Anemia Sense: Leveraging Machine Learning for Precise Anemia Recognitions

Team ID: SWTID1720078683

Team Members:

1. **Dinesh. R**
2. **G. Achuth**
3. **Lakshmanan. L**
4. **Agash. JP**

1. Introduction

Date	10 July 2024
Team ID	SWTID1720078683
Project Name	Anemia Sense: Leveraging Machine Learning for Precise Anemia Recognitions

1.1. Project Overview

The Anemia Detection System is a cutting-edge health application designed to accurately predict the presence of anemia based on a user's blood report. Utilizing machine learning techniques, this system analyzes key blood parameters such as Hemoglobin, Mean Corpuscular Hemoglobin (MCH), Mean Corpuscular Hemoglobin Concentration (MCHC), and Mean Corpuscular Volume (MCV) to determine anemia status.

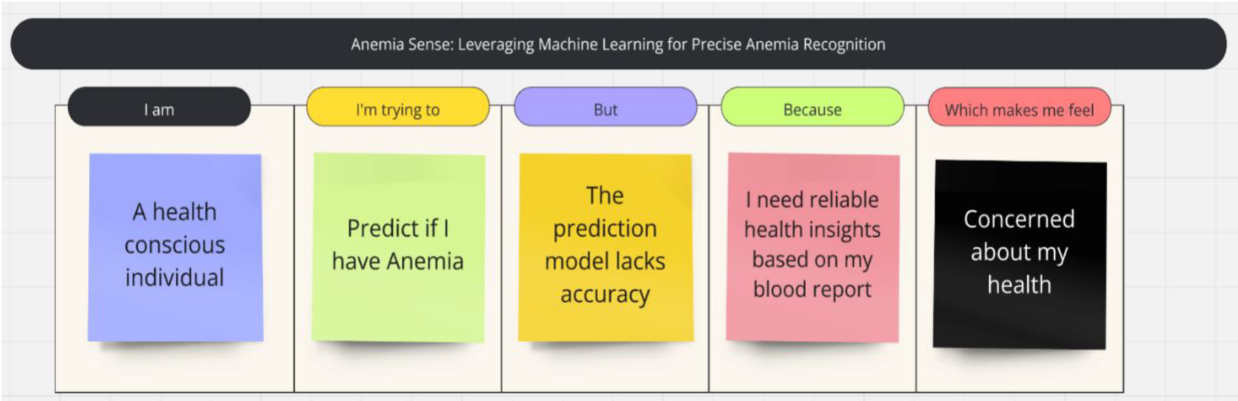
1.2. Objectives

The primary objective of this project is to provide a reliable, user-friendly tool for early anemia detection, enabling individuals to take proactive steps towards managing their health. The system employs advanced data preprocessing methods to ensure high-quality input data and leverages a Gradient Boosting model for robust performance in classification tasks. Comprehensive evaluation metrics, including accuracy, precision, recall, and confusion matrix, are used to validate the model's predictions.

2. Project Initialization and Planning Phase

2.1. Define Problem Statements:

Developing an anemia prediction system aimed at health-conscious individuals who seek to assess their health status based on detailed blood reports. The system must accurately classify the presence of anemia using key blood parameters such as Hemoglobin, MCH, MCHC, and MCV. This initiative addresses the need for reliable health insights, ensuring users can make informed decisions about their well-being promptly and effectively.



Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
Anemia Prediction	A health-conscious individual	Predict if I have anemia	The prediction model lacks accuracy	I need reliable health insights based on my blood report	Concerned about my health

2.2. Project Proposal (Proposed Solution)

This project proposal outlines a solution to address a specific problem. With a clear objective, defined scope, and a concise problem statement, the proposed solution details the approach, key features, and resource requirements, including hardware, software, and personnel.

Project Overview	
Objective	The objective of Anemia sense is to develop a machine learning-based system for the accurate detection and management of anemia. By
Scope	The Anemia sense project will focus on developing a machine learning system for accurate anemia detection and management. This includes
Problem Statement	
Description	Anemia, marked by a deficiency of red blood cells or hemoglobin, often goes undetected or is diagnosed late due to traditional, time-consuming
Impact	Solving the problem of timely and accurate anemia detection with Anemia sense will enable early diagnosis and prompt treatment, reducing health

Proposed Solution	
Approach	To detect the presence of anemia using patient data, we will develop a Gradient Boosting model utilizing features such as Gender, Hemoglobin
Key Features	Our approach includes thorough data preprocessing, emphasizing under sampling to handle class imbalance effectively. Critical features such as

Resource Requirements

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	Integrated GPUs
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	512 GB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	Matplotlib, Seaborn, Scikit-learn, pandas, NumPy
Development Environment	IDE, version control	Jupyter Notebook, Git
Data		
Data	Source, size, format	Smart Wallet Platform, 1421 rows of data, CSV file

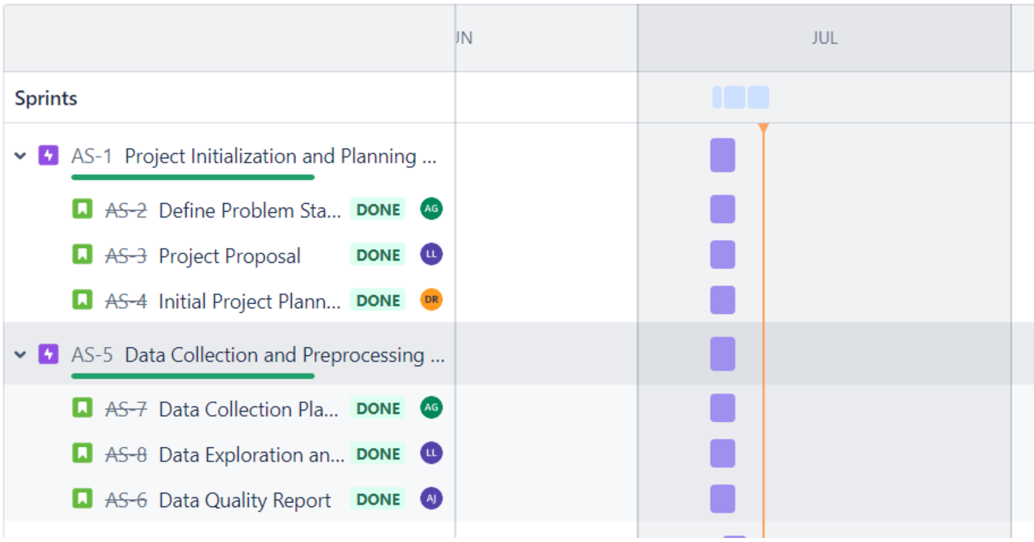
2.3. Initial Project Planning

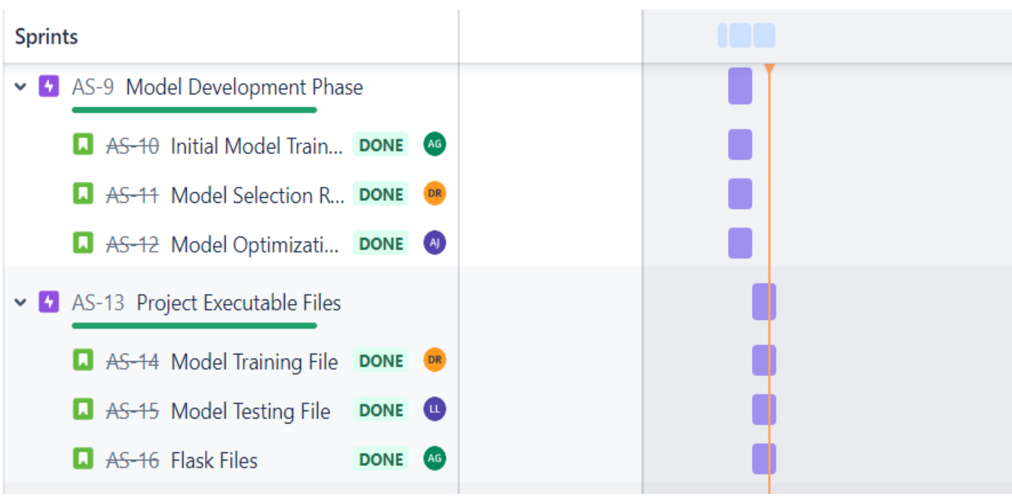
Product Backlog, Sprint Schedule, and Estimation (4 Marks)

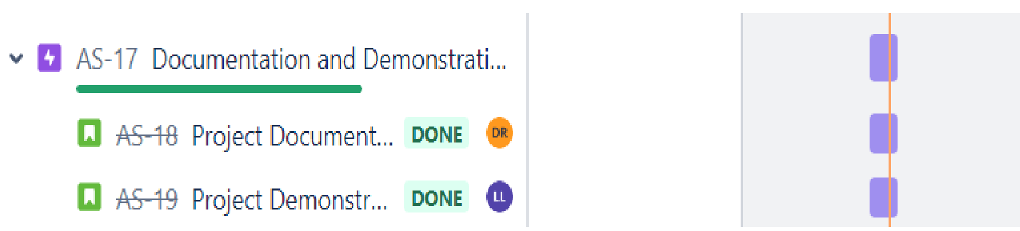
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Project Initialization and Planning Phase	AS-2	Define Problem Statements	3	High	G.Achuth	7-07-2024	8-07-2024
Sprint-1	Project Initialization and Planning Phase	AS-3	Project Proposal	2	Medium	Lakshmanan.L	7-07-2024	8-07-2024
Sprint-1	Project Initialization and Planning Phase	AS-4	Initial Project Planning Report	2	Medium	Dinesh .R	7-07-2024	8-07-2024
Sprint-1	Data Collection and Preprocessing Phase	AS-6	Data Quality Report	2	Medium	G.Achuth	7-07-2024	8-07-2024
Sprint-1	Data Collection and Preprocessi	AS-7	Data Collection Plan and Raw Data Sources Identification Report	2	Medium	Lakshmanan.L	7-07-2024	8-07-2024
Sprint-1	Data Collection and Preprocessi	AS-8	Data Exploration and Preprocessing Report	2	Medium	Agash.JP	7-07-2024	8-07-2024

Sprint-2	Model Development Phase	AS-10	Initial Model Training Code, Model Validation and	3	High	G.Achuth	8-07-2024	9-07-2024
Sprint-2	Model Development Phase	AS-11	Model Selection Report	3	High	Dinesh.R	8-07-2024	9-07-2024
Sprint-2	Model Development Phase	AS-12	Model Optimization and Tuning Report	3	High	Agash.JP	8-07-2024	9-07-2024
Sprint-3	Project Executable Files	AS-14	Model Training File	3	High	Dinesh.R	10-07-2024	11-07-2024
Sprint-3	Project Executable Files	AS-15	Model Testing File	3	High	Lakshmanan.L	10-07-2024	11-07-2024
Sprint-3	Project Executable Files	AS-16	Flask Files	2	Medium	G.Achuth	10-07-2024	11-07-2024
Sprint-3	Documentation and Demonstrat	AS-18	Project Documentation	3	High	Dinesh.R	10-07-2024	11-07-2024
Sprint-3	Documentation and Demonstrat	AS-19	Project Demonstration	2	Medium	Lakshmanan.L	10-07-2024	11-07-2024

Screenshots:







3. Data Collection and Preprocessing Phase

3.1. Data Collection Plan & Raw Data Sources Identification

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

Data Collection Plan

Section	Description
Project Overview	Anemia sense leverages machine learning algorithms to provide precise recognition and management of anemia, a condition characterized by a
Data Collection Plan	Skill Wallet Platform
Raw Data Sources Identified	File Name: anemia.csv File Size: 33.8 KB

Raw Data Sources

Source Name	Description	Location/ URL	Format	Size	Access Permissions
Dataset 1	The dataset contains 1,421 entries with 6 columns: Gender, Hemoglobin, MCH, MCHC, MCV, and Result, all with non-null values. It includes information on blood parameters and the presence or absence of anemia. Gender is likely encoded as 0 and 1, while Result indicates anemia status, with 0 for no anemia and 1 for anemia.	https://drive.google.com/file/d/1KMJFNFGwoaQoAouIPabMEHcT1bvqEXau/view?usp=sharing	CSV	33.8 KB	Public

3.2. Data Quality Report

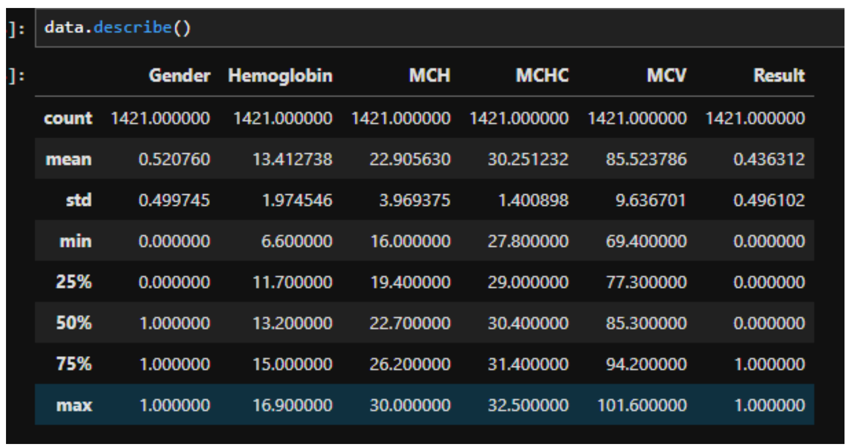
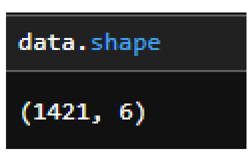
The Data Quality Report will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

Data Source	Data Quality Issue	Severity	Resolution Plan

https://drive.google.com/file/d/1KMJFNFGwoaQoAoulPabMEHcT1bvqEXau/view?usp=sharing	Data Imbalance in the Gender Column.	Low	Used under sampling technique to balance the dataset.

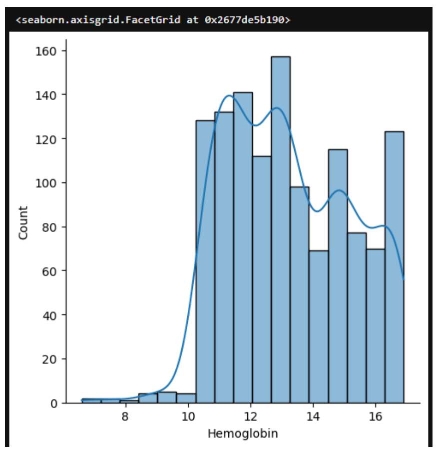
3.3. Data Exploration and Preprocessing

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

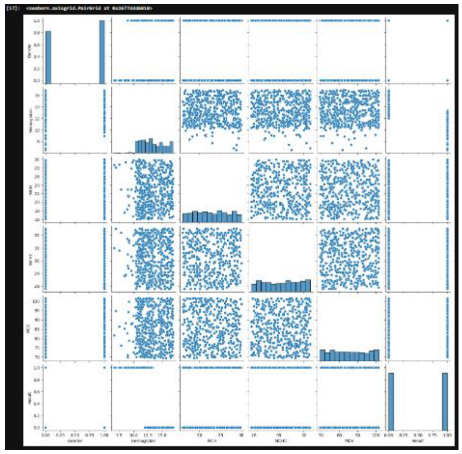
<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data Overview</td><td>

<pre>]: data.describe()</pre>
<table>
<tr><th></th><th>Gender</th><th>Hemoglobin</th><th>MCH</th><th>MCHC</th><th>MCV</th><th>Result</th></tr>
<tr><td>count</td><td>1421.000000</td><td>1421.000000</td><td>1421.000000</td><td>1421.000000</td><td>1421.000000</td><td>1421.000000</td></tr>
<tr><td>mean</td><td>0.520760</td><td>13.412738</td><td>22.905630</td><td>30.251232</td><td>85.523786</td><td>0.436312</td></tr>
<tr><td>std</td><td>0.499745</td><td>1.974546</td><td>3.969375</td><td>1.400898</td><td>9.636701</td><td>0.496102</td></tr>
<tr><td>min</td><td>0.000000</td><td>6.600000</td><td>16.000000</td><td>27.800000</td><td>69.400000</td><td>0.000000</td></tr>
<tr><td>25%</td><td>0.000000</td><td>11.700000</td><td>19.400000</td><td>29.000000</td><td>77.300000</td><td>0.000000</td></tr>
<tr><td>50%</td><td>1.000000</td><td>13.200000</td><td>22.700000</td><td>30.400000</td><td>85.300000</td><td>0.000000</td></tr>
<tr><td>75%</td><td>1.000000</td><td>15.000000</td><td>26.200000</td><td>31.400000</td><td>94.200000</td><td>1.000000</td></tr>
<tr><td>max</td><td>1.000000</td><td>16.900000</td><td>30.000000</td><td>32.500000</td><td>101.600000</td><td>1.000000</td></tr>
</table>

<pre>data.shape

(1421, 6)</pre>
</td></tr>
</table>

Univariate Analysis



Bivariate Analysis

Multivariate Analysis	`sns.pairplot(data)`  `[18]: sns.heatmap(data.corr(),annot = True,cmap = 'RdYlGn',linewidths = 0.2)` `figure = plt.gcf()` `figure.set_size_inches(10,8)` `plt.show()` 
Data Preprocessing Code Screenshots	
Loading Data	`data = pd.read_csv('anemia.csv')`
Handling Missing Data	`data.isnull().any()` `data.isnull().sum()`

<table>
<tr>
<td>Data Transformation</td>
<td><pre><code>from sklearn.utils import resample

major = data[data['Result'] == 0]
minor = data[data['Result'] == 1]
undersampling = resample(major,replace = False,n_samples = len(minor),random_state = 47)
data = pd.concat([undersampling,minor])
print(data['Result'].value_counts())

Result
0   620
1   620
Name: count, dtype: int64</code></pre></td>
</tr>
<tr>
<td>Save Processed Data</td>
<td><pre><code>data.to_csv('anemia.csv',index=False)</code></pre></td>
</tr>
</table>

4. Model Development Phase

4.1. Feature Selection Report

In the forthcoming update, each feature will be accompanied by a brief description. Users will indicate whether it's selected or not, providing reasoning for their decision. This process will streamline decision-making and enhance transparency in feature selection.

Feature	**Description**	**Selected (Yes/No)**	**Reasoning**
Gender	Binary indicator of gender (0: Male, 1: Female)	Yes	Relevant for potential gender differences in anemia
Hemoglobin	Hemoglobin level	Yes	Primary indicator of anemia

MCH	Mean Corpuscular Hemoglobin is a measure of the average amount of hemoglobin per red blood cell	Yes	Indicator for red blood cell characteristics

MCHC	Mean Corpuscular Hemoglobin Concentration indicates the concentration of hemoglobin in a given volume of packed red blood cells	Yes	Indicator for red blood cell concentration
MCV	Mean Corpuscular Volume measures the average volume of red blood cells	Yes	Indicator for red blood cell volume

4.2. Model Selection Report

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model Selection Report:

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
Logistic Regression	Logistic regression is a statistical method for binary classification that models the probability of a binary outcome using a logistic function to constrain the output between 0 and 1.	-	Accuracy – 0.9798
Random Forest Classifier	Random Forest is an ensemble learning method that builds multiple decision trees and merges their results to improve accuracy and control over-fitting.	-	Accuracy – 1.00
Decision Tree Classifier	A decision tree is a flowchart-like structure where each internal node represents a decision based on a feature, each branch represents the outcome of the decision, and each leaf node represents a class label.	-	Accuracy – 1.00
Gaussian Naïve Bayes	Gaussian NB is a variant of the Naive Bayes classifier that assumes the features follow a Gaussian (normal) distribution, used for probabilistic classification.	-	Accuracy – 0.9516
Support Vector Machine	SVM is a supervised learning model that finds the optimal hyperplane which maximizes the margin between different classes in the feature space.	-	Accuracy – 0.9032

Gradient Boosting Classifier	Gradient Boosting is an ensemble technique that builds models sequentially, with each new model attempting to correct the errors of the previous models,	-	Accuracy – 1.00

4.3. Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

```
log = LogisticRegression()
```

```
log.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
rf = RandomForestClassifier()
```

```
rf.fit(x_train,y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
dec = DecisionTreeClassifier()
```

```
dec.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
NB = GaussianNB()
```

```
NB.fit(x_train,y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
SVM = SVC()
```

```
SVM.fit(x_train,y_train)
```

```
▾ SVC
SVC()
```

```
GB = GradientBoostingClassifier()
```

```
GB.fit(x_train,y_train)
```

```
▾ GradientBoostingClassifier
GradientBoostingClassifier()
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Logistic Regression	acc_lr = accuracy_score(y_test,y_predict) acc_lr 0.9798387096774194 rep_lr = classification_report(y_test,y_predict) print(rep_lr) precision recall f1-score support 0 0.99 0.97 0.98 123 1 0.97 0.99 0.98 125 accuracy 0.98 248 macro avg 0.98 0.98 0.98 248 weighted avg 0.98 0.98 0.98 248	0.9798	confusion_matrix(y_test,y_predict) array([[119, 4], [1, 124]], dtype=int64)
Random Forest Classifier	acc_rf = accuracy_score(y_test,y_predict) acc_rf 1.0 rep_rf = classification_report(y_test,y_predict) print(rep_rf) precision recall f1-score support 0 1.00 1.00 1.00 123 1 1.00 1.00 1.00 125 accuracy 1.00 248 macro avg 1.00 1.00 1.00 248 weighted avg 1.00 1.00 1.00 248	1.00	confusion_matrix(y_test,y_predict) array([[123, 0], [0, 125]], dtype=int64)
Decision Tree Classifier	acc_dc = accuracy_score(y_test,y_predict) acc_dc 1.0 rep_dc = classification_report(y_test,y_predict) print(rep_dc) precision recall f1-score support 0 1.00 1.00 1.00 123 1 1.00 1.00 1.00 125 accuracy 1.00 248 macro avg 1.00 1.00 1.00 248 weighted avg 1.00 1.00 1.00 248	1.00	confusion_matrix(y_test,y_predict) array([[123, 0], [0, 125]], dtype=int64)

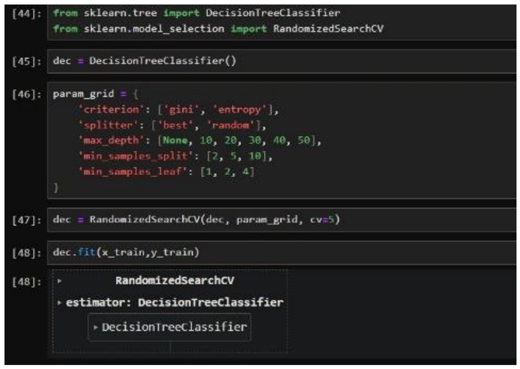
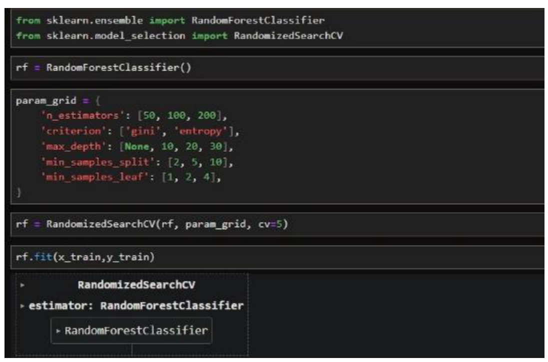
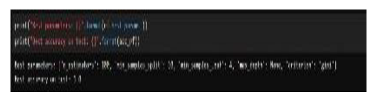
Gaussian Naïve Bayes	acc_NB = accuracy_score(y_test,y_predict) acc_NB 0.9516129032258065 rep_NB = classification_report(y_test,y_predict) print(rep_NB) precision recall f1-score support 0 0.97 0.93 0.95 123 1 0.93 0.98 0.95 125 accuracy 0.95 248 macro avg 0.95 0.95 0.95 248 weighted avg 0.95 0.95 0.95 248	0.9516	confusion_matrix(y_test,y_predict) array([[113, 10], [2, 123]], dtype=int64)
Support Vector Machine	acc_svm = accuracy_score(y_test,y_predict) acc_svm 0.9032258064516129 rep_svm = classification_report(y_test,y_predict) print(rep_svm) precision recall f1-score support 0 0.98 0.82 0.89 123 1 0.85 0.98 0.91 125 accuracy 0.90 248 macro avg 0.91 0.90 0.90 248 weighted avg 0.91 0.90 0.90 248	0.9032	confusion_matrix(y_test,y_predict) array([[101, 22], [2, 123]], dtype=int64)
Gradient Boosting Classifier	acc_GB = accuracy_score(y_test,y_predict) acc_GB 1.0 rep_GB = classification_report(y_test,y_predict) print(rep_GB) precision recall f1-score support 0 1.00 1.00 1.00 123 1 1.00 1.00 1.00 125 accuracy 1.00 248 macro avg 1.00 1.00 1.00 248 weighted avg 1.00 1.00 1.00 248	1.00	confusion_matrix(y_test,y_predict) array([[119, 4], [1, 124]], dtype=int64)

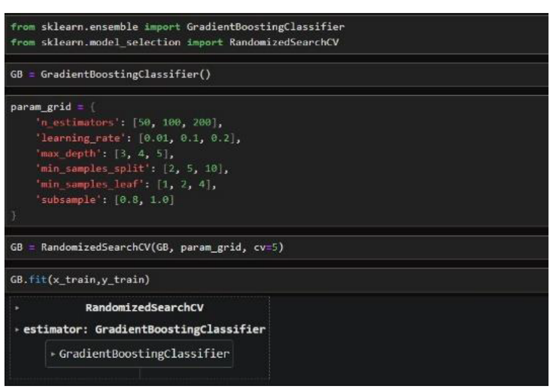
Out of all the 6 above mentioned models, we selected the Gradient Boosting Classifier Model for our project, due to the high accuracy that we got.

5. Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learningmodels for peakperformance. It includes optimizedmodel code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

5.1. Hyperparameter Tuning Documentation:

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th><th>Optimal Values</th></tr>
<tr><td>Decision Tree</td><td>
[44]: from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV
[45]: dec = DecisionTreeClassifier()
[46]: param_grid = {
'criterion': ['gini', 'entropy'],
'splitter': ['best', 'random'],
'max_depth': [None, 10, 20, 30, 40, 50],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4]
}
[47]: dec = RandomizedSearchCV(dec, param_grid, cv=5)
[48]: dec.fit(x_train,y_train)
[48]: RandomizedSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier</td><td>
print("Best parameters: {}".format(dec.best_params_))
print("Best accuracy on test: {}".format(acc_dc))
Best parameters: {'splitter': 'best', 'min_samples_split': 10, 'min_samples_leaf': 4, 'max_depth': 30, 'criterion': 'gini'}
Best accuracy on test: 1.0</td></tr>
<tr><td>Random Forest</td><td>
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
rf = RandomForestClassifier()
param_grid = {
'n_estimators': [50, 100, 200],
'criterion': ['gini', 'entropy'],
'max_depth': [None, 10, 20, 30],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4],
}
rf = RandomizedSearchCV(rf, param_grid, cv=5)
rf.fit(x_train,y_train)
RandomizedSearchCV
estimator: RandomForestClassifier
RandomForestClassifier</td><td>
print("Best parameters: {}".format(rf.best_params_))
print("Best accuracy on test: {}".format(acc_rf))
Best parameters: {'n_estimators': 100, 'min_samples_split': 10, 'min_samples_leaf': 4, 'max_depth': None, 'criterion': 'gini'}
Best accuracy on test: 1.0</td></tr>
</table>

<table>
<tr><td>Gradient
Boosting</td><td>
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import RandomizedSearchCV
GB = GradientBoostingClassifier()
param_grid = {
'n_estimators': [50, 100, 200],
'learning_rate': [0.01, 0.1, 0.2],
'max_depth': [3, 4, 5],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4],
'subsample': [0.8, 1.0]
}
GB = RandomizedSearchCV(GB, param_grid, cv=5)
GB.fit(x_train,y_train)
RandomizedSearchCV
estimator: GradientBoostingClassifier
GradientBoostingClassifier</td><td>
print("Best parameters: {}".format(GB.best_params_))
print("Best accuracy on test: {}".format(acc_GB))
Best parameters: {'subsample': 0.8, 'n_estimators': 100, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_depth': 4, 'learning_rate': 0.01}
Best accuracy on test: 1.0</td></tr>
</table>

5.2. Performance Metrics Comparison Report:

<table>
<tr><th>Model</th><th>Optimized Metric</th></tr>
<tr><td>Decision Tree</td><td><pre>
rep_dc = classification_report(y_test,y_predict)
print(rep_dc)

              precision   recall f1-score   support

           0   1.00   1.00   1.00   123
           1   1.00   1.00   1.00   125

    accuracy   1.00   248
   macro avg   1.00   1.00   1.00   248
weighted avg   1.00   1.00   1.00   248

confusion_matrix(y_test,y_predict)

array([[123,   0],
       [0, 125]], dtype=int64)
</pre></td></tr>
<tr><td>Random Forest</td><td><pre>
rep_rf = classification_report(y_test,y_predict)
print(rep_rf)

              precision   recall f1-score   support

           0   1.00   1.00   1.00   123
           1   1.00   1.00   1.00   125

    accuracy   1.00   248
   macro avg   1.00   1.00   1.00   248
weighted avg   1.00   1.00   1.00   248

confusion_matrix(y_test,y_predict)

array([[123,   0],
       [0, 125]], dtype=int64)
</pre></td></tr>
<tr><td>Gradient Boosting</td><td><pre>
rep_GB = classification_report(y_test,y_predict)
print(rep_GB)

              precision   recall f1-score   support

           0   1.00   1.00   1.00   123
           1   1.00   1.00   1.00   125

    accuracy   1.00   248
   macro avg   1.00   1.00   1.00   248
weighted avg   1.00   1.00   1.00   248

confusion_matrix(y_test,y_predict)

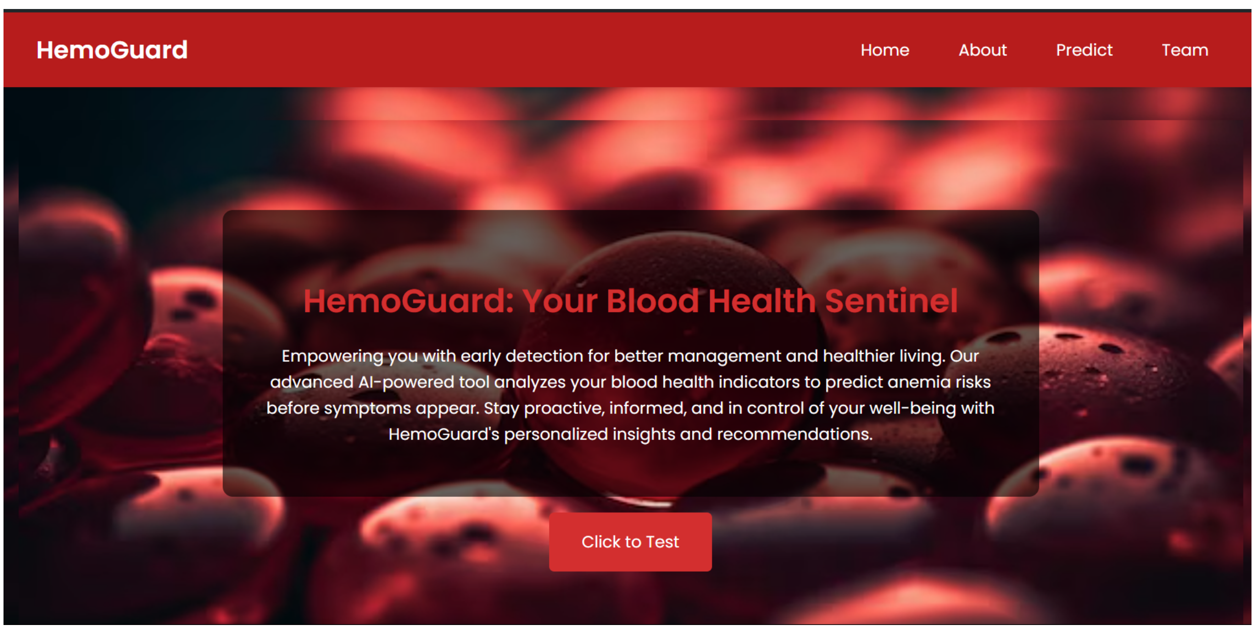
array([[123,   0],
       [0, 125]], dtype=int64)
</pre></td></tr>
</table>

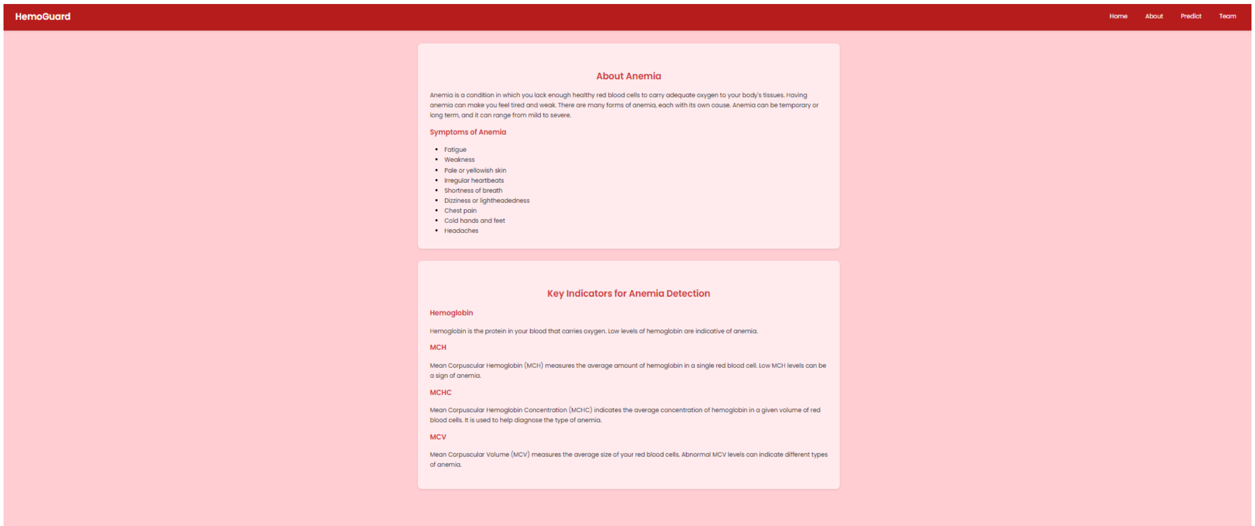
5.3. Final Model SelectionJustification:

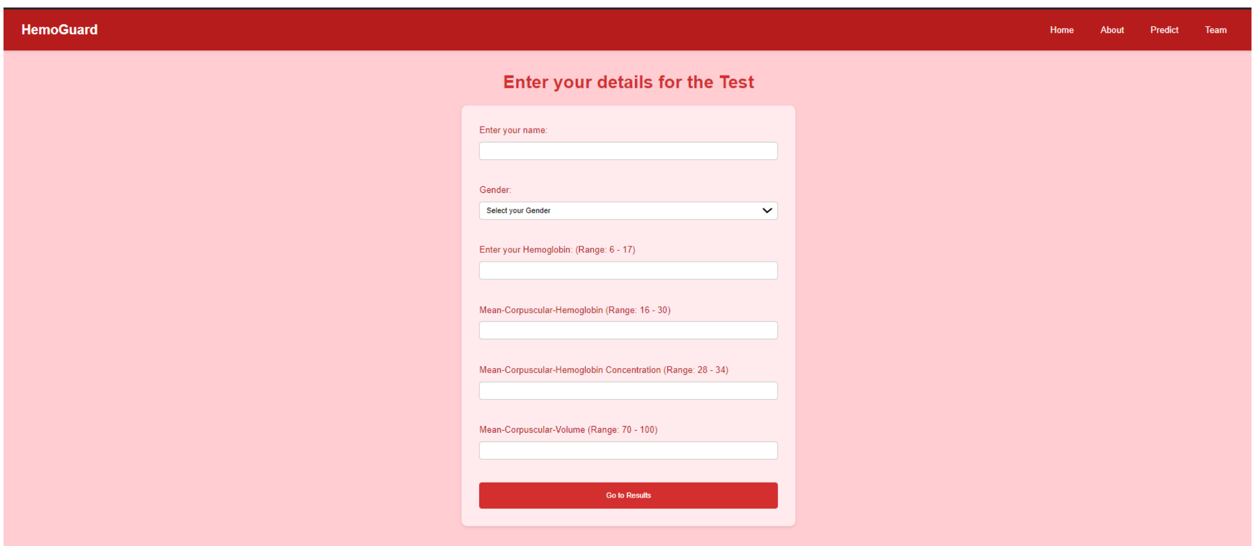
Final Model	Reasoning
Gradient Boosting	The Gradient Boosting model was selectedfor its superiorperformance, exhibiting high accuracy duringhyperparameter tuning. Its ability to handle complexrelationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifyingits selection as the final model

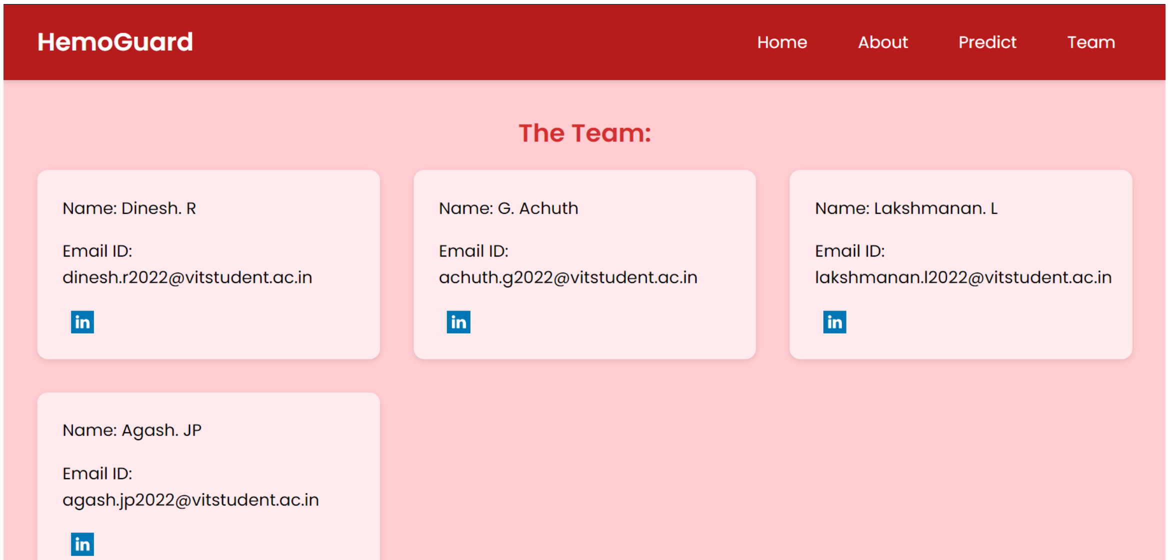
6. Results

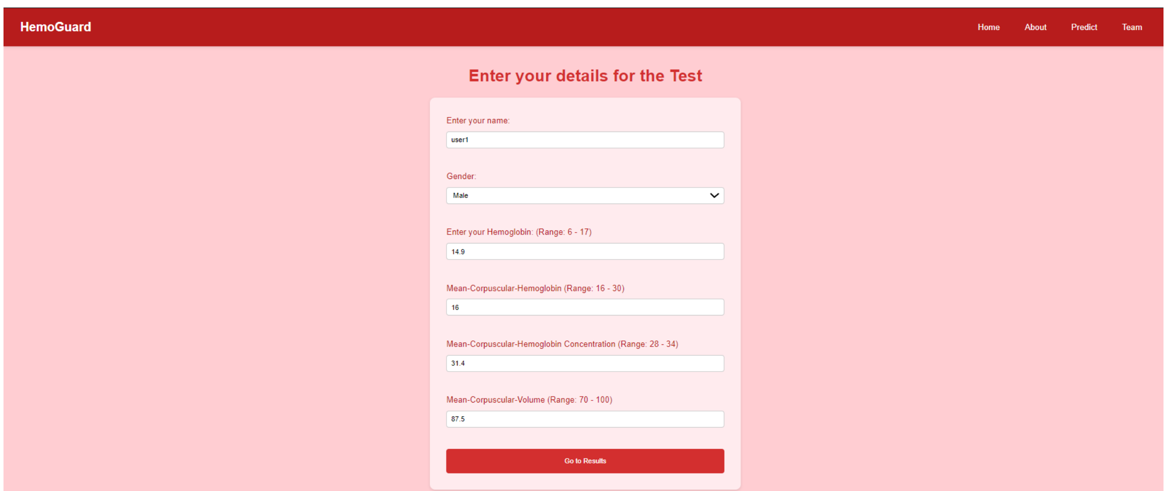
6.1. Output Screenshots

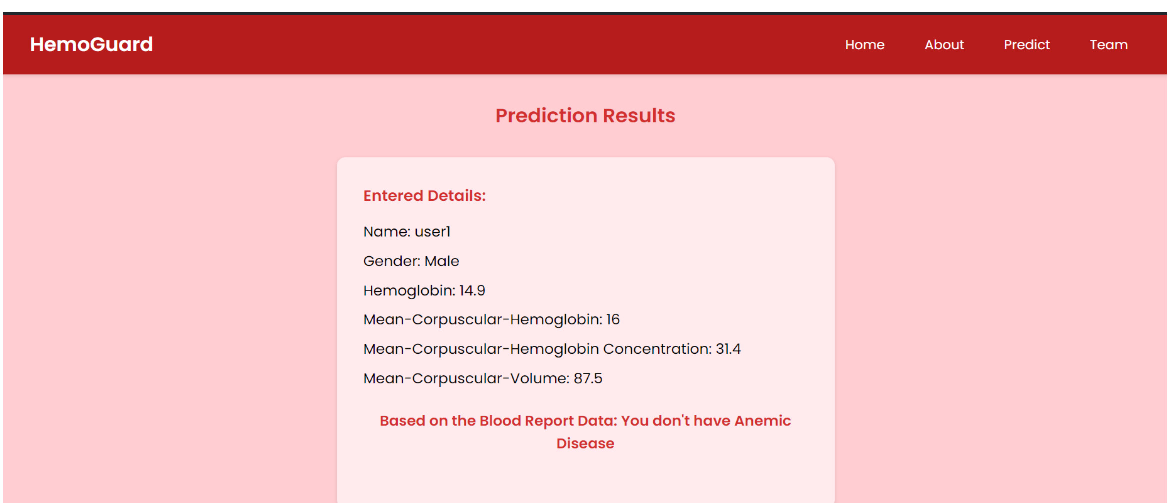



HemoGuard
Home
About
Predict
Team
About Anemia
Anemia is a condition in which you lack enough healthy red blood cells to carry adequate oxygen to your body's tissues. Having anemia can make you feel tired and weak. There are many forms of anemia, each with its own cause. Anemia can be temporary or long term, and it can range from mild to severe.
Symptoms of Anemia
Fatigue
Weakness
Pale or yellowish skin
Irregular heartbeats
Shortness of breath
Dizziness or lightheadedness
Chest pain
Cold hands and feet
Headaches
Key Indicators for Anemia Detection
Hemoglobin
Hemoglobin is the protein in your blood that carries oxygen. Low levels of hemoglobin are indicative of anemia.
MCH
Mean Corpuscular Hemoglobin (MCH) measures the average amount of hemoglobin in a single red blood cell. Low MCH levels can be a sign of anemia.
MCHC
Mean Corpuscular Hemoglobin Concentration (MCHC) indicates the average concentration of hemoglobin in a given volume of red blood cells. It is used to help diagnose the type of anemia.
MCV
Mean Corpuscular Volume (MCV) measures the average size of your red blood cells. Abnormal MCV levels can indicate different types of anemia.


HemoGuard
Home
About
Predict
Team
Enter your details for the Test
Enter your name:
Gender:
Select your Gender
Enter your Hemoglobin: (Range: 6 - 17)
Mean-Corpuscular-Hemoglobin (Range: 16 - 30)
Mean-Corpuscular-Hemoglobin Concentration (Range: 28 - 34)
Mean-Corpuscular-Volume (Range: 70 - 100)
Go to Results


HemoGuard
Home
About
Predict
Team
The Team:
Name: Dinesh. R
Email ID:
dinesh.r2022@vitstudent.ac.in
Name: G. Achuth
Email ID:
achuth.g2022@vitstudent.ac.in
Name: Lakshmanan. L
Email ID:
lakshmanan.l2022@vitstudent.ac.in
Name: Agash. JP
Email ID:
agash.jp2022@vitstudent.ac.in


HemoGuard
Home
About
Predict
Team
Enter your details for the Test
Enter your name:
user1
Gender:
Male
Enter your Hemoglobin: (Range: 6 - 17)
14.9
Mean-Corpuscular-Hemoglobin (Range: 16 - 30)
16
Mean-Corpuscular-Hemoglobin Concentration (Range: 28 - 34)
31.4
Mean-Corpuscular-Volume (Range: 70 - 100)
87.5
Go to Results


HemoGuard
Home
About
Predict
Team
Prediction Results
Entered Details:
Name: user1
Gender: Male
Hemoglobin: 14.9
Mean-Corpuscular-Hemoglobin: 16
Mean-Corpuscular-Hemoglobin Concentration: 31.4
Mean-Corpuscular-Volume: 87.5
Based on the Blood Report Data: You don't have Anemic Disease

7. Advantages & Disadvantages

Advantages

1. **Early Detection**:
   - Enables timely medical intervention by identifying anemia early.
2. **Accuracy**:
   - Utilizes a robust Gradient Boosting model for reliable predictions.
3. **User-Friendly Interface**:
   - Designed for ease of use, making it accessible to a wide range of users.
4. **Real-Time Predictions**:
   - Provides instant results, allowing for quick understanding and action.

Disadvantages

1. **Dependence on Data Quality**:
   - Accurate predictions rely on high-quality input blood report data.
2. **Data Privacy Concerns**:
   - Handling sensitive health data requires stringent security measures.
3. **Not a Substitute for Medical Advice**:
   - Provides valuable insights but cannot replace professional medical diagnosis and consultation.

8. Conclusion

The Anemia Detection application leverages machine learning to offer a reliable method for early anemia diagnosis. Utilizing features like Gender, Hemoglobin, MCH, MCHC, and MCV, and a Gradient Boosting model, the application ensures accurate predictions. This tool aids in early detection, facilitating timely medical intervention and improving health outcomes. While it is not a substitute for professional medical advice, the application provides valuable preliminary insights, making it a beneficial addition to healthcare technology.

9. Future Scope

The future scope of the Anemia Detection application includes:

1. **Enhanced Model Accuracy**: Incorporate advanced machine learning algorithms and larger datasets to improve prediction accuracy.
2. **Mobile Integration**: Develop mobile applications for iOS and Android to increase

accessibility.

3. **Integration with Wearable Devices**: Enable integration with wearable health devices for real-time monitoring.
4. **Comprehensive Health Analysis**: Expand the application to include predictions for other related health conditions

10. Appendix

10.1. Source Code

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('anemia.csv')
```

```
data.head()
```

```
data.isnull().any()
```

```
data.info()
```

```
data.shape
```

```
data.isnull().sum()
```

```
data.describe()
```

```
data.isnull().sum()
```

```
results = data['Result'].value_counts()
results.plot(kind = 'bar',color = ['red','green'])
plt.xlabel('Results')
plt.ylabel('Frequency')
plt.title('Imbalance data analysis')
```

```
from sklearn.utils import resample
```

```
major = data[data['Result'] == 0]
minor = data[data['Result'] == 1]
undersampling = resample(major,replace = False,n_samples = len(minor),random_state = 47)
data = pd.concat([undersampling,minor])
print(data['Result'].value_counts())
```

```
res_balanced = data['Result'].value_counts()
res_balanced.plot(kind = 'bar',color = ['red','green'])
plt.xlabel('Results')
plt.ylabel('Frequency')
plt.title('Imbalance data analysis(Balanced)')
```

```
gender = data['Gender'].value_counts()
gender.plot(kind = 'bar',color = ['blue','orange'])
plt.xlabel('Gender')
plt.ylabel('Frequency')
plt.title('Gender analysis')
```

```
sns.displot(data['Hemoglobin'],kde = True)
```

```
df = pd.DataFrame(data)


df['Result'] = df['Result'].map({0: '0', 1: '1'})


mean_hemoglobin = df.groupby(['Gender', 'Result'])['Hemoglobin'].mean().reset_index()


plt.figure(figsize=(10, 6))
plot = sns.barplot(x='Gender', y='Hemoglobin', hue='Result', data=mean_hemoglobin)


plt.title('Mean Hemoglobin by Gender and Result')
plt.xlabel('[male,female]')
plt.ylabel('Hemoglobin')

for i in plot.containers:
    plot.bar_label(i,fmt = '%.4f',label_type = 'edge')

plt.ylim(0,plt.ylim()[1]*1.1)

plt.show()
```

```
sns.pairplot(data)
```

```
sns.heatmap(data.corr(),annot = True,cmap = 'RdYlGn',linewidths = 0.2)
figure = plt.gcf()
figure.set_size_inches(10,8)
plt.show()
```

```
y = data['Result']
x = data.drop('Result',axis = 1)
```

```
x.head()
```

```
y.head()
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state = 234)
```

```
x_train.shape
```

```
x_test.shape
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

```
log = LogisticRegression()
```

```
log.fit(x_train,y_train)
```

```
y_predict = log.predict(x_test)
```

```
acc_lr = accuracy_score(y_test,y_predict)
```

```
acc_lr
```

```
rep_lr = classification_report(y_test,y_predict)
```

```
print(rep_lr)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
```

```
rf = RandomForestClassifier()
```

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}
```

```python
rf = RandomizedSearchCV(rf, param_grid, cv=5)
```

```python
rf.fit(x_train,y_train)
```

```python
y_predict = rf.predict(x_test)
```

```python
acc_rf = accuracy_score(y_test,y_predict)
acc_rf
```

```python
print("Best parameters: {}".format(rf.best_params_))
print("Best accuracy on test: {}".format(acc_rf))
```

```python
rep_rf = classification_report(y_test,y_predict)
print(rep_rf)
```

```python
confusion_matrix(y_test,y_predict)
```

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV
```

```python
dec = DecisionTreeClassifier()
```

```python
param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```python
dec = RandomizedSearchCV(dec, param_grid, cv=5)
```

```python
dec.fit(x_train,y_train)
```

```python
y_predict = dec.predict(x_test)
```

```python
acc_dc = accuracy_score(y_test,y_predict)
acc_dc
```

```python
print("Best parameters: {}".format(dec.best_params_))
print("Best accuracy on test: {}".format(acc_dc))
```

```python
rep_dc = classification_report(y_test,y_predict)
print(rep_dc)
```

```python
confusion_matrix(y_test,y_predict)
```

```python
from sklearn.naive_bayes import GaussianNB
```

```python
NB = GaussianNB()
```

```python
NB.fit(x_train,y_train)
```

```python
y_predict = NB.predict(x_test)
```

```python
acc_NB = accuracy_score(y_test,y_predict)
acc_NB
```

```python
rep_NB = classification_report(y_test,y_predict)
print(rep_NB)
```

```python
from sklearn.svm import SVC
```

```python
SVM = SVC()
```

```python
SVM.fit(x_train,y_train)
```

```python
y_predict = SVM.predict(x_test)
```

```python
acc_svm = accuracy_score(y_test,y_predict)
acc_svm
```

```python
rep_svm = classification_report(y_test,y_predict)
print(rep_svm)
```

```python
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import RandomizedSearchCV
```

```python
GB = GradientBoostingClassifier()
```

```python
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'subsample': [0.8, 1.0]
}
```

```python
GB = RandomizedSearchCV(GB, param_grid, cv=5)
```

```python
GB.fit(x_train,y_train)
```

```python
y_predict = GB.predict(x_test)
```

```python
acc_GB = accuracy_score(y_test,y_predict)
acc_GB
```

```
rep_GB = classification_report(y_test,y_predict)
print(rep_GB)
```

```
confusion_matrix(y_test,y_predict)
```

```
print("Best parameters: {}".format(GB.best_params_))
print("Best accuracy on test: {}".format(acc_GB))
```

```
model = pd.DataFrame({'Model' : ['Logistic Regression','Random Forest Classifier',
```

```
model
```

```
import pickle
with open('model.pkl','wb') as f:
    pickle.dump(GB,f)
```

```
with open('model.pkl','rb') as f:
    model = pickle.load(f)
```

```
from flask import Flask, render_template, request
import pickle
import numpy as np

model = pickle.load(open("model.pkl","rb"))
app = Flask(__name__)

@app.route('/')

def home():
    return render_template('home.html')

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

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

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

```python
@app.route('/results', methods = ["POST"])
def prediction():

    Name = request.form["name"]
    Hemo = request.form["hb"]
    Gender = request.form["gender"]
    MCH = request.form["mch"]
    MCHC = request.form["mchc"]
    MCV = request.form["mcv"]

    if Gender == "Male":
        g = 0
    else:
        g = 1
    x_test = [[g,float(Hemo),float(MCH),float(MCHC),float(MCV)]]
    print(x_test)

    p = np.array(x_test)
    p = p.astype(np.float32)

    prediction = model.predict(p)


    if (prediction == 0):
        text = "You don't have Anemic Disease"
    else:
        text = "You have Anemic Disease"


    return render_template("results.html",f = Name, e = Gender, a = Hemo, b = MCH, c = MCHC, d = MCV, predicted_r


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

10.2 GitHub and Project Demo Link

- **GitHub Repo Link** - https://github.com/Achuth-0908/Anemia-Sense-Leveraging-Machine-Learning-for-Precise-Anemia-Recognitions.git

- **Demonstration Video Link** - https://drive.google.com/file/d/1nwClhjTxJoWCycUFWg3kcRugEpSMFOv8/view