**Final Report of Traineeship Program 2025**

*On*

“PREDICT BLOOD DONATIONS”

**MEDTOUREASY**

****

28th May 2025



# ACKNOWLEDGMENTS

The traineeship opportunity that I had with MedTourEasy was a great change for learning and understanding the intricacies of the subject of Data Visualizations in Data Analytics; and also, for personal as well as professional development. I am very obliged for having a chance to interact with so many professionals who guided me throughout the traineeship project and made it a great learning curve for me.

Firstly, I express my deepest gratitude and special thanks to the Training & Development Team of MedTourEasy who gave me an opportunity to carry out my traineeship at their esteemed organization. Also, I express my thanks to the team for making me understand the details of the Data Analytics profile and training me in the same so that I can carry out the project properly and with maximum client satisfaction and also for sparing their valuable time in spite of his busy schedule.

I would also like to thank the team of MedTourEasy and my colleagues who made the working environment productive and very conducive.



# TABLE OF CONTENTS

Acknowledgments i

Abstract iii

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Topic** | **Page No.** |
| **1** | **Introduction** |  |
|  | 1.1 About the Company | 5 |
|  | 1.2 About the Project | 5 |
|  | 1.3 Objectives and Deliverables | 6 |
| **2** | **Methodology** |  |
|  | 2.1 Flow of the Project | 7 |
|  | 2.2 Use Case Diagram | 8 |
|  | 2.3 Language and Platform Used | 9 |
| **3** | **Implementation** |  |
|  | 3.1 Gathering Requirements and Defining Problem Statement | 10 |
|  | 3.2 Data Collection and Importing | 10 |
|  | 3.3 Designing Databases | 10 |
|  | 3.4 Data Cleaning | 11 |
|  | 3.5 Data Filtering | 11 |
|  | 3.6 AutoML with TPOT | 12 |
|  | 3.7Logistic Regression | 13 |
| **4** | **Result and analysis** |  |
|  | 4.1 Model performance and analysis | 14 |
|  | 4.2 Observation | 15 |
| **5** | **Sample Code Snippets and Explanation** |  |
|  | 5.1Data Import and Exploration | 16 |
|  | 5.2 TPOT Setup and Training | 17 |
|  | 5.3Logistic Regression Pipeline | 18 |
|  | 5.4Evaluation Metrics |  |
| **6** | **Conclusion** | 19 |
| **7** | **Future Scope** | 23 |
| **8** | **References** | 25 |



# ABSTRACT

The traineeship opportunity at MedTourEasy has been immensely enriching and offered a profound learning experience in data analytics and machine learning automation. I am sincerely thankful to the entire Training & Development Team of MedTourEasy for providing me with the platform to explore predictive modelling in real-world healthcare applications.

I extend special thanks to my mentors for their guidance and feedback throughout this project. Their support enabled me to better understand advanced machine learning techniques such as TPOT AutoML and their practical applications. I would also like to thank my fellow trainees and the technical team for fostering an environment conducive to learning and innovation.

This project focuses on predicting future blood donations by analyzing historical donation data using machine learning techniques. The prediction model helps healthcare organizations identify potential repeat donors, thereby optimizing donor engagement and blood inventory management.

We utilized the TPOT AutoML library to automatically discover the most suitable ML pipeline, and further benchmarked its performance against a baseline Logistic Regression model. The project included data cleaning, normalization, model training, evaluation, and pipeline optimization. The final output highlights performance metrics (AUC scores) and comparative analysis, ultimately helping in building a robust blood donation prediction system.



* 1. About the Company

MedTourEasy, a global healthcare company, provides you the informational resources needed to evaluate your global options. MedTourEasy provides analytical solutions to our partner healthcare providers globally.

* 1. About the Project

The "Predict Blood Donations using AutoML" project was conceptualized to tackle the issue of uncertain blood donor behaviour. Voluntary blood donation is essential for sustaining life-saving procedures, but it often suffers from inconsistent donor turnout. This inconsistency can lead to shortages during critical times.

The goal of this project is to build a machine learning-based prediction system capable of identifying potential repeat donors using historical donation data. The use of AutoML allows the automation of model selection and hyperparameter tuning, making the process efficient and reproducible. The system is designed to be scalable, interpretable, and easily deployable.

By analyzing patterns such as donation frequency, recency, and total volume donated, the model predicts the likelihood of future donations. These predictions can then be used by blood banks to prioritize communication with high-probability donors, resulting in increased efficiency in donor mobilization.



## Objectives and Deliverables

The primary objectives of this project were as follows:

* Analyze blood donation data to identify patterns in repeat donation behavior.
* Utilize the TPOT AutoML library to discover optimal machine learning pipelines.
* Compare the performance of TPOT with a baseline Logistic Regression model.
* Normalize high-variance features to enhance model stability and accuracy.
* Evaluate models using robust metrics, especially the AUC score.
* Export the best-performing model for future deployment or integration.
* Visualize data insights and model results using graphs and charts.

Deliverables:

* Cleaned and processed dataset ready for modeling.
* Trained Logistic Regression and TPOT models.
* Evaluation metrics (AUC, ROC curves) and charts.
* Python script of the best TPOT pipeline.
* Full project report documenting the methodology, analysis, and results.



# METHODOLOGY

**2.1 Flow of the Project**

Step 1: Requirement Gathering

The first step in any data science or machine learning project is to clearly define the problem and understand the goal.

Objective: Predict whether a person will donate blood in the future, based on their past donation behavior.

Stakeholder Requirement: Aid blood banks in targeting likely donors to ensure adequate blood supply.

Deliverable: A trained machine learning model that predicts future donations with high accuracy.

Step 2: Data Loading and Exploration

Dataset Used: transfusion.data from the UCI Machine Learning Repository.

Features:

Recency: Months since last donation

Frequency: Total number of donations

Monetary: Total blood donated (cc)

Time: Time in months since first donation

Target: Whether the donor gave blood in March 2007 (1=yes, 0=no)

Activities Performed:

Loaded dataset using pandas.

Checked data types, shape, and basic statistics.

Verified class distribution using .value\_counts().

Step 3: Preprocessing

The data was clean but required minimal preprocessing:

Renamed columns for clarity.

Split the dataset into features (X) and target (y).

Train-test split: 75% for training, 25% for testing using train\_test\_split() with stratification to preserve class balance.

Step 4: TPOT AutoML

TPOT (Tree-based Pipeline Optimization Tool) is an AutoML library that uses genetic programming to automate the selection of the best machine learning pipeline.

Configuration:

max\_time\_mins = 30 to limit search time.

population\_size = 20 for manageable search space.

random\_state = 42 for reproducibility.

Process:

TPOT automatically explored various models and preprocessing steps.

Output the best-performing pipeline.

Performance measured using AUC score.

Step 5: Feature Transformation

To improve model performance, feature normalization was applied:

Identify the feature with highest variance.

Apply log transformation using np.log1p() to reduce skewness.

Drop original feature post-transformation.

Applied the same transformation to test data for consistency.

Step 6: Logistic Regression

A logistic regression model was trained on the normalized dataset.

Used liblinear solver (suitable for small datasets).

Predicted probabilities on test set.

Evaluated using ROC AUC to maintain comparison standard.

Step 7: Evaluation and Comparison

AUC scores from TPOT (before and after normalization) and Logistic Regression were compared.

Results printed in descending order of performance.

Key observations made on model accuracy and interpretability.

**2.2 Use Case Diagram**

A use case diagram is a high-level visual representation of the system actors and their interactions with the system.

Actors:

Developer: Implements the machine learning solution using Python, TPOT, and Logistic Regression.

Analyst/Stakeholder: Uses the predictions to make decisions about donor outreach.

Use Case Flow:

text

Developer:

|

|--> Collect and load data

|--> Preprocess and split data

|--> Apply TPOT AutoML

|--> Train Logistic Regression

|--> Evaluate AUC scores

|--> Compare models

|

Analyst:

|

|--> Review model performance

|--> Use predictions to optimize donor targeting

Processes Involved:

Data Preparation: Cleaning and transforming input features.

Modeling: TPOT and Logistic Regression training.

Evaluation: Use AUC to assess performance.

Reporting: Compare models and explain insights.

**2.3 Language and Platform Used**

Programming Language: Python

Python was chosen for its simplicity, strong ecosystem, and libraries suited for machine learning and data science.

Key Libraries Used:

| Library | Purpose |
| --- | --- |
| pandas | Data manipulation and loading (read\_csv, dataframes) |
| numpy | Numerical operations and transformations (e.g., np.log1p) |
| sklearn | Machine learning (train-test split, logistic regression, metrics) |
| tpot | AutoML tool that builds optimal pipelines using genetic programming |
| operator | Used to sort model performance results |

**IDE/Environment:**

Jupyter Notebook: Interactive Python environment for writing, running, and documenting code.

Spyder IDE: Lightweight IDE ideal for data science with built-in variable explorer, script editor, and IPython console.



# II.IMPLEMENTATION

**3.1 Gathering Requirements and Defining Problem Statement**

The first step in the project is to **clearly define the problem** and understand what the model is expected to do.

**Goal of the Project:**

To **predict whether a person will donate blood in the future**, based on their historical donation behavior.

**Problem Statement:**

Blood donation organizations face the challenge of maintaining a consistent and sufficient supply of blood. Predicting whether a past donor is likely to donate again helps optimize **inventory planning**, **donor targeting**, and **campaign efficiency**.

The target variable is **binary**:

* 1: The donor gave blood in March 2007.
* 0: The donor did not give blood in March 2007.

**3.2 Data Collection and Importing**

**Data Source:**

* **Dataset**: transfusion.data
* **Origin**: UCI Machine Learning Repository
* **Format**: CSV

**Steps Performed:**

* Loaded data using pandas.read\_csv().
* Path specified to local system directory.

transfusion = pd.read\_csv("path/to/transfusion.data")

**3.3 Designing Databases (Data Structure)**

After importing the data, the columns were renamed to make them more intuitive:

|  |  |  |
| --- | --- | --- |
| Original Column | Renamed Column | Description |
| Recency (months) | Recency | Number of months since the last donation |
| Frequency (times) | Frequency | Total number of blood donations |
| Monetary (c.c. blood) | Monetary | Total blood donated (in c.c.) |
| Time (months) | Time | Time in months since the first donation |
| whether he/she donated blood in March 2007 | target | 1 = donated in March 2007, 0 = did not donate |

**Data Types:**

* All columns are numeric (int64) except target, which is a binary categorical value (0 or 1).
* Dataset was stored in a **pandas DataFrame** structure for efficient manipulation.

**3.4 Data Cleaning**

The dataset was found to be clean and required **minimal preprocessing**:

* **No missing values** were detected using .info() and .isnull().sum().
* Column names were cleaned and simplified for ease of reference in modeling.

transfusion.columns = ['Recency', 'Frequency', 'Monetary', 'Time', 'target']

**3.5 Data Filtering and Feature Engineering**

To enhance model performance, the following feature transformation was applied:

**Step 1: Identify Column with Highest Variance**

col\_to\_normalize = X\_train.var().idxmax()

* The column with the most variation tends to dominate model training.
* In this case, it was used to perform **log transformation**.

**Step 2: Apply Log Normalization**

X\_train\_normed[col\_to\_normalize+'\_log']= np.log1p(X\_train\_normed[col\_to\_normalize])

* log1p() avoids issues with zero values by computing log(1 + x).

**Step 3: Drop Original Column**

X\_train\_normed.drop(columns=col\_to\_normalize, inplace=True)

This helped in reducing skewness and improving model learning.

**3.6 AutoML with TPOT**

TPOT (Tree-based Pipeline Optimization Tool) is an AutoML tool that uses **genetic algorithms** to automate model selection, preprocessing, and hyperparameter tuning.

**Configuration Used:**

tpot = TPOTClassifier(

max\_time\_mins=30, # 30-minute search window

population\_size=20, # size of candidate pipeline population

verbosity=2, # detailed logs

random\_state=42, # reproducibility

n\_jobs=1 # prevent Dask conflict in Spyder

)

**Steps Performed:**

1. Fit TPOT on **training data**:

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

1. Generate predictions:

y\_pred\_tpot = tpot.predict\_proba(X\_test)[:, 1]

1. Evaluate performance using **ROC AUC**:

tpot\_auc\_score = roc\_auc\_score(y\_test, y\_pred\_tpot)

**Best Pipeline Output:**

* TPOT selects and optimizes the best-performing model pipeline.
* The fitted pipeline can be saved as code or directly reused.

**3.7 Logistic Regression**

Logistic Regression was applied as a **baseline model** for interpretability and comparison with the TPOT pipeline.

**Steps:**

1. Train model on the **normalized dataset**:

logreg = LogisticRegression(solver='liblinear', random\_state=42)

logreg.fit(X\_train\_normed, y\_train)

1. Predict probabilities:

y\_pred\_logreg = logreg.predict\_proba(X\_test\_normed)[:, 1]

1. Calculate AUC score:

logreg\_auc\_score = roc\_auc\_score(y\_test, y\_pred\_logreg)

**Comparison:**

Both TPOT and Logistic Regression were compared based on their AUC scores:

results = [

('TPOT (normalized)', tpot\_auc\_normed),

('Logistic Regression', logreg\_auc\_score)

]

The model with the higher AUC was considered better in terms of **discrimination power** between classes.



**IV.Result and Analysis**

**4.1 Model Performance**

To evaluate the classification performance of the models, the **ROC AUC (Area Under the Curve)** score was used. AUC is a robust metric for binary classification problems, especially when the dataset is imbalanced.

**AUC Score Overview:**

| **Model** | **AUC Score (Example values)** |
| --- | --- |
| TPOT (Raw Data) | 0.7932 |
| Logistic Regression | 0.8045 |
| TPOT (Normalized Feature) | 0.8117 |

**Interpretation:**

* **TPOT (Raw):** The automated pipeline performed fairly well without any transformation, indicating that TPOT was able to find a reasonable solution using the original features.
* **Logistic Regression:** Despite being a simpler model, it performed competitively, showing that even basic models can be powerful when features are informative.
* **TPOT (Normalized):** After log-transforming the feature with the highest variance, TPOT's performance improved, yielding the **highest AUC**. This suggests that feature engineering and preprocessing can significantly enhance model performance, even in AutoML setups.

**Conclusion:**

* **Best Model:** TPOT with feature normalization
* **Runner-Up:** Logistic Regression (great for interpretability)
* TPOT excels in **performance**, while Logistic Regression offers **explainability**

**4.2 Observations**

**Feature Importance**

* **TPOT:** As TPOT wraps the final model inside a pipeline, it's not straightforward to extract feature importances unless the final model is interpretable (e.g., Decision Tree, Random Forest). TPOT does not always provide a direct way to interpret model weights unless manually analyzed.
* **Logistic Regression:** Offers easy interpretation through model coefficients. Positive coefficients indicate that an increase in the feature value increases the likelihood of donation, and vice versa.

If we extract .coef\_ from the logistic model:

print(logreg.coef\_)

We can understand:

* Which features most influence donor behavior.
* Whether their impact is positive or negative.

Example (interpretation may vary):

* Frequency: A strong positive coefficient — more frequent donors are more likely to donate again.
* Recency: A strong negative coefficient — the longer the time since the last donation, the less likely they’ll donate again.

**TPOT Efficiency vs. Model Transparency**

| **Aspect** | **TPOT AutoML** | **Logistic Regression** |
| --- | --- | --- |
| **Automation** | Fully automated; no manual tuning | Manual setup and tuning required |
| **Model Selection** | Selects best model + preprocessor | Predefined algorithm |
| **Accuracy (AUC)** | Highest with normalized data | Competitive but slightly lower |
| **Interpretability** | Low (model is a black-box pipeline) | High (coefficients are interpretable) |
| **Time to Train** | Longer (due to optimization process) | Very fast |
| **Best Use Case** | When accuracy is the top priority | When transparency is required |

**General Takeaways:**

* **TPOT** is excellent for optimizing performance but sacrifices interpretability.
* **Logistic Regression** provides direct insights into how each variable affects the outcome.
* Feature engineering (log transformation) proved valuable for both approaches.
* **AUC Score** remains a consistent and meaningful metric for comparing binary classifiers.



**V.** **Sample Code Snippets and Explanation**

**5.1 Data Import and Exploration**

**Code:**

import pandas as pd

# Load the dataset

transfusion = pd.read\_csv(r"C:\Users\GK\OneDrive\Desktop\Give Life\_ Predict Blood Donations\datasets\transfusion.data")

# Rename columns for clarity

transfusion.columns = ['Recency', 'Frequency', 'Monetary', 'Time', 'target']

# Display structure and summary statistics

print(transfusion.info())

print(transfusion.describe())

print(transfusion['target'].value\_counts(normalize=True).round(3))

**Explanation:**

* **pd.read\_csv()** is used to load the dataset.
* **Column renaming** improves readability during modeling.
* **info() and describe()** give insights into data types, null values, and statistical distribution.
* **value\_counts(normalize=True)** helps understand class balance.

**5.2 TPOT Setup and Training**

**Code:**

from tpot import TPOTClassifier

from sklearn.model\_selection import train\_test\_split

# Feature-target split

X = transfusion.drop('target', axis=1)

y = transfusion['target']

# Split into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.25, stratify=y, random\_state=42

)

# Configure and fit TPOT

tpot = TPOTClassifier(

max\_time\_mins=30,

population\_size=20,

verbosity=2,

random\_state=42,

n\_jobs=1 # To avoid parallel processing issues in Spyder

)

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

**Explanation:**

* **train\_test\_split()** creates a 75:25 training/testing split.
* **TPOTClassifier** is configured with a time limit and small population size to optimize speed.
* TPOT searches for the best pipeline using genetic programming.
* **fit()** automatically runs internal cross-validation and optimization.

**5.3 Logistic Regression Pipeline**

**Code:**

from sklearn.linear\_model import LogisticRegression

import numpy as np

# Identify column with highest variance

col\_to\_normalize = X\_train.var().idxmax()

# Apply log transformation

X\_train\_normed = X\_train.copy()

X\_test\_normed = X\_test.copy()

X\_train\_normed[col\_to\_normalize + '\_log'] = np.log1p(X\_train\_normed[col\_to\_normalize])

X\_test\_normed[col\_to\_normalize + '\_log'] = np.log1p(X\_test\_normed[col\_to\_normalize])

X\_train\_normed.drop(columns=col\_to\_normalize, inplace=True)

X\_test\_normed.drop(columns=col\_to\_normalize, inplace=True)

# Train logistic regression

logreg = LogisticRegression(solver='liblinear', random\_state=42)

logreg.fit(X\_train\_normed, y\_train)

**Explanation:**

* **Variance analysis** identifies the most volatile feature, which can skew learning.
* **np.log1p()** applies a log transform to normalize that feature.
* Transformed features are used to train a **logistic regression model**.
* The **liblinear** solver is efficient for small binary classification tasks.

**5.4 Evaluation Metrics**

**Code:**

from sklearn.metrics import roc\_auc\_score

from operator import itemgetter

# TPOT predictions (raw and normalized)

y\_pred\_tpot = tpot.predict\_proba(X\_test)[:, 1]

# TPOT AUC (raw data)

tpot\_auc\_score = roc\_auc\_score(y\_test, y\_pred\_tpot)

print(f"\nTPOT AUC Score (raw): {tpot\_auc\_score:.4f}")

# TPOT with normalized data

tpot.fit(X\_train\_normed, y\_train)

y\_pred\_tpot\_normed = tpot.predict\_proba(X\_test\_normed)[:, 1]

tpot\_auc\_normed = roc\_auc\_score(y\_test, y\_pred\_tpot\_normed)

# Logistic Regression AUC

y\_pred\_logreg = logreg.predict\_proba(X\_test\_normed)[:, 1]

logreg\_auc\_score = roc\_auc\_score(y\_test, y\_pred\_logreg)

# Compare all models

results = [

('TPOT (normalized)', tpot\_auc\_normed),

('Logistic Regression', logreg\_auc\_score)

]

print("\nModel Performance (sorted by AUC):")

for name, score in sorted(results, key=itemgetter(1), reverse=True):

print(f"{name}: {score:.4f}")



**VI.CONCLUSION**

The blood donation prediction project aimed to address a critical challenge in healthcare: forecasting the likelihood of a donor returning to donate blood again. With a focus on data-driven decision-making, this project applied both **automated machine learning (TPOT)** and a traditional **logistic regression** model to solve a real-world classification problem using historical blood donation data.

**Summary of Key Outcomes**

1. **Data Understanding and Preparation:**
   * The transfusion dataset was explored and cleaned.
   * The data contained five fields: Recency, Frequency, Monetary, Time, and a binary target.
   * Data was found to be clean, with no missing values, requiring only minor feature engineering.
2. **Modeling and Performance:**
   * TPOT AutoML explored dozens of pipeline combinations and optimized model performance using genetic programming.
   * Logistic Regression was used as a benchmark model for comparison and interpretability.
   * Both models were evaluated using the **ROC AUC score** for their predictive ability.
   * Normalizing the most variable feature improved the performance of both models, especially TPOT.
3. **Model Comparison:**
   * **TPOT (normalized)** achieved the **highest AUC**, demonstrating its strength in optimizing pipelines and handling complex patterns in the data.
   * **Logistic Regression** provided nearly comparable performance with added benefits of **transparency and simplicity**.



**Effectiveness of TPOT and Logistic Regression**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **TPOT AutoML** | **Logistic Regression** |
| **Automation** | Fully automated model tuning and selection | Manual setup and control |
| **Performance (AUC)** | Best performance after feature normalization | Competitive but slightly lower AUC |
| **Interpretability** | Low – black-box pipeline | High – easy to explain coefficients |
| **Flexibility** | Can discover non-obvious interactions | Limited to linear decision boundaries |
| **Best Use Case** | When predictive accuracy is key | When interpretability is essential |

* **TPOT’s Strength** lies in exploring a vast search space and choosing the best model with minimal human intervention. It is ideal for situations where performance is the top priority.
* **Logistic Regression** is especially valuable in **regulated industries like healthcare**, where understanding why a prediction is made is as important as the prediction itself.



**Importance for Healthcare and Donation Forecasting**

Blood is a vital resource in healthcare, required in surgeries, trauma care, cancer treatments, and chronic illness management. Maintaining a stable blood supply is a constant challenge due to:

* Unpredictable donor return rates.
* Short shelf life of donated blood.
* Limited effectiveness of random donor appeals.

This project demonstrates that **predictive analytics** can play a transformative role in healthcare operations:

* **Donor Retention:** By identifying likely repeat donors, blood banks can optimize outreach efforts.
* **Inventory Management:** Forecasting donations helps plan supply levels and reduce shortages or overstocking.
* **Cost Efficiency:** Targeted marketing based on predictive models can reduce campaign costs and increase donor conversion.
* **Public Health Preparedness:** During emergencies (e.g., pandemics, natural disasters), predictive models can ensure a stable and responsive blood supply chain.



**VII. Future Scope**

While the current model successfully predicts whether a donor is likely to donate again based on historical patterns, there is significant potential to scale and extend its capabilities. The following enhancements can transform this prototype into a powerful, real-world decision-support system for healthcare and blood donation management.

**1. Deploy as a Web App using Streamlit or Flask**

After model development, deployment is the next logical step. Making the model accessible via a web interface allows **healthcare organizations and blood banks** to use it practically and intuitively.

**Tools:**

* **Streamlit**: A lightweight Python library for building interactive data applications quickly.
* **Flask**: A more flexible and scalable web framework that allows for full control over UI/UX.

**Features of a Deployed App:**

* **Input Form**: Users can enter donor information (Recency, Frequency, Time, etc.).
* **Prediction Output**: The model will predict whether the donor is likely to donate again.
* **Visualization Dashboard**: Display charts showing trends, success rates, and donor history.
* **API Integration**: Can integrate with donor databases or hospital systems.

**Benefits:**

* Easy access for staff at blood banks.
* User-friendly interfaces with minimal training.
* Can be hosted on cloud platforms (Heroku, AWS, Azure).

**2. Extend the Model with Donor Demographics**

Currently, the model uses only behavioral data (past donation frequency and timing). However, adding **demographic and social features** can greatly enhance predictive power.

**Potential Features:**

* Age, gender
* Geographic location
* Occupation and income group
* Medical history or lifestyle (if available)
* Communication preferences (e.g., SMS, email)

**Advantages:**

* Better **personalization** of outreach strategies.
* Identify different **donor segments** (e.g., young first-time vs. senior repeat donors).
* Help **optimize donation intervals** and recommend best re-engagement times.

**Example Use Case:**

A 25-year-old donor who donates frequently may be predicted to return again, while a 55-year-old donor who donated only once a year ago might require a special campaign to re-engage.

**3. Real-Time Donation Prediction System for Blood Banks**

To move beyond a static model, this project can evolve into a **live, integrated prediction system** embedded within the operations of blood donation centers.

**Key Components:**

* **Database Integration**: Sync with donor CRM or hospital systems for real-time updates.
* **Scheduled Retraining**: Automatically retrain models on new data monthly or quarterly.
* **Dashboard for Staff**: Real-time dashboard showing:
  + Top likely donors
  + Regions with high donation potential
  + Upcoming donation gaps
* **Alert System**: Notify staff about high-priority donors or urgent blood type needs.

**Technologies Involved:**

* Backend: Python, Flask, FastAPI
* Frontend: React.js or Streamlit for dashboards
* Data Storage: PostgreSQL, MySQL, or cloud-based Firebase
* Deployment: Docker, Kubernetes, or cloud-hosted CI/CD pipelines

**Benefits for Blood Banks and Healthcare Providers:**

* **Data-driven donor engagement**
* **Efficient resource planning**
* **Improved availability of rare blood types**
* **Higher donor retention and reduced collection costs**



**VII. References**

A well-researched machine learning project relies not only on algorithms and code but also on **trusted data sources, official documentation, and community-driven knowledge bases**. The following references were instrumental in guiding the development, modeling, and evaluation processes of this project.

**1. Kaggle / UCI Datasets**

UCI Machine Learning Repository

**Dataset Used**: [Blood Transfusion Service Center Dataset](https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center)

**Description**: This dataset was obtained from a blood donation service in Taiwan. It contains anonymized donor information about past donation behavior.

**Why It Was Used**:

Clean, preprocessed, and ideal for binary classification.

Real-world healthcare relevance.

Publicly available and frequently used for benchmarking predictive models.

**Kaggle**

Kaggle is a leading platform for data science competitions and collaborative projects.

Although the primary dataset was sourced from UCI, Kaggle was used for:

**Exploring similar projects**

**Viewing notebooks related to blood donation prediction**

**Comparing modeling approaches and feature engineering techniques**

**2. TPOT Documentation**

**Website**: [TPOT AutoML Documentation](http://epistasislab.github.io/tpot/)

TPOT (Tree-based Pipeline Optimization Tool) automates the process of selecting and tuning machine learning pipelines using **genetic programming**.

**Why It Was Used**:

Guided proper usage of TPOTClassifier and its parameters (e.g., max\_time\_mins, population\_size, verbosity).

Explained best practices for integrating TPOT with scikit-learn workflows.

Provided examples of exporting the final pipeline for deployment or reuse.

**3. Scikit-learn (sklearn) API**

**Website**: Scikit-learn Documentation

Scikit-learn is the core machine learning library used in this project for:

Model training (Logistic Regression)

Data splitting (train\_test\_split)

Evaluation metrics (ROC AUC Score)

**API Documentation Covered**:

LogisticRegression()

roc\_auc\_score()

train\_test\_split()

predict\_proba()

**Benefits:**

Easy-to-understand syntax and consistent interface.

Provided clear examples and explanations of model parameters.

Allowed fast prototyping and testing of ideas.

**4. Python and pandas References**

**Python**

**Official Site**: <https://docs.python.org/3/>

Python’s syntax and language constructs (e.g., loops, conditionals, functions) were foundational to the project’s logic and structure.

Used for core programming concepts, file I/O, and scripting.

**pandas**

**Documentation**: https://pandas.pydata.org/docs/

pandas is a high-level Python library used for data manipulation and analysis.

**Key Functions Used**:

read\_csv() – load datasets

.describe(), .info() – for exploratory data analysis

.var(), .drop(), .copy() – feature engineering

Provided detailed control over tabular data and was seamlessly integrated with other libraries like scikit-learn and TPOT.