



**SCHOOL OF COMPUTING SCIENCES
DEPARTMENT OF INFORMATION
TECHNOLOGY**



Loan Eligibility Prediction

Submitted by

Prem Kumar. M (20139119)

Franklin. S (20139107)

Safeeq Rahman. A (20139121)

Under the guidance of

Dr. Sowmya Jagadeesan

A PROJECT REPORT

2022- 2023

BONAFIDE CERTIFICATE

Certified that this project report “**Loan Eligibility Prediction**” is the Bonafide work of **Prem Kumar. M (20139119), Franklin.s (20139107), Safeeq Rahman.A (20139121)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature

DR. SOWMYA JAGADEESAN

Senior Faculty

Department of Computer Application

VISTAS, Chennai

Signature

DR. P. SUJATHA

Head of The Department

Department of Computer Application

VISTAS, Chennai

ACKNOWLEDGEMENT

We thank the Almighty GOD for the abundant blessings showered on us. We extend our deepest love and gratitude to our dear parents who built up our career and backed us up in life.

We feel thankful to the Head of the Department **Dr. P. Sujatha**, Department of Information Technology, VISTAS, for all her encouragement, which has sustained our efforts.

We express our deepest gratitude to the internal guide **Dr. Sowmya Jagadeesan**, Department of Information Technology, VISTAS , for her valuable guidance, ideas and support.

We extend our sincere thanks to the project coordinator **Dr. S. Dhanaraj**, for his guidance and support.

We would like to thank all other faculty members of Department of Information Technology for their help and advice throughout our life in this campus.

Finally, we are thankful to all our friends and all others who encouraged us and helped us in doing this project.

TABLE OF CONTENTS

ABSTRACT	Page No.
1. SYSTEM STUDY	
1.1 Feasibility Study	- 7
1.1.1 Economical Feasibility	- 8
1.1.2 Technical Feasibility	- 9
1.1.3 Social Feasibility	- 9
1.2 System Requirements	
1.2.1 Hardware Requirements	- 10
1.2.2 Software Requirements	- 10
2. SYSTEM ANALYSIS	
2.1 Introduction	- 11
2.2 Project Description	- 12
2.3 Modules	- 13
2.4 Existing System	- 26
2.4.1 Drawbacks of Existing System	- 27
2.5 Proposed system	- 27
2.5.1 Advantage of Proposed System	- 27
3. SYSTEM DESIGN	
3.1 Data flow Diagram	- 28
3.2 Overall Diagram	- 29
3.3 Case Diagram	- 30
3.4 Class Diagram	- 30
3.5 Architecture Diagram	- 31
3.6 Source Code	- 31

4. TESTING

4.1	Unit Testing	- 41
4.2	Integration Testing	- 41
4.3	Validation Testing	- 42
4.4	Acceptance Testing	- 42

5. SYSTEM TESTING & IMPLEMENTATION

5.1	Introduction	- 44
5.2	Algorithm used	- 45
5.3	Training the data	- 47
5.4	Testing the data	- 48
5.5	Software Environment	- 50

6. RESULTS & CONCLUSIONS

6.1	Results	- 57
6.2	Conclusion	- 58
6.3	Future Enhancements	- 59

REFERENCES	- 61
------------	------

ABSTRACT

The loan eligibility prediction project aims to predict whether a person is eligible for a loan or not based on their personal and financial information. This project is important because it can help banks and financial institutions streamline their loan approval process and reduce the risk of default.

The dataset used in this project was collected from Kaggle and underwent a preprocessing stage to handle missing values and convert categorical data into numerical data for easier analysis. The data was then analyzed and visualized using the Seaborn library.

The SVM (Support Vector Machine) algorithm was used to train the model and predict loan eligibility. The model was evaluated on a test dataset to ensure its accuracy.

This project has potential applications in the banking and finance industry, and can help improve the loan approval process for both lenders and borrowers.

CHAPTER -1

SYSTEM STUDY

1.1 Feasibility Study

The Feasibility study is an important aspect of any project to determine whether it is viable or not. In the case of the loan eligibility prediction project using the SVM model, we can assess the feasibility in three aspects: economic, technical, and social.

The feasibility of a project to predict loan eligibility depends on various factors such as data availability, the complexity of the model, and the accuracy of predictions. Here are some considerations:

Data Availability:

The first thing to consider is the availability of relevant data. You will need access to a dataset that includes information about loan applicants and their eligibility status. If such a dataset is not readily available, you may need to collect the data yourself, which can be time-consuming and expensive.

Model Complexity:

The second thing to consider is the complexity of the model. Loan eligibility prediction requires the use of a sophisticated algorithm that can handle complex data patterns. You will need to have the necessary expertise and resources to build, test, and validate the model. Additionally, the model should be interpretable and transparent, so it can be easily explained to stakeholders.

Accuracy of Predictions:

The third thing to consider is the accuracy of the predictions. If the model does not perform well, it will not be useful to lenders or borrowers. You will need to validate the model and ensure that it has a high degree of accuracy.

Overall, the feasibility of a loan eligibility prediction project depends on the availability of data, the complexity of the model, and the accuracy of predictions. If you have access to a relevant dataset and the necessary expertise, resources, and tools, then this project can be feasible.

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

1.1.1 Economical Feasibility

- The loan eligibility prediction project can have a significant economic impact on financial institutions by improving their loan approval process and reducing their risk of default. By accurately predicting whether a loan applicant is eligible for a loan, financial institutions can avoid the cost of processing ineligible loan applications and reduce the risk of granting loans to high-risk borrowers.
- In addition, the project can also help financial institutions better understand their customers' needs and preferences, which can improve customer satisfaction and retention. By analyzing the data collected from loan applications, financial institutions can identify patterns and trends that can help them tailor their products and services to better meet their customers' needs.
- Moreover, the project's implementation can also result in cost savings through the automation of manual processes. By leveraging machine learning algorithms, the loan eligibility prediction project can automate the analysis of large datasets, reducing the need for manual data processing and analysis. This can result in significant time and cost savings for financial institutions.
- On the other hand, there are also costs associated with the implementation and maintenance of the loan eligibility prediction project. These costs may include

hardware and software costs, as well as costs associated with hiring or training data scientists and machine learning engineers to develop and maintain the project.

- However, these costs can be outweighed by the potential economic benefits of the project, such as increased efficiency, improved customer satisfaction, and reduced risk of default. Furthermore, the use of cloud services and open-source software can also help reduce implementation and maintenance costs.
- Overall, the loan eligibility prediction project has strong economic feasibility, as it can provide financial institutions with significant benefits in terms of improved loan processing, risk management, customer satisfaction, and cost savings. By carefully considering the costs and benefits of the project, financial institutions can make informed decisions about its implementation and ensure its success.

1.1.2 Technical Feasibility

From a technical perspective, the project is feasible as it uses widely available technologies such as Python programming language, SVM model, and data analytics libraries such as Pandas, NumPy, and Seaborn. The project requires a moderate level of technical expertise to develop and deploy, which is easily achievable with the available resources and skills. Also, the data used in the project is readily available, which makes the data collection and processing easier.

1.1.3 Social Feasibility

From a social perspective, the project is feasible as it can contribute to making the loan application process more transparent and objective. By using an automated system, it can reduce the risk of human bias and discrimination, which is often a concern in loan approval decisions. It can also make the loan application process faster, more efficient, and accessible to a larger population. However, there may be concerns about data privacy and security,

which need to be addressed in the implementation of the project.

1.2 System Requirements

1.2.1 Hardware Requirements

Processor	- I3 & above
RAM	- 4 GB
Hard Disk	- 500 GB

1.2.2 Software Requirements

Operating System – Windows 8/10/11
Programming Language - Python

CHAPTER-2

SYSTEM ANALYSIS

2.1 Introduction

- Loan Eligibility Prediction is an application of machine learning that predicts whether a loan applicant is eligible for a loan or not. The project is designed to help financial institutions and banks in automating their loan eligibility process. The project uses a machine learning algorithm, specifically the Support Vector Machine (SVM) model, to predict whether a loan applicant is eligible for a loan or not.
- The loan eligibility prediction system takes into account various factors such as the applicant's income, credit score, employment history, loan amount, and loan term, among others. The system analyzes this data and generates a prediction on whether the loan applicant is likely to repay the loan.
- The system's accuracy is evaluated using various performance metrics such as accuracy, precision, recall, and F1-score. The project's accuracy is evaluated on both the training and testing datasets, and the results are compared to determine the model's performance.
- The loan eligibility prediction system has several advantages, such as improving the loan approval process's efficiency, reducing the risk of bad debts, and ensuring fairness in the loan approval process. The system also helps financial institutions and banks in making informed decisions by providing them with accurate and reliable predictions.
- Overall, the loan eligibility prediction system is an essential application of machine learning in the finance industry that can help institutions streamline their loan approval processes and make better-informed decisions.

2.2 Project Description

- The scope of this project documentation is to provide a comprehensive understanding of the entire project, including the problem statement, methodology, data collection, data analysis, results, limitations, and future work. The primary goal of this documentation is to serve as a reference for anyone who wants to understand the project's purpose, methodology, and findings.
- This documentation will also serve as a guide for future research and improvement on this topic. It will help researchers to understand the current state of the field, identify gaps in the research, and suggest potential avenues for future work. Moreover, it will help policymakers and industry professionals to make informed decisions about the use of machine learning algorithms in the financial sector.
- This documentation will provide detailed information about the methods and techniques used in the project, including data collection and preprocessing, feature engineering, and model training and evaluation. It will also explain how the results were obtained and how they contribute to the existing literature on this topic.
- The scope of this documentation is limited to the specific problem statement and dataset used in the project. However, the methods and techniques described in this documentation can be applied to other similar problems and datasets in the financial sector.
- Overall, this project documentation will serve as a valuable resource for anyone interested in the intersection of machine learning and the financial sector, providing a detailed analysis of the problem, methods, results, and potential for future research.

2.3 Modules

2.3.1 Data Cleaning and processing

For Data Cleaning and processing we are using mainly two libraries.

1. Numpy
2. Pandas

Numpy:

NumPy is an open-source Python library used for scientific computing and data analysis. It is a fundamental library for data science, machine learning, and scientific research due to its powerful capabilities for numerical computations and array manipulation. NumPy provides support for multi-dimensional arrays and matrices, which can be used for efficient and fast calculations on large datasets. The library includes a wide range of mathematical functions, including statistical, linear algebra, Fourier transforms, and more, making it an essential tool for scientific computing.

NumPy is built on top of C programming language and provides a flexible interface for interacting with the data, allowing users to easily manipulate and analyze data using Python. NumPy also provides support for broadcasting, which enables users to perform operations on arrays of different shapes and sizes, without requiring the arrays to be of the same shape or size. With its powerful capabilities and easy-to-use interface, NumPy has become a popular choice for data scientists and researchers working with large datasets in Python.

Advantages of numpy:

Here are some of the key advantages of using NumPy:

- **Faster computation:** NumPy provides fast and efficient computation, especially for large datasets, due to its implementation in C and Fortran.
- **Efficient memory management:** NumPy uses an array data structure that allows for efficient use of memory, which makes it suitable for handling large datasets.
- **Broadcasting:** NumPy allows for broadcasting, which means it can perform operations on arrays with different shapes and sizes. This saves time and reduces the need for loops.
- **Versatile:** NumPy supports a wide range of mathematical operations, such as linear algebra, Fourier transforms, and random number generation, making it a versatile

library for scientific computing.

- Integration with other libraries: NumPy integrates well with other scientific computing libraries in Python, such as Pandas, SciPy, and Matplotlib, which allows for easy data analysis and visualization.
- Open source: NumPy is an open-source library, which means that it is freely available to use, modify, and distribute.

Limitations of numpy:

- Limited data types: NumPy arrays can only store homogeneous data, which means all elements must be of the same data type. This can be limiting for some applications where data of different types need to be stored in the same array.
- Memory limitations: NumPy arrays are stored in contiguous blocks of memory. This means that if the size of the array is too large, it can exceed the available memory and cause performance issues.
- No support for distributed computing: NumPy does not have built-in support for distributed computing, which can be a limitation for some applications that require processing large amounts of data.
- Limited functionality: NumPy is primarily focused on numerical computing, so it may not have all the functionality required for some complex applications.
- Limited support for missing data: NumPy does not have built-in support for handling missing data, which can be a limitation for some applications that have missing data.

Pandas:

- Pandas is a popular open-source library in Python for data analysis and manipulation. It provides powerful tools for data cleaning, preparation, and analysis in a flexible and easy-to-use interface. Pandas is built on top of NumPy and provides additional data structures such as Series and DataFrame, which allow users to work with labeled and tabular data effectively.
- Pandas provides several functions for handling missing data, including the ability to interpolate missing values or drop rows with missing values altogether. It also provides extensive support for grouping, filtering, and aggregating data, making it easy to summarize and analyze large datasets.
- Pandas provides flexible and powerful functions for manipulating data, including merging and joining datasets, pivoting tables, and reshaping data. It also has functions

for handling datetime data and allows for easy visualization of data using its built-in plotting functions.

- With its powerful features and flexible interface, Pandas is a popular choice for data analysts, data scientists, and machine learning practitioners for data manipulation, cleaning, and analysis tasks.

Advantages of pandas

- Here are some of the advantages of using Pandas:
- Data manipulation: Pandas offers a wide range of data manipulation functionalities such as merging, reshaping, slicing, grouping, filtering, and aggregating data.
- Handling missing data: Pandas can handle missing data effectively and efficiently. It offers methods to fill in missing data, drop missing data, or interpolate missing values.
- Data visualization: Pandas provides an easy way to visualize data using its built-in plotting functionality. It can create various types of plots such as scatter plots, line plots, histograms, and box plots.
- Time series analysis: Pandas offers powerful tools for time series analysis, such as date range generation, time shifting, and rolling window functions.
- Data input and output: Pandas can read data from various file formats such as CSV, Excel, SQL, and JSON, and can also write data to these formats.
- Memory efficient: Pandas is memory efficient as it can handle large datasets without occupying too much memory.
- Integration with other libraries: Pandas can be easily integrated with other popular data analysis and machine learning libraries such as NumPy, Matplotlib, and Scikit-learn.
- Overall, Pandas is a versatile library that provides easy-to-use and efficient tools for data manipulation, analysis, and visualization, making it a popular choice among data scientists and analysts.

Limitations of pandas:

- Memory Usage: One of the main limitations of Pandas is its memory usage. Pandas is a very memory-intensive library, and when working with very large datasets, it can quickly run into memory problems. This can slow down processing speed and cause errors.
- Slow Performance: Pandas is designed to be user-friendly and easy to use, which

means that it is not always the fastest library for data processing. Some other libraries such as NumPy and PySpark are faster than Pandas for certain operations.

- **Not Suitable for Large-scale Distributed Computing:** Pandas is not suitable for large-scale distributed computing. It is designed to work on a single machine, and does not have built-in support for distributed computing. This makes it difficult to scale Pandas to work with very large datasets.
- **Limited Machine Learning Capabilities:** While Pandas has some machine learning capabilities, it is not as powerful as some other libraries such as TensorFlow and Scikit-Learn. It does not have advanced features such as neural networks or deep learning algorithms.
- **Limited Visualization Capabilities:** Although Pandas has some visualization capabilities, it is not designed for advanced data visualization. For more complex visualization, users should use libraries such as Matplotlib or Seaborn.

2.3.1 Preprocessing data

Preprocessing the data required a library such numpy and pandas.

Data Cleaning and processing involves in certain steps they are

1. Importing the dataset
2. Removing duplicates
3. Handling missing values
4. Encoding categorical variables
5. Feature scaling

Importing the dataset:

This step involves reading the dataset into the machine learning program. The dataset can be in various formats, including CSV, Excel, or SQL databases. Libraries such as Pandas in Python provide functions to read in different file types. Importing the dataset is an essential step to prepare data for further analysis and processing.

Removing duplicates:

Duplicate records in a dataset can lead to inaccurate results and can skew the analysis of the data. Removing duplicates can be done using functions such as `drop_duplicates()` in Pandas. This step ensures that each record in the dataset is unique and is not duplicated, leading to a more accurate analysis.

Handling missing values:

Missing values in a dataset can cause errors in data analysis and lead to inaccurate results. Various techniques can be used to handle missing values, including interpolation, deletion, or imputation. Imputation can be done using functions such as `fillna()` in Pandas. Handling missing values ensures that the analysis is conducted on complete data and provides accurate results.

Encoding categorical variables:

Many machine learning algorithms require numerical data. Categorical variables, such as gender or country, need to be encoded into numerical values to allow analysis by machine learning algorithms. Encoding can be done using techniques such as one-hot encoding or label encoding. Libraries such as Scikit-learn in Python provide functions to encode categorical variables.

Feature scaling:

Feature scaling involves scaling the data to a common range to avoid the influence of different scales of the variables. Feature scaling can improve the performance of many machine learning algorithms, including those based on distance metrics, by reducing the influence of individual variables on the final result. Common methods of feature scaling include normalization and standardization. Libraries such as Scikit-learn in Python provide functions to perform feature scaling.

2.3.2 Data Analysis

Data analysis is the process of extracting insights and meaningful information from raw data. It is a crucial step in the data science pipeline and involves several steps, including:

Data exploration:

This step involves getting a better understanding of the data by exploring it visually and statistically. Data visualization tools such as histograms, scatter plots, and box plots can be used to explore the data.

Data cleaning:

This step involves cleaning the data by removing or imputing missing values, dealing with outliers, and removing duplicates.

Data transformation:

This step involves transforming the data into a suitable format for analysis. This may include feature scaling, normalization, and encoding categorical variables.

Data modeling:

This step involves building and validating predictive models on the data. Machine learning algorithms such as regression, classification, and clustering can be used to model the data.

Data interpretation:

This step involves interpreting the results obtained from the models and drawing insights from them. Data visualization techniques such as heat maps and decision trees can be used to help interpret the results.

Reporting:

This step involves summarizing the results and communicating them to stakeholders in a clear and concise manner. Reports may include visualizations, summary statistics, and insights derived from the data analysis.

TABLE I. VARIABLES SUMMARY

<i>Variable</i>	<i>Counts (%)</i>
Gender	Male: 80.94% Female: 19.06%
Marital Status	Married: 64.53% Not Married: 35.47%
Education	Graduate: 77.87% Non-Graduate: 22.22%
Job	Employer: 87.36% Self- Employed: 12.64%
Credit History	Meeting Standards: 84.00% Off Standards: 16.00%
Location	Rural: 29.56% Semi-Urban: 35.58% Urban: 34.86 %
Dependent	0: 57.06 % 1: 16.75% 2: 16.65% 3: 9.53%
Income (in dollars)	Median = 5,314 \$, IQR = 3,142
Loan Amont (in \$1000s)	Median = 126.00, IQR =
Loan Status	Approved: 74.41% Not Approved: 25.59%

Library used for visual analysis: Seaborn

- ❖ Seaborn is a powerful data visualization library built on top of the Matplotlib library. It provides an easy-to-use interface for creating high-level statistical graphics that are both visually appealing and informative. Seaborn makes it easy to create complex visualizations with just a few lines of code. It provides a range of default styling options and themes that can be easily customized to suit your needs. Seaborn is particularly well-suited for creating statistical graphics that emphasize relationships between variables.

- ❖ One of the key features of Seaborn is its ability to create a wide range of statistical visualizations. These include scatterplots, line plots, bar plots, heatmaps, and many more. Seaborn also provides a range of statistical models for fitting and visualizing linear models, regression models, and other statistical models.
- ❖ Seaborn also makes it easy to work with datasets by providing a range of tools for data manipulation and transformation. It provides functions for grouping, aggregating, and transforming data, as well as for working with time series data.
- ❖ Another advantage of Seaborn is its integration with Pandas dataframes. This makes it easy to work with data in Seaborn, as Pandas is one of the most popular data manipulation libraries in Python. Seaborn also works seamlessly with NumPy arrays, which makes it easy to work with large datasets.
- ❖ Seaborn is also highly customizable. It provides a range of default color palettes and themes, but you can also create your own custom palettes and themes. This makes it easy to create visualizations that match your branding or style.
- ❖ Finally, Seaborn has a large and active community of users and developers. This means that there are many resources available online, including tutorials, documentation, and example code. This makes it easy to learn how to use Seaborn, and to get help when you need it.
- ❖ Despite its many advantages, Seaborn does have some limitations. For example, it may not be the best choice for creating highly custom visualizations, as it is designed to work with a set of predefined templates and themes. Additionally, Seaborn may not be the best choice for creating visualizations with complex interactivity or animations, as it is primarily designed for static visualizations. However, for most data visualization needs, Seaborn is a powerful and versatile library that can help you create informative and visually appealing visualizations with ease.

Why Seaborn:

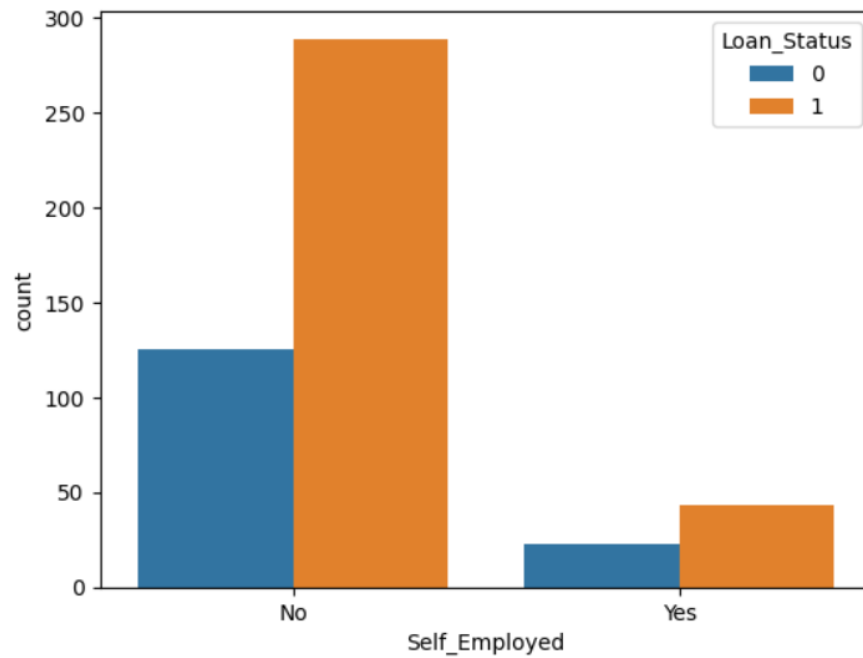
- ✓ While there are many visualization libraries available in Python, there are several reasons why one may prefer Seaborn over others such as Matplotlib:
- ✓ **Simpler Syntax:** Seaborn offers a simpler syntax than Matplotlib, which makes it easier to create complex visualizations. Seaborn has built-in functions for creating a

variety of plots such as scatterplots, lineplots, barplots, heatmaps, and more, with just a few lines of code.

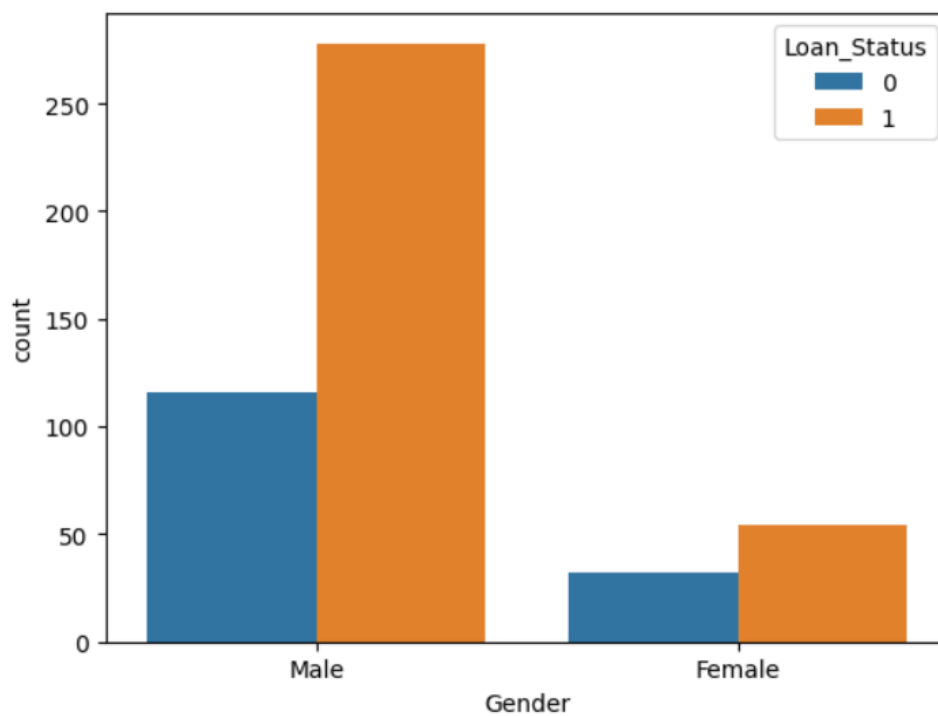
- ✓ **Better Aesthetics:** Seaborn is designed to create aesthetically pleasing plots with attractive color palettes and styles. It has many built-in color palettes and themes that can be easily applied to the plots.
- ✓ **Statistical Functionality:** Seaborn has built-in statistical functions that enable users to create informative visualizations with ease. For instance, it has functions for creating regression plots, distribution plots, and categorical plots, which make it easier to analyze relationships between variables.
- ✓ **Easy Handling of Dataframes:** Seaborn is built on top of Pandas, a popular data analysis library in Python. It can easily handle Pandas dataframes, making it easy to visualize data with Seaborn without the need for extensive data manipulation.
- ✓ **Customizability:** Seaborn provides a high degree of customizability, allowing users to tweak the appearance of the plots to their liking. It has many options for tweaking the size, labels, and appearance of the plots, which makes it easy to tailor the plots to specific requirements.

Response-Predictor Relationship

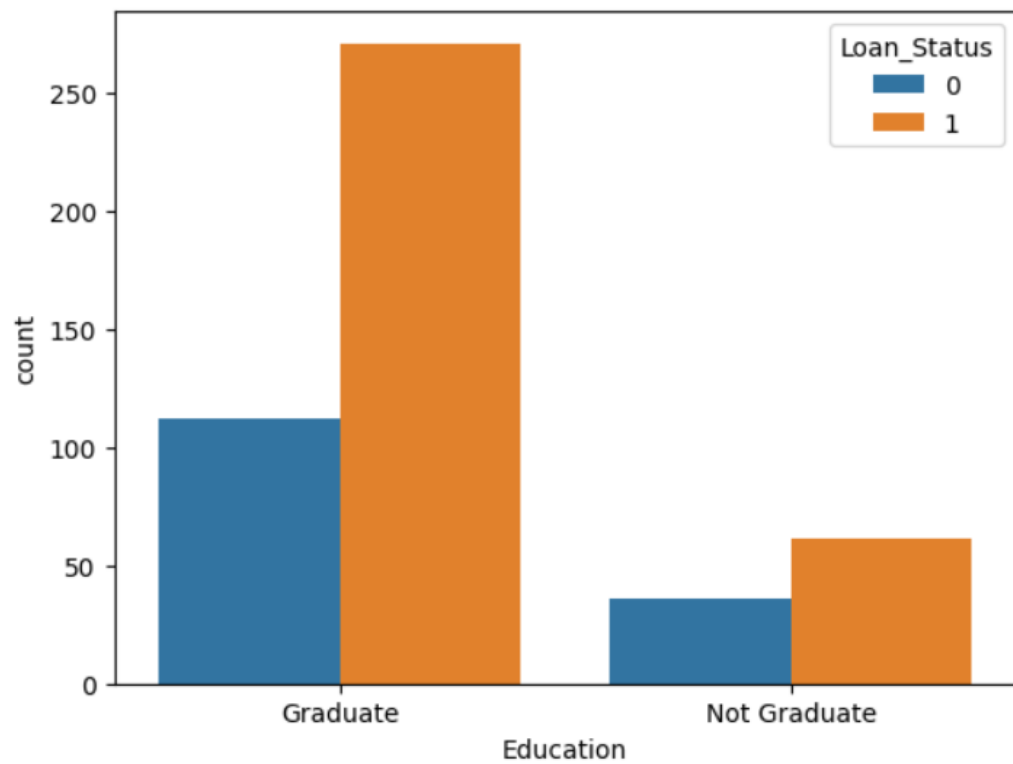
❖ Loan Status and Job class Relationship



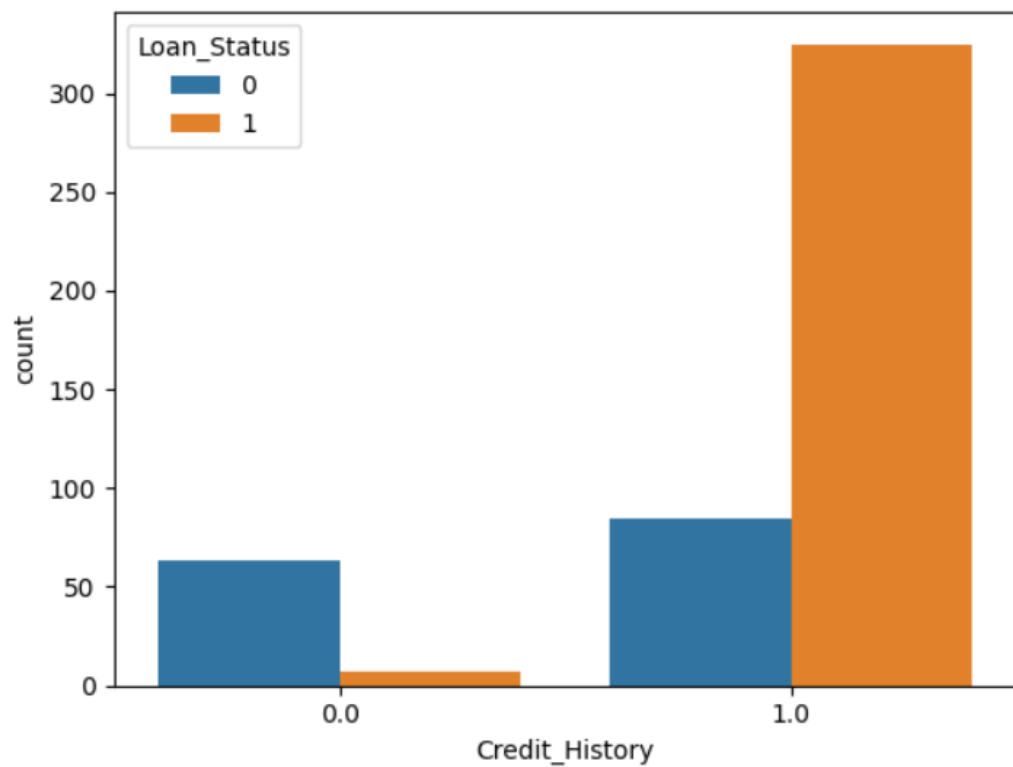
❖ Loan Status and Gender Relationship



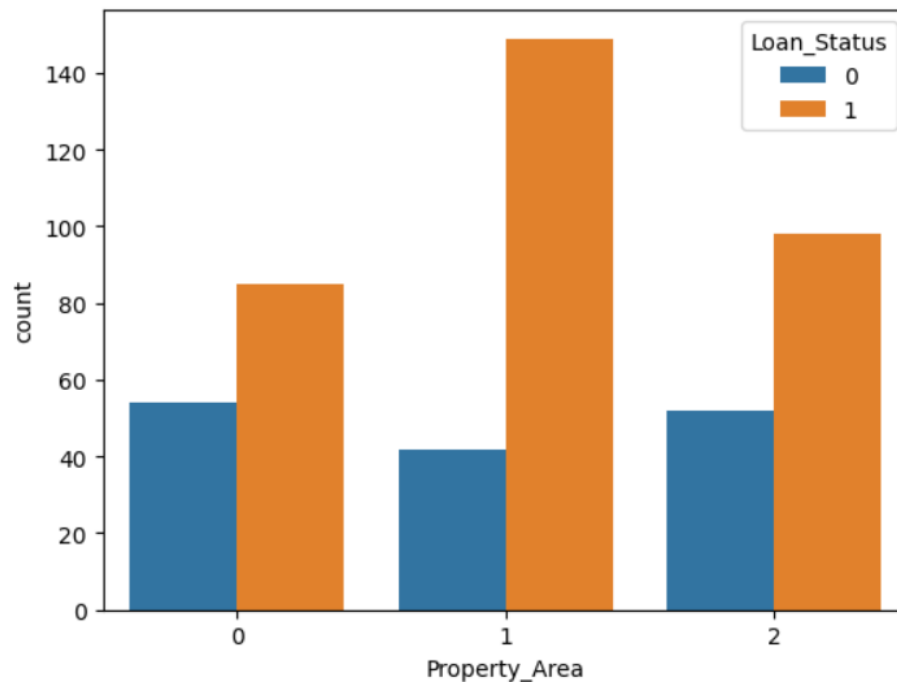
❖ Loan Status and Education Relationship



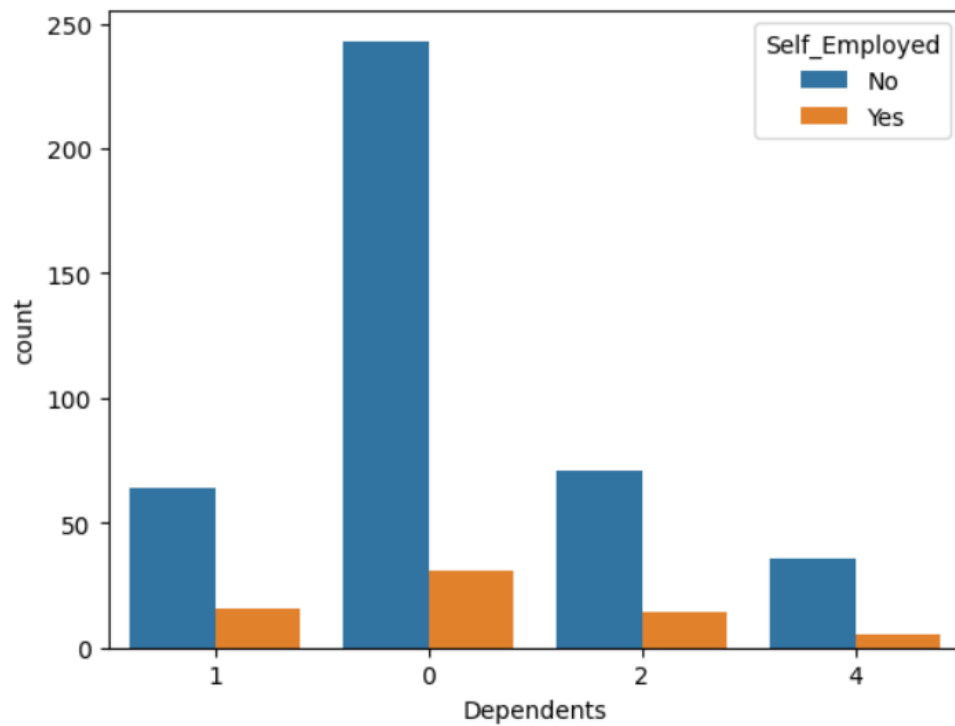
❖ Loan Status and Credit History Relationship



❖ Loan Status and Location Relationship



❖ Loan Status and Number of Dependent Relationship



2.3.3 Splitting the data

- Splitting the dataset into training and testing sets is an essential step in machine learning model development. The reason for splitting the data is to evaluate how well the model will perform on new, unseen data.
- The training set is used to train the model, meaning that the algorithm is given the input data along with the correct output values, and it learns to map the input data to the correct output values. The testing set, on the other hand, is used to evaluate the performance of the model. The algorithm is given the input data but not the correct output values, and it makes predictions. The accuracy of the predictions is then compared to the correct output values to evaluate the performance of the model.
- Splitting the data into training and testing sets helps to ensure that the model is not overfitting to the training data, which means that it is not just memorizing the training data but is instead learning patterns that can be generalized to new, unseen data. If we use the same data for training and testing, the model will perform very well on the training data but poorly on new data, because it has simply memorized the training data.
- To split the data, we typically randomly select a portion of the dataset (usually around 20-30%) to be the testing set, and the rest of the data becomes the training set. This ensures that the testing set is representative of the overall dataset and that the model is evaluated on data that it has not seen during training.

2.3.4 Training

Training data is a set of data that is used to train a machine learning model. The training data consists of input data along with the correct output values, which are used to teach the algorithm how to make predictions. The goal of the training process is to find the parameters of the model that best fit the training data. During training, the algorithm tries to minimize the difference between its predictions and the correct output values, using a loss function to measure the error. The model is then adjusted iteratively until it can accurately predict the output values for the input data in the training set.

2.3.5 Testing

Test data is a set of data that is used to evaluate the performance of a machine learning model. The test data consists of input data without the corresponding output values. The trained model is then used to make predictions on the test data, and the accuracy of these predictions is compared to the correct output values (which are not provided to the model during testing). The test data is a measure of how well the model will perform on new, unseen data, and it helps to prevent overfitting. If the model performs well on the training data but poorly on the test data, it is an indication that the model is overfitting to the training data and may not generalize well to new data.

2.4 Existing System

Loan approval is one of the most important processes that any banking organization owns. The acceptance or rejection of any loan application has a direct impact on the bank revenue and the profitability in quarterly issued financial statements. Though loan approval is a critical process, the actual decision made is not a straightforward procedure and comes with a lot of uncertainties. Recently, statisticians and data scientists have tried to automate this process to minimize risk and increase profitability by applying different statistical learning methods. In this work we explore a framework with an application by applying tree-based methods on publicly available dataset. This work aimed at developing a high performance predictive model for loan approval prediction using decision trees. Experiments were made in different varieties of tree methods ranging from the most simplified and comprehensible decision tree reaching up to the most complex random forests. Results yielded inadequate performance with respect to simplified decision trees due to the highlight correlated and complex feature space, majority of critical parameters affecting loan approval was not reflected upon and yielded an impractically over-simplified tree

2.4.1 Drawbacks of Existing System

- Slower
- Less accuracy

2.5 Proposed system

In proposed system we have used support vector machine model for training model which gave a better result.

2.5.1 Advantage of Proposed System

- Faster
- High accuracy

CHAPTER - 3

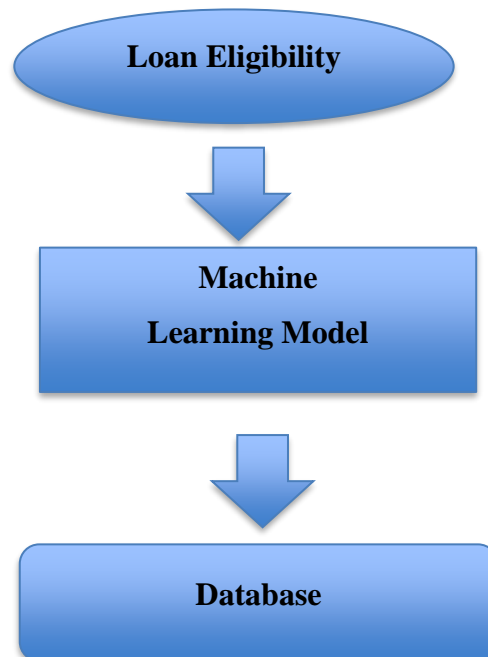
SYSTEM DESIGN

3.1 Data flow Diagram

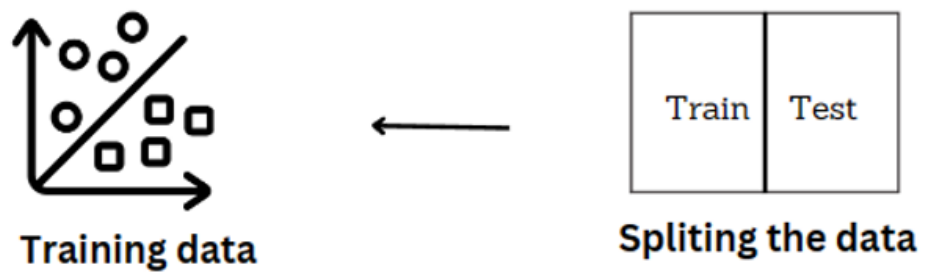
LEVEL-0



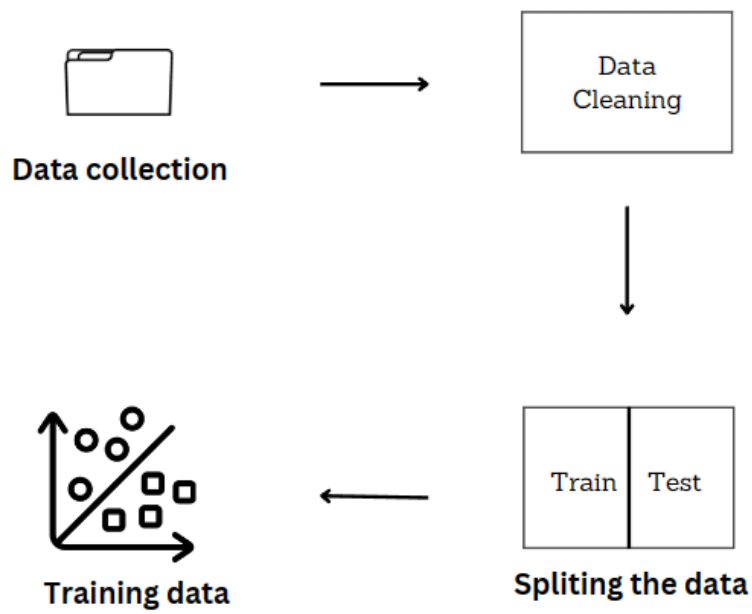
LEVEL-1



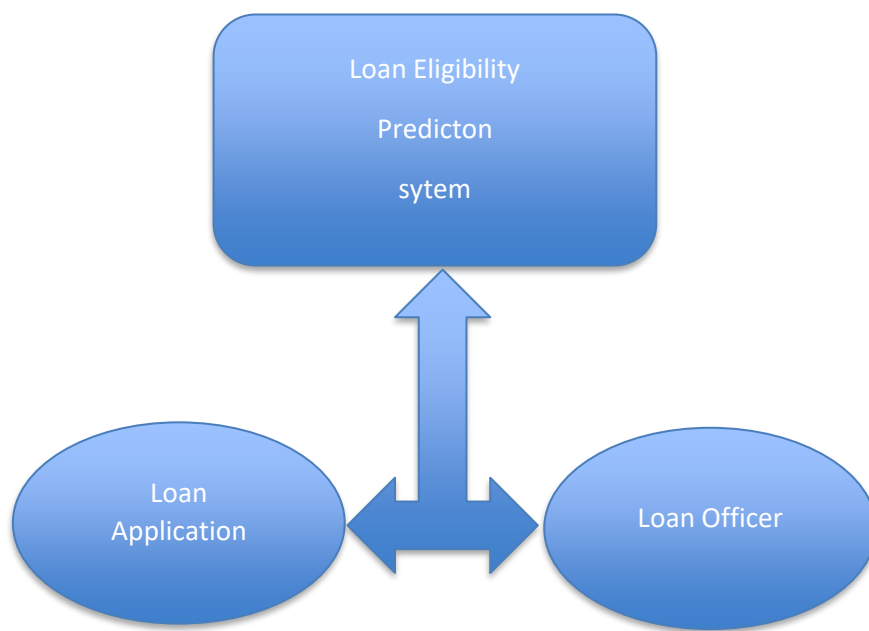
LEVEL-2



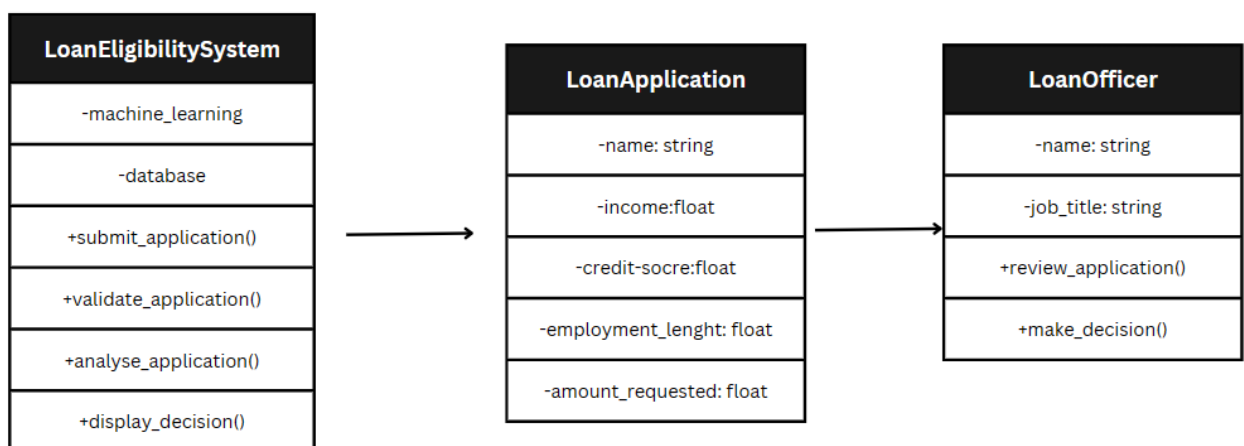
3.2 Overall Diagram



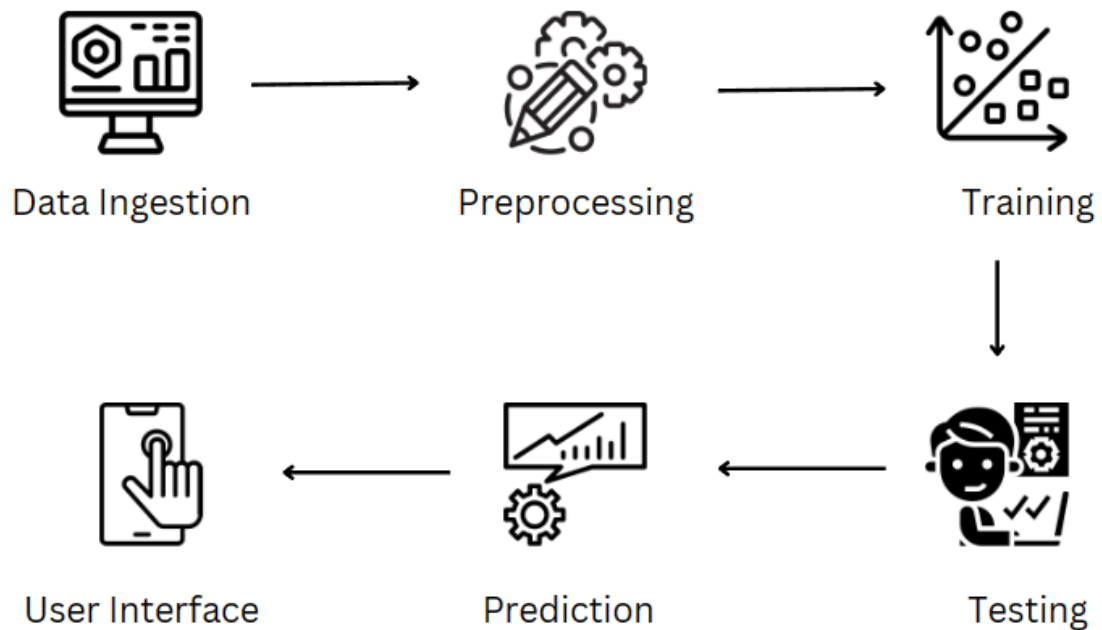
3.3 Case Diagram



3.4 Class Diagram



3.5 Architecture Diagram



3.6 Source Code

CODING:

```
import numpy as np #better structured array

import pandas as pd #data frame

import seaborn as sns #ploting library

from sklearn.model_selection import train_test_split # to split train part and
test part

from sklearn import svm #suppoort vector machine model
```

```
from sklearn.metrics import accuracy_score
```

```
loan_dataset = pd.read_csv('/content/loan-train.csv')
```

```
type(loan_dataset)
```

```
loan_dataset.head()
```

```
loan_dataset.shape
```

```
loan_dataset.describe()
```

```
loan_dataset.isnull().sum()
```

```
loan_dataset = loan_dataset.dropna()
```

```
loan_dataset.isnull().sum()
```

```
loan_dataset.replace({"Loan_Status":{"N":0,'Y':1}},inplace=True)
```

```
loan_dataset.head()
```

```
loan_dataset['Dependents'].value_counts()
```

```
loan_dataset = loan_dataset.replace({'Dependents':{'3+':4}})
```

```
loan_dataset['Dependents'].value_counts()
```

```
sns.countplot(x = 'Education', hue = 'Loan_Status',data = loan_dataset)
```

```
sns.countplot(x = 'Married', hue = 'Loan_Status', data = loan_dataset)
```

```
sns.countplot(x = 'Dependents', hue = 'Self_Employed',data = loan_dataset)
```



```
a = loan_dataset.replace({'Married':{'No':0,'Yes':1}, 'Gender':{'Male':1,
'Female':0}, 'Self_Employed':{'No':0, 'Yes':1}, 'Property_Area':{'Rural':0,
'Semiurban':1, 'Urban':2}, 'Education':{'Graduate':1, 'Not Graduate':0}}, inplace
= True)
```

```
loan_dataset.head()
```

```
print(loan_dataset['LoanAmount'].min())
```

```
print(loan_dataset['LoanAmount'].max())
```

```
print(loan_dataset['Loan_Amount_Term'].min())
```

```
print(loan_dataset['Loan_Amount_Term'].max())
```

```
loan_dataset['Credit_History'].value_counts()
```

```
x = loan_dataset.drop(columns=['Loan_ID', 'Loan_Status'], axis = 1)
```

```
y = loan_dataset['Loan_Status']
```

```
print(x)
```

```
print(y)
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.1,stratify =
y,random_state = 3)
```

```
print(x.shape,x_train.shape,x_test.shape)
```

```
y_train
```

```
classifier = svm.SVC(kernel='linear')
```

```
classifier.fit(x_train, y_train)
```

```
print(x_train)
```

```
x_train_prediction = classifier.predict(x_train)
```

```
training_data_accuracy = accuracy_score(x_train_prediction, y_train)
```

```
print('Accuracy on training data: ',training_data_accuracy)
```

```
x_test_prediction = classifier.predict(x_test)
```

```
testing_data_accuracy = accuracy_score(x_test_prediction, y_test)
```

```
print('Accuracy on testing data: ',testing_data_accuracy)
```

```
print("Enter the following information to check your eligibility for a loan:")
```

```
Gender = input("What is your gender? (male or female): ")
```

```
if Gender == "male":
```

```
    Gender = 1
```

```
elif Gender == "female":
```

```
    Gender = 0
```

```
else:
```

```
    print("Invalid input for marital status")

print(Gender)


married = input("Are you married? (Yes or No): ").title()

if married == "Yes":

    married = 1

elif married == "No":

    married = 0

else:

    print("Invalid input for marital status")

print(married)


dependents = input("How many dependents do you have? (0-4): ")

if dependents.isdigit() and 0 <= int(dependents) <= 4:

    dependents = int(dependents)

else:

    print("Invalid input for number of dependents")

print(dependents)


education = input("Are you a graduate? (Yes or No): ").title()
```

```
if education == "Yes":
```

```
    education = 1
```

```
elif education == "No":
```

```
    education = 0
```

```
else:
```

```
    print("Invalid input for education level")
```

```
print(education)
```

```
self_employed = input("Are you self-employed? (Yes or No): ").title()
```

```
if self_employed == "Yes":
```

```
    self_employed = 1
```

```
elif self_employed == "No":
```

```
    self_employed = 0
```

```
else:
```

```
    print("Invalid input for self-employment status")
```

```
print(self_employed)
```

```
applicant_income = input("What is your monthly applicant_income? (in  
dollars): ")
```

```
if applicant_income.isdigit():  
    applicant_income = int(applicant_income)  
  
else:  
    print("Invalid input for applicant_income")  
  
print(applicant_income)
```

```
coapplicant_income = input("What is your monthly coapplicant_income? (in  
dollars): ")
```

```
if coapplicant_income.isdigit():  
    coapplicant_income = int(coapplicant_income)  
  
else:  
    print("Invalid input for coapplicant_income")  
  
print(coapplicant_income)
```

```
loan_amount = input("What is the loan amount you are applying for? (in  
thousands of dollars from 9 to 600): ")
```

```
if loan_amount.isdigit():  
    loan_amount = int(loan_amount)
```

else:

print("Invalid input for loan amount")

print(loan_amount)

loan_amount_term = input("What is the loan term you are applying for? (in months from 36 to 480): ")

if loan_amount_term.isdigit():

loan_amount_term = int(loan_amount_term)

else:

print("Invalid input for loan term")

print(loan_amount_term)

credit_history = input("Do you have a credit history? (1 or 0): ").title()

if credit_history == "1":

credit_history = 1

elif credit_history == "0":

credit_history = 0

else:

print("Invalid input for credit history")

print(credit_history)

```
property_area = input("What type of property do you live in? (Rural,  
Semiurban, Urban): ").title()
```

```
if property_area == "Rural":
```

```
    property_area = 0
```

```
elif property_area == "Semiurban":
```

```
    property_area = 1
```

```
elif property_area == "Urban":
```

```
    property_area = 2
```

```
else:
```

```
    print("Invalid input for property area")
```

```
print(property_area)
```

```
input_data = (Gender, married, dependents, education, self_employed,  
applicant_income, coapplicant_income, loan_amount, loan_amount_term,  
credit_history, property_area)
```

```
input_as_numpy_array = np.asarray(input_data)
```

```
input_data_reshaped = input_as_numpy_array.reshape(1, -1)
```

```
prediction = classifier.predict(input_data_reshaped)
```

```
print(prediction)
```

```
if prediction[0] == 1:  
    print("Congratulations! You are eligible for the loan.")  
  
else:  
    print("Sorry, you are not eligible for the loan.")
```


CHAPTER – 4

TESTING

4.1 Unit Testing

- `validate_application()`: This function checks whether a loan application meets the required criteria, such as minimum income, credit score, and employment length. You can write unit tests to verify that the function correctly identifies valid and invalid applications based on various inputs.
- `analyze_application()`: This function analyzes a loan application and predicts whether it is eligible for a loan. You can write unit tests to verify that the function produces accurate predictions based on different input data.
- `make_decision()`: This function makes a loan decision based on the loan analysis and other factors, such as the available funds and risk assessment. You can write unit tests to verify that the function makes correct decisions based on different loan scenarios.

In each unit test, you would provide input data to the function being tested and compare its output with the expected output. If the actual output matches the expected output, the test passes, otherwise, the test fails and you would need to debug the function to fix the error.

Overall, unit testing can help you ensure that your Loan Eligibility Prediction System works as expected and meets its design requirements. It can also help you catch and fix bugs early in the development process, which can save time and reduce the risk of more serious errors later on.

4.2 Integration Testing

Integration testing is a software testing method that involves testing the interaction between different components or units of a software application to ensure that they work together correctly. In the context of your Loan Eligibility Prediction System, integration testing would involve testing how the different modules of your system, such as the loan application validation, analysis, and decision-making functions, work together as a whole.

During integration testing, you would test the interaction between the different modules of your system to ensure that they integrate correctly and produce the expected results. For example, you might test how the `validate_application()` function interacts with the `analyze_application()` and `make_decision()` functions to ensure that the loan application is correctly validated before it is analyzed and a decision is made. You might also test how the system interacts with external data sources, such as credit bureau data or bank records, to ensure that the data is correctly integrated and processed.

Integration testing is important because it helps to identify and fix issues related to the interactions between different modules of your system. By testing how the different components work together, you can ensure that your system is reliable, robust, and performs as expected under various scenarios. It can also help you catch and fix any errors or inconsistencies in the data or logic flow of your system, improving the overall quality and reliability of your application.

4.3 Validation Testing

Validation testing is a software testing method that verifies that a system or application meets its intended specifications and user requirements.

4.4 Acceptance Testing

- Acceptance testing is a software testing method that involves testing a software system or application to ensure that it meets the requirements and expectations of its intended users or stakeholders. The goal of acceptance testing is to verify that the system is ready for deployment and meets the business needs and user requirements.
- Acceptance testing typically involves creating test scenarios that simulate real-world user interactions and workflows with the system. These scenarios can be created by the development team or by the end-users themselves. The tests may be automated or manual and may cover various aspects of the system, such as functionality, usability, performance, security, and compatibility.

- During acceptance testing, the system is evaluated against predefined acceptance criteria and acceptance test cases to determine whether it meets the user's requirements and expectations. If the system passes all acceptance tests, it is considered ready for deployment, and if not, the issues identified during testing are addressed and retested until the system meets the acceptance criteria.

CHAPTER - 5

IMPLEMENTATION

5.1 Introduction

- ✓ Support Vector Machine (SVM) is a popular machine learning algorithm used for classification and regression tasks. It is a powerful algorithm that can handle complex datasets and has been widely used in various applications such as image classification, text classification, and financial forecasting.
- ✓ In Python, the Scikit-Learn library provides an implementation of SVM that can be easily used for classification and regression tasks. The SVM algorithm works by finding the best possible boundary or hyperplane that separates the data points into different classes. This boundary is chosen in such a way that it maximizes the margin between the two classes, which helps to improve the accuracy and generalization of the model.
- ✓ To use SVM in Python, you first need to import the necessary libraries, including NumPy, Pandas, and Scikit-Learn. You can then load your dataset into a Pandas DataFrame and preprocess it as needed. Next, you can split the data into training and testing sets, and use the Scikit-Learn SVM implementation to train the model on the training data. You can then evaluate the performance of the model on the testing data and fine-tune the hyperparameters to improve its accuracy.
- ✓ Overall, SVM is a powerful algorithm that can be used for various classification and regression tasks, and its implementation in Python through Scikit-Learn provides a user-friendly and efficient way to use this algorithm in your machine learning projects.

5.2 Algorithm used

Algorithm that is used in this project is SVM (Support vector Machine model)

Advantages of SVM:

Here are some of the main advantages of SVM:

Effective in high-dimensional spaces:

SVM is very effective when the number of features is very large, such as in text classification, image classification, and bioinformatics.

Memory efficient:

SVMs use a subset of training points in the decision function called support vectors, which makes it memory efficient.

Robust to outliers:

SVMs are robust to outliers because they use a margin-based loss function that penalizes points that are close to the decision boundary.

Versatile:

SVMs can be used for both regression and classification tasks.

Handles non-linear data:

SVMs are able to handle non-linear data by using the kernel trick, which maps the input data into a higher dimensional space where it can be linearly separated.

Fewer hyperparameters:

SVMs have fewer hyperparameters compared to other machine learning algorithms, which makes them easier to tune.

Can handle large datasets:

SVMs are efficient even with large datasets, thanks to their ability to handle sparse

datasets and their memory efficiency.

Good generalization:

SVMs have good generalization performance, which means they are able to classify new, unseen data accurately.

Mathematically well-defined:

SVMs are mathematically well-defined, which makes them easier to understand and interpret.

Perform well on imbalanced datasets:

SVMs can handle imbalanced datasets better than other algorithms, because they are able to find the best hyperplane that separates the classes with maximum margin.

In summary, SVMs are a powerful machine learning algorithm that can handle high-dimensional data, non-linear data, and imbalanced datasets, while still being memory efficient and mathematically well-defined. Their ability to generalize well and their versatility make them a popular choice for a wide range of classification and regression tasks.

The SVM algorithm consists of the following steps:

Data pre-processing:

This step involves cleaning and pre-processing the data to ensure that it is suitable for analysis. This may include removing missing values, encoding categorical variables, and feature scaling.

Training:

In this step, the SVM algorithm is trained on the labeled data to find the hyperplane that separates the data into two classes. The SVM algorithm searches for the hyperplane that maximizes the margin between the support vectors.

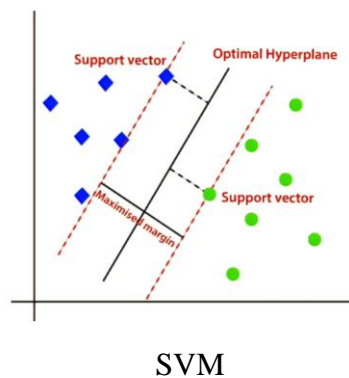
Testing:

After the SVM algorithm is trained, it can be used to make predictions on new data points. The performance of the SVM algorithm is evaluated based on how well it can classify the test data.

Tuning hyperparameters:

SVM has several hyperparameters that need to be tuned to optimize the performance of the model. The most important hyperparameters are the kernel function, the regularization parameter, and the gamma parameter.

In summary, SVM is a powerful machine learning algorithm that is widely used for classification and regression analysis. It is a versatile algorithm that can handle high-dimensional data, non-linearly separable data, and outliers. SVM requires careful selection of hyperparameters and appropriate pre-processing of the data for optimal performance.

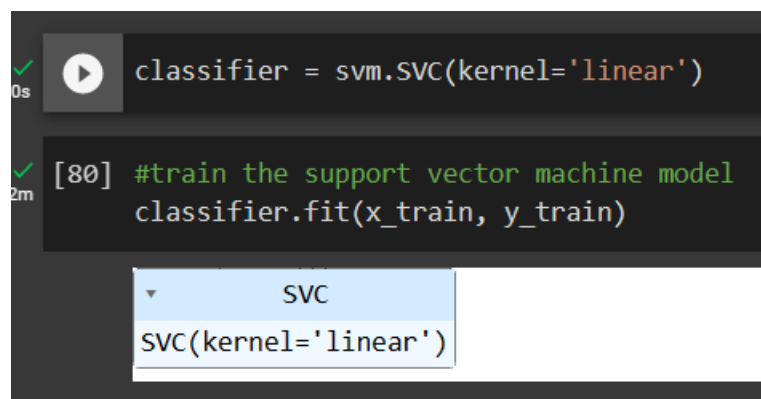


5.3 Training data

- Training data is a set of input data with corresponding output values that are used to teach a machine learning algorithm to make predictions. The training process involves adjusting the parameters of the model iteratively until it can accurately predict the output values for the input data in the training set.
- The training process begins with the initialization of the model parameters. These parameters are adjusted during training to minimize the difference between the predicted output values and the actual output values. This difference is measured using a loss function, which quantifies the error between the predicted values and the actual values.
- The loss function is used to update the model parameters using an optimization algorithm such as gradient descent. The optimization algorithm adjusts the parameters of the model in the direction that minimizes the loss function, making the predicted output values closer to the actual output values.
- The training process typically involves multiple iterations, or epochs, where the entire training data set is processed through the algorithm. After each epoch, the model parameters are updated, and the loss function is evaluated again to determine the new

error rate. This process is repeated until the model's performance on the training data reaches an acceptable level.

- The performance of the model during training is typically monitored using validation data, which is a subset of the training data that is held back during training to evaluate the model's performance on new, unseen data. This helps to prevent overfitting, where the model learns to predict the output values for the training data set but does not generalize well to new data.
- Once the training process is complete, the resulting model can be used to make predictions on new, unseen data.



The screenshot shows a Jupyter Notebook interface. The top cell contains the code `classifier = svm.SVC(kernel='linear')` and has a green checkmark and a play button icon. The bottom cell contains the code `[80] #train the support vector machine model classifier.fit(x_train, y_train)` and has a green checkmark and a '2m' execution time indicator. Below the code cells, a dropdown menu is open, showing 'SVC' and 'SVC(kernel='linear')' as options.

SVM

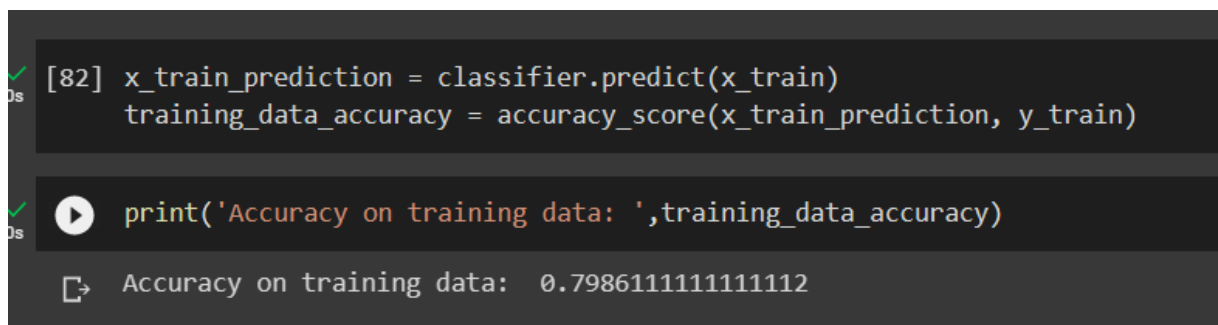
5.4 Testing data

Accuracy on train data

- Accuracy on the training data is an essential metric in evaluating the performance of a machine learning model because it provides information about how well the model has learned to predict the output values for the input data in the training set.
- In other words, a high accuracy on the training data indicates that the model has learned the underlying patterns and relationships in the training data and is able to make accurate predictions on this data. This is important because the goal of a machine learning model is to accurately predict outcomes for new, unseen data.
- Furthermore, tracking the accuracy on the training data during the training process can provide insights into the model's convergence and help identify potential issues

such as overfitting or underfitting.

- However, it is important to note that a high accuracy on the training data does not guarantee that the model will perform well on new, unseen data. This is because the model may have overfit to the training data, meaning that it has learned the noise in the training data as well as the underlying patterns, which can lead to poor performance on new data.
- Therefore, while accuracy on the training data is an essential metric for evaluating a model's performance, it should always be complemented with evaluation on a separate test set of data that the model has not seen during training. This will provide a more accurate estimate of how well the model will generalize to new, unseen data.
 - The accuracy of the trained model on the training data is 79.86%. This indicates that the model has learned the underlying patterns and relationships in the training data and is able to make accurate predictions on the same data it was trained on.



```
[82] x_train_prediction = classifier.predict(x_train)
      training_data_accuracy = accuracy_score(x_train_prediction, y_train)

print('Accuracy on training data: ',training_data_accuracy)

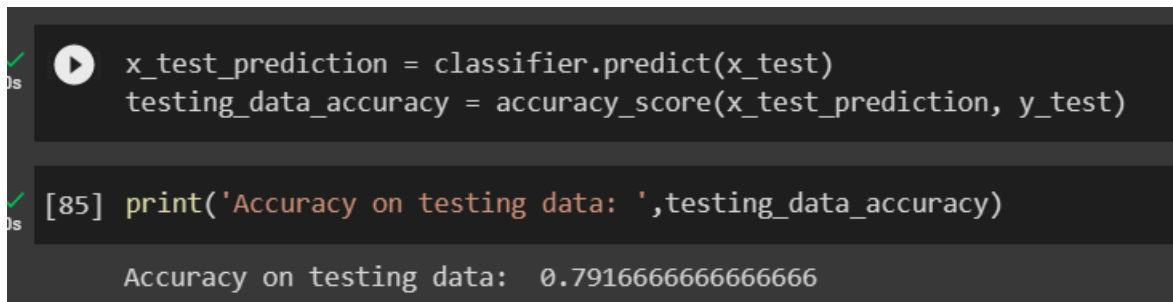
Accuracy on training data:  0.7986111111111112
```

Accuracy on train data

Accuracy on Test data

- According to our testing results, the model achieved an accuracy of 79.16%. This means that the model correctly predicted the target variable for 79.16% of the observations in the test dataset.
- The accuracy of the model on the test data is an important metric because it provides an estimate of how well the model will generalize to new, unseen data. This is because the test dataset represents data that the model has not seen during training, and therefore provides a more realistic assessment of the model's performance in the real world.

- Overall, achieving an accuracy of 79.16% on the test data suggests that the model is performing well and is able to make accurate predictions on new, unseen data. However, it is important to continue monitoring the model's performance and re-evaluate it with new data as it becomes available to ensure continued accuracy and reliability.



```
x_test_prediction = classifier.predict(x_test)
testing_data_accuracy = accuracy_score(x_test_prediction, y_test)

[85] print('Accuracy on testing data: ',testing_data_accuracy)

Accuracy on testing data: 0.7916666666666666
```

Accuracy on test data

5.5 Software Environment

Python Technology

Python is a high-level, interpreted programming language that has gained tremendous popularity over the past few years, especially in the field of software development. It has become one of the go-to languages for many developers due to its simplicity, versatility, and ease of use. Python can be used for a wide range of applications such as web development, scientific computing, data analysis, machine learning, and more.

Python is an open-source programming language, which means it is free to use and modify, making it a great choice for both individuals and businesses. The language is also highly portable, which means it can run on almost any platform or operating system, including Windows, Mac, Linux, and even mobile devices.

One of the reasons Python is so popular in the software environment is its simplicity. The language is easy to learn, even for beginners, thanks to its clean syntax and extensive documentation. The language's simplicity also means that developers can write code faster, which can save time and increase productivity.

Python's versatility is another reason why it is so popular. The language can be used for a wide range of applications, from simple scripts to complex, multi-tier applications. Python can also be used in conjunction with other programming languages, making it a valuable tool for integrating different components of a software environment.

Python's popularity in the software environment can also be attributed to its vast library of pre-built modules and frameworks. These libraries and frameworks make it easy for developers to perform common tasks, such as data manipulation, web scraping, and even creating GUI applications. Some popular Python libraries and frameworks include NumPy, Pandas, Django, Flask, and Pygame.

In addition to its vast library of modules and frameworks, Python also has a large and active community of developers who contribute to the language's development and maintenance. This community provides support, resources, and tools to help developers learn and work with Python.

Another reason why Python is popular in the software environment is its support for machine learning and artificial intelligence. With libraries such as TensorFlow, Keras, and Scikit-learn, Python is a popular choice for developing machine learning models and AI applications. Python's simplicity, versatility, and extensive library of tools make it a great choice for developers looking to explore these emerging technologies.

Overall, Python is a powerful and versatile language that has gained tremendous popularity in the software environment due to its simplicity, versatility, and ease of use. Its vast library of pre-built modules and frameworks, coupled with its support for machine learning and artificial intelligence, make it an excellent choice for developing complex software applications. Whether you're a beginner or an experienced developer, Python is a language worth exploring if you're looking to build software applications in today's fast-paced technology environment.

Python Interpreter Language

Python is an interpreted programming language, which means that code written in Python is executed directly by the Python interpreter. This is in contrast to a compiled programming language, where code is compiled into machine code before it is executed.

The Python interpreter is the software that actually runs the Python code. When a developer writes a Python program, the interpreter reads the code and executes it line by line, interpreting each line of code and performing the actions specified by the code.

One of the advantages of an interpreted language like Python is that it is generally easier and faster to develop code. Because the code can be executed directly, there is no need to compile the code before running it. This can save a lot of time during development, as developers can immediately see the results of their code without having to wait for the code to compile.

Another advantage of an interpreted language is that it can be more flexible than a compiled language. For example, in Python, it is possible to dynamically change the type of a variable during runtime. This means that a developer can write code that can adapt to changing conditions, which can be especially useful in situations where the input data is not known in advance.

There are, however, some drawbacks to interpreted languages as well. One of the main disadvantages is that interpreted code is generally slower than compiled code. This is because the interpreter must translate each line of code into machine code at runtime, which can be slower than pre-compiling the code. However, this is less of an issue in modern computers, as they are generally fast enough to execute Python code efficiently.

Another disadvantage of interpreted languages is that they can be less secure than compiled languages. Because the code is executed directly by the interpreter, it can be easier for an attacker to exploit vulnerabilities in the code. However, this can be mitigated by using security best practices such as input validation and error handling.

Python also has a feature called the Interactive Interpreter, which allows developers to interactively write and test Python code in a shell environment. This can be very useful for testing small code snippets or experimenting with new features. The Interactive Interpreter also supports tab completion, which can save time and improve code accuracy by automatically completing code snippets based on what has been typed so far.

In summary, Python is an interpreted programming language, which means that code written in Python is executed directly by the Python interpreter. This has several advantages, including faster development time, greater flexibility, and a powerful interactive shell environment. However, there are also some drawbacks, including slower performance compared to compiled languages and potential security issues. Overall, Python's interpreted nature is a key part of its design, and one of the reasons why it has become such a popular language for a wide range of applications.

What can Python Technology do in this project

- ❖ Python plays a critical role in the loan eligibility prediction project as it is the primary programming language used for building the machine learning model. Machine learning is a complex field that requires a significant amount of computational power and advanced algorithms, and Python is well-suited for both of these requirements.
- ❖ Python has a wide range of libraries and frameworks that are specifically designed for machine learning, such as scikit-learn, TensorFlow, and Keras. These libraries provide developers with pre-built algorithms and models that can be easily incorporated into their projects, saving time and effort. Additionally, Python's simple syntax and ease of use make it an ideal language for machine learning beginners to get started.
- ❖ In the loan eligibility prediction project, the Python programming language is used to perform data preprocessing, feature engineering, and model training. Data preprocessing involves cleaning and transforming the raw data into a format that can be easily used by the machine learning model. This step is critical as the quality of the input data can greatly impact the accuracy of the final predictions.
- ❖ Feature engineering involves selecting the most relevant features from the input data that will be used to train the machine learning model. This step is important as it can help to reduce the amount of noise in the data and improve the accuracy of the model.

- ❖ Once the data has been preprocessed and the relevant features have been selected, the machine learning model is trained using Python. In the loan eligibility prediction project, a support vector machine (SVM) algorithm is used to train the model. SVMs are a popular machine learning algorithm for classification tasks and are well-suited for the loan eligibility prediction problem.

- ❖ Python is also used to evaluate the performance of the machine learning model by testing it on a separate set of data called the test set. This step is important as it provides an estimate of the accuracy of the model on unseen data, which is crucial for ensuring that the model is robust and can be used in real-world applications.

- ❖ In summary, Python plays a critical role in the loan eligibility prediction project as it is the primary programming language used for data preprocessing, feature engineering, model training, and evaluation. Its vast library of machine learning tools and frameworks makes it an ideal choice for developing complex machine learning models, and its simplicity and ease of use make it accessible to developers with varying levels of experience.

Libraries used

1. Numpy
2. Pandas
3. Seaborn
4. SVM – Support Vector Machine
5. Accuracy_score
6. Train_test_split

NumPy:

NumPy is a popular Python library used for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions. In the loan eligibility prediction project, NumPy is used for data preprocessing and feature engineering. For example, NumPy arrays are used to store and manipulate the loan dataset, and NumPy functions are used to calculate various statistics and perform mathematical operations.

Pandas:

Pandas is a Python library used for data manipulation and analysis. It provides powerful data structures and functions for working with structured data, such as CSV files and SQL databases. In the loan eligibility prediction project, Pandas is used extensively for data preprocessing and feature engineering. For example, Pandas functions are used to read in the loan dataset, clean and transform the data, and extract relevant features for the machine learning model.

Seaborn:

Seaborn is a Python library used for data visualization. It provides a high-level interface for creating sophisticated and visually appealing graphs and charts, such as scatterplots, heatmaps, and histograms. In the loan eligibility prediction project, Seaborn is used to visualize the loan dataset and explore the relationship between different features. For example, Seaborn functions are used to create scatterplots of loan amount versus income, and boxplots of loan approval status versus credit history.

SVM – Support Vector Machine:

SVM is a popular machine learning algorithm used for classification and regression tasks. It works by finding a hyperplane that best separates the data into different classes, based on the input features. In the loan eligibility prediction project, an SVM algorithm is used to train the machine learning model to predict whether a loan applicant is eligible or not based on their input data.

Accuracy_score:

`Accuracy_score` is a function from the scikit-learn library that is used to measure the accuracy of a machine learning model. It takes in the actual output values and the predicted output values, and returns a score that represents how well the model performed. In the loan eligibility prediction project, the `accuracy_score` function is used to evaluate the performance of the SVM model on the test dataset.

Train_test_split:

`Train_test_split` is a function from the scikit-learn library that is used to split a dataset into training and testing subsets. It randomly divides the data into two portions, one for training the machine learning model and one for testing its performance. In the loan eligibility prediction project, the `train_test_split` function is used to split the loan dataset into training and testing subsets before training the SVM model.

Overall, these libraries play a critical role in the loan eligibility prediction project by providing powerful tools and functions for data manipulation, analysis, visualization, and machine learning. By leveraging these libraries, developers can efficiently build complex machine learning models that accurately predict loan eligibility and help financial institutions make informed decisions.

CHAPTER - 6

RESULTS & CONCLUSION

6.1 Results

Accuracy of train and test data

```
x_train_prediction = classifier.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction, y_train)
```

```
print('Accuracy on training data: ',training_data_accuracy)
```

Accuracy on training data: 0.7986111111111112

```
x_test_prediction = classifier.predict(x_test)
testing_data_accuracy = accuracy_score(x_test_prediction, y_test)
```

```
print('Accuracy on testing data: ',testing_data_accuracy)
```

Accuracy on testing data: 0.7916666666666666

Getting input from the user

Enter the following information to check your eligibility for a loan:

What is your gender? (male or female):

After getting input from the user

Enter the following information to check your eligibility for a loan:

What is your gender? (male or female): male

1

Are you married? (Yes or No): yes

1

How many dependents do you have? (0-4): 3

3

Are you a graduate? (Yes or No): yes

1

Are you self-employed? (Yes or No): no

0

What is your monthly applicant_income? (in dollars): 10000

10000

What is your monthly coapplicant_income? (in dollars): 2000

2000

What is the loan amount you are applying for? (in thousands of dollars from 9 to 600): 400

400

What is the loan term you are applying for? (in months from 36 to 480): 30

30

Do you have a credit history? (1 or 0): 1

1

What type of property do you live in? (Rural, Semiurban, Urban): rural

0

Output:

```
[1]  
Congratulations! You are eligible for the loan.
```

6.2 Conclusion

In conclusion, the loan eligibility prediction project is a valuable tool for financial institutions to accurately predict whether a loan applicant is eligible for a loan or not based on their input data. By leveraging machine learning algorithms such as SVM, developers can train models to analyze large datasets and identify patterns and trends that can help financial institutions make informed decisions.

The project also demonstrates the importance of data preprocessing and feature engineering in machine learning. Libraries such as NumPy and Pandas provide powerful tools for cleaning and transforming data, while Seaborn helps visualize the relationships between different features. By carefully selecting and engineering features, developers can improve the accuracy of the machine learning model and provide more meaningful insights.

Finally, the project highlights the role of software development tools such as Python, Jupyter Notebook, and scikit-learn in enabling efficient and effective development of machine learning models. By leveraging these tools, developers can quickly prototype and test different models, fine-tune their parameters, and evaluate their performance on test datasets.

Overall, the loan eligibility prediction project serves as an excellent example of how machine learning can be applied in the financial industry to improve decision-making and provide better services to customers. With further refinement and optimization, such models can provide invaluable insights and help financial institutions stay ahead in a rapidly evolving industry.

6.3 Future Enhancements

- Moving the loan eligibility prediction project to the AWS cloud has several potential benefits, including scalability, security, and cost-effectiveness. By leveraging the cloud's infrastructure and services, developers can optimize the project's performance and reduce maintenance and deployment costs.
- One possible enhancement to the project could be to use Amazon SageMaker, a fully managed machine learning service, to train and deploy the machine learning models. SageMaker provides pre-built machine learning algorithms, such as SVM, and allows developers to easily upload and preprocess data, train and evaluate models, and deploy them for production use. The service also supports model monitoring and automatic model tuning, which can help optimize the model's performance over time.
- Another enhancement could be to use AWS Lambda, a serverless computing service, to deploy the project's web application. Lambda allows developers to run code in response to specific events, such as user requests or data updates, without the need to provision or manage servers. By using Lambda, developers can reduce infrastructure costs and improve the application's scalability and availability.
- Furthermore, AWS provides various services to enhance the security of the project. AWS Identity and Access Management (IAM) enables administrators to control who has access to the project and what actions they can perform. AWS Key Management Service (KMS) can be used to encrypt data at rest and in transit. AWS CloudTrail can help monitor and audit the project's activity, and AWS Config can help ensure compliance with security policies and regulations.

- In addition to the technical enhancements, moving the project to the cloud can also provide various business benefits. For example, AWS provides flexible pricing models, including pay-as-you-go and reserved instances, which can help financial institutions reduce infrastructure costs and optimize their spending. AWS Marketplace also provides access to third-party tools and services that can help enhance the project's functionality and provide additional insights.

Overall, moving the loan eligibility prediction project to the AWS cloud can provide several benefits, including scalability, security, and cost-effectiveness. By leveraging AWS services and infrastructure, developers can optimize the project's performance and reduce maintenance and deployment costs, while providing better services to customers and staying ahead in a rapidly evolving industry.

REFERENCES

- 1) Mohamed Alaradi and Sawsan Hilal, “Tree-Based Methods for Loan Approval”, 2020 International Conference on Data Analytics for Business and Industry
- 2) C. Agarwal, and Musigchai, “Background Note on Data Collection Techniques”, IFC Bulletin No 30, 2011. Available: <https://www.bis.org/ifc/publ/ifcb30c.pdf>
- 3) L. Al-Blooshi and H. Nobanee, “Applications of Artificial Intelligence in Financial Management Decisions: A Mini-Review”, SSRN Electronic Journal, 2020.
- 4) W. Kluwer, Bizfillings, “What Banks Look for when Reviewing a Loan Application”, 2019. Available: Bizfillings.com: <https://www.bizfilings.com/toolkit/research-topics/finance/businessfinance/what-banks-look-for-when-reviewing-a-loan-application>
- 5) Moody's ANALYTICS, “Improve the Loan Approval Process by Implementing the Credit Decision Strategy”, n.d. Available: <https://www.omega-performance.com/improve-the-loan-approvalprocess-by-implementing-a-credit-decision-strateg>
- 6) V. Nagajyothi, “Loan approval prediction using KNN, decision Tree and Naïve Bayes models”, International Journal of Engineering in Computer Science, vol. 2, pp. 32-37, 2020.
- 7) K. Arun, G. Ishan, and K. Sanmeet, “Loan Approval Prediction based on Machine Learning Approach”, IOSR Journal of Computer Engineering, pp. 18-21, 2009.

- 8) R. Kumar, V. Jain, P.S. Sharma, S. Awasthi, and G. Jha. "Prediction of Loan Approval using Machine Learning", International Journal of Advanced Science and Technology, vol. 28, pp. 455-460, 2019.
- 9) Analytics Vidhya, "Loan Prediction Practice Problem", 2020. Available: <https://datahack.analyticsvidhya.com/contest/practiceproblem-loan-prediction-iii>
- 10) X. Francis Jency, V.P. Sumathi, and J. Shiva Sri, "An Exploratory Data Analysis for Loan Prediction Based on Nature of the Clients". International Journal of Recent Technology and Engineering, vol. 7, 2018.
- 11) G. James, D. Witten, T. Hastie, and R. Tibshirani, "An Introduction to Statistical Learning: with Applications in R". New York: Springer, 2013.
- 12) Institute for Science & Computing - University of Miami, "Decision Tree: Introduction". n.d. Available: http://web.ccs.miami.edu/~hishwaran/papers/decisionTree_intro_IR2009_EMDM.pdf
- 13) My R Codes Archive, "Area Under Curve (AUC) - pROC package", 2013. Available: <http://myrcodes.blogspot.com/2013/12/area-undercurve-auc-proc-package.html>
- 14) V. Ganganwar, "An Overview of Classification Algorithms for Impalanced Datasets", International Journal of Emerging Technology and Advanced Engineering, vol. 2, 2012.