BLOOD DONATION PREDICTION

Project Report Submitted

In Partial Fulfillment of the Requirements

For the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

Submitted By

Shahela Qurrath (1604-17-737-010)
Syeda Ruqhaiya Fatima (1604-17-737-066)
Arjumand Sakina(1604-17-737-074)

INFORMATION TECHNOLOGY DEPARTMENT MUFFAKHAM JAH COLLEGE OF ENGINEERING & TECHNOLOGY
(Affiliated to Osmania University)
Mount Pleasant, 8-2-249, Road No. 3, Banjara Hills, Hyderabad-34
2020-21

CERTIFICATE

It is certified that the work contained in the project report titled "**Blood Donation System**" by

Shahela Qurrath (1604-17-737-010)

Syeda Ruqhaiya Fatima (1604-17-737-066)

Arjumand Sakina(1604-17-737-074)

has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

Project Supervisor Signature of Head **Mrs. MUNNAWARA TAHSEEN** **Dr. MOUSAMI AJAY CHARASIA** Information Technology Dept.
Information Technology Dept. MJ College of Engineering & Tech.
MJ College of Engineering & Tech Hyderabad – 500 034.
Hyderabad – 500 034.

External Examiner

DECLARATION

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Shahela Qurrath (1604-17-737-010)

Syeda Ruqhaiya Fatima (1604-17-737-066)

Arjumand Sakina(1604-17-737-074)

ACKNOWLEDGEMENT

We would like to take the opportunity to express our sincere thanks to our guide Mrs. Munawara Tahseen, Assistant Professor , Department of Information Technology, MJ College of Engineering & Technology, for her invaluable support and guidance throughout our project work. Without her kind guidance & support this was not possible. We are grateful to her for her timely feedback which helped us track and schedule the process effectively. Her time, ideas and encouragement that she gave helped us complete our project efficiently.

We would also like to thank Mr. Mohammed Afroze,Assistant Professor, and Dr Mousami Ajay Chaurasia, Head, Department of Information Technology, MJ College of Engineering & Technology for his encouragement and for providing an outstanding academic environment, also for providing the adequate facilities resources.

We also extend our sincere thanks to all the faculty members, non-teaching staff and friends for their cooperation. Last but not the least, we are thankful to all our family members whose constant support and encouragement in every aspect helped us to complete our project.

Shahela Qurrath (1604-17-737-010)

Syeda Ruqhaiya Fatima (1604-17-737-066)

Arjumand Sakina(1604-17-737-074)

ABSTRACT

Blood is the most precious gift that anyone can give to someone — the gift of life. A decision to donate your blood can save a life, or even several if your blood is separated into its components — red cells, platelets, and plasma.

The donation of blood is very important because most often people needing blood do not receive it on time causing deaths. Patients suffering from malaria or organ transplants, extreme health conditions such as Leukemia and bone marrow cancer, experience sudden high blood loss and need an urgent blood supply and not providing it can lead to loss of life.

One of the exciting features of blood is that it is not a characteristic product. Blood has a life of approximately 42 days. For example, platelets must be stored around 22 degrees Celsius. Moreover, platelets can often be stored for at most 5 days, red blood cells up to 42 days, and plasma up to one year. Amazingly, only around 5% of the eligible donor population actually donate. This low percentage highlights the risk humans are faced today.

There is a time lag between the demand for blood required by patients suffering extreme blood loss and the supply of blood from blood banks. We try to improve this supply-demand lag by building a predictive model that helps identify the potential donors.

In our study, we focus on building a data-driven system for predicting potential blood donors. We investigate the use of various binary classification techniques to estimate the probability that a person will donate blood in this year or not based on his past donation behavior.

TABLE OF CONTENTS

LIST OF FIGURES

List Of Tables

1. INTRODUCTION

1.1 Overview

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly.

Classification is one of the most fundamental concepts in data science. Classification algorithms are predictive calculations used to assign data to preset categories by analyzing sets of training data. Classification is the process of recognizing, understanding, and grouping ideas and objects into preset categories or “sub-populations.” Using pre-categorized training datasets, machine learning programs use a variety of algorithms to classify future datasets into categories. Classification algorithms in machine learning use input training data to predict the likelihood that subsequent data will fall into one of the predetermined categories. In short, a classification is a form of “pattern recognition,” with classification algorithms applied to the training data to find the same pattern (similar words or sentiments, number sequences, etc.) in future sets of data.

In our project, we test different classification algorithms against each other and choose the one that is best for our problem statement. Models like Logistic Regression, Adaboost Classifier, Stochastic Gradient Descent, K-Nearest Neighbours, Decision Tree, Random Forest, Support Vector Machine, etc.

The main purpose of this project is to make a prediction model that would generate potential donors based on their past donation patterns. Of all the parameters present in our dataset, we have chosen only those who majorly affect the results. Through simple data visualization techniques we have found great insights into the dataset and relations between all the features. Data Visualization helps in determining the good and bad features that will influence the final predictions. Based on these helpful insights, we remove less important features from the dataset. Then we run cross-validation on various algorithms, from the statistics obtained we chose a few algorithms and trained our model with them. The algorithm with the highest accuracy is tested against the test dataset to generate probabilities. The probability is in the range of 0-1. Where 0-0.5 indicates that the probability of a person donating is low and 0.5-1 denotes that the person is more likely to donate.

1.2 Problem Statement

Blood Transfusion plays a vital role in saving a human life. Every day there are urgent blood requirements for critical patients in need of blood components. The major problem faced by the medical sector is the lack of blood and blood donor availability. And the main factor here is the shelf life of blood which is only 35 days from the date of donation.

The primary solution is to find healthy volunteers who donate regularly, but only a fraction of the population donates on a regular basis. This is why the demand for blood is sky-rocketing. The donors also need to take an 8-16 week break before donating again. This is why finding donors is a tedious task. There needs to be a system that could analyze donors' information and retrieve donors based on their donation history.

1.3 Objectives

1. Build a supervised machine learning model that predicts the probabilities based on well-defined influential parameters.
2. Test different classification models to find the most suitable one for our problem statement.
3. Use the final models to predict the chances or the probability of a person who may or may not donate blood.

1.4 Organization of Report

The rest of the report is laid out as follows:

Chapter 2- Literature Survey presents an extensive review of the existing methods and technology applied along with the parameters involved.

Chapter 3- System Analysis is an overview of the prediction process and the proposed solution of our project along with all the technical specifications.

Chapter 4- System Design is a general framework of the proposed method which is followed by a conclusion, future scope, and references section, also including an appendix at the end of the report.

2. LITERATURE SURVEY

2.1 Related Research

Analyzing Blood Donation Probabilities and Numbers Of Possible Donors:

In the paper, many critical data of donors were used. A donor's blood donation frequency and the last donation time were included in the utilized data. The utilization of these data types was very important for providing a solution to determine blood donation probabilities. By using many machine learning approaches on blood transfusion data, it is tried to be estimated, if a possible donor will provide blood donation again. Used algorithms were compared by computing their classification performances.

Machine learning algorithms use statistics to find patterns in large-scale data. These algorithms complete the task of learning by using the available data and giving the machine experiences as in humans. When new data is added, the machine continues to learn and improves without external programming. In this study, a structured analytical methodology called the Cross-industry Standard process for data mining (CRISPDM), which is widely known in the data mining community is followed.[1]

Find eligible blood donors using Decision Tree and Naive Bayes Classifier.

This work proposed a classification model to decrease the time process using both the decision tree and naive Bayes classifier. In the evaluation phase, both algorithms will compare by its accuracy and performance. The aim is to create an interconnected system by merging different blood donation management apps and platforms. While using data in real-time and from various different sources all around the world. The only drawback of this is that the decision tree may neglect key points in training data. And small changes in data can lead to inaccuracies.

In the final stage, the method compares the value of (P | eligible) and (P | ineligible). The current data training is classified into eligible classes because the eligible class has a greater value than the ineligible classes. [2]

2.1.1 mHealth: Blood donation application using android smartphone.

mHealth is a new horizon for health that offers healthcare services by utilizing mobile devices and communication technologies. In health care services, blood donation is a complex process and consumes time to find a donor who has the compatibility of blood group with the patient. An android-based blood donation application as mHealth solutions to establish a connection between the requester and donor at any time and anywhere.

The objective of this application is to provide information about the requested blood and the number of available donors around those localities. It assists the requester to broadcast the message across the maintained volunteer blood donor network by our application and update the requester at the same time who is willing to donate the requested blood. To evaluate our application, we created requester-donor profiles and analyzed that it will help to improve the timely access of the information and rapid response in emergency situations.[3]

2.1.2 Factors That Affects Blood Donation Cycle:

Blood has a perishable nature. According to the American Red Cross Organization, the shelf life of blood is 42 days. For this reason, it becomes difficult to find an adequate number of blood donors in situations that may increase the need for blood. The primary way to meet the blood transfusion needs of people is to receive regular donations from healthy volunteers. Unfortunately, only 5% of the eligible donor population regularly donates. By following the blood supply requirements and provided donations, a model can be presented to estimate the potential donors. Also, by improving the supply chain, the death risk of many people may be decreased with providing them to reach the needed blood.

In our project, we explore all other models to find if there’s a better model or approach for this particular problem statement. We are testing all the classification algorithms on our project’s dataset and the one that gives good accuracy and a learning curve becomes the final algorithm or model for the project. This model will then generate the most accurate probabilities for our test set.

2.1.3 Classification algorithms

a) **Decision tree**: Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity.

b) **Random forest**: Random forest classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

c) **Logistic regression**: In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object is detected in the image would be assigned a probability between 0 and 1, with a sum of one.

d) **Gradient boosting:** Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When a decision tree is a weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms random forest. It builds the model in a stage-wise fashion as other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function

e) **Support-Vector Machines:** In machine learning, support vector machines (SVMs, also support-vector networks are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. SVM maps training examples to points in space so as to maximize the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

2.2 Technology Used

2.2.1 Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one— and preferably only one — obvious way to do it" design philosophy.

Python's developers strive to avoid premature optimization and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler

An important goal of Python's developers is to keep it fun to use. This is reflected in the language's name—a tribute to the British comedy group Monty Python—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar

2.2.2 Anaconda

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface (CLI).

In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g. the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

2.2.3 Jupyter Notebook

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots, and rich media, usually ending with the ".ipynb" extension.

To simplify the visualization of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

2.2.4 Support Vector Machine

Support Vector Machine(SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

In Python, scikit-learn is a widely used library for implementing machine learning algorithms. SVM is also available in the scikit-learn library and we follow the same structure for using it(Import library, object creation, fitting model, and prediction)

A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the tags. This line is the decision boundary: anything that falls to one side of it we will classify as blue, and anything that falls to the other as red.

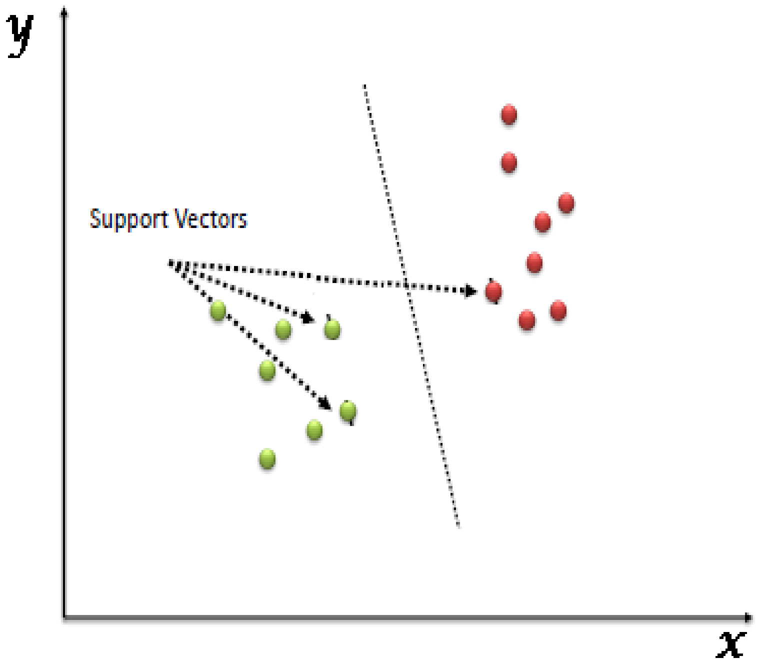


Fig 2.2.3.1 SVM Classification

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points .To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

Support Vector Machine gives the best results compared to other classification models for our problem statement. A basic UI could be developed that takes real-time datasets from blood banks and allows healthcare professionals to find donors. Factors like geographical data could be added so that donors in the nearest distance could be found.

3. SYSTEM ANALYSIS

3.1 Problems with Existing System

The existing system was very time-consuming and there was no proper way of knowing which blood group is present and in how much quantity. Suppose there is some patient who needs blood urgently, people will go from one blood bank to another to get the blood which is time-consuming and sometimes could not find the right donors.

In some blood banks, sometimes we get the blood but that is not sufficient so we need to search for another blood bank for more blood. If some person wants to donate the blood he/she needs to come to the bank and fill the form then the first doctor will check his blood group after that he will allow donating.

Disadvantages:

1. It is a time-consuming process.
2. The certainty of the information is not known.
3. The history of donations is not accessible to common people.
4. Loss of several lives on a daily basis due to less reliable resources.

3.2 Proposed system:

1. To build a machine learning model using the best-suited algorithm that predicts potential blood donors.
2. Data from various trusted Blood Bank or websites can be taken to find donors according to their donation history.
3. Using this system, Healthcare professionals can fetch the information of donors hassle-free.

3.3 Feasibility Study

The preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational, and Economical feasibility for adding new modules and debugging old running systems. All systems are feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

• Technical Feasibility
• Economical Feasibility
• Operation Feasibility

3.3.1. Technical Feasibility:

A Technical feasibility study assesses the details of how you intend to deliver a product or service to customers. It is the process of validating the technology assumptions, architecture and design of a product or project. The main technologies and tools that are associated are:

- Python
- Anaconda
- Jupyter Notebook

Each of the technologies is freely available and the technical skills required are manageable. Time limitations of the model development and the ease of implementing using these technologies are synchronized. The proposed system consists of free-to-use software and it is platform-independent. It can be used with different software versions by only doing minor changes to the code and has some different execution steps involved.

From there it’s clear that this project is technically feasible.

3.3.2. Economical Feasibility:

Economic feasibility is the cost and logistical outlook for a business project or endeavor. It is a kind of cost-benefit analysis of the examined project, which assesses whether it is possible to implement it or not. Being an ML model, the project will not have any cost.

The system will follow the freeware software standards. No cost will be charged from the potential consumers. At the initial stage, the potential market space will be the local hospitals and blood banks.

This project uses Python libraries like Pandas, NumPy, matplotlib, sklearn, etc using Anaconda. Python is a free, open-source programming language that is available for everyone to use. We have used the Jupyter notebook for the graphical representation of algorithms and to find the accuracy of each individually.

From there it’s clear that the project is economically feasible.

3.3.3. Operational Feasibility:

Operational feasibility is a measure of how well a proposed system solves the problems. This system operates based on the various features of the dataset from UCI repository, and using the best algorithms accurately finds the potential donors and the chances of them donating based on features. This needs proper and valid data from trusted blood banks, with that kind of data it is highly feasible.

3.4 Effort, Duration and Cost Estimation using COCOMO & Function Point.

COCOMO(Constructive Cost Estimation Model) is one of the most generally used software estimation models in the world. COCOMO predicts the efforts and schedule of a software product based on the size of the software.

In COCOMO, projects are categorized into three types:

1. Organic
2. Semi-Detached
3. Embedded

ORGANIC: Relatively small, simple software projects in which small teams with good application experience work to a set of less than rigid requirements.

SEMI-DETACHED: An intermediate, (in size and complexity), a software project in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements.

EMBEDDED: A software project that must be developed within a set of tight hardware, software and operation constraints.

Basic COCOMO Model:

The basic COCOMO model provides an accurate size of the project parameters. The following expressions give the basic COCOMO estimation model:

$$\text{Effort}=a_1*(\text{KLOC})\ a_2\ \text{PM}$$

$$\text{Tdev}=b_1*(\text{efforts})b_2\ \text{Months}$$

$$\text{Staffing} = \text{effort}/\text{duration}$$

Where,

KLOC is the estimated size of the software product indicate in Kilo Lines of Code,

a_1,a_2,b_1,b_2 are constants for each group of software products,

Tdev is the estimated time to develop the software, expressed in months,

Effort is the total effort required to develop the software product, expressed in person months (PMs).

Estimation of development effort

For the three classes of software products, the formulas for estimating the effort based on the code size are shown below:

Organic: Effort = 2.4(KLOC) 1.05 PM

Semi-detached: Effort = 3.0(KLOC) 1.12 PM

Embedded: Effort = 3.6(KLOC) 1.20 PM

Estimation of development time

For the three classes of software products, the formulas for estimating the development time based on the effort are given below:

Organic: Tdev = 2.5(Effort) 0.38 Months

Semi-detached: Tdev = 2.5(Effort) 0.35 Months

Embedded: Tdev = 2.5(Effort) 0.32 Months

Estimation for our Project:

KLOC=0.18

	ORGANIC	SEMI-DETACHED	EMBEDDED
Variable A	2.4	3.0	3.6
Variable B	1.05	1.12	1.2
Variable C	2.5	2.5	2.5
Variable D	0.38	0.35	0.32
KLOC	0.18	0.18	0.18
Effort(Person/Month)	0.3965038166340373	0.43956839638325296	0.45986409985413573
Duration(Months)	1.75903309326578	1.8749913556578968	1.949761321853349
Staffing	0.22541009498456763	0.23443755890224743	0.2358566121401009

Table 3.4.1 Project Estimation(1)

Function Point (FP) is an element of software development which helps to approximate the cost of development early in the process. It may measure functionality from the user's point of view.

User Input = 4
User Output = 1
User Inquiries = 0
User Files = 0
External Interface = 1

Counting Function Point (FP):

Step-1:

F = 14 * scale

Scale varies from 0 to 5 according to character of Complexity Adjustment Factor (CAF).

Below table shows scale:

0 - No Influence

1 - Incidental

2 - Moderate

3 - Average

4 - Significant

5 - Essential

F = 14 * 3 = 42

Step-2: Calculate Complexity Adjustment Factor (CAF).

CAF = 0.65 + (0.01 * F)

CAF = 0.65 + (0.01 * 42) = 1.07

Step-3: Calculate Unadjusted Function Point (UFP)

UFP = (4*4) + (1*5) + (0*4) + (0*10) + (1*7) = 28

Step-4:

Function Point = CAF * UFP

Function Point = 1.07 * 28 = 29.96

FUNCTION UNITS	COUNT	AVERAGE	COUNT * AVG
User Input	4	4	16

User Output	1	5	5
User Inquiries	0	4	0
User Files	0	10	0
User Interface	1	7	7

Fig 3.4.2 Project Estimation (2)

3.5 Software Requirement Specification

3.5.1 Introduction

3.5.1.1 Purpose

The purpose of the Blood Donation Prediction System is to analyze blood transfusion dataset and build a model that predicts probability of a person donating blood.

3.5.1.2 Scope

This system is used to analyze the history of a person that helps in various ways including the following: Professionals require this information for finding donors. Critical Patients would be getting blood components in time through filtering of 1000s of donors. Concepts and techniques of data analysis and machine learning, are very much useful to explain the past and predict the future by analyzing and exploring the data.

3.5.2 Overall Description

3.5.2.1 Product Perspective

In our project, we test different classification algorithms against each other and choose the one that is best for our problem statement. Models like Logistic Regression, Adaboost Classifier, Stochastic Gradient Descent, K-Nearest Neighbours, Decision Tree, Random Forest, Support Vector Machine, etc.

The main purpose of this project is to make a prediction model that would generate potential donors based on their past donation patterns. Of all the parameters present in our dataset, we have chosen only those who majorly affect the results. Through simple data visualization techniques we have found great insights into the dataset and relations between all the features. Data Visualization helps in determining the good and bad features that will influence the final predictions. Based on these helpful insights, we remove less important features from the dataset. Then we run cross-validation on various algorithms, from the statistics obtained we chose a few algorithms and trained our model with them. The algorithm with the highest accuracy is tested against the test dataset to generate probabilities. The probability is in the range of 0-1. Where 0-0.5 indicates that the probability of a person donating is low and 0.5-1 denotes that the person is more likely to donate.

Once our model is trained, we will use the trained model and run it on the test set and predict the output. Then we will compare the predicted results with the actual results that we have to see how our model performed. This whole process of training the model using features and known labels and later testing it to predict the output is called Supervised Learning. In further sections, we explore the different models and try to understand their functioning.

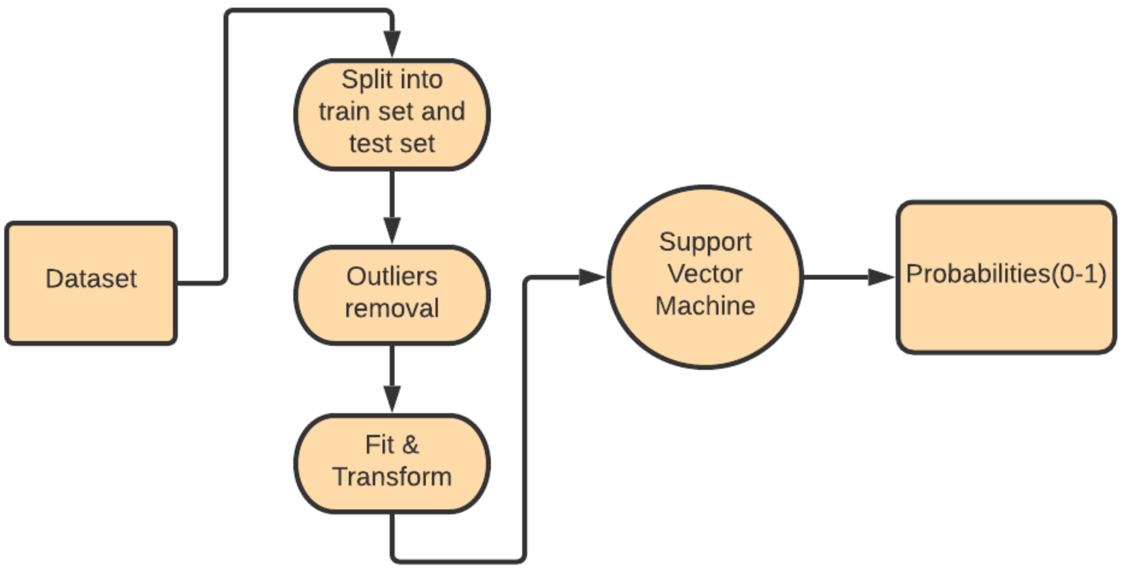


Fig 3.5.2.1 System Design

3.5.2.2 Product Functions

The product's goal is to predict the number of potential donors. The functions of the products are as follows:

1. To take the input from the user for the predictions.

2. Use the support vector model to predict the potential donors.

3. Display the results in terms of probabilities.

3.5.2.3 User Classes and Characteristics

There is one user class of this product as follows:

i. Healthcare Official (User): This person is responsible for providing all the relevant donor information and parameter data that when fed to the model gives the probability of donation.

3.5.2.4 Operating Environment

The operating environment for the system is as listed below.

1. Operating System: Windows 7 or above
2. Programming Language: Python
3. Programming Environment: Anaconda
4. Front End Environment: Streamlit
5. Dataset : UCI Transfusion Data Set

   [TransfusionData.csv]

3.5.2.5 Software Interfaces

1.Python is an interpreted, high-level, general-purpose programming language. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented,and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. It uses the following libraries:

2.NumPy(Numerical Python) is a perfect tool for scientific computing and performing basic and advanced array operations.

3. Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython etc. Pyplot is a Matplotlib module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python, and the advantage of being free and open-source.

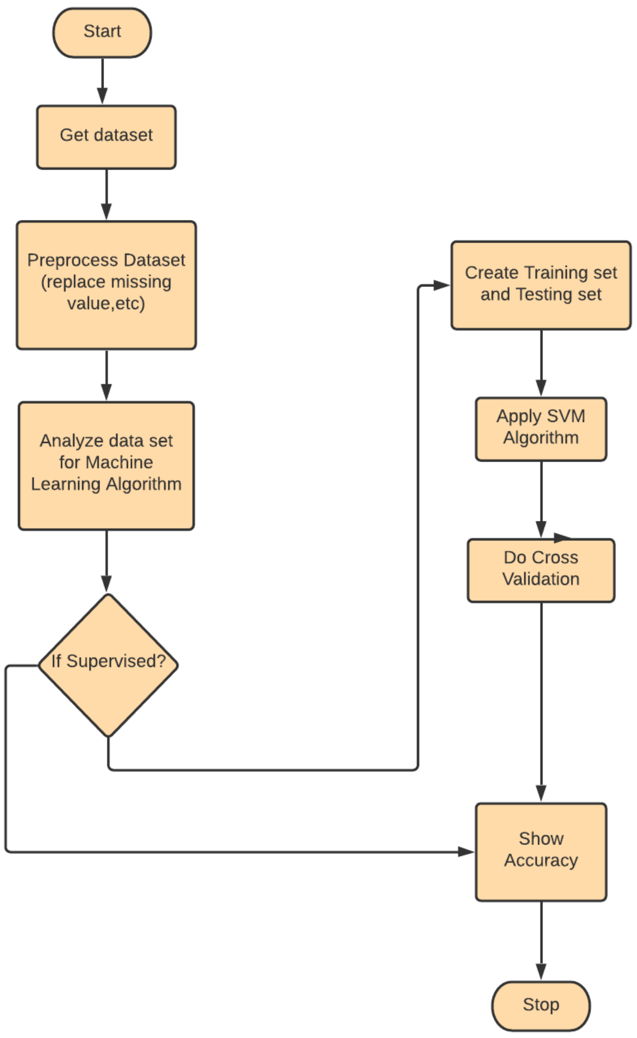
4.Pandas is a library created to help developers work with "labelled" and "relational" data intuitively. It's based on two main data structures: "Series" (one-dimensional, like a list of items) and "Data Frames" (two-dimensional, like a table with multiple columns). Pandas allows converting data structures to Data Frame objects, handling missing data, and adding/deleting columns from Data Frame, imputing missing files, and plotting data with histogram or plot box. It’s a must-have for data wrangling, manipulation, and visualization.

5. Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

4. SYSTEM DESIGN

4.1 System Architecture:

The System Architecture consists of :



Start
Get dataset
Preprocess Dataset (replace missing value,etc)
Analyze data set for Machine Learning Algorithm
If Supervised?
Create Training set and Testing set
Apply SVM Algorithm
Do Cross Validation
Show Accuracy
Stop

Fig 4.1 System Architecture

4.2 Data Flow Diagram

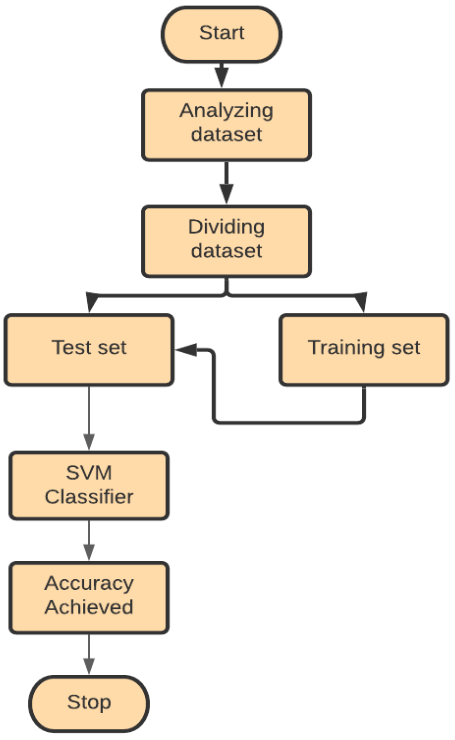
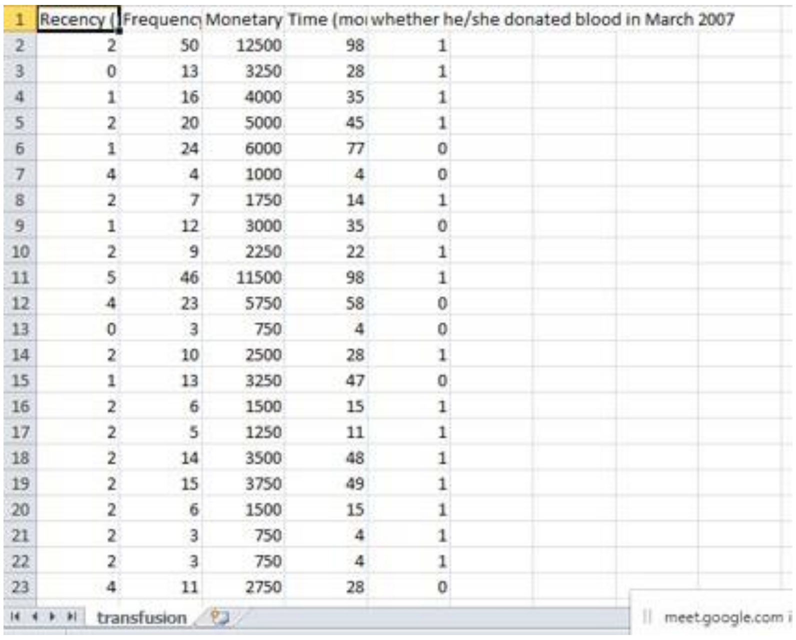


Fig 4.2 Data Flow

5. IMPLEMENTATION



1	Recency (	Frequenc	Monetary	Time (mo	whether he/she donated blood in March 2007
2	2	50	12500	98	1
3	0	13	3250	28	1
4	1	16	4000	35	1
5	2	20	5000	45	1
6	1	24	6000	77	0
7	4	4	1000	4	0
8	2	7	1750	14	1
9	1	12	3000	35	0
10	2	9	2250	22	1
11	5	46	11500	98	1
12	4	23	5750	58	0
13	0	3	750	4	0
14	2	10	2500	28	1
15	1	13	3250	47	0
16	2	6	1500	15	1
17	2	5	1250	11	1
18	2	14	3500	48	1
19	2	15	3750	49	1
20	2	6	1500	15	1
21	2	3	750	4	1
22	2	3	750	4	1
23	4	11	2750	28	0

Fig 5.1 Dataset

5.1 Execution

blood-donation-prediction.ipynb

~~~
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import math
import matplotlib.pyplot as plt
%matplotlib inline
~~~

```python
#Creating training DF
df = pd.read_csv('train.csv', index_col=False)
#Renaming last column
```

~~~
df.columns= ['id','months_since_last_donation','num_donations','vol_donations','months_since_first_donation', 'class']
#Creating test DF
test = pd.read_csv("test.csv")
#Renaming columns
test.columns = ['id','months_since_last_donation','num_donations','vol_donations','months_since_first_donation']
#Creating ID for testset
IDtest = test["id"]

print(df.shape) #prints (rows, cols)
print(df.isnull().sum()) #checking for null values
df
```


```python
from collections import Counter
#function for outlier detection columnwise
def detect_outliers(df,n,features):
    outlier_indices=[]
    for col in features:
        quartile_1 = np.percentile(df[col],25)
        quartile_3 = np.percentile(df[col],75)
        inter_quartile_range= quartile_3 - quartile_1

        outlier_step= 1.5*inter_quartile_range

        outlier_list_for_each_col= df[(df[col]< quartile_1 - outlier_step) | (df[col] > quartile_3 + outlier_step)].index
        outlier_indices.extend(outlier_list_for_each_col)
    outlier_indices= Counter(outlier_indices)
~~~

    mul_outliers = list(k for k,v in outlier_indices.items() if v > n)
    return mul_outliers
outliers_to_drop= detect_outliers(df, 2,
['months_since_last_donation','num_donations','vol_donations','months_since_first_donation'
])

df.loc[outliers_to_drop]

# Joining Train and Test Data

```python
train_len = len(df)
dataset =  pd.concat(objs=[df, test], axis=0,sort=True).reset_index(drop=True)
# Fill empty and NaNs values with NaN
dataset = dataset.fillna(np.nan)
# Check for Null values
dataset.isnull().sum()
train_len
```
# Feature Analysis

```python
g =
sns.heatmap(df[["class","months_since_last_donation","num_donations","months_since_first
_donation"]].corr(),annot=True, fmt = ".2f", cmap = "coolwarm")
```

```python
#features class 0 and 1
g = sns.FacetGrid(df, col='class')
```

```python
g = g.map(sns.distplot, "num_donations")
```

```python
g = sns.FacetGrid(df, col='class')
g = g.map(sns.distplot, "months_since_last_donation")

```

```python
g = sns.FacetGrid(df, col='class')
g = g.map(sns.distplot, "months_since_first_donation")
```

```python
g = sns.FacetGrid(df, col='class')
g = g.map(sns.distplot, "vol_donations")
```

```python
plt.figure(figsize=(15,10));
_ = sns.lmplot(x='num_donations',
               y='vol_donations',
               hue='class',
               fit_reg=False,
               data=df);
_ = plt.title("Correlation between frequency and monetary");
plt.show();
```

```
    <Figure size 1080x720 with 0 Axes>
```

![png](output_12_1.png)

```python
dataset.drop(['id', 'vol_donations'], axis=1, inplace=True)
```

# Feature Engineering

```python
dataset["new_variable"] = (dataset["months_since_first_donation"] -
dataset["months_since_last_donation"])
```

```python
## Separate train dataset and test dataset
train = dataset[:train_len]
test = dataset[train_len:]
test.drop(labels=["class"],axis = 1,inplace=True)
dataset
```

```python
## Separate train features and label

train["class"] = train["class"].astype(int)

Y_train = train["class"]

X_train = train.drop(labels = ["class"],axis = 1)
```

```
    <ipython-input-17-ae3fc8d9a1e8>:3: SettingWithCopyWarning:
```

```
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
      train["class"] = train["class"].astype(int)
```

```python
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_scaled = sc.fit_transform(X_train)
```

```python
# Import the packages
from collections import Counter
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,
GradientBoostingClassifier, ExtraTreesClassifier, VotingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, cross_val_score, StratifiedKFold,
learning_curve
```

```python
# Cross validate model with Kfold stratified cross val
kfold = StratifiedKFold(n_splits=10)
# Modeling step Test differents algorithms
random_state = 7
```

```
classifiers = []
classifiers.append(SVC(random_state=random_state))
classifiers.append(DecisionTreeClassifier(random_state=random_state))
classifiers.append(AdaBoostClassifier(DecisionTreeClassifier(random_state=random_state),r
andom_state=random_state,learning_rate=0.1))
classifiers.append(RandomForestClassifier(random_state=random_state))
classifiers.append(ExtraTreesClassifier(random_state=random_state))
classifiers.append(GradientBoostingClassifier(random_state=random_state))
classifiers.append(MLPClassifier(random_state=random_state))
classifiers.append(KNeighborsClassifier())
classifiers.append(LogisticRegression(random_state = random_state))
classifiers.append(LinearDiscriminantAnalysis())

cv_results = []
for classifier in classifiers :
    cv_results.append(cross_val_score(classifier, X_train_scaled, y = Y_train, scoring =
"accuracy", cv = kfold, n_jobs=4))

cv_means = []
cv_std = []
for cv_result in cv_results:
    cv_means.append(cv_result.mean())
    cv_std.append(cv_result.std())

cv_res = pd.DataFrame({"CrossValMeans":cv_means,"CrossValerrors":
cv_std,"Algorithm":["SVC","DecisionTree","AdaBoost",
"RandomForest","ExtraTrees","GradientBoosting","MultipleLayerPerceptron","KNeighboors
","LogisticRegression","LinearDiscriminantAnalysis"]})

g = sns.barplot("CrossValMeans","Algorithm",data = cv_res, palette="Set3",orient =
"h",**{'xerr':cv_std})
g.set_xlabel("Mean Accuracy")
g = g.set_title("Cross validation scores")
```

~~~~
```

```python
# RFC Parameters tuning
RFC = RandomForestClassifier()


## Search grid for optimal parameters
rf_param_grid = {"max_depth": [80, 90, 100, 110],
              "max_features": [2, 3],
              "min_samples_split": [8, 10, 12],
              "min_samples_leaf": [3, 4, 5],
              "bootstrap": [False],
              "n_estimators" :[100, 200, 300, 1000],
              "criterion": ["gini"]}


gsRFC = GridSearchCV(RFC,param_grid = rf_param_grid, cv=kfold, scoring="accuracy",
n_jobs= 4, verbose = 1)

gsRFC.fit(X_train_scaled,Y_train)

RFC_best = gsRFC.best_estimator_

# Best score
gsRFC.best_score_

```
~~~~

```
Fitting 10 folds for each of 288 candidates, totalling 2880 fits
```

```
#ExtraTrees
ExtC = ExtraTreesClassifier()

## Search grid for optimal parameters
ex_param_grid = {"max_depth": [None],
              "max_features": [2, 3],
              "min_samples_split": [2, 3, 10],
              "min_samples_leaf": [1, 3, 10],
              "bootstrap": [False],
              "n_estimators" :[100, 200, 300, 1000],
              "criterion": ["gini"]}

gsExtC = GridSearchCV(ExtC,param_grid = ex_param_grid, cv=kfold, scoring="accuracy",
n_jobs= 4, verbose = 1)

gsExtC.fit(X_train_scaled,Y_train)

ExtC_best = gsExtC.best_estimator_

# Best score
gsExtC.best_score_

# Gradient boosting tuning

GBC = GradientBoostingClassifier()
gb_param_grid = {'loss' : ["deviance"],
              'n_estimators' : [100,200,300],
              'learning_rate': [0.1, 0.05, 0.01],
              'max_depth': [4, 8],
              'min_samples_leaf': [100,150],
              'max_features': [0.3, 0.1]
              }
```

```
gsGBC = GridSearchCV(GBC,param_grid = gb_param_grid, cv=kfold, scoring="accuracy",
n_jobs= 4, verbose = 1)

gsGBC.fit(X_train_scaled,Y_train)

GBC_best = gsGBC.best_estimator_

# Best score
gsGBC.best_score_
```

```python
### SVC classifier
SVMC = SVC(probability=True)
svc_param_grid = {'kernel': ['rbf'],
                  'gamma': [ 0.001, 0.01, 0.1, 1],
                  'C': [1, 10, 50, 100,200,300, 1000]}

gsSVMC = GridSearchCV(SVMC,param_grid = svc_param_grid, cv=kfold,
scoring="accuracy", n_jobs= 4, verbose = 1)

gsSVMC.fit(X_train_scaled,Y_train)

SVMC_best = gsSVMC.best_estimator_

# Best score
gsSVMC.best_score_
```
```python
# Plot learning curve
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=-1, train_sizes=np.linspace(.1, 1.0, 5)):
    """Generate a simple plot of the test and training learning curve"""
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")

    plt.legend(loc="best")
    return plt

#g = plot_learning_curve(gsRFC.best_estimator_,"RF mearning
curves",X_train,Y_train,cv=kfold)
#g = plot_learning_curve(gsExtC.best_estimator_,"ExtraTrees learning
curves",X_train,Y_train,cv=kfold)
g = plot_learning_curve(gsGBC.best_estimator_,"GradientBoosting learning
curves",X_train,Y_train,cv=kfold)
```

```
g = plot_learning_curve(gsSVMC.best_estimator_,"SVC learning
curves",X_train,Y_train,cv=kfold)
```

```python
# Scaling the test data
test_scaled = sc.transform(test)
# predicting the results

predictions = gsSVMC.predict_proba(test_scaled)

predictions = predictions[:,1]

pred_report = pd.DataFrame(predictions.tolist(),index=IDtest,columns=["Made Donation in

March 2007"])

pred_report.to_csv("result.csv")
pred_report
```

```python
for i in pred_report["Made Donation in March 2007"]:
    if i >0.5:
        print(i*100)
```

```python
x=pred_report["Made Donation in March 2007"]

x.hist(bins=20)

plt.xlabel("Probabilities")

plt.ylabel("Number of donors")

plt.title("Output")
```

```python

labels=np.array(pred_0_1)

labels[labels>=0.5]=1

labels[labels<=0.5]=0
`
```

6. TESTING

Test case id	**Test case Name**	**Test case Description**	**Steps to be executed**	**Expected Result**	**Actual Result**	**Test case Status**	**Test case Priority**
01	upload the transfusion dataset	verify either file is loaded or not	If the dataset is not uploaded	It cannot display the file loaded message	The file is loaded which displays the donor information	high	high
02	preprocessing	whether preprocessing on the dataset applied or not	If not applied	It cannot display the necessary data for further process	It can display the necessary data for further process	Medium	high
02	prediction SVM	whether prediction algorithm applied to the data or not	If not applied	The SVM doesn’t generate predictions	The SVM generates predictions in terms of probabilities	high	high
04	Recommendation	whether predicted data is displayed or not	If not displayed	It cannot view prediction containing donor data	It can display prediction of donors	high	high

Table 6.1 Testing

7. SCREENSHOT

Features in the dataset

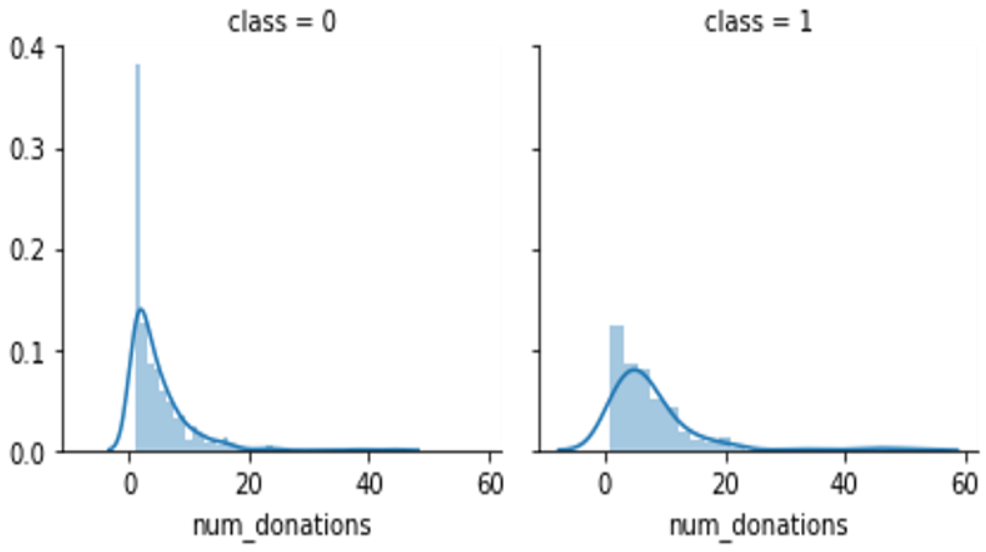


Fig 7.1 "Number of donations"

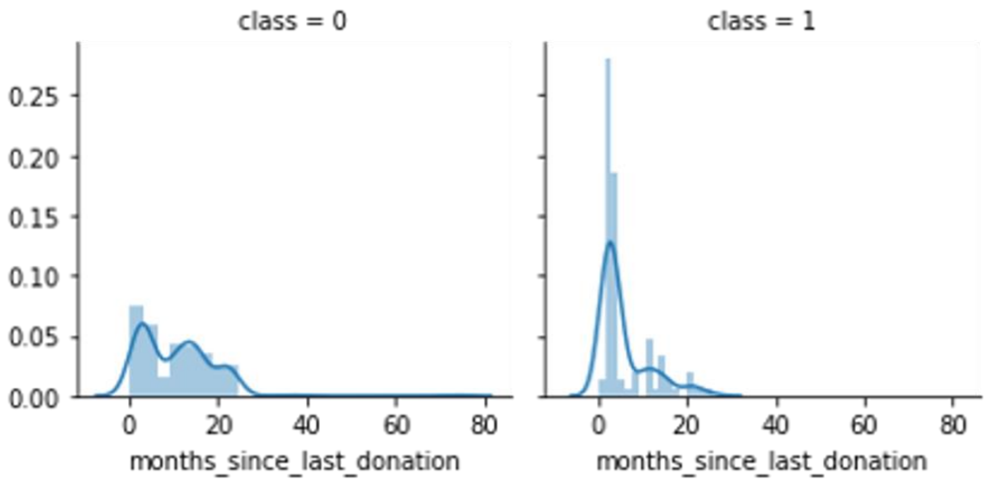


Fig 7.2 "Months since last donation"

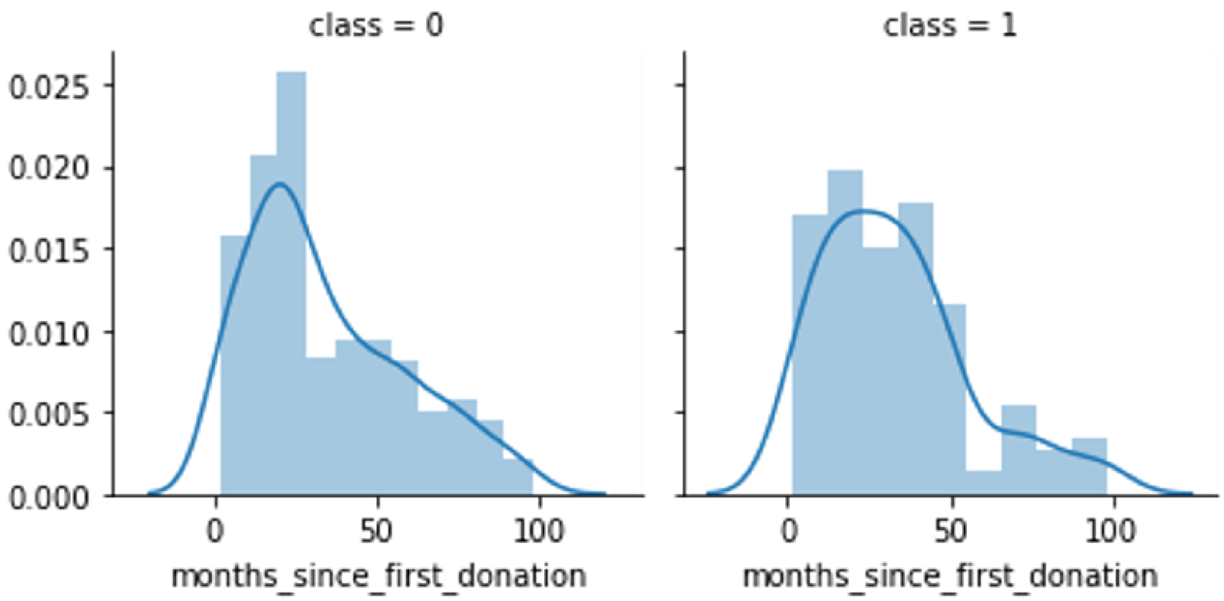


Fig 7.3 "Months since first donation"

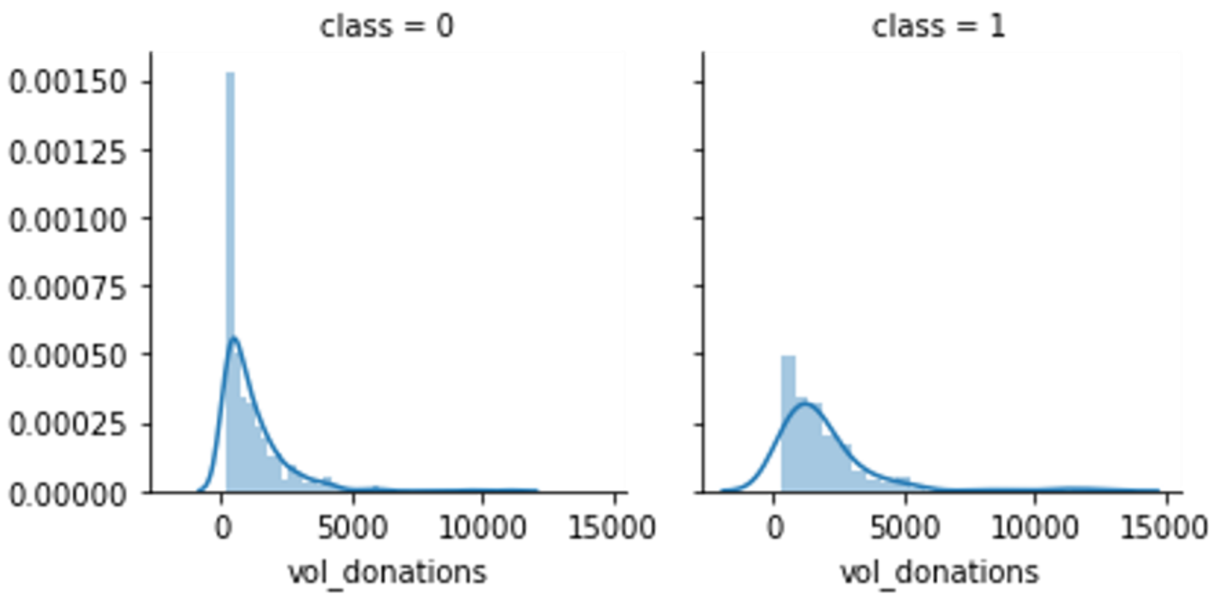


Fig 7.4 "Volume of donation(cc)"

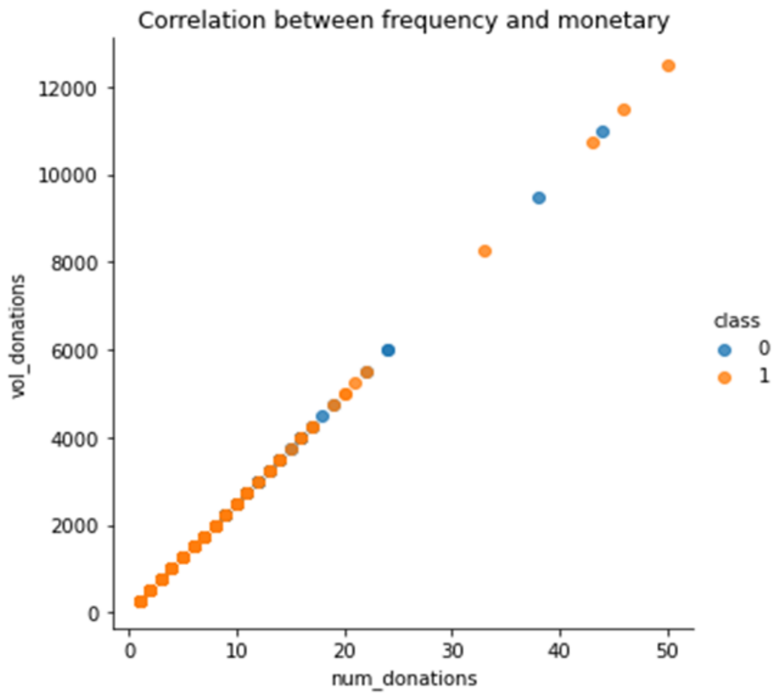


Fig 7.5 Correlation between two features

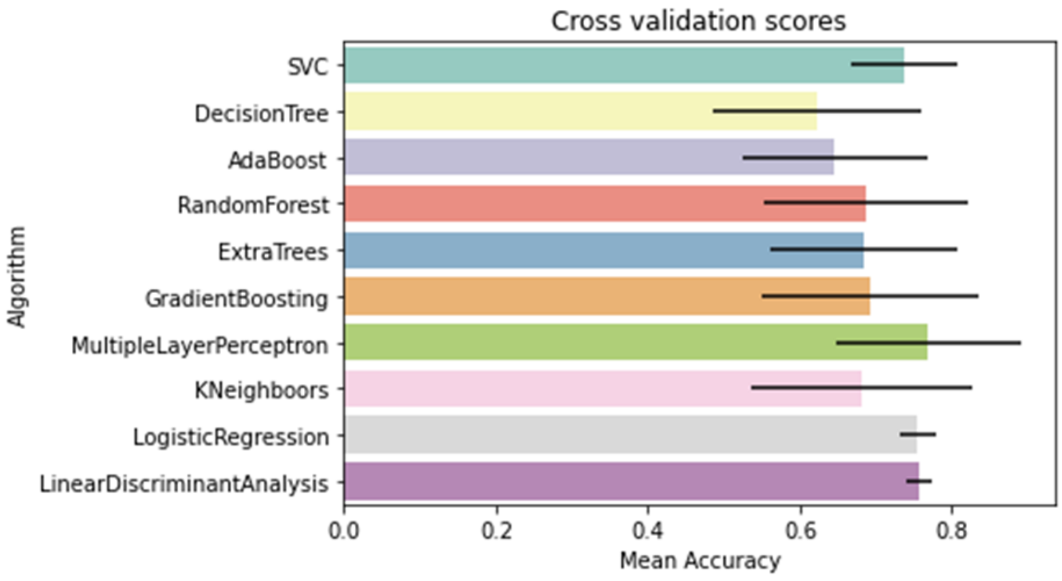


Fig 7.6 Cross Validation Score

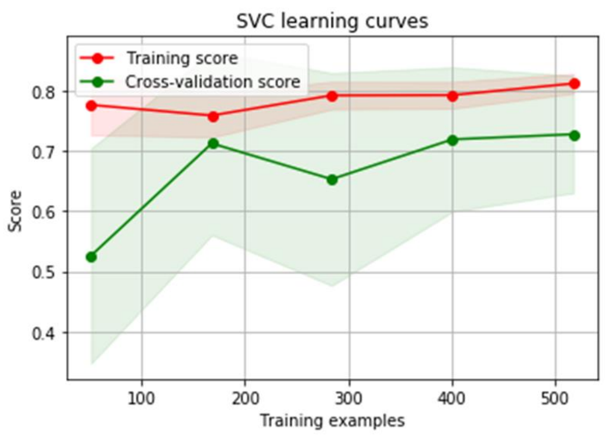
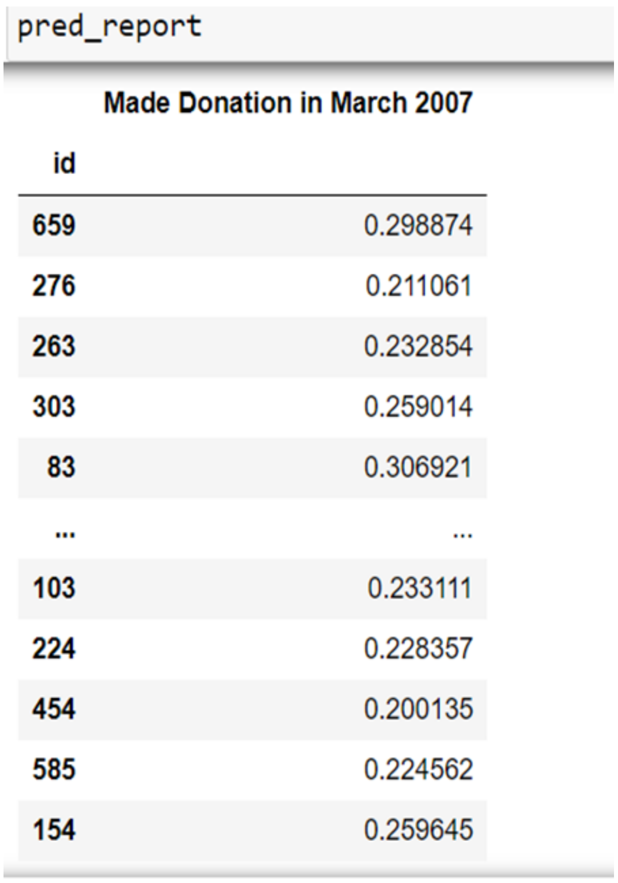


Fig 7.8 Learning curve of SVC

Output



```
pred_report
```

	Made Donation in March 2007
id	
659	0.298874
276	0.211061
263	0.232854
303	0.259014
83	0.306921
...	...
103	0.233111
224	0.228357
454	0.200135
585	0.224562
154	0.259645

Fig 7.9 Predicted probabilities

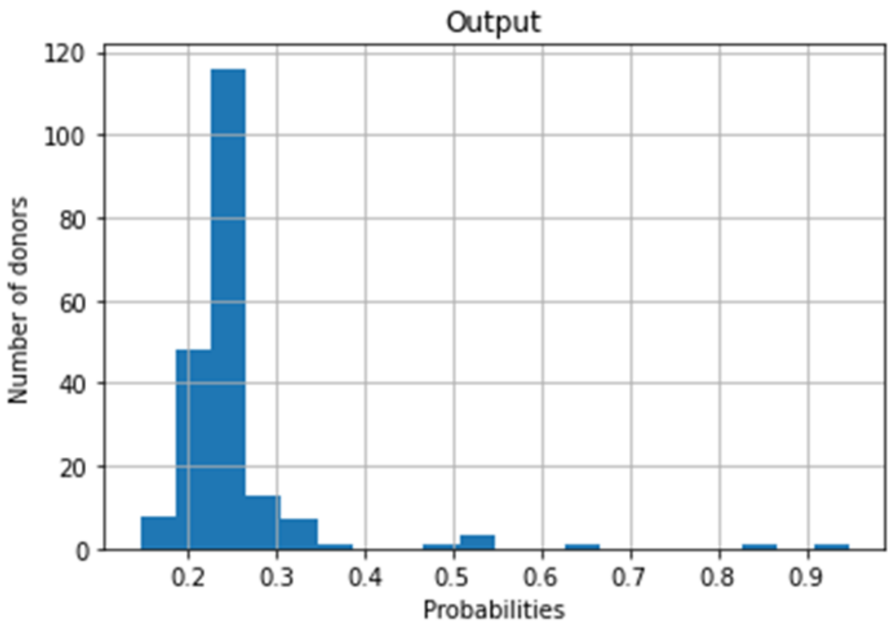


Fig 7.9.1 Potential donors in the test set

8. CONCLUSION

Support Vector Machine gives the best results compared to other classification models for our problem statement. A basic UI could be developed that takes real-time datasets from blood banks and allows healthcare professionals to find donors.

Factors like geographical data could be added so that donors in the nearest distance could be found. Functions for different blood groups could be made, which increases the efficiency and speed of the model

REFERENCES

[1] P. Kirci, S. Aktas and B. Sevinc "Analyzing Blood Donation probabilities and number of possible donors," 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2020, pp. 1-4, doi: 10.1109/HORA49412.2020.9152872. (Base Paper))

[2] W. B. Zulfikar, Y. A. Gerhana and A. F. Rahmania, "An Approach to Classify Eligibility Blood Donors Using Decision Tree and Naive Bayes Classifier," 2018 6th International Conference on Cyber and IT Service Management (CITSM), Parapat, Indonesia, 2018, pp. 1-5, doi: 10.1109/CITSM.2018.8674353.

[3] SM. Fahim, H. I. Cebe, J. Rasheed and F. Kiani, "mHealth: Blood donation application using android smartphone," 2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), Konya, 2016, pp. 35-38, doi: 10.1109/DICTAP.2016.7543997.

APPENDIX 1

RELEVANCE OF PROJECT TO POs / PSOs

Title of Project	BLOOD DONATION PREDICTION
Implementation Details	Machine Learning model
Cost (hardware or software cost)	N/A
Type (Application, Product, Research, Review, etc.)	Application

Mapping with POs and PSOs with Justification

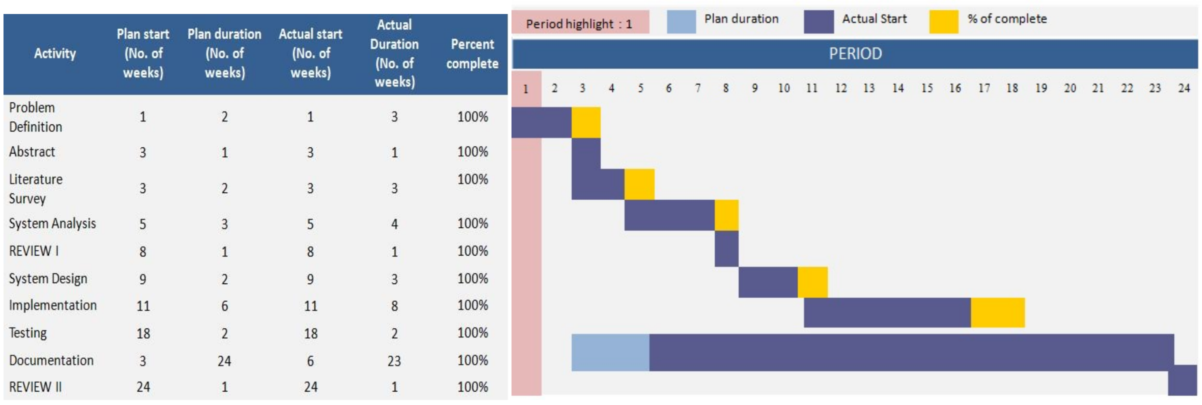
Relevance	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
	1	2	3	4	4	1	1	1	2	3	-	2	1	1

Program Outcomes Justification	PO1: Engineering Knowledge: SDLC phases are followed in the execution of the project. PO2: Problem Analysis: The different steps involved in Problem Analysis for formulation of the solution i.e. literature survey and use of fundamental subject knowledge has been followed. PO3: Design/Development of solutions – Existing strategy has been enhanced using the design principles. PO5: Modern Tool Usage: Python is used. PO8: Ethics: Students have followed professional ethics during the various stages of Project completion. PO9: Individual and Team Work: Students have worked both in individual as well as team capacity during the various stages of project work. PO10: Communication: Effective communication with team members and during project reviews, project seminar and viva-voce has been exhibited. PO12: Lifelong Learning. The project carried out gives the students scope to continue the work in the Machine Learning area in future.
Program Specific Outcomes Justification	PSO1: Use of open source software: Python PSO2: Use of Rational Software Architect is during analysis and testing.

APPENDIX 2

GANTT CHART

Blood Donation Prediction



Activity	Plan start (No. of weeks)	Plan duration (No. of weeks)	Actual start (No. of weeks)	Actual Duration (No. of weeks)	Percent complete
Problem Definition	1	2	1	3	100%
Abstract	3	1	3	1	100%
Literature Survey	3	2	3	3	100%
System Analysis	5	3	5	4	100%
REVIEW I	8	1	8	1	100%
System Design	9	2	9	3	100%
Implementation	11	6	11	8	100%
Testing	18	2	18	2	100%
Documentation	3	24	6	23	100%
REVIEW II	24	1	24	1	100%