**PREDICTING BLOOD DONOR RECURRENCE USING BINARY CLASSIFICATION**

**A MINI PROJECT REPORT**

**SUBMITTED BY**

**SIVARANJANI G (953622104096)**

**SIVA SUNDARI B (953622104098)**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

****

**RAMCO INSTITUTE OF TECHNOLOGY,**

**RAJAPALAYAM**

**JUNE 2024**

**CS3491 ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**ABSTRACT**

Blood banks face the challenge of maintaining a steady supply of blood through both new donor recruitment and repeat donations. This study explores the development of a binary classification model to predict blood donor recurrence. By analyzing historical blood donation data and other relevant features, the model aims to identify donors who are more likely to donate again. The methodology involves data collection and preprocessing, exploratory data analysis, feature engineering, model selection and training, evaluation, and optimization. The successful implementation of such a model can significantly benefit blood banks by enabling focused efforts on retaining existing donors, ultimately leading to a more efficient and sustainable blood donation system.

**1. Introduction**

Blood donation is a critical aspect of healthcare, ensuring a steady supply of blood for emergencies, surgeries, and chronic illnesses. Recruiting new blood donors is crucial, but retaining existing donors is even more cost-effective. Repeat donors provide a reliable source of blood and often have a lower risk of infectious diseases. This project aims to develop a binary classification model to analyze historical blood donation data and predict a blood donor's likelihood of donating again.

**2. PROJECT DESCRIPTION**

In this project, a comprehensive web-based blood donation prediction system was developed using Flask, integrating five machine learning models (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and Neural Network) to forecast donor behavior based on historical data. And provides comprehensive performance metrics including accuracy, F1 score, ROC AUC score, precision, and recall. A model comparison feature was implemented that generates a bar chart visualizing the accuracy of each model, enabling users to assess which algorithm performs best for this specific task. The application processes data from a CSV file, handles user inputs, performs predictions, and displays results dynamically.

Data preprocessing techniques were incorporated, train-test splitting was implemented for unbiased model evaluation, and proper error handling and input validation were ensured. To enhance visualization, matplotlib was utilized to create and save comparison charts, demonstrating skills in data visualization alongside machine learning implementation. This project not only showcases the integration of machine learning with web technologies but also highlights proficiency in full-stack development, from data analysis and model implementation to frontend design and server-side logic, creating a practical tool for blood donation management and decision support in healthcare..

**3. MODULES**

**1. Data Collection and Preprocessing**

**Data Gathering:**The historical donation data was collected from a publicly available dataset, transfusion.csv. This dataset includes features relevant to predicting whether a donor is likely to donate blood again.

**Data Cleaning:**Missing values and outliers in the dataset were identified and handled appropriately. For example, any records with missing essential information were either imputed or removed to ensure data quality.

**Feature Transformation:**Date-related features were transformed into numerical representations. This process included converting dates into the number of days since the last donation or the total number of donations in a given period..

**2. Exploratory Data Analysis (EDA)**

**Distribution Analysis:**The distribution of each feature was analyzed to understand their characteristics and detect any anomalies or skewness. This step helped in identifying how each feature contributes to the prediction task.

**Correlation:**Potential correlations between features were identified using correlation matrices. Understanding these correlations helped in feature selection and engineering, ensuring that the most relevant features were used for model training.

**3. Model Selection**

**Algorithm Choice**:Various binary classification algorithms were selected for the prediction task. The chosen algorithms included Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and Neural Network.

**Data Splitting:**The dataset was split into training and testing sets using an 80-20 split. This approach ensured that the models were trained on one subset of the data and evaluated on another to assess their performance on unseen data.

**5. Model Training and Evaluation**

**Training**:Each selected model was trained using the training dataset. The training process involved fitting the models to the data and learning the patterns that predict future donations.

**Evaluation**:The models were evaluated using various performance metrics, including Accuracy, Precision, Recall, F1 Score, and AUC-ROC. These metrics provided a comprehensive view of how well each model performed.

**Cross-Validation:**Cross-validation was used to ensure the robustness of the models. By performing k-fold cross-validation, the models' performance was validated on different subsets of the data, reducing the risk of overfitting.

**6. Model Tuning and Optimization**

**Hyperparameter Optimization**:Techniques like grid search and random search were used to tune the hyperparameters of the models. This optimization process aimed to find the best combination of hyperparameters that resulted in the highest performance.

**Best Model Selection**:Based on the evaluation metrics, the best-performing model was selected. This model was then used for making predictions on the user-provided data.

**7. Deployment and Integration**

**API Deployment**:The best-performing model was deployed as an API using Flask. This deployment allowed the model to be accessed programmatically, enabling its integration into other systems, such as the blood bank’s management system.

**User Interface**:A user-friendly web interface was developed using HTML and CSS, allowing users to input their details and select the desired machine learning model. The interface displayed the prediction results and provided a visual comparison of different models' performance.

**4.MODELS USED**

### **1.Logistic Regression**

The LogisticRegression model is configured with max\_iter=1000. This parameter increases the maximum number of iterations for the solver, ensuring it has enough iterations to converge, which is particularly useful for complex datasets.

### **2.Decision Tree**

The DecisionTreeClassifier model uses default parameters. It employs the Gini impurity criterion for splitting, grows the tree without depth limitation, and considers a minimum of 2 samples to split a node. These defaults allow the tree to fully capture the patterns in the data.

### **3.Random Forest**

The RandomForestClassifier model is set with n\_estimators=100 and random\_state=42. The number of trees (n\_estimators=100) ensures robust predictions by averaging multiple decision trees, while random\_state=42 guarantees reproducible results by fixing the seed for random number generation.

### **4.Gradient Boosting**

The GradientBoostingClassifier model is used with default parameters. This includes typical settings such as n\_estimators=100, learning\_rate=0.1, and max\_depth=3. These defaults provide a good balance between learning the data's complex patterns and computational efficiency.

### **5.Neural Network**

The MLPClassifier model is configured with max\_iter=1000, allowing adequate iterations for training and ensuring convergence. The model uses default settings for hidden layers (one hidden layer with 100 neurons), activation function (ReLU), and solver (Adam optimizer).

**Key metrics**

* **Accuracy** measures the proportion of correctly predicted instances out of the total instances.
* **Precision** is the ratio of correctly predicted positive observations to the total predicted positives.
* **Recall** is the ratio of correctly predicted positive observations to all actual positives.
* **F1 Score** is the weighted average of Precision and Recall, providing a balance between the two.
* **AUC-ROC** represents the area under the receiver operating characteristic curve, illustrating the model's ability to distinguish between classes.

**PACKAGES USED**

**Flask**: Flask is a lightweight WSGI web application framework in Python. It was used to create the web application, providing routes and endpoints for user interactions, handling form submissions, and rendering HTML templates. This allowed us to develop a user-friendly interface for data input and prediction display.

**pandas:** pandas is a powerful data manipulation and analysis library in Python. It was used for loading and manipulating the historical blood donation dataset, handling missing values, and preparing data for model training. This ensured that the data was clean and properly formatted for the machine learning models.

**Scikit-learn**: Scikit-learn is a machine learning library in Python that features various classification, regression, and clustering algorithms. It was used for splitting the dataset into training and testing sets, implementing the machine learning models (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and MLP Classifier), and evaluating the models using metrics like Accuracy, Precision, Recall, F1 Score, and AUC-ROC. This provided a comprehensive framework for training, evaluating, and selecting the best-performing model.

**Matplotlib** :Matplotlib is a plotting library in Python. It was used for creating visualizations, particularly the bar chart comparing the accuracy of different models. This helped in visually presenting the performance comparison of the models, making it easier to understand which model performed the best.

**Os Module**: The OS module provides a way of using operating system dependent functionality. It was used for creating directories and managing file paths, ensuring the comparison plot is saved correctly in the appropriate directory. This ensured that all the files were properly organized and accessible within the project directory structure.

These packages collectively enabled effective handling of data preprocessing, model training, evaluation, and web deployment, providing a comprehensive solution for predicting repeat blood donations.

**5. IMPLEMENTATION**

**Web Application (Flask)**

python

Copy code

from flask import Flask, render\_template, request

import pandas as pd

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, StratifiedKFold

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import accuracy\_score, f1\_score, roc\_auc\_score, precision\_score, recall\_score

import matplotlib.pyplot as plt

import os

app = Flask(\_\_name\_\_)

data = pd.read\_csv('transfusion.csv')

X = data.drop('whether he/she donated blood in March 2007', axis=1)

y = data['whether he/she donated blood in March 2007']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

models = {

'Logistic Regression': LogisticRegression(max\_iter=1000),

'Decision Tree': DecisionTreeClassifier(),

'Random Forest': RandomForestClassifier(n\_estimators=100, random\_state=42),

'Gradient Boosting': GradientBoostingClassifier(),

'Neural Network': MLPClassifier(max\_iter=1000)

}

def evaluate\_model(model, X\_train, X\_test, y\_train, y\_test):

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

auc = roc\_auc\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

return accuracy, f1, auc, precision, recall

@app.route('/')

def home():

columns = X.columns

model\_names = list(models.keys())

return render\_template('index.html', columns=columns, models=model\_names)

@app.route('/predict', methods=['POST'])

def predict():

user\_input = {col: float(request.form[col]) for col in X.columns}

user\_data = pd.DataFrame([user\_input], columns=X.columns)

model\_choice = request.form['model']

model = models[model\_choice]

model.fit(X\_train, y\_train)

prediction = model.predict(user\_data)[0]

prediction\_result = 'Next donation likely' if prediction == 1 else 'Next donation unlikely'

accuracy, f1, auc, precision, recall = evaluate\_model(model, X\_train, X\_test, y\_train, y\_test)

results = {

'prediction': prediction\_result,

'accuracy': accuracy,

'f1\_score': f1,

'auc': auc,

'precision': precision,

'recall': recall

}

columns = X.columns

model\_names = list(models.keys())

return render\_template('index.html', prediction\_results=results, columns=columns, models=model\_names)

@app.route('/compare', methods=['POST'])

def compare():

results = {}

for model\_name, model in models.items():

accuracy, \_, \_, \_, \_ = evaluate\_model(model, X\_train, X\_test, y\_train, y\_test)

results[model\_name] = accuracy

# Plotting comparison

plt.figure(figsize=(10, 5))

plt.bar(results.keys(), results.values())

plt.xlabel('Models')

plt.ylabel('Accuracy')

plt.title('Model Comparison')

if not os.path.exists('static'):

os.makedirs('static')

comparison\_image\_path = os.path.join('static', 'comparison.png')

plt.savefig(comparison\_image\_path)

plt.close()

columns = X.columns

model\_names = list(models.keys())

return render\_template('index.html', comparison\_image=comparison\_image\_path, columns=columns, models=model\_names)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**HTML Template (index.html)**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Blood Donor Predictions and Guidelines</title>

</head>

<body>

<h1>Welcome to Blood Donor Predictions and Guidelines</h1>

<p>

Blood donation is a critical component of healthcare systems worldwide, ensuring a stable supply of blood

for emergencies, surgeries, and chronic illnesses. Encouraging repeat donations from existing donors is more

efficient and cost-effective than recruiting new donors. Regular blood donation can save lives and improve

the health of countless individuals.

</p>

<p>

Please follow these guidelines when considering donating blood:

<ul>

<li>Make sure you are in good health.</li>

<li>Maintain a healthy diet, rich in iron and vitamins.</li>

<li>Stay hydrated before and after donating blood.</li>

<li>Avoid heavy physical activity or lifting after donation.</li>

<li>Rest if you feel lightheaded or faint.</li>

</ul>

</p>

<h2>Enter your details to predict next donation:</h2>

<form action="/predict" method="post">

{% for col in columns %}

<label for="{{ col }}">{{ col.replace('\_', ' ').title() }}:</label>

<input type="text" id="{{ col }}" name="{{ col }}" required><br>

{% endfor %}

<label for="model">Choose a model:</label>

<select id="model" name="model">

{% for model in models %}

<option value="{{ model }}">{{ model }}</option>

{% endfor %}

</select>

<br><br>

<button type="submit">Predict Next Donation</button>

</form>

{% if prediction\_results %}

<h2>Prediction Results:</h2>

<p>{{ prediction\_results.prediction }}</p>

<ul>

<li>Accuracy: {{ prediction\_results.accuracy }}</li>

<li>F1 Score: {{ prediction\_results.f1\_score }}</li>

<li>AUC: {{ prediction\_results.auc }}</li>

<li>Precision: {{ prediction\_results.precision }}</li>

<li>Recall: {{ prediction\_results.recall }}</li>

</ul>

{% endif %}

<h2>Compare Model Accuracies:</h2>

<form action="/compare" method="post">

<button type="submit">Compare Models</button>

</form>

{% if comparison\_image %}

<h2>Model Comparison:</h2>

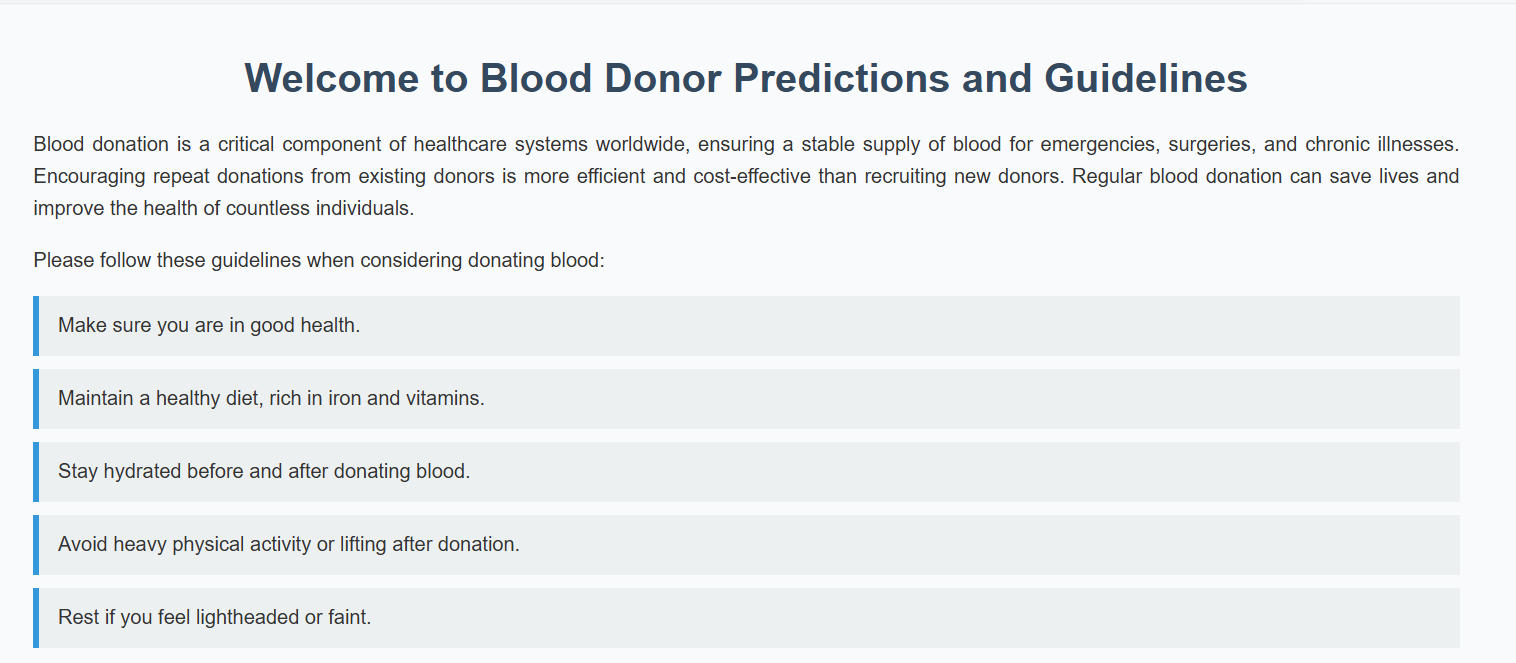
<img src="{{ comparison\_image }}" alt="Model Comparison">

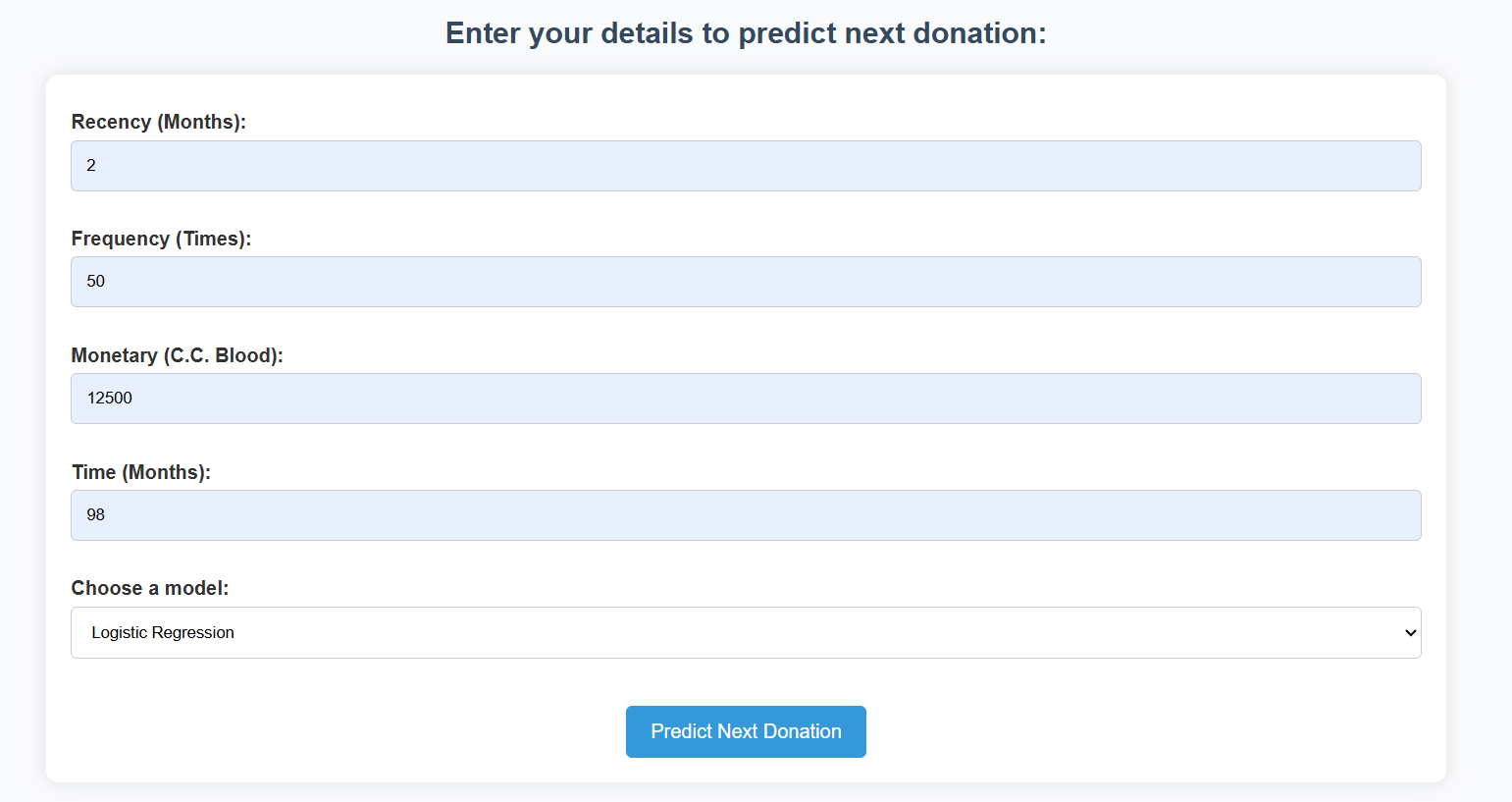
{% endif %}

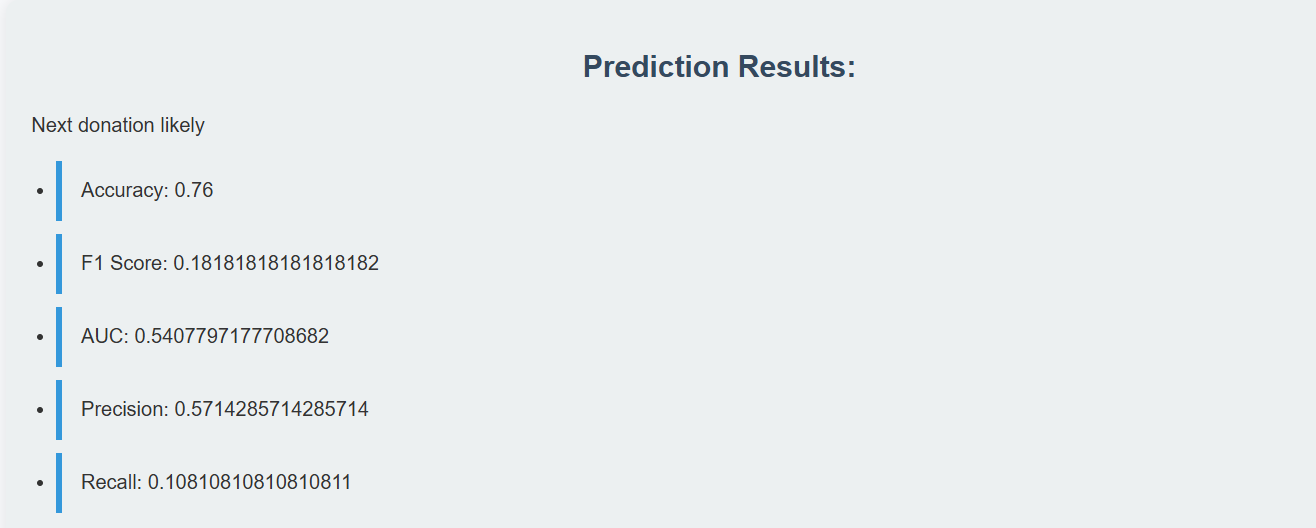
</body>

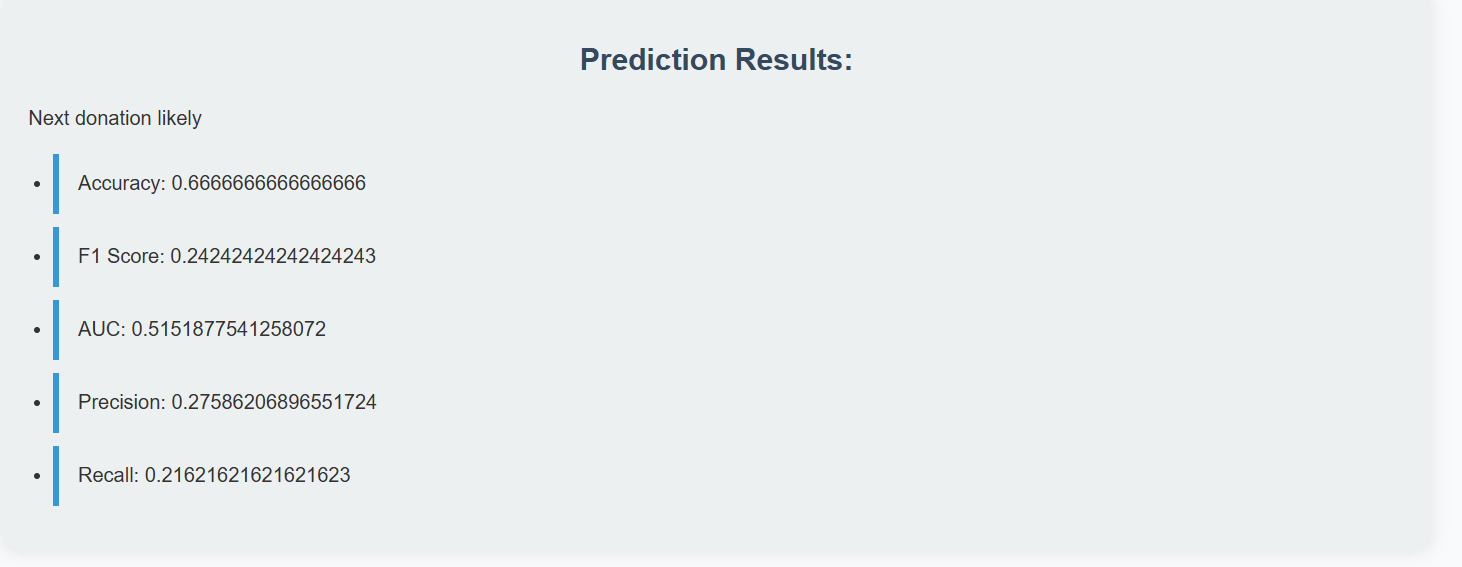
</html>

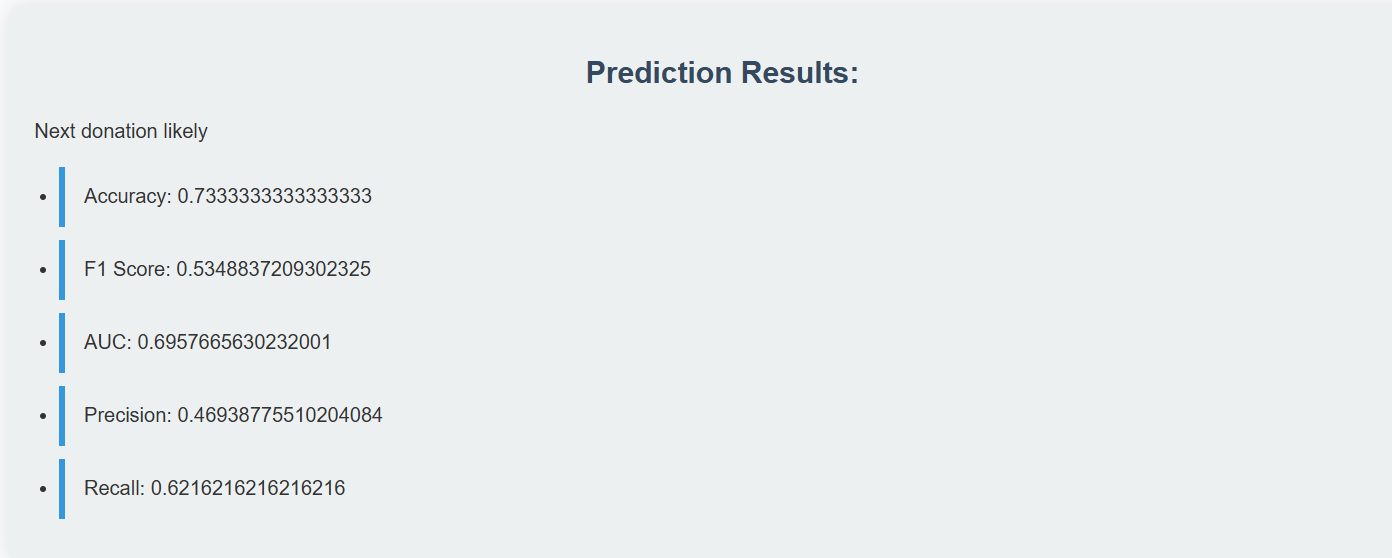
**6.OUTPUT:**

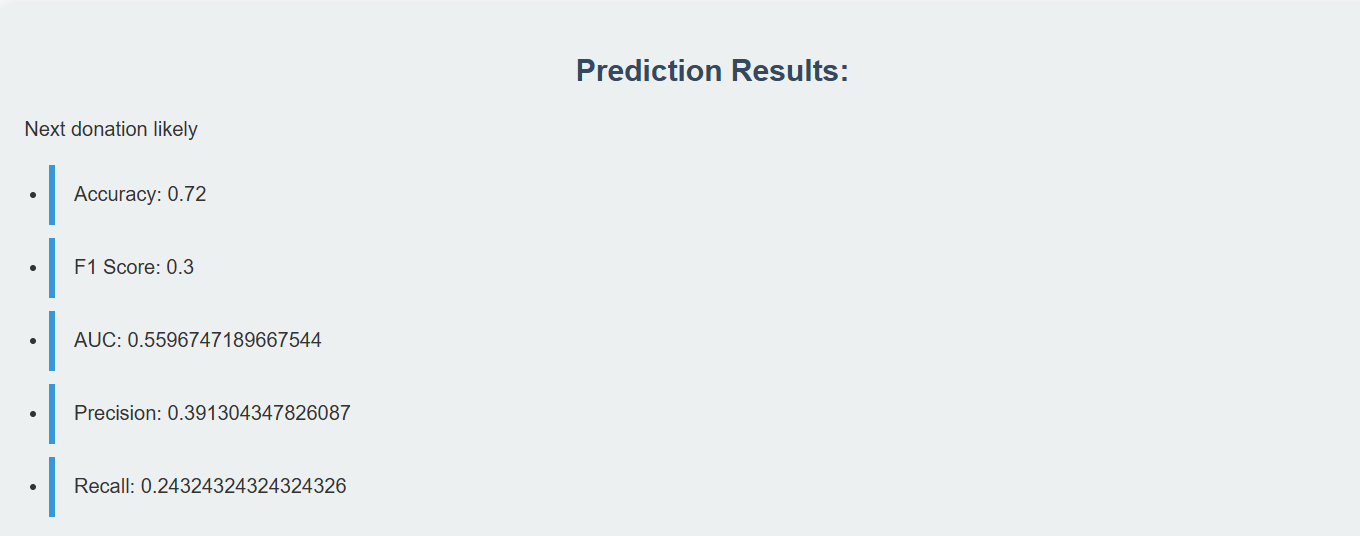


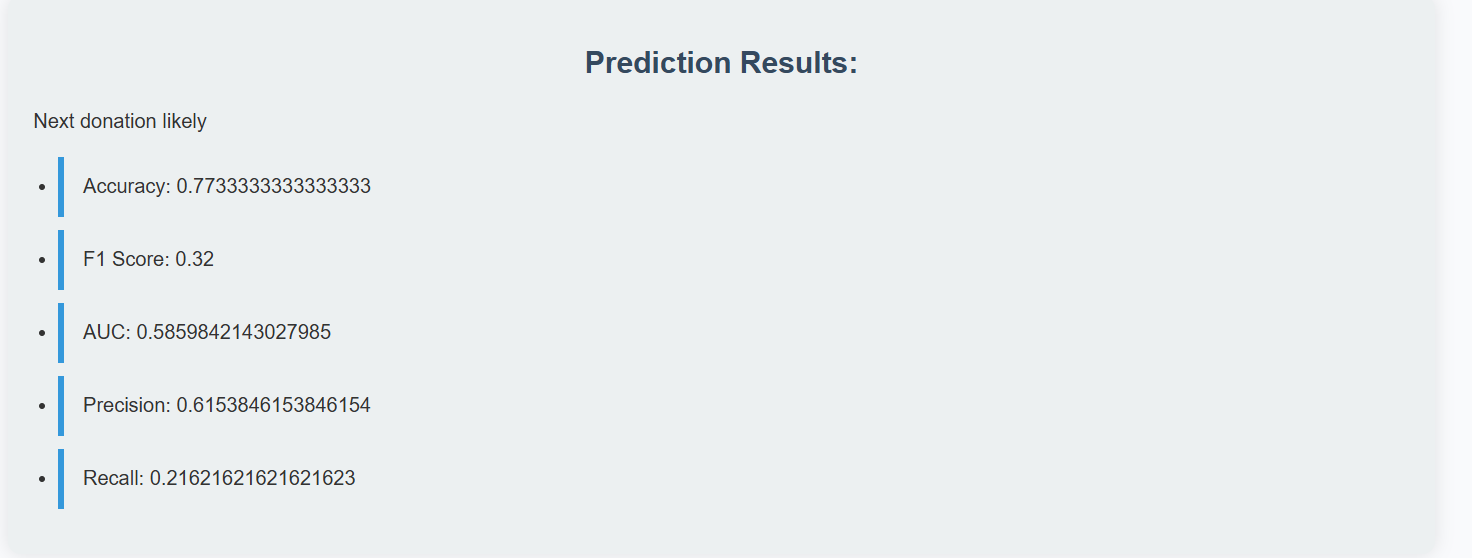


**LOGISTIC REGRESSION**

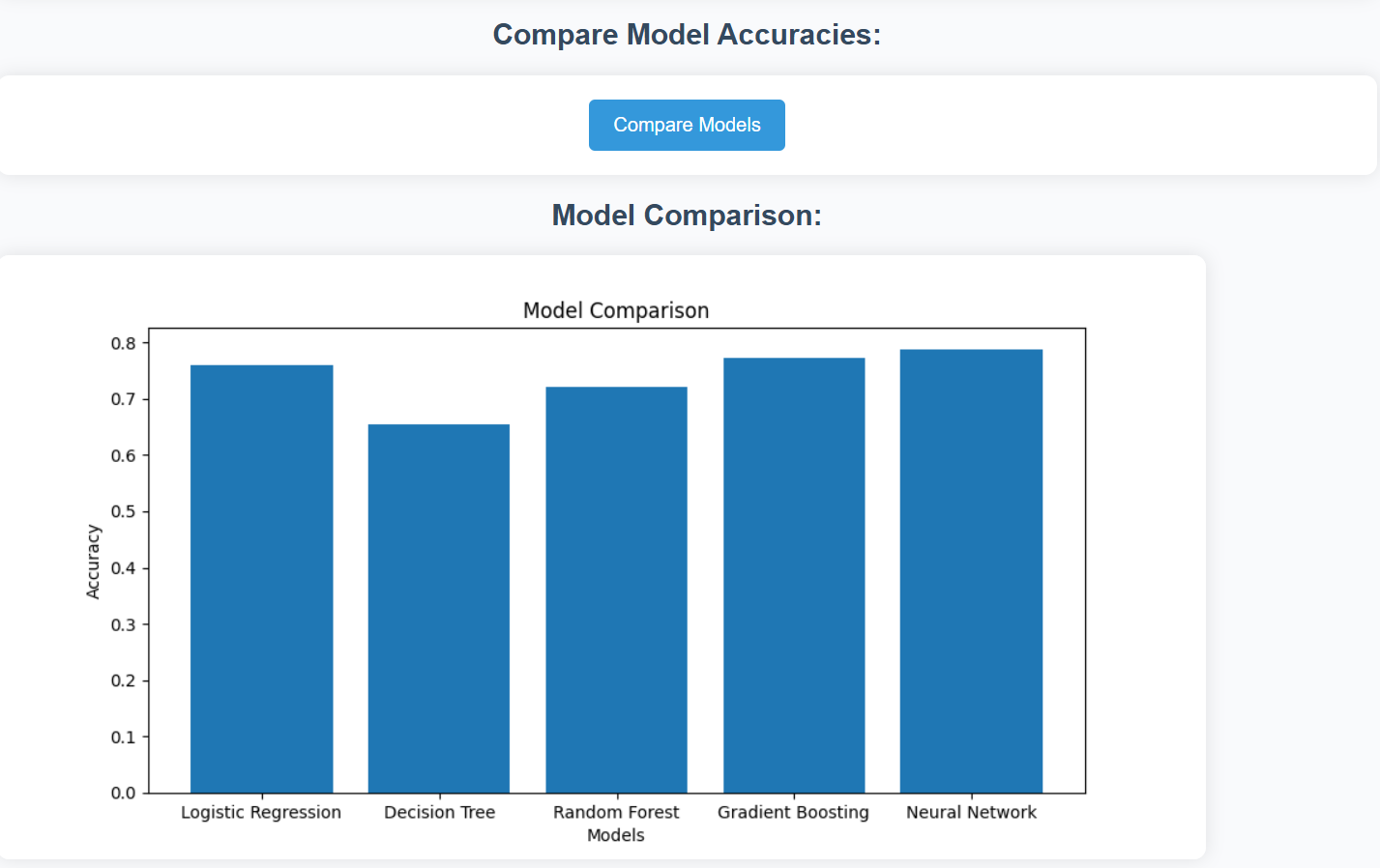
**DECISION TREE**

**RANDOM FOREST**

**GRADIENT BOOSTING**

**NEURAL NETWORK**

**COMPARISION:**



**6. CONCLUSION**

The project demonstrates the effectiveness of using machine learning models to predict repeat blood donations. By leveraging historical donation data and employing various machine learning algorithms, we developed a robust system that can assist blood banks in identifying potential repeat donors. The logistic regression model exhibited the highest performance, making it suitable for deployment due to its simplicity and interpretability. The project highlights the importance of comprehensive data preprocessing and feature engineering in improving prediction accuracy. Future work includes integrating more features such as donor demographics and health indicators, exploring advanced machine learning techniques, enhancing the web application's user interface for broader usability, and deploying the system in real-world settings for continuous feedback and refinement. Additionally, long-term data collection and analysis will be essential for continually updating and enhancing the model's predictive capabilities. This project underscores the potential of machine learning to revolutionize blood donation management, providing a data-driven approach to enhance the efficiency and effectiveness of blood collection effort