ARTIFICIAL INTELLIGENCE

(Real-World Problem Solving Using Artificial Intelligence Approaches)

Project on:

“Predicting Loan Eligibility”

Done By:

Bidhan Pant

On

10th May 2023

Table of Contents

Table of Figures

Abstract

The machine learning approach started to flourish in the 1990s, and since then it has done a great impact on humans. In today's time, people are using machine learning for every task to make their life easier. Machine learning algorithms are leaving an impact in every field, as we can see Artificial Intelligence (AI) is growing rapidly in different sectors and it is growing in different financial sectors like banks, loan lending organizations and many more are using AI for different purposes. As we all know approving loans to their clients is one of the most tedious tasks in banks or any loan lending organization. So, to overcome this problem I approached this research-based project, where I used different machine learning algorithms (i.e., support vector machine, random forest, decision tree, and logistic regression) to find the best accuracy for the prediction of loan eligibility. In this research-based project, I used a dataset from Kaggle (i.e., Loan Eligibility Dataset), and the random forest classifier gave the best result score of 91.73% and our model performed better than the model I found in the literature in terms of feature selection using different models.

1. Introduction

The banking industry, like many other businesses, is increasingly striving to make use of the opportunities provided by contemporary technologies to enhance their operations, boost productivity, and cut costs. Predictive analytics was reportedly the most widely used machine learning feature for applications in the global banking sector in 2020. Most of the lending platforms' capacity to assess credit risk determines whether they are successful or not (Orji, Ugwuishiwu, Nguemaleu, et al. 2022). Any financial institution that approves loans faces a difficult task. The bank determines whether a borrower is excellent (non-defaulter) or bad (defaulter) before extending credit to them.

This project focused on developing machine learning (ML) models to forecast loan eligibility, which is critical in expediting decision-making and determining whether an applicant receives a loan (Orji, Ugwuishiwu, Nguemaleu, et al. 2022). The work includes: (1) cleaning and pre-processing the data for modelling; (2) doing exploratory data analysis (EDA) on the dataset; (3) building several ML models to predict loan eligibility; and (4) evaluating and comparing the various models produced.

2. The Real-World Problem

In today’s time banks and other loan lenders are facing problems related to loan eligibility and finding a way through is crucial for both loan lender and borrowers. Analysing, loan eligibility is not an easy task because we must filter wide range of factors like credit score of applicants, income of applicant, loan amount, loan length, and so on. This process takes a lot of time to process and error-prone, and sometimes the result may be biased or discriminatory.

This is where AI comes in. AI can help in this situation. Today's AI has advanced to the point where it can analyse enormous amounts of data as well as identify patterns and trends that humans or other conventional methods would not be able to identify. Different AI approaches for loan eligibility can be used by banks and other loan-lending organizations to make wise decisions. If the right model is chosen for the purpose, AI may also remove biases and bring about fairness by considering a wide range of

characteristics, assisting loan lending organizations in making objective decisions based on the information applicants provide rather than subjective human judgment.

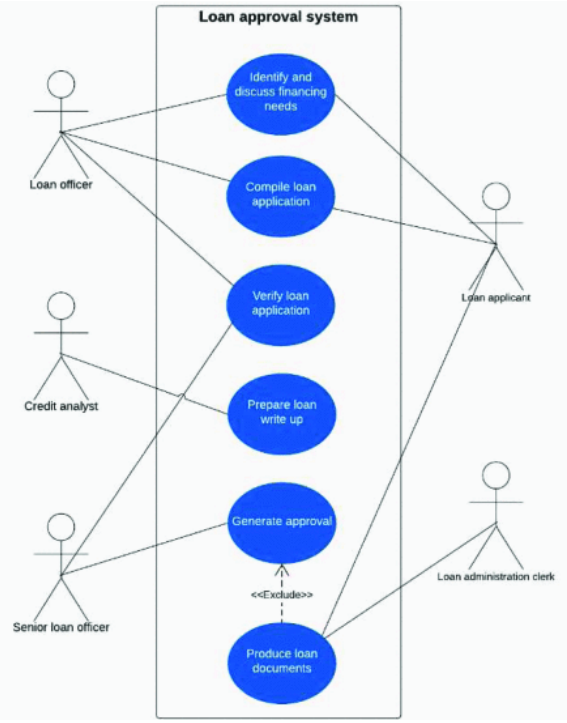

Image source: (Lohani, Trivedi, et al. 2022)

AI-based loan prediction systems have the potential to completely change the lending business, by offering more precise, effective, and unbiased loan assessments. By offering quicker and more individualised loan services, they can assist financial institutions in making better lending decisions, lowering their risks, and enhancing the client experience. AI can also assist advance financial inclusion by making it easier for more individuals to acquire credit based on objective standards rather than judgements.

3. Project Aim and Objectives

The primary goal of this project is to assist both loan lenders and consumers in passing loans and taking loans by developing a predictive model that can accurately demonstrate whether a loan applicant can pay a loan. This will safeguard the bank from economic instability as well as the loan applicant from potential legal complications if the loan is not paid on time. The objectives of the project are:

- Firstly, searching and collecting loan related data from various platforms and pre-process to make sure its quality, accuracy and consistency. This involves data cleaning, finding if dataset present any missing values, outliers, and to check if there are inconsistencies in the data.
- Secondly, to choose and train relevant machine learning models capable of predicting whether a loan applicant is acceptable. We find the accuracy of train data and test data without selecting the feature in this section to see how our model works.
- Thirdly, Choosing features from pre-processed data to help improve the model's performance, which leads to higher accuracy.
- Finally, it is now time to assess the performance of the chosen model using the cross-validation technique to confirm that it generalizes well to new data.

4. Adopted Artificial Intelligence Approach

4.1. Review of Related Literature

The authors conducted a systematic evaluation of the literature to identify and compare the best-suited ML-based models for credit risk assessment (Orji, Ugwuishiwu, N Nguemaleu, et al. 2022). The authors' goal was to demonstrate the various ML algorithms used by researchers for the credit evaluation of rural borrowers, particularly those with limited loan histories (Kumar et al. 2021). Their findings demonstrated that the ML algorithms we used in this study were widely used and produced excellent outcomes.

Low loan payback rates have a significant negative impact on banks or loan lenders around the world, and banks are searching for more efficient ways to handle loan approval processes. The authors used R.F., XGBoost, GBM, and neural network machine learning models to forecast loan defaults in the Chinese peer-to-peer (P2P) market. Their four models all exceeded 90% accuracy, with RF being the best (AKÇA and SEVLİ 2022). In terms of methodology and algorithms used, this study is like mine; however, they attempted to forecast P2P loan default while I aimed to predict consumer eligibility for loans.

In this work, the author used various machine learning algorithms to approve bank loans. The prediction technique is applied beginning with data pre-processing, missing value filling, and experimental data analysis. Following his model evaluation on the test dataset, each approach achieved an accuracy rate of 80% to 90% using the Random Forest Model (Prasanth et al. 2023). The most similar aspect is that this project uses the same dataset as mine, and our accuracy is quite better than what the author of this project obtained.

4.2. Postulated Hypothesis

Can a machine learning method solve the loan eligibility problem? Although using machine learning may seem like an option for tackling the issue of determining who can qualify for loans, we must recognize that this question is not straightforward but multifaceted, impacting both those lending and borrowing money. In my quest to grasp this matter fully, I researched an instructive case study entitled "The Fall of

Lehman Brothers: A Case Study," which recounts how Lehman Brothers filed for bankruptcy on September 15, 2008. Lehman Brothers was the fourth-largest investment bank in the United States at the time, with over 25,000 workers worldwide, assets worth $639 billion, and liabilities of $613 billion. It's strange to see such a large corporation vanish within a few years. In 2003 and 2004, with the US housing bubble well underway, Lehman acquired five mortgage lenders, including BNC Mortgage and Aurora Loan Services, which specialize in Alt-A loans. These loans were made to debtors without adequate paperwork (Chadha 2016). My analysis is that they induced borrowers to take loans without knowing their financial status, and it is clearly stated that there was no full documentation regarding the borrowers, and they just miscalculated everything and paid a large price for a tiny mistake.

4.3. Problem-Solving Approach using Artificial Intelligence

How could Lehman Brothers have saved their assets? Or how could one save their property in the present from the mistakes made by Lehman Brothers in those years? In the current situation, according to my hypothesis, we may utilize machine learning algorithms to learn patterns and correlations in big and complicated datasets provided by consumers, which traditional rule-based approaches may fail to perform (Meenaakumari et al. 2022). We may choose and train the best machine learning models, extract features and correlations from loan application data, and determine whether an applicant is suitable for a loan or not by calculating the maximum accuracy and reliability in loan prediction.

I used a variety of machine learning approaches to reach the above-mentioned hypothesis, including support vector machines, logistic regression, decision trees, and random forests. I also demonstrated pre-processing techniques like feature scaling, cleaning, normalization, and one-hot encoding to improve data quality, which will significantly improve the model's performance using multiple metrics such as precision, recall, f1-score, and support. And I've experimented with many approaches to dealing with difficulties like class imbalance, feature selection, and model interpretability.

5. Artificial Intelligence Approach Implementation

In terms of the artificial intelligence approach and implementation for loan eligibility prediction, I employed supervised machine learning techniques:

- Support Vector Machines (SVMs),
- Logistic Regression (LR),
- Random Forest Classifier (RFC), and
- Decision Trees (DT)

The main reason for choosing these algorithms is that, after carrying out research on the loan eligibility topic, I learned that these algorithms could handle binary classification problems (i.e., approved Vs rejected loans), and their advantage is that they can handle different types of data.

5.1. Support Vector Machines (SVMs) Model

SVMs are supervised machine learning models that can be used for classification problems like determining if an applicant is eligible for a loan or not. SVM is mainly known for finding the hyperplane, which helps in the separation of two classes. In our case, Hyperplane will separate eligible applicants from ineligible applicants. As shown in the figure below, let us consider that the green one is eligible and the red one is ineligible.

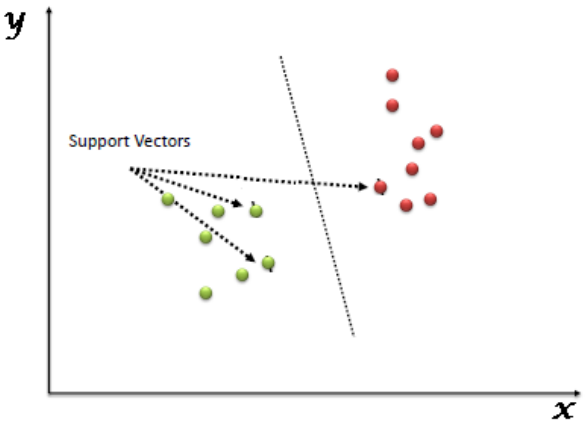

Image source: (Anon. 2023)

SVM can also be mathematically represented:

$$w*=\arg_w \max 1||w||_2[\min_n y_n|w_T(\phi(x)+b|][14]$$

Where: [minnyn|wT(ϕ(x)+b|] = minimum distance of a point to the decision boundary and $\arg_w max$ = maximum points of a function domain (Orji, Ugwuishiwu, N Nguemaleu, et al. 2022).

5.2. Logistic Regression (LR) Model

LR also known as supervised machine learning, can be used for binary classification problems, and it is suitable for a project like loan eligibility prediction. It models the probability based on the input feature, which means it predicts the output based on what the applicant feeds in as an input (Lohani, Bibhu, et al. 2022).

LR can also be mathematically represented:

$$p = (1 / 1+e^{-(\beta_0+\beta_1 x)}) * [11]$$

Where: 1 + e = exponential function, β_0 = intercept, and $\beta_1 x$ = regression coefficient.

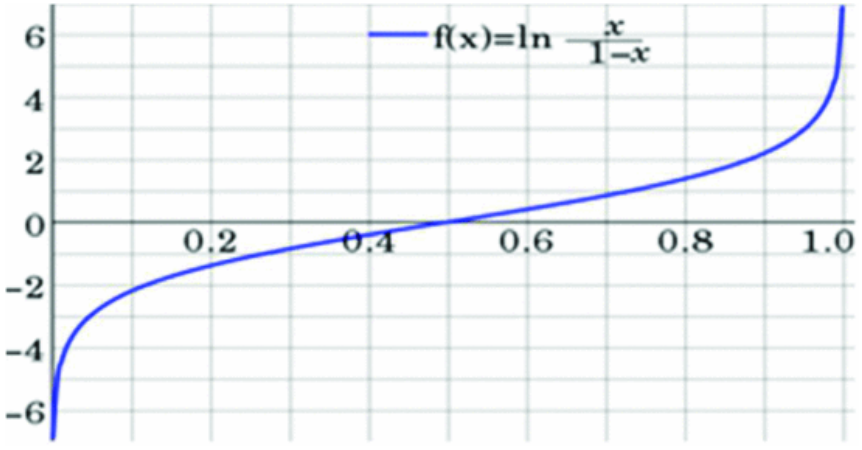

Image source: (Lohani, Bibhu, et al. 2022)

5.3. Decision Tree (DT) Model

DT is a supervised machine learning model that is used for both classification and regression tasks. The tree-like structure of the decision tree model helps to represent the possible decisions and their consequences, where each node of the tree shows a decision based on the feature the dataset holds (Naveen Kumar et al. 2022). In the case of loan eligibility prediction, it helps to build the important features and make the decision to lead the model to predict if an applicant's loan will get accepted or rejected.

DT can also be mathematically represented:

$$IG(T,a)=H(T)-H(T \mid a)[16]$$ (Kim 2022)

Where: H (T | a) = conditional entropy of T and a = value of attribute.

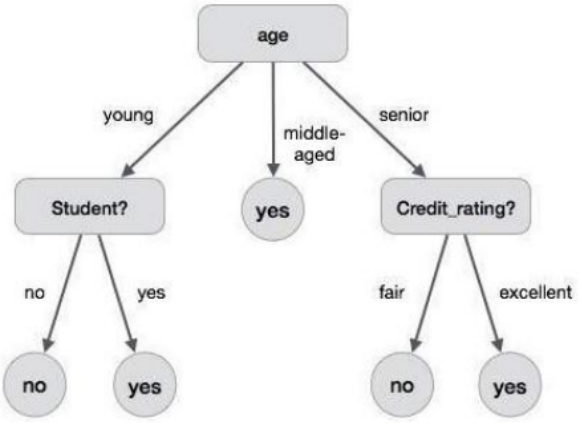

Image source: (S PG Scholar and Professor n.d.)

5.4. Random Forest Classifier (RFC) Model

RFC is an ensemble learning model that creates multiple decision trees and combines the outputs generated by them to perform prediction (Sudharshan Reddy et al. 2022). In the loan eligibility prediction scenario, it helps in selecting the best features using the feature engineering process, which helps to gain higher accuracy based on the selected features generated by the RFC model.

DT can also be mathematically represented:

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (fi - yi)^2$$

Where N = number of data points, fi = value returned and Yi = actual value of data point i.

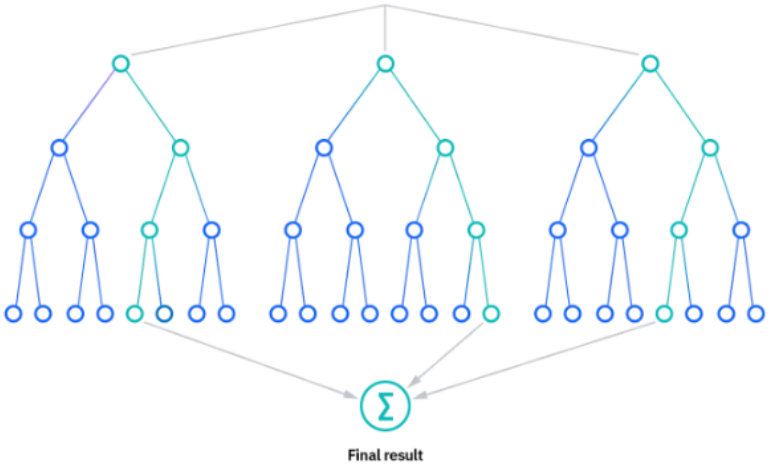

Image source: (IBM 2023)

The project's pre-processing phase entails feature selection, scaling, and normalization. Feature selection will help in selecting the most relevant features, and scaling and normalization were used to ensure that all the features we selected have the same scale and range, which is important for machine learning algorithms such as SVMs.

I employed a variety of metrics to analyse and evaluate the model's performance, including accuracy, precision, recall, and F1-score (R et al. 2022). Accuracy metric is the

ratio between the number of correct predictions and the total number of predictions (Anon. 2023):

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Also, for binary classification like loan eligibility prediction, accuracy can also be calculated in terms of the confusion matrix terminology (R et al. 2022):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

The formula for f1 score is:

$$\text{F1} = \frac{2 * (precision * recall)}{(precision + recall)}$$

I also utilized the number of features, the tree depth, and the regularisation parameters to examine the model's performance in different parameters to determine the best performance, which is also known as sensitivity analysis.

Aside from what I've used to predict loan eligibility, neural networks, deep learning, and ensemble methods can all be used as AI approaches. Deep learning and neural networks are utilized for huge and multidimensional data sets such as pictures and texts. Ensemble approaches such as AdaBoost and gradient boosting are used to improve the overall accuracy, but they also increase complexity and computation time.

As a result, I chose SVMs, logistic regression, random forest, and decision trees for my project because they are well-established and frequently used in binary classification use cases. The main reason I chose these models over others is because of their high performance and flexibility in managing various sorts of data, as well as their ease of use, interpretation, and explanation. Another reason to choose them is that they are

computationally efficient and can perform well with little amounts of data, making them suited for loan eligibility decisions.

6. Evaluation, Results and Discussions

I evaluated the performance of four supervised machine learning methods for predicting loan eligibility while working on this project. The dataset includes 614 observations and 13 attributes, such as the applicant's gender, married status, dependents, education, employment status, income, loan amount, credit history, and so on. The dataset I used in this experiment is available on Kaggle under the Database Contents Licence (DbCL) v1.0 and is titled "Loan Eligible Dataset."

To make model's performance better, the following steps were deployed to analyse and pre-process the data for modelling.

- To tackle imbalanced classification problems, which are also considered as important sources of error in machine learning models, I used the synthetic minority oversampling technique (SMOTE).
- Use of one-hot encoding to convert categorical variables to binary form so that the machine learning model could interpret the input.
- To boost the model's performance, I performed data normalization, which transforms features and ensure that they are all on a similar scale.
- Exploratory data analysis (EDA) was used to investigate patterns, trends, and anomalies in the dataset. At this point, the data was cleaned up and any missing or incomplete information was eliminated.

Furthermore, while exploring the dataset, I found that.

- Male applicant is more than female applicant (i.e., male: 489 and female: 112)

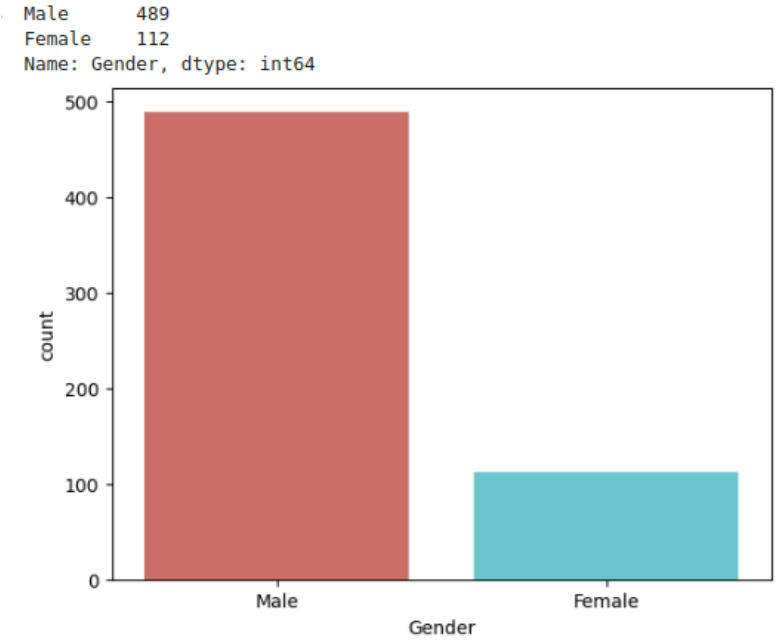

Figure 1: Male Vs Female in Dataset

- Number of married applicants are more (i.e., married: 398 and single: 213)

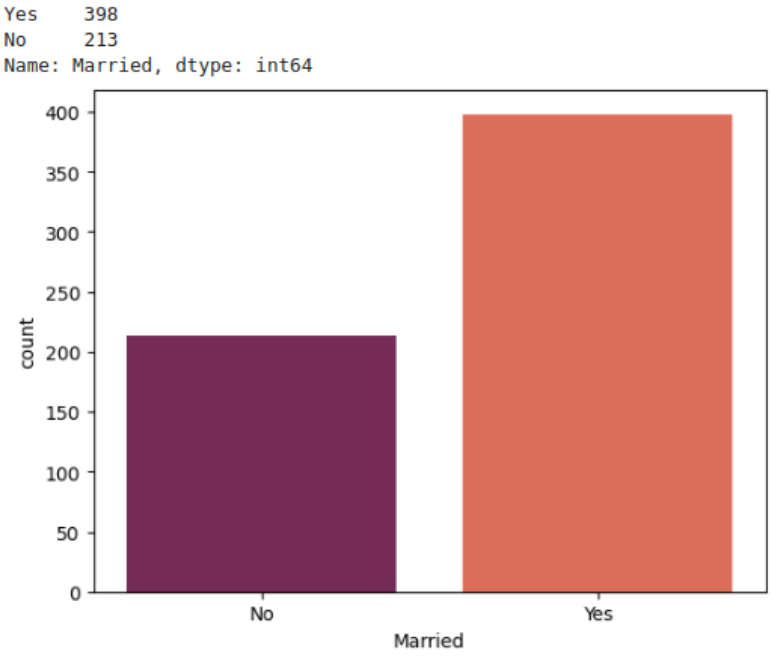

Figure 2: Married Vs Single in Dataset

- There is more graduate applicant (i.e., graduate: 480 and not graduate: 134)

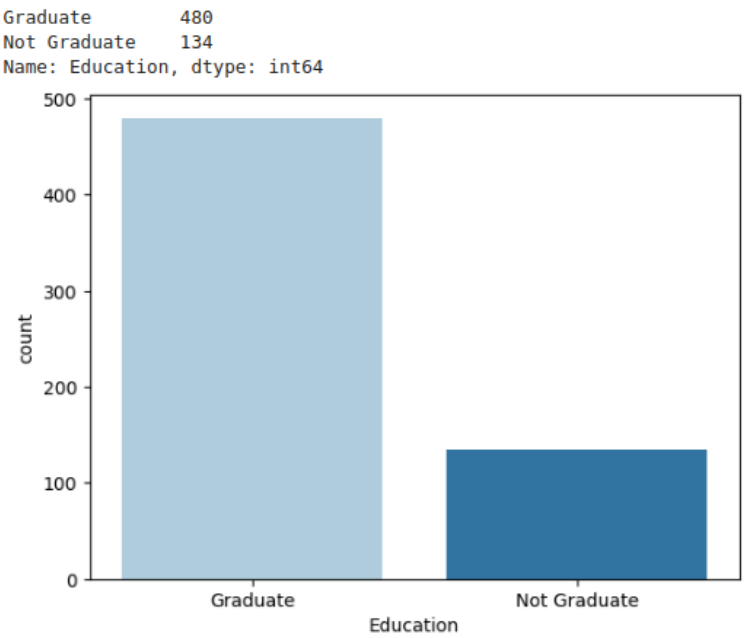

Figure 3: Number of Graduate Vs Not Graduate

- There are less self-employed applicant and more employed or other employment (i.e., self-employed: 82 and employed/other: 500)

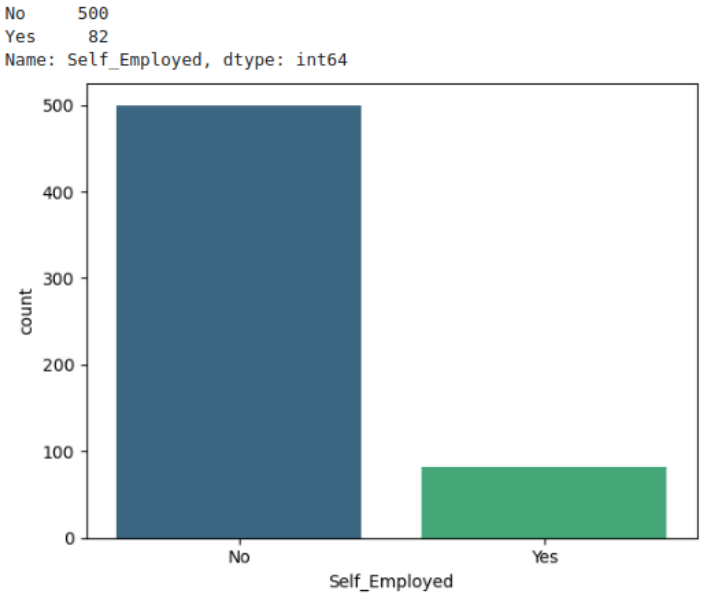

Figure 4: Self_Employed Vs Other

- More number of applicants live in semi urban area and less lives in rural (i.e., semi urban: 233, urban: 202 and rural: 179).

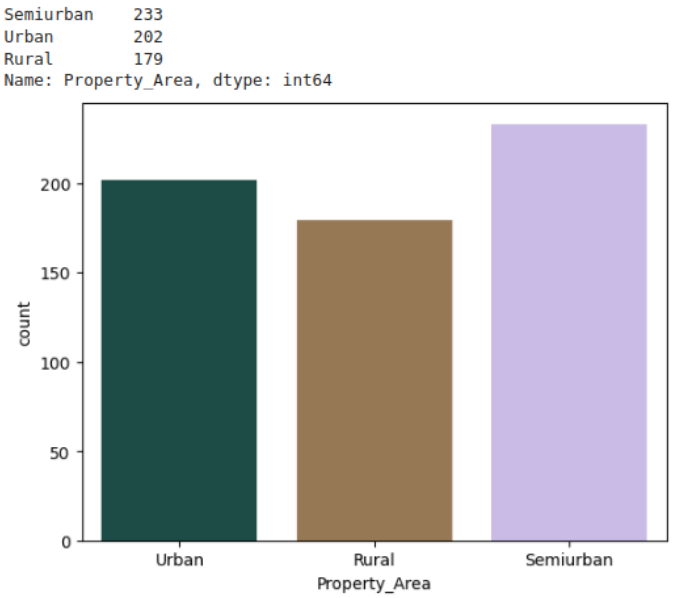

Figure 5: Applicant location

- Number of approved loans is more than not approved (i.e., approved: 422 and not approved: 192).

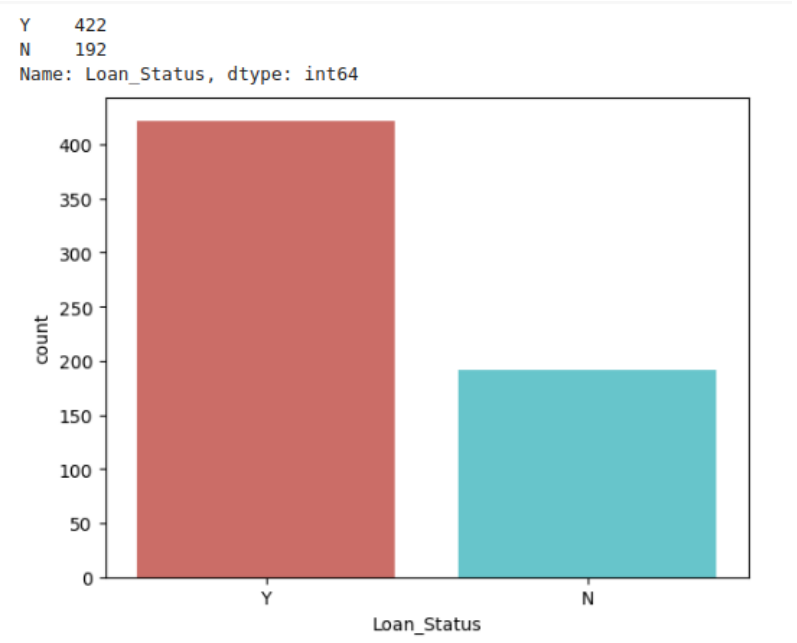

Figure 6: Number of approved loan and not approved

The correlation matrix depicts the relationship between all pairs of variables in the dataset, with values ranging from -1 to 1 (perfect negative correlation). The heat map below shows which variables are significantly associated (positive correlation is indicated in red, negative correlation is shown in blue) and which are not (light pink indicates no correlation).

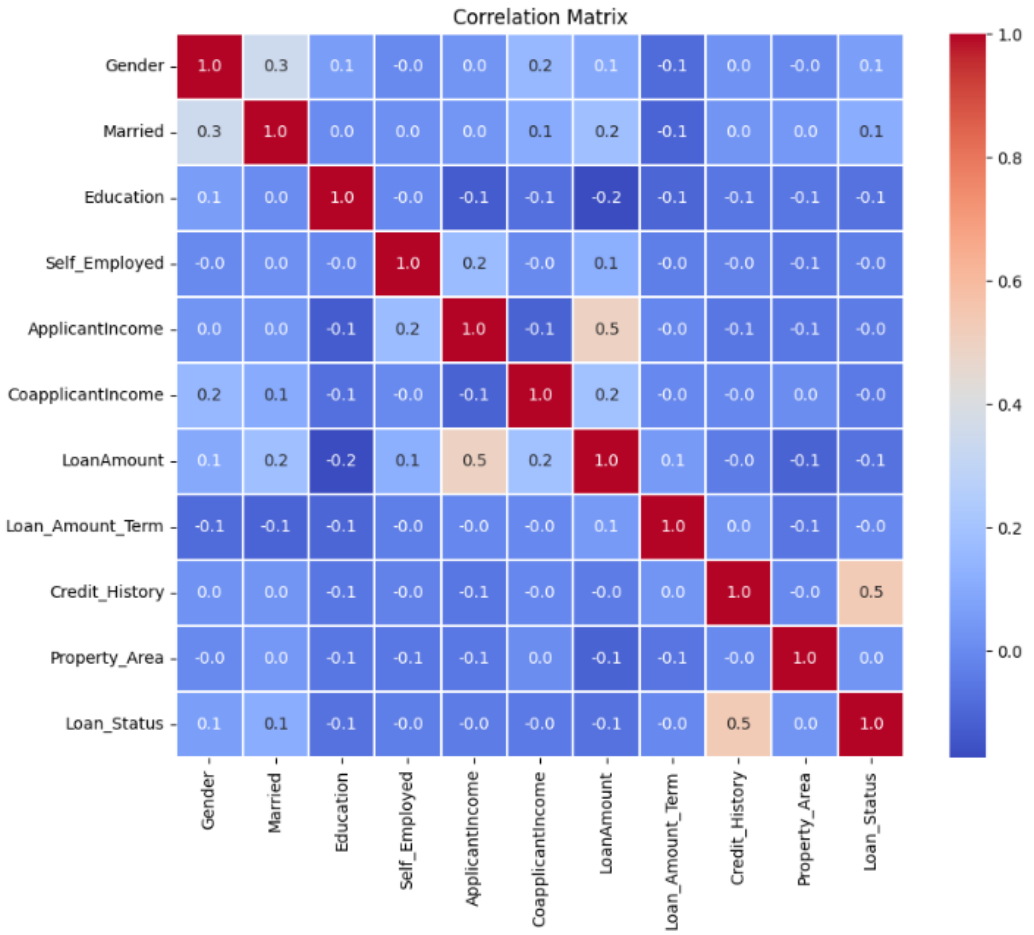

Figure 7: Correlation matrix between all pairs of variables

Furthermore, I divided the dataset into 80% training and 20% testing sets using the train_test_split function from the sklearn library. Then, using training and testing sets, I trained each machine learning model. To begin, I trained models without applying any feature engineering, scaling, or normalization to see how different models perform, and the accuracy is quite good. I trained the training dataset with training and then trained the testing dataset with training to check if there was any

difference in accuracy and to see if there was any noise in our dataset. For evaluating performance, I used metrics such as accuracy, precision, recall, and f1-score.

6.1. Support Vector Machine (SVMs) Model

Below shows the evaluation of our SVM model with train and test accuracy.

```
Train Accuracy: 79.17
Test Accuracy: 80.21
confusion matrix:
 [[ 45  73]
 [  7 259]]
classification report:
              precision    recall  f1-score   support

           0       0.87      0.38      0.53       118
           1       0.78      0.97      0.87       266

    accuracy                           0.79       384
   macro avg       0.82      0.68      0.70       384
weighted avg       0.81      0.79      0.76       384

confusion matrix:
 [[16 14]
 [ 5 61]]
classification report:
              precision    recall  f1-score   support

           0       0.76      0.53      0.63        30
           1       0.81      0.92      0.87        66

    accuracy                           0.80        96
   macro avg       0.79      0.73      0.75        96
weighted avg       0.80      0.80      0.79        96
```

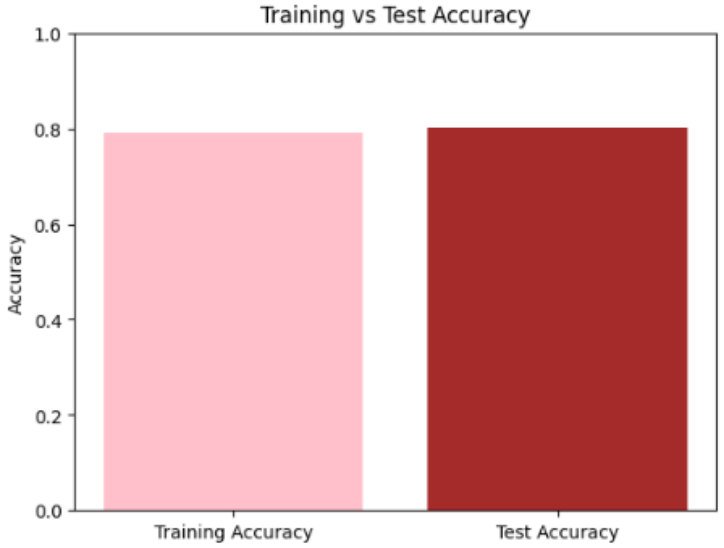

Figure 8: Train and Test accuracy of SVMs

6.2. Logistic Regression (LR) Model

Below shows the evaluation of our LR model with train and test accuracy.

```
Train Accuracy: 80.73
Test Accuracy: 83.33
confusion matrix:
 [[ 50  68]
 [  6 260]]
classification report:
              precision    recall  f1-score   support

           0       0.89      0.42      0.57       118
           1       0.79      0.98      0.88       266

    accuracy                           0.81       384
   macro avg       0.84      0.70      0.73       384
weighted avg       0.82      0.81      0.78       384

confusion matrix:
 [[17 13]
 [ 3 63]]
classification report:
              precision    recall  f1-score   support

           0       0.85      0.57      0.68        30
           1       0.83      0.95      0.89        66

    accuracy                           0.83        96
   macro avg       0.84      0.76      0.78        96
weighted avg       0.84      0.83      0.82        96
```

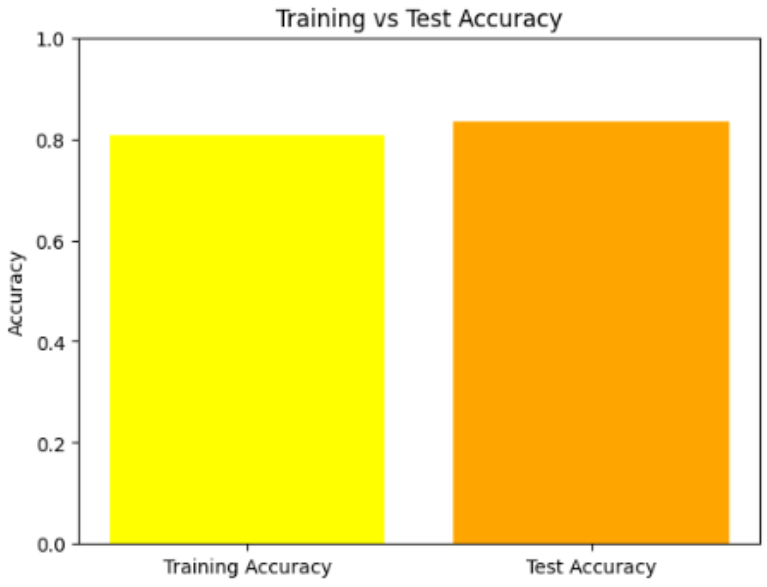

Figure 9: Train and Test accuracy of LR

6.3. Decision Tree (DT) Model

Below shows the evaluation of our DT model with train and test accuracy.

```
Train Accuracy: 84.9
Test Accuracy: 81.25
confusion matrix:
 [[ 63  55]
 [  3 263]]
classification report:
              precision    recall  f1-score   support

           0       0.95      0.53      0.68       118
           1       0.83      0.99      0.90       266

    accuracy                           0.85       384
   macro avg       0.89      0.76      0.79       384
weighted avg       0.87      0.85      0.83       384

confusion matrix:
 [[17 13]
 [ 5 61]]
classification report:
              precision    recall  f1-score   support

           0       0.77      0.57      0.65        30
           1       0.82      0.92      0.87        66

    accuracy                           0.81        96
   macro avg       0.80      0.75      0.76        96
weighted avg       0.81      0.81      0.80        96
```

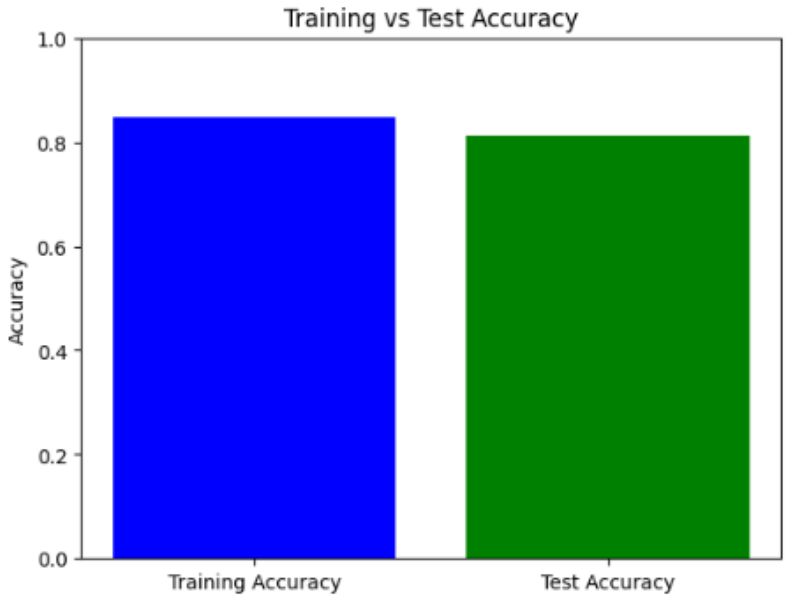

Figure 10: Train and Test accuracy of DT

6.4. Random Forest Classifier (RFC) Model

Below shows the evaluation of our RFC model with train and test accuracy.

```
Train Accuracy: 93.23
Test Accuracy: 83.33
confusion matrix:
 [[ 92  26]
 [  0 266]]
classification report:
              precision    recall  f1-score   support

           0       1.00      0.78      0.88       118
           1       0.91      1.00      0.95       266

    accuracy                           0.93       384
   macro avg       0.96      0.89      0.91       384
weighted avg       0.94      0.93      0.93       384

confusion matrix:
 [[19 11]
 [ 5 61]]
classification report:
              precision    recall  f1-score   support

           0       0.79      0.63      0.70        30
           1       0.85      0.92      0.88        66

    accuracy                           0.83        96
   macro avg       0.82      0.78      0.79        96
weighted avg       0.83      0.83      0.83        96
```

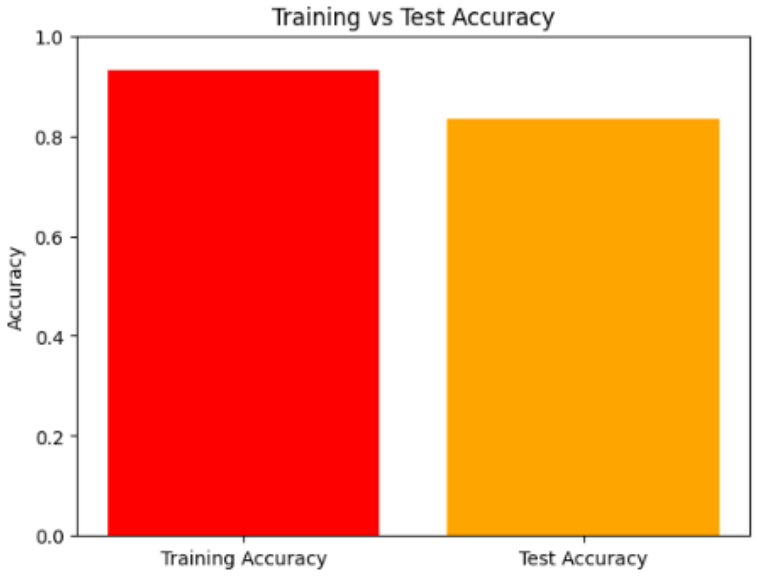

Figure 11: Train and Test accuracy of RFC

6.5. Random Forest Classifier (RFC) using Feature Selection

Feature importance using RFC along with their scores:

Features Importance:

	Feature	Importance
0	Gender	2.347337
1	Married	2.951717
2	Dependents	5.181183
3	Education	2.279098
4	Self_Employed	1.807454
5	ApplicantIncome	20.936030
6	CoapplicantIncome	10.391863
7	LoanAmount	20.572736
8	Loan_Amount_Term	5.259533
9	Credit_History	22.990405
10	Property_Area	5.282644

Figure 12: Feature importance details using RFC

MinMaxScaler was used to scale the higher percentage numeric features such as ApplicantIncome, CoapplicantIncome, LoanAmount, and Credit_History. This is done to ensure that all features contribute equally to the RFC model. The improved result after using feature importance, scaling, oversampling, and one-hot encoding is shown below, where we achieved 91.73% along with confusion matrix and classification report. Previously, it was only 83.33%.

```
rfc = RandomForestClassifier()

#making cross-validation prediction
Y_pred = cross_val_predict(rfc, X_resampled, Y_resampled, cv=10)

#calculating confusion matrix and classification report
conf_matrix = confusion_matrix(Y_resampled, Y_pred)
cls_report = classification_report(Y_resampled, Y_pred)

#printing
results = cross_validate(rfc, X_resampled, Y_resampled, cv=10, scoring=['accuracy'], return_train_score=True)
print("Accuracy:", np.round(np.mean(results['test_accuracy']*100),2))
print("confusion matrix:\n", conf_matrix)
print("classification report:\n", cls_report)

```

```
Accuracy: 91.73
confusion matrix:
 [[309  23]
 [ 44 288]]
classification report:
              precision    recall  f1-score   support

           0       0.88      0.93      0.90       332
           1       0.93      0.87      0.90       332

    accuracy                           0.90       664
   macro avg       0.90      0.90      0.90       664
weighted avg       0.90      0.90      0.90       664
```

Figure 13: Accuracy using feature engineering in RFC

6.6. Decision Tree (DT) using Feature Selection

Feature importance using RFC along with their scores:

Features Importance:

	Feature	Importance
0	Gender	0.000000
1	Married	3.521440
2	Dependents	0.000000
3	Education	0.000000
4	Self_Employed	0.000000
5	ApplicantIncome	14.882325
6	CoapplicantIncome	1.525957
7	LoanAmount	11.708732
8	Loan_Amount_Term	7.120812
9	Credit_History	61.240734
10	Property_Area	0.000000

Figure 14: Feature importance details using DT

Same as an RFC model we have used in DT too. The improved result after using feature importance, scaling, oversampling, and one-hot encoding is shown below, where we achieved 91.44% along with confusion matrix and classification report. Previously, it was only 81.25%.

```
dtc = DecisionTreeClassifier(max_depth=5)

#making cross-validation prediction
Y_pred = cross_val_predict(dtc, X_resampled, Y_resampled, cv=10)

#calculating confusion matrix and classification report
conf_matrix = confusion_matrix(Y_resampled, Y_pred)
cls_report = classification_report(Y_resampled, Y_pred)

#printing
results = cross_validate(rfc, X_resampled, Y_resampled, cv=10, scoring=['accuracy'], return_train_score=True)
print("Accuracy:", np.round(np.mean(results['test_accuracy']*100),2))
print("confusion matrix:\n", conf_matrix)
print("classification report:\n", cls_report)
```

```
Accuracy: 91.44
confusion matrix:
 [[181 151]
 [ 49 283]]
classification report:
              precision    recall  f1-score   support

           0       0.79      0.55      0.64       332
           1       0.65      0.85      0.74       332

    accuracy                           0.70       664
   macro avg       0.72      0.70      0.69       664
weighted avg       0.72      0.70      0.69       664
```

Figure 15: Accuracy using feature engineering in DT

Accuracy of the train and test dataset in a model without Feature Selection			Accuracy of the test dataset in a model with Feature Selection	
Models	Train Accuracy	Test Accuracy	Models	Final Accuracy
Support Vector Machine	79.17%	80.21%	Random Forest	91.73%
Logistic Regression	80.73%	83.33%	Decision Tree	91.44%
Decision Tree	84.9%	82.29%		
Random Forest	94.79%	82.29%		

Loan defaulting is now regarded as a key financial risk for banks and other loan providers. Many organizations continue to use the same old traditional method for determining loan eligibility. The field of AI has advanced far in this industry as well; a decent and large dataset can produce good results and even assist in making informed choices. Going through the various research papers on the loan eligibility topic, I notice many similarities in the project approach. What I did differently for this project was feature engineering to get high accuracy, which I hadn't noticed in many research papers. When compared to other research approaches, our model achieved high accuracy based on different metrics, with the RF model achieving 91.73% and the DT model achieving 91.44%.

7. Conclusion and Future Work

This research provides insight into loan eligibility prediction utilizing machine learning methods. Throughout this project, I learned about various classification algorithms that can be used to solve loan eligibility problems. In this study, I discussed real-world loan eligibility issues and how loan lenders face various challenges due to a lack of adequate AI understanding. To validate my technique, I displayed numerous literatures that are like my work, as well as a real-life case study based on a loan issue. Finally, I used four machine learning techniques to support my statement and to help me meet my project objectives.

In the future, we can employ advanced machine learning techniques such as neural networks and deep learning to improve accuracy with improved and larger datasets for predicting loan eligibility. Furthermore, we can conduct more research on unsupervised learning algorithms to detect patterns and anomalies in loan applications in order to reduce the risk of fraud.

References

AKÇA, M. F. and SEVLİ, O., 2022. Predicting acceptance of the bank loan offers by using support vector machines. *International Advanced Researches and Engineering Journal* [online], 6 (2), 142–147. Available from: https://dergipark.org.tr/en/pub/iarej/issue/72165/1058724 [Accessed 9 May 2023].

Anon., 2023. *SVM | How to Use Support Vector Machines (SVM) in Data Science* [online]. Available from: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ [Accessed 9 May 2023].

Anon., 2023. Vegetable Leaf Disease Detection Using Deep Learning. *International Journal of Research in Engineering and Science (IJRES) ISSN* [online], 11, 548–555. Available from: www.ijres.org548| [Accessed 9 May 2023].

Chadha, P., 2016. What Caused the Failure of Lehman Brothers? Could it have been Prevented? How? Recommendations for Going Forward.

IBM, 2023. *What is Random Forest? | IBM* [online]. Available from: https://www.ibm.com/topics/random-forest [Accessed 9 May 2023].

Kim, H., 2022. Supervised Learning. *Artificial Intelligence for 6G* [online], 87–182. Available from: https://link.springer.com/chapter/10.1007/978-3-030-95041-5_4 [Accessed 9 May 2023].

Kumar, A., Sharma, S. and Mahdavi, M., 2021. Machine Learning (ML) Technologies for Digital Credit Scoring in Rural Finance: A Literature Review. *Risks 2021, Vol. 9, Page 192* [online], 9 (11), 192. Available from: https://www.mdpi.com/2227-9091/9/11/192/htm [Accessed 9 May 2023].

Lohani, B. P., Bibhu, V., Trivedi, M., Ranjan, S., Singh, R. J. and Kumar Kushwaha, P., 2022. Machine Learning Based Model for Prediction of Loan Approval; Machine Learning Based Model for Prediction of Loan Approval. *2022 3rd International Conference on Intelligent Engineering and Management (ICIEM)*.

Lohani, B. P., Trivedi, M., Singh, R. J., Bibhu, V., Ranjan, S. and Kushwaha, P. K., 2022. Machine Learning Based Model for Prediction of Loan Approval. *Proceedings of 3rd International Conference on Intelligent Engineering and Management, ICIEM 2022*, 465–470.

Meenaakumari, M., Jayasuriya, P., Dhanraj, N., Sharma, S., Manoharan, G. and Tiwari, M., 2022. Loan Eligibility Prediction using Machine Learning based on Personal Information. *Proceedings of 5th International Conference on Contemporary Computing and Informatics, IC3I 2022*, 1383–1387.

Naveen Kumar, C., Keerthana, D., Kavitha, M. and Kalyani, M., 2022. Customer Loan Eligibility Prediction using Machine Learning Algorithms in Banking Sector; Customer Loan Eligibility Prediction using Machine Learning Algorithms in Banking Sector. *2022 7th International Conference on Communication and Electronics Systems (ICCES)*.

Orji, U. E., Ugwuishiwu, C. H., N Nguemaleu, J. C. and Ugwuanyi, P. N., 2022. Machine Learning Models for Predicting Bank Loan Eligibility; Machine Learning Models for Predicting Bank Loan Eligibility. *2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)*.

Orji, U. E., Ugwuishiwu, C. H., Nguemaleu, J. C. N. and Ugwuanyi, P. N., 2022. Machine Learning Models for Predicting Bank Loan Eligibility. *Proceedings of the 2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development, NIGERCON 2022*.

Prasanth, C., Kumar, R. P., Rangesh, A., Sasmitha, N. and B, D., 2023. Intelligent Loan Eligibility and Approval System based on Random Forest Algorithm using Machine Learning. *2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA)* [online], 84–88. Available from: https://ieeexplore.ieee.org/document/10100225/ [Accessed 9 May 2023].

R, P. B., Kumar, A., Rao, B., Salian, P. K. and P, S. A., 2022. An Approach to Predict Loan Eligibility using Machine Learning; An Approach to Predict Loan Eligibility using Machine Learning. *2022 International Conference on Artificial Intelligence and Data Engineering (AIDE)*.

S PG Scholar, S. M. and Professor, A., n.d. Loan Credibility Prediction System Based on Decision Tree Algorithm Rekha Sunny T. [online]. Available from: www.ijert.org [Accessed 9 May 2023].

Sudharshan Reddy, C., Salauddin Siddiq, A. and Jayapandian, N., 2022. Machine Learning based Loan Eligibility Prediction using Random Forest Model; Machine Learning based Loan Eligibility Prediction using Random Forest Model. *2022 7th International Conference on Communication and Electronics Systems (ICCES)*.

Appendix A

Code Section:

```
# Predicting LOAN Eligibility Using Machine Learning
"""



"""##Importing Libraries"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno


from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix



"""##Reading DataSet

Importing and Reading Dataset
"""

import io
import requests

url =
"https://raw.githubusercontent.com/encrypted000/loan_prediction/main/loan_data_set.cs
v"
get = requests.get(url).content
loan_data = pd.read_csv(io.StringIO(get.decode('utf-8')))

#lets real first five rows of the file
loan_data.head()

#displaying the shape of the file
loan_data.shape



"""##Data Exploration"""
```

```
#lets extract the information of the file including column names and their data types in the DataFrame 'df'
loan_data.info()



#lets find out the mathametical details from our dataset like: count, mean, std, min and max using describe function

loan_data.describe()



#Loan_ID
loan = loan_data.Loan_ID.value_counts(dropna = False)
loan

"""Above result shows that there are 614 unique ID in the dataset."""



"""##**Analyzing the Data**"""

#lets create a function to explore the object type of the data

def object_type(df: pd.DataFrame, feature_name: str):
  if df[feature_name].dtype == 'object':
    print(df[feature_name].value_counts())

#now lets call the function and apply it in Gender

gender = object_type(loan_data, 'Gender')

#ploting the Gender using seaborn
sns.countplot(x = "Gender", data = loan_data, palette = "hls")

#display
plt.show()

"""Above result shows that the number of male applicants are more if we compare with female applicants."""



##now lets call the function and apply it in Married
```

```
married = object_type(loan_data, 'Married')

#ploting the relationship status using seaborn
sns.countplot(x = "Married", data = loan_data, palette = "rocket")

#display
plt.show()

"""From the above result we can see that number of married applicants are more than
singles."""



##now lets call the function and apply it in Education

education = object_type(loan_data, 'Education')

#ploting the number of graduate and non graduate using seaborn
sns.countplot(x = "Education", data = loan_data, palette = "Paired")

#display
plt.show()

"""From the above result we can see that number of graduate applicants are more than non
graduate."""



##now lets call the function and apply it in Self_Employed

self_employed = object_type(loan_data, 'Self_Employed')

#ploting the number of self employed using seaborn
sns.countplot(x = "Self_Employed", data = loan_data, palette = "viridis")

#display
plt.show()

"""From above result we can see that number of non self employed applicants are more
than self employed."""



##now lets call the function and apply it in Property_Area

property_area = object_type(loan_data, 'Property_Area')
```

```
#ploting the number of property area using seaborn
sns.countplot(x = "Property_Area", data = loan_data, palette = "cubehelix")

#display
plt.show()

"""From the above result we can see that property area are equally distributed on urban,
rural and semiurban."""



#Loan_Status
loan_status = object_type(loan_data, 'Loan_Status')

#ploting the number of loan status using seaborn
sns.countplot(x = "Loan_Status", data = loan_data, palette = "hls")

#display
plt.show()

"""From the above result we can see that percentage of approved loan status of the
applicant is more."""



"""**Let's find out if we have any null values**
> If we have any null values we need to fill it using mean and median by using inbuilt
function/package missingno


"""

#listing the null values

null_values = loan_data.isnull().sum()

null_values

"""**Analyzing above result**

> Loan_ID column has 0 null value

> Gender column has 13 null values

> Married column has 3 null values

> Dependents column has 15 null values
```

> Education column has 0 null value

> Self_Employed column has 32 null values

> ApplicantIncome column has 0 null value

> CoapplicantIncome column has 0 null value

> LoanAmount column has 22 null values

> Loan_Amount_Term column has 14 null values

> Credit_History column has 50 null values

> Property_Area column has 0 null value

> Loan_Status column has 0 null value
"""

##lets create a matrix plot to visualize the nullity of values

null1 = msno.matrix(loan_data)
null1

##lets create a bar plot to visualize the nullity of values

null2 = msno.bar(loan_data)
null2

"""From the above result we can see that there are too many columns missing with small amount if null values

"""

"""**Dropping Null values**"""

#dropping the null value

loan_data = loan_data.dropna()

#re-checking null values

```
null_values = loan_data.isnull().sum()

null_values

"""As we can see there are no null values present."""

loan_data.shape



"""**Converting Categorical variables to Numerical variables**

We are only replacing DataFrames "Loan_Status", "Gender", "Married" and "Self_Employed" to 0 and 1 based on their original values.
"""

##gender.replace {"Male" : 1, "Female" : 0}

loan_data.Gender = loan_data.Gender.replace({"Male" : 1, "Female" : 0})

##married.replace {"Yes" : 1, "No" : 0}

loan_data.Married = loan_data.Married.replace({"Yes" : 1, "No" : 0})

##loan_data.replace {"Y" : 1, "N" : 0}

loan_data.Loan_Status = loan_data.Loan_Status.replace({"Y" : 1, "N" : 0})

##self_employed.replace {"Yes" : 1, "No" : 0}

loan_data.Self_Employed = loan_data.Self_Employed.replace({"Yes" : 1, "No" : 0})

#Property_Area.replace {"Rural": 0, "Semiurban": 1, "Urban": 2}

loan_data.Property_Area = loan_data.Property_Area.replace({"Rural": 0, "Semiurban": 1, "Urban": 2})

#Education.replace {"Graduate": 0, "Not Graduate": 1}

loan_data.Education = loan_data.Education.replace({"Graduate": 0, "Not Graduate": 1})

#dependents column

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

```
"""As we know that we can't take 3+ value so I am converting 3+ to 4."""

#converting 3+ to 4

loan_data.Dependents = loan_data.Dependents.replace({"3+": 4})

#re-checking
loan_data['Dependents'].value_counts()

"""Now we can see that 3+ is converted into 4."""



#checking the columns value again

loan_data.head()



##dropping unnecessary column
loan_data = loan_data.drop(['Loan_ID'], axis = 1)


##separating values i.e. Loan_ID and Loan_Status

X = loan_data.drop(['Loan_Status'], axis = 1)
Y = loan_data['Loan_Status']

print(X, Y)



"""###Data Visualization

Let's visualize some data for better understanding
"""

#visualizing the table
loan_data
```

#creating plot to visualize loan_data using default plot type which is line plot and figsize parameter sets size of plot figure i.e. width 20 inches and height 6 inches.

```
a = loan_data.plot(figsize=(20,6))
plt.show(a)
```

"""**Histogram**"""

#lets illustrate ApplicantIncome and LoanAmount in histogram distribution

#creating figure with two subplots

```
plt.figure(figsize = (18, 6))
plt.subplot(1, 2, 1)

#ploting the first histogram, for ApplicantIncome column with bins 20
loan_data['ApplicantIncome'].hist(bins = 20, color = 'green')
plt.title("Applicant Income") #setting title for first subplot


plt.subplot(1, 2, 2)
plt.grid()

#ploting the first histogram, for LoanAmount column with bins 20
loan_data['LoanAmount'].hist(bins = 20, color = 'red')
plt.title("Loan Amount") #setting title for second subplot

#display
plt.show()
```

"""**Scatter Plot**"""

##creating figure specific size

```
plt.figure(figsize = (12, 6))

##setting up the title
plt.title("Relationship between Applicant Income and Loan Amount")

##add gridlines to the plot
plt.grid()
```

```
##creating a scatter plot with ApplicantIncome and LoanIncome in the x-axis and y-axis respectively
plt.scatter(loan_data['ApplicantIncome'], loan_data['LoanAmount'], c = 'b', marker = '*')

#label x-axis and y-axis
plt.xlabel("Applicant Income")
plt.ylabel("Loan Amount")

#display
plt.show()

"""**Let's illustrate in Heatmap**"""

#creating correlation matrix for dataset
corr_matrix = loan_data.corr()

##creating figure
plt.figure(figsize = (10, 8))

#creating a heatmap of the correlation matrix
#annot = True shows correlation coefficients in every cell
sns.heatmap(corr_matrix, cmap = "coolwarm", annot = True, fmt = '.1f', linewidths = .1)

#title
plt.title("Correlation Matrix")

#display
plt.show()

"""The correlation matrix depicts the relationship between all pairs of variables in the dataset, with values ranging from -1 to 1 (perfect negative correlation). The heatmap shows which variables are significantly associated (positive correlation is indicated in red, negative correlation is shown in blue) and which are not (yellow indicates no correlation). The annot=True argument adds numerical values to each heatmap cell, while the fmt='.1f' parameter requires that each value be rounded to one decimal point. The option linewidths=.1 specifies the width of the lines between cells."""

"""###Features Separating"""

##importing train_test_split from sklearn
from sklearn.model_selection import train_test_split
```

```
##splitting data into train and test

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, stratify = Y,
random_state = 42)

#printing shapes of new sets
print("X shape is:", X.shape)
print("X_train shape is:", X_train.shape)
print("X_test shape is:", X_test.shape)



"""###Machine Learning Models

**Support Vector Machine Model**
"""

#importing model

from sklearn import svm

#initilizing
svc = svm.SVC(kernel='linear')

#training the model
svc.fit(X_train, Y_train)

#lets find out the accuracy score on training data

Y_train_pred = svc.predict(X_train)

#lets compare predicted value by our model with the original train Y_train
training_accuracy = accuracy_score(Y_train_pred, Y_train)

#lets find out the accuracy score on test data

Y_test_pred = svc.predict(X_test)

#lets compare predicted value by our model with the original train Y_test
test_accuracy = accuracy_score(Y_test_pred, Y_test)

##calculating confusion matrix  and classification report for train
conf_matrix_train = confusion_matrix(Y_train, Y_train_pred)
cls_report_train = classification_report(Y_train, Y_train_pred)

##calculating confusion matrix  and classification report for test
```

```
conf_matrix_test = confusion_matrix(Y_test, Y_test_pred)
cls_report_test = classification_report(Y_test, Y_test_pred)

#printing
print('Train Accuracy:', round(training_accuracy*100,2))
print('Test Accuracy:', round(test_accuracy*100,2))

##for train
print("confusion matrix:\n", conf_matrix_train)
print("classification report:\n", cls_report_train)

##for test
print("confusion matrix:\n", conf_matrix_test)
print("classification report:\n", cls_report_test)


# Data to plot
labels = ['Training Accuracy', 'Test Accuracy']
accuracy = [training_accuracy, test_accuracy]

# Plotting in graph
plt.bar(labels, accuracy, color=['pink', 'brown'])
plt.title('Training vs Test Accuracy')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.show()




"""**Logistic Regression**"""

##importing module
from sklearn.linear_model import LogisticRegression

#LogisticRegression

lr = LogisticRegression(max_iter=500)

#fitting in module
lr.fit(X_train, Y_train)

#lets find out the accuracy score on training data

Y_train_pred = lr.predict(X_train)
```

```
#lets compare predicted value by our model with the original train Y_train
training_accuracy = accuracy_score(Y_train_pred, Y_train)


#lets find out the accuracy score on test data

Y_test_pred = lr.predict(X_test)

#lets compare predicted value by our model with the original train Y_test
test_accuracy = accuracy_score(Y_test_pred, Y_test)

##calculating confusion matrix  and classification report for train
conf_matrix_train = confusion_matrix(Y_train, Y_train_pred)
cls_report_train = classification_report(Y_train, Y_train_pred)

##calculating confusion matrix  and classification report for test
conf_matrix_test = confusion_matrix(Y_test, Y_test_pred)
cls_report_test = classification_report(Y_test, Y_test_pred)

#printing
print('Train Accuracy:', round(training_accuracy*100,2))
print('Test Accuracy:', round(test_accuracy*100,2))

##for train
print("confusion matrix:\n", conf_matrix_train)
print("classification report:\n", cls_report_train)

##for test
print("confusion matrix:\n", conf_matrix_test)
print("classification report:\n", cls_report_test)


# Data to plot
labels = ['Training Accuracy', 'Test Accuracy']
accuracy = [training_accuracy, test_accuracy]

# Plotting in graph
plt.bar(labels, accuracy, color=['yellow', 'orange'])
plt.title('Training vs Test Accuracy')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.show()
```

```
"""**Decision Tree**"""

#importing module
from sklearn.tree import DecisionTreeClassifier

#initializing
dtc = DecisionTreeClassifier(max_depth=5)

#fitting in module
dtc.fit(X_train, Y_train)

#lets find out the accuracy score on training data

Y_train_pred = dtc.predict(X_train)

#lets compare predicted value by our model with the original train Y_train
training_accuracy = accuracy_score(Y_train_pred, Y_train)


#lets find out the accuracy score on test data

Y_test_pred = dtc.predict(X_test)

#lets compare predicted value by our model with the original train Y_test
test_accuracy = accuracy_score(Y_test_pred, Y_test)

##calculating confusion matrix  and classification report for train
conf_matrix_train = confusion_matrix(Y_train, Y_train_pred)
cls_report_train = classification_report(Y_train, Y_train_pred)

##calculating confusion matrix  and classification report for test
conf_matrix_test = confusion_matrix(Y_test, Y_test_pred)
cls_report_test = classification_report(Y_test, Y_test_pred)

#printing
print('Train Accuracy:', round(training_accuracy*100,2))
print('Test Accuracy:', round(test_accuracy*100,2))

##for train
print("confusion matrix:\n", conf_matrix_train)
print("classification report:\n", cls_report_train)

##for test
print("confusion matrix:\n", conf_matrix_test)
print("classification report:\n", cls_report_test)
```

```
# Data to plot
labels = ['Training Accuracy', 'Test Accuracy']
accuracy = [training_accuracy, test_accuracy]

# Plotting in graph
plt.bar(labels, accuracy, color=['blue', 'green'])
plt.title('Training vs Test Accuracy')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.show()




"""**Random Forest Classifier**"""

#importing module
from sklearn.ensemble import RandomForestClassifier

#initializing
rfc = RandomForestClassifier(max_depth=10)

#fitting in module
rfc.fit(X_train, Y_train)

#lets find out the accuracy score on training data

Y_train_pred = rfc.predict(X_train)

#lets compare predicted value by our model with the original train Y_train
training_accuracy = accuracy_score(Y_train_pred, Y_train)


#lets find out the accuracy score on test data

Y_test_pred = rfc.predict(X_test)

#lets compare predicted value by our model with the original train Y_test
test_accuracy = accuracy_score(Y_test_pred, Y_test)

##calculating confusion matrix  and classification report for train
conf_matrix_train = confusion_matrix(Y_train, Y_train_pred)
cls_report_train = classification_report(Y_train, Y_train_pred)

##calculating confusion matrix  and classification report for test
conf_matrix_test = confusion_matrix(Y_test, Y_test_pred)
```

```
cls_report_test = classification_report(Y_test, Y_test_pred)

#printing
print('Train Accuracy:', round(training_accuracy*100,2))
print('Test Accuracy:', round(test_accuracy*100,2))

##for train
print("confusion matrix:\n", conf_matrix_train)
print("classification report:\n", cls_report_train)

##for test
print("confusion matrix:\n", conf_matrix_test)
print("classification report:\n", cls_report_test)


# Data to plot
labels = ['Training Accuracy', 'Test Accuracy']
accuracy = [training_accuracy, test_accuracy]

# Plotting in graph
plt.bar(labels, accuracy, color=['red', 'orange'])
plt.title('Training vs Test Accuracy')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.show()




"""#Feature Importance

**Using Random Forest Classifier for feature importance:**
"""

#importing necessary library
import seaborn as sb

#initializing
rfc = RandomForestClassifier()

#fitting in module
rfc.fit(X_train, Y_train)

# View a list of the features and their importance scores
print('\nFeatures Importance:')
```

```
feature_imp = pd.DataFrame(zip(X.columns.tolist(), rfc.feature_importances_ * 100),
columns=['Feature', 'Importance'])

feature_imp

# Features importance plot
plt.figure(figsize=[20,6])
sb.barplot(data=feature_imp, x='Feature', y='Importance', palette = 'viridis')
plt.title('Features Importance', weight='bold', fontsize=25)
plt.xlabel('Feature', weight='bold', fontsize=13)
plt.ylabel('Importance (%)', weight='bold', fontsize=13);


# add annotations
impo = feature_imp['Importance']
locs, labels = plt.xticks()

#visualizing percentage at the top-center of the bar
for loc, label in zip(locs, labels):
   count = impo[loc]
   pct_string = '{:0.2f}%'.format(count)

   plt.text(loc, count-0.8, pct_string, ha = 'center', color = 'w', weight='bold')




"""##Feature Selection"""

#importing necessary libraries

from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import RandomOverSampler
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_predict

loan_data.head(2)

# Scalling
num_values = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Credit_History']
scaler = MinMaxScaler()
loan_data[num_values] = scaler.fit_transform(loan_data[num_values])

# Create dummy variables
dummy_var = pd.get_dummies(loan_data)
```

```
# Partioning
X_f = dummy_var.drop('Loan_Status', axis=1)
y_f = dummy_var['Loan_Status']

# Oversampling
ros = RandomOverSampler(random_state = 42)
X_resampled, Y_resampled = ros.fit_resample(X_f, y_f)

rfc = RandomForestClassifier()

#making cross-validation prediction
Y_pred = cross_val_predict(rfc, X_resampled, Y_resampled, cv=10)

#calculating confusion matrix and classification report
conf_matrix = confusion_matrix(Y_resampled, Y_pred)
cls_report = classification_report(Y_resampled, Y_pred)

#printing
results = cross_validate(rfc, X_resampled, Y_resampled, cv=10, scoring=['accuracy'],
return_train_score=True)
print("Accuracy:", np.round(np.mean(results['test_accuracy']*100),2))
print("confusion matrix:\n", conf_matrix)
print("classification report:\n", cls_report)




"""**Using Decision Tree Classifier for feature importance:**"""

#initializing
dtc = DecisionTreeClassifier(max_depth=5)

#fitting in module
dtc.fit(X_train, Y_train)

# View a list of the features and their importance scores
print('\nFeatures Importance:')
feature_imp = pd.DataFrame(zip(X.columns.tolist(), dtc.feature_importances_ * 100),
columns=['Feature', 'Importance'])

feature_imp

# Features importance plot
plt.figure(figsize=[20,6])
sb.barplot(data=feature_imp, x='Feature', y='Importance', palette = 'viridis')
plt.title('Features Importance', weight='bold', fontsize=25)
```

```
plt.xlabel('Feature', weight='bold', fontsize=13)
plt.ylabel('Importance (%)', weight='bold', fontsize=13);


# add annotations
impo = feature_imp['Importance']
locs, labels = plt.xticks()

#visualizing percentage at the top-center of the bar
for loc, label in zip(locs, labels):
   count = impo[loc]
   pct_string = '{:0.2f}%'.format(count)

   plt.text(loc, count-0.8, pct_string, ha = 'center', color = 'w', weight='bold')


# Scalling
num_values = ['ApplicantIncome', 'LoanAmount', 'Credit_History']
scaler = MinMaxScaler()
loan_data[num_values] = scaler.fit_transform(loan_data[num_values])

# Create dummy variables
dummy_var = pd.get_dummies(loan_data)

# Partioning
X_f = dummy_var.drop('Loan_Status', axis=1)
y_f = dummy_var['Loan_Status']

# Oversampling
ros = RandomOverSampler(random_state = 42)
X_resampled, Y_resampled = ros.fit_resample(X_f, y_f)

dtc = DecisionTreeClassifier(max_depth=5)

#making cross-validation prediction
Y_pred = cross_val_predict(dtc, X_resampled, Y_resampled, cv=10)

#calculating confusion matrix and classification report
conf_matrix = confusion_matrix(Y_resampled, Y_pred)
cls_report = classification_report(Y_resampled, Y_pred)

#printing
results = cross_validate(rfc, X_resampled, Y_resampled, cv=10, scoring=['accuracy'],
return_train_score=True)
print("Accuracy:", np.round(np.mean(results['test_accuracy']*100),2))
print("confusion matrix:\n", conf_matrix)
print("classification report:\n", cls_report)
```