Anemia

Project Report

Anemia Sense: Leveraging Machine Learning For Precise Anemia Recognitions

Prepared by

Vinay Kumar

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Anemia is a widespread hematological condition characterized by a deficiency in the number or quality of red blood cells or hemoglobin, resulting in reduced oxygen transport to the body's tissues. It affects over 1.6 billion people globally, particularly impacting women and children in developing regions. Early and accurate detection is crucial for preventing serious complications, such as fatigue, cognitive impairment, and even heart failure. However, traditional diagnostic methods like complete blood count (CBC) tests, though effective, require access to laboratories, skilled personnel, and infrastructure, which are often lacking in rural and under-resourced areas.

With the advancement of artificial intelligence and data science, machine learning (ML) has emerged as a powerful tool in the healthcare sector. “Anemia Sense” is an innovative approach that leverages machine learning algorithms to enhance the accuracy, speed, and accessibility of anemia detection. By analyzing patterns in medical datasets — including patient demographics, physiological parameters, and laboratory values — ML models can identify subtle indicators of anemia that may not be immediately obvious to human clinicians. These models not only aid in early detection but also help in classifying different types of anemia, such as iron-deficiency anemia, vitamin B12 deficiency anemia, or anemia of chronic disease.

The core advantage of integrating machine learning into anemia detection lies in its scalability and adaptability. Once trained, these models can be deployed in mobile applications or clinical decision support systems, making diagnostic tools more accessible in remote or underserved areas. Additionally, the predictive capability of ML can support physicians by providing real-time risk assessments and personalized treatment recommendations.

In this project, we aim to develop and evaluate machine learning models that can accurately detect anemia using minimal input features. The proposed system, “Anemia Sense,” is not only a step towards automation in healthcare diagnostics but also a contribution to reducing the global burden of anemia through intelligent, data-driven solutions. By bridging the gap between clinical needs and technological innovation, this work highlights the transformative potential of machine learning in modern medicine.

1.1 Project Overview:

AnemiaSense is an innovative healthcare technology initiative focused on the precise detection of anemia using advanced machine learning algorithms. The primary goal of the project is to enhance the accuracy and efficiency of anemia diagnosis, facilitating early intervention and better patient management. By leveraging extensive datasets of blood parameters and patient profiles, AnemiaSense aims to identify early signs of anemia and flag potential cases for further investigation by healthcare professionals.

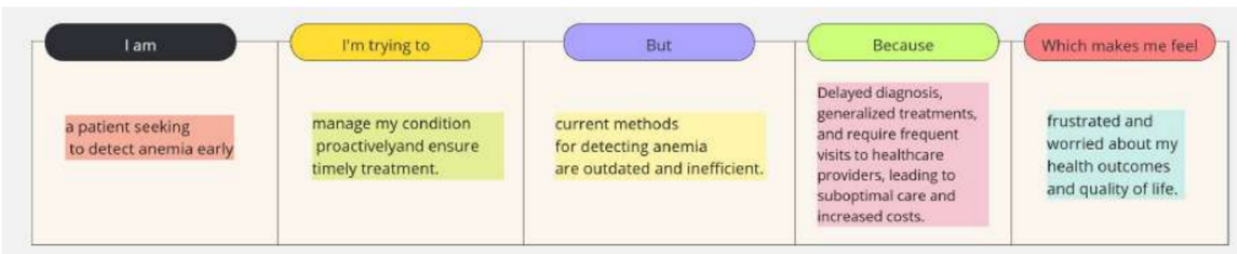
1.2 Objectives

1. **Develop Robust Machine Learning Models for Anemia Detection**
    a. **Objective:** Create and refine machine learning models trained on comprehensive datasets to accurately detect anemia based on key blood parameters such as hemoglobin levels, red blood cell counts, and other relevant biomarkers.
    b. **Goal:** Improve the precision and reliability of anemia diagnoses, ensuring that potential cases are identified promptly and accurately.
2. **Enhance Data Collection and Quality**
    a. **Objective:** Improve the quality and breadth of data collected from patients to support more accurate machine learning model training and validation.
    b. **Goal:** Ensure that the models have access to high-quality data, including diverse patient profiles and comprehensive blood parameter records, to enhance their predictive accuracy.
3. **Enable Early and Proactive Intervention**
    a. **Objective:** Use machine learning models to identify early signs of anemia, allowing for timely medical interventions and treatment plans.
    b. **Goal:** Reduce the progression and severity of anemia through early detection and management, ultimately improving patient health outcomes.

CHAPTER 2
PROJECT INITIALIZATION AND PLANNING PHASE

2.1 Define Problem Statement

Anemia remains a widespread and underdiagnosed condition, particularly in rural and underserved areas, due to outdated and inefficient diagnostic and management methods. Patients often experience delayed detection, generalized treatment plans, and the necessity for frequent in-person visits. These issues result in suboptimal care, increased healthcare costs, and significant patient dissatisfaction. The current approach fails to provide timely, personalized, and continuous care, exacerbating the condition's impact on patients' health and quality of life. Addressing these challenges is crucial for improving patient outcomes and overall satisfaction with anemia management.



2.2 Project Proposal (Proposed Solution)

This project aims to significantly improve anemia management using machine learning, resulting in better patient outcomes and satisfaction. Through timely diagnosis, individualized treatment, and continuous care, we can mitigate the impact of anemia and enhance the quality of life for affected individuals

2.3 Resource Requirements

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	T4 GPU
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, numpy, matpotlib, seaborn
Development Environment	IDE, version control	Jupyter Notebook, Git,Spyder
Data		
Data	Source, size, format	Kaggle dataset, 37KB,csv

CHAPTER 3

DATA COLLECTION AND PREPROCESSING PHASE

3.1 Data collection plan

Section	Description
Project Overview	Anaemia - sense utilizes machine learning models trained on vast datasets of blood parameters and patient profiles to detect early signs of anaemia. By analysing key indicators such as haemoglobin levels, red blood cell counts, and other relevant biomarkers, the system can flag potential cases for further investigation by healthcare professionals. Early detection enables timely interventions and treatment plans, improving patient outcomes.
Data Collection Plan	Kaggle dataset
Raw Data Sources Identified	The raw data sources for this project include datasets obtained from Kaggle, a popular platform for data science competitions and repositories. The provided sample data represents a subset of the collected information, encompassing variables such as gender, Hemoglobin, MCH, MCV and MCHC details for machine learning analysis

3.2 Raw data sources

Source name	**Description**	**Location / URL**	**Format**	**Size**	**Access permissions**
Kaggle dataset	The dataset comprises Gender, MCV, MCH, MCHC and the overall result	https://www.kaggle.com/datasets/biswaranjanrao/anemiadataset	CSV	37KB	Public

3.3 Data Quality Report

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

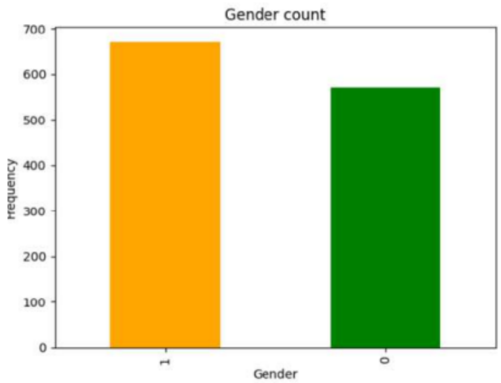
Data Source	Data Quality	Issue Severity	Resolution Plan
Kaggle Dataset	Female count is observed to be more than male count	Moderate	The dataset has more females than males, it means the data is imbalanced. To address this issue, Undersampling can be used. Undersampling involves reducing the number of samples in the majority class (females, in this case) to match the number of samples in the minority class (males). This creates a balanced dataset where both classes have an equal number of samples.

3.4 Data Exploration and Preprocessing

Dataset variables will be statistically analyzed to identify patterns and outliers, with Python employed for preprocessing tasks like normalization and feature engineering. Data cleaning will address missing values and outliers, ensuring quality for subsequent analysis and modeling, and forming a strong foundation for insights and predictions

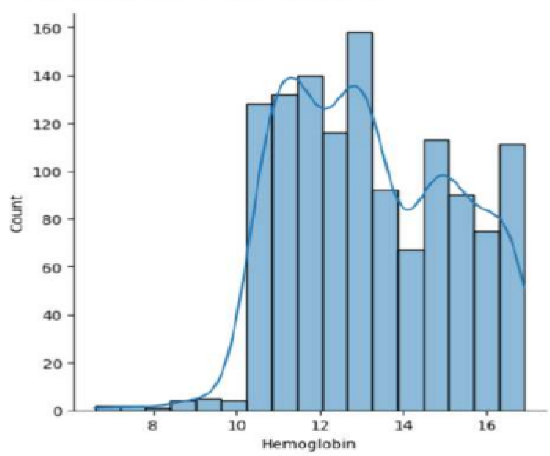
<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data overview</td><td>Dimension:
1421 rows x 6 columns
#Descriptive statistical
df.describe()
<table>
<tr><th></th><th>Gender</th><th>Hemoglobin</th><th>MCH</th><th>MCHC</th><th>MCV</th><th>Result</th></tr>
<tr><td>count</td><td>1240.000000</td><td>1240.000000</td><td>1240.000000</td><td>1240.000000</td><td>1240.000000</td><td>1240.000000</td></tr>
<tr><td>mean</td><td>0.540323</td><td>13.218145</td><td>22.903952</td><td>30.277984</td><td>85.620968</td><td>0.500000</td></tr>
<tr><td>std</td><td>0.498573</td><td>1.976190</td><td>3.993624</td><td>1.394515</td><td>9.673794</td><td>0.500202</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>11.500000</td><td>19.400000</td><td>29.100000</td><td>77.300000</td><td>0.000000</td></tr>
<tr><td>50%</td><td>1.000000</td><td>13.000000</td><td>22.700000</td><td>30.400000</td><td>85.300000</td><td>0.500000</td></tr>
<tr><td>75%</td><td>1.000000</td><td>14.900000</td><td>26.200000</td><td>31.500000</td><td>94.225000</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></td></tr>
<tr><td>Univariate analysis</td><td></td></tr>
</table>

```
#Univariate Analysis: Bar graph

output=df['Gender'].value_counts()
output.plot(kind='bar',color=['orange','green'])
plt.xlabel('Gender')
plt.ylabel('Frequency')
plt.title('Gender count')
plt.show()
```



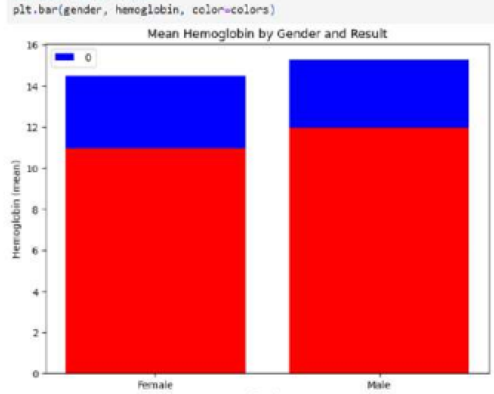

```
#Univariate analysis: displot

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

```
<seaborn.axisgrid.FacetGrid at 0x1e3b0ce2b10>
```



Bivariate Analysis

```
mean_hg = df.groupby(['Gender', 'Result'])['Hemoglobin'].mean().reset_index()
print(mean_hg)

gender = mean_hg['Gender'].tolist()
result = mean_hg['Result'].tolist()
hemoglobin = mean_hg['Hemoglobin'].tolist()

# Define colors based on result
colors = ['blue' if r == 0 else 'red' for r in result]
plt.figure(figsize=(8, 6))

# Create the bar chart
plt.bar(gender, hemoglobin, color=colors)
```



3.5 Data Preprocessing Code Screenshots

Loading Data

```
df = pd.read_csv("data/anemia.csv")
df.head()
```

	Gender	Hemoglobin	MCH	MCHC	MCV	Result
0	1	14.9	22.7	29.1	83.7	0
1	0	15.9	25.4	28.3	72.0	0
2	0	9.0	21.5	29.6	71.2	1
3	0	14.9	16.0	31.4	87.5	0
4	1	14.7	22.0	28.2	99.5	0

Handling Missing Data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1421 entries, 0 to 1420
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Gender      1421 non-null   int64
 1   Hemoglobin  1421 non-null   float64
 2   MCH         1421 non-null   float64
 3   MCHC        1421 non-null   float64
 4   MCV         1421 non-null   float64
 5   Result      1421 non-null   int64
dtypes: float64(4), int64(2)
memory usage: 66.7 KB
```

```
df.isnull().sum()
```

```
Gender        0
Hemoglobin    0
MCH           0
MCHC          0
MCV           0
Result        0
dtype: int64
```

Data Transformation

```
# female count is observed to be more than male so we balance it using undersampling
from sklearn.utils import resample
majorclass = df[df['Result'] == 0]
minorclass = df[df['Result'] == 1]
major_downsample = resample(majorclass, replace=False, n_samples=len(minorclass),random_state=42)
df = pd.concat([major_downsample,minorclass])
df['Result'].value_counts()
```

```
Result
0    620
1    620
Name: count, dtype: int64
```

CHAPTER 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	User's gender	Yes	Gender play a vital role in diagonsis of health issues
Hemoglobine	Hemoglobin is a protein in RBCs that carries oxygenn	Yes	Vital indicator for blood related diagnoses
MCH	Mean corpuscular emoglobin signifies the average amount of hemoglobin within a blood cell	Yes	Vital indicator for blood related diagnoses
MCHC	Mean corpuscular hemoglobin concentration is a measure of concentration of hemoglobin in RBCs	Yes	Vital indicator for blood related diagnoses
MCV	Mean corpuscular volume measures the average size of RBCs	Yes	Vital indicator for blood related diagnoses

4.2 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.

```
import pickle
import warnings
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, f1_score, confusion_matrix
```

```python
# Load and split data
df = pd.read_csv("data/anemia.csv")
X = df.drop('Result', axis=1)
Y = df['Result']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=20)

# Train and evaluate models
models = {
    'Logistic Regression': LogisticRegression(random_state=20),
    'Random Forest': RandomForestClassifier(random_state=20),
    'Decision Tree': DecisionTreeClassifier(random_state=20),
    'Gaussian Naive Bayes': GaussianNB(),
    'SVM': SVC(random_state=20),
    'Gradient Boosting': GradientBoostingClassifier(random_state=20)
}
```

4.3 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	Description	Hyper-parameters	Performance metrics
Logistic Regression	Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks.	--	F1= 1
Random Forest	Ensemble of decision trees; robust, handles complex relationships, reduces overfitting, and provides feature importance for anemia diagnosis.	--	F1= 1
Decision Tree	Simple tree structure; interpretable, captures non-linear relationships, suitable for initial insights into anemia diagnosis patterns.	--	F1= 1
Gaussian Naive Bayes	Interpretable model for initial exploration, good at capturing some non-linear relationships in anemia diagnosis patterns.	--	F1= 0.940
SVM	Powerful for creating separation hyperplanes to divide healthy and anemic patients.	--	F1= 0.902
Gradient Boosting	Gradient boosting with trees; optimizes predictive performance, handles complex relationships, and is suitable anemia diagnosis.	--	F1= 1

CHAPTER 5

MODEL OPTIMIZATION AND TUNING PHASE

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

5.1 Hypeparameter tuning documentation

Model	Tuned Hyperparameters	Optimal Values

Logistic Regression	`'Logistic Regression': {`
	`'C': [0.01, 0.1, 1, 10],`
	`'penalty': ['l1', 'l2']`
		`Model: Logistic Regression - Best Parameters:`
		`{'C': 10, 'penalty': 'l2'}`

Random Forest	`'Random Forest': {`
	`'n_estimators': [100, 200, 300],`
	`'max_depth': [4, 6, 8],`
	`},`
		`Model: Random Forest - Best Parameters:`
		`{'max_depth': 4, 'n_estimators': 100}`

Decision Tree	`'Decision Tree': {`
	`'max_depth': [3, 5, 8],`
	`'min_samples_split': [2, 5, 10]`
		`Model: Decision Tree - Best Parameters:`
		`{'max_depth': 3, 'min_samples_split': 2}`

Gradient Boosting	`'Gradient Boosting': {`
	`'n_estimators': [100, 200, 300],`
	`'learning_rate': [0.1, 0.01, 0.001]`
	`},`
		`Model: SVM - Best Parameters:`
		`{'C': 10, 'kernel': 'linear'}`

SVM	`'SVM': {`
	`'C': [0.1, 1, 10],`
	`'kernel': ['linear', 'rbf']`
	`}`
		`Confusion Matrix:`
		`[[146  21]`
		`[  7 111]]`

5.2 Performance metrics comparision report

Model	Optimized Metric

Logistic Regression	`Classification Report:`
	`precision recall f1-score support`
	`0 1.00 1.00 1.00 167`
	`1 1.00 1.00 1.00 118`
	`accuracy 1.00 285`
	`macro avg 1.00 1.00 1.00 285`
	`weighted avg 1.00 1.00 1.00 285`

	Confusion Matrix: [[167 0] [0 118]]
Random Forest	Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 167 1 1.00 1.00 1.00 118 accuracy 1.00 285 macro avg 1.00 1.00 1.00 285 weighted avg 1.00 1.00 1.00 285 Confusion Matrix: [[167 0] [0 118]]
Decision Tree	Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 167 1 1.00 1.00 1.00 118 accuracy 1.00 285 macro avg 1.00 1.00 1.00 285 weighted avg 1.00 1.00 1.00 285 Confusion Matrix: [[167 0] [0 118]]
Gradient Boosting	Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 167 1 1.00 1.00 1.00 118 accuracy 1.00 285 macro avg 1.00 1.00 1.00 285 weighted avg 1.00 1.00 1.00 285 Confusion Matrix: [[167 0] [0 118]]
SVM	Classification Report: precision recall f1-score support 0 0.95 0.87 0.91 167 1 0.84 0.94 0.89 118 accuracy 0.90 285 macro avg 0.90 0.91 0.90 285 weighted avg 0.91 0.90 0.90 285 Confusion Matrix: [[146 21] [7 111]]

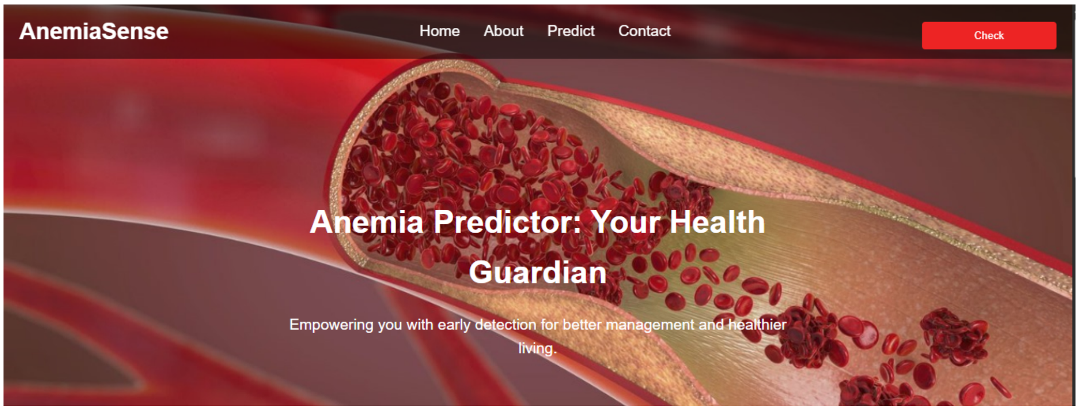
5.3 FINAL MODEL JUSTIFICATION

Final Model	Reasoning
Gradient Boosting	The Gradient Boosting model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.

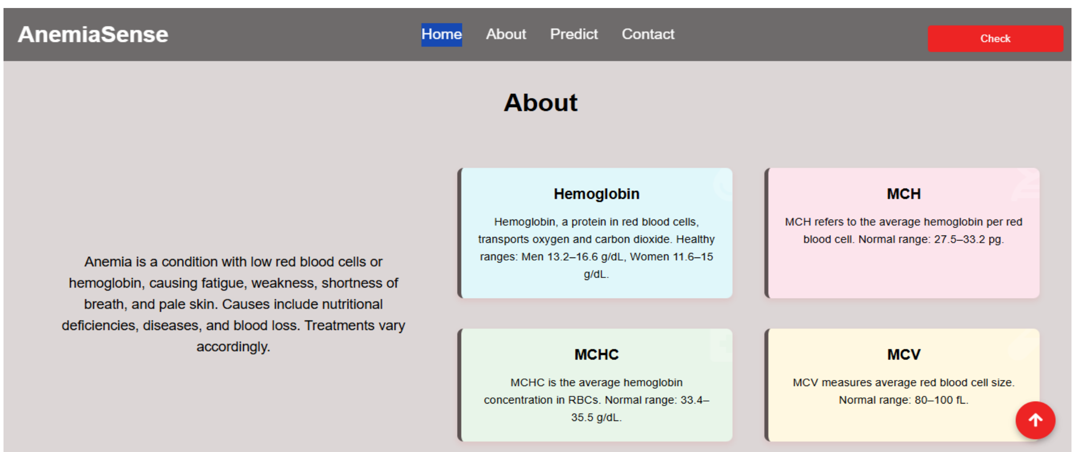
CHAPTER 6
RESULTS

6.1 Output Screenshots

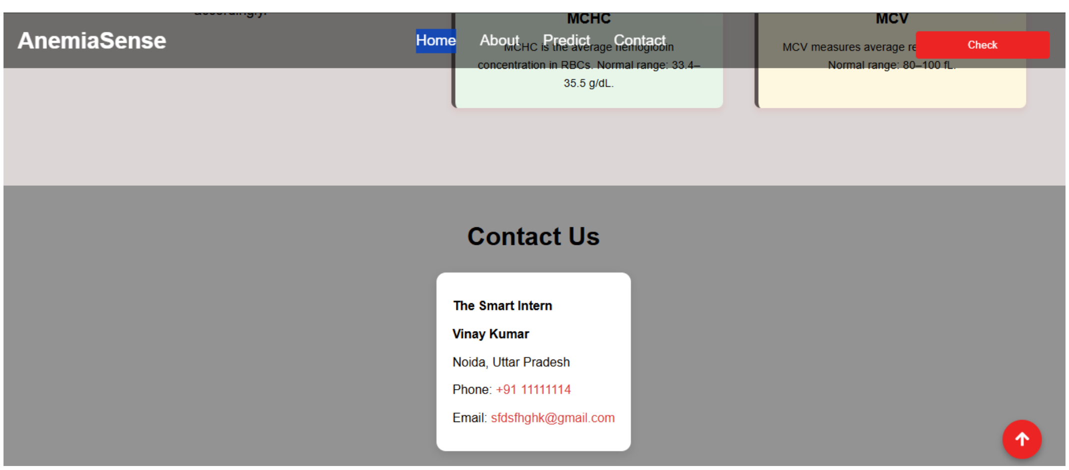
a. Home Page



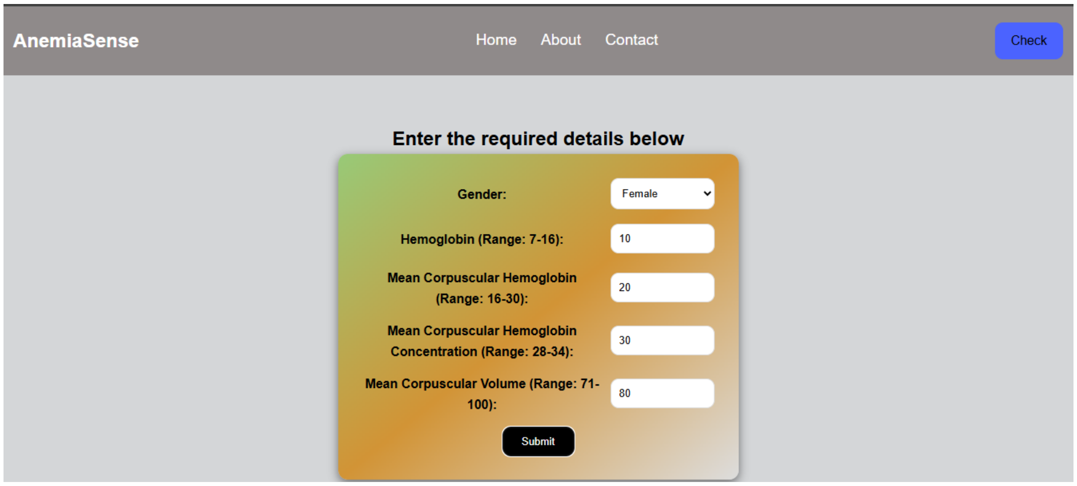
b. About



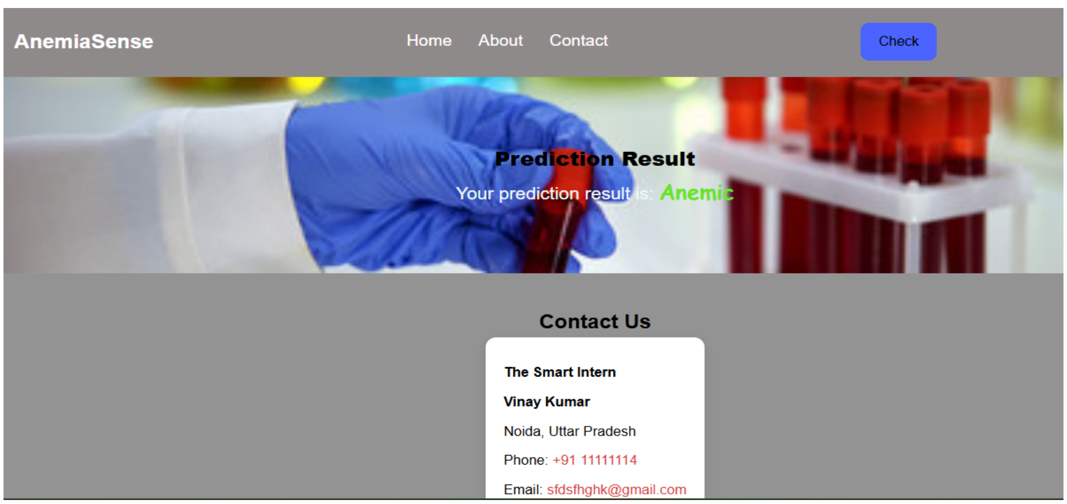
c. Contact

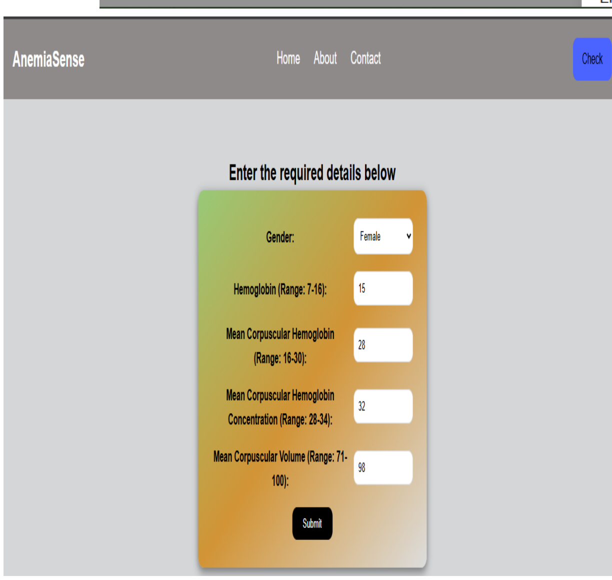


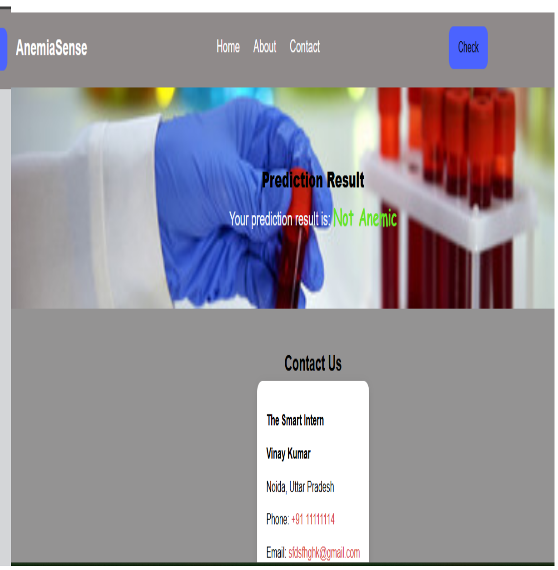
d. Predict_ User_Details Form



e. Output







CHAPTER 7
ADVANTAGES AND DISADVANTAGES

7.1 Advantages

1. **Improved Diagnostic Accuracy**
    - Machine learning algorithms can analyze large datasets and identify patterns that may be missed by traditional diagnostic methods, leading to more accurate anemia detection.
2. **Early Detection**
    - By identifying early signs of anemia, AnemiaSense allows for timely medical interventions, preventing the condition from progressing and reducing the risk of complications.
3. **Remote Monitoring**
    - Patients can be monitored remotely, reducing the need for frequent in-person visits and making it easier to manage anemia in remote or underserved areas.
4. **Efficiency for Healthcare Providers**
    - Automating the anemia detection process streamlines clinical workflows, allowing healthcare providers to focus on patient care and critical decisionmaking.
5. **Comprehensive Data Analysis**
    - AnemiaSense can analyze a wide range of blood parameters and patient demographics, providing a holistic view of a patient's health status and supporting more informed clinical decisions.

7.2 Disadvantages

1. **Integration Challenges**
    - Integrating AnemiaSense with existing healthcare systems and electronic health records (EHR) can be complex and may require significant time and resources.
2. **Initial Costs**
    - Developing, implementing, and maintaining AnemiaSense can involve significant initial costs, which may be a barrier for some healthcare providers.
3. **Limited Access to Technology**
    - Patients in areas with limited access to technology or digital health platforms may not benefit fully from AnemiaSense’s capabilities, potentially widening the healthcare gap.

CHAPTER 8
CONCLUSION

AnemiaSense represents a significant advancement in the field of healthcare technology, specifically targeting the detection and management of anemia through sophisticated machine learning algorithms. By focusing on precise and early detection, the project promises to enhance patient outcomes. The integration of remote monitoring capabilities further broadens its reach, offering continuous care for patients in diverse settings. Overall, AnemiaSense has the potential to revolutionize anemia care, making it more accurate, proactive, and patient-centered. By addressing the associated challenges and leveraging its innovative features, healthcare providers can significantly improve the diagnosis and management of anemia, ultimately leading to better health outcomes for patients.

CHAPTER 9
FUTURE SCOPE

Global Implementation and Accessibility

- **Scalability:** Design scalable solutions that can be deployed in diverse healthcare settings, from well-equipped urban hospitals to resource-limited rural clinics.
- **Low-cost Solutions:** Develop cost-effective versions of AnemiaSense to ensure accessibility for underserved populations and low-income regions, potentially through partnerships with NGOs and government health programs.

Integration with Telemedicine Platforms

- **Telehealth Integration:** Integrate AnemiaSense with telemedicine platforms to facilitate remote consultations and follow-ups, especially beneficial during pandemics or in areas with limited healthcare access.
- **Mobile Health Applications:** Develop mobile applications that allow patients to monitor their anemia status, receive alerts, and communicate with healthcare providers, enhancing patient engagement and self-management.

Personalized Treatment Optimization

- **Genomic Data Integration:** Incorporate genomic and epigenetic data to further personalize treatment plans based on individual genetic profiles and potential predispositions to different types of anemia.

- **Lifestyle and Environmental Factors:** Analyze lifestyle and environmental factors in conjunction with clinical data to optimize treatment plans and preventive measures.

Collaborative Research and Development

- **Partnerships:** Foster collaborations with academic institutions, research organizations, and pharmaceutical companies to advance the science of anemia detection and treatment.
- **Clinical Trials:** Conduct large-scale clinical trials to validate the effectiveness and reliability of AnemiaSense in diverse patient populations and clinical settings.

Regulatory Approvals and Standards

- **Regulatory Compliance:** Ensure AnemiaSense meets regulatory standards and obtains necessary approvals from health authorities such as the FDA, EMA, and other international bodies.
- **Industry Standards:** Contribute to the development of industry standards for AIdriven diagnostic tools, promoting best practices and interoperability across different healthcare systems.

Advanced Data Analytics

- **Predictive Analytics:** Develop predictive analytics capabilities to forecast anemia progression and treatment outcomes, enabling proactive and preventive healthcare.
- **Big Data Utilization:** Leverage big data analytics to identify trends and patterns in anemia incidence, treatment efficacy, and patient outcomes on a population level.

CHAPTER 10
APPENDIX

model.py

```
import pickle
import warnings
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Load and split data
df = pd.read_csv("data/anemia.csv")


X = df.drop('Result', axis=1)
Y = df['Result']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=20)

# Train and evaluate models
models = {
    'Logistic Regression': LogisticRegression(random_state=20),
    'Random Forest': RandomForestClassifier(random_state=20),
    'Decision Tree': DecisionTreeClassifier(random_state=20),
    'Gaussian Naive Bayes': GaussianNB(),
    'SVM': SVC(random_state=20),
    'Gradient Boosting': GradientBoostingClassifier(random_state=20)
}

results = {}

for name, model in models.items():
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    results[name] = {'Accuracy': acc, 'Report': report}
```

```
# Print comparison of models
compare_models = pd.DataFrame.from_dict({name: data['Accuracy'] for name, data in
results.items()}, orient='index', columns=['Accuracy'])
print(compare_models)

# Save the best model (Gradient Boosting Classifier)
best_model = models['Gradient Boosting']
with open("model.pkl", "wb") as model_file:
    pickle.dump(best_model, model_file)

# Test prediction
test_input = [[0, 12.4, 23, 32.2, 76.1]]
prediction = best_model.predict(test_input)
print(f"Test prediction for input {test_input}: {prediction}")

# Handle potential warnings
warnings.warn("Ensure the input data has valid feature names when making predictions.")
```

app.py

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

app = Flask(__name__)

# Load the machine learning model
with open('model.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

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

@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        # Get form data
        gender = request.form['gender']
        hemoglobin = float(request.form['hemoglobin'])
        mch = float(request.form['mch'])
        mchc = float(request.form['mchc'])
```

```
        mcv = float(request.form['mcv'])

        # Preprocess input data
        gender = 1 if gender == 'male' else 0  # Example: convert gender to numerical

        # Create a numpy array for prediction
        input_features = np.array([[gender, hemoglobin, mch, mchc, mcv]])

        # Make prediction
        prediction = model.predict(input_features)

        # Determine result based on prediction
        result = 'Anemic' if prediction[0] == 1 else 'Not Anemic'

        return render_template('result.html', result=result)
    return render_template('predict.html')

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

Github link: https://github.com/vinaygupta88/Anemia-Sense.git