Anemia Sense: Leveraging Machine Learning for Precise Anemia Recognitions

Team ID: SWTID1720078683

Team Members:

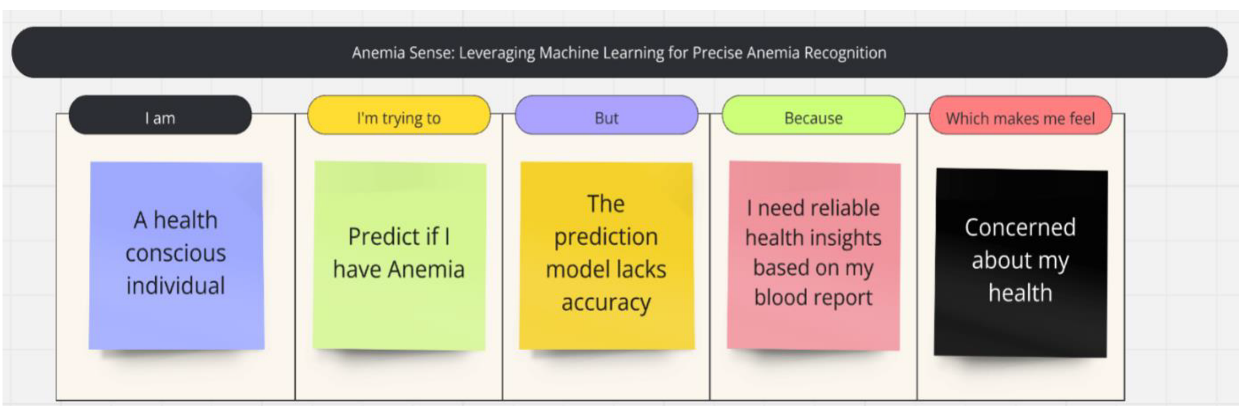
1. **Dinesh. R**
2. **G. Achuth**
3. **Lakshmanan. L**
4. **Agash. JP**

Project Initialization and Planning Phase

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

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

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

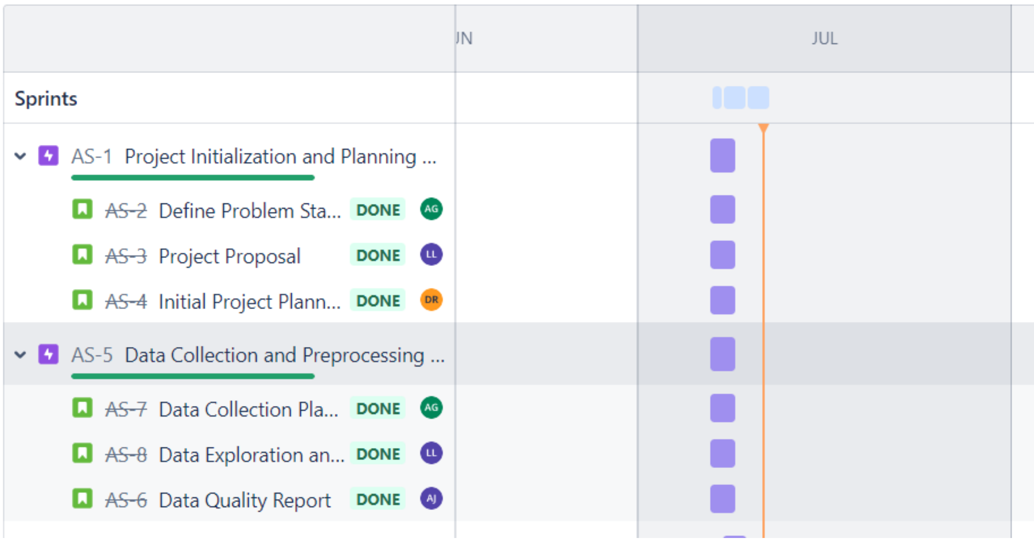
Product Backlog, Sprint Schedule, and Estimation (4 Marks)

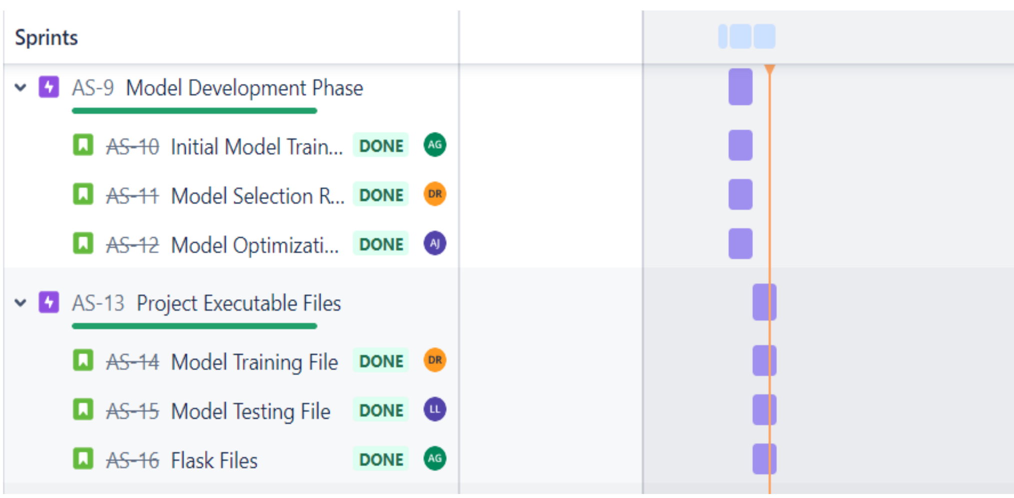
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Project Initialization and Planning Phase	AS-2	Define Problem Statements	3	High	G.Achuth	7-07-2024	8-07-2024
Sprint-1	Project Initialization 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

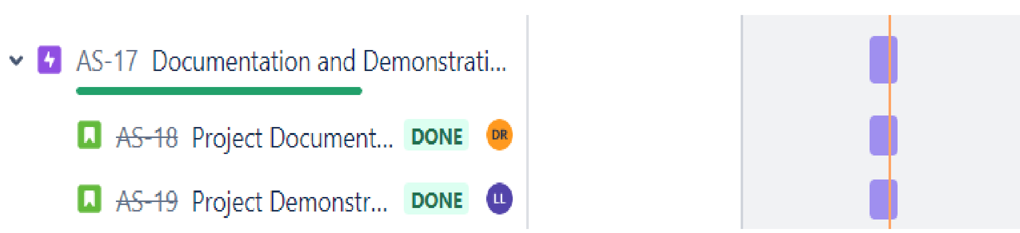
Sprint-1	Data Collection and Preprocessing Phase	AS-6	Data Quality Report	2	Medium	G.Achuth	7-07-2024	8-07-2024
Sprint-1	Data Collection and Preprocessi	AS-7	Data Collection Plan and Raw Data Sources Identification Report	2	Medium	Lakshmanan.L	7-07-2024	8-07-2024
Sprint-1	Data Collection and Preprocessi	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	Lakshman an.L	10-07-2024	11-07-2024

Screenshots:







Data Collection and Preprocessing Phase

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.

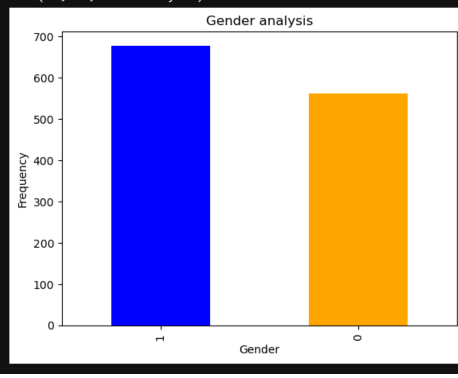
<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data Overview</td><td>
<pre>]: data.describe()</pre>
<table>
<tr><th></th><th>Gender</th><th>Hemoglobin</th><th>MCH</th><th>MCHC</th><th>MCV</th><th>Result</th></tr>
<tr><td>count</td><td>1421.000000</td><td>1421.000000</td><td>1421.000000</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

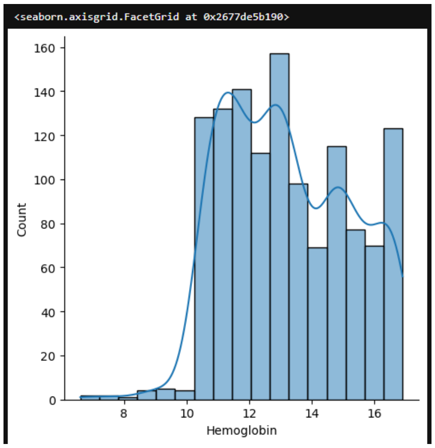
```
[14]: gender = data['Gender'].value_counts()
      gender.plot(kind = 'bar',color = ['blue','orange'])
      plt.xlabel('Gender')
      plt.ylabel('Frequency')
      plt.title('Gender analysis')
```

```
[14]: Text(0.5, 1.0, 'Gender analysis')
```



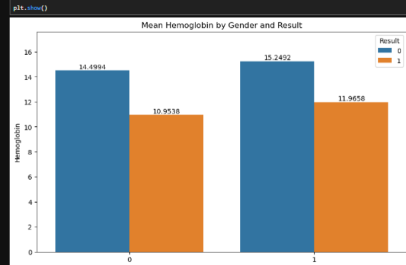

```
[15]: sns.displot(data['Hemoglobin'],kde = True)
```

```
<seaborn.axisgrid.FacetGrid at 0x2677de5b190>
```



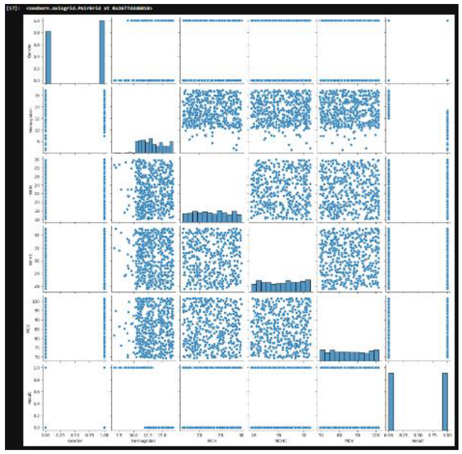
Bivariate Analysis

```
[16]: 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()
```



<table>
<tr>
<td>Multivariate Analysis</td>
<td>

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



</td>
</tr>
<tr>
<td colspan="2">Data Preprocessing Code Screenshots</td>
</tr>
<tr>
<td>Loading Data</td>
<td>

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

</td>
</tr>
<tr>
<td>Handling Missing Data</td>
<td>

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

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

</td>
</tr>
</table>

Data Transformation	`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`

Save Processed Data	`data.to_csv('anemia.csv',index=False)`

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.

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/1KMJFNFGwoaQoAoulPabMEHcT1bvqEXau/view?usp=sharing	CSV	33.8 KB	Public

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

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)`

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

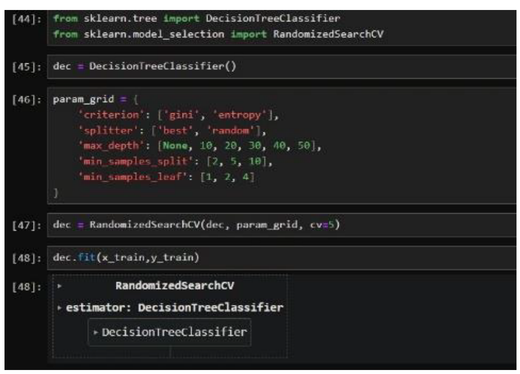
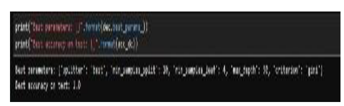
Gradient Boosting Classifier	Gradient Boosting is an ensemble technique that builds models sequentially, with each new model attempting to correct the errors of the previous models,	-	Accuracy – 1.00

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.

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.

Hyperparameter Tuning Documentation:

Model	Tuned Hyperparameters	Optimal Values
Decision Tree		

<table>
<tr>
<td>Random Forest</td>
<td><pre>
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
</pre></td>
<td><pre>
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
</pre></td>
</tr>
</table>

<table>
<tr>
<td>Gradient

Boosting</td>
<td><pre>
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
</pre></td>
<td><pre>
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
</pre></td>
</tr>
</table>

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>
</table>

Random Forest	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)
Gradient Boosting	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)

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