

Loan Approval Prediction Using Supervised Learning Algorithm

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering by

Muvva Anudeep (38110343)



Kattela Manoj Kumar (38110304)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by
AICTE**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600 119

MAY – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC I 12B Status
byUGC I Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Muvva Anudeep (38110343)** and **Kattela Manoj Kumar (Reg. No. 38110304)** who carried out the project entitled “**Loan Approval Prediction Using Supervised Learning Algorithm**” under my supervision from November 2021 to March 2022.

Internal Guide

Dr. A. Pravin M.E., Ph.D.,

Head of the Department

Dr. L. Lakshmanan M.E., Ph.D.,

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I **Muvva Anudeep** hereby declare that the Project Report entitled “**Loan Approval Prediction Using Supervised Learning Algorithm**” is done by under the guidance of Dr. A. Pravin M.E., Ph.D., Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind of encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D.**, Dean School of Computing and

Dr. L. Lakshmanan M.E., Ph.D., and **Dr. S. Vigneshwari M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. Pravin M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Technology has boosted the existence of humankind the quality of life they live. Every day we are planning to create something new and different. We have a solution for every other problem we have machines to support our lives and make us somewhat complete in the banking sector candidate gets proofs/ backup before approval of the loan amount. The application approved or not approved depends upon the historical data of the candidate by the system. Every day lots of people applying for the loan in the banking sector but Bank would have limited funds. In this case, the right prediction would be very beneficial using some classes-function algorithm. An example the logistic regression, random forest classifier, support vector machine classifier, etc. A Bank's profit and loss depend on the amount of the loans that is whether the Client or customer is paying back the loan. Recovery of loans is the most important for the banking sector. The improvement process plays an important role in the banking sector. The historical data of candidates was used to build a machine learning model using different classification algorithms. The main objective of this paper is to predict whether a new applicant granted the loan or not using machine learning models trained on the historical data set.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
	2.1 LITERATURE SURVEY 1	2
	2.2 LITERATURE SURVEY 2	2
	2.3 LITERATURE SURVEY 3	3
	2.4 LITERATURE SURVEY 4	4
	2.5 LITERATURE SURVEY 5	4
3	METHODOLOGY	5
	3.1 EXISITING SYSTEM	5
	3.2 PROPODED SYSTEM	5
	3.3 SYSTEM ARCHITECTURE	6
	3.4 FLOW CHART	6
	3.5 UML DIAGRAMS	7
	3.5.1 USE CASE DIAGRAM	8
	3.5.2 CLASS DIAGRAM	9
	3.5.3 SEQUENCE DIAGRAM	10
	3.5.4 ACTIVITY DIAGRAM	10
	3.6 SYSTEM REQUIREMENTS	11
	3.6.1 HARDWARE REQUIREMENTS	11
	3.6.2 SOFTWARE REQUIREMENTS	11
4	SOFTWARE MODULES	12
	4.1 PYTHON	12
	4.1.1 PYTHON HISTORY	12
	4.2 NUMPY	15
	4.2.1 INSTALL NUMPY	16

	4.2.2 IMPORT NUMPY	16
	4.2.3 USE OF NUMPY	17
	4.2.4 NUMPY ARRAY	17
	4.3 MATPLOTLIB	18
	4.4 MODULE DESCRIPTION	18
	4.4.1 DATA COLLECTION	18
	4.4.2 DATASET	18
	4.4.3 DATA PREPERATION	19
	4.4.4 MODEL SELECTION	19
	4.4.5 ANALYSE AND PREDICTION	19
	4.4.6 ACCURACY ON TEST SET	20
	4.4.7 SAVING THE TRAINED MODEL	20
	4.5 PROPOSED ALGORITHM	20
	4.5.1 SUPPORT VECTOR MACHINE (SVM)	20
	4.6 SYSTEM TESTING	37
5	CONCLUSION	40
	5.1 CONCLUSION	40
	5.2 FUTURE ENHANCEMENT	40
	5.3 APENDICES	42
	A.SAMPLE CODE	42
	B.SCREENSHOTS	46

LIST OF ABBREVIATIONS

ABBREVIATION

EXPANSION

SVM

Support Vector Machine

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	Architecture Diagram	6
3.2	Flow chart	7
3.3	Use case diagram	9
3.4	Class diagram	9
3.5	Sequence Diagram	10
3.6	Activity diagram	11
5.1	Home page	48
5.2	Login page	49
5.3	Entering credentials	50
5.4	Upload page	51
5.5	Uploaded page	51
5.6	Preview of data set	52
5.7	Training data set	53
5.8	Prediction page	53
5.9	Prediction result	54
5.10	Performance Analysis	55
5.11	Chart Representation of Data set	56

CHAPTER 1

INTRODUCTION

As the data are increasing daily due to digitization in the banking sector, people want to apply for loans through the internet. Artificial intelligence (AI), as a typical method for information investigation, has gotten more consideration increasingly. Individuals of various businesses are utilizing AI calculations to take care of the issues dependent on their industry information. Banks are facing a significant problem in the approval of the loan. Daily there are so many applications that are challenging to manage by the bank employees, and also the chances of some mistakes are high. Most banks earn profit from the loan, but it is risky to choose deserving customers from the number of applications. One mistake can make a massive loss to a bank. Loan distribution is the primary business of almost every bank. This project aims to provide a loan to a deserving applicant out of all applicants. An efficient and non-biased system that reduces the bank's time employs checking every applicant on a priority basis. The bank authorities complete all other customer's other formalities on time, which positively impacts the customers. The best part is that it is efficient for both banks and applicants. This system allows jumping on particular applications that deserve to be approved on a priority basis. There are some features for the prediction like- 'Gender', 'Married', 'Dependents', 'Education', 'Self_ Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY 1

“Prediction for Loan Approval using Machine Learning Algorithm”

AUTHORS: Ashwini S. Kadam, Shraddha R Nikam, Ankita A. Aher, Gayatri V. Shelke, Amar S. Chandgude

Content:

In our banking system, banks have many products to sell but main source of income of any banks is on its credit line. So they can earn from interest of those loans which they credits. A bank's profit or a loss depends to a large extent on loans i.e. whether the customers are paying back the loan or defaulting. By predicting the loan defaulters, the bank can reduce its Non-performing Assets. This makes the study of this phenomenon very important. Previous research in this era has shown that there are so many methods to study the problem of controlling loan default. But as the right predictions are very important for the maximization of profits, it is essential to study the nature of the different methods and their comparison. A very important approach in predictive analytics is used to study the problem of predicting loan defaulters (i) Collection of Data, (ii) Data Cleaning and (iii) Performance Evaluation. Experimental tests found that the Naïve Bayes model has better performance than other models in terms of loan forecasting.

2.2 LITERATURE SURVEY 2

“An Approach for Prediction of Loan Approval using Machine Learning Algorithm”

AUTHORS: Mohammad Ahmad Sheikh, Amit Kumar Goel, Tapas Kumar

Content:

In our banking system, banks have many products to sell but main source of income of any banks is on its credit line. So they can earn from interest of those loans which they credits. A bank's profit or a loss depends to a large extent on loans i.e. whether the customers are paying back the loan or defaulting. By predicting the loan

defaulters, the bank can reduce its Non- Performing Assets. This makes the study of this phenomenon very important. Previous research in this era has shown that there are so many methods to study the problem of controlling loan default. But as the right predictions are very important for the maximization of profits, it is essential to study the nature of the different methods and their comparison. A very important approach in predictive analytics is used to study the problem of predicting loan defaulters: The Logistic regression model. The data is collected from the Kaggle for studying and prediction. Logistic Regression models have been performed and the different measures of performances are computed. The models are compared on the basis of the performance measures such as sensitivity and specificity. The final results have shown that the model produce different results. Model is marginally better because it includes variables (personal attributes of customer like age, purpose, credit history, credit amount, credit duration, etc.) other than checking account information (which shows wealth of a customer) that should be taken into account to calculate the probability of default on loan correctly. Therefore, by using a logistic regression approach, the right customers to be targeted for granting loan can be easily detected by evaluating their likelihood of default on loan. The model concludes that a bank should not only target the rich customers for granting loan but it should assess the other attributes of a customer as well which play a very important part in credit granting decisions and predicting the loan defaulters.

2.3 LITERATURE SURVEY 3

“An exploratory Data Analysis for Loan Prediction based on nature of clients “

AUTHORS: X.FrencisJensy, V.P.Sumathi,Janani Shiva Shri

Content:

In India, the number of people applying for the loans gets increased for various reasons in recent years. The bank employees are not able to analyse or predict whether the customer can payback the amount or not (good customer or bad customer) for the given interest rate. The aim of this paper is to find the nature of the client applying for the personal loan. An exploratory data analysis technique is used to deal with this problem. The result of the analysis shows that short term loans are preferred by majority of the clients and the clients majorly apply loans for debt

consolidation. The results are shown in graphs that helps the bankers to understand the client's behaviour.

2.4 LITERATURE SURVEY 4

“ACCURATE LOAN APPROVAL PREDICTION BASED ON MACHINE LEARNING APPROACH”

AUTHORS: J. Tejaswini¹, T. Mohana Kavya, R. Devi Naga Ramya, P. Sai Triveni Venkata Rao Maddumala

Content: Loan approval is a very important process for banking organizations. Banking Industry always needs a more accurate predictive modeling system for many issues. Predicting credit defaulters is a difficult task for the banking industry. The system approved or rejects the loan applications. Recovery of loans is a major contributing parameter in the financial statements of a bank. It is very difficult to predict the possibility of payment of loan by the customer. Machine Learning (ML) techniques are very useful in predicting outcomes for large amount of data. In this paper three machine learning algorithms, Logistic Regression (LR), Decision Tree (DT) and Random Forest (RF) are applied to predict the loan approval of customers. The experimental results conclude that the accuracy of Decision Tree machine learning algorithm is better as compared to Logistic Regression and Random Forest machine learning approaches.

2.5 LITERATURE SURVEY 5

“Predictive and probabilistic approach using logistic regression: Application to prediction of loan approval”

AUTHORS: Vaidya

Decision taking is attained by probabilistic and predictive approaches developed by various machine learning algorithms. This paper discusses about logistic regression and its mathematical representation. This paper adheres to logistic regression as a machine learning tool in order to actualize the predictive and probabilistic approaches to a given problem of loan approval prediction. Using logistic regression as a tool, this paper specifically delineates about whether or not loan for a set of records of an applicant will be approved.

CHAPTER 3

METHODOLOGY

3.1 EXISTING SYSTEM

- ❖ Y. Shi and P. Song proposed a method for evaluating project loans using risk analysis. The method evaluates the risk involved in loans of commercial banks.
- ❖ R. Zhang and D. Li used machine learning approaches in prediction systems. The machine learning approach was used for assessment of water quality. The paper concluded that machine learning is a very unimportant tool in prediction systems.
- ❖ C. Frank et al. used machine learning in prediction of smoking status. Different machine learning approaches were applied and investigated for finding the smoking status. From the results it was ensured that logistic algorithm performs better.
- ❖ R. Lopes et al. applied machine learning approach for the prediction of credit recovery. Credit recovery is a very important issue for banking system. The prediction of credit recovery is a challenging task. Different machine learning approaches were applied to predict the credit recovery and gradient expansion algorithms (GBM) outperformed the other machine learning approaches.

3.2 PROPOSED SYSTEM

- ❖ This proposed model will characterize the behavior of customers on the basis of their record. These records are taken from the customers, and create a data set. With the help of these data sets and training machine learning model, we predict that the customer's loan will pass or not.
- ❖ The aim of this paper is to provide a quick, immediate and easy way to choose the deserving applicants. It can provide special advantages to the bank. The Loan Prediction System can automatically calculate the weight of each feature taking

part in loan processing and on new test data same features are processed with respect to their associated weight. A time limit can be set for the applicant to check whether his/her loan can be sanctioned or not.

- ❖ Loan Prediction System allows jumping to specific application so that it can be check on priority basis. This Paper is exclusively for the managing authority of Bank/finance company, whole process of prediction is done privately no stakeholders would be able to alter the processing.

3.3 SYSTEM ARCHITECTURE

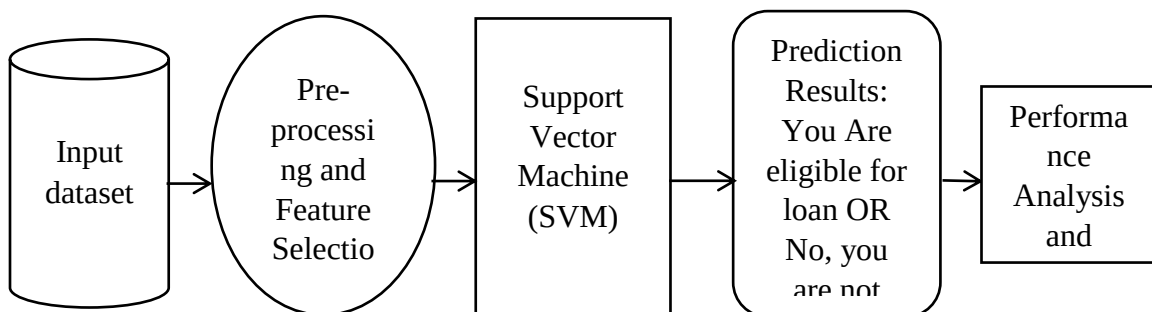


Fig 3.1-Architecture Diagram

3.4 FLOW CHART

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

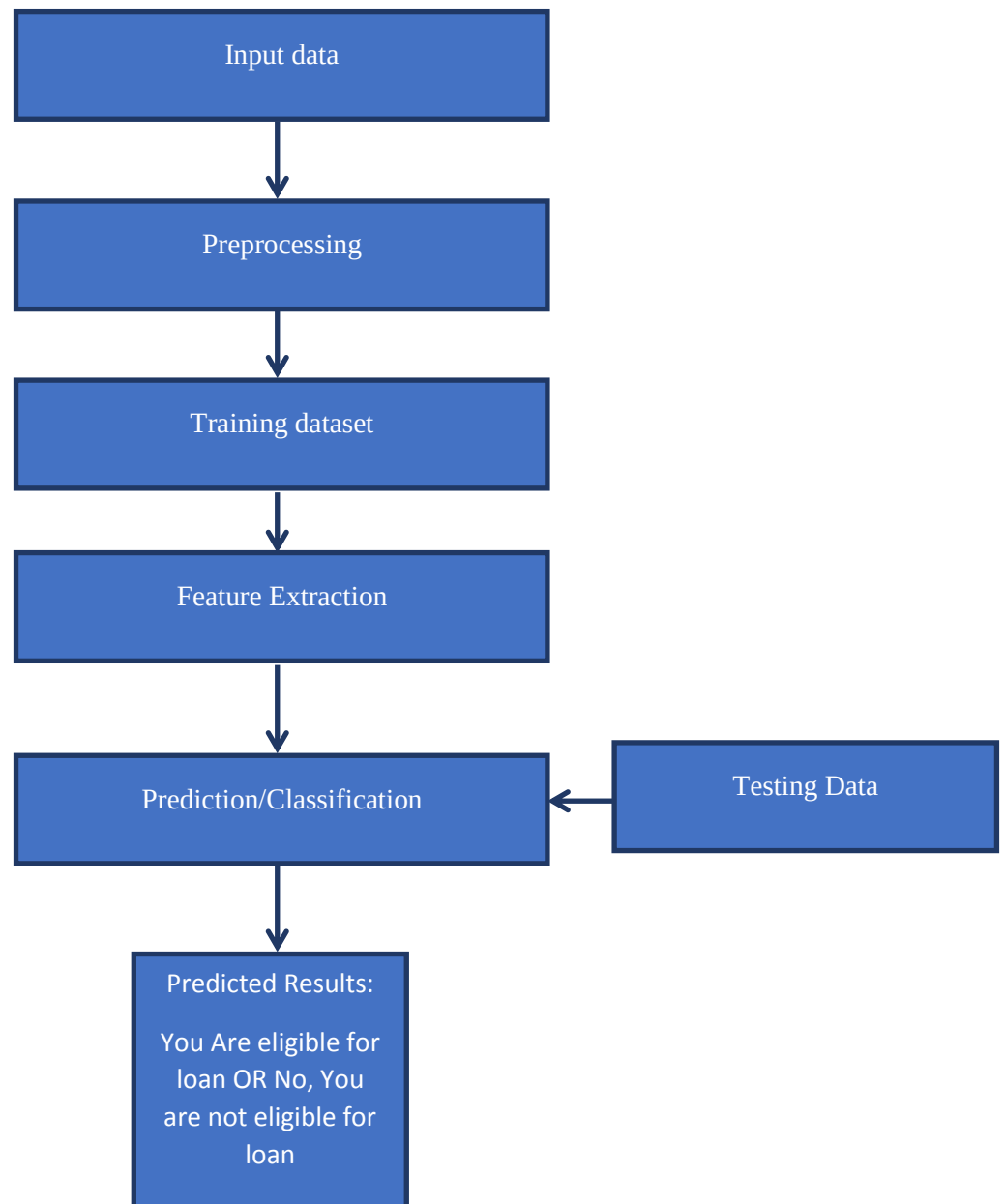


Fig 3.2 - Flow chart

3.5 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

3.5.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is

to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

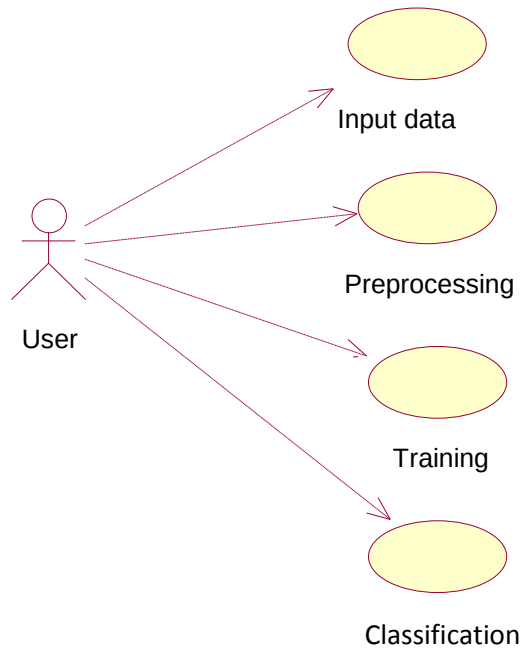


Fig 3.3 - Use case Diagram

3.5.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

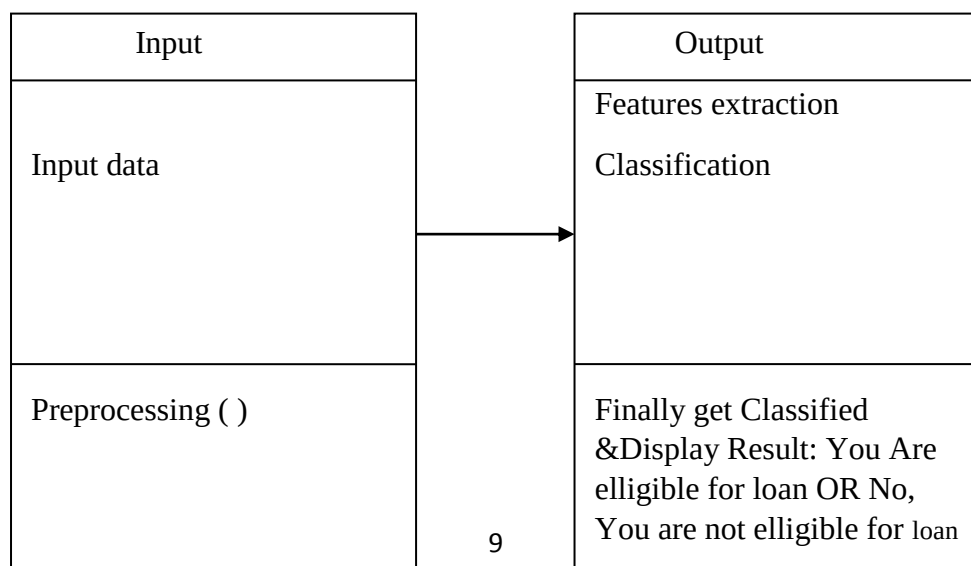


Fig 3.4 - Class Diagram

3.5.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

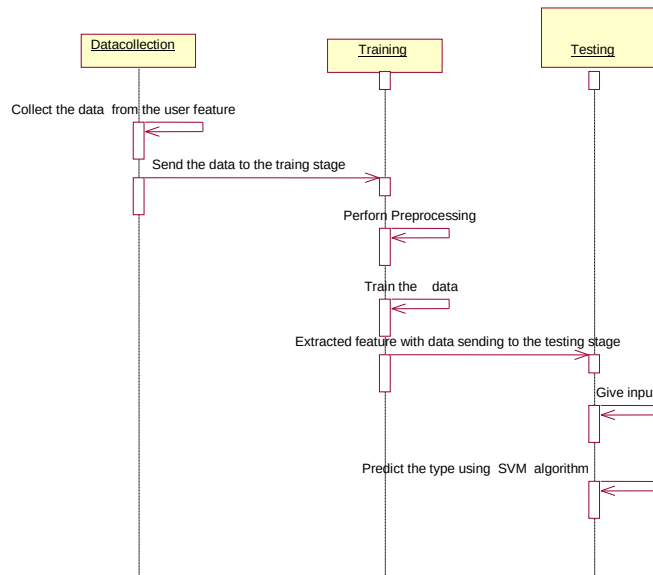


Fig 3.5 - Sequence Diagram

3.5.4 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

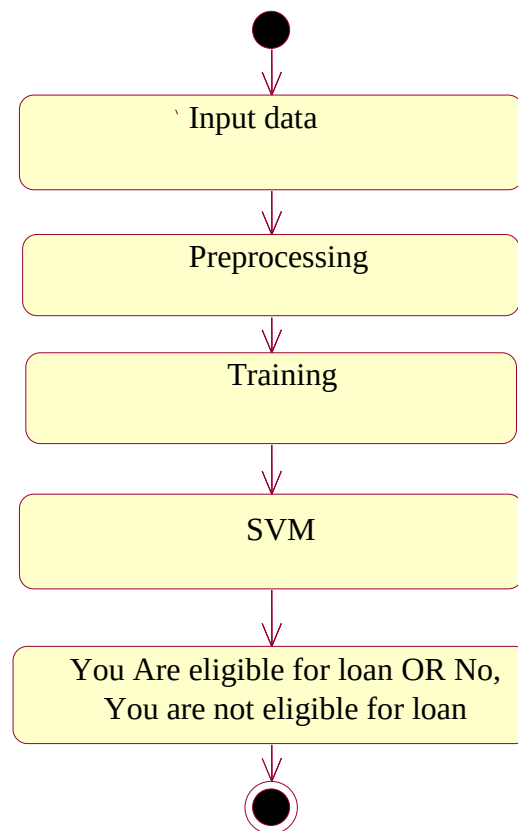


Fig 3.6 - Activity Diagram

3.6 SYSTEM REQUIREMENTS

3.6.1 HARDWARE REQUIREMENTS

- System : Pentium i3 Processor.
- Hard Disk : 500 GB.
- Monitor : 15" LED
- Input Devices : Keyboard, Mouse
- Ram : 4 GB

3.6.2 SOFTWARE REQUIREMENTS

- Operating system : Windows 10.
- Coding Language : Python
- Web Framework : Flask

CHAPTER 4

SOFTWARE MODULES

4.1 PYTHON (PROGRAMMING LANGUAGE)

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

4.1.1 PYTHON HISTORY

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

APPLICATION OF PYTHON

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

FEATURES OF PYTHON

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system is well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

4.2 NUMPY

NumPy (**Numerical Python**) is an opensource Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages.

The NumPy library contains multidimensional array and matrix data structures (you'll find more information about this in later sections). It provides **ndarray**, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

4.2.1 INSTALL NUMPY

To install NumPy, we strongly recommend using a scientific Python distribution. If you're looking for the full instructions for installing NumPy on your operating system

If you already have Python, you can install NumPy with:

```
conda install numpy
```

or

```
pip install numpy
```

If you don't have Python yet, you might want to consider using [Anaconda](#). It's the easiest way to get started. The good thing about getting this distribution is the fact that you don't need to worry too much about separately installing NumPy or any of the major packages that you'll be using for your data analyses, like pandas, Scikit-Learn, etc.

4.2.2 IMPORT NUMPY

To access NumPy and its functions import it in your Python code like this:

```
import numpy as np
```

We shorten the imported name to np for better readability of code using NumPy. This is a widely adopted convention that you should follow so that anyone working with your code can easily understand it.

Reading the example code

If you aren't already comfortable with reading tutorials that contain a lot of code, you might not know how to interpret a code block that looks like this:

```
>>> a = np.arange(6)
>>> a2 = a[np.newaxis, :]
>>> a2.shape
(1, 6)
```

If you aren't familiar with this style, it's very easy to understand. If you see >>>, you're looking at **input**, or the code that you would enter. Everything that doesn't have >>> in front of it is **output**, or the results of running your code. This is the style you see when you run python on the command line, but if you're using IPython, you might see a different style. Note that it is not part of the code and will cause an error

if typed or pasted into the Python shell. It can be safely typed or pasted into the IPython shell; the >>> is ignored.

4.2.3 USE OF NUMPY

NumPy arrays are faster and more compact than Python lists. An array consumes less memory and is convenient to use. NumPy uses much less memory to store data and it provides a mechanism of specifying the data types. This allows the code to be optimized even further.

4.2.4 NUMPY ARRAY

An array is a central data structure of the NumPy library. An array is a grid of values and it contains information about the raw data, how to locate an element, and how to interpret an element. It has a grid of elements that can be indexed in **various ways**. The elements are all of the same type, referred to as the array dtype.

An array can be indexed by a tuple of nonnegative integers, by booleans, by another array, or by integers. The rank of the array is the number of dimensions. The shape of the array is a tuple of integers giving the size of the array along each dimension.

One way we can initialize NumPy arrays is from Python lists, using nested lists for two- or higher-dimensional data.

For example:

```
>>> a = np.array([1, 2, 3, 4, 5, 6])
```

or:

```
>>> a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
```

We can access the elements in the array using square brackets. When you're accessing elements, remember that indexing in NumPy starts at 0. That means that if you want to access the first element in your array, you'll be accessing element "0".

```
>>> print(a[0])  
[1 2 3 4]
```

4.3 MATPLOTLIB

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

MODULES:

- ❖ Data Collection
- ❖ Dataset
- ❖ Data Preparation
- ❖ Model Selection
- ❖ Analyse and Prediction
- ❖ Accuracy on test set
- ❖ Saving the Trained Model

MODULES DESCRIPTION:

4.4.1 Data Collection:

This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get, the better our model will perform.

There are several techniques to collect the data, like web scraping, manual interventions and etc.

Prediction of Modernized Loan Approval System Based on Machine Learning Approach

We give the data set in the project folder

4.4.2 Dataset:

The dataset consists of 511 individual data. There are 10 columns in the dataset, which are described below.

1. **Application ID:** Unique Loan ID
2. **Gender:** Male/ Female
3. **Married:** Applicant married (Y/N)
4. **Dependents:** Number of dependents
5. **Education:** Applicant Education (Graduate/ Undergraduate)
6. **Self-employed:** Self-employed (Y/N)
7. **Credit History:** credit history meets guidelines
8. **Property Area:** Urban/ Semi Urban/ Rural
9. **Income:** Applicant income
10. **Application Status:** (Target) Loan approved (Y/N)

4.4.3 Data Preparation:

Wrangle data and prepare it for training. Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.)

Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data

Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis

Split into training and evaluation sets

4.4.4 Model Selection:

We used support vector machine algorithm, We got a accuracy of 0.82 test set so we implemented this algorithm.

4.4.5 Analyse and Prediction:

In the actual dataset, we chose only 7 features:

1. **Gender:** Male/ Female
2. **Married:** Applicant married (Y/N)

3. **Education:** Applicant Education (Graduate/ Undergraduate)
4. **Self-employed:** Self-employed (Y/N)
5. **Credit History:** credit history meets guidelines
6. **Property Area:** Urban/ Semi Urban/ Rural
7. **Income:** Applicant income
8. **Application Status:** (Target) Loan approved (Y/N)

4.4.6 Accuracy on test set:

We got an accuracy of 0.82% on test set.

4.4.7 Saving the Trained Model

Once you're confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle.

Make sure you have pickle installed in your environment.

Next, let's import the module and dump the model into.pkl file

4.5 PROPOSED ALGORITHM

4.5.1 Support Vector Machine (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

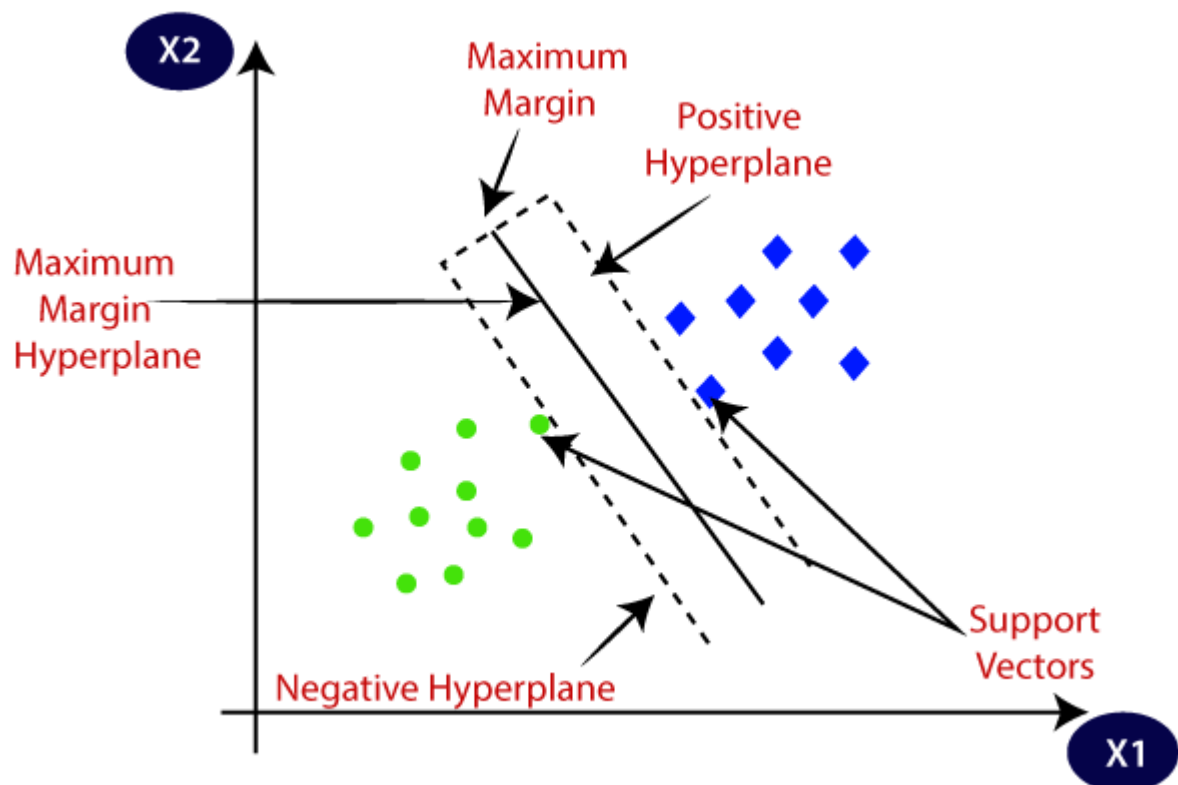


Fig 4.1 - SVM

SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog.

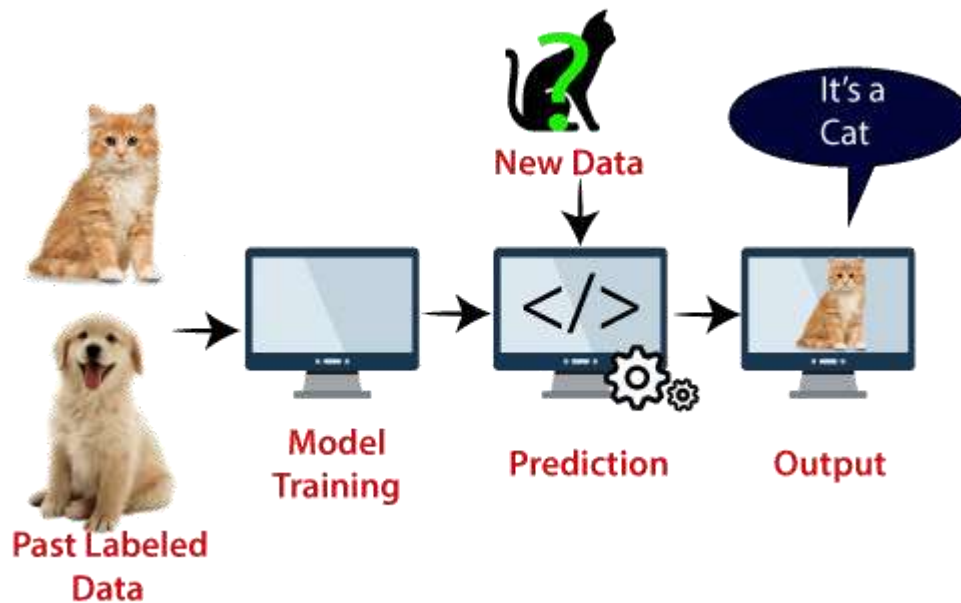


Fig 4.2- Classification of labeled data

Types of SVM

SVM can be of two types:

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

How does SVM works?

Linear SVM:

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x_1 and x_2 . We want a classifier that can classify the pair(x_1 , x_2) of coordinates in either green or blue. Consider the below image:

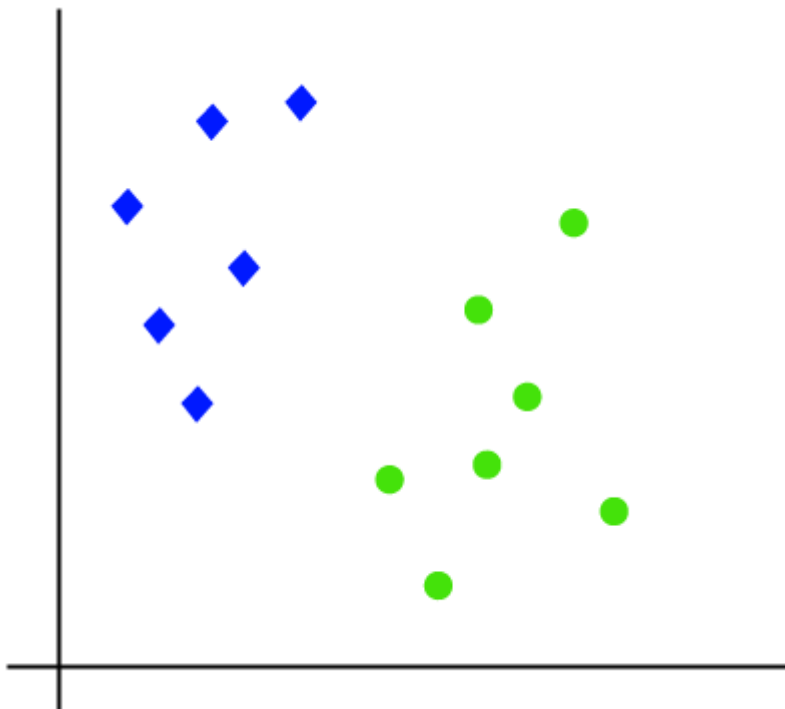


Fig 4.3 - Linear SVM

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

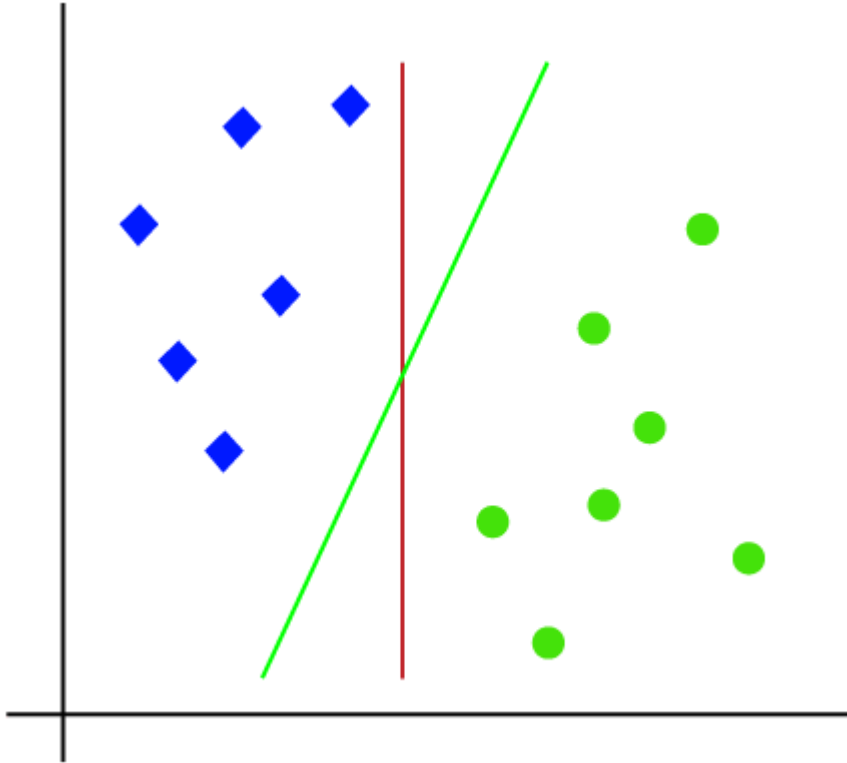


Fig 4.4 - Separation of classes

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.

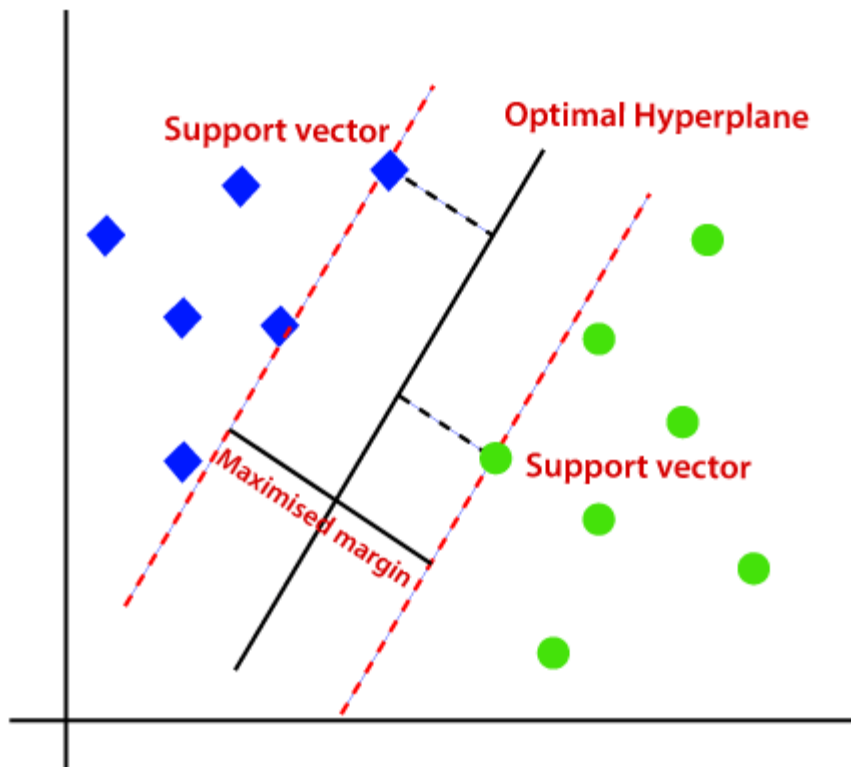


Fig 4.5 - Finding Hyperplane

Non-Linear SVM:

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:

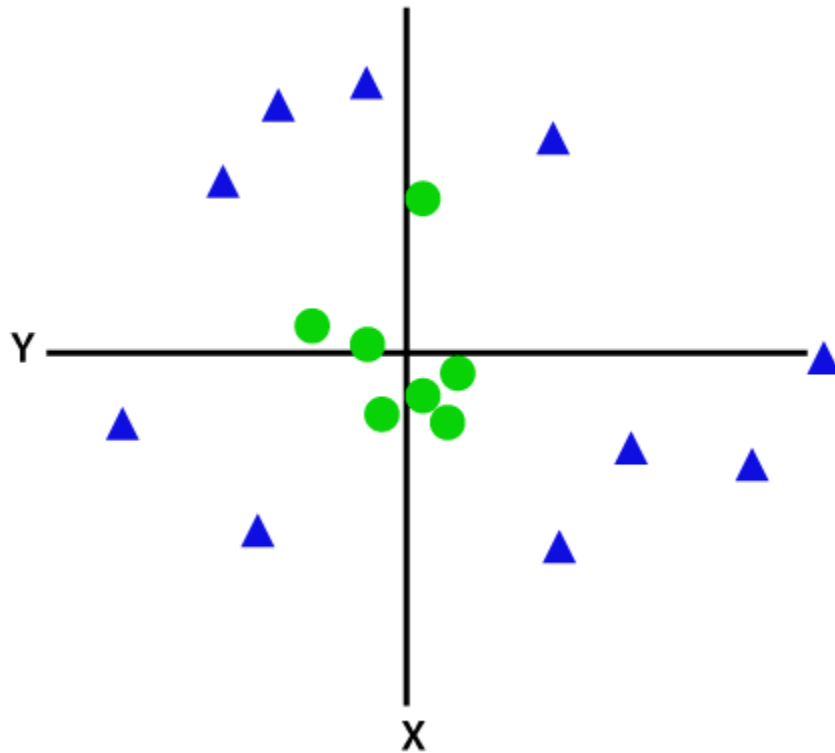


Fig 4.6 - Non Linear SVM

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$$z=x^2+y^2$$

By adding the third dimension, the sample space will become as below image:

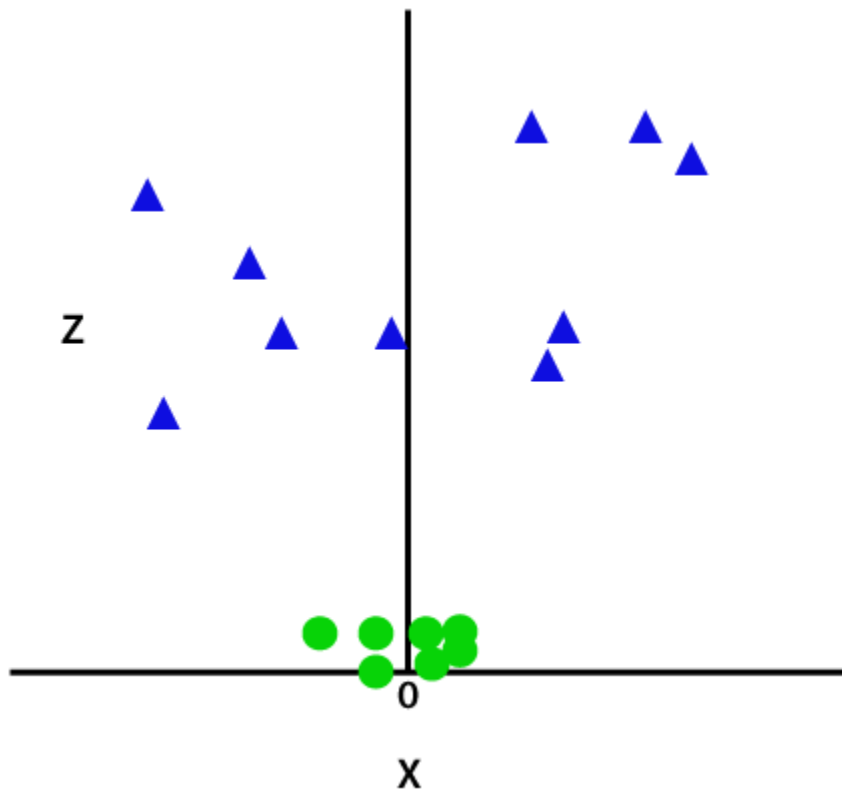


Fig 4.7 - Data separation of non linear data

So now, SVM will divide the datasets into classes in the following way. Consider the below image:

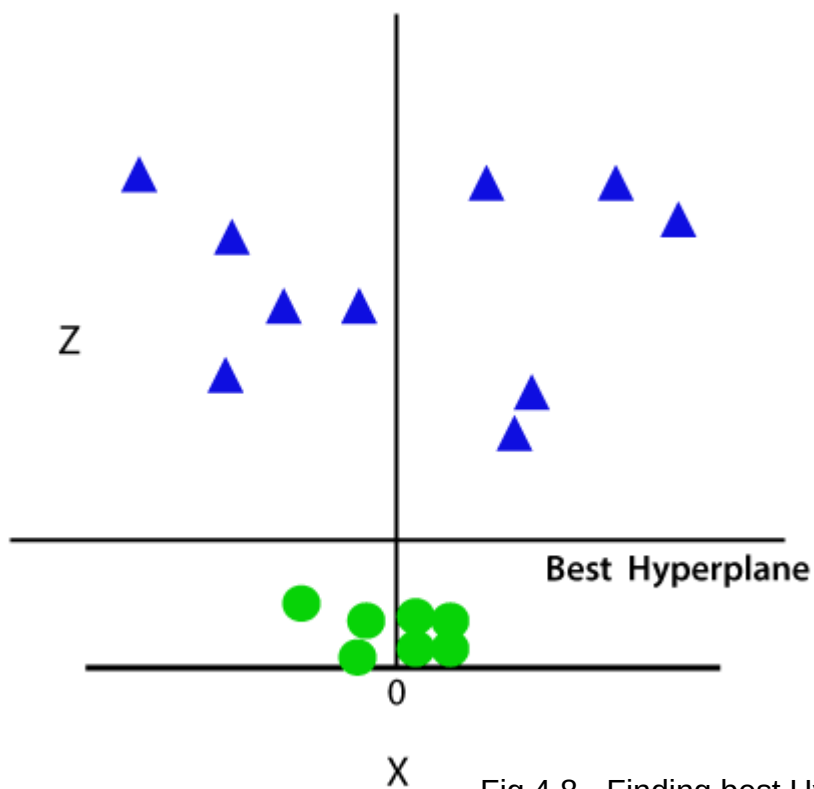


Fig 4.8 - Finding best Hyperplane

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with $z=1$, then it will become as:

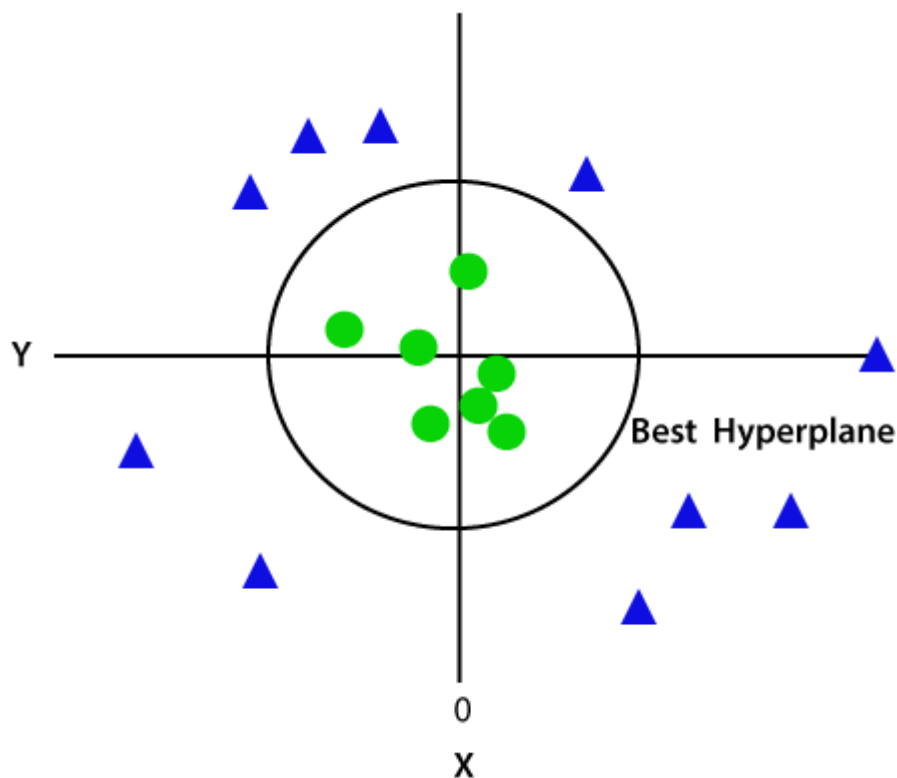


Fig 4.9 - Finding radius of non linear data

Hence we get a circumference of radius 1 in case of non-linear data.

Python Implementation of Support Vector Machine

Now we will implement the SVM algorithm using Python. Here we will use the same dataset **user_data**, which we have used in Logistic regression and KNN classification.

- **Data Pre-processing step**

Till the Data pre-processing step, the code will remain the same. Below is the code:

1. #Data Pre-processing Step
2. # importing libraries
3. **import** numpy as nm
4. **import** matplotlib.pyplot as mtp

5. **import** pandas as pd
- 6.
7. #importing datasets
8. data_set= pd.read_csv('user_data.csv')
- 9.
10. #Extracting Independent and dependent Variable
11. x= data_set.iloc[:, [2,3]].values
12. y= data_set.iloc[:, 4].values
- 13.
14. # Splitting the dataset into training and test set.
15. from sklearn.model_selection **import** train_test_split
16. x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_s
tate=0)
17. #feature Scaling
18. from sklearn.preprocessing **import** StandardScaler
19. st_x= StandardScaler()
20. x_train= st_x.fit_transform(x_train)
21. x_test= st_x.transform(x_test)

After executing the above code, we will pre-process the data. The code will give the dataset as:

Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0

Format Resize ☒ Background color ☒ Column min/max Save and Close Close

Fig 4.10 - Data set

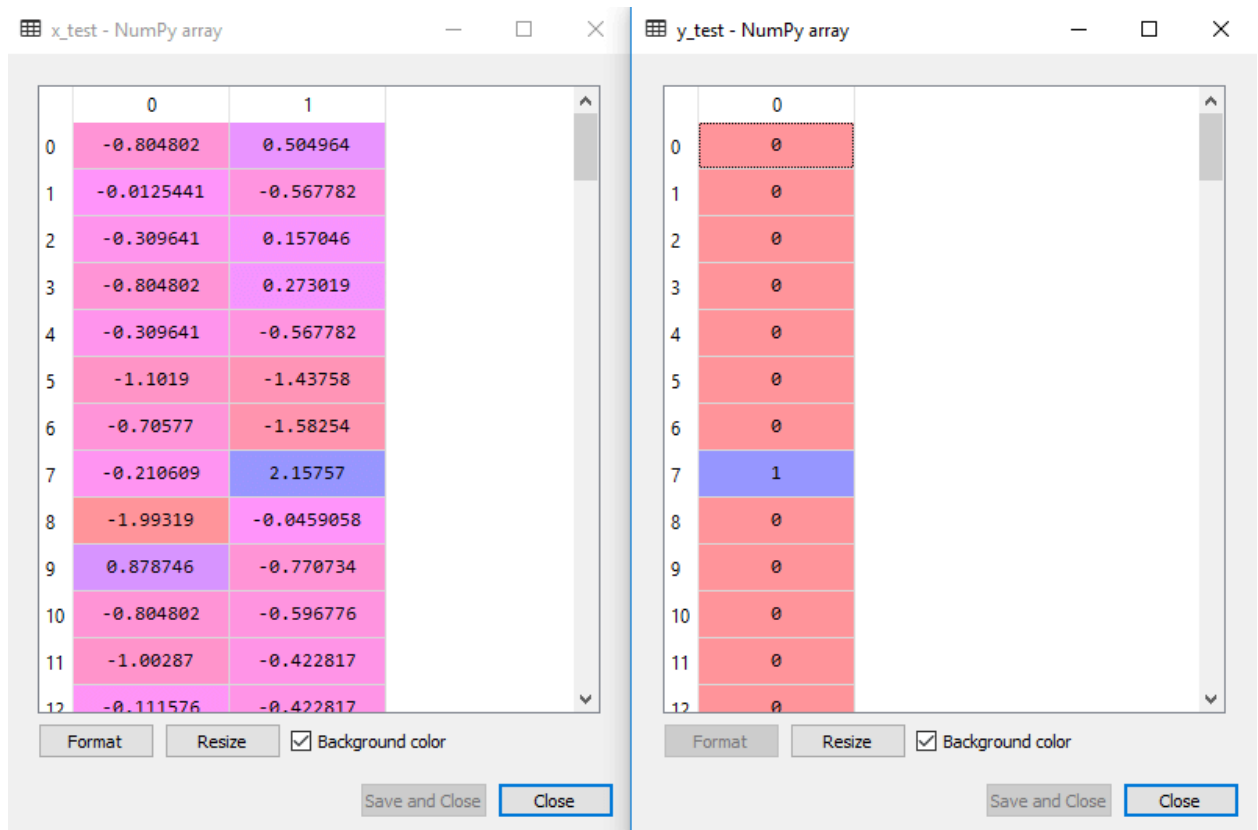


Fig 4.11 -The scaled output for the test set will

Fitting the SVM classifier to the training set:

Now the training set will be fitted to the SVM classifier. To create the SVM classifier, we will import **SVC** class from **Sklearn.svm** library. Below is the code for it:

1. from sklearn.svm **import** SVC # "Support vector classifier"
2. classifier = SVC(kernel='linear', random_state=0)
3. classifier.fit(x_train, y_train)

In the above code, we have used **kernel='linear'**, as here we are creating SVM for linearly separable data. However, we can change it for non-linear data. And then we fitted the classifier to the training dataset(x_train, y_train)

Output:

```
Out[8]:
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
```

```
kernel='linear', max_iter=-1, probability=False, random_state=0,  
shrinking=True, tol=0.001, verbose=False)
```

Fig 4.12 - Fitting the SVM classifier to the training set.

The model performance can be altered by changing the value of **C(Regularization factor), gamma, and kernel**.

- **Predicting the test set result:**

Now, we will predict the output for test set. For this, we will create a new vector `y_pred`. Below is the code for it:

1. `#Predicting the test set result`
2. `y_pred= classifier.predict(x_test)`

After getting the `y_pred` vector, we can compare the result of **y_pred** and **y_test** to check the difference between the actual value and predicted value.

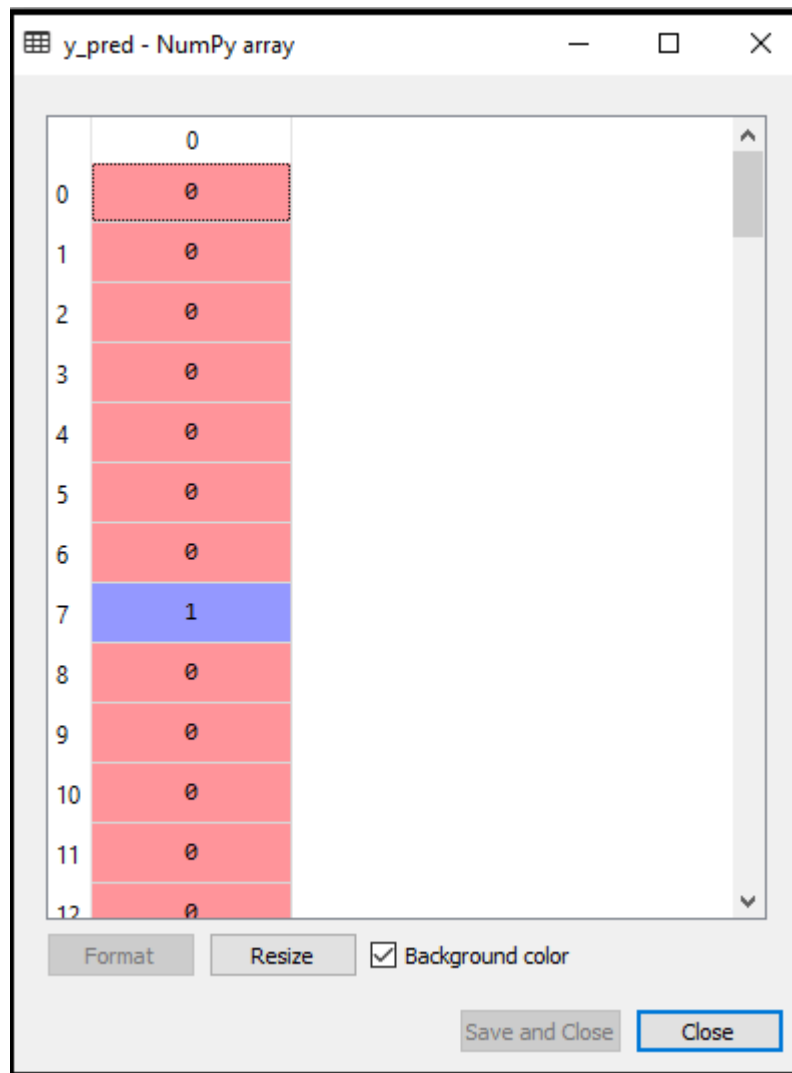


Fig 4.13 - output for the prediction of the test set

CREATING THE CONFUSION MATRIX

Now we will see the performance of the SVM classifier that how many incorrect predictions are there as compared to the Logistic regression classifier. To create the confusion matrix, we need to import the **confusion_matrix** function of the sklearn library. After importing the function, we will call it using a new variable **cm**. The function takes two parameters, mainly **y_true**(the actual values) and **y_pred** (the targeted value return by the classifier). Below is the code for it:

1. #Creating the Confusion matrix
2. from sklearn.metrics **import** confusion_matrix
3. cm= confusion_matrix(y_test, y_pred)

Output:

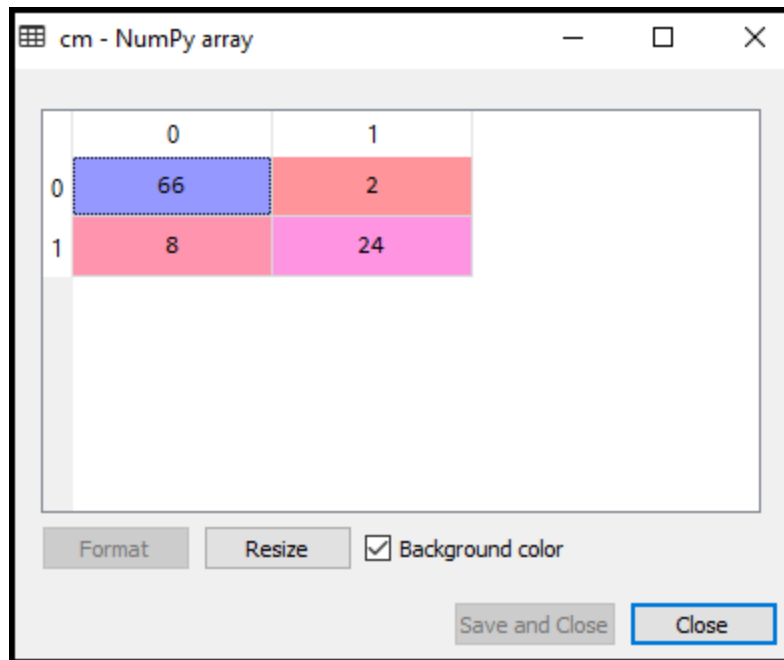


Fig 4.14 - CREATING THE CONFUSION MATRIX.

As we can see in the above output image, there are $66+24=90$ correct predictions and $8+2=10$ correct predictions. Therefore we can say that our SVM model improved as compared to the Logistic regression model.

- **Visualizing the training set result:**

Now we will visualize the training set result, below is the code for it:

1. `from matplotlib.colors import ListedColormap`
2. `x_set, y_set = x_train, y_train`
3. `x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01),`
`nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))`
5. `mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),`
`alpha = 0.75, cmap = ListedColormap(('red', 'green')))`
7. `mtp.xlim(x1.min(), x1.max())`
8. `mtp.ylim(x2.min(), x2.max())`
9. `for i, j in enumerate(nm.unique(y_set)):`
10. `mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],`

```

11.     c = ListedColormap(('red', 'green'))(i), label = j)
12. mtp.title('SVM classifier (Training set)')
13. mtp.xlabel('Age')
14. mtp.ylabel('Estimated Salary')
15. mtp.legend()
16. mtp.show()

```

Output:

By executing the above code, we will get the output as:

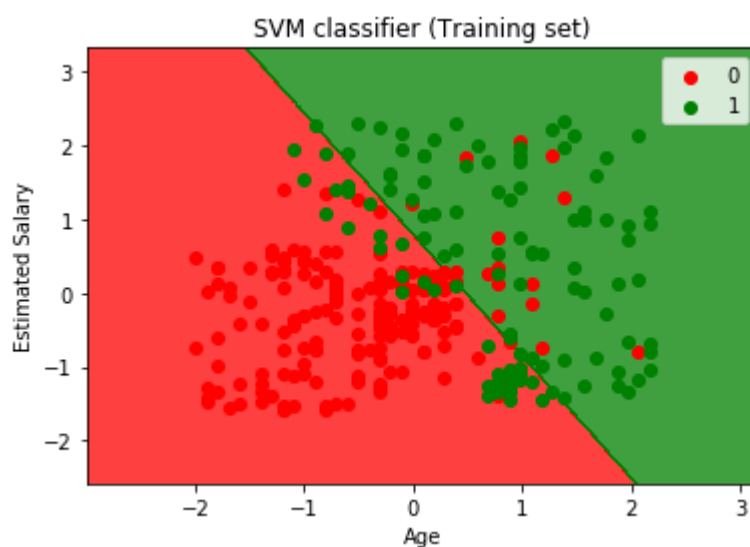


Fig 4.15 - SVM classifier

As we can see, the above output is appearing similar to the Logistic regression output. In the output, we got the straight line as hyperplane because we have **used a linear kernel in the classifier**. And we have also discussed above that for the 2d space, the hyperplane in SVM is a straight line.

○ Visualizing the test set result:

```

1. #Visulaizing the test set result
2. from matplotlib.colors import ListedColormap
3. x_set, y_set = x_test, y_test
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() -
    1, stop = x_set[:, 0].max() + 1, step =0.01),

```

```

5. nm.arange(start = x_set[:, 1].min() -
    1, stop = x_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).resh
    ape(x1.shape),
7. alpha = 0.75, cmap = ListedColormap(('red','green' )))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. for i, j in enumerate(nm.unique(y_set)):
11.     mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12.         c = ListedColormap(('red', 'green'))(i), label = j)
13. mtp.title('SVM classifier (Test set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()

```

Output:

By executing the above code, we will get the output as:

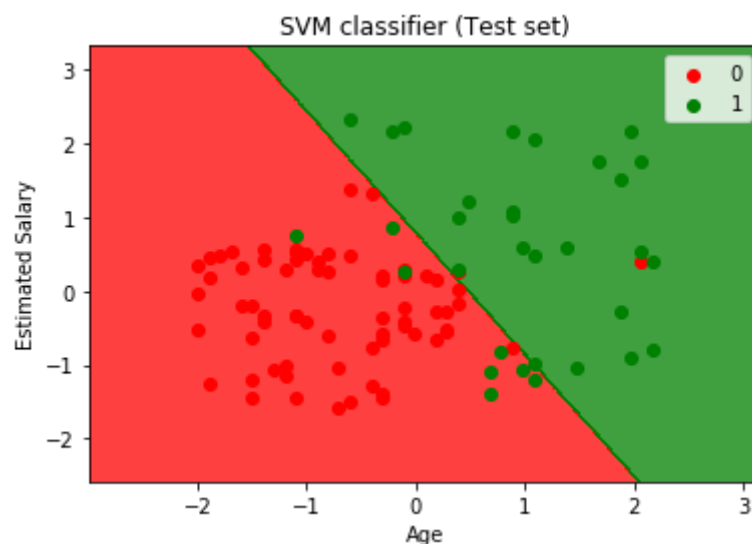


Fig 4.16 - SVM classifier (Test set).

As we can see in the above output image, the SVM classifier has divided the users into two regions (Purchased or Not purchased). Users who purchased the SUV are

in the red region with the red scatter points. And users who did not purchase the SUV are in the green region with green scatter points. The hyperplane has divided the two classes into Purchased and not purchased variable.

4.6 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as

specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

This paper applied machine learning in prediction of loan approval. Support Vector Machine (SVM) is used to predict the loan approval status of customers for bank loans. The results shown that the prediction accuracy is Train Accuracy: 80% and Test Accuracy: 82%. According to this research paper prediction accuracy is good when compared to other existing systems. In some situations like client going through some disaster so here the algorithm cannot predict the appropriate result. This research paper can find out the client is potential or not.

5.2 Future Enhancement

In future a rigorous analysis of other machine learning algorithms other than these can also be done in future to investigate the power of machine learning algorithms for loan approval prediction.

REFERENCE

- 1] Ashwini S. Kadam, Shraddha R Nikam, Ankita A. Aher, Gayatri V. Shelke, Amar S. Chandgude. "Prediction for Loan Approval using Machine Learning Algorithm", Apr 2021 International Research Journal of Engineering and Technology (IRJET)
- [2] Mohammad Ahmad Sheikh, Amit Kumar Goel, Tapas Kumar. "An Approach for Prediction of Loan Approval using Machine Learning Algorithm", 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020
- [3] X.FrencyJensy, V.P.Sumathi, Janani Shiva Shri, "An exploratory Data Analysis for Loan Prediction based on nature of clients", International Journal of Recent Technology and Engineering (IJRTE), Volume-7 Issue-4S, November 2018
- [4] J. Tejaswini¹, T. Mohana Kavya, R. Devi Naga Ramya, P. Sai Triveni Venkata Rao Maddumala. "ACCURATE LOAN APPROVAL PREDICTION BASED ON MACHINE LEARNING APPROACH" Vol 11, www.jespublication.com, page 523 Issue 4, April/ 2020 ISSN NO: 0377-9254
- [5] "Prediction for Loan Approval using Machine Learning Algorithm" International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 08 Issue: 04 | Apr 2021 www.irjet.net p-ISSN: 2395-0072
- [6] Vaidya, "Predictive and probabilistic approach using logistic regression: Application to prediction of loan approval," 2017 8th International Conference on

Computing, Communication and Networking Technologies (ICCCNT), Delhi, 2017, pp. 1-6.doi: 10.1109/ICCCNT.2017.8203946

[7] M. Bayraktar, M. S. Aktaş, O. Kalıpsız, O. Susuz and S. Bayracı, "Credit risk analysis with classification Restricted Boltzmann Machine," 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, 2018, pp. 1-4.doi: 10.1109/SIU.2018.840 4397

[8] Y. Shi and P. Song, "Improvement Research on the Project Loan Evaluation of Commercial Bank Based on the Risk Analysis," 2017 10th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, 2017, pp. 3-6.doi: 10.1109/ISCID.2017.60

[9] V. C. T. Chan et al., "Designing a Credit Approval System Using Web Services, BPEL, and AJAX," 2009 IEEE International Conference on e-Business Engineering, Macau, 2009, pp. 287- 294.doi: 10.1109/ICEBE.2009.46

5.3. APPENDICES

A. SAMPLE CODE

```
import numpy as np

import pandas as pd

from flask import Flask, request, jsonify, render_template, redirect, flash, send_file

from sklearn.preprocessing import MinMaxScaler

from werkzeug.utils import secure_filename

import pickle

import numpy as np
```

```

import pandas as pd

from flask import Flask, request, jsonify, render_template, redirect, flash, send_file

from sklearn.preprocessing import MinMaxScaler

from werkzeug.utils import secure_filename

import pickle

import numpy as np

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler

app = Flask(__name__) #Initialize the flask App


svm = pickle.load(open('loan.pkl','rb'))

@app.route('/')

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

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

#@app.route('/future')

```

```

#def future():
    #return render_template('future.html')

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

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

@app.route('/preview',methods=["POST"])
def preview():
    if request.method == 'POST':
        dataset = request.files['datasetfile']
        df = pd.read_csv(dataset,encoding = 'unicode_escape')
        df.set_index('Id', inplace=True)
        return render_template("preview.html",df_view = df)

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

@app.route('/prediction', methods = ['GET', 'POST'])
def prediction():
    return render_template('prediction.html')

```

```

#@app.route('/upload')

#def upload_file():

#    return render_template('BatchPredict.html')


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

def predict():

    int_feature = [x for x in request.form.values()]

    final_features = [np.array(int_feature)]

    result=svm.predict(final_features)

    return render_template('prediction.html', prediction_text= result[0])


@app.route('/performance')

def performance():

    return render_template('performance.html')


if __name__ == "__main__":

    app.run(debug=True)

```

B.SCREENSHOTS

HOMEPAGE

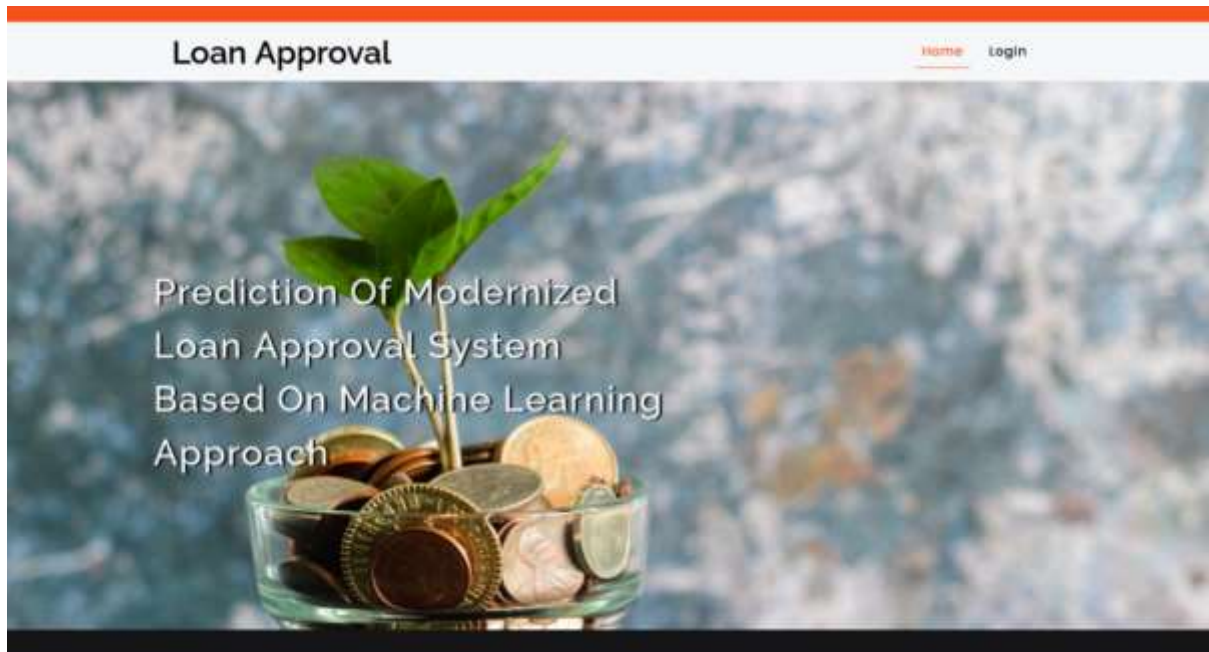


Fig 5.1-Home page

LOGIN PAGE

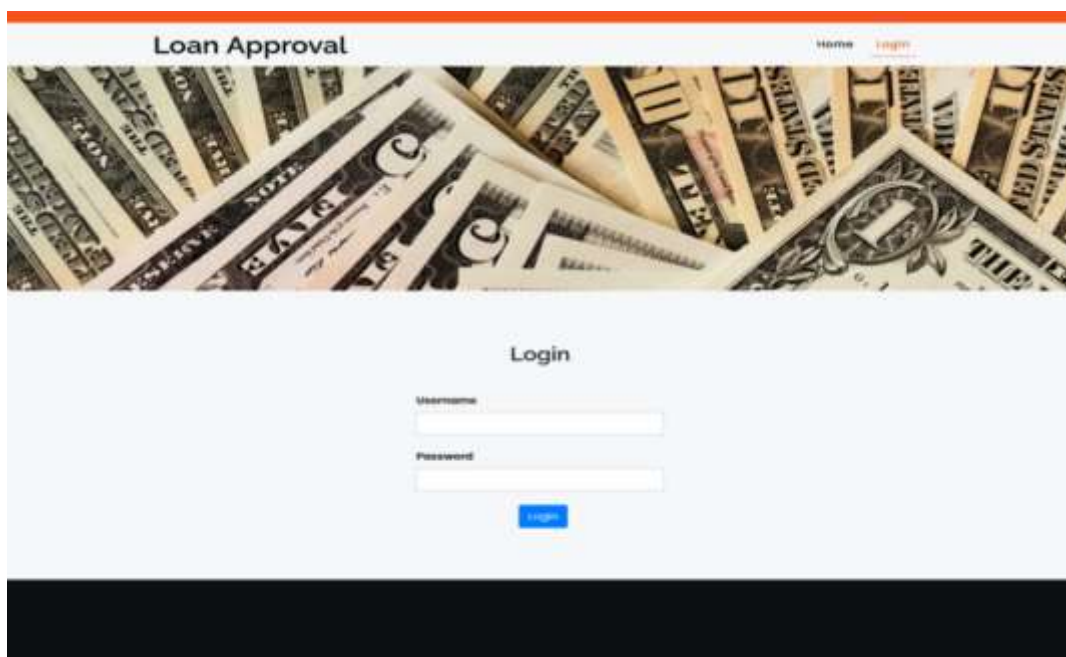


Fig 5.2 - Login page

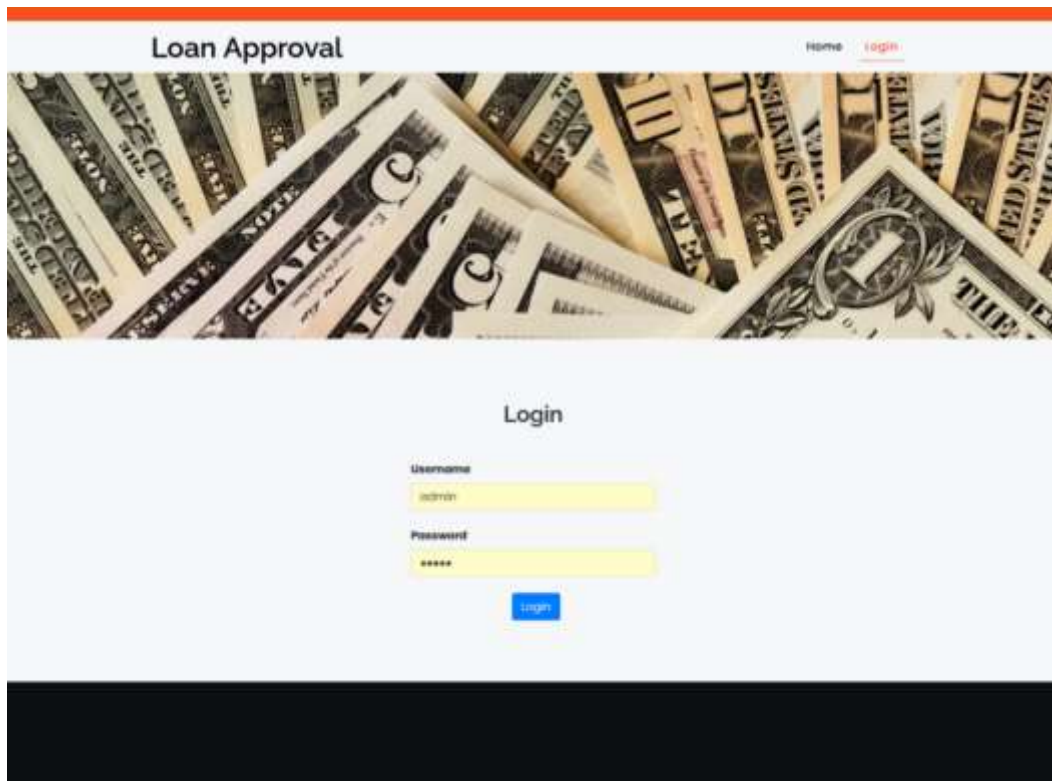


Fig 5.3 - Entering Credentials

UPLOAD PAGE

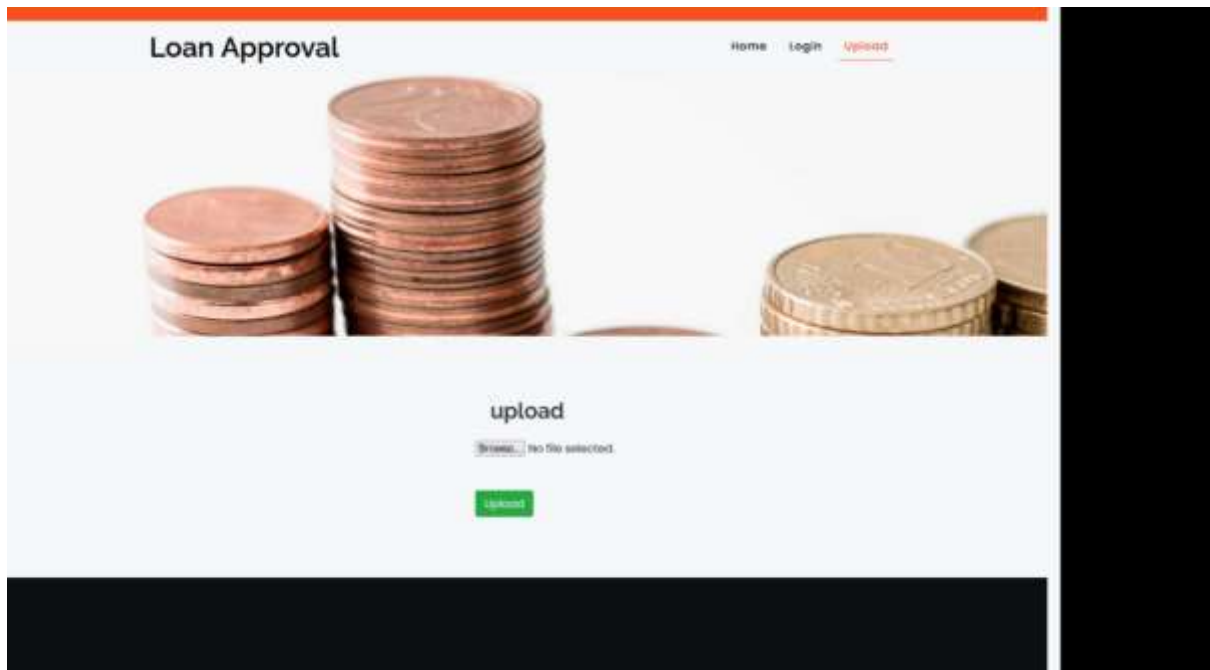


Fig 5.4- Upload page

Uploaded page

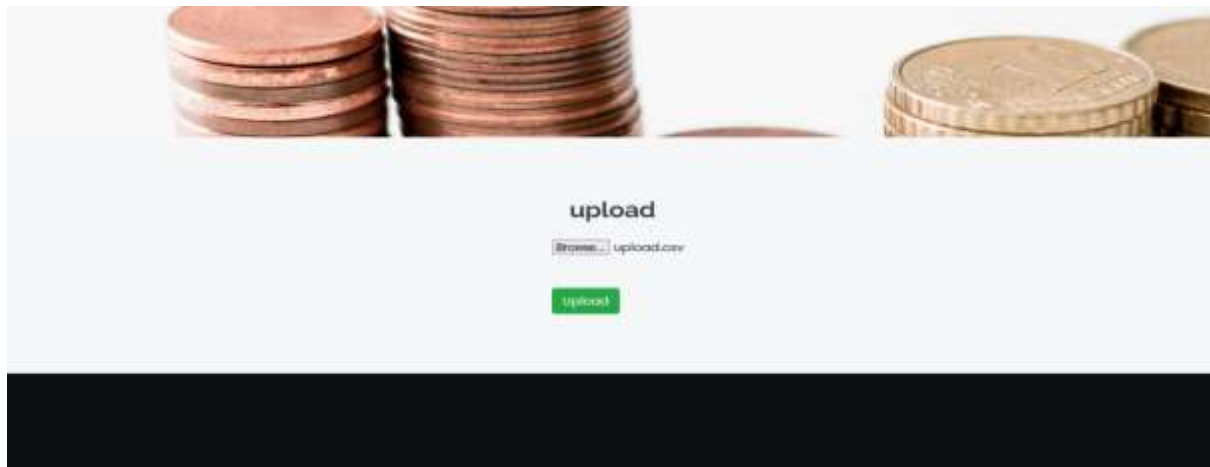


Fig 5.5 - uploaded page.

Preview page

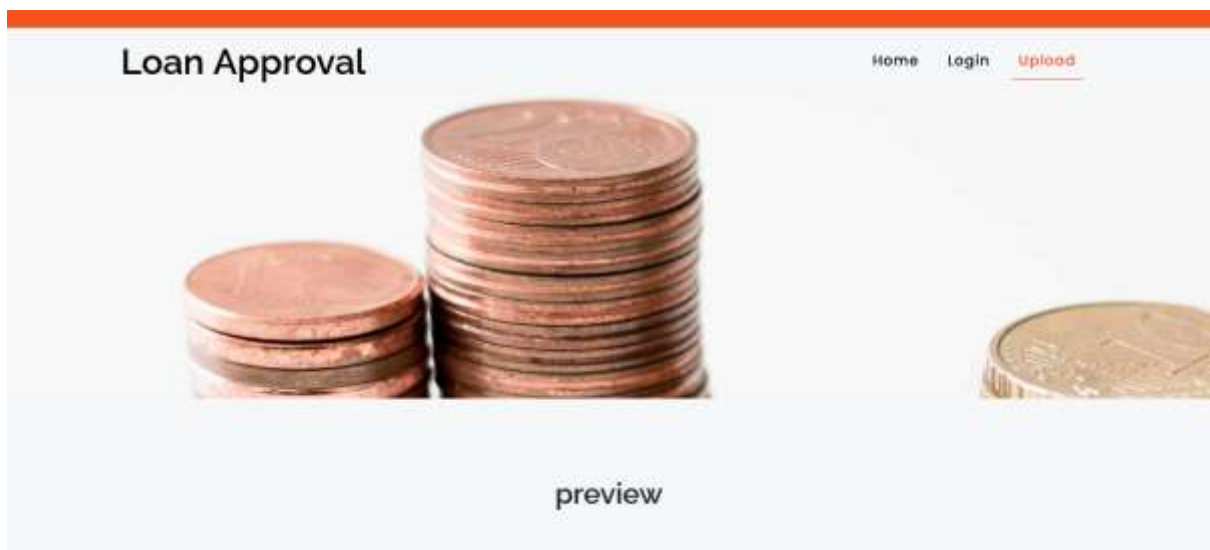


Fig 5.6 - Preview of data set

preview

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Income	Application_Status
16									
1	Male	No	0	Graduate	No	1	Urban	medium	Y
2	Male	Yes	1	Graduate	No	1	Rural	medium	N
3	Male	Yes	0	Graduate	Yes	1	Urban	low	Y
4	Male	Yes	0	Not Graduate	No	1	Urban	low	Y
5	Male	No	0	Graduate	No	1	Urban	medium	Y
6	Male	Yes	2	Graduate	Yes	1	Urban	medium	Y
7	Male	Yes	0	Not Graduate	No	1	Urban	low	Y
S09	Male	Yes	1	Graduate	No	1	Urban	medium	Y
S10	Male	Yes	2	Graduate	No	1	Urban	medium	Y
S11	Female	No	0	Graduate	Yes	0	Suburban	medium	N

[Click to Train | Test](#)

Fig 5.7 - training data set

Prediction Page

Loan Approval

[Home](#)
[login](#)
[upload](#)
[prediction](#)
[Performance_analysis](#)



Loan Approval prediction

Gender:

Married:

Education:

Self_Employed:

Credit_History:

Property_Area:

Income:

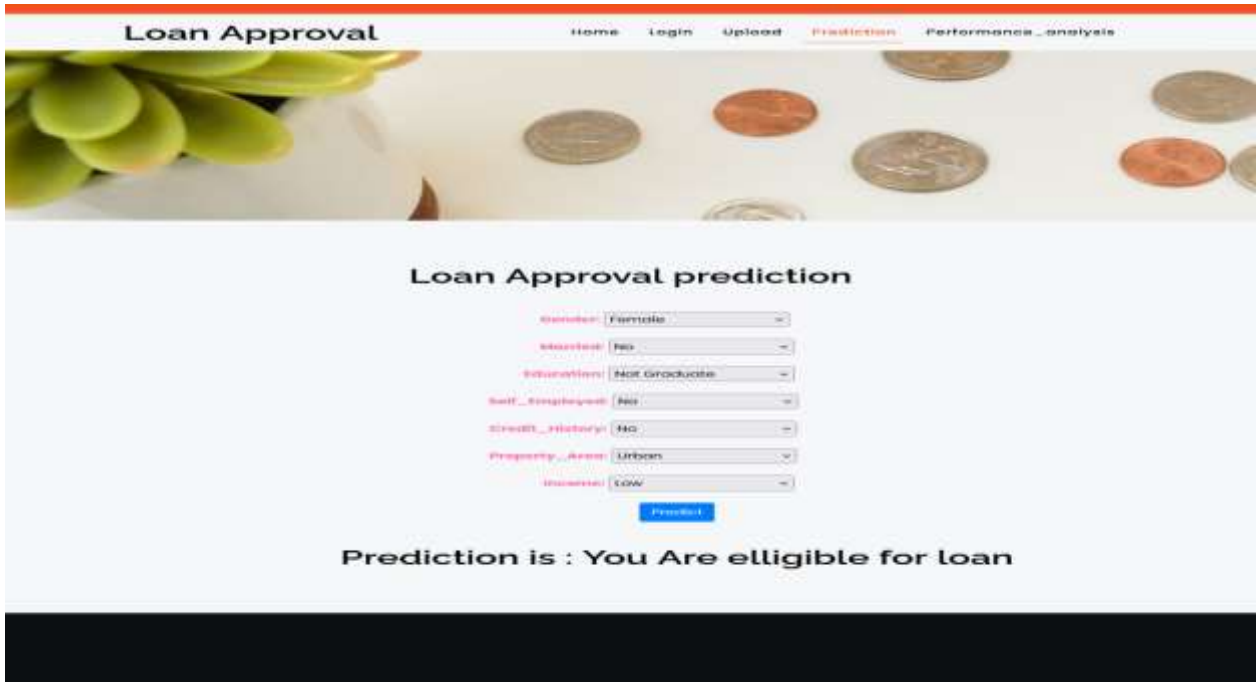
[predict](#)

Prediction is :

Fig 5.8 - Prediction Page

In the prediction page we have to provide the necessary details required to predict the loan approval. Here in this page internally it will predict the loan approval based on the details provided and by using the machine learning techniques it will predict the loan approval whether it is eligible or not.

Loan approval prediction result page



The screenshot displays the 'Loan Approval prediction' interface. At the top, there is a navigation bar with links: Home, Login, Upload, Prediction (highlighted), and Performance_analysis. Below the navigation bar is a header image showing green leaves and coins. The main content area is titled 'Loan Approval prediction' and contains a form with the following fields and values:

Field	Value
Gender	Female
MaritalStatus	No
Education	Not Graduate
Self_Employment	No
Credit_History	No
Property_Area	Urban
Income	LOW

Below the form is a blue 'Predict' button. The result of the prediction is displayed as: 'Prediction is : You Are elligible for loan'.

Fig 5.9 - Prediction result

Performance analysis

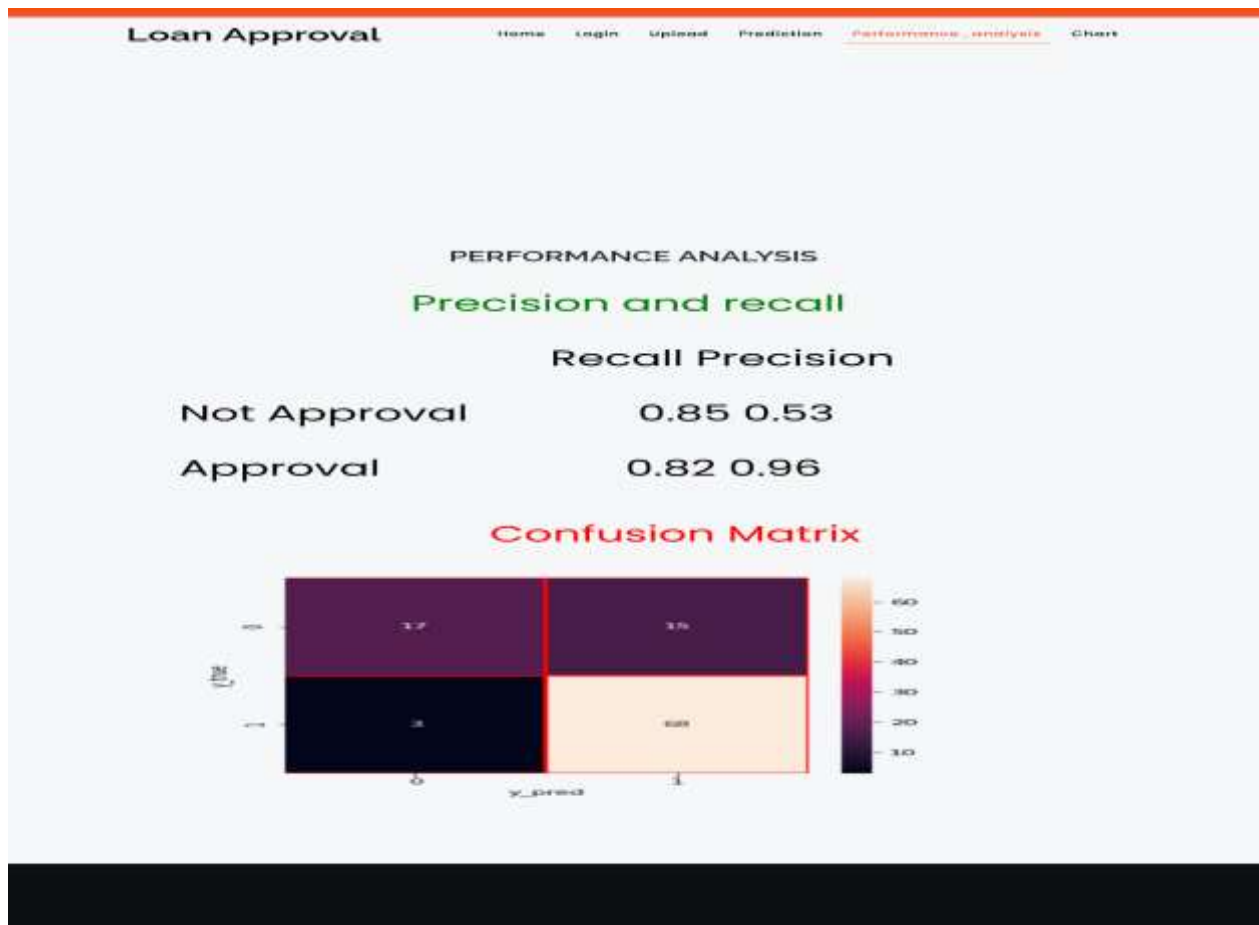


Fig 5.10 -Performance analysis

Chart page



Fig 5.11 - Chart Representation of Data set.