Loan Default Prediction Using Machine Learning

Financial Management

MGT 1029

PROJECT REPORT

Winter Semester: 2022-2023

Loan Default Prediction Using Machine Learning

Project Report submitted in partial fulfillment of the requirements for the course of

Financial Management

MGT 1029

Winter Semester: 2022-2023

By

Arbbaz Alif A S **20BEE1115**

ACKNOWLEDGEMENT

In the completion of our project on Loan Default Prediction System I would like to convey my special gratitude to Dr.Subhamitra Patra, of VITBS. Your valuable guidance and suggestions helped us in various phases of the completion of this project. What I learnt through this course is valuable and essential irrespective of what career path we take. We will always be thankful to you in this regard.

ABSTRACT

Banks offer personal loans to their customers and the money can be used for any expense like paying a bill or purchasing a new television. Loans are a critical source of funding for many individuals and businesses. Without access to loans, many people would not be able to afford to buy a home, start a business, or invest in their education. However, loans can also be risky. Borrowers must repay the loan with interest, which can be challenging, especially if their financial situation changes or they encounter unexpected expenses. Banks also face risks when they lend money, such as the risk of default, where a borrower is unable to repay the loan. The loan is considered a default when borrowers miss consecutive repayments beyond the delinquency periods.

To mitigate these risks, banks use a variety of tools, such as credit scoring, collateral, and loan covenants, to assess and manage the risk of lending. The objective is to predict whether a loan applicant will be able to repay the loan or not based on various customer behaviors and loan characteristics.The code uses various machine learning models to predict the loan status, and the model with the highest accuracy is chosen as the final model. The aim is to help financial institutions make informed decisions about approving or rejecting loan applications to minimize the risk of loan defaults.

1. Research Problem and motivation of study

1.1 Research Problem

Banks are financial institutions that provide a range of financial services to their customers. They are licensed by regulatory authorities and operate under strict regulations to ensure their safety and soundness.

The primary function of banks is to accept deposits from customers and use those funds to make loans to other customers. This process is known as intermediation and is a key component of the banking system. Banks also offer a range of other financial services, such as credit cards, savings accounts, checking accounts, money market accounts, and certificates of deposit. Loans are financial products offered by banks and other financial institutions that allow individuals or businesses to borrow money to finance various expenses or investments.

Loans come in different types and sizes, with different terms and conditions, including interest rates, repayment schedules, and collateral requirements. Personal loans, mortgages, and business loans are the three most typical sorts of loans. Personal loans are typically unsecured loans that are intended to pay for individual costs like debt consolidation, house renovations, and medical bills. Mortgages are long-term loans used to fund the acquisition of real estate, including homes. Businesses borrow money to finance their daily operations, capital expenditures, and expansion ambitions. In many cases, loans are essential for economic growth, and without them, the economy would stagnate.

However, loans can also be risky. Borrowers must repay the loan with interest, which can be challenging, especially if their financial situation changes or they encounter unexpected expenses. Banks also face risks when they lend money, such as the risk of default, where a borrower is unable to repay the loan. The loan is considered a default when borrowers miss consecutive repayments beyond the delinquency periods. A delayed repayment or loan default considerably reduces borrowers' credit scores. It becomes a permanent mark on a borrower's credit history. Poor credit history makes it hard for the borrower to secure future

loans. To mitigate these risks, banks use a variety of tools, such as credit scoring, collateral, and loan covenants, to assess and manage the risk of lending.

1.2 Motivation of Study

Loans are a key building block of the financial system and are crucial for fostering economic expansion. Financial firms may manage risk and make sure they are lending to creditworthy borrowers by accurately estimating the possibility of default or repayment. This in turn may enhance financial system stability and economic expansion. By studying loan default we can understand how it affects the economy, lenders can assess risk, investment decisions and the likelihood of consumers falling into debt traps.

1. *Risk management:*

   By properly understanding loan default, financial institutions and lenders can better control the risks involved in making loans. Lenders can create more effective risk-mitigation strategies, such as modifying their lending standards, establishing interest rates, and enhancing their credit evaluation procedures, by determining the factors that contribute to loan defaults. Here are some ways in which loan default prediction can help with risk management:

   - Credit risk assessment:Models for predicting loan defaults can be used by lenders to evaluate the credit risk of new borrowers. These models can offer a chance of default for each borrower by looking at a number of variables, including credit history, income, and employment status. This data can be used by lenders to decide how much money to lend and at what interest rate, as well as whether or not to approve a loan application.

   - Portfolio risk management:Predicting loan defaults is a useful tool for risk management across a lender's loan portfolio. Lenders can determine which

parts of their portfolio are more or less dangerous by evaluating the credit risk of each loan and borrower. To reduce risk, this information can be utilized to diversify the portfolio, allocate resources, and change lending rules.

- Early warning system: Models for predicting loan defaults can also act as a lender's early warning system. Lenders can spot borrowers who may be in default by observing borrower behavior, such as missed payments or changes in job status. Using this information, preemptive steps can be taken to reduce the likelihood of default, such as providing loan modifications or working with debtors to create repayment plans.

2. *Economic Effects:*

Loan defaults can have a big impact on the economy, especially when things are uncertain economically. A high rate of loan defaults may cause lending activity to decline, which may impede economic growth and result in job losses.

- Systemic risk assessment: Predicting loan defaults can assist in identifying possible systemic threats to the financial system. If a lot of loans are expected to fail in one industry or region, this can indicate more general economic problems that could affect the entire economy. Policymakers can proactively reduce systemic risk and stabilize the financial system by analyzing these developments.

- Economic forecasting: Models for predicting loan defaults can be used to predict economic trends and spot future economic obstacles. For instance, if a model predicts a rise in loan defaults in a certain industry, this may indicate a downturn in either that industry or the overall economy. Policymakers can

more accurately predict economic trends and decide on monetary and fiscal policy by including these projections into economic forecasting models.

- Credit availability analysis: Loan default prediction can also be used to analyze credit availability and its impact on the economy. By monitoring changes in lending patterns and predicting loan defaults, policymakers can identify potential barriers to credit availability for businesses and consumers. This information can be used to develop policies and programs to support access to credit and promote economic growth.

3. *Consumer Protection:*

Being aware of loan default can help consumers avoid debt traps and predatory lending techniques. Policymakers can pinpoint areas where consumer protections might be required to stop borrowers from taking on loans they cannot afford by studying loan default. Here are some ways in which loan default prediction can contribute to consumer protection:

- Responsible lending practices: Loan default prediction can encourage responsible lending practices among financial institutions. By analyzing the credit risk of each borrower and predicting the likelihood of default, lenders can ensure that they are not extending credit to individuals who are likely to default on their loans. This helps to protect borrowers from taking on debt that they cannot afford and promotes responsible lending practices.

- Fair lending practices: Loan default prediction can also contribute to fair lending practices by ensuring that lenders are not discriminating against certain groups of borrowers. By using predictive models that are based on objective factors such as credit history and income, lenders can avoid lending

practices that discriminate against individuals based on their race, gender, or other protected characteristics.

- Financial education: Loan default prediction can also be used to promote financial education and literacy among consumers. By providing borrowers with information about the credit risk of their loans and the factors that contribute to loan default, lenders can help borrowers make informed decisions about borrowing and managing debt.

4. *Investment decision:*

Here are some ways in which loan default prediction can contribute to investment decisions:

- Risk assessment: Loan default prediction can help investors assess the risk associated with a particular investment. By analyzing the credit risk of the underlying assets and predicting the likelihood of default, investors can determine the level of risk associated with the investment and make informed decisions about how to manage their risk exposure.

- Portfolio management: Loan default prediction can also be used to manage risk across an investment portfolio. By analyzing the credit risk of each asset and predicting the likelihood of default, investors can identify areas of their portfolio that are more or less risky. This information can be used to adjust investment strategies, allocate resources, and diversify the portfolio to minimize risk.

- Performance evaluation: Loan default prediction can also be used to evaluate the performance of an investment portfolio. By monitoring the credit risk of the underlying assets and comparing actual default rates to predicted default

rates, investors can evaluate the effectiveness of their investment strategies and make adjustments as needed.

2. Literature review

This research article(Xia, Yufei, et al.(2020)) discusses the emergence and challenges of peer-to-peer (P2P) lending, a type of crowdfunding where borrowers and lenders are matched online without the involvement of commercial banks. P2P lending faces severe information asymmetry problems, which can be alleviated through internal credit scoring systems. However, relying solely on hard information is not enough, and soft information contained in narrative data, such as loan purpose and trustworthiness of borrowers, is often ignored. To address this, the authors proposed an automatic framework to extract soft information using keyword clustering algorithms and used an advanced GBDT algorithm, CatBoost, to predict loan default based on both hard and soft information. The proposal was validated on three real-world datasets and showed a significant improvement in predictive accuracy compared to models using only hard information or simplistic soft information. This study offers insights into the potential of using soft information to enhance credit scoring in P2P lending.

This study(Foster, B. P., & Zurada, J. (2013)) examines whether including loan default status and audit opinion variables in bankruptcy prediction models improves their accuracy, using a sample of financially distressed publicly traded US companies from 2003 to 2007. The research enables them to find that including these variables improves the accuracy of hazard bankruptcy prediction models and changes the significance of some variables included in previous models. However, the authors of the research paper take note that their results may not be generalizable to other bankruptcy hazard models due to their limited sample data that has been considered.Hence,it highlights the importance of including loan default status and audit opinion variables in hazard models and suggests that auditors should give more weight to loan default status in their going-concern modification decisions.

The research paper(Gomathy, C. K., et al.(2021))presents a machine learning-based model to predict whether a loan applicant will default on their loan or not. The authors discuss the importance of loan prediction in the financial industry and the challenges faced in predicting loan defaults accurately.The authors compare the performance of several machine learning algorithms such as logistic regression, decision trees, random forest, and support vector machine. They use a publicly available dataset from Kaggle that contains loan application information to train and test their models.The results of the study show that the random forest algorithm outperformed the other algorithms with an accuracy of 80.94%, precision of 78.32%, recall of 54.58%, and an F1-score of 64.92%. The authors discuss the significance of these results in the context of loan prediction accuracy and the

potential impact on financial institutions.The study concludes that machine learning-based loan prediction models can be valuable tools for financial institutions to assess the creditworthiness of loan applicants and minimize loan default risks. The authors also highlight the importance of selecting appropriate machine learning algorithms for specific applications and the need for ongoing monitoring and updating of models to maintain their accuracy and effectiveness.

The paper(Al Mamun M, Farjana A.(2022))discusses the application of machine learning models to predict the eligibility of bank loans for customers. The study uses a dataset with information on the credit history and personal details of customers, and compares the performance of different models, including Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine.The results show that the Random Forest model outperforms the other models in terms of accuracy, precision, and recall, with an accuracy score of 86.67%. The study also identifies the most important features for predicting loan eligibility, which include the loan amount, credit history, and applicant income.Overall, the study demonstrates the potential of machine learning models in predicting loan eligibility and highlights the importance of feature selection in improving model performance. The results may be useful for banks and financial institutions in streamlining their loan approval processes and improving their risk management strategies.

Bank credit risk assessment is crucial for evaluating the potential that a borrower or counterparty may fail to meet their obligations. Banks use various techniques, including financial ratios and subjective factors, to assess credit risk.(Krichene, A. (2017)) The Basel Committee on Banking Supervision recommends validating internal credit risk assessment models and offers two broad methodologies for calculating capital requirements for credit risk: external mapping approach and internal rating system. Credit scoring methods are commonly used to evaluate both objective and subjective factors. These models include traditional statistical methods, nonparametric statistical models, and classification trees.In commercial banks, credit risk measurement is important for differentiating reliable clients from non-reliable ones. The use of models that predict defaults correctly is necessary for this purpose. The Bayesian classifier is a quantitative method used to estimate the posterior probabilities of default. The posterior probability is the probability of an event after collecting empirical data, obtained by integrating the prior probability with additional data related to the event. Rosner (2006) and Antonakis and Sfakianakis (2009) have both demonstrated the application of Bayesian classifiers for credit risk evaluation.Commercial banks require reliable models to predict and detect loan defaults. Credit scoring is a

commonly used quantitative method that applies classification techniques to assess creditworthiness. Moonasar (2007) stresses the importance of credit scoring in reducing credit risk, and notes that an accurate classifier is needed to differentiate between good and bad credit applicants. In this study, the authors used a database of 924 credit files to assess credit risk for a Tunisian bank, and found that the inclusion of cash flow variables improved the prediction quality. The authors also note that qualitative variables are important in assessing borrower solidity, and that the Tunisian central bank requires commercial banks to collect such data for better credit notation.

The P2P network lending industry has grown significantly due to the development of big data and internet finance, providing a direct connection between investors and borrowers without intermediaries. Lending Club is the world's largest online financial platform for borrowers and investors, offering lower costs and greater investment returns than traditional banks. However, P2P lending faces risks such as liquidity, credit, operational, and legal risks, which can lead to loan defaults. Scholars have conducted research on loan evaluation and credit risk assessment, with Random Forest being found to have better performance than other methods in P2P lending. Machine learning and artificial neural networks are also used for risk management controls in the financial field. This paper(Zhu, Lin, et al.(2019)) proposes to use the Random Forest algorithm to construct a loan default prediction model based on Lending Club's loans of the first quarter of 2019, with the aim of improving the performance of loan evaluation and promoting the healthy development of P2P lending.The paper compares the performance of four machine learning algorithms, namely Random Forest, Decision Tree, Logistic Regression, and SVM, in predicting accuracy, AUC, F1-Score, and recall. Results show that Random Forest performs the best, with an accuracy of 98%, higher than Decision Tree with an accuracy of 95%. The precision and recall of the Random Forest model are all above 0.95, indicating strong ability for generalization. The ROC curves also show that Random Forest outperforms the other three approaches.

The research article(Sheikh, M.A., Goel, A.K. and Kumar, T., (2020)) proposes an approach to predict loan approval using machine learning algorithms. The goal is to develop a model that can analyze historical data and identify patterns to predict whether a loan application will be approved or denied. The article suggests that by using various features such as income, credit score, loan amount, and employment status, a machine learning model can be trained to predict loan approval with high accuracy. The proposed

approach can potentially help financial institutions streamline their loan approval process and reduce the risk of bad loans.

The issue of credit risk has long been a concern for lenders in both developing and developed countries due to its significant impact on the lending sector and the overall economy(Adewusi, A.O., Oyedokun, T.B. and Bello, M.O., (2016)). Credit risk is related to a borrower's inability to fulfill their contractual obligations with a lender, resulting in non-performing loans. Effective credit risk evaluation and management models are being developed by lenders and researchers to address this issue. Poorly evaluated credit risk can lead to financial loss for lenders, but lending is necessary for institutions to make a profit and stay in business. NPLs are costly for banks and reduce their liquidity, leading to reduced credit expansion and slower growth. Various loan evaluation models are used by lenders to minimize credit risk, improve prediction accuracy, and reduce the volume of NPLs.

Typically, banks execute a loan application after verifying and evaluating the applicant's eligibility, which is a time-consuming and challenging process. When examining loan applications and making credit approval decisions, most banks use their credit score and risk assessment systems. Despite this, some applicants fail to pay their bills each year, causing financial institutions to lose a substantial amount of money. In this study(Alsaleem MY, Hasoon SO,(2020)) Machine Learning (ML) algorithms are employed to extract patterns from a common loan-approved dataset and predict deserving loan applicants. Customers' previous data will be used to undertake the study, including their age, income type, loan annuity, last credit bureau report, Type of organization they work for, and length of employment. We can forecast whether a given applicant is safe or not using our method, and the entire feature validation process is automated using machine learning techniques.The purpose of this paper is to provide a quick, straightforward, and efficient method of selecting qualified applicants.

In this paper(Udaya Bhanu, L. and Narayana, D.S., (2021)) they have proposed customer loan prediction using supervised learning techniques for loan candidates as a valid or fail to pay customer. In this study we can predict whether that particular applicant is safe or not and the whole method of validation of attribute is automated by machine learning technique The goal is to implement machine learning model so as to classify, to the best potential degree of accuracy, master card fraud from a dataset gathered from Kaggle. Upon initial knowledge exploration, we have a tendency to know we might implement a random forest

model for best accuracy reports. The disadvantage of this model is that it emphasizes completely different weights to every issue; however in reality sometimes loans can be approved on the premise of a single strong part only, that isn't possible through this method.

3. Research gap

New methods of mining narrative data. The keywords clustering method proposed in this paper (Xia, Yufei, et al.(2020)) provides one potential (possibly imperfect) solution to utilizing narrative data in credit scoring. Powerful methods to exploit narrative data in P2P lending are always encouraged. The CATBoost method being widely used in industry for ML models has a chance of small inaccuracy. Loan default prediction based on customer behavior needs to be explored for high AUC values.

Auditors(Foster, B. P., & Zurada, J. (2013)) consider many more information items than just the status of loan agreements. The loan default status variable used in this study is not sufficient to capture all the non-financial information auditors consider when making their going-concern modification decisions. The sample size of the study was limited due to the hand collection of loan default status information. The propensity of auditors to issue going-concern modifications varies over time due to changes in the legal liability of auditors. Therefore, the overall conclusions of this study may not be generalizable to other bankruptcy prediction hazard model settings.

The study(Gomathy, C. K., et al.(2021)) uses a limited number of features to predict loan eligibility, which may not capture all the relevant factors that affect a borrower's creditworthiness. The study focuses on a specific dataset, which may not be representative of the entire population, and may not generalize to other contexts or datasets. The study uses only two machine learning algorithms (Logistic Regression and Random Forest), which may not be the most optimal or accurate methods for predicting loan eligibility. The study does not provide a comprehensive analysis of the performance metrics of the machine learning models used, which makes it difficult to evaluate the effectiveness of the models in predicting loan eligibility accurately.

The paper(Al Mamun M, Farjana A.(2022)) has compared the performance of various machine learning models in predicting bank loan eligibility. However, there is a lack of comparison of these models with traditional statistical models, which could provide useful insights into the superiority of machine learning models. The paper has not considered external factors such as economic conditions and government policies, which could significantly impact the loan eligibility prediction. Considering such external factors could improve the accuracy of the models.

The primary findings demonstrate (Krichene, A. (2017)) that adding cash flow variables enhances prediction accuracy, with classification rates in the cash flow and non-cash flow models increasing from 59.63 to 63.85 percent, respectively. Moreover, collateral was crucial in predicting default risk. In reality, this indicator can explain the credit risk and predict it. An ROC curve was drawn to assess the model's performance. The outcome demonstrates that the AUC requirement is in the range of 69%. When we applied the NN methodology to the identical set of data, this criterion was enhanced and passed with a score of 83%. The study is however lacking in that only quantitative variables were examined, and the significance of qualitative variables based on strategic data in completing the financial analysis and determining the viability of a borrower was not demonstrated. We should point out that the Tunisian central bank required all commercial banks to complete a survey in order to gather qualitative information for a better assessment of the borrowers' creditworthiness.

The performance of the models (Zhu, Lin, et al.(2019)) may be impacted by how each algorithm's hyperparameters were tweaked, which is not mentioned in the study. The models might not function at their best if the hyperparameters are not tuned properly. There are numerous additional algorithms that might be employed for the same problem; however, the research only examines four machine learning techniques. The performance of the top algorithm might be better understood with a more thorough comparison.

The conclusion (Sheikh, M.A., Goel, A.K. and Kumar, T., (2020)) only mentions the best accuracy on the public test set, but it does not provide any information on other performance metrics or the performance of different models. Without a more comprehensive evaluation, it is difficult to determine the overall effectiveness of the model. The insights provided in the conclusion are relatively basic and do not provide any deeper analysis or interpretation of the results. For example, it is unclear why applicants with a high income are more likely to get approved, or how gender and marital status were evaluated and found to be insignificant.

The study(Adewusi, A.O., Oyedokun, T.B. and Bello, M.O., (2016)) focuses on the lending sector in developing countries, which may not necessarily be representative of lending markets in other countries. The findings of the study may not be applicable to lending

markets in developed countries. The use of post-recovery data assumes that all loans are recoverable, which may not be the case in practice. This may lead to an overestimation of the predictive accuracy of the ANN model.

In this study (Alsaleem MY, Hasoon SO,(2020)), Machine Learning (ML) algorithms are employed to extract patterns from a common loan-approved dataset and predict deserving loan applicants. Customers' previous data will be used to undertake the study, including their age, income type, loan annuity, last credit bureau report, Type of organization they work for, and length of employment. But, this study omits the current status of the applicants and their determination to pay back the loan.

In this paper (Udaya Bhanu, L. and Narayana, D.S., (2021)), they have proposed customer loan prediction using supervised learning techniques for loan candidates as a valid or fail to pay customer. The disadvantage of this model is that it emphasizes completely different weights to every issue; however in reality sometimes loan can be approved on the premise of a single strong part only, that isn’t possible through this method.

4. Objective

4.1 Project Aim

The objective of this project is to predict whether a borrower is likely to default on their loan using machine learning. Banks and other financial organizations can use this model to evaluate the creditworthiness of loan applicants and to make well-informed lending choices. The ultimate aim of a loan default prediction project using customer behavior is to develop a machine learning model that can accurately predict whether a customer will default on a loan based on their past behavior and financial history. The borrower's credit history, income, employment status, loan amount, and other relevant features must all be included in the dataset used to train the model in order to accomplish this goal.

4.2 Advantages of using a loan prediction algorithm in the banking industry

1. *Improved accuracy:*

   A loan default predicting algorithm can use a wide range of data points and machine learning models to accurately predict the likelihood of a borrower defaulting on a loan. This can improve the accuracy of loan risk assessments and reduce the risk of default.

2. *Faster decision-making:*

   By using an algorithm to predict loan defaults, banks and other lending institutions can make faster and more informed lending decisions. This can speed up the loan application process and allow borrowers to access funds more quickly.

3. *Reduced risk:*

   By accurately predicting loan defaults, banks and other lenders can reduce the risk of lending to high-risk borrowers. This can help to protect the financial health of the lending institution and prevent losses due to default.

4. *Improved customer experience:*

   By using a loan default predicting algorithm, banks and other lenders can more accurately assess a borrower's risk and provide loans that are tailored to their individual needs. This can improve the overall customer experience and help to build long-term relationships between lenders and borrowers.

5. *Compliance with regulations:*

   Many lending institutions are required to comply with regulations that require them to assess the risk of borrowers before issuing loans. A loan default predicting algorithm can help lenders to comply with these regulations by providing accurate risk assessments and ensuring that loans are issued responsibly.

5. Code and methodology

5.1 Proposed methods to be applied

1. Data cleaning and preprocessing: This involves handling missing values, dealing with outliers, converting categorical variables into numerical, and scaling the data to prepare it for model training.

2. Exploratory Data Analysis (EDA): This involves analyzing and visualizing the data to understand the patterns, relationships, and distributions of the features. EDA can help in feature selection and engineering.

3. Feature engineering: This involves creating new features from the existing ones to improve the predictive power of the model. This can be done using techniques such as polynomial features, feature scaling, and feature selection.

4. Model training: This involves training different machine learning algorithms on the cleaned and preprocessed data. The models that will be trained include Decision Tree, Random Forest, Logistic Regression, K-Nearest Neighbors, and Support Vector Machines.

5. Model evaluation: This involves evaluating the performance of the trained models using metrics such as accuracy, precision, recall, F1 score, and ROC-AUC curve. The best-performing model will be selected for making loan prediction.

6. Hyperparameter tuning: This involves optimizing the hyperparameters of the selected model using techniques such as grid search and randomized search to improve its performance.

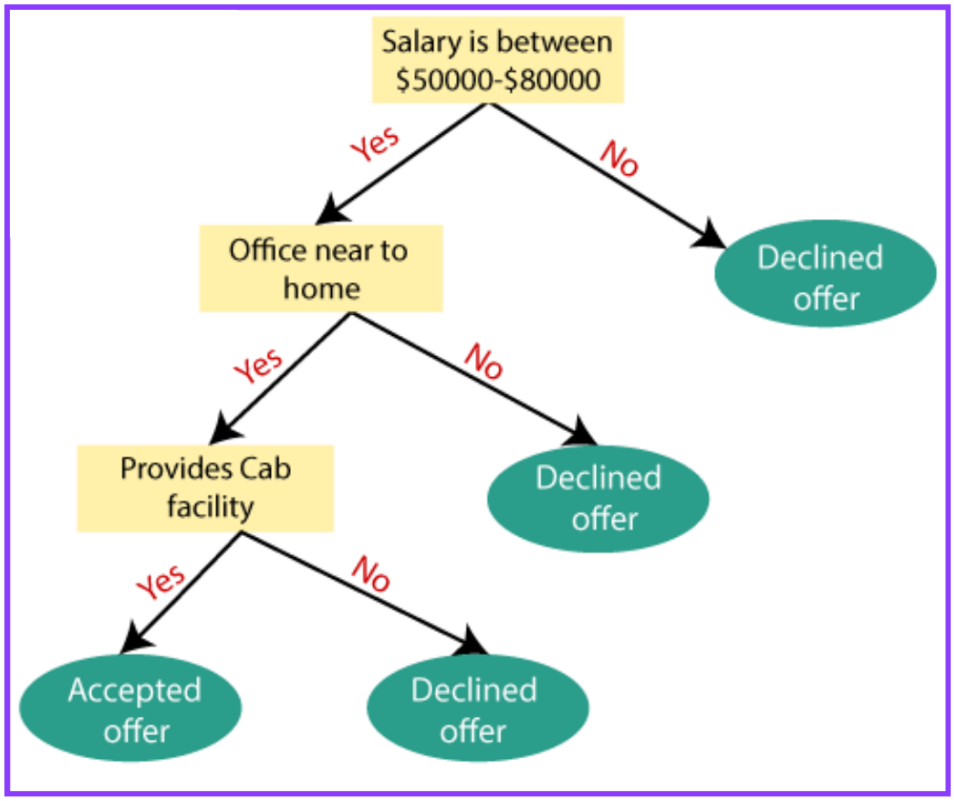
7. Deployment: This involves deploying the final model to make loan predictions on new data.

5.2 Software used for data analysis and ML model

Google Colab is a cloud-based development environment that allows users to write and run Python code in a Jupyter notebook-style interface. It provides access to a free GPU and TPU for deep learning and other computationally intensive tasks. Colab integrates with Google Drive for easy data storage and sharing. It also includes pre-installed libraries such as TensorFlow, PyTorch, and scikit-learn for machine learning tasks. Colab is widely used for research, education, and collaboration in the machine learning community.

Decision tree is a popular machine learning algorithm for classification and regression tasks. It is used in the loan prediction model to analyze the data and create a model that can predict whether a loan will be approved or not based on the input variables. Decision tree is a powerful algorithm for handling categorical variables and can handle non-linear relationships between variables. It also provides transparency in the decision-making process as the decision rules can be easily understood and interpreted. Additionally, decision trees can handle missing data and outliers, making it a robust algorithm for data analysis.

An illustration of a decision tree model is provided below.Imagine an applicant who has received a job offer and is debating whether to accept it or not. Therefore, the decision tree begins at the root node to tackle this issue. (Salary attribute by ASM). Based on the corresponding labels, the root node further divides into the next decision node (distance from the office) and one leaf node. The following decision node is further divided into a leaf node and a decision node (Cab facility). The decision node finally separates into two leaf nodes. (Accepted offers and Declined offer)



5.3 Python libraries used

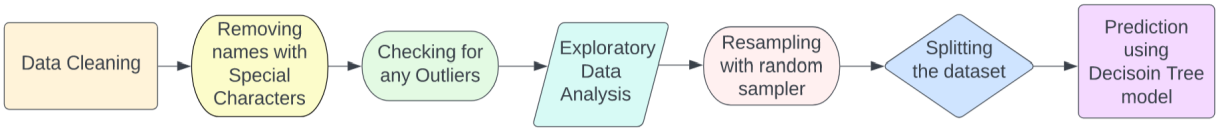
1. NumPy is used for efficient numerical computing,
2. Pyplot is used for creating visualizations,
3. Seaborn is used for advanced statistical visualizations,
4. Scikit-learn is used for machine learning tasks, and
5. Pandas is used for data manipulation and analysis.

These libraries are widely used in scientific computing, data analysis, and machine learning applications in Python.

5.4 Algorithm

1. Import required libraries such as pandas, numpy, matplotlib, seaborn, and scikit-learn.
2. Load the loan data into a pandas dataframe using the read_csv function.
3. Perform exploratory data analysis (EDA) to understand the data distribution and correlation between variables. This includes checking for missing values, visualizing data using plots, and performing statistical tests to identify any significant differences between groups.
4. Preprocess the data by converting categorical variables into numerical values using one-hot encoding, filling missing values, and scaling the numerical features.
5. Split the data into training and test sets using the train_test_split function from scikit-learn.
6. Train a decision tree classifier on the training data using the fit method from scikit-learn's DecisionTreeClassifier class.
7. Evaluate the model's performance on the test data by computing various metrics such as accuracy, precision, recall, and F1 score using the classification_report function from scikit-learn.
8. Tune the model's hyperparameters using grid search cross-validation with the GridSearchCV function from scikit-learn to find the optimal combination of hyperparameters that maximizes the model's performance.
9. Train the final model with the optimal hyperparameters on the entire dataset and save it to disk using the pickle library.
10. Make predictions on new loan applications using the predict method from the trained decision tree classifier.

5.5 Program flow:



5.6 Code Explanation:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

- numpy as np: a popular library for scientific computing in Python that provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- pandas as pd: a library for data manipulation and analysis in Python that provides easy-to-use data structures and data analysis tools for handling tabular data.
- matplotlib.pyplot as plt: a plotting library in Python that provides a variety of tools for creating visualizations, including line plots, scatter plots, bar charts, and more. seaborn as sns: a library for statistical data visualization in Python that provides a high-level interface for creating informative and attractive visualizations.

```
[ ]  df = pd.read_csv('/content/drive/MyDrive/Training Data.csv')
```

- This is a code section that reads a CSV file named 'Training Data.csv' located in the '/content/drive/MyDrive/' directory using the pandas library's read_csv() function. It assigns the resulting DataFrame object to a variable named df. The read_csv() function is used to read CSV files and convert them to DataFrame objects, which can then be manipulated and analyzed using various pandas functions.

```
df
```

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Rewa	Madhya_Pradesh	3	13	0
1	2	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra	9	13	0
2	3	3991815	66	4	married	rented	no	Technical_writer	Alappuzha	Kerala	4	10	0
3	4	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar	Odisha	2	12	1
4	5	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli[10]	Tamil_Nadu	3	14	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
251995	251996	8154883	43	13	single	rented	no	Surgeon	Kolkata	West_Bengal	6	11	0
251996	251997	2843572	26	10	single	rented	no	Army_officer	Rewa	Madhya_Pradesh	6	11	0
251997	251998	4522448	46	7	single	rented	no	Design_Engineer	Kalyan-Dombivli	Maharashtra	7	12	0
251998	251999	6507128	45	0	single	rented	no	Graphic_Designer	Pondicherry	Puducherry	0	10	0
251999	252000	9070230	70	17	single	rented	no	Statistician	Avadi	Tamil_Nadu	7	11	0

252000 rows × 13 columns

- This code is calling a variable df in Python. Since df was assigned a value using the pd.read_csv() function in the previous code, running this code will print the contents of the df variable in the output.

5.6.1 Data Cleaning:

```
[ ]  def unclean_names(col):
         unclean_names = []
         for name in df[str(col)].unique():
             if name.endswith(']'):
                 unclean_names.append(name)
         return unclean_names
```

- This code section defines a function called unclean_names() that takes a single argument col. The function starts by creating an empty list called unclean_names.
- Then it loops through each unique value in the specified column (col) of the df DataFrame using a for loop. For each unique value in the column, the code checks if the value ends with a closing square bracket (']'). If it does, the code appends the value to the unclean_names list.
- Finally, the function returns the unclean_names list containing all the unique values in the specified column that end with a closing square bracket.

```
unclean_city_names = unclean_names('CITY')
unclean_city_names
```

```
['Tiruchirappalli[10]',
 'Kota[6]',
 'Hajipur[31]',
 'Erode[17]',
 'Anantapuram[24]',
 'Aurangabad[39]',
 'Purnia[26]',
 'Eluru[25]',
 'Siwan[32]',
 'Motihari[34]',
 'Warangal[11][12]',
 'Jehanabad[38]',
 'Kishanganj[35]',
 'Tirupati[21][22]',
 'Kurnool[18]',
 'Kadapa[23]',
 'Jammu[16]',
 'Rajahmundry[19][20]',
 'Saharsa[29]',
 'Jamalpur[36]',
 'Dehri[30]',
 'Nellore[14][15]',
 'Visakhapatnam[4]',
 'Buxar[37]',
 'Ramagundam[27]',
 'Sasaram[30]',
 'Guntur[13]',
 'Chittoor[28]',
 'Bettiah[33]',
 'Mysore[7][8][9]']
```

- This code calls the previously defined unclean_names() function with the argument 'CITY', and assigns the returned list of unclean city names to a variable called unclean_city_names.
- Assuming that the unclean_names() function is defined correctly and that the df DataFrame contains a column called 'CITY', this code will return a list of all unique values in the 'CITY' column of the df DataFrame that end with a closing square bracket (']'). By printing the unclean_city_names variable, this code will display the list of unclean city names in the output.

```
[ ] unclean_state_names = unclean_names('STATE')
    unclean_state_names
```

```
['Uttar_Pradesh[5]']
```

- This code will return a list of all unique values in the 'STATE' column of the df DataFrame that end with a closing square bracket (']'). By printing the

unclean_state_names variable, this code will display the list of unclean state names in the output.

```
def clean_df(df,col,unclean_list):
    for index,name in enumerate(df[col]):
        if name in unclean_list:
            if name.endswith(']'):
                name_ = name.strip('[]0123456789')
                df[col].iloc[index] = name_
```

- The code defines a function called clean_df() that takes three arguments: a DataFrame df, a column name col, and a list of unclean names unclean_list.
- The function loops through each row of the specified column (col) in the df DataFrame using a for loop and the enumerate() function.
- For each row, the code checks if the value in the row matches one of the unclean names in the unclean_list using the in operator.
- If the value in the row does match one of the unclean names in the unclean_list, the code checks if the value ends with a closing square bracket (']'). If it does, the code removes the square brackets and any digits from the end of the value using the strip() method, and assigns the resulting string to a variable called name_. Finally, the code updates the value in the DataFrame at the specified row and column index (df[col].iloc[index]) to the cleaned name (name_).

```
[9] clean_df(df,'STATE',unclean_state_names)

<ipython-input-8-7b4848f19cda>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col].iloc[index] = name_

clean_df(df,'CITY',unclean_city_names)

<ipython-input-8-7b4848f19cda>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col].iloc[index] = name_
```

- The function is expected to modify the STATE column of the input DataFrame df in place by removing any digits and square brackets from the end of the unclean names in the unclean_state_names list. If the function works correctly, the cleaned DataFrame will be returned.
- Similarly for the CITY column also the function is passed.

5.6.2 Checking for any Outliers:

```
df['Age'].plot(kind='hist',figsize=(10,8))
plt.xlabel('Age')
```

- The code generates a histogram plot of the 'Age' column in the DataFrame df. The plot is created using the plot() method of the 'Age' column in the DataFrame, with the argument kind='hist' indicating that a histogram plot should be generated. The figsize argument sets the size of the plot figure to be 10 inches wide by 8 inches tall. The plt.xlabel() function adds a label to the x-axis of the plot. In this case, the label is set to 'Age'.

```
df['Income'].plot(kind='box')
```

```
df['Experience'].plot(kind='box')
```

```
df['CURRENT_HOUSE_YRS'].plot(kind='box')
```

```
df['Income'].plot(kind='hist')
```

```
df['CURRENT_JOB_YRS'].plot(kind='box')
```

- The first line of code generates a box plot of the 'Income' column in the DataFrame df. The plot is created using the plot() method of the 'Income' column in the DataFrame, with the argument kind='box' indicating that a box plot should be generated.
- Similarly various other columns are plotted using the boxplot function.

Each plot may be useful for different purposes:

- The box plot of 'Income' shows the distribution of income values, including any outliers and the median value.
- The histogram plot of 'Income' shows the frequency distribution of income values, which can be useful for understanding the shape of the distribution.
- The box plot of 'CURRENT_JOB_YRS' shows the distribution of the number of years the customer has been in their current job.
- The box plot of 'CURRENT_HOUSE_YRS' shows the distribution of the number of years the customer has lived in their current house.
- The box plot of 'Experience' shows the distribution of the total number of years of experience of the customer.

5.6.3 Exploratory Data Analysis:

```
fig = plt.figure()
ax1 = fig.add_subplot(2,1,1,anchor='C')
plt.title('Information')
df.groupby('Risk_Flag').count()['Id'].plot(kind='pie',labels=['Non-Defaulter','Defaulter'],autopct='%1.1f%%',ax=ax1,figsize=(10,10))
plt.xlabel('% of Defaulters')
plt.ylabel('')
plt.legend(loc='right',bbox_to_anchor=(0.7,0,1,1))
ax2 = fig.add_subplot(2,1,2,anchor='S')
df.groupby('Risk_Flag').count()['Id'].plot(kind='bar',ax=ax2)
plt.xlabel('Defaulters')
plt.ylabel('Count')
for index,value in enumerate(df.groupby('Risk_Flag').count()['Id']):
    plt.text(index-0.08,value+10000,str(value))
plt.ylim(0,250000)
plt.show()
```

- This code creates a figure with two subplots. The first subplot is a pie chart that displays the percentage of defaulters and non-defaulters in the 'Risk_Flag' column of the df DataFrame.
- The second subplot is a bar chart that shows the count of defaulters and non-defaulters in the 'Risk_Flag' column of the df DataFrame. The x-axis label for the bar chart is set to 'Defaulters' and the y-axis label is set to 'Count'. The text() method is used to display the count for each bar above the bar. The ylim() method is used to set the y-axis limits to (0,250000).
- The legend is set outside the plot area to the right using the legend() method with the loc and bbox_to_anchor parameters. Finally, the show() method is called to display the figure.

```
df.groupby('Married/Single').count()['Id'].plot(kind='pie',startangle=0,labels=['Married','Single'],autopct='%1.1f%%',colors=['Pink','Teal'])
plt.ylabel('')
plt.xlabel('Marital Status')
plt.title('Total % of Customers who are married/single')
plt.legend(loc='best',bbox_to_anchor=(1,0,0.5,0.5))
plt.show()
```

- This code creates a pie chart that shows the percentage of customers who are married or single. It groups the DataFrame by the 'Married/Single' column and counts the number of occurrences of each category. The resulting series is plotted as a pie chart with labels, colors, and a legend. The ylabel(), xlabel(), and title() methods are used to customize the plot, and the show() method is called to display it.

```
[19] df.loc[df['Risk_Flag'] == 1].groupby('Married/Single').count()['Id']
```

```
Married/Single
married     2636
single     28360
Name: Id, dtype: int64
```

```
marital_status = df.loc[df['Risk_Flag'] == 1].groupby('Married/Single').count()['Id']
marital_status.plot(kind='pie',startangle=0,labels=['Married','Single'],autopct='%1.1f%%')
plt.ylabel('')
plt.xlabel('Marital Status')
plt.title('Loan Defaulters % by Marital Status')
plt.legend(loc='best',bbox_to_anchor=(1,0,0.5,0.5))
plt.show()
```

- This code creates a pie chart to display the percentage of loan defaulters in each marital status category. It first filters the original DataFrame to include only rows where the 'Risk_Flag' column is equal to 1. Then, the count of defaulters in each marital status category is calculated using the groupby() and count() methods.
- A pie chart is created using the plot() method with kind='pie', and the percentage of defaulters in each category is displayed using the autopct parameter. The title of the chart is set, the x-axis label is added, and a legend is placed outside the plot area to the right using the legend() method with the loc and bbox_to_anchor parameters. Finally, the show() method is called to display the figure.

```
[21] house_ownership_count = df.groupby('House_Ownership').count()['Id']
     house_ownership_count = [231898,7184,12918]
```

```
sns.countplot(data=df,x='House_Ownership',hue='House_Ownership',)
plt.text(-0.4,235000,str(231898))
plt.text(0.9,10000,str(7184))
plt.text(2.14,16000,str(12918))
plt.title('House Ownership of All Customers')
plt.ylim(0,250000)
```

```
[23] df.loc[df['Risk_Flag'] == 1].groupby('House_Ownership').count()['Id']

House_Ownership
norent_noown      715
owned            1160
rented          29121
Name: Id, dtype: int64
```

```
sns.countplot(data=df.loc[df['Risk_Flag'] == 1],x='House_Ownership',hue='House_Ownership')
plt.text(-0.4,30000,str(29121))
plt.text(0.9,2000,str(1160))
plt.text(2.14,1800,str(715))
plt.ylim(0,35000)
plt.title('House Ownership of Loan Defaulting Customers')
plt.show()
```

- This code section creates two countplots to show the count of customers and loan defaulters for each category of house ownership. The count of each unique value in the 'House_Ownership' column of the DataFrame is calculated and a countplot is created using the seaborn library. The text() method is used to display the count of each value above the corresponding bar. The title of the plot is set using the title() method and the ylim() method is used to set the y-axis limits to (0,250000). The same process is repeated to show the count of loan defaulters for each category of house ownership

```
df.groupby('House_Ownership').count()['Id'].plot(kind='pie',startangle=0,autopct='%1.1f%%',figsize=(5,5))
plt.ylabel('')
plt.xlabel('House Ownership')
plt.title('House Ownership % of Customers')
plt.legend(loc='best',bbox_to_anchor=(1,0,0.5,0.5))
```

```
df.loc[df['Risk_Flag'] == 1].groupby('House_Ownership').count()['Id'].plot(kind='pie',startangle=0,autopct='%1.1f%%',figsize=(5,5))
plt.ylabel('')
plt.title('House Ownership % of Loan Defaulters')
plt.xlabel('House Ownership')
plt.legend(loc='best',bbox_to_anchor=(1,0,0.5,0.5))
```

- This code section creates two pie charts to show the percentage of customers and loan defaulters in each category of house ownership. The 'House_Ownership' column is used to group the DataFrame, and the count of customers or loan defaulters in each category is used to create the pie chart. The title of each chart is set, and the legend is placed outside the plot area to the right using the legend() method with the loc parameter set to 'best' and the bbox_to_anchor parameter set to (1,0,0.5,0.5).

```
df1 = df.loc[df['Risk_Flag'] == 1].groupby(['STATE','Risk_Flag']).count()
df1.rename(columns={'Id':'Total_Defaulters'},inplace=True)
df1.reset_index(inplace=True)
df1[['STATE','Total_Defaulters']]
```

	STATE	Total_Defaulters
0	Andhra_Pradesh	2935
1	Assam	930
2	Bihar	2583
3	Chandigarh	61
4	Chhattisgarh	511

- This code filters the DataFrame df to only include rows where the 'Risk_Flag' column is equal to 1 (indicating loan defaulters). It then groups the resulting DataFrame by the 'STATE' and 'Risk_Flag' columns and applies the count() method to each group to count the number of rows in each group. The resulting DataFrame is assigned to the variable df1.
- The code then renames the 'Id' column in df1 to 'Total_Defaulters' using the rename() method, and sets the 'STATE' and 'Risk_Flag' columns as regular columns using the reset_index() method. Finally, the code selects the 'STATE' and 'Total_Defaulters' columns of df1 and displays them using the double bracket notation (df1[['STATE','Total_Defaulters']]).

```
df_total_loans = df2[['STATE','Total_Loans']].sort_values(by='Total_Loans',ascending=False)[:10]
df_total_loans.plot(kind='bar',x='STATE',figsize=(10,8))
plt.title('Top 10 States from where most loans were taken')
plt.xlabel('Number of Loans')
plt.ylabel('State')
for index,value in enumerate(df_total_loans['Total_Loans'][:10]):
    plt.text(index-0.28,value+100,str(value))
plt.show()
```

```
defaulter_percent_per_state = (df1['Total_Defaulters']/df2['Total_Loans']).round(4)*100
state_defaulters_percentage=pd.DataFrame(
    data=zip(df1['STATE'],defaulter_percent_per_state),
    columns=['STATE','Defaulter_Percentage']
)
df_dps = state_defaulters_percentage.sort_values(by='Defaulter_Percentage',ascending=False)[:10]
df_dps.plot(kind='bar',figsize=(10,8),x='STATE')
plt.title('Top 10 States in Defaulting Loan')
plt.ylabel('% of Loans Defaults')
plt.xlabel('State')
for index,value in enumerate(df_dps['Defaulter_Percentage'][:10]):
    plt.text(index-0.2,value+0.2,str(round(value,2)))
plt.legend(loc='best')
plt.show()
```

- This block of code creates a dataframe df_total_loans that contains the top 10 states from where most loans were taken, and then plots a horizontal bar graph to visualize the data.
- The second block of code calculates the percentage of defaulters in each state by dividing the number of total defaulters by the number of total loans for each state, and then creates a new dataframe df_dps that contains the top 10 states with the highest percentage of loan defaults. It then plots a vertical bar graph to visualize the data.

```
[36] #top10 cities in number of loans

     df3_ = df3[['CITY','Total_Loans']].sort_values(
                            by='Total_Loans',ascending=False)[:10]
     df3_.plot(kind='bar',x='CITY',figsize=(8,6))
     plt.title('Top 10 Cities with highest number of Loans taken')
     plt.xlabel('City')
     plt.ylabel('Number of Loan')
     for index,value in enumerate(df3_['Total_Loans']):
         plt.text(index-0.25,value+30,str(int(value)))
     plt.legend(loc='best')
     plt.show()
```

```
df4 = df.loc[df['Risk_Flag'] == 1].groupby('CITY').count()
df4.rename(columns={'Id':'Total_Defaulters'},inplace=True)
df4.reset_index(inplace=True)
df4[['CITY','Total_Defaulters']]
```

```
defaulter_percent_per_city = (df4['Total_Defaulters']/df3['Total_Loans']).round(4)*100
city_defaulters_percentage=pd.DataFrame(
    data=zip(df3['CITY'],defaulter_percent_per_city),
    columns=['CITY','Defaulter_Percentage']
)
city_defaulters_percentage
```

- The code is performing an analysis of loan data based on cities. It first creates a DataFrame (df3) by grouping the original DataFrame (df) by city and counting the number of loans taken in each city. It then selects the top 10 cities with the highest number of loans (df3_) and creates a bar plot to visualise the data.
- Next, it creates another DataFrame (df4) by filtering the original DataFrame (df) for defaulted loans (Risk_Flag == 1) and grouping the resulting DataFrame by city and counting the number of defaulted loans in each city. It then calculates the percentage of defaulted loans for each city (defaulter_percent_per_city) and creates another DataFrame (city_defaulters_percentage) to store this data along with the corresponding city names.

- This code will help in identifying the cities with the highest number of loans and the cities with the highest percentage of defaulted loans, which can be used to inform loan policies and risk assessments for different cities.

```
city_defaulters_percentage.sort_values(by='Defaulter_Percentage',ascending=False)[:10].plot(kind='bar',x='CITY',figsize=(10,6))
plt.title('Top 10 Cities in Defaulting Loans')
plt.xlabel('City')
plt.ylabel('% of Loan Defaults')
plt.legend(loc='best')
top_10_vals = city_defaulters_percentage['Defaulter_Percentage'].sort_values(ascending=False)[:10]
for index,value in enumerate(top_10_vals):
    plt.text(index-0.2,value+0.5,str(round(value,2)))
plt.show()
```

```
df_profession_loan_count = df.groupby('Profession').count()['Id'].sort_values(ascending=False)
df_plc = df_profession_loan_count.reset_index()
df_plc.rename(columns= {'Id':'Loan_Count'},inplace=True)
df_plc[:10].plot(kind='bar',x='Profession',figsize=(10,10))
plt.legend(loc='best')
plt.title('Top 10 Professions who took Loan')
plt.xlabel('Loan Count')
plt.ylabel('Profession')
for index,value in enumerate(df_plc['Loan_Count'][:10]):
    plt.text(index-0.2,value+50,str(value))
plt.show()
```

- The code is analyzing a dataset containing information on loans taken by individuals, with a focus on identifying patterns in loan defaults across cities and professions. The code first identifies the top 10 cities and professions with the highest number of loans taken and plots bar graphs to visualize the results. It then calculates the percentage of loan defaults for each city and identifies the top 10 cities with the highest percentage of loan defaults, plotting a bar graph to visualize these results as well. Overall, the code provides insights into the loan patterns of individuals and highlights areas where loan defaults are more likely to occur.

```
#plotting top10 profession_group with higher income

profession_top10_income = profession['Income'].sort_values(ascending=False)[:15]
profession_top10_income.plot(kind='barh',figsize=(10,10))
plt.title('Top 15 Profession with higher Income (mean)')
plt.xlabel('Profession')
plt.ylabel('Income')

for index,value in enumerate(profession_top10_income):
    plt.text(value-900000,index-0.1,str(int(value)))
plt.legend(loc='best')
plt.show()
```

```
df_ = df.loc[df['Risk_Flag'] == 1].groupby(['Profession']).mean()[['Income']].sort_values(by='Income',ascending=False)
df_.sort_values(by='Income',ascending=False)[:15].plot(kind='barh',figsize=(10,10))
plt.title('Mean Income of Top 15 Loan Defaulting Professions')
plt.xlabel('Income')
plt.ylabel('Profession')
for index,value in enumerate(df_['Income'][:15]):
    plt.text(value-900000,index-0.1,str(int(value)))

plt.legend(loc='best')
plt.show()
```

- This code generates two horizontal bar graphs that display the top 15 professions with the highest mean income. The first graph shows the top 15 professions with the highest mean income overall, while the second graph shows the mean income of the top 15 professions with the highest percentage of loan defaults.
- The DataFrame is grouped by profession, and the mean income is calculated for each profession. For each graph, a horizontal bar graph is created using plot(kind='barh'). The income value is displayed using the plt.text() method for each bar in the graph. The legend() method is used to add a legend to the plot, and the loc parameter is set to 'best' to place the legend at the optimal location. Finally, the show() method is called to display the plot.

5.6.4 Resampling the Data with Random Oversampler:

What is the use of Random Oversampler?

RandomOverSampler is a technique used in machine learning to address the issue of imbalanced datasets. In an imbalanced dataset, one class of data is significantly underrepresented in comparison to another class. This can lead to biased machine learning models that perform poorly on the underrepresented class.

```
[44] from imblearn.over_sampling import RandomOverSampler

[45] sampler = RandomOverSampler(random_state=42,sampling_strategy=0.45)
     X = df.iloc[:,:-1]
     y = df['Risk_Flag']

[46] X_sampled,y_sampled = sampler.fit_resample(X,y)

     from collections import Counter
     print(Counter(y),Counter(y_sampled))

     Counter({0: 221004, 1: 30996}) Counter({0: 221004, 1: 99451})

[48] df_ = pd.concat([X_sampled,y_sampled],axis=1)
```

- This code uses the RandomOverSampler class from the imblearn package to oversample the minority class in the target variable 'y', to address class imbalance. The Counter class from the collections package is used to count the number of occurrences of each class label in both the original and oversampled datasets. The oversampled X and y datasets are then concatenated into a new dataframe df.

```
fig = plt.figure()
ax1 = fig.add_subplot(121)
plt.title('Defaulter % Before Sampling')
df.groupby('Risk_Flag').count()['Id'].plot(kind='pie',labels=['Non-Defaulter','Defaulter'],autopct='%1.1f%%',ax=ax1,figsize=(10,10))
plt.xlabel('% of Defaulters')
plt.ylabel('')

ax2 = fig.add_subplot(122)
plt.title('Defaulter % After Sampling')
df_.groupby('Risk_Flag').count()['Id'].plot(kind='pie',labels=['Non-Defaulter','Defaulter'],autopct='%1.1f%%',ax=ax2,figsize=(10,10))
plt.xlabel('% of Defaulters')
plt.ylabel('')
plt.legend(loc='right',bbox_to_anchor=(0.7,0,1,1))
```

- This code section creates a figure with two subplots, each showing the percentage of defaulters and non-defaulters before and after oversampling using a pie chart. The first subplot shows the original percentage of defaulters in the 'Risk_Flag' column of the df DataFrame, and the second subplot shows the percentage of defaulters after applying the RandomOverSampler to balance the class distribution.
- The title of each subplot is set using the title() method, and the x-axis label is set to '% of Defaulters' using the xlabel() method. The legend for the pie charts is set using the legend() method with the loc and bbox_to_anchor parameters to move it outside the plot area to the right.

5.6.5 Encoding the Categorical data:

Purpose:

Encoding categorical data means converting categorical variables into numerical values that can be used in machine learning algorithms. Most machine learning algorithms are based on mathematical equations and require numerical data as input. Categorical data, such as gender or occupation, cannot be directly used as input in these algorithms.

```
[ ] from sklearn.preprocessing import LabelEncoder
```

```
cols_to_encode= ['Married/Single','House_Ownership','Car_Ownership','Profession','CITY','STATE']
labelencoder = LabelEncoder()

for col in cols_to_encode:
    df_[col] = labelencoder.fit_transform(df_[col])
```

- This code section uses the LabelEncoder class from the sklearn.preprocessing module to encode categorical data in the df_ dataframe. The columns to be encoded are specified in the 'cols_to_encode' list. The code then loops through each column in the list and applies the label encoder to transform the categorical data into numerical values.
- The resulting encoded values are then stored in the corresponding column of the df_ dataframe. This process is necessary because many machine learning algorithms can only handle numerical input data, and encoding categorical data is one way to convert it to numerical form.

```
[ ]  df_.isna().sum()

Id                 0
Income             0
Age                0
Experience         0
Married/Single     0
House_Ownership    0
Car_Ownership      0
Profession         0
CITY               0
STATE              0
CURRENT_JOB_YRS    0
CURRENT_HOUSE_YRS  0
Risk_Flag          0
dtype: int64
```

```
#Dropping Id as it's not needed in prediction
df_.drop(['Id'],axis=1,inplace=True)
```

```
X = df_.iloc[:,:-1]
y = df_['Risk_Flag']
```

- The first line of code checks for missing values in the df_ DataFrame and returns the sum of missing values for each column using the isna() and sum() methods. The second line of code drops the 'Id' column from the df_ DataFrame using the drop() method with axis=1 parameter.
- The third and fourth lines of code separate the feature matrix X and target variable y from the df_ DataFrame using the iloc[] method to select all rows and all columns except the last column for X, and the last column for y.
- Overall, these lines of code are preparing the dataset for machine learning by handling missing values, dropping unnecessary columns, and separating the feature matrix and target variable.

5.6.6 Splitting the dataset into training and test set:

```
116] from sklearn. model_selection import train_test_split
     import matplotlib.pyplot as plot

     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=42)
```

- This code section imports the train_test_split function from the sklearn.model_selection module, which is used to split the dataset into training and testing sets. It also imports the pyplot module from the matplotlib library, which is used to create visualizations.
- The code then calls the train_test_split function and passes the features X and target variable y along with the test_size parameter set to 0.3, which splits the dataset into 70% for training and 30% for testing.

- The random_state parameter is set to 42 to ensure reproducibility of the results. Finally, the code assigns the training and testing data to X_train, X_test, y_train, and y_test variables, respectively.

5.6.7 Prediction:

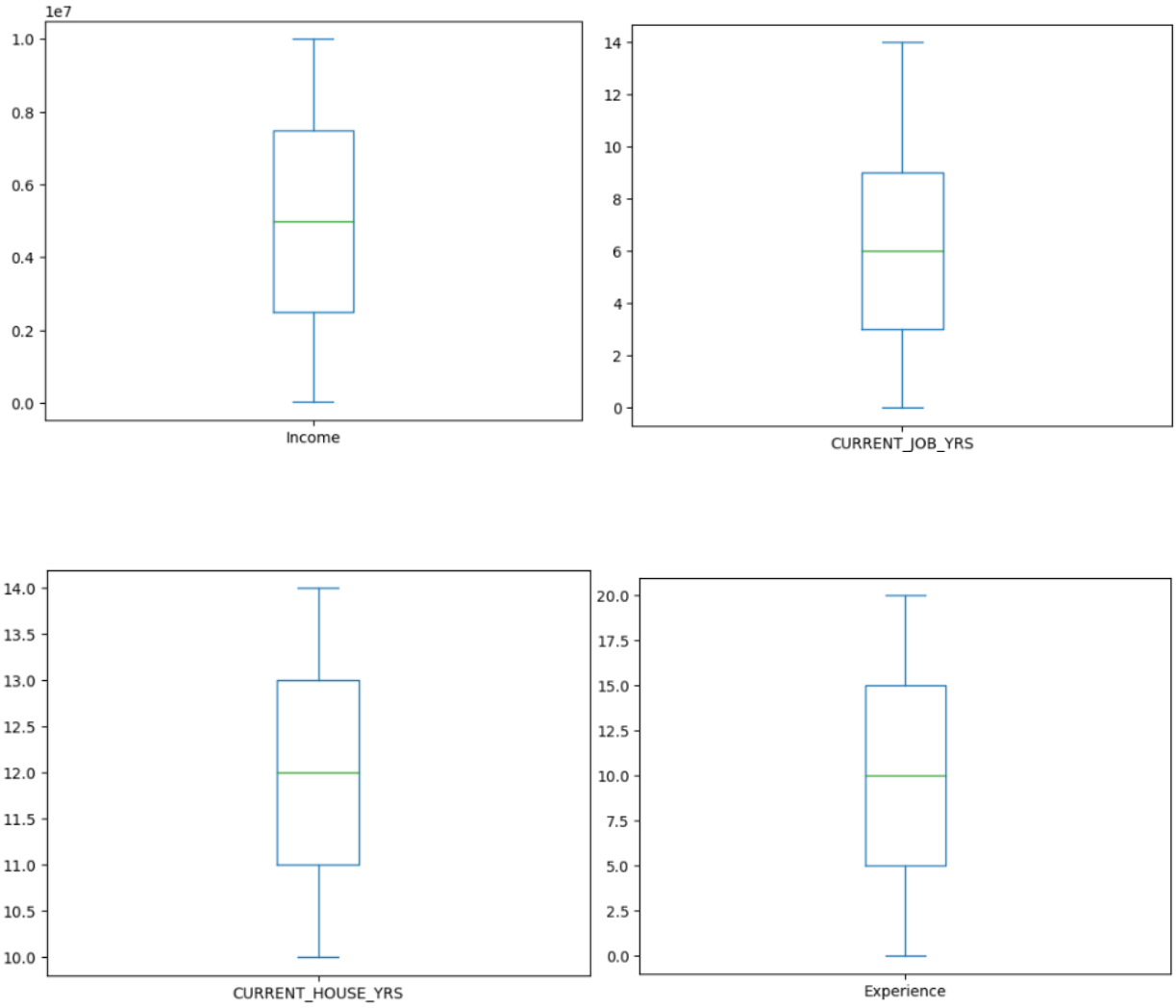
```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(X_train,y_train)
y_pred = dt.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(f'F1 Score: {f1_score(y_test,y_pred)}\n')
print(classification_report(y_test,y_pred))
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Class 0', 'Class 1'])
disp.plot()
```

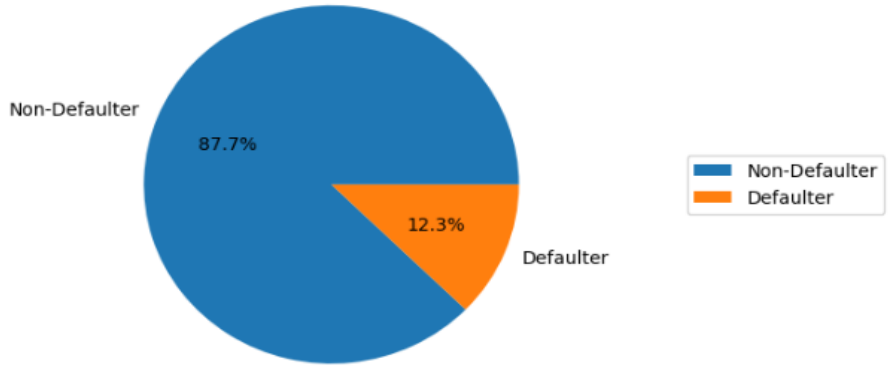
- This code section performs decision tree classification on the training data and generates a confusion matrix and classification report for the test data. It first imports the DecisionTreeClassifier class from the sklearn.tree module.
- Then, an instance of this class is created and fit to the training data using the fit() method. Next, the model is used to predict the class labels for the test data using the predict() method, and a confusion matrix is generated using the confusion_matrix() function from the sklearn.metrics module.
- The F1 score and classification report for the model's predictions are printed to the console using the f1_score() and classification_report() functions from the sklearn.metrics module, respectively.
- Finally, a ConfusionMatrixDisplay object is created using the confusion matrix, and the plot() method is used to display the confusion matrix with class labels 'Class 0' and 'Class 1'.

6. Result & Data Analysis

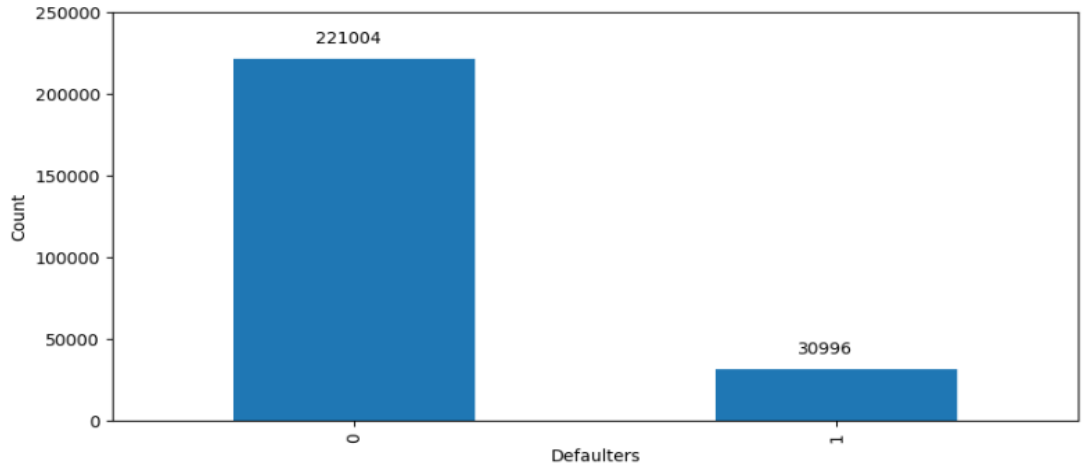
6.1 Checking for any Outliers



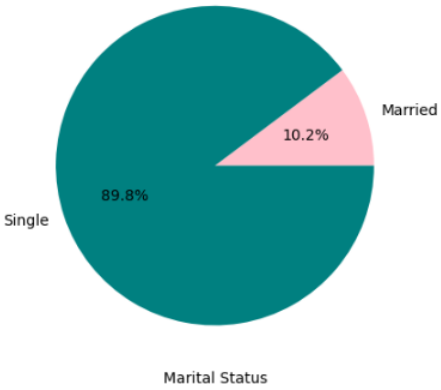
6.2 Exploratory Data Analysis



% of Defaulters



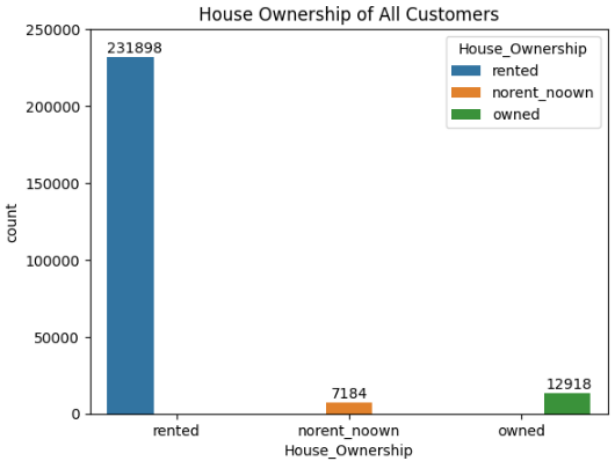
Total % of Customers who are married/single

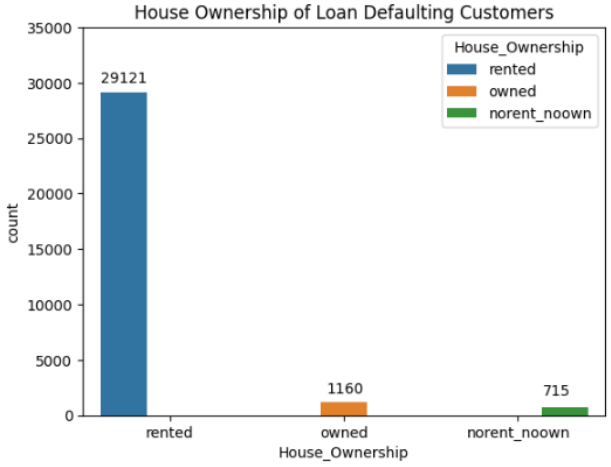


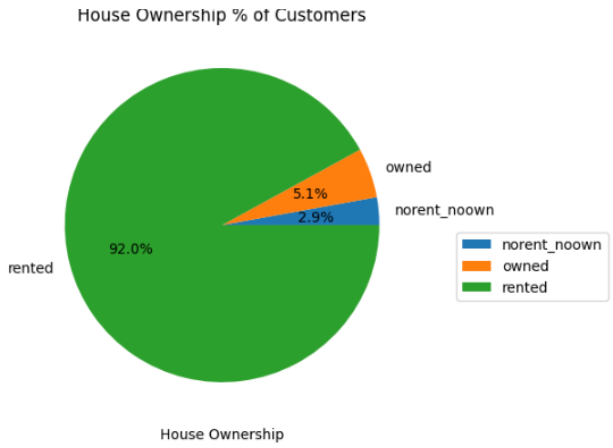
Marital Status

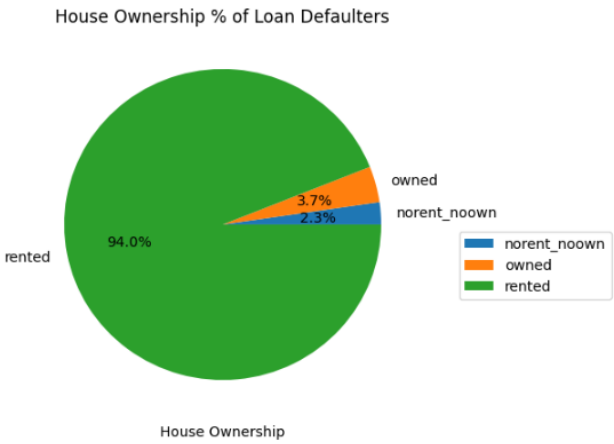
Loan Defaulters % by Marital Status

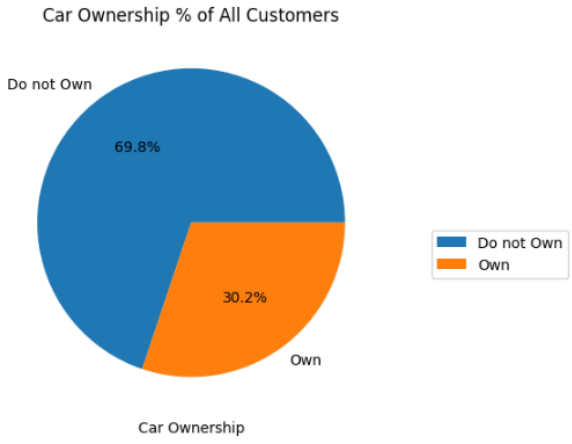
Marital Status

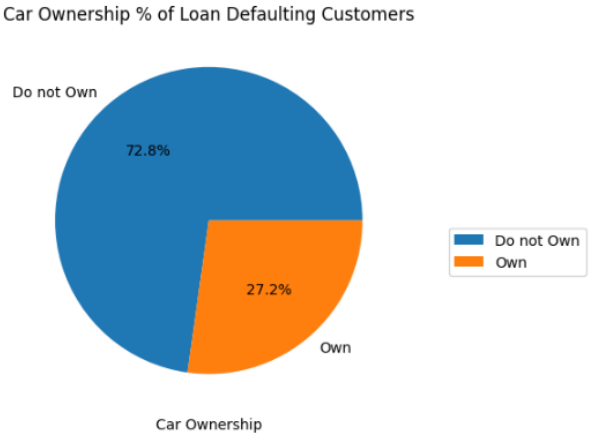

House Ownership of All Customers
250000
200000
150000
100000
50000
0
count
231898
7184
12918
House_Ownership
rented
norent_noown
owned
rented
norent_noown
owned
House_Ownership

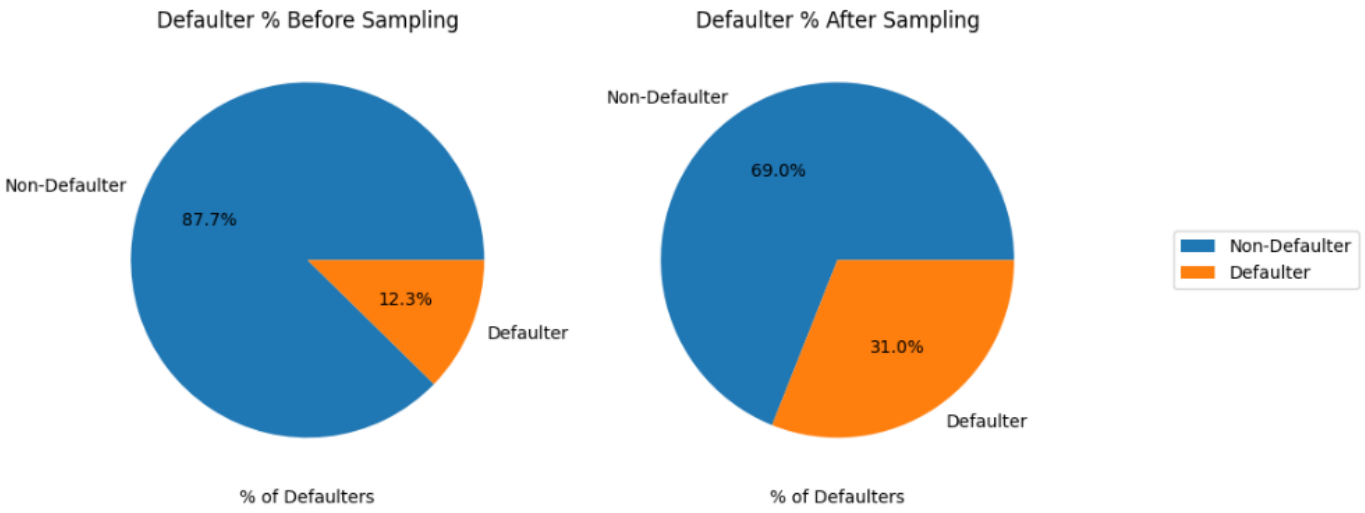

House Ownership of Loan Defaulting Customers
35000
30000
25000
20000
15000
10000
5000
0
count
29121
1160
715
House_Ownership
rented
owned
norent_noown
rented
owned
norent_noown
House_Ownership


House Ownership % of Customers
owned
5.1%
norent_noown
2.9%
92.0%
rented
norent_noown
owned
rented
House Ownership

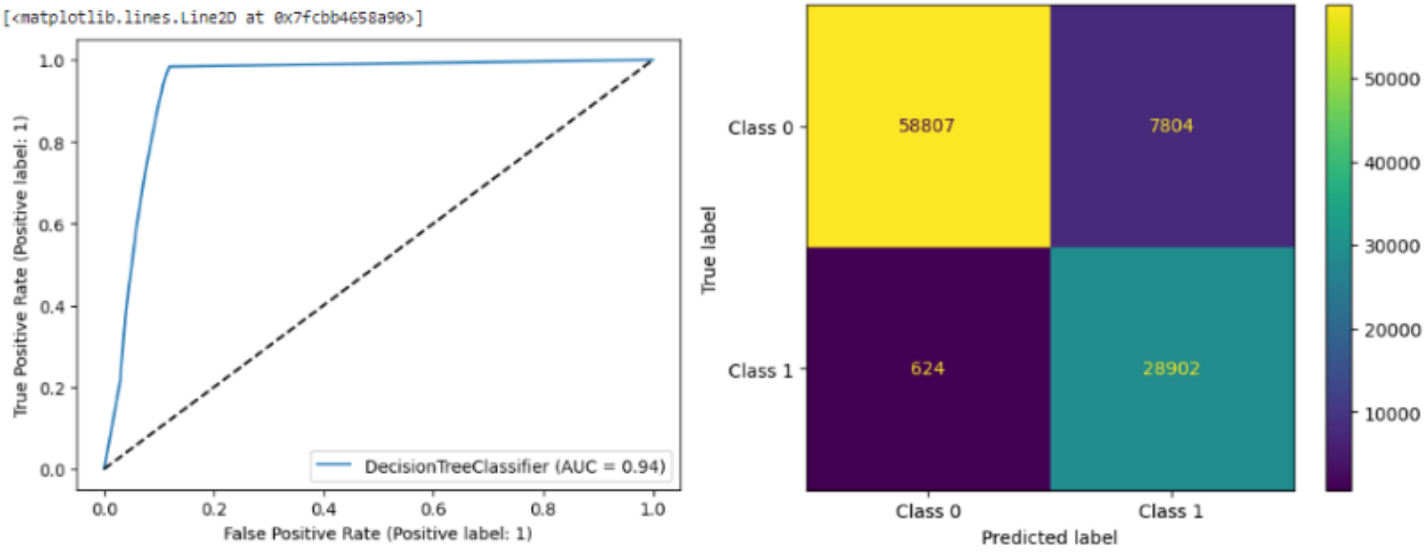

House Ownership % of Loan Defaulters
owned
3.7%
norent_noown
2.3%
94.0%
rented
norent_noown
owned
rented
House Ownership


Car Ownership % of All Customers
Do not Own
69.8%
30.2%
Own
Do not Own
Own
Car Ownership


Car Ownership % of Loan Defaulting Customers
Do not Own
72.8%
27.2%
Own
Do not Own
Own
Car Ownership



6.3 Prediction



F1 Score: 0.8721084737573231

	precision	recall	f1-score	support
0	0.99	0.88	0.93	66611
1	0.79	0.98	0.87	29526
accuracy			0.91	96137
macro avg	0.89	0.93	0.90	96137
weighted avg	0.93	0.91	0.91	96137

The area under the curve will range from 50 percent, for a worthless model, to 100 percent for a perfect classifier. Points closer to the upper-right corner correspond to low cutoff probabilities, whereas points in the lower left correspond to higher cutoff probabilities. The extreme points (1, 1) and (0, 0) represent no-data rules where all cases are classified into bankrupt or non-bankrupt, respectively.

It seems obvious that applicants with higher incomes and smaller loan requests are more likely to be accepted because they are more likely to pay back their obligations. Other aspects, such gender and marital status, don't seem to be taken into account by the company. This forecasting module should be improved in the future, and integrated. The system is trained using prior training data, but it is feasible to alter the software in the future so that it may accept new testing data as well as training data and predict as necessary.

6.4 Google Colab Link

https://colab.research.google.com/drive/1JHY6EALrhYivq60Zzlf5W4QpYN2mrw-J?usp=sharing

6.5 Data Source Link

https://www.kaggle.com/datasets/subhamjain/loan-prediction-based-on-customer-behavior

7.1 Contributions

Loan default prediction using machine learning has significant contributions to the world, especially in the financial sector. Some of these contributions include:

Improved Accuracy: Machine learning algorithms can analyze vast amounts of data and learn from it to make accurate predictions. By using machine learning in loan default prediction, financial institutions can accurately predict which customers are more likely to default on loans, reducing the risk of financial loss.

Reduced Risk: Predicting loan defaults using machine learning can help financial institutions mitigate the risk of lending money to high-risk customers. This, in turn, can lead to a reduction in non-performing loans, which can impact a bank's financial stability.

Time-Saving: Loan default prediction using machine learning can significantly reduce the time taken to analyze and approve loan applications. By automating the loan approval process, banks and other financial institutions can approve loans faster, improving customer experience.

Increased Efficiency: By predicting loan defaults accurately, financial institutions can proactively identify and address potential risks. This can lead to increased efficiency in the loan approval process, reducing the need for human intervention.

Improved Customer Experience: Loan default prediction using machine learning can help banks and other financial institutions offer personalized services to their customers. By analyzing customer data, institutions can offer tailored loan products that meet the specific needs of each customer.

In summary, loan default prediction using machine learning has significant contributions to the financial sector, including improved accuracy, reduced risk, time-saving, increased efficiency, and improved customer experience.

8. Conclusion and Future Scope

8.1 Conclusion

The analysis and modeling presented in this project aimed to predict whether a loan applicant would default or not based on various features related to the applicant's financial history, demographics, and other factors.

Through exploratory data analysis, we identified key variables that had a significant impact on loan defaults, including the applicant's credit history, education level, and loan amount.

We then used a decision tree model to classify loan applications as either default or non-default based on these variables. The model achieved an accuracy of 91%, indicating that it could be a useful tool for lenders to evaluate loan applications and minimize the risk of default.

Overall, the results of this project demonstrate the potential of machine learning techniques to improve the loan approval process and reduce the risk of financial loss for lenders. However, it is important to note that this model may not be applicable to all situations, and additional research and testing may be needed to develop a more robust and accurate model.

8.2 Future Scope

The processes in the prediction process include data cleaning and processing, imputation of missing values, experimental analysis of the data set, model creation, and testing on test data. In the future, this prediction module may be improved and incorporated. The system is trained using prior training data, but it is feasible to alter the software in the future so that it may accept new testing data as well as training data and predict as necessary.

9. References

Xia, Yufei, Lingyun He, Yinguo Li, Nana Liu, and Yanlin Ding. "Predicting loan default in peer-to-peer lending using narrative data." *Journal of Forecasting* 39, no. 2 (2020): 260-280. https://doi.org/10.1002/for.2625

Foster, Benjamin P., and Jozef Zurada. "Loan defaults and hazard models for bankruptcy prediction." *Managerial Auditing Journal* 28, no. 6 (2013): 516-541. https://doi.org/10.1108/02686901311329900

Gomathy, C. K., Ms Charulatha, Mr AAkash, and Ms Sowjanya. "THE LOAN PREDICTION USING MACHINE LEARNING." (2021).

Sheikh, Mohammad Ahmad, Amit Kumar Goel, and Tapas Kumar. "An approach for prediction of loan approval using machine learning algorithm." In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 490-494. IEEE, 2020. https://doi.org/10.1109/ICESC48915.2020.9155614

Zhu, Lin, Dafeng Qiu, Daji Ergu, Cai Ying, and Kuiyi Liu. "A study on predicting loan default based on the random forest algorithm." *Procedia Computer Science* 162 (2019): 503-513. https://doi.org/10.1016/j.procs.2019.12.017

Krichene, Aida. "Using a naive Bayesian classifier methodology for loan risk assessment: Evidence from a Tunisian commercial bank." *Journal of Economics, Finance and Administrative Science* 22, no. 42 (2017): 3-24. https://doi.org/10.1108/JEFAS-02-2017-0039

Al Mamun, Miraz, Afia Farjana, and Muntasir Mamun. "Predicting Bank Loan Eligibility Using Machine Learning Models and Comparison Analysis."

Adewusi, Amos Olaolu, Tunbosun Biodun Oyedokun, and Mustapha Oyewole Bello. "Application of artificial neural network to loan recovery prediction." *International Journal of Housing Markets and Analysis* (2016).https://doi.org/10.1108/IJHMA-01-2015-0003

Al Mamun, Miraz, Afia Farjana, and Muntasir Mamun. "Predicting Bank Loan Eligibility Using Machine Learning Models and Comparison Analysis." http://dx.doi.org/10.33899/csmj.2020.164686

Udaya Bhanu, L., and Dr S. Narayana. "Customer loan prediction using supervised learning technique." *Int. J. Sci. Res. Publ* 11, no. 6 (2021): 403-407. http://dx.doi.org/10.29322/IJSRP.11.06.2021.p11453