Predictive Modelling for Loan Approval Rates

A Project Work Report

Submitted by:

VATHDA LIKHTIH SAI – 20BCS4593

HITESH GARG - 20BCS6614

JAIN KARAN ANAND - 21BCS8804

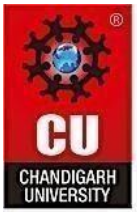
MITALI GUPTA - 20BCS6890

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING



CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Under the Supervision of:

Prof. Pramod Vishwakarma

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING APEX INSTITUTE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI – 140413, PUNJAB

May, 2024

Bonafide Certificate

Certified that this project report "**Predictive Modeling for Loan Approval Rates**" is the Bonafede work of **"Vathada Likhith Sai", "HITESH GARG", "JAIN KARAN ANAND", "MITALI GUPTA"** who carried out the project work under my/our supervision.

Prof. Pramod Vishwakarma

SUPERVISOR

HEAD OF THE DEPARTMENT

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I, “Vathada Likhith Sai”, “HITESH GARG”, “JAIN KARAN ANAND”, “MITALI GUPTA" students of "Bachelor of Engineering in Internet of Things," session: 2020-24, from the Department of Computer Science and Engineering at Apex Institute of Technology, Chandigarh University, Punjab, proudly present our Project Work titled "Predictive Modelling for Loan Approval Rates." We attest that this endeavor reflects our genuine efforts and adheres to the highest standards of Engineering Ethics. It represents our original work, devoid of any previously published or plagiarized content. We affirm that no aspect of this project has been submitted for any other academic recognition elsewhere. With meticulous care and unwavering dedication, we have crafted a solution that redefines convenience and security in user authentication. Together, we embrace innovation and uphold academic integrity in every facet of our academic journey.

Date:

Place:

CHANDIGARH UNIVERSITY,
GHARUAN, MOHALI –140413,
PUNJAB

ACKNOWLEDGMENT

Presentation inspiration and motivation have always played a key role in the success of any venture.

We pay our deep sense of gratitude to “Dr. U Hariharan”, Project Supervisor to encourage us to the highest peak and to provide us the opportunity to prepare the project. I am immensely obliged to our teachers for their elevating inspiration, encouraging guidance and kind supervision in the completion of our project.

We also want to thank Chandigarh University faculty members and others to provide us such an opportunity which made us learn something new and interesting.

Last but not the least, our parents are also an important inspiration for us. So, with due regards, we express our gratitude to them.

Date:

Vathada Likhith Sai(20BCS4593)
Hitesh Garg (20BCS6614)
Karan Jain Anand (21BCS8804)
Mitali Gupta (20BCS6890)

LIST OF FIGURES

TABLE OF CONTENTS

ABSTRACT

In today's financial landscape, predicting loan approval rates accurately is vital for ensuring responsible lending practices and managing risk effectively. This project aims to develop a predictive model for loan approval rates using machine learning techniques. The project leverages historical loan data, encompassing various applicant attributes such as credit score, income, employment status, and debt-to-income ratio. Through meticulous data preprocessing, exploratory data analysis, and feature selection, we identify the most relevant features for model development. Subsequently, we implement and evaluate several machine learning algorithms, including logistic regression, decision trees, random forest, and gradient boosting. The performance of each model is assessed using standard evaluation metrics, and the best-performing model is identified. The results of this project offer insights into the factors influencing loan approval rates and provide a valuable tool for financial institutions to make informed lending decisions.

Deep Learning, Open AI, Open CV, Artificial Intelligence, Concept Generation. Technology has boosted the existence of human kind 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.

The subsequent phase of exploratory data analysis serves as an illuminating expedition into the dataset's landscape, unraveling nuanced relationships, distributions, and insights among variables. Visualizations illuminate patterns and trends, offering a deeper understanding of the underlying dynamics governing loan approval rates across various demographic cohorts. Feature selection, a pivotal stage in model construction, is undertaken with meticulous precision. Employing a blend of correlation analysis, feature importance metrics, and domain expertise, the most salient features driving loan approval decisions are identified and curated for model input.

Keywords: Win Duino: Say goodbye to logins, hello to seamless access with a tap.

Chapter 1: Introduction

Due to financial constraints on business and economic development, bank loans have become an important source of external financing for businesses and households. Most banks only benefit from lending, but the increase in lending brings with it many risks such as default risk or credit risk, which results in the borrower being unable to repay the loan as agreed. Banks have limited resources, so it is important to choose the right applicant and repay the loan on time. Selection of suitable candidates plays an important role in banking. Many problems arise in choosing the right reviewer when the process is done manually. Banks need to choose the right one, otherwise the bank will face financial problems and become unprofitable. The aim of banks is to deposit their assets in safe hands. If a bank lends money to someone, the bank must explain the purpose of the loan, whether they will repay the loan, whether their information is valid, etc. things need to be taken into consideration. These machine learning models provide support to both employers and applicants. The main purpose of this model is to find the best option and reduce the selection time. There are many ways to predict, but data is worth saving as it only requires historical data from customers and trains the process to predict approval. Some of the estimators used here are Naive Bayes, Logistic Regression, Support Vector Machine, Distribution, Random Forest. Therefore, we can easily predict loan approval depending on the accuracy of the algorithms used. According to machine learning, we have two types of data sets; one is training data, the other is testing data. Here, each model makes a few differences and tries to give us the results of whether we approve someone's loan or not. The main purpose of this article is to create a credit structure that is more difficult to predict.

This project proposes a novel approach to classification problem where we have to predict whether a loan will be approved or not. Specifically, it is a binary classification problem where we have to predict either one of the two classes given i.e. approved (Y) or not approved (N). Another way to frame the problem is to predict whether the loan will likely to default or not, if it is likely to default, then the loan would not be approved, and vice versa. The dependent variable or target variable is the Loan_Status, while the rest are independent variable or features. We need to develop a model using the features to predict the target variable.

Loan prediction is a very common real-life problem that every retail bank faces in their lending operations. If the loan approval process is automated, it can save a lot of man hours and improve the speed of service to the customers. The increase in customer satisfaction and savings in operational costs are significant. However, the benefits can only be reaped if the bank has a robust model to accurately predict which customer's loan it should approve and which to reject, in order to minimize the risk of loan default.

They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan. Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers."

Dream Housing Finance company deals in all kinds of home loans. They have a presence across all urban, semi-urban and rural areas. The customer first applies for a home loan and after that, the company validates the customer eligibility for the loan.The company wants to automate the loan eligibility process (real-time) based on customer detail provided while filling out online application forms. These details are Gender, Marital Status, Education, number of Dependents, Income, Loan Amount, Credit History, and others.

To automate this process, they have provided a dataset to identify the customer segments that are eligible for loan amounts so that they can specifically target these customers.

Harnessing machine learning models for loan approval prediction is on the cutting edge of lending technology. These models can learn patterns from historical data to predict new loan approvals. They can provide quicker and more datadriven decisions by automating the loan approval process, allowing continuous improvement through monitoring and feedback. Fintech firms are increasingly looking to evaluate a credit applicant's ability and willingness to repay and speed up the loan underwriting process using machine learning. The table below offers 10 steps fintech vendors could use to leverage machine learning for loan approval prediction.

With the enhancement in the banking sector lots of people are applying for bank loans but the bank has its limited assets which it has to grant to limited people only, so finding out to whom the loan can be granted which will be a safer option for the bank is a typical process. So in this paper we try to reduce this risk factor behind selecting the safe person so as to save lots of bank efforts and assets. This is done by mining the Big Data of the previous records of the people to whom the loan was granted before and on the basis of these records/experiences the machine was trained using the machine learning model which give the most accurate result. The main objective of this paper is to predict whether assigning the loan to particular person will be safe or not. This paper is divided into four sections (i)Data

Collection (ii) Comparison of machine learning models on collected data (iii) Training of system on most promising model (iv) Testing. Loan Approval Prediction using Machine Learning

Assessing the risk, which is involved in a loan application, is one of the most important concerns of the banks for survival in the highly competitive market and for profitability. These banks receive number of loan applications from their customers and other people on daily basis. Not everyone gets approved. Most of the banks use their own credit scoring and risk assessment techniques in order to analyze the loan application and to make decisions on credit approval. In spite of this, there are many cases happening every year, where people do not repay the loan amounts or they default, due to which these financial institutions suffer huge amount of losses. Loan Approval Prediction using Machine Learning\

The primary goal of this project is to extract patterns from a common loan-approved dataset, and then build a model based on these extracted patterns, in order to predict the likely loan defaulters by using classification data mining algorithms. The historical data of the customers like their age, income, loan amount, employment length etc. will be used in order to do the analysis. Later on, some analysis will also be done to find the most relevant attributes, i.e. The factors that affect the prediction result the most. Using a different type of Machine Learning algorithm and Predicting an Accuracy Result and Plotting a graph.

Based on the domain knowledge, we can come up with new features that might affect the target variable. We will create the following three new features:

- **Total Income** — As discussed during bivariate analysis we will combine the Applicant Income and Coapplicant Income. If the total income is high, the chances of loan approval might also be high.

- **EMI** — EMI is the monthly amount to be paid by the applicant to repay the loan. The idea behind making this variable is that people who have high EMI's might find it difficult to pay back the loan. We can calculate the EMI by taking the ratio of the loan amount with respect to the loan amount term.

- **Balance Income** — This is the income left after the EMI has been paid. The idea behind creating this variable is that if this value is high, the chances are high that a person will repay the loan and hence increasing the chances of loan approval.

1.1 Problem Definition

With the increase in banking sector many people are applying for loans in bank. All these loans are not approvable. The main income of bank assets comes from gain earned from loans. The main objective of banks is to invest their assets in safe customers. Today many banks approve loan after many process of verification and validation but still there is no surety that selected customer is safe or not. Therefore it is important to apply various techniques in banking sector for selecting a customer who pays loan on time. In this report we use random forest algorithm for the classification of data. Random forests algorithm builds a model from trained dataset and this model is applied on test data and we get the required output.

1.2 Project Overview/Specification

Banks, Housing Finance Companies and some NBFC deal in various types of loans like housing loan, personal loan, business loan etc., in all over the part of countries. These companies have existence in Rural, Semi Urban and Urban areas. After applying loan by customer these companies validates the eligibility of customers to get the loan or not. This project provides a solution to automate this process by employing machine learning algorithm. So the customer will fill an online loan application form. This form consist details like Sex, Marital Status, Qualification, Details of Dependents, Annual Income, Amount of Loan, Credit History of Applicant and others.

1.3 Hardware Specification

1. PC / Laptop
2. Keyboard
3. Mouse

1.4 Software Specifications

1. Anaconda Navigator
   Anaconda is a popular Python distribution that includes a variety of tools and libraries for data science and machine learning. It is designed to be easy to use and includes features such as a graphical user interface, pre-installed packages, and support for multiple programming languages. Anaconda is widely used in industry and academia and is a popular choice for data science and machine learning projects.

2. Jupyter Notebook
   Jupyter is a free and open-source computing platform that runs on a local computer or remote server. Jupyter Notebook is a web-based application that allows users to create, organize, and edit documents containing live code, equations, visualizations and narrative text. Jupyter is popular among data scientists, researchers, and scientists for its ease of use and flexibility in handling multiple programming languages, including Python, R, and Julia. It's also widely used for machine learning, statistical analysis, and scientific computing.

Chapter 2: Literature Survey

2.0 Existing System

Historically, lending risk prediction has used statistical methods, including Linear Discriminant Analysis and Logistic Regression. However, with large credit datasets, ML-driven risk estimation algorithms like k-Nearest Neighbor, Random Forest, and Support Vector Machines are better at capturing complex relationships.

Moreover, deep learning methods have gained a particular advantage in modeling non-linear relationships between risk and risk factors for large-scale lending risk and loan prediction datasets. Novel frameworks like DEAL (Deep Ensemble Algorithm), or improvements over existing models of Recurrent Neural Networks (RNN) or Boosted Decision Tree or Autoencoders, give satisfactory accuracy over large datasets and generate features with domain expertise. However, more work is available on machine learning models than deep learning architectures since the latter's performances are often specific to the dataset they were designed and tested on. We can see that algorithms like SVM, Random forest, and the Logistic regression model perform better than ELM and ANN. However, decision trees and boosting also give a competitive performance on this dataset.

Machine Learning Techniques for Loan Default Prediction: This review article by Brown et al. explores the application of machine learning techniques in loan default prediction. It discusses the strengths and limitations of different algorithms and provides insights into feature engineering and model interpretability.

Predicting Loan Defaults in Peer-to-Peer Lending: Focusing on the domain of peer-to-peer lending, this review by Smith et al. surveys predictive modeling approaches for loan default prediction. The paper discusses the challenges specific to peer-to-peer lending platforms and evaluates the effectiveness of machine learning algorithms in this context.

Feature Selection Techniques in Machine Learning with Applications to Credit Risk Assessment: This research paper by Garcia et al. delves into feature selection techniques for credit risk assessment. It compares various feature selection methods and their impact on model performance, providing practical insights for building predictive models in the financial domain.

Literature Review Summary (Minimum 7 articles should refer)

Year and Citation	Article/ Author	Source	Evaluation Parameter

2011	K. Hanumantha Rao., G. Srinivas., A. Damodhar., M.Vikas Krishna	Vol. 2, Issue 3, 2011	Implementation of Anomaly Detection Technique Using Machine Learning Algorithms
2002	S.S. Keerthi., E.G. Gilbert	Vol. 4, Issue 1, pp. 351360, 2002	Convergence of a generalize SMO algorithm for SVM classifier design, Machine Learning
2002	Andy Liaw., Matthew Wiener	Vol. 2, Issue 3, pp. 9-22	Classification and Regression by random Forest
2014	Ekta Gandotra., Divya Bansal., Sanjeev Sofat	Vol. 05, Issue 02, pp. 5664	Malware Analysis and Classification: A Survey, Journal of Information Security
2013	Aafer Y., Du W., Yin H., Droid	pp 86-103	Mining API-Level Features for Robust Malware Detection in Android, Security and privacy in Communication Networks

2021	J. R. Quinlan	Vol. 1, No. 1. pp. 81-106.	Induction of Decision Tree, Machine Learning
2022	Q. Zhang, J. Yang, X. Zhang and T. Cao	(ICIVC), Xi'an, China, 2022, pp. 848-853, doi: 10.1109/ICIVC55077.2022.9886154.	Generating Adversarial Examples in Audio Classification with Generative Adversarial Network

2.1 Proposed System

The proposed system is Loan Approval System software used for approval of loan in banking sector. In this proposed system we have used machine learning algorithm. Machine Learning is process in which a symmetric model is build from the existing dataset; this model is applied for the testing of the new dataset. The system consists of trained dataset and test dataset. The trained dataset is used for construction of model. This model is applied on testing dataset for the required result. We have used Ensemble approach for building of the model. Random forest algorithm uses this ensemble approach and builds a model from the existing training dataset.

System Name: Automated Loan Approval Prediction System (ALAPS)

System Components: 1. **Data Acquisition Module:** ○ Integrates with various data sources like credit bureaus, bank accounts, and internal loan applications.

- ○ Performs data cleaning and preprocessing to ensure data quality.

2. **Model Training Module:** ○ Houses pre-built machine learning models like Logistic Regression, Random Forest, and SVM. ○ Allows for easy integration and experimentation with new models. ○ Splits the incoming clean data into training and testing sets.
   - ○ Trains the chosen model(s) on the training data.

3. **Model Evaluation Module:** ○ Evaluates the trained model(s) on the testing data using metrics like accuracy, precision, recall, and F1 score.
   - ○ Provides visualization tools to understand model performance and identify potential biases.

- Allows for hyperparameter tuning to optimize model performance.

4. **Loan Application Scoring Module:** ○ Takes a new loan application as input. ○ Extracts relevant features from the application data. ○ Feeds the extracted features into the chosen, well-performing model.
   - Generates a loan approval probability score for the applicant.
5. **Decision Support Module:** ○ Presents the loan application details, credit report (if available), and the approval probability score to the loan officer.
   - Allows the loan officer to set score thresholds for auto-approval, manual review, or rejection.
   - Provides explanations for the model's predictions to enhance transparency.
6. **Reporting & Monitoring Module:** ○ Tracks the performance of the deployed model over time. ○ Identifies potential performance degradation or bias creep.
   - Generates reports on approval rates, model fairness metrics, and key performance indicators (KPIs). **Benefits of ALAPS:**

- **Improved Efficiency:** Streamlines loan processing by auto-approving low-risk applications.
- **Enhanced Accuracy:** Provides data-driven predictions to improve decision-making.
- **Reduced Risk:** Minimizes defaults by identifying high-risk borrowers.
- **Fairer Lending:** Promotes fair and unbiased loan decisions based on objective criteria. **Increased Transparency:** Explains model predictions for better understanding. **Security Considerations:**

- Implement robust access controls and data encryption.
- Regularly monitor for security vulnerabilities.
- Ensure compliance with data privacy regulations.

Overall, ALAPS offers a comprehensive solution for loan approval prediction. It can empower lenders with datadriven insights while promoting responsible and fair lending practices

PROBLEM FORMULATION

The main purpose of this project is to provide immediate and accurate results for the approval of loan to the eligible customers. In banking sector there will be n number of people who apply loans. It is difficult to check customer's eligibility through paper work. The system can provide accurate results for then number of people. In this project we have discussed about credit risk and credit analysis. Banking sectors success mainly depends of credit risk analysis. In this report we have used Random Forest [3] approach to build the model. The use of Random Forest is because Random Forest Approach provides accurate results than the K Nearest Neighbor and Decision Tree. In this project we have used Random Forest approach for building a model. In this report two or more classifiers are combined together and identify a perfect model for loan prediction. Ensemble method compares two or more models and identifies a perfect model from two or more models for better loan prediction which makes banking sector to make a right choice for approval of loan application.

PROJECT OBJECTIVES

The key objectives of Predictive Modelling for Loan Approval Rate, increase efficiency, promote user adoption, streamline authentication procedures, and strengthen security measures. By fulfilling these goals:

Develop Accurate Predictive Models: The primary objective is to develop predictive models that accurately forecast the likelihood of loan approval based on applicant information. These models should leverage advanced machine learning algorithms to achieve high levels of predictive accuracy. Enhance Risk Assessment: Improve the risk assessment process by integrating predictive modeling techniques that can effectively evaluate the creditworthiness of loan applicants. This includes identifying applicants who are more likely to default on their loans and assessing the overall risk exposure of the lending portfolio. Optimize Decision-Making: Enable financial institutions to make data-driven decisions regarding loan approvals by providing insights derived from predictive models. This involves identifying key factors that influence loan approval rates and using these insights to optimize lending strategies and policies. Improve Efficiency: Streamline the loan approval process by automating certain tasks through predictive modeling. By accurately predicting loan approval rates, financial institutions can reduce manual effort and expedite the decisionmaking process, leading to increased operational efficiency. Increase Transparency and Fairness: Promote transparency and fairness in the loan approval process by ensuring that predictive models are built using unbiased data and transparent methodologies. This includes validating models for fairness and interpretability and ensuring compliance with regulatory requirements. Mitigate Risk Exposure: Minimize the risk exposure of financial institutions by identifying and mitigating potential risks associated with loan approvals. This involves identifying patterns of fraudulent behavior, detecting anomalies in loan applications, and implementing risk mitigation strategies based on predictive insights. Continuous Improvement: Continuously monitor and refine predictive models to ensure their effectiveness and relevance over time. This includes incorporating new data sources, updating model parameters, and retraining models periodically to adapt to changing market conditions and customer behaviors.

METHODOLOGY:

Data Collection: Gather historical loan data from financial institutions, including applicant information, loan attributes, and approval outcomes. Data Preprocessing: Handle missing values: Impute missing data using appropriate techniques such as mean imputation or predictive imputation.

Encode categorical variables: Convert categorical variables into numerical format using techniques like one-hot encoding or label encoding. Scale numerical features: Standardize numerical features to ensure they have a similar scale and distribution.

Exploratory Data Analysis (EDA): Conduct descriptive statistics to understand the distribution of variables. Visualize relationships between features and loan approval outcomes. Identify potential outliers and anomalies that may require further investigation.
Feature Engineering: Create new features from existing ones to capture additional information. Transform variables if necessary to improve model performance (e.g., log transformation for skewed distributions).
Feature Selection: Use techniques such as correlation analysis, feature importance ranking, and domain knowledge to select the most relevant features. Employ dimensionality reduction techniques like PCA (Principal Component Analysis) if dealing with high-dimensional data.

Model Selection and Training: Choose appropriate machine learning algorithms such as logistic regression, decision trees, random forest, gradient boosting, or neural networks. Split the data into training and testing sets to evaluate model performance. Train multiple models and compare their performance using evaluation metrics like accuracy, precision, recall, F1-score, and ROC-AUC.

Model Evaluation: Assess the performance of each model using the testing dataset. Fine-tune hyperparameters using techniques like grid search or random search to optimize model performance. Validate models using techniques like cross-validation to ensure robustness and generalizability.

Model Interpretability: Utilize techniques such as SHAP (SHapley Additive exPlanations) values, LIME (Local Interpretable Model-agnostic Explanations), or feature importance plots to interpret model predictions. Provide explanations for model decisions to enhance transparency and trust in the predictive model.

Deployment and Integration: Deploy the trained model into the production environment, integrating it with the loan approval system. Develop APIs or web services to enable real-time scoring of loan applications. Monitor model performance in the production environment and retrain/update the model as needed.

Documentation and Reporting: Document the entire methodology, including data preprocessing steps, feature engineering techniques, model selection criteria, and evaluation results. Prepare a

comprehensive project report outlining the methodology, findings, and recommendations for stakeholders.

Chapter 3: Design flow/Process

Data Collection: Gather historical data on loan applications from various sources such as banking databases, credit bureaus, and financial institutions. Ensure that the data includes information about applicants, loan details, and approval status.

Data Preprocessing: Handle Missing Values: Impute missing values using techniques like mean/mode imputation or predictive imputation. Outlier Detection and Removal: Identify and remove outliers that could adversely affect the model's performance.

Data Encoding: Encode categorical variables using techniques like one-hot encoding or label encoding.

Feature Scaling: Scale numerical features to a similar range to prevent any bias in model training.

Feature Engineering: Create new features: Generate additional features that may provide valuable insights for predicting loan approval, such as debt-to-income ratio, loan-to-income ratio, and credit utilization ratio.

Feature Selection: Select the most relevant features using techniques like correlation analysis, feature importance from tree-based models, or dimensionality reduction techniques like PCA (Principal Component Analysis).

Model Selection: Choose appropriate algorithms: Select machine learning algorithms suitable for binary classification tasks, such as logistic regression, decision trees, random forests, gradient boosting machines (GBM), or neural networks.

Ensemble methods: Consider ensemble techniques like bagging (e.g., Random Forests) or boosting (e.g., AdaBoost, XGBoost) for improved predictive performance. Model Training: Split the data: Divide the preprocessed data into training and testing sets (e.g., 70/30 or 80/20 split).

Train the model: Fit the selected algorithms on the training data to learn the patterns and relationships between features and loan approval status.

Model Evaluation: Evaluate model performance: Assess the performance of the trained model using appropriate evaluation metrics such as accuracy, precision, recall, F1-score, and ROC AUC. Cross-validation: Perform k-fold cross-validation to validate the model's performance and ensure its robustness.

Model Tuning: Hyperparameter tuning: Fine-tune the hyperparameters of the selected model(s) using techniques like grid search or randomized search to optimize performance. Regularization: Apply regularization techniques like L1 or L2 regularization to prevent overfitting.

Model Deployment: Deploy the trained model into a production environment where it can be integrated into the loan approval process. Develop APIs or web services to facilitate model inference

for new loan applications. Implement monitoring mechanisms to track the model's performance and ensure timely updates if necessary.

Workflow: Data Collection and Preprocessing: Gather historical loan application data from relevant sources. Clean the data by handling missing values, outliers, and encoding categorical variables. Feature Engineering and Selection: Create new features and select the most relevant ones for modeling.

Model Development: Choose appropriate algorithms and split the data into training and testing sets. Train the model on the training data and evaluate its performance using validation techniques.

Model Optimization: Fine-tune the model's hyperparameters and apply regularization techniques to improve its performance. Model Evaluation: Assess the model's performance using various evaluation metrics and cross-validation techniques.

Model Deployment: Deploy the trained model into a production environment, ensuring integration with the loan approval process. Implement monitoring mechanisms to track the model's performance and make updates as needed.

Maintenance and Updates: Continuously monitor the model's performance and retrain it periodically with new data to ensure its accuracy and relevance over time. Stay updated with changes in regulations and industry standards to maintain compliance and fairness in loan approval decisions.

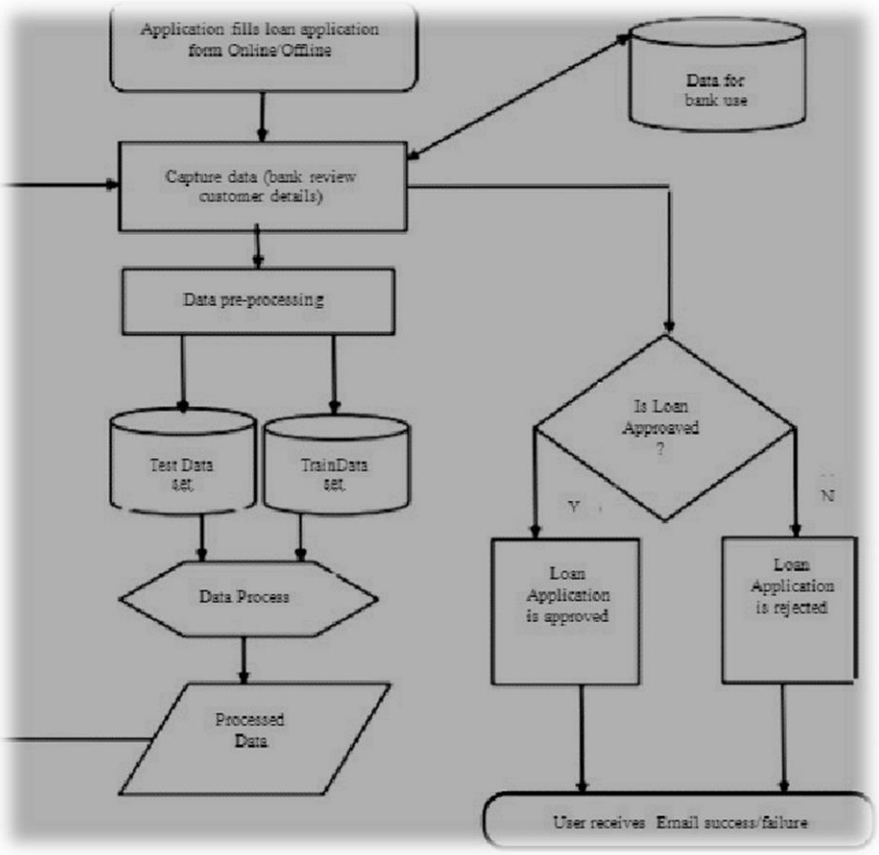
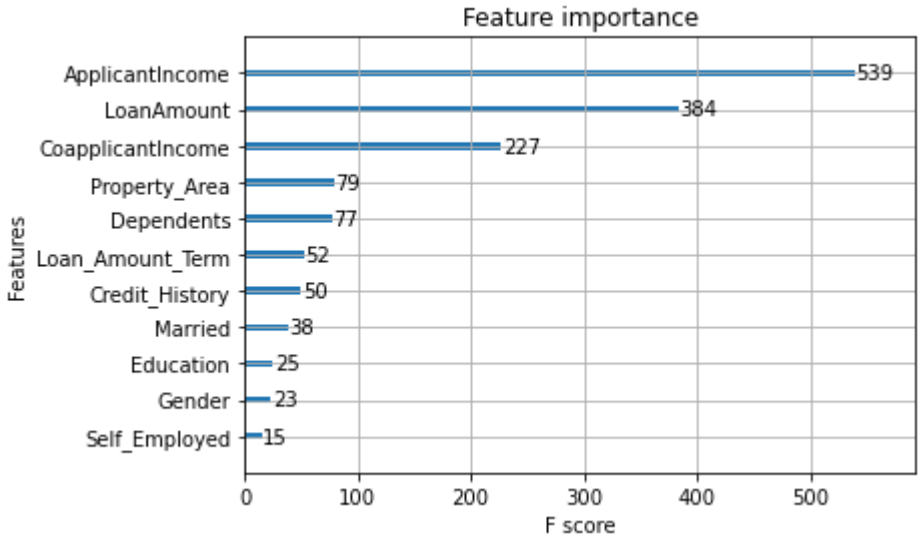
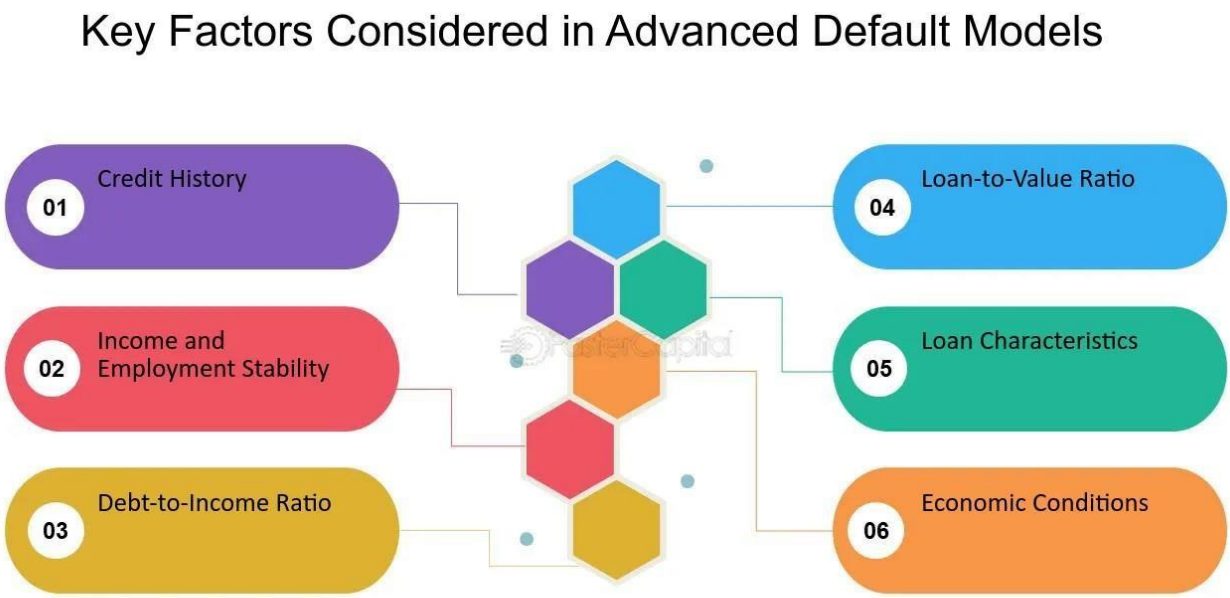


Fig 1.1 flow chart





Several key factors influence loan approval decisions. While specific criteria may vary depending on the lender and the type of loan, here are some common factors considered by most financial institutions:

Credit Score: A credit score is a numerical representation of an individual's creditworthiness based on their credit history. Lenders typically use credit scores from credit bureaus (e.g., FICO Score, VantageScore) to assess the risk associated with lending to an individual. Higher credit scores indicate lower credit risk and may result in better loan terms and higher approval chances.

Income and Employment Stability: Lenders evaluate an applicant's income level and employment history to determine their ability to repay the loan. Stable employment and a steady income stream are viewed favorably as they indicate financial stability and the capacity to meet loan obligations.
Debt-to-Income Ratio (DTI): The DTI ratio compares an individual's total monthly debt payments to their gross monthly income. Lenders use this ratio to assess the borrower's ability to manage additional debt responsibly. A lower DTI ratio indicates that the borrower has sufficient income to cover existing debts as well as the proposed loan payments.

Loan-to-Value Ratio (LTV): For secured loans (e.g., mortgages, auto loans), lenders consider the LTV ratio, which compares the loan amount to the appraised value of the collateral. A lower LTV ratio implies less risk for the lender, as there is a greater equity cushion in the collateral to cover potential losses in case of default.

Loan Purpose: The purpose of the loan may influence the approval decision. Lenders may have specific eligibility criteria or risk preferences based on the intended use of the funds. For example, loans for education or home purchase may be viewed more favorably than loans for speculative investments.

Credit History and Payment History: In addition to credit scores, lenders assess an applicant's credit history and payment behavior. A positive credit history with a record of timely payments demonstrates responsible financial management and increases the likelihood of loan approval.

Collateral (for Secured Loans): Collateral provides security for the lender in case of default. The type and value of collateral offered by the borrower may impact the lender's decision to approve the loan and the terms offered.

Down Payment (for Some Loans): For certain types of loans, such as mortgages or auto loans, a larger down payment reduces the loan amount and associated risk for the lender. Applicants offering a significant down payment may have a higher chance of loan approval and may qualify for better terms.

Credit Utilization: Lenders consider how much of the available credit a borrower is currently using. High credit utilization suggests a higher risk of default, whereas low utilization indicates responsible credit management.

Regulatory and Compliance Factors: Lenders must comply with regulatory requirements and lending standards established by government agencies and industry regulators. These factors may influence loan approval criteria, documentation requirements, and risk assessment processes.

What are the benefits of using predictive modeling techniques in loan performance analysis

1. Improved Credit Risk Assessment
2. Enhanced Loan Pricing
3. Early Warning Signals
4. Portfolio Optimization
5. Fraud Detection
6. Streamlined Decision-Making
7. Regulatory Compliance
8. Continuous Improvement

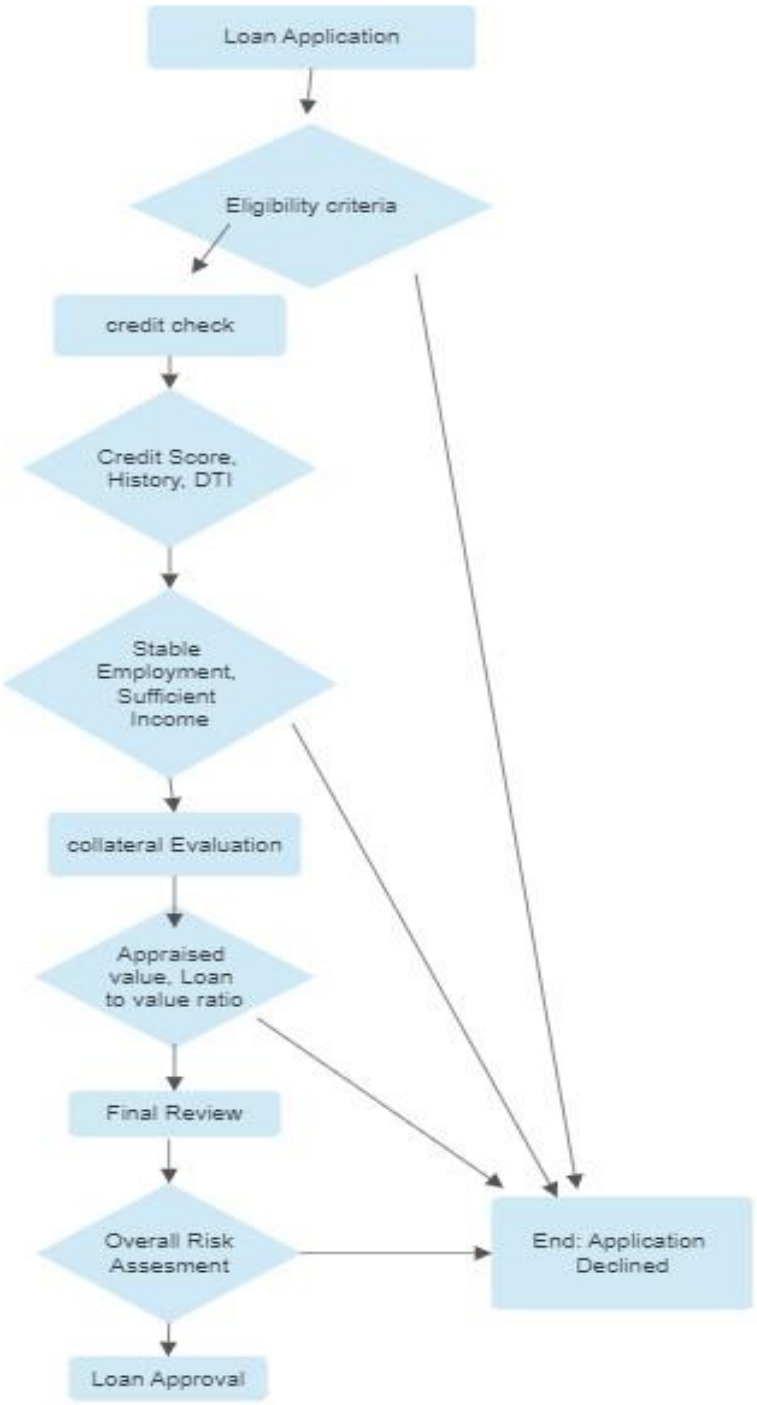
Streamlining Loan Approval Processes

Automated Document Collection and Verification

Enhanced Credit Analysis

Efficient Risk Assessment

Flowchart


Loan Application
Eligibility criteria
credit check
Credit Score, History, DTI
Stable Employment, Sufficient Income
collateral Evaluation
Appraised value, Loan to value ratio
Final Review
Overall Risk Assesment
End: Application Declined
Loan Approval

Flowchart Explanation:

Loan Application: This marks the beginning of the loan approval process, initiated when a borrower submits an application to a lender seeking financial assistance. The application typically includes personal information, loan amount, purpose, and sometimes details about collateral.
Initial Review (Eligibility Criteria): In this step, the lender assesses whether the loan application meets basic eligibility criteria set by the institution. This may include factors such as the applicant's age, citizenship, and the type of loan being requested. If the application satisfies these criteria, it proceeds to the next stage.
Credit Check: Once the application passes the initial review, the lender conducts a credit check to evaluate the applicant's creditworthiness. This involves obtaining the applicant's credit report from credit bureaus and analyzing factors such as credit score, payment history, and debt-to-income ratio (DTI). A higher credit score and positive credit history indicate lower credit risk**.**
Income Verification: Applicants with acceptable credit undergo income verification to ensure they have a stable source of income to repay the loan. Lenders may request documents such as pay stubs, tax returns, or bank statements to verify income. The goal is to assess the applicant's ability to meet monthly loan payments based on their income level.
Collateral Evaluation: For secured loans where collateral is offered to mitigate risk, the lender evaluates the collateral's value and condition. This involves appraising the collateral to determine its market value and calculating the loan-to-value (LTV) ratio, which compares the loan amount to the collateral's value. A lower LTV ratio indicates less risk for the lender.
Final Review: A final review is conducted to assess the overall risk associated with the loan application. This includes considering factors such as credit score, income stability, debt levels, and collateral value. The lender evaluates the applicant's financial profile comprehensively to make an informed decision. **Loan Approval or Decline**: Based on the results of the final review, the lender decides whether to approve or decline the loan application. If approved, the borrower receives the loan offer detailing terms and conditions. If declined, the borrower is notified of the decision, and the process ends.

Specifications and Features:

The **specifications and features** are listed below.

Scalability: Scalability is crucial for accommodating varying levels of demand and ensuring the system can handle a large volume of loan applications without performance degradation. The system should be designed with scalability in mind, utilizing distributed computing technologies and cloud-based infrastructure to scale up or down dynamically based on workload fluctuations. Horizontal scalability, achieved through load balancing and auto-scaling mechanisms, allows the system to add or remove computing resources as needed to meet demand spikes or accommodate growth. Additionally, efficient resource allocation and optimization techniques should be implemented to maximize system performance and minimize operational costs.

Security: Security is paramount in a loan approval system to protect sensitive applicant information and prevent unauthorized access or data breaches. The system should adhere to industry-standard security practices, including encryption of data in transit and at rest, robust authentication mechanisms (e.g., multifactor authentication), access controls, and audit trails. Role-based access control (RBAC) should be enforced to restrict access to sensitive features and data based on user roles and permissions. Regular security audits and penetration testing should be conducted to identify and remediate vulnerabilities, ensuring the system remains resilient against evolving cybersecurity threats.

Performance: Performance is essential for providing a seamless user experience and processing loan applications efficiently. The system should have low latency and high throughput to handle concurrent requests and deliver quick responses to applicants. Performance optimization techniques, such as caching, query optimization, and asynchronous processing, should be employed to minimize response times and maximize system throughput. Load testing and performance monitoring tools should be used to identify performance bottlenecks and optimize system components for optimal performance under various load conditions.

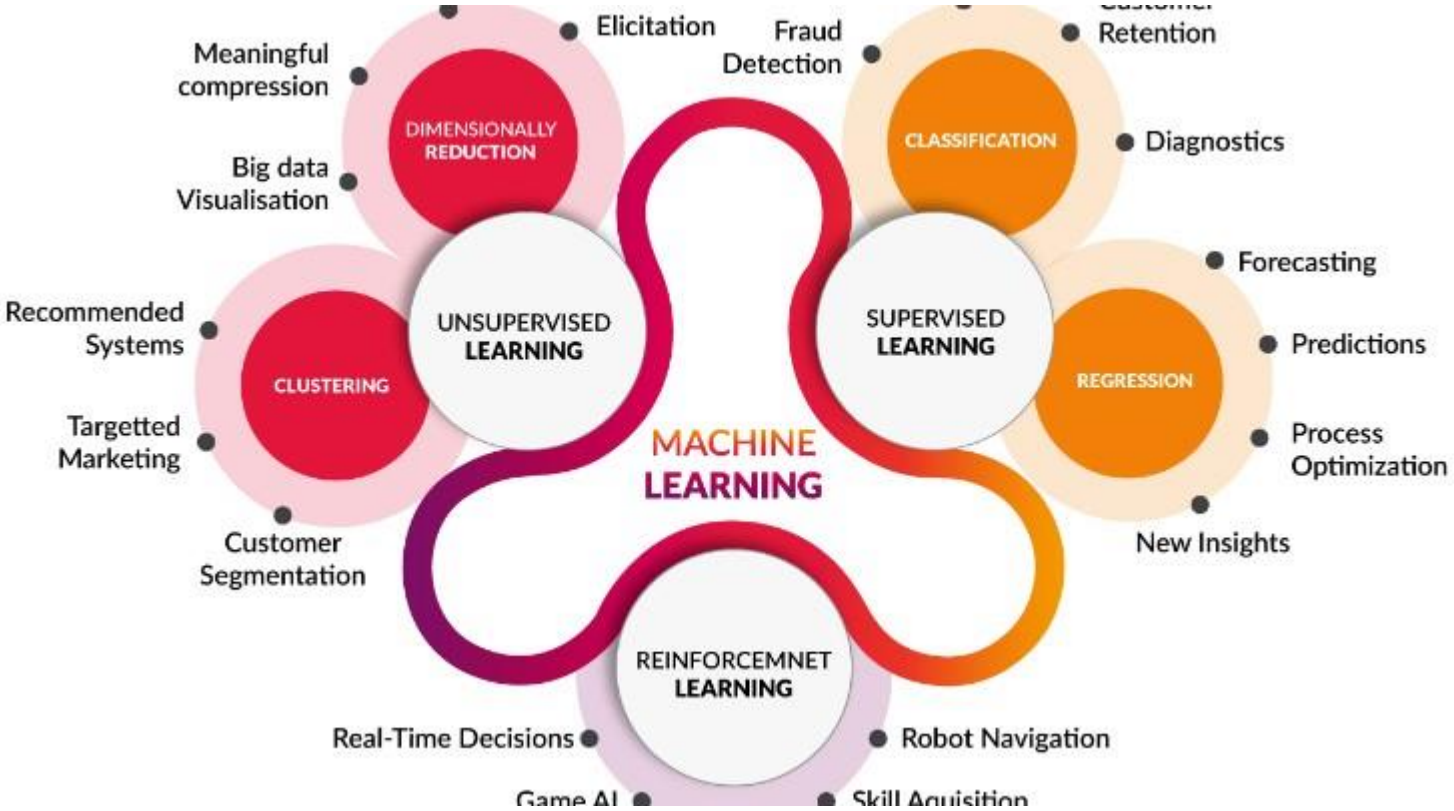
Integration: Seamless integration with external systems and data sources is critical for gathering necessary information to evaluate loan applications effectively. The system should integrate with banking systems, credit bureaus, identity verification services, and other third-party APIs to retrieve applicant data, credit reports, income verification, and collateral information. Application programming interfaces (APIs) and data integration technologies should be utilized to facilitate data exchange and interoperability between disparate systems while ensuring data integrity and security.

Compliance: Compliance with regulatory requirements and industry standards is paramount to mitigate legal and reputational risks associated with lending operations. The system should adhere to relevant regulatory frameworks, such as the Truth in Lending Act (TILA), Equal Credit Opportunity Act (ECOA), and Fair Credit Reporting Act (FCRA), ensuring fair lending practices and consumer protection. Compliance checks should be integrated into the loan approval workflow to verify adherence to anti-money laundering (AML), know your customer (KYC), and customer due diligence (CDD) requirements. Regulatory reporting capabilities should be built into the system to generate and submit compliance reports to regulatory authorities as mandated.

Customization: Flexibility and customization are essential to accommodate the unique requirements and policies of different lenders. The system should offer configurable parameters and business rules to tailor eligibility criteria, credit scoring models, approval workflows, and decisioning rules based on the lender's specific preferences and risk appetite. Customization options should be provided through user-friendly administrative interfaces, allowing administrators to modify system settings without requiring extensive technical expertise.

Auditability: Comprehensive auditability features should be incorporated into the system to maintain a detailed audit trail of all loan application activities and user interactions. Audit logs should capture relevant information, including user actions, system events, data modifications, and access attempts. Audit trails should be tamper-evident and immutable, ensuring the integrity and reliability of audit data for compliance, forensic analysis, and accountability purposes. Advanced logging and monitoring capabilities should be implemented to detect and alert on suspicious or unauthorized activities in real-time, enabling timely response and investigation.

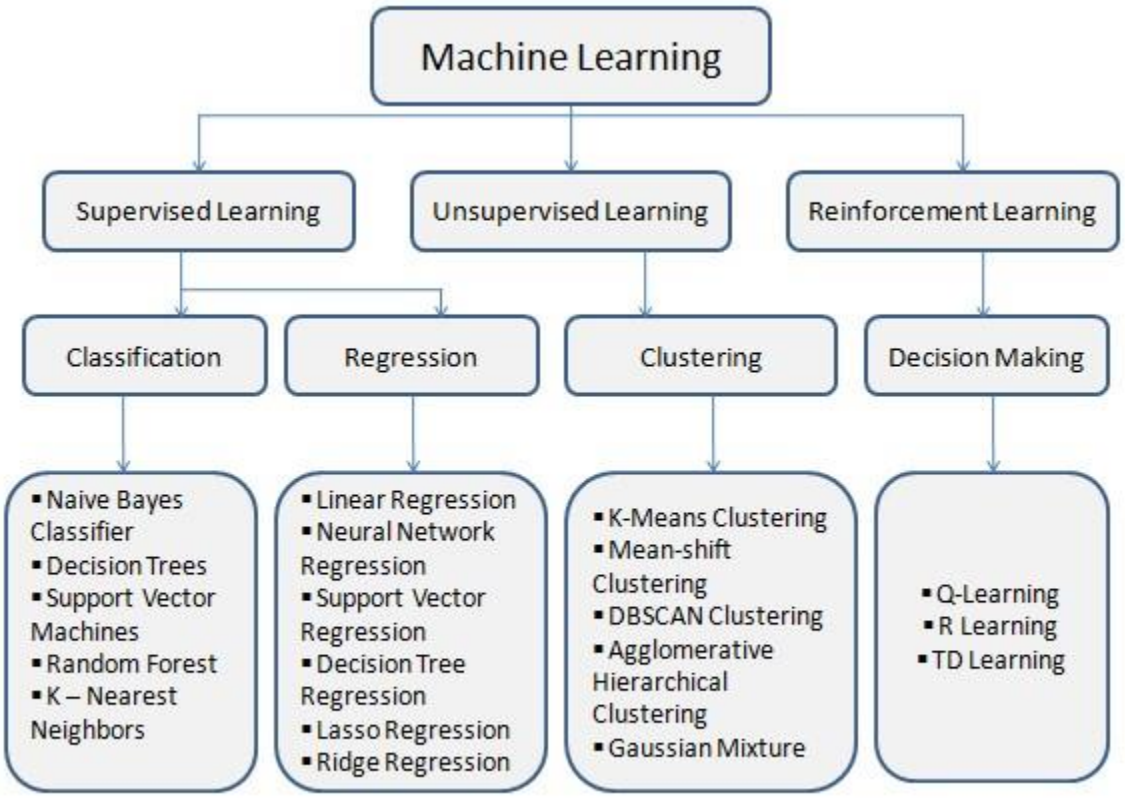
User Access Control: Granular user access control mechanisms should be implemented to manage user permissions and restrict access to sensitive features and data based on user roles, responsibilities, and organizational hierarchy. Role-based access control (RBAC) should be enforced to assign permissions to users based on predefined roles and privileges, allowing administrators to control access rights effectively. Access controls should be configurable and enforceable across all system components and modules, ensuring consistent application of security policies and minimizing the risk of unauthorized access or data leakage.



Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Types of learning algorithms:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve. Supervised learning: Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task. Supervised learning algorithms include classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification. 16 In the case of semi-supervised learning algorithms, some of the training examples are missing training labels, but they can nevertheless be used to improve the quality of a model. In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets. Unsupervised learning: Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, though unsupervised learning encompasses other domains involving summarizing and explaining data features. Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more pre designated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity. Semi-supervised learning: Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy



Benefits of Predictive Modeling for Loan Approval Rates:

1. Enhanced Accuracy and Efficiency: Predictive models leverage advanced analytics to discern subtle patterns and signals in the data, thereby improving the accuracy of loan approval predictions. By automating the decision-making process, lenders can expedite loan processing times, reduce manual errors, and enhance operational efficiency.

2. Risk Mitigation and Portfolio Optimization: Accurate risk assessment enables lenders to identify highrisk applicants and tailor loan terms accordingly to mitigate potential losses. By diversifying their loan portfolios and allocating resources prudently, financial institutions can optimize risk-return trade-offs and maintain a healthy balance between profitability and risk exposure.

3. Inclusive Lending Practices: Predictive modeling can promote financial inclusion by extending credit access to underserved populations who may lack traditional credit histories or collateral. Alternative data sources, such as utility payments, rental histories, and social media profiles, can supplement conventional credit information and enable lenders to assess the creditworthiness of previously overlooked segments of the population.

Challenges and Considerations:

1. Data Privacy and Regulatory Compliance: The proliferation of sensitive personal data in predictive modeling raises concerns about data privacy, security, and regulatory compliance. Lending institutions must adhere to stringent data protection laws such as GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act) and implement robust data governance frameworks to safeguard customer information and uphold ethical standards.

2. Model Bias and Fairness: Biases inherent in historical data or algorithmic decisions can perpetuate disparities and discrimination in lending practices. Lenders must actively address issues of fairness and equity by monitoring model performance across demographic groups, mitigating bias through algorithmic interventions, and promoting diversity and inclusion in model development teams.

3. Model Interpretability and Trust: The opacity of complex machine learning models poses challenges in terms of interpretability and trust among stakeholders, including regulators, customers, and internal decision-makers. Lenders must prioritize model transparency, explainability, and accountability to foster trust and confidence in predictive modeling outcomes.

Financial institutions are focused on expanding their revenue streams, by selling various financial solutions, to their customers, a big chunk of this revenue comes from the credit line of business. The profitability of a financial institution is dependent on how well the credit business is yielding revenue, hence there is a huge focus on optimizing this process and an ardent desire to reduce the risk of loan defaulters. Adoption of AI/ML technologies are transforming credit process by significantly reducing the risk by predicting loan defaults. Data Science has paved the way for enabling predictive analytics. Several data science techniques such as Logistic regression, SVM, Neural Networks, Random Forest are discussed in this paper on how they enable increasing the accuracy of predicting loan defaulters. This paper deals with how a credit score is predicted to help financial institutions set the terms of loan disbursements to their customers. The focus of this paper is to present a loan prediction solution - Seven Seas to financial institutions. Several aspects of loan origination have been dealt with in this paper. A high-level process of loan application and an alternative credit scoring model using Machine Learning has been described. This paper also entails the overall market scope for such a solution and identifies several financial institutions that can embark on their transformation initiatives with such a disruptive technology. The extent of the existing market and its scope to embrace this technology is phenomenal not just in India but also globally.

LOANS are the major requirement of the modern world. By this only, Banks get a major part of the total profit. It is beneficial for students to manage their education and living expenses, and for people to buy any kind of luxury like houses, cars, etc.

But when it comes to deciding whether the applicant’s profile is relevant to be granted with loan or not. Banks have to look after many aspects.

Design and Implementation Constraints

Constraints in Analysis

♦ Constraints as Informal Text

♦ Constraints as Operational Restrictions

♦ Constraints Integrated in Existing Model Concepts

♦ Constraints as a Separate Concept

♦ Constraints Implied by the Model Structure

Constraints in Design

♦ Determination of the Involved Classes

♦ Determination of the Involved Objects

♦ Determination of the Involved Actions

♦ Determination of the Require Clauses

♦ Global actions and Constraint Realization

Constraints in Implementation

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

Other Nonfunctional Requirements

Performance Requirements

The application at this side controls and communicates with the following three main general components.

➢ embedded browser in charge of the navigation and accessing to the web service; ➢

Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.
5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
6. Systems with different requirements for safety levels must be separated.

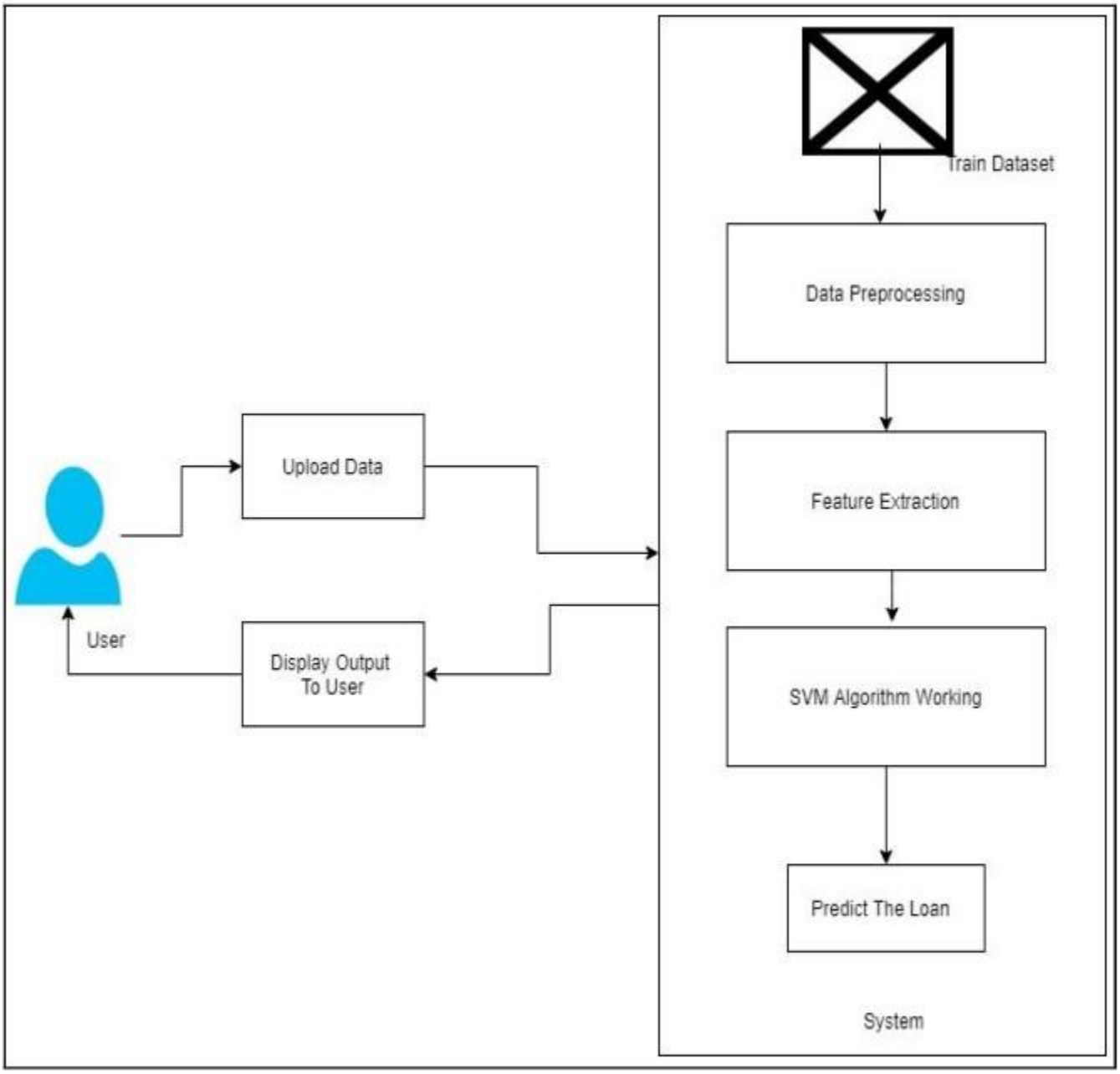
38

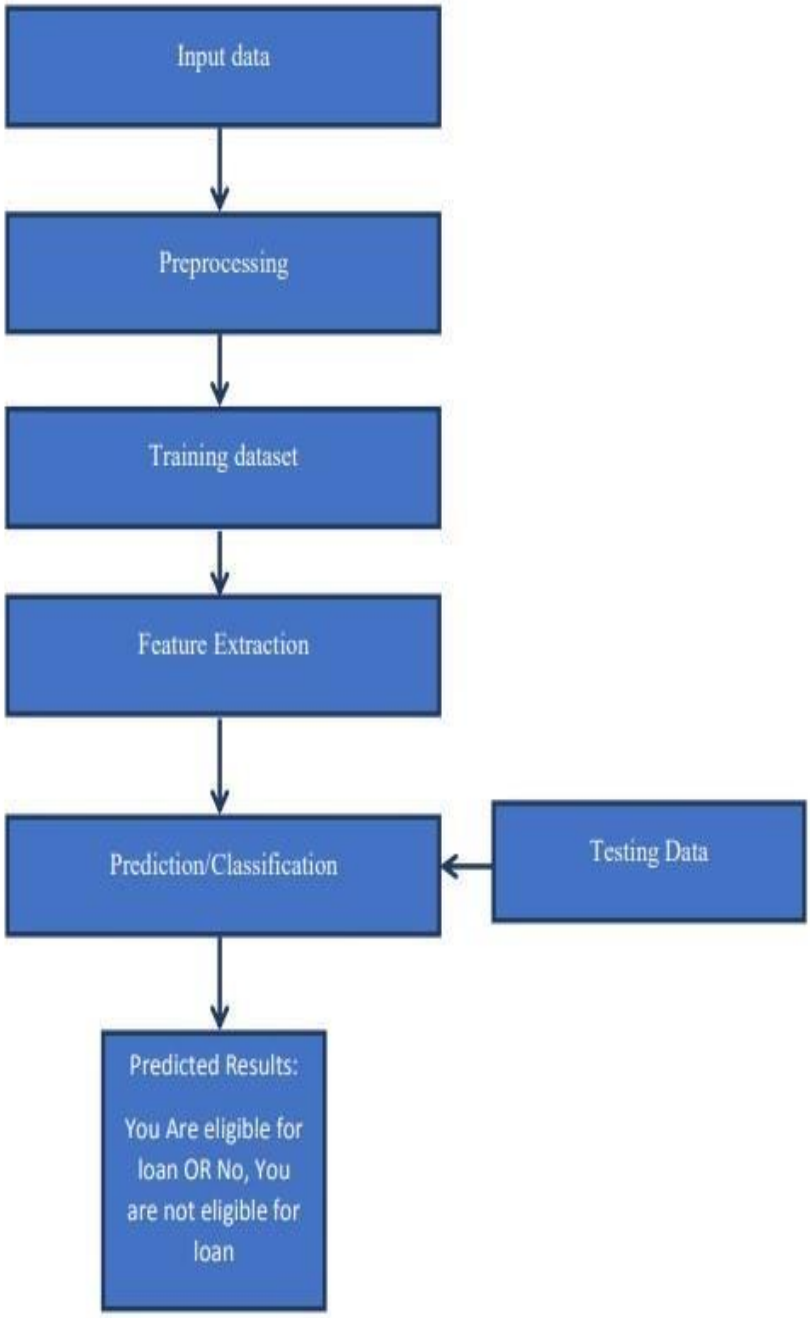
7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

Technology has boosted the existence of human kind 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

Loan approval is a very important process for banking organizations. The systems approved or reject 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. In recent years many researchers worked on loan approval prediction systems. Machine Learning (ML) techniques are very useful in predicting outcomes for large amount of data. In this paper different machine learning algorithms are applied to predict the loan approval of customers..In this paper, various machine learning algorithms that have been used in past are discussed and their accuracy is evaluated. The main focus of this paper is to determine whether the loan given to a particular person or an organization shall be approved or not.

Prediction of modernized loan approval system based on machine learning approach is a loan approval system from where we can know whether the loan will pass or not. In this system, we take some data from the user like his monthly income, marriage status, loan amount, loan duration, etc. Then the bank will decide according to its parameters whether the client will get the loan or not. So there is a classification system, in this system, a training set is employed to make the model and the classifier may classify the data items into their appropriate class. A test dataset is created that trains the data and gives the appropriate result that, is the client potential and can repay the loan. Prediction of a modernized loan approval system is incredibly helpful for banks and also the clients. This system checks the candidate on his priority basis. Customer can submit his application directly to the bank so the bank will do the whole process, no third party or stockholder will interfere in it. And finally, the bank will decide that the candidate is deserving or not on its priority basis. The only object of this research paper is that the deserving candidate gets straight forward and quick results.


Train Dataset
Data Preprocessing
Feature Extraction
SVM Algorithm Working
Predict The Loan
System
Upload Data
Display Output To User
User


Input data
Preprocessing
Training dataset
Feature Extraction
Prediction/Classification
Testing Data
Predicted Results:
You Are eligible for loan OR No, You are not eligible for loan

Chapter 4: Result and Output

Code

Type: Binary Classification Loan approval prediction is classic problem to learn and apply lots of data analysis techniques to create best Classification model.

Given with the dataset consisting of details of applicants for loan and status whether the loan application is approved or not. Basis on the a binary classification model is to be created with maximum accuracy.

In [1]:

```
#Basic and most important libraries
import pandas as pd , numpy as np
from sklearn.utils import resample
from sklearn.preprocessing import StandardScaler , MinMaxScaler
from collections import Counter from scipy import stats import
matplotlib.pyplot as plt import seaborn as sns import
plotly.express as px import plotly.figure_factory as ff import plotly

#Classifiers
from sklearn.ensemble import AdaBoostClassifier , GradientBoostingClassifier , VotingClassifier , RandomForestCla
ssifier
from sklearn.linear_model import LogisticRegression , RidgeClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import RepeatedStratifiedKFold from
sklearn.neighbors import KNeighborsClassifier from
sklearn.model_selection import GridSearchCV from sklearn.tree
import DecisionTreeClassifier  from sklearn.naive_bayes import
GaussianNB from xgboost import plot_importance from xgboost
import XGBClassifier
from sklearn.svm import SVC

#Model evaluation tools
from sklearn.metrics import classification_report , accuracy_score , confusion_matrix
from sklearn.metrics import accuracy_score,f1_score from sklearn.model_selection
import cross_val_score

#Data processing functions from
sklearn.preprocessing import StandardScaler from
sklearn.model_selection import train_test_split
from sklearn import model_selection
from sklearn.preprocessing import LabelEncoder le
= LabelEncoder()

import warnings
warnings.filterwarnings("ignore")
```

In [2]:

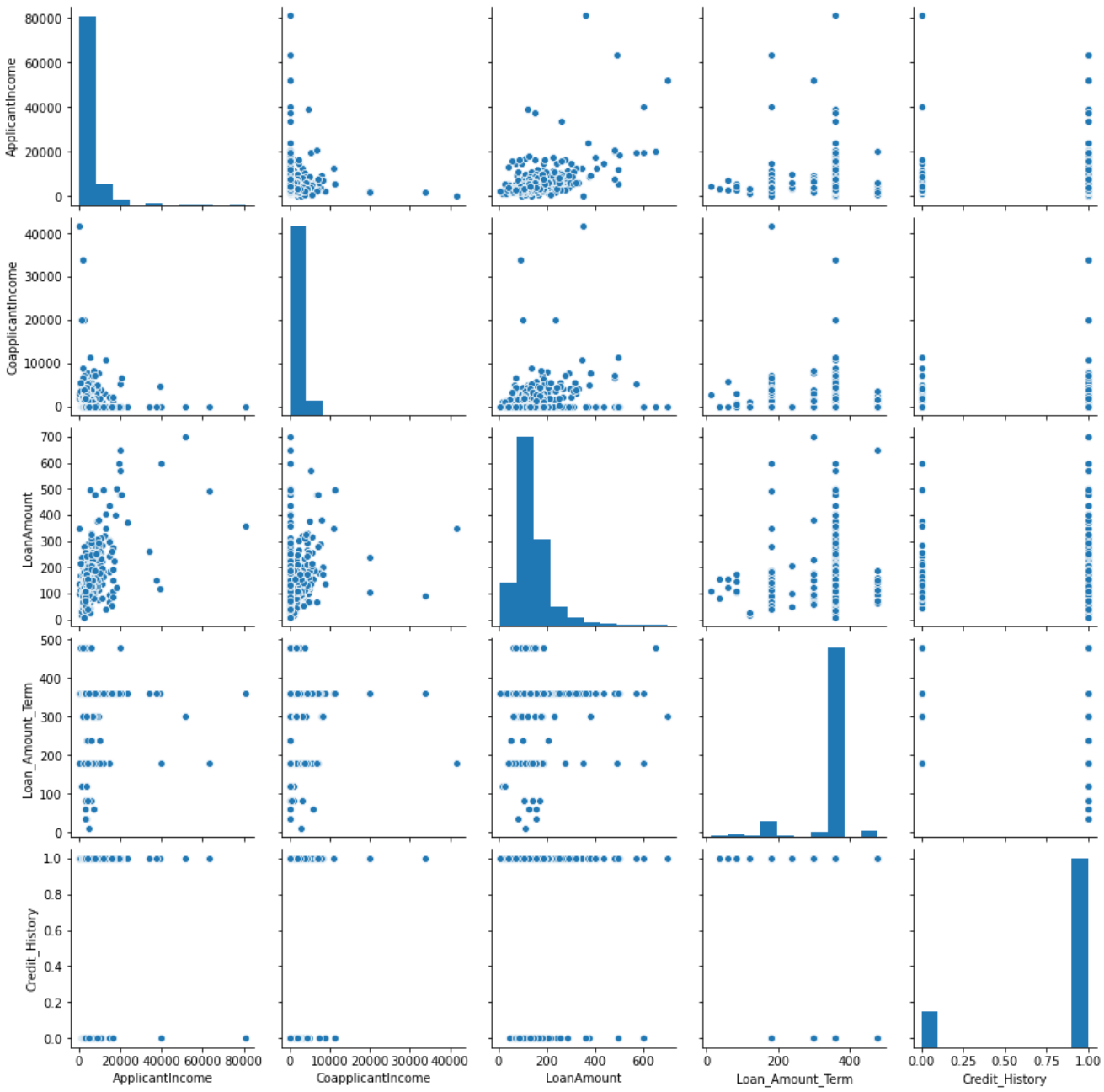
```
data = pd.read_csv(r"C:\Master\Learning\Analytics_Vidhya\Loan_Prediction-Hackathon\train.csv") data.head(5)
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [3]:

```
sns.pairplot(data) plt.show()
```



	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [5]:

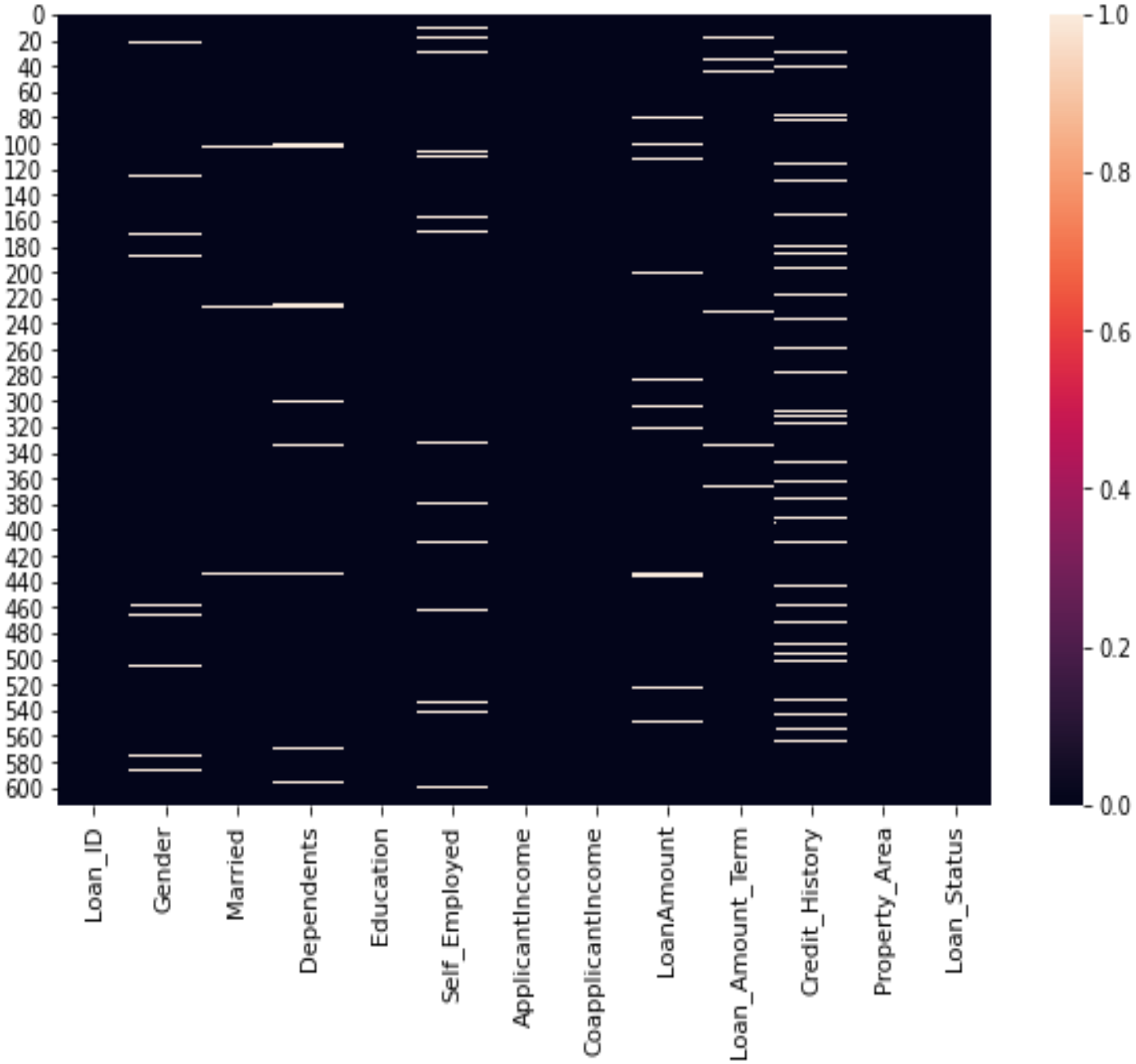
```
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613 Data
columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
0    Loan_ID           614 non-null    object
1    Gender            601 non-null    object
2    Married           611 non-null    object
3    Dependents        599 non-null    object
4    Education         614 non-null    object
5    Self_Employed     582 non-null    object
6    ApplicantIncome   614 non-null    int64
7    CoapplicantIncome 614 non-null    float64
8    LoanAmount        592 non-null    float64
9    Loan_Amount_Term  600 non-null    float64
10   Credit_History    564 non-null    float64
11   Property_Area     614 non-null    object   12  Loan_Status       614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
fig = px.scatter_matrix(data["ApplicantIncome"])
fig.update_layout(width=700,height=400) fig.show()
020k40k60k80k020k40k60k80k
ApplicantIncomeApplicantIncome
```

Seems need to work on data preperation

-Loan Amount column does is not fit in Normal Distribution
-Outliers in Applicant's Income and Co-applicant's income

```
data.isnull().sum() Loan_ID
0
Gender              13
```

```
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
plt.figure(figsize=(10,6))
sns.heatmap(data.isnull())
<AxesSubplot:>
```



Normal Distribution

Central limit theorem In simple language we can say that maximum amount of data / or maximum number of data points are near the Mean of the all data points.

To validate he normal distribution of the data:- Mean Mode Median are Equal.\n

We can gen identified the distribution of entire data with the help of Mean and Standard Deviation.

When the data is normally distributed maximum data is centralized near the mean value of the data.

To get understanding of distribtuion we can simply plot Distribution plot i.e. Simple Histogram.

Normally Distributed data represents a Bell Shaped curve.

Also Mean , Mode , Median on Normaly Distributed data are equal (Mean=Mode=Median)
One more method is to calculate mean which should be 0 or near to 0 and Standard deviation 1 or near 1.

Mean = sum(All Data Points)/count(Data Points)

Standard Deviation = Root of { sum [Square (each data point - mean of whole data)] }

In [9]:

```
#Checking if the non-categorical variables are Normally Distributed or Not. i.e. Checking outliers...

print("Data distribution analysis:->-------------------------------------\n")
print("\nMean:->\n") print("ApplicantIncome:
",np.mean(data["ApplicantIncome"])) print("CoapplicantIncome:
",np.mean(data["CoapplicantIncome"])) print("LoanAmount:
",np.mean(data["LoanAmount"]))

print("\nMode:->\n") print("ApplicantIncome:
",stats.mode(data["ApplicantIncome"])[0]) print("CoapplicantIncome:
",stats.mode(data["CoapplicantIncome"])[0]) print("LoanAmount:
",stats.mode(data["LoanAmount"])[0])

print("\nMedian:->\n") print("ApplicantIncome:
",np.median(data["ApplicantIncome"])) print("CoapplicantIncome:
",np.median(data["CoapplicantIncome"])) print("LoanAmount:
",np.median(data["LoanAmount"]))

print("\nStandard Deviation:->\n") print("ApplicantIncome:
",np.std(data["ApplicantIncome"])) print("CoapplicantIncome:
",np.std(data["CoapplicantIncome"])) print("LoanAmount:
",np.std(data["LoanAmount"]))

fig = px.histogram(data["ApplicantIncome"],x ="ApplicantIncome" ,y = "ApplicantIncome" )
fig.update_layout(title="ApplicantIncome") fig.show()

fig = px.histogram(data["CoapplicantIncome"],x ="CoapplicantIncome" ,y = "CoapplicantIncome" )
fig.update_layout(title="CoapplicantIncome") fig.show()

fig = px.histogram(data["LoanAmount"],x ="LoanAmount" ,y = "LoanAmount" )
fig.update_layout(title="LoanAmount") fig.show()
Data distribution analysis:->-------------------------------------


Mean:->

ApplicantIncome:  5403.459283387622
CoapplicantIncome:  1621.245798027101
LoanAmount:  146.41216216216216
```

Mode:->

ApplicantIncome: [2500]
CoapplicantIncome: [0.]
LoanAmount: [120.]

Median:->

ApplicantIncome: 3812.5
CoapplicantIncome: 1188.5
LoanAmount: nan

Standard Deviation:->

ApplicantIncome: 6104.064856533888
CoapplicantIncome: 2923.8644597700627
LoanAmount: 85.51500809120331
010k20k30k40k50k60k70k80k0100k200k300k400k500k
ApplicantIncomeApplicantIncomesum of ApplicantIncome
05k10k15k20k25k30k35k40k020k40k60k80k100k120k140k160k CoapplicantIncomeCoapplicantIncomesum
of CoapplicantIncome
0100200300400500600700002k4k6k8k10k12k14k LoanAmountLoanAmountsum
of LoanAmount
From above graphs found these variables are not normaly distributed.

Foud right-skewed distribution in these three variabels.

In [10]:

```
plt.figure(figsize=(10,5)) fig = px.bar(data,x=data["Gender"]) fig.show()

fig = px.bar(data,x=data["Married"]) fig.show()

fig = px.bar(data,x=data["Education"],color="Education") fig.show()

fig = px.bar(data,x=data["Self_Employed"]) fig.show()

fig = px.bar(data,x=data["Dependents"]) fig.show()

fig = px.bar(data,x=data["Property_Area"]) fig.show()

fig = px.bar(data,x=data["Loan_Status"],color="Loan_Status") fig.show()
```

MaleFemale0100200300400500
Gendercount
NoYes0501001502002503003504000
Marriedcount
GraduateNot Graduate0100200300400500
EducationGraduateNot GraduateEducationcount
NoYes0100200300400500

Self_Employedcount
−0.500.511.522.5050100150200250300350
Dependentscount
UrbanRuralSemiurban050100150200
Property_Areacount
YN050100150200250300350400
Loan_StatusYNLoan_Statuscount <Figure
size 720x360 with 0 Axes>
Prepare data for model training i.e. removing ouliers , filling null values , removing skewness

In [11]:

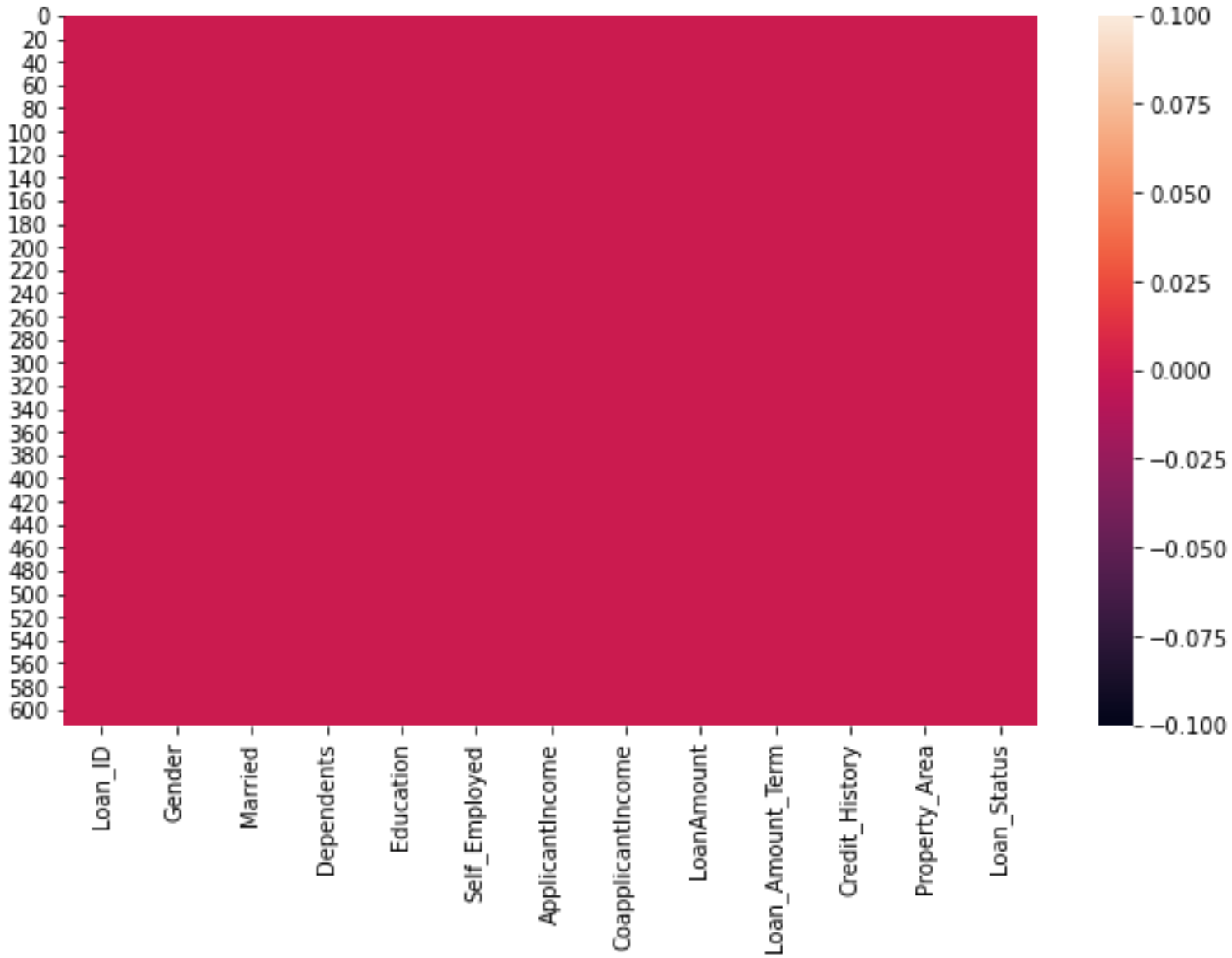
```
print(data["Gender"].value_counts()) print(data["Married"].value_counts())
print(data["Self_Employed"].value_counts()) print(data["Dependents"].value_counts())
print(data["Credit_History"].value_counts()) print(data["Loan_Amount_Term"].value_counts())
```

Male 489
Female 112
Name: Gender, dtype: int64
Yes 398
No 213
Name: Married, dtype: int64
No 500
Yes 82
Name: Self_Employed, dtype: int64
0 345
1 102
2 101
3+ 51
Name: Dependents, dtype: int64
1.0 475
0.0 89
Name: Credit_History, dtype: int64
360.0 512
180.0 44
480.0 15
300.0 13
84.0 4
240.0 4
120.0 3
36.0 2 60.0
2
12.0 1
Name: Loan_Amount_Term, dtype: int64
->Taking mode of values in a column will be best way to fill null values. ->Not mean because values are not ordinal but are categorical.

In [12]:

```
#Filling all Nan values with mode of respective variable
data["Gender"].fillna(data["Gender"].mode()[0],inplace=True)
data["Married"].fillna(data["Married"].mode()[0],inplace=True)
```

```
data["Self_Employed"].fillna(data["Self_Employed"].mode()[0],inplace=True)
data["Loan_Amount_Term"].fillna(data["Loan_Amount_Term"].mode()[0],inplace=True)
data["Dependents"].fillna(data["Dependents"].mode()[0],inplace=True)
data["Credit_History"].fillna(data["Credit_History"].mode()[0],inplace=True)

#All values of "Dependents" columns were of "str" form now converting to "int" form.
data["Dependents"] = data["Dependents"].replace('3+',int(3)) data["Dependents"] =
data["Dependents"].replace('1',int(1)) data["Dependents"] =
data["Dependents"].replace('2',int(2)) data["Dependents"] =
data["Dependents"].replace('0',int(0))

data["LoanAmount"].fillna(data["LoanAmount"].median(),inplace=True)

print(data.isnull().sum())

#Heat map for null values
plt.figure(figsize=(10,6)) sns.heatmap(data.isnull())
Loan_ID            0
Gender             0
Married            0
Dependents         0
Education          0
Self_Employed      0
ApplicantIncome    0
CoapplicantIncome  0
LoanAmount         0
Loan_Amount_Term   0
Credit_History     0
Property_Area      0
Loan_Status        0 dtype:
int64
                                                                                    Out[12]:
<AxesSubplot:>
```



In [13]:

#Treating outliers and Converting data to Normal Distribution
#Before removing outlier

print("\n**n**Mean:->\n**n**") print("ApplicantIncome:
",np.mean(data["ApplicantIncome"])) print("CoapplicantIncome:
",np.mean(data["CoapplicantIncome"])) print("LoanAmount:
",np.mean(data["LoanAmount"]))

print("\n**n**Mode:->\n**n**") print("ApplicantIncome:
",stats.mode(data["ApplicantIncome"])[0]) print("CoapplicantIncome:
",stats.mode(data["CoapplicantIncome"])[0]) print("LoanAmount:
",stats.mode(data["LoanAmount"])[0])

print("\n**n**Median:->\n**n**")

```
print("ApplicantIncome: ",np.median(data["ApplicantIncome"]))
print("CoapplicantIncome: ",np.median(data["CoapplicantIncome"])) print("LoanAmount:
",np.median(data["LoanAmount"]))

print("\nStandard Deviation:->\n") print("ApplicantIncome:
",np.std(data["ApplicantIncome"])) print("CoapplicantIncome:
",np.std(data["CoapplicantIncome"])) print("LoanAmount:
",np.std(data["LoanAmount"]))

fig = px.histogram(data["ApplicantIncome"],x ="ApplicantIncome" ,y = "ApplicantIncome" )
fig.update_layout(title="ApplicantIncome") fig.show()

fig = px.histogram(data["CoapplicantIncome"],x ="CoapplicantIncome" ,y = "CoapplicantIncome" )
fig.update_layout(title="CoapplicantIncome") fig.show()

fig = px.histogram(data["LoanAmount"],x ="LoanAmount" ,y = "LoanAmount" )
fig.update_layout(title="LoanAmount") fig.show()

##########################################################################################################
######
#Getting log value :->

data["ApplicantIncome"] = np.log(data["ApplicantIncome"])
#As "CoapplicantIncome" columns has some "0" values we will get log values except "0"
data["CoapplicantIncome"] = [np.log(i) if i!=0 else 0 for i in data["CoapplicantIncome"]] data["LoanAmount"]
= np.log(data["LoanAmount"])
##########################################################################################################
######

print("----------------------------After converting to Normal Distributed data----------------------")

print("\nMean:->\n")
print("ApplicantIncome: ",np.mean(data["ApplicantIncome"]))
print("CoapplicantIncome: ",np.mean(data["CoapplicantIncome"])) print("LoanAmount:
",np.mean(data["LoanAmount"]))

print("\nMode:->\n") print("ApplicantIncome:
",stats.mode(data["ApplicantIncome"])[0]) print("CoapplicantIncome:
",stats.mode(data["CoapplicantIncome"])[0]) print("LoanAmount:
",stats.mode(data["LoanAmount"])[0])

print("\nMedian:->\n")
print("ApplicantIncome: ",np.median(data["ApplicantIncome"]))
print("CoapplicantIncome: ",np.median(data["CoapplicantIncome"])) print("LoanAmount:
",np.median(data["LoanAmount"]))
```

```
print("\nStandard Deviation:->\n") print("ApplicantIncome:
",np.std(data["ApplicantIncome"])) print("CoapplicantIncome:
",np.std(data["CoapplicantIncome"])) print("LoanAmount:
",np.std(data["LoanAmount"]))

plt.figure(figsize=(10,4))
fig = px.histogram(data["ApplicantIncome"],x ="ApplicantIncome" ,y = "ApplicantIncome" )
fig.update_layout(title="ApplicantIncome") fig.show()

fig = px.histogram(data["CoapplicantIncome"],x ="CoapplicantIncome" ,y = "CoapplicantIncome" )
fig.update_layout(title="CoapplicantIncome") fig.show()

fig = px.histogram(data["LoanAmount"],x ="LoanAmount" ,y = "LoanAmount" )
fig.update_layout(title="LoanAmount") fig.show()
```

Mean:->

ApplicantIncome: 5403.459283387622
CoapplicantIncome: 1621.245798027101
LoanAmount: 145.75244299674267

Mode:->

ApplicantIncome: [2500]
CoapplicantIncome: [0.]
LoanAmount: [128.]

Median:->

ApplicantIncome: 3812.5
CoapplicantIncome: 1188.5
LoanAmount: 128.0

Standard Deviation:->

ApplicantIncome: 6104.064856533888
CoapplicantIncome: 2923.8644597700627
LoanAmount: 84.03871423798938
010k20k30k40k50k60k70k80k0100k200k300k400k500k
ApplicantIncomeApplicantIncomesum of ApplicantIncome
05k10k15k20k25k30k35k40k020k40k60k80k100k120k140k160k CoapplicantIncomeCoapplicantIncomesum of CoapplicantIncome
0100200300400500600700002k4k6k8k10k12k14k16k18k
LoanAmountLoanAmountsum of LoanAmount
----------After converting to Normal Distributed data----------

Mean:->

ApplicantIncome: 8.341213093227005
CoapplicantIncome: 4.289733227820405
LoanAmount: 4.8572501948110895

Mode:->

ApplicantIncome: [7.82404601]
CoapplicantIncome: [0.]
LoanAmount: [4.85203026]

Median:->

ApplicantIncome: 8.246040412315828
CoapplicantIncome: 7.080283635438671
LoanAmount: 4.852030263919617

Standard Deviation:->

ApplicantIncome: 0.6447375297521025
CoapplicantIncome: 3.8725225987499146
LoanAmount: 0.49559166063018056
5678910110501001502002503003504004 50
ApplicantIncomeApplicantIncomesum of ApplicantIncome
3456789101102004006008001000120014 00
CoapplicantIncomeCoapplicantIncomesum of CoapplicantIncome
2.533.544.555.566.5050100150200250300350400450
LoanAmountLoanAmountsum of LoanAmount
<Figure size 720x288 with 0 Axes>
Now we can see that Bell Curve for all three variables and data is normally distributed now.

In [14]:

data.head(5)

data["Gender"] = le.fit_transform(data["Gender"]) data["Married"] = le.fit_transform(data["Married"]) data["Education"] = le.fit_transform(data["Education"]) data["Self_Employed"] = le.fit_transform(data["Self_Employed"]) data["Property_Area"] = le.fit_transform(data["Property_Area"]) data["Loan_Status"] = le.fit_transform(data["Loan_Status"])

#data = pd.get_dummies(data)
data.head(5)

Out[15]:

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status

0	LP001002	1	0	0	0	0	8.674026	0.000000	4.852030	360.0	1.0	2	1
1	LP001003	1	1	1	0	0	8.430109	7.318540	4.852030	360.0	1.0	0	0
2	LP001005	1	1	0	0	1	8.006368	0.000000	4.189655	360.0	1.0	2	1
3	LP001006	1	1	0	1	0	7.856707	7.765569	4.787492	360.0	1.0	2	1
4	LP001008	1	0	0	0	0	8.699515	0.000000	4.948760	360.0	1.0	2	1

In [ ]:

In [ ]:

Feature Importance

In order to create best predictive model we need to best understand the available data and get most information from the data.

In multivariate data it is important to understand the iortance of varialbes and how much they are contributing towards the target variable. Such that we can remove unnecessary variables to increase model performance.

Many times dataset consists of exta columns which do not identically serve information to classify the data. This leads in Wrong Assumption of model while training.

To understand the importance of the data we are going to use Machine Learning classifiers and then will plot bar graph based on importance.
Also XGBoost has built-in Feature Importance Plotting tool which we are going to use.

Using more than one classifier will increase the confidence on our assumption of which variables to keep and which to remove.

In [16]: *#Dividing data into Input X variables and Target Y variable*

```
X = data.drop(["Loan_Status","Loan_ID"],axis=1) y
= data["Loan_Status"]
```

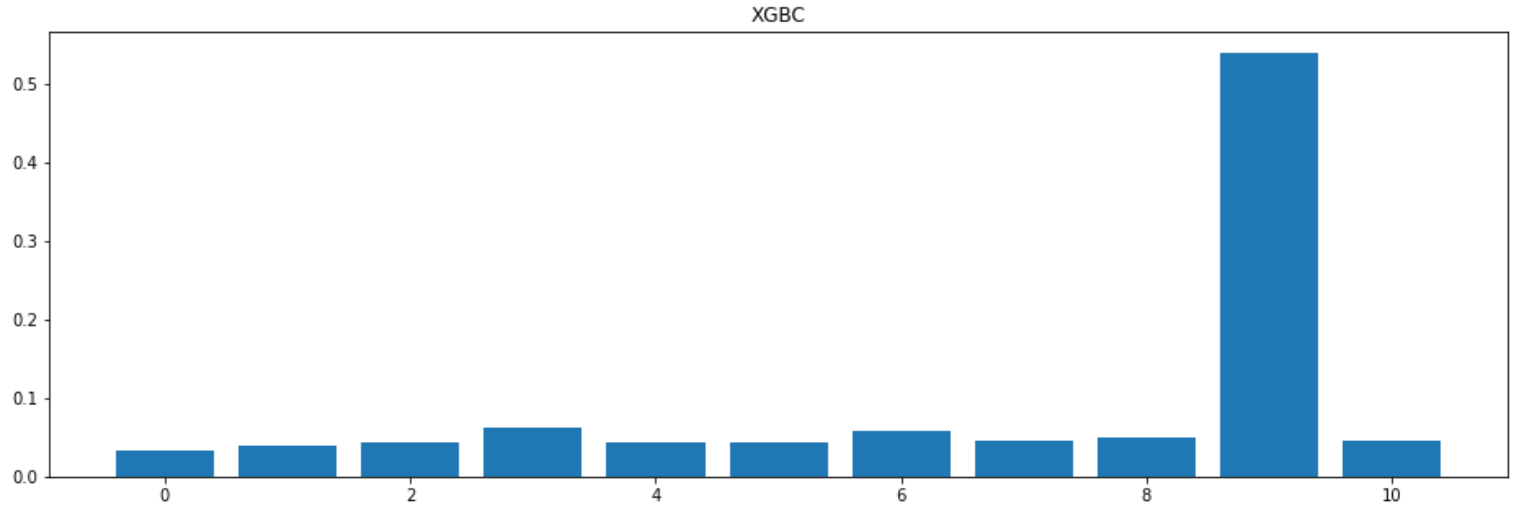
In [17]: print("Feature importance by XGBoost:->\n")

```
XGBR = XGBClassifier()
XGBR.fit(X,y)
features = XGBR.feature_importances_
Columns = list(X.columns) for i,j in
enumerate(features):
print(Columns[i],"->",j)
plt.figure(figsize=(16,5))
plt.title(label="XGBC")
plt.bar([x for x in range(len(features))],features) plt.show()

plot_importance(XGBR)

print("Feature importance by Random Forest:->\n")
RF = RandomForestClassifier() RF.fit(X,y)
features = RF.feature_importances_
Columns = list(X.columns) for i,j in
enumerate(features):
print(Columns[i],"->",j)
plt.figure(figsize=(16,5))
plt.title(label="RF")
plt.bar([x for x in range(len(features))],features) plt.show()

print("Feature importance by Decision Tree:->\n")
DT = DecisionTreeClassifier()
DT.fit(X,y)
features = DT.feature_importances_
Columns = list(X.columns) for i,j in
enumerate(features):
print(Columns[i],"->",j)
plt.figure(figsize=(16,5))
plt.title(label="DT")
plt.bar([x for x in range(len(features))],features) plt.show()

print("Feature importance by Suppoprt Vector Machine:->\n")
SVM = SVC(kernel="linear")
SVM.fit(X,y) features =
SVM.coef_[0] Columns =
list(X.columns) for i,j in
enumerate(features):
print(Columns[i],"->",j)
plt.figure(figsize=(16,5))
plt.bar([x for x in range(len(features))],features) plt.show()

print("Feature importance by Logistic Regression:->\n")
```

```
LOGC = LogisticRegression()
LOGC.fit(X,y)  features  =
LOGC.coef_[0]  Columns  =
list(X.columns)  for  i,j  in
enumerate(features):
print(Columns[i],"->",j)
plt.figure(figsize=(16,5))
plt.title(label="LOGC")
plt.bar([x for x in range(len(features))],features) plt.show()
```

Feature importance by XGBoost:->

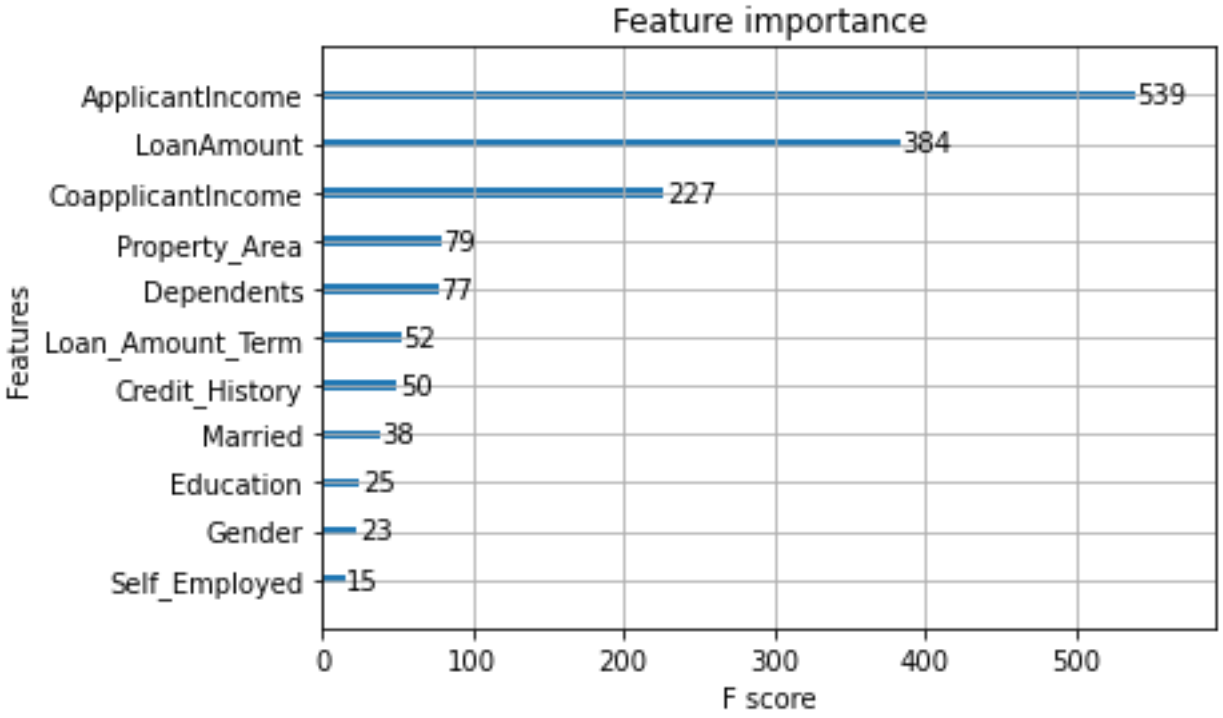
Gender -> 0.032498196
Married -> 0.038461618
Dependents -> 0.042435512
Education -> 0.06297734
Self_Employed -> 0.04353367
ApplicantIncome -> 0.04360314
CoapplicantIncome -> 0.057352304
LoanAmount -> 0.04579362
Loan_Amount_Term -> 0.049817037
Credit_History -> 0.53902644
Property_Area -> 0.044501156

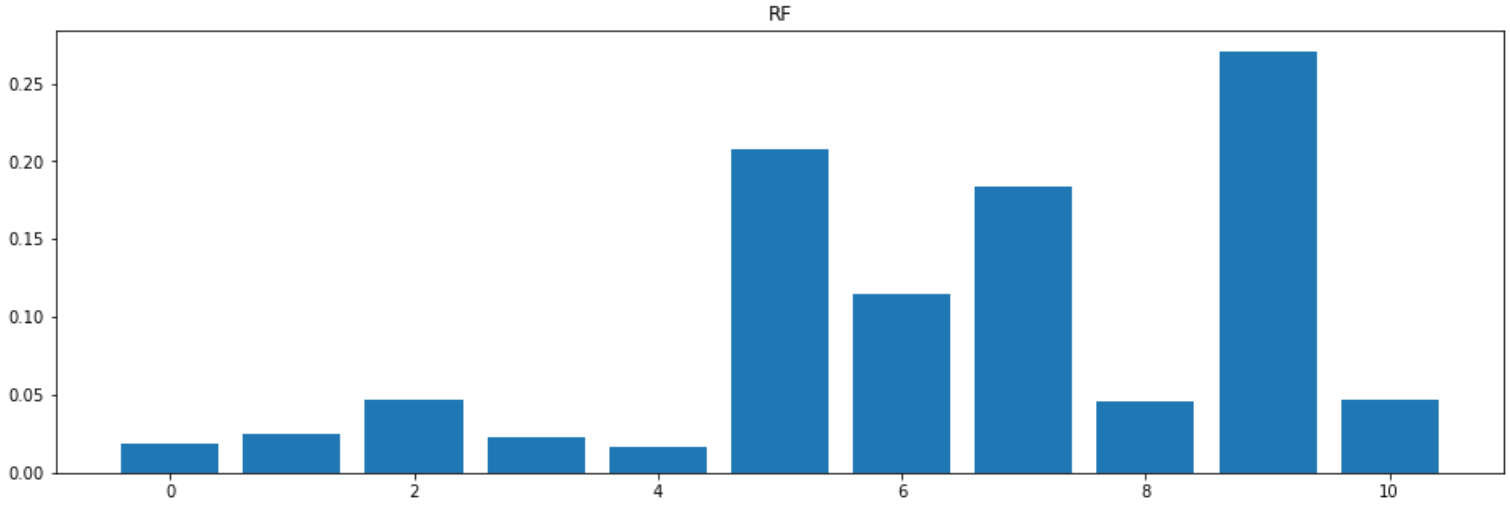


Feature importance by Random Forest:->

Gender -> 0.018900487204807852
Married -> 0.02462931105690738
Dependents -> 0.047052466747390456
Education -> 0.022639340128868403
Self_Employed -> 0.01657632874630836
ApplicantIncome -> 0.20779381878351624
CoapplicantIncome -> 0.1145844478048256
LoanAmount -> 0.1839704267368444
Loan_Amount_Term -> 0.046140336171376174

Credit_History -> 0.27057938232022927
Property_Area -> 0.047133654298925964

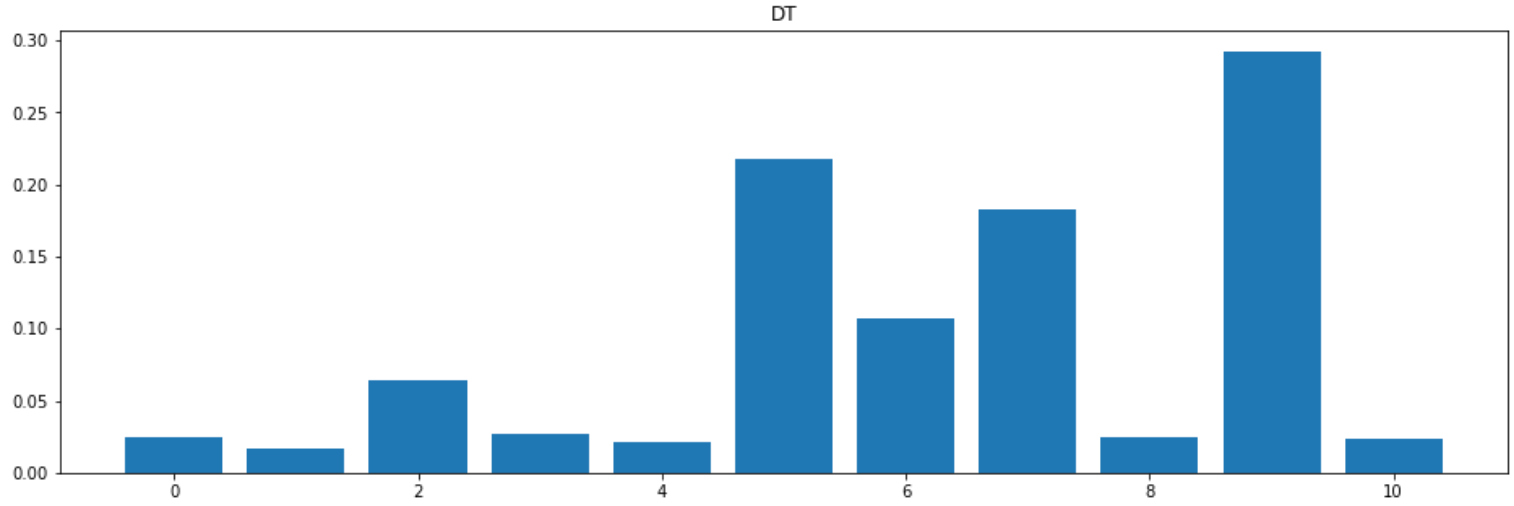




Feature importance by Decision Tree:->

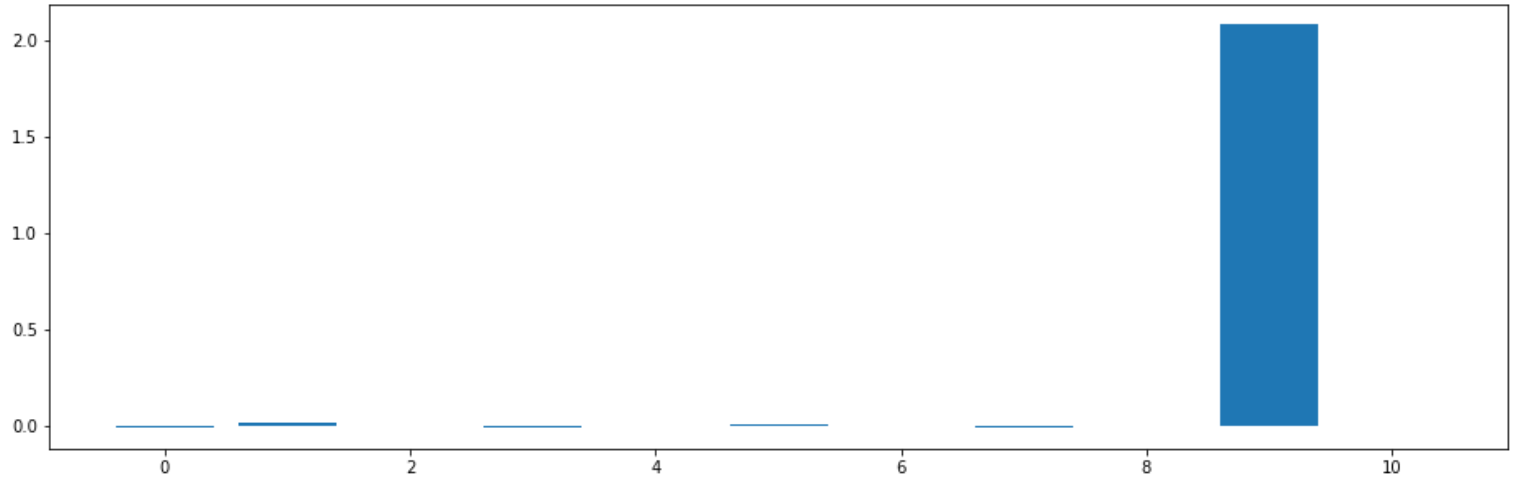
Gender -> 0.024418005090398452
Married -> 0.016326588769442065
Dependents -> 0.06400168357447024
Education -> 0.026528466142634998
Self_Employed -> 0.020983027101584565
ApplicantIncome -> 0.21759903699076355
CoapplicantIncome -> 0.10665253394698983
LoanAmount -> 0.18273808075459913
Loan_Amount_Term -> 0.024630673969621018
Credit_History -> 0.2922008668920113

Property_Area -> 0.023921036767484947



Feature importance by Suppoprt Vector Machine:->

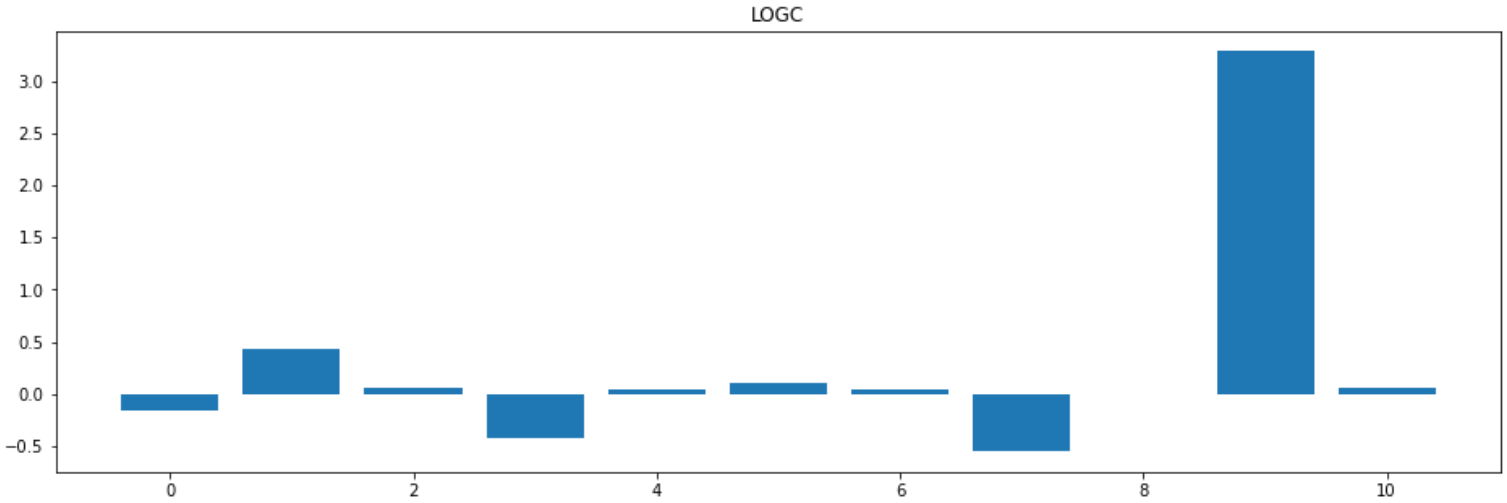
Gender -> -0.011153748611395287
Married -> 0.016433621802949716
Dependents -> -0.0003948864299205823
Education -> -0.007897250281862611
Self_Employed -> -0.0045186612877454735
ApplicantIncome -> 0.009509713938893327
CoapplicantIncome -> 0.0009391121595605512
LoanAmount -> -0.012713675348784648
Loan_Amount_Term -> 8.910680668350324e-05
Credit_History -> 2.0812104159306477
Property_Area -> -0.0006557085562250223



Feature importance by Logistic Regression:->

Gender -> -0.1615139600532564
Married -> 0.4341098090301747
Dependents -> 0.05871757548193793
Education -> -0.415446117064946
Self_Employed -> 0.04313150698288537

ApplicantIncome -> 0.1020827246750018
CoapplicantIncome -> 0.04475513414904771
LoanAmount -> -0.5526893355733061
Loan_Amount_Term -> -0.0012174092655106736
Credit_History -> 3.28383317084153
Property_Area -> 0.05809243644023144



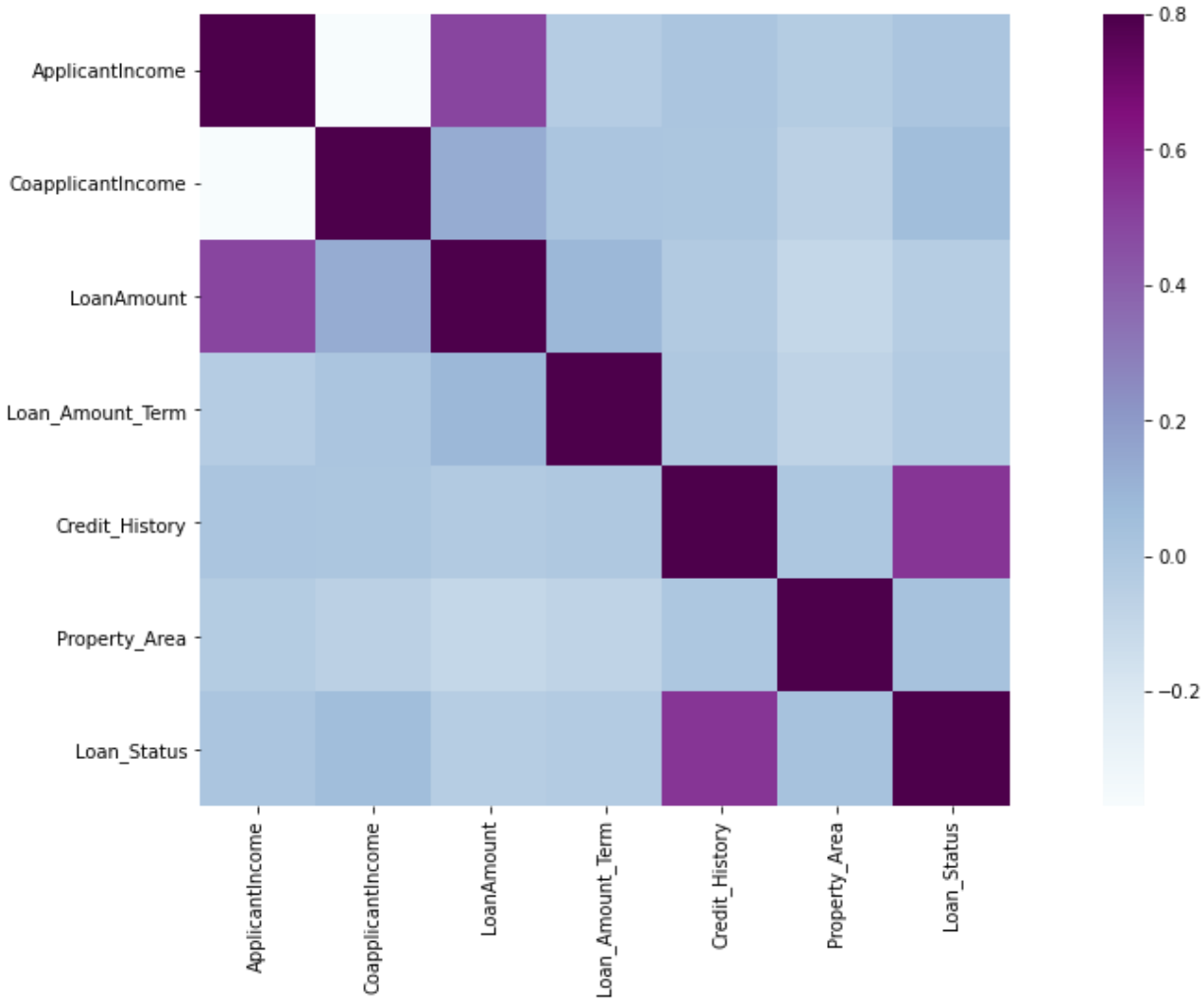
From feature importance => Credit History , ApplicantIncome , CoapplicantIncome, LoanAmount are the most important features

Is data Balanced ?

In [18]:

```
#Heat map of dataset with relative importance
matrix = data.drop(["Gender","Married","Dependents","Education","Self_Employed"],axis=1).corr()
#f , ax = plt.subplots(figsize=(18,6)) plt.figure(figsize=(18,8))
sns.heatmap(matrix,vmax=0.8,square=True,cmap="BuPu")
```

Out[18]:



<AxesSubplot:>

It seems Application income and Loan Amount is correlated , also Coapplication income correlated with Loan Aount then Credit history is corrleated with Loan Status

In [19]:

A = list(data.Loan_Status).count(1) B
= list(data.Loan_Status).count(0)
print("Count of 1<Approved>: ",A,"**n**Count of 0<Rejected>: ",B)

fig = px.bar((A,B),x=["Approved","Rejected"],y=[A,B],color=[A,B]) fig.show()
Count of 1<Approved>: 422
Count of 0<Rejected>: 192
ApprovedRejected0501001502002503003504002002503003504002colorxy
It seems that data is highly Imbalanced.

When the target classes does not have equal count then the data is considered as imbalanced data.

From above graph it seems that dataset contains more records with Approved Loan_Status than Rejected Loan_Status. 422 over 192

If data would have maximum of 20-30 records difference that time this imabalnced would be ignorable.

Which will lead to make wrong assumptions by model and also model will be biased after training. We will overcome this issue by balancing the data.

To overcome this problem we will balance the data using Resampling technique with Upsample and Downsample.

In [20]:
```
#To keep original data as it is to use the same for later.
new_data = data.copy()

#Getting seperated data with 1 and 0 status.
df_majority = new_data[new_data.Loan_Status==1] df_minority
= new_data[new_data.Loan_Status==0]

#Here we are downsampling the Majority Class Data Points.
#i.e. We will get equal amount of datapoint as Minority class from Majority class

df_manjority_downsampled = resample(df_majority,replace=False,n_samples=192,random_state=123)
df_downsampled = pd.concat([df_manjority_downsampled,df_minority]) print("Downsampled data:-
>\n",df_downsampled.Loan_Status.value_counts())

#Here we are upsampling the Minority Class Data Points.
#i.e. We will get equal amount of datapoint as Majority class from Minority class
df_monority_upsampled = resample(df_minority,replace=True,n_samples=422,random_state=123)
df_upsampled = pd.concat([df_majority,df_monority_upsampled]) print("Upsampled data:-
>\n",df_upsampled.Loan_Status.value_counts()) Downsampled data:->
 1    192
0    192
Name: Loan_Status, dtype: int64
Upsampled data:->
1    422
0    422
Name: Loan_Status, dtype: int64
```

In [ ]:

Data Standardization / Normalization

Data normalization is required when the vriable values are in very distinct range.

For Ex. Suppose we have 2 columns "Age" and "Income"

Where value range of "Age" lying in 0-100 Approx. and value range of "Income" lying in 20,000 to 100,000

At this time model will perform poorly on testig data as all input values are not in same value range.

So not every time but whenever we get such type of data we need to normalized it i.e. Rescale it.

Widely used scaling tools are Min-Max Scaler and Standard-Scaler

Data Normalization is done by Min-Max Scaler which scales all th values between 0 to 1 range.

Data standardization is done by Standard-Scaler which scales the data so that Mean of observed data is 0 and Standard Deviation is 1.

As our data is not much normally distributed we will choose Standardization using Standard-Scaler aiming that it will reduce more skewness and contribute in accuracy gain.

In [ ]:

Experimental Modeling

In order to gain maximum posible accuracy one needs to conduct much emor experiments.

We will pass data on by one with different state i.e.

-Only Scaled data

-Scaled + Down Sampled Data

-Scaled + Up Sampled Data

-Scaled + Up Sampled Data + Selected feature with respective importance.

In [21]:

```
#Experiment 1: Only Scaled data with all variables

#X = new_data.drop(["Loan_ID","Gender","Married","Education","Self_Employed","Loan_Amount_Term","Loan_S
tatus",'Property_Area'],axis=1)
X = new_data.drop(["Loan_Status","Loan_ID"],axis=1) y
= new_data["Loan_Status"]
counter = Counter(y)
print("Counter: ",counter)

X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.25,random_state=0)

#Scaling data here:---------->

StSc = StandardScaler()
X_train  = StSc.fit_transform(X_train)
X_test  = StSc.fit_transform(X_test)

#Check mean is 0 and Standard deviation is 1
print("After Standardization\nMean ",np.mean(X_train),"Standard Deviation ",np.std(X_train),"\n")

#Voting ensemble mathod. Combining all tree based algorithms.
models = [] models.append(("XGB",XGBClassifier()))
models.append(("RF",RandomForestClassifier()))
models.append(("DT",DecisionTreeClassifier()))
models.append(("ADB",AdaBoostClassifier()))
models.append(("GB",GradientBoostingClassifier()))
```

```
ensemble = VotingClassifier(estimators=models)
ensemble.fit(X_train,y_train) y_pred =
ensemble.predict(X_test)
print(classification_report(y_pred,y_test))
print("Voting Ensemble:>",accuracy_score(y_pred,y_test))



SVM = SVC(kernel="linear",class_weight="balanced",probability=True)
SVM.fit(X_train,y_train) y_pred =
SVM.predict(X_test)
print(classification_report(y_pred,y_test))
print("SVM:>",accuracy_score(y_pred,y_test))


XGBC = XGBClassifier(learning_rate =0.1,n_estimators=10000,max_depth=4,min_child_weight=6,gamma=0,subsa
mple=0.6,colsample_bytree=0.8,
 reg_alpha=0.005, objective= 'binary:logistic', nthread=2, scale_pos_weight=1, seed=27)
XGBC.fit(X_train,y_train) y_pred = XGBC.predict(X_test)
print(classification_report(y_pred,y_test))
print("XGBoost:>",accuracy_score(y_pred,y_test))

Model1 = RandomForestClassifier(n_estimators=1000,random_state=0,n_jobs=1000,max_depth=70,bootstrap=True)
Model1.fit(X_train,y_train) y_pred = Model1.predict(X_test) print(classification_report(y_pred,y_test))
print("RandomForestClassifier:>",accuracy_score(y_pred,y_test))


Model2 = GradientBoostingClassifier()
Model2.fit(X_train,y_train) y_pred =
Model2.predict(X_test)
print(classification_report(y_pred,y_test))
print("GradientBoostingClassifier:>",accuracy_score(y_pred,y_test))


Model3 = DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=100,
max_features=1.0, max_leaf_nodes=10,  min_impurity_split=1e-07,
min_samples_leaf=1,  min_samples_split=2, min_weight_fraction_leaf=0.10,
presort=False, random_state=27, splitter='best') Model3.fit(X_train,y_train) y_pred =
Model3.predict(X_test) print(classification_report(y_pred,y_test))
print("DecisionTreeClassifier:>",accuracy_score(y_pred,y_test))


Model4 = AdaBoostClassifier()
Model4.fit(X_train,y_train) y_pred =
Model4.predict(X_test)
print(classification_report(y_pred,y_test))
print("AdaBoostClassifier:>",accuracy_score(y_pred,y_test))
```

```
Model5 = LinearDiscriminantAnalysis()
Model5.fit(X_train,y_train) y_pred =
Model5.predict(X_test)
print(classification_report(y_pred,y_test))
print("LinearDiscriminantAnalysis:>",accuracy_score(y_pred,y_test),"\n")


KNN = KNeighborsClassifier(leaf_size=1,p=2,n_neighbors=20)
KNN.fit(X_train,y_train) y_pred =
KNN.predict(X_test)
print(classification_report(y_pred,y_test))
print("KNeighborsClassifier:>",accuracy_score(y_pred,y_test))


Model7 = GaussianNB()
Model7.fit(X_train,y_train) y_pred =
Model7.predict(X_test)
print(classification_report(y_pred,y_test))
print("GaussianNB:>",accuracy_score(y_pred,y_test))


Model8 = LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
        intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
verbose=0, warm_start=False) Model8.fit(X_train,y_train) y_pred =
Model8.predict(X_test) print(classification_report(y_pred,y_test))
print("Logistic Regression:>",accuracy_score(y_pred,y_test))
Counter:  Counter({1: 422, 0: 192})
After Standardization
Mean  -1.2357264969740873e-16 Standard Deviation  1.0

              precision    recall  f1-score   support

0       0.47      0.74      0.57        27
1       0.94      0.82      0.87       127

    accuracy                           0.81       154
macro avg       0.70      0.78      0.72       154
weighted avg       0.85      0.81      0.82       154

Voting Ensemble:> 0.8051948051948052
              precision    recall  f1-score   support

0       0.44      0.90      0.59        21
1       0.98      0.82      0.89       133
```

```
   accuracy                         0.83       154
macro avg       0.71      0.86      0.74       154
weighted avg       0.91      0.83      0.85       154

SVM:> 0.8311688311688312
              precision    recall  f1-score   support

0       0.30      0.45      0.36        29
1       0.86      0.76      0.81       125

   accuracy                         0.70       154
macro avg       0.58      0.60      0.58       154
weighted avg       0.75      0.70      0.72       154
XGBoost:> 0.7012987012987013
              precision    recall  f1-score   support

0       0.44      0.73      0.55        26
1       0.94      0.81      0.87       128

   accuracy                         0.80       154
macro avg       0.69      0.77      0.71       154
weighted avg       0.85      0.80      0.82       154

RandomForestClassifier:> 0.7987012987012987
              precision    recall  f1-score   support

0       0.47      0.80      0.59        25
1       0.95      0.82      0.88       129

   accuracy                   0.82      154    macro
avg       0.71      0.81      0.74      154 weighted
avg       0.88      0.82      0.84      154

GradientBoostingClassifier:> 0.8181818181818182
              precision    recall  f1-score   support

0       0.44      0.90      0.59        21
1       0.98      0.82      0.89       133

   accuracy                   0.83      154    macro
avg       0.71      0.86      0.74      154 weighted
avg       0.91      0.83      0.85      154

DecisionTreeClassifier:> 0.8311688311688312
              precision    recall  f1-score   support
```

0 0.49 0.81 0.61 26
1 0.95 0.83 0.89 128

accuracy 0.82 154
macro avg 0.72 0.82 0.75 154
weighted avg 0.88 0.82 0.84 154

AdaBoostClassifier:> 0.8246753246753247
precision recall f1-score support

0 0.44 0.90 0.59 21
1 0.98 0.82 0.89 133
accuracy 0.83 154
macro avg 0.71 0.86 0.74 154
weighted avg 0.91 0.83 0.85 154

LinearDiscriminantAnalysis:> 0.8311688311688312

precision recall f1-score support

0 0.47 0.91 0.62 22
1 0.98 0.83 0.90 132

accuracy 0.84 154
macro avg 0.72 0.87 0.76 154
weighted avg 0.91 0.84 0.86 154

KNeighborsClassifier:> 0.8376623376623377
precision recall f1-score support

0 0.44 0.90 0.59 21
1 0.98 0.82 0.89 133

accuracy 0.83 154
macro avg 0.71 0.86 0.74 154
weighted avg 0.91 0.83 0.85 154

GaussianNB:> 0.8311688311688312
precision recall f1-score support

0 0.44 0.90 0.59 21
1 0.98 0.82 0.89 133

accuracy 0.83 154
macro avg 0.71 0.86 0.74 154
weighted avg 0.91 0.83 0.85 154

Logistic Regression:> 0.8311688311688312

In [22]:

```
#Experiment 2: Sclaed + Down Sampled Data

#X = df_downsampled.drop(["Loan_ID","Gender","Married","Education","Self_Employed","Loan_Amount_Term","
Loan_Status",'Property_Area'],axis=1)
X = df_downsampled.drop(["Loan_Status","Loan_ID"],axis=1) y
= df_downsampled.Loan_Status
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.25,random_state=0)

#Scaling data here:---------->

StSc = StandardScaler()
X_train  = StSc.fit_transform(X_train)
X_test  = StSc.fit_transform(X_test)

#Check mean is 0 and Standard deviation is 1
print("After Standardization\nMean ",np.mean(X_train),"Standard Deviation ",np.std(X_train),"\n")

#Voting ensemble mathod. Combining all tree based algorithms.
models = [] models.append(("XGB",XGBClassifier()))
models.append(("RF",RandomForestClassifier()))
models.append(("DT",DecisionTreeClassifier()))
models.append(("ADB",AdaBoostClassifier()))
models.append(("GB",GradientBoostingClassifier()))

ensemble = VotingClassifier(estimators=models)
ensemble.fit(X_train,y_train) y_pred =
ensemble.predict(X_test)
print(classification_report(y_pred,y_test))
print("Voting Ensemble:>",accuracy_score(y_pred,y_test))



SVM = SVC(kernel="linear",class_weight="balanced",probability=True)
SVM.fit(X_train,y_train) y_pred =
SVM.predict(X_test)
print(classification_report(y_pred,y_test))
print("SVM:>",accuracy_score(y_pred,y_test))


XGBC = XGBClassifier(learning_rate =0.1,n_estimators=10000,max_depth=4,min_child_weight=6,gamma=0,subsa
mple=0.6,colsample_bytree=0.8,
 reg_alpha=0.005, objective= 'binary:logistic', nthread=2, scale_pos_weight=1, seed=27)
XGBC.fit(X_train,y_train) y_pred = XGBC.predict(X_test)
print(classification_report(y_pred,y_test))
print("XGBoost:>",accuracy_score(y_pred,y_test))
```

```
Model1 = RandomForestClassifier(n_estimators=1000,random_state=0,n_jobs=1000,max_depth=70,bootstrap=True)
Model1.fit(X_train,y_train) y_pred =
Model1.predict(X_test)
print(classification_report(y_pred,y_test))
print("RandomForestClassifier:>",accuracy_score(y_pred,y_test))

Model2 = GradientBoostingClassifier()
Model2.fit(X_train,y_train) y_pred =
Model2.predict(X_test)
print(classification_report(y_pred,y_test))
print("GradientBoostingClassifier:>",accuracy_score(y_pred,y_test))


Model3 = DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=100,
max_features=1.0, max_leaf_nodes=10,  min_impurity_split=1e-07,
min_samples_leaf=1,  min_samples_split=2, min_weight_fraction_leaf=0.10,
presort=False, random_state=27, splitter='best') Model3.fit(X_train,y_train) y_pred =
Model3.predict(X_test) print(classification_report(y_pred,y_test))
print("DecisionTreeClassifier:>",accuracy_score(y_pred,y_test))


Model4 = AdaBoostClassifier()
Model4.fit(X_train,y_train) y_pred =
Model4.predict(X_test)
print(classification_report(y_pred,y_test))
print("AdaBoostClassifier:>",accuracy_score(y_pred,y_test))


Model5 = LinearDiscriminantAnalysis()
Model5.fit(X_train,y_train) y_pred =
Model5.predict(X_test)
print(classification_report(y_pred,y_test))
print("LinearDiscriminantAnalysis:>",accuracy_score(y_pred,y_test))

KNN = KNeighborsClassifier(leaf_size=1,p=2,n_neighbors=20)
KNN.fit(X_train,y_train) y_pred =
KNN.predict(X_test)
print(classification_report(y_pred,y_test))
print("KNeighborsClassifier:>",accuracy_score(y_pred,y_test))


Model7 = GaussianNB()
Model7.fit(X_train,y_train) y_pred =
Model7.predict(X_test)
print(classification_report(y_pred,y_test))
print("GaussianNB:>",accuracy_score(y_pred,y_test))
```

Model8 = LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
verbose=0, warm_start=False) Model8.fit(X_train,y_train) y_pred =
Model8.predict(X_test) print(classification_report(y_pred,y_test))
print("Logistic Regression:>",accuracy_score(y_pred,y_test))
After Standardization
Mean -3.064327691705293e-16 Standard Deviation 1.0

	precision	recall	f1-score	support
0	0.60	0.76	0.67	38
1	0.81	0.67	0.74	58
accuracy			0.71	96
macro avg	0.71	0.72	0.71	96
weighted avg	0.73	0.71	0.71	96

Voting Ensemble:> 0.7083333333333334

	precision	recall	f1-score	support
0	0.42	1.00	0.59	20
1	1.00	0.63	0.77	76
accuracy			0.71	96
macro avg	0.71	0.82	0.68	96
weighted avg	0.88	0.71	0.74	96

SVM:> 0.7083333333333334

	precision	recall	f1-score	support
0	0.48	0.62	0.54	37
1	0.71	0.58	0.64	59
accuracy			0.59	96
macro avg	0.59	0.60	0.59	96
weighted avg	0.62	0.59	0.60	96

XGBoost:> 0.59375

	precision	recall	f1-score	support
0	0.58	0.78	0.67	36
1	0.83	0.67	0.74	60
accuracy			0.71	96 macro
avg	0.71	0.72	0.70	96 weighted
avg	0.74	0.71	0.71	96

RandomForestClassifier:> 0.7083333333333334
precision recall f1-score support

0 0.56 0.64 0.60 42
1 0.69 0.61 0.65 54

accuracy 0.62 96 macro
avg 0.62 0.63 0.62 96 weighted
avg 0.63 0.62 0.63 96

GradientBoostingClassifier:> 0.625
precision recall f1-score support

0 0.54 0.87 0.67 30
1 0.92 0.67 0.77 66

accuracy 0.73 96
macro avg 0.73 0.77 0.72 96
weighted avg 0.80 0.73 0.74 96

DecisionTreeClassifier:> 0.7291666666666666
precision recall f1-score support

0 0.58 0.85 0.69 33
1 0.90 0.68 0.77 63

accuracy 0.74 96
macro avg 0.74 0.77 0.73 96
weighted avg 0.79 0.74 0.75 96

AdaBoostClassifier:> 0.7395833333333334
precision recall f1-score support

0 0.50 0.77 0.61 31
1 0.85 0.63 0.73 65

accuracy 0.68 96 macro
avg 0.68 0.70 0.67 96 weighted
avg 0.74 0.68 0.69 96

LinearDiscriminantAnalysis:> 0.6770833333333334 precision recall f1-score support

0 0.50 0.83 0.62 29
1 0.90 0.64 0.75 67

```
   accuracy                         0.70        96
macro avg       0.70      0.73      0.69        96
weighted avg       0.78      0.70      0.71        96

KNeighborsClassifier:> 0.6979166666666666
              precision    recall  f1-score   support

           0       0.42      0.91      0.57        22
1       0.96      0.62      0.75        74

   accuracy                         0.69        96
macro avg       0.69      0.77      0.66        96
weighted avg       0.83      0.69      0.71        96

GaussianNB:> 0.6875
              precision    recall  f1-score   support

           0       0.54      0.74      0.63        35
1       0.81      0.64      0.72        61

   accuracy                         0.68        96
macro avg       0.68      0.69      0.67        96
weighted avg       0.71      0.68      0.68        96

Logistic Regression:> 0.6770833333333334
```

In [23]:

```
#Experiment 3: Sclaed + Up Sampled Data

#X = df_upsampled.drop(["Loan_ID","Gender","Married","Education","Self_Employed","Loan_Amount_Term","Lo
an_Status",'Property_Area'],axis=1)
X = df_upsampled.drop(["Loan_Status","Loan_ID"],axis=1)
y = df_upsampled.Loan_Status print(len(X),len(y))
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.25,random_state=0)

#Scaling data here:---------->

StSc = StandardScaler()
X_train  = StSc.fit_transform(X_train)
X_test  = StSc.fit_transform(X_test)

#Check mean is 0 and Standard deviation is 1
print("After Standardization\nMean ",np.mean(X_train),"Standard Deviation ",np.std(X_train),"\n")

#Voting ensemble mathod. Combining all tree based algorithms.
models = [] models.append(("XGB",XGBClassifier()))
models.append(("RF",RandomForestClassifier()))
models.append(("DT",DecisionTreeClassifier()))
```

```
models.append(("ADB",AdaBoostClassifier()))
models.append(("GB",GradientBoostingClassifier()))

ensemble = VotingClassifier(estimators=models)
ensemble.fit(X_train,y_train) y_pred =
ensemble.predict(X_test)
print(classification_report(y_pred,y_test))
print("Voting Ensemble:>",accuracy_score(y_pred,y_test))



SVM = SVC(kernel="linear",class_weight="balanced",probability=True)
SVM.fit(X_train,y_train) y_pred =
SVM.predict(X_test)
print(classification_report(y_pred,y_test))
print("SVM:>",accuracy_score(y_pred,y_test))


XGBC = XGBClassifier(learning_rate =0.1,n_estimators=10000,max_depth=4,min_child_weight=6,gamma=0,subsa
mple=0.6,colsample_bytree=0.8,
 reg_alpha=0.005, objective= 'binary:logistic', nthread=2, scale_pos_weight=1, seed=27)
XGBC.fit(X_train,y_train) y_pred = XGBC.predict(X_test)
print(classification_report(y_pred,y_test))
print("XGBoost:>",accuracy_score(y_pred,y_test))

Model1 = RandomForestClassifier(n_estimators=1000,random_state=0,n_jobs=1000,max_depth=70,bootstrap=True)
Model1.fit(X_train,y_train) y_pred =
Model1.predict(X_test)
print(classification_report(y_pred,y_test))
print("RandomForestClassifier:>",accuracy_score(y_pred,y_test))


Model2 = GradientBoostingClassifier()
Model2.fit(X_train,y_train) y_pred =
Model2.predict(X_test)
print(classification_report(y_pred,y_test))
print("GradientBoostingClassifier:>",accuracy_score(y_pred,y_test))


Model3 = DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=100,
max_features=1.0, max_leaf_nodes=10,  min_impurity_split=1e-07,
min_samples_leaf=1,  min_samples_split=2, min_weight_fraction_leaf=0.10,
presort=False, random_state=27, splitter='best') Model3.fit(X_train,y_train) y_pred =
Model3.predict(X_test) print(classification_report(y_pred,y_test))
print("DecisionTreeClassifier:>",accuracy_score(y_pred,y_test))
```

```
Model4 = AdaBoostClassifier()
Model4.fit(X_train,y_train) y_pred =
Model4.predict(X_test)
print(classification_report(y_pred,y_test))
print("AdaBoostClassifier:>",accuracy_score(y_pred,y_test))


Model5 = LinearDiscriminantAnalysis()
Model5.fit(X_train,y_train) y_pred =
Model5.predict(X_test)
print(classification_report(y_pred,y_test))
print("LinearDiscriminantAnalysis:>",accuracy_score(y_pred,y_test))

KNN = KNeighborsClassifier(leaf_size=1,p=2,n_neighbors=20)
KNN.fit(X_train,y_train) y_pred =
KNN.predict(X_test)
print(classification_report(y_pred,y_test))
print("KNeighborsClassifier:>",accuracy_score(y_pred,y_test))


Model7 = GaussianNB()
Model7.fit(X_train,y_train) y_pred =
Model7.predict(X_test)
print(classification_report(y_pred,y_test))
print("GaussianNB:>",accuracy_score(y_pred,y_test))


Model8 = LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
        intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
verbose=0, warm_start=False) Model8.fit(X_train,y_train) y_pred =
Model8.predict(X_test) print(classification_report(y_pred,y_test))
print("Logistic Regression:>",accuracy_score(y_pred,y_test))
844 844
After Standardization
Mean  7.143184188310644e-17 Standard Deviation  1.0

# Experiment 4: Sclaed + Selected features with respective importance
#Droping features which are less important and keeping features as per importance analysis.
X = new_data.drop(["Loan_ID","Gender","Married","Education","Self_Employed","Loan_Amount_Term","Loan_St
atus","Property_Area"],axis=1)
#X = new_data.drop(["Loan_Status","Loan_ID"],axis=1)
y = new_data.Loan_Status print(len(X),len(y))
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.25,random_state=0)

#Scaling data here:---------->
```

```
StSc = StandardScaler()
X_train  = StSc.fit_transform(X_train)
X_test  = StSc.fit_transform(X_test)

#Check mean is 0 and Standard deviation is 1
print("After Standardization\nMean ",np.mean(X_train),"Standard Deviation ",np.std(X_train),"\n")

#Voting ensemble mathod. Combining all tree based algorithms.
models = [] models.append(("XGB",XGBClassifier()))
models.append(("RF",RandomForestClassifier()))
models.append(("DT",DecisionTreeClassifier()))
models.append(("ADB",AdaBoostClassifier()))
models.append(("GB",GradientBoostingClassifier()))

ensemble = VotingClassifier(estimators=models)
ensemble.fit(X_train,y_train) y_pred =
ensemble.predict(X_test)
print(classification_report(y_pred,y_test))
print("Voting Ensemble:>",accuracy_score(y_pred,y_test))



SVM = SVC(kernel="linear",class_weight="balanced",probability=True)
SVM.fit(X_train,y_train) y_pred =
SVM.predict(X_test)
print(classification_report(y_pred,y_test))
print("SVM:>",accuracy_score(y_pred,y_test))
XGBC = XGBClassifier(learning_rate
=0.1,n_estimators=10000,max_depth=4,min_c
hild_weight=6,gamma=0,subsa
mple=0.6,colsample_bytree=0.8,
 reg_alpha=0.005, objective= 'binary:logistic', nthread=2, scale_pos_weight=1, seed=27)
XGBC.fit(X_train,y_train) y_pred = XGBC.predict(X_test)
print(classification_report(y_pred,y_test))
print("XGBoost:>",accuracy_score(y_pred,y_test))

Model1 = RandomForestClassifier(n_estimators=1000,random_state=0,n_jobs=1000,max_depth=70,bootstrap=True)
Model1.fit(X_train,y_train) y_pred =
Model1.predict(X_test)
print(classification_report(y_pred,y_test))
print("RandomForestClassifier:>",accuracy_score(y_pred,y_test))


Model2 = GradientBoostingClassifier()
Model2.fit(X_train,y_train) y_pred =
Model2.predict(X_test)
print(classification_report(y_pred,y_test))
```

```
print("GradientBoostingClassifier:>",accuracy_score(y_pred,y_test))


Model3 = DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=100,
max_features=1.0, max_leaf_nodes=10,  min_impurity_split=1e-07,
min_samples_leaf=1,  min_samples_split=2, min_weight_fraction_leaf=0.10,
presort=False, random_state=27, splitter='best') Model3.fit(X_train,y_train) y_pred =
Model3.predict(X_test) print(classification_report(y_pred,y_test))
print("DecisionTreeClassifier:>",accuracy_score(y_pred,y_test))


Model4 = AdaBoostClassifier()
Model4.fit(X_train,y_train) y_pred =
Model4.predict(X_test)
print(classification_report(y_pred,y_test))
print("AdaBoostClassifier:>",accuracy_score(y_pred,y_test))


Model5 = LinearDiscriminantAnalysis()
Model5.fit(X_train,y_train) y_pred =
Model5.predict(X_test)
print(classification_report(y_pred,y_test))
print("LinearDiscriminantAnalysis:>",accuracy_score(y_pred,y_test))

KNN = KNeighborsClassifier(leaf_size=1,p=2,n_neighbors=20)
KNN.fit(X_train,y_train) y_pred
= KNN.predict(X_test)
print(classification_report(y_pred,y_test))
print("KNeighborsClassifier:>",accuracy_score(y_pred,y_test))


Model7 = GaussianNB()
Model7.fit(X_train,y_train) y_pred =
Model7.predict(X_test)
print(classification_report(y_pred,y_test))
print("GaussianNB:>",accuracy_score(y_pred,y_test))


Model8 = LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
        intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
verbose=0, warm_start=False) Model8.fit(X_train,y_train) y_pred =
Model8.predict(X_test) print(classification_report(y_pred,y_test))
print("Logistic Regression:>",accuracy_score(y_pred,y_test))
614 614
After Standardization
Mean  -3.2669519263752433e-16 Standard Deviation  1.0
```

```
#Hyperparameters tuning for KNN

#X = new_data.drop(["Loan_ID","Gender","Married","Education","Self_Employed","Loan_Amount_Term","Loan_S
tatus","Property_Area"],axis=1)
X = new_data.drop(["Loan_Status","Loan_ID"],axis=1)
y = new_data.Loan_Status print(len(X),len(y))
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.25,random_state=0)
leaf_size = list(range(1,50)) n_neighbors = list(range(1,30)) p=[1,2]
#Convert to dictionary
hyperparameters = dict(leaf_size=leaf_size, n_neighbors=n_neighbors, p=p)
#Create new KNN object knn_2
= KNeighborsClassifier()
#Use GridSearch
clf = GridSearchCV(knn_2, hyperparameters, cv=10)
#Fit the model
best_model = clf.fit(X_train,y_train)
#Print The value of best Hyperparameters
print('Best leaf_size:', best_model.best_estimator_.get_params()['leaf_size']) print('Best
p:', best_model.best_estimator_.get_params()['p'])
print('Best n_neighbors:', best_model.best_estimator_.get_params()['n_neighbors'])

LS = best_model.best_estimator_.get_params()['leaf_size']
P = best_model.best_estimator_.get_params()['p']
Num = best_model.best_estimator_.get_params()['n_neighbors']

KNN = KNeighborsClassifier(leaf_size=LS,p=P,n_neighbors=Num)
KNN.fit(X_train,y_train) y_pred
= KNN.predict(X_test)
print(classification_report(y_pred,y_test))
print("KNeighborsClassifier:>",accuracy_score(y_pred,y_test)) 614
614
Best leaf_size: 1
Best p: 1
Best n_neighbors: 10
              precision    recall  f1-score   support

0       0.49      0.84      0.62        25
1       0.96      0.83      0.89       129

    accuracy                           0.83       154
macro avg       0.73      0.83      0.75       154
weighted avg       0.89      0.83      0.85       154

KNeighborsClassifier:> 0.8311688311688312

# Tuning SVM parameters
```

```
#X = new_data.drop(["Loan_ID","Gender","Married","Education","Self_Employed","Loan_Amount_Term","Loan_Status","Property_Area"],axis=1)
X = new_data.drop(["Loan_Status","Loan_ID"],axis=1)
y = new_data.Loan_Status print(len(X),len(y))
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.25,random_state=0) model
= SVC()
kernel = ['poly', 'rbf', 'sigmoid'] C
= [50, 10, 1.0, 0.1, 0.01]
gamma = ['scale'] #
define grid search
grid = dict(kernel=kernel,C=C,gamma=gamma)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv, scoring='accuracy',error_score=0
)
grid_result = grid_search.fit(X, y)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score'] stds =
grid_result.cv_results_['std_test_score'] params =
grid_result.cv_results_['params'] for mean, stdev, param in zip(means, stds,
params):     print("%f (%f) with: %r" % (mean, stdev, param))
614 614
Best: 0.689468 using {'C': 50, 'gamma': 'scale', 'kernel': 'rbf'}
0.687308 (0.005314) with: {'C': 50, 'gamma': 'scale', 'kernel': 'poly'}
0.689468 (0.016542) with: {'C': 50, 'gamma': 'scale', 'kernel': 'rbf'}
0.686215 (0.027277) with: {'C': 50, 'gamma': 'scale', 'kernel': 'sigmoid'}
0.687308 (0.005314) with: {'C': 10, 'gamma': 'scale', 'kernel': 'poly'}
0.682434 (0.009840) with: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
0.687308 (0.026685) with: {'C': 10, 'gamma': 'scale', 'kernel': 'sigmoid'}
0.687308 (0.005314) with: {'C': 1.0, 'gamma': 'scale', 'kernel': 'poly'}
0.687308 (0.005314) with: {'C': 1.0, 'gamma': 'scale', 'kernel': 'rbf'}
0.687308 (0.005314) with: {'C': 1.0, 'gamma': 'scale', 'kernel': 'sigmoid'}
0.687308 (0.005314) with: {'C': 0.1, 'gamma': 'scale', 'kernel': 'poly'}
0.687308 (0.005314) with: {'C': 0.1, 'gamma': 'scale', 'kernel': 'rbf'}
0.687308 (0.005314) with: {'C': 0.1, 'gamma': 'scale', 'kernel': 'sigmoid'}
0.687308 (0.005314) with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'poly'}
0.687308 (0.005314) with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'rbf'}
0.687308 (0.005314) with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'sigmoid'}
```

CHAPTER 5.
CONCLUSION AND FUTURE WORK

5.1 Conclusion

In conclusion, the development of a loan approval system is a multifaceted endeavor that requires careful consideration of various factors and specifications to ensure its effectiveness, security, and compliance with regulatory requirements. By prioritizing scalability, security, performance, integration, compliance, customization, auditability, and user access control, the system can meet the diverse needs of financial institutions while providing a seamless user experience for borrowers.

Scalability is essential for accommodating fluctuations in workload and ensuring the system can handle a large volume of loan applications efficiently. Robust security measures, including encryption, authentication, access controls, and audit trails, are paramount to protect sensitive applicant information and prevent unauthorized access or data breaches. Performance optimization techniques, such as caching and load balancing, help minimize response times and maximize system throughput, enhancing the user experience. Seamless integration with external systems and data sources enables the system to gather necessary information for evaluating loan applications effectively.

Compliance with regulatory requirements, including anti-money laundering (AML), know your customer (KYC), and fair lending laws, is critical to mitigate legal and reputational risks associated with lending

operations. Customization options allow lenders to tailor eligibility criteria, credit scoring models, and approval workflows to their specific preferences and risk appetite.

Comprehensive auditability features maintain a detailed audit trail of all loan application activities and user interactions, ensuring accountability and compliance with regulatory reporting requirements. Granular user access control mechanisms manage user permissions and restrict access to sensitive features and data based on predefined roles and privileges, minimizing the risk of unauthorized access or data leakage. In summary, a well-designed loan approval system that incorporates these specifications can streamline the lending process, mitigate risk, ensure regulatory compliance, and provide a positive user experience for both lenders and borrowers. By leveraging advanced technologies and best practices, financial institutions can make informed lending decisions while maintaining the highest standards of security, reliability, and efficiency.

5.2 Future Work:

Advanced Machine Learning Models: Incorporate more advanced machine learning models, such as deep learning algorithms or ensemble methods, to enhance credit scoring accuracy and prediction capabilities. Continuously refine and optimize the models using additional data and feedback loops to improve performance over time.

We can make the Bank Loan Approval prediction to connect with Cloud for future use to optimize the work to implement in Artificial Intelligence environment.

Real-time Data Integration: Implement real-time data integration capabilities to access and analyze up-todate information from external sources, such as transaction data, social media activity, or alternative credit data. This allows for more dynamic and accurate assessment of applicant creditworthiness.

Predictive Analytics: Develop predictive analytics capabilities to forecast loan default risk and identify early warning signs of potential delinquency. Utilize predictive modeling techniques to proactively manage credit risk and optimize lending decisions.

Automation and AI-driven Decisioning: Further automate the loan approval process using artificial intelligence (AI) and natural language processing (NLP) technologies to analyze applicant data, assess credit risk, and make real-time loan decisions. Implement AI-driven chatbots or virtual assistants to assist applicants and provide personalized support throughout the loan application process.

Blockchain Technology: Explore the use of blockchain technology to enhance security, transparency, and efficiency in loan origination, processing, and documentation. Implement smart contracts to automate loan agreements and facilitate secure, immutable record-keeping.

Enhanced Regulatory Compliance: Continuously monitor changes in regulatory requirements and update the system accordingly to ensure ongoing compliance with evolving laws and regulations. Implement automated compliance checks and reporting features to streamline regulatory compliance processes.

Customer Relationship Management (CRM) Integration: Integrate the loan approval system with CRM platforms to improve customer relationship management and streamline communication with loan applicants. Capture and analyze customer interactions to better understand borrower needs and preferences.

Mobile Accessibility: Enhance the system's mobile accessibility and responsiveness to cater to the growing number of users accessing financial services through mobile devices. Develop mobile applications or responsive web interfaces optimized for a seamless user experience on smartphones and tablets.

Data Privacy and Security Enhancements: Implement additional data privacy and security enhancements, such as advanced encryption techniques, data anonymization, and secure data sharing protocols, to protect sensitive applicant information and ensure compliance with global privacy regulations.

User Experience (UX) Enhancements: Continuously improve the system's user experience through usability testing, user feedback, and iterative design enhancements. Focus on creating intuitive interfaces, streamlined workflows, and personalized experiences to enhance user satisfaction and engagement.

REFERENCES

[1] K. Hanumantha Rao., G. Srinivas., A. Damodhar., M.Vikas Krishna., Implementation of Anomaly Detection Technique Using Machine Learning Algorithms: International Journal of Computer Science and Telecommunications, Vol. 2, Issue 3, 2011.

[2] S.S. Keerthi., E.G. Gilbert., Convergence of a generalize SMO algorithm for SVM classifier design, Machine Learning, Springer, Vol. 4, Issue 1, pp. 351-360, 2002.

[3] Andy Liaw., Matthew Wiener., Classification and Regression by random Forest, Vol. 2, Issue 3, pp. 922,2002.

[4] Ekta Gandotra., Divya Bansal., Sanjeev Sofat., Malware Analysis and Classification: A Survey, Journal of Information Security, Vol. 05, Issue 02, pp. 56-64,2014.

[5] Rattle data mining tool, http://rattle.togaware.com/rattle-download.html.

[6] Aafer Y., Du W., Yin H., Droid APIMiner: MiningAPI-Level Features for Robust Malware Detection inAndroid, Security and privacy in CommunicatioNetworks, Springer, pp 86-103, 2013. [7] J. R. Quinlan., Induction of Decision Tree, MachineLearning, Vol. 1, No. 1. pp. 81-106