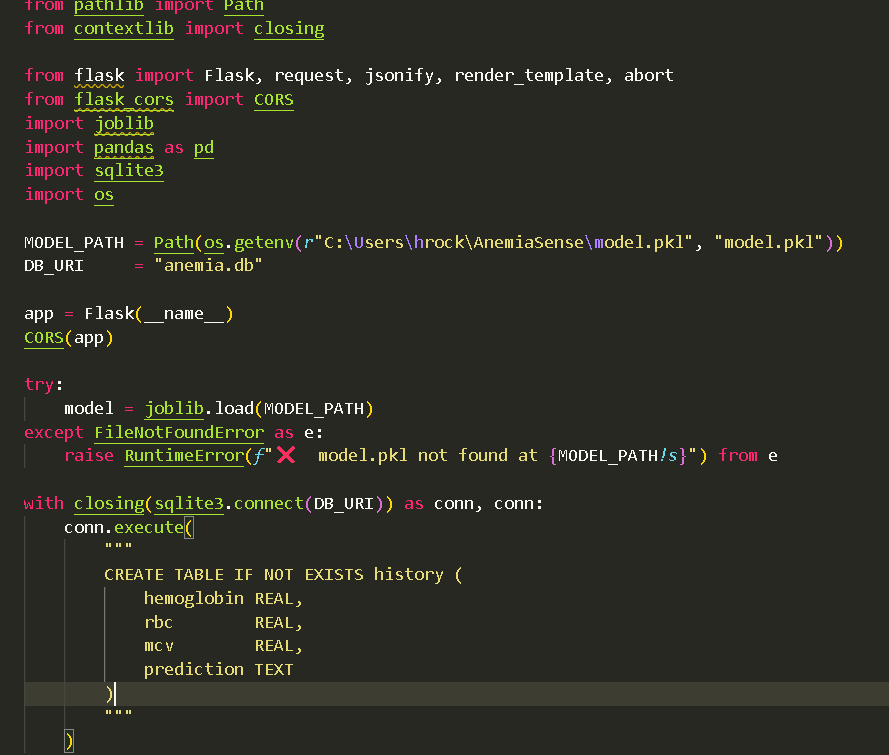
● Run a web application

**Activity 2.1: Building HTML Pages:**

* + Index.html
  + Predict.html

**Activity-2.2: Build Python code:**

Import the libraries



Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module (name) as an argument.



**Activity-3: Run the Web App**

This activity describes the simple steps required to launch the developed web application locally for testing and demonstration.

● **Open Anaconda Prompt:** Begin by opening the Anaconda Prompt (or your preferred command-line interface).

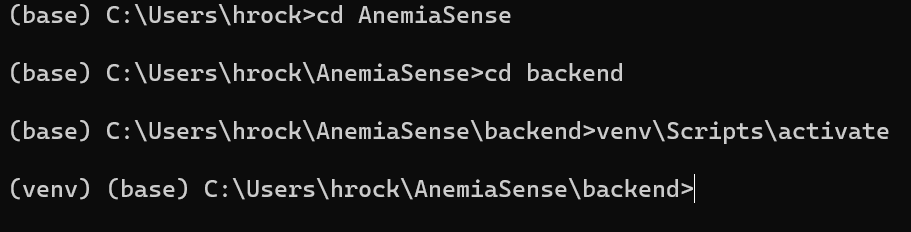
● **Navigate to Project Folder:** Use the cd command to navigate to the directory where your app.py script and other project files are located.

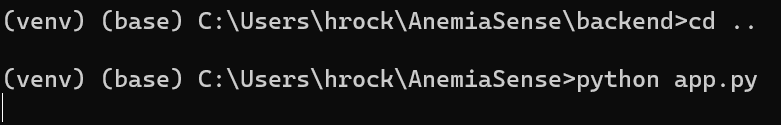
● **Run Flask Application:** Execute the command python app.py. This command starts the Flask development server.

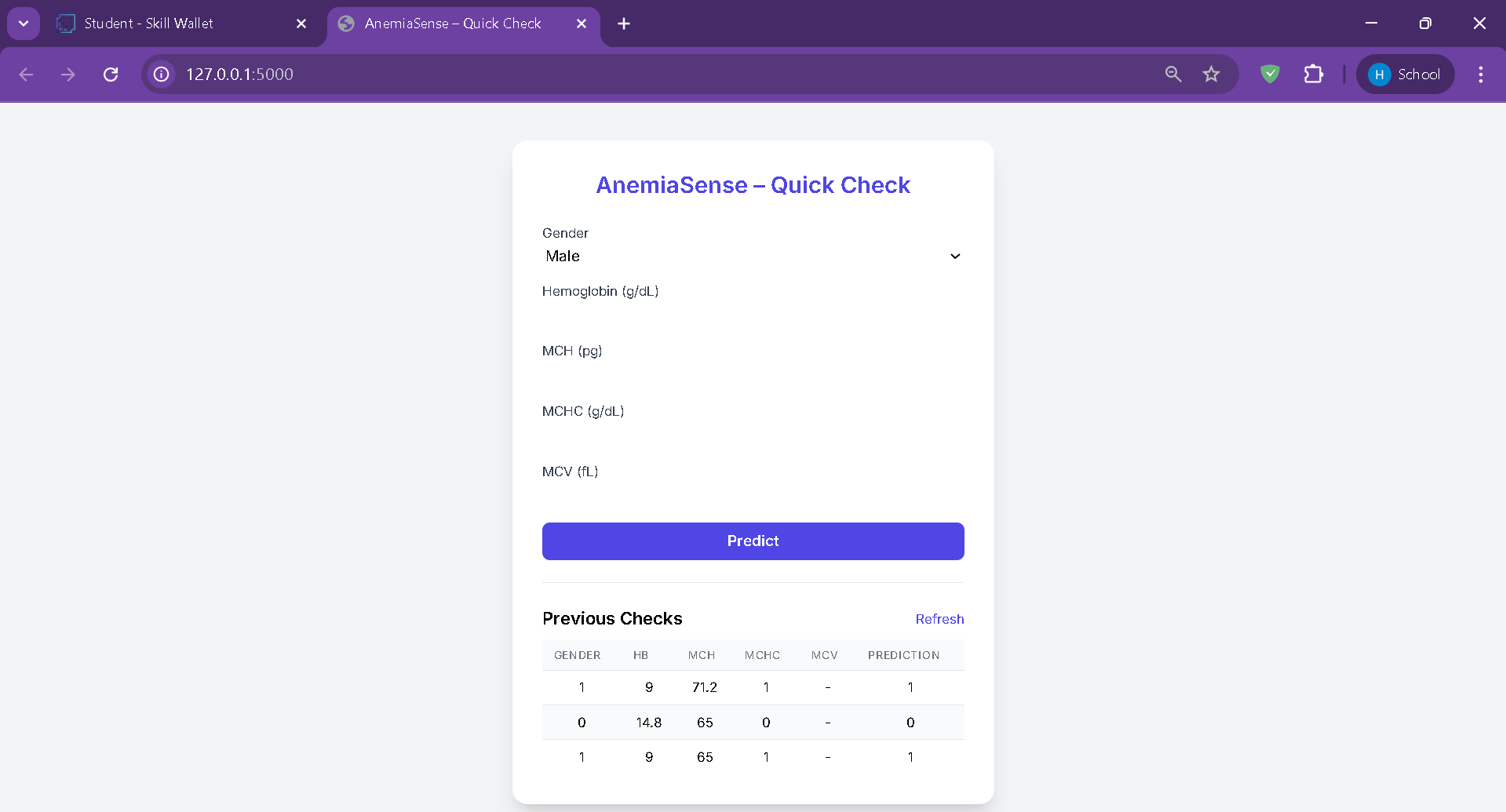
● **Access in Browser:** Once the server is running, open a web browser and navigate to the specified local URL, typically http://127.0.0.1:5000. This will load the web application's home page (index.html).

● **Interact and Test:** From the web page, users can input the required agricultural parameters, click the "Predict" or "Submit" button, and observe the model's recommendations or predictions displayed on the UI.

Activating virtual environment



Starting the program  




**Milestone-7: Project Demonstration & Documentation**

This final milestone focuses on presenting the completed project and providing comprehensive documentation for future reference and reproducibility.

**Activity-1: Record Video Demo**

Covers:

- UI walkthrough

- Sample input and prediction

- Explanation of ML workflow

**Activity-2: Step-by-step Documentation**

Comprehensive documentation is vital for the long-term viability, maintainability, and understanding of any software project.

● Purpose: This document (the one you are currently reading) serves as the official project report. It details every step of the project development, from problem definition to deployment.

● Content: The documentation includes:

○ Problem Understanding: Detailed explanation of the business problem, requirements, literature review, and social/business impact.

○ Data Aspects: Information on data collection, sources, and all data preparation steps (missing values, outliers, encoding, scaling, balancing, splitting).

○ Exploratory Data Analysis: Insights gained from descriptive statistics and visual analyses (univariate, bivariate, multivariate).

○ Model Development: Description of algorithms used, training procedures, and testing methodologies.

○ Performance Evaluation: Comparison of models and results from cross-validation.

○ **Deployment Details:** How the model is saved and integrated into the web framework, along with instructions for running the application.

○ **References:** A list of books and websites consulted during the project.

○ **Appendix:** Supplementary materials such as UI screenshots, code snippets, and the Jupyter Notebook for model training.

● **Format:** The document is structured logically with clear headings and subheadings, making it easy to navigate and understand. It adheres to a standard project report template to ensure consistency and professionalism.

**Conclusion**

Anemia Sense effectively demonstrates how ML can be leveraged in healthcare to improve diagnostic accuracy and patient care. With real-time prediction, personalization, and remote monitoring features, it enables early detection and efficient anemia management—particularly beneficial in low-resource settings.

**Appendix**

- UI Screenshots (form input & prediction)

- Flask code snippet (route, model load, predict)

- Jupyter Notebook: `anemia\_sense.ipynb`

**References:**

**Websites**

**1. https://www.kaggle.com: A popular platform for data science and machine learning competitions, providing access to numerous datasets and community-contributed notebooks.**

**2. https://scikit-learn.org: The official documentation for Scikit-learn, a widely used machine learning library in Python, offering guides, examples, and API references.**

**3. https://towardsdatascience.com: A publication featuring articles and tutorials on data science, machine learning, and AI, often providing practical insights and code examples.**