

PREDICTING THE LOAN DELINQUENCY RATE OF THE CUSTOMERS

Submitted in partial fulfilment of the requirements for the award of

MASTER OF BUSINESS ADMINISTRATION

by

SEDHUPATHY S

REGISTER NO:42410260



**DEPARTMENT OF BUSINESS ADMINISTRATION
SCHOOL OF MANAGEMENT STUDIES**

SATHYABAMA

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

**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

DEC 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

[DEEMED TO BE UNIVERSITY]

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF MANAGEMENT STUDIES

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SEDHUPATHY.S 42410260** who carried out the Project Work "**PREDICTING THE LOAN DELINQUENCY RATE OF THE CUSTOMERS**" under my supervision from July 2023 to December 2023.

Dr. A.S. PRINCY

Internal guide

External Guide

Dr. BHUVANESWARI .G, MBA., Ph.D
Dean – School of Management Studies

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I **SEDHUPATHY.S (42410260)** hereby declare that the Project Report entitled **“PREDICTING THE LOAN DELINQUENCY RATE OF THE CUSTOMERS”** done by me under the guidance of **DR.A.S. PRINCY** is submitted in partial fulfilment of the requirements for the award of Master ofBusiness Administration degree.

DATE:

PLACE:

SEDHUPATHY.S

ACKNOWLEDGEMENT

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

I convey my sincere thanks to **Dr. G. Bhuvaneswari, MBA., Ph.D., Dean - School of Management Studies** and **Dr. A. Palani, M.Com., M.Phil., M.B.A., Ph.D., - Head - School of Management Studies** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **DR.A.S. PRINCY** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **School of Management Studies** who were helpful in many ways for the completion of the Training.

SEDHUPATHY.S

TABLE OF CONTENT

S.No	Topic	Page No.
1	Executive Summary 1.1 Domain 1.2 Problem statement, Scope and Objective 1.3 Data source, rows & columns, Missing values count, Outliers count and data description 1.4 Models, Tools and Language used 1.5 Comparison of Model Performance	7
2	Literature Survey and Detailed Report 2.1 Introduction 2.2 Review of Literature 2.3 Data Source and Description 2.4 Data pre-processing 2.5 Exploratory data analysis 2.6 Model Building 2.7 Model Evaluation 2.8 Model Inference	10
3	Business Insights and Recommendations	40
4	References and Bibliography	41
5	Appendix	42

LIST OF FIGURES

Fig no.	Content
1	Distance plot
2	Bin values
3	Pie chart: home_ownership
4	Pie chart: term
5	Pie chart: purpose of loan
6	Count plot: gender
7	Count plot: FICO
8	Count plot: isDelinquent
9	Bar plot
10	Joint plot
11	kdeplot
12	Cross tab: term, purpose
13	Cross tab: isDelinquent, age
14	Cross tab: gender, home_ownership
15	Count plot: purpose, isDelinquent
16	Count plot: home_ownership, isDelinquent
17	Count plot: term, isDelinquent
18	Count plot: FICO, isDelinquent
19	Heat map
20	Pair plot: loan
21	Pair plot: isDelinquent
22, 23, 24	Logistic Regression, Decision Tree, Random Forest
25, 26	Confusion matrix of Logistic Regression on train and test data
27, 28	Confusion matrix of Decision Tree on train and test data
29, 30	Confusion matrix of Random Forest on train and test data

Chapter 1: EXECUTIVE SUMMARY

- The delinquency rate refers to the percentage of loans that are past due. It indicates the quality of a lending company's or a bank's loan portfolio.
- The delinquency rate is commonly used by analysts to determine the quality of the loan portfolio of lending companies or banks. It compares the percentage of loans that are overdue to the total number of loans. A lower rate is always desirable, as it indicates that there are fewer loans in the lender's loan portfolio that are paying outstanding debt late.
- In the industry, lenders typically do not label a loan as being delinquent until the loan is 60 days past due. However, the figure is not absolute and varies from lender to lender. For example, one lender may consider a 30-day overdue loan as delinquent while another lender may only consider a 45-day overdue loan as delinquent.
- When a loan is labelled as delinquent, lending companies generally work with third-party collection agencies to recover the loan. If the delinquent loan is unable to be recovered after an extended period of time, it is written off by the lender.

1.1 Domain explanation

The term delinquent refers to the state of being in arrears. When someone is delinquent, they are past due on their financial obligation(s), such as a loan, credit card, or bond payments. This means a borrower's payments are not made to satisfy their debt(s) in a timely manner. Delinquent entities can be either individuals or corporations. Financial delinquency often leads to default if the arrears are not brought up to date. The term may also be used to describe the failure to perform a duty by financial professionals.

Loans work a little differently than other types of debt. When you sign up for a loan, you agree to repay the lender a specific amount of money at regular intervals until the debt is paid off. The lender determines the due date and, in some cases, may allow you to set this date based on your personal financial situation.

Most lenders also include a grace period, which may be a few days after your due date. If you make your payment on or before this date, it may not be considered late but you may still incur interest but not a late payment fee. If you fail to make the payment before, you are considered delinquent. Your loan is in delinquent status even if you make your payment a day or two after the due date.

1.2 Problem statement, scope of study and objectives

Problem statement:

To identify the criteria to approve loans for an individual customer such that the likelihood of the loan delinquency is minimized.

Scope of the study:

This study investigates the determinants of individual loan and loan delinquent rate. The delinquency rate is commonly used by analysts to determine the quality of the loan portfolio of lending companies or banks. It compares the percentage of loans that are overdue to the total number of loans.

Objective:

Our primary motives are to analyze the dataset, perform feature engineering to come up with suitable independent features and building a good model that will help us in predicting the loan delinquency rate of the customers.

1.3 Data source, rows & columns, Missing values count and data description

Source of data

The dataset was taken from Kaggle (<http://www.kaggle.com/>). **Kaggle** is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

Rows and Columns

There are about 11548 rows and 8 columns

Missing values

There are no missing values in the dataset.

Data Description

Our data contains two different data types i.e., integer and categorical objects

1.4 Models, Tools and language used:

Models:

- Logistic regression model
- Random Forest model
- Decision tree

Tools:

Anaconda Navigator, Jupyter notebook

Languages used:

Python

1.5 Comparison of Model Performance

The evaluation metrics are accuracy, precision, re-call, f1 score

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)$$

$$Precision = TP / (TP + FP)$$

$$Recall = TP / (TP + FN)$$

$$F1Score = 2 * ((precision * recall) / (precision + recall))$$

- Where TP is True Positive
- Where TF is True Negative
- Where FP is False Positive
- Where FN is False Negative

Chapter 2: LITERATURE SURVEY AND DETAILED REPORT

2.1 Introduction

The term delinquent refers to the state of being in arrears. When someone is delinquent, they are past due on their financial obligation(s), such as a loan, credit card, or bond payments. This means a borrower's payments are not made to satisfy their debt(s) in a timely manner. Delinquent entities can be either individuals or corporations. Financial delinquency often leads to default if the arrears aren't brought up to date. The term may also be used to describe the failure to perform a duty by financial professionals.

Loans work a little differently than other types of debt. When you sign up for a loan, you agree to repay the lender a specific amount of money at regular intervals until the debt is paid off. The lender determines the due date and, in some cases, may allow you to set this date based on your personal financial situation.

Most lenders also include a grace period, which may be a few days after your due date. If you make your payment on or before this date, it may not be considered late but you may still incur interest but not a late payment fee. If you fail to make the payment before, you are considered delinquent. Your loan is in delinquent status even if you make your payment a day or two after the due date.

2.2 Review Of Literature

The issue of loan defaults has recently gained significant attention in both financial institutions and the scientific community. With the growth of data and the advancement of machine learning, numerous authors have studied the topic by constructing various models and evaluating their performance.

- In the paper of (Tariq, et al., 2019), it is provided a comprehensive examination of the ML process and the creation of a predictive model for loan defaults, using the SEMMA (Sample, Explore, Modify, Model and Assess) methodology. In this paper, the data is obtained from the well-known platform, Kaggle. The first stage is sampling, or in other words the selection of the data from the initial dataset, with which the model will be trained. It is worth mentioning that this stage is optional. The second phase is exploration, where the data which the work will be done is understood and explored for patterns. The dataset contains about 1K records and 13 attributes in this paper. The type of attributes was explored, where 4 of them are continuous numerical and the others are categorical. In this phase, there were made statistical analysis, various visualizations such as histogram, scatter plot, box plot, distribution of variables in tabular and visual format such as bar chart, etc. The 3rd phase involves data pre-processing, where the data is

processed and adapted for modeling. In this paper, the transformation of the data type was done at first. All categorical data has been converted to numeric ones, in order that some algorithms could perform better. Then, the missing values were treated by replacing them with the mean method. In the modeling phase, 3 different algorithms were used: Decision Tree, Logistic Regression and Neural Network. The last stage is the performance evaluation. This phase was implemented in parallel with the modeling phase, where different operators were used, to see which ones show the best performance. The ROC curve, Accuracy Sensitivity, Specificity and Error were taken as performance evaluation measures. According to the Accuracy metric, the best performing algorithm was Neural Network with 83%, leaving behind Logistic Regression with 81% and Decision Tree with 80%. However, in this paper, the algorithm with the best performance was chosen Logistic Regression, since it was the best in predicting "True Negatives" or negative values (how accurate is the model for predicting loans that default), which is the most important issue.

- The paper (Alomari, 2017), also explains the methodology and phases involved in predicting loan defaults. The data used in the study was obtained from "Lending Club", a publicly accessible online lending platform. This dataset contains information about customer characteristics and loan repayment history. To begin, the dataset was firstly explored to understand what data would be used and what the dependent variable was. It is worth mentioning, that the dependent variable in this paper and other papers reviewed was the variable indicating whether the loan had defaulted or not, which typically has two values (0 and 1 representing whether the loan has defaulted or not). Afterwards, descriptive statistics analysis was performed on the dataset, analyzing the central tendencies and data distribution of the main numerical variables through various visualizations such as histograms, bar charts, and scatter plots. Next, the data was pre-processed to make it suitable for the chosen ML algorithms. In this study, the dataset of 188K records was reduced to 40K records and from 52 variables to 20 variables through various methods such as removing records and variables with a large number of missing data, replacing missing data, and removing irrelevant variables. Additionally, the "Filter Attribute Selection" method was used to eliminate variables that had little or no effect on the dependent variable, allowing the models to run faster without sacrificing performance. After the data was prepared, ML algorithms were applied and the model was trained. Several algorithms were used in this study, including Naive Bayes, Decision Tree, kNN, 1R, Neural Network, and ensemble models like Random Forest. Upon evaluating the performance of the models, Random Forest was found to be the most successful with a performance of 71.75% as measured by Accuracy.
- Another paper, (Zhu, et al., 2019), deals with the same problem and uses the same dataset from the "Lending Club" platform, where data was taken for the first quarter of 2019. In this paper, an initial exploration of the dataset is also done to understand what data will be worked with. Unlike the (Alomari, 2017) study, this paper doesn't give much attention to the exploration process. After this phase, the data preparation begins, where the original dataset with 102 variables is reduced to 15. The main method used for selecting the most important variables is a similar operator to "Filter Attribute Selection" called "Recursive Feature Elimination". This operator works based on the logic of removing all variables that have a low correlation with the dependent variable. In addition to this operator,

many other data preparation methods were used, such as replacing missing values with the mean, transforming the values of the dependent variable into only 2 values (0 and 1) because there were more value types, etc. After the data is processed, different ML models were applied, such as Decision Tree, SVM, Logistic Regression, and Random Forest as an ensemble model. What sets this study apart, is that the SMOTE method was used together with the Random Forest algorithm. This method addresses the issue of data imbalance. In the initial dataset, the dependent variable was distributed with 99% successful loans and only 1% that have defaulted. In such cases, models usually don't give the desired accuracy due to the data imbalance. After training each model, the highest accuracy was achieved by the Random Forest algorithm along with the SMOTE method with an "Accuracy" of 98% and an F1 Score of 98%.

- The issue of data imbalance regarding the dependent variable was addressed in (Zhou & Wang, 2012), which focuses mainly on "Random Forest" algorithm. The data for this study was obtained from the Kaggle platform, a customer dataset with loan defaults. The dataset contains around 150K records and 12 attributes. There is no explanation of how the data processing/transformation was carried out. The algorithm used in this work is "Random Forest," where the authors develop an improved Random Forest algorithm (Algorithm 2) in addition to the normal Random Forest (Algorithm 1). This algorithm is an ensemble model that works by creating many Decision Trees and distributing the data among these trees based on the sample, and the prediction/training is made based on the majority vote. Meanwhile, the improved algorithm developed by the authors provides that the votes collected from the Decision Trees are weighted based on their importance. Furthermore, the SMOTE and Samp-size methods are used to address the data imbalance issue. The SMOTE method increases the number of records for smaller data by creating new records based on existing data. On the other hand, the Samp-size method is based on the stratified sampling that Favors smaller data during the model training. Then, these models were trained and the performance was evaluated for both Random Forest algorithms, as well as for the SVM, KNN, and Decision Tree algorithms. The results show that the proposed/developed Random Forest has the highest accuracy of 93.66% compared to the normal Random Forest with 93.58%. To assess the complete model performance, it is also necessary to evaluate the minor class, i.e., True Negatives, to see how well these cases have performed. The Balanced Accuracy was used as a measure of this aspect. For addressing this issue, the SMOTE and Samp-size methods were used, and they were very successful in significantly increasing the Balanced Accuracy. There was no significant difference in terms of which method was more accurate, as they were very similar.
- Also, in the study of (Iain & Mues, 2012), the issue of loan default is addressed by taking into consideration that the dependent variable is imbalanced. Five different datasets, collected from various state institutions, were analysed and modelled in this research. Initially, all the algorithms that are used are explained. Then, the datasets are explored and the data is described. Next, the authors test how different algorithms perform on these datasets by changing the data balance of the dependent variable. Initially, the dataset was divided into 70/30 balance between 2 values up to a balance of 99/1 of the target variable values. After applying different models, it was observed that the performance of algorithms varies depending on the balance of the dependent variable. Performance in this

work is measured through the AUC metric. According to the results, it is observed that in cases where the dependent variable was imbalanced, Random Forest and Gradient Boosting performed the best. Meanwhile, when the dependent variable was balanced at a 70/30 level, the SVM algorithm performed the best. It should be noted that 10 different algorithms were analysed and tested during this study.

- The paper of (Victor & Raheem, 2021), is based on the "Genetic Algorithm" method, which selects the most important variables impacting the dependent variable during the building of the ML model. The problem of loan defaults is also addressed, based on a dataset obtained from the Kaggle platform on the Lending Club company dataset. The original dataset consists of approximately 2.2 million records and 151 variables resulting from 2007 to 2018. In this study, only the data from the years 2015 and 2016, totalling 855K records and the same number of variables, were used as a sample. The methodology used in this study starts with data sampling, continuing with exploration and processing, followed by division into training and test data, and finally, the building and evaluating of the model. Initially, the data is explored to gain a better understanding. The exploration in this study focuses mainly on the description of each variable. After this step, the data is processed to be ready for ML model training. Data processing of this study includes: removing variables with more than 30% missing data, replacing missing data with the mean for numerical data and mode for categorical data, transforming categorical data into numerical data, removing outliers, and normalizing the data. The "Genetic Algorithm" method is used as the main method for selecting input variables in the model. This method is based on Darwin's theory of evolution, which after many calculations selects the variables that have the most impact on the dependent/target variable. Logistic Regression, Random Forest, and SVM algorithms were used for predictive modelling. Accuracy, AUC, MCC, Precision, Recall, and F1 score were used as model performance metrics. In most of these metrics, the most successful algorithm was Random Forest, achieving 82% F1 score, leaving behind Logistic Regression at 72% and SVM at 71%.
- In another study (Bayraci & Susuz, 2019), the issue of loan default prediction was addressed. The authors used two datasets, collected from commercial banks in Turkey, one with 80K records and the other with approximately 500K. Both datasets had a significant number of variables, 64 and 60 respectively. The authors briefly described the data preparation process, without going into much detail. Then, they explained the training of various algorithms. An interesting aspect of the paper is that the datasets were divided into two categories, one showing the client's loan application data and the other showing the loan performance data. In the dataset containing loan application data, the "Deep Neural Network" algorithm achieved the highest accuracy at 85.7%, surpassing the "Decision Tree" at 82.3%, "Logistic Regression" at 78%, "SVM" at 77.9% and "Naive Bayes" at 75.2%. On the other hand, in the dataset with loan performance data, the SVM algorithm showed the best results with 78.1%, followed by the Deep Neural Network at 77.9%.

- The issue of loan default prediction has not only been explored in scientific papers, but also in master's level papers. One such paper is (Adelabu, 2021). This study has taken the data from the "UCI Machine Learning Repository" website, which includes around 1,000 records with 20 variables. An exploration of the data was carried out initially, where each variable was analysed through visualizations and compared with the dependent variable "Default". The paper does not provide much information about the data processing phase, only stating that the data was normalized prior to applying the algorithms. The author used three algorithms for building the model: Random Forest, Logistic Regression, and Decision Tree. The paper also outlines the performance metrics like Accuracy, Precision, Recall, and F1 score in advance. Based on the author, the F1 score is the best metric for evaluating the models, as it takes into account the accuracy across all predicted classes. After the models were trained, the results were compared, and Random Forest resulted as the best algorithm with an F1 score of 87%, followed by Logistic Regression with 85% and Decision Tree with 82%.
- Data pre-processing/transformation is one of the main challenges in building an ML model as it directly impacts the training and performance of the model. One aspect of data processing is handling with missing values. This issue has been thoroughly addressed in the master's thesis by (Martin, 2015), which focuses on missing values in city data (Open City Data). The data was gathered from various web pages of different cities by extracting it and uploading it into a database. During this process, data processing was done by removing duplicates, transforming it into tabular form, cleaning/removing records and variables (when a record or variable had more than 90% of missing values). After these steps, there were about 683K records remaining, with 46% of them being with missing values. To handle the missing values problem, the author used KNN, Linear Regression, and Decision Tree algorithms. These algorithms were used in two stages. The first stage was "Building Complete Subsets" or creating smaller datasets based on predicting missing values through variables that have the most correlation with the dependent variable, which was the missing value in this case. The second approach, Principal Component Regression, predicted missing values by reducing the size of variables and keeping the most important ones that impact missing values. After applying these algorithms, it was determined that the first approach was more accurate but performed poorly when there were many variables with missing data. On the other hand, the second approach was less accurate compared to the first but was more suitable for forecasting when there were numerous missing variables.

2.3 Data Source and Description

Source of data

The dataset was taken from Kaggle (<http://www.kaggle.com/>). **Kaggle** is an online community of data scientists and machine learning practitioners. Kaggle

allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

Data Description

Our data contains two different data types i.e., integers and categorical variables. Since some of our categorical variables are in string format. So, we cannot pass this variable to our model directly so we must use one hot encoding to convert it into numerical variable having binary integers 0 and 1.

Data Dictionary

- ID: Customer ID
- isDelinquent : indicates whether the customer is delinquent or not (1 => Yes, 0 => No)
- term: Loan term in months
- gender: Gender of the borrower
- age: Age of the borrower
- purpose: Purpose of Loan
- home_ownership: Status of borrower's home
- FICO: FICO (i.e., the bureau score) of the borrower

2.4 Data pre-processing

Import python Libraries

- Pandas for dataframes
- Numpy for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions
- Matplotlib for visualization
- Seaborn for visualization

Import the data set

- `pd.read_csv` – This is used to load the data into the ML Model

2.5 Exploratory data analysis

In this step, we will perform the below operations to check what the data set comprises of. We will check the below things:

- Head of the dataset - Head function will tell you the top records in the data set. By default, python shows you only top 5 records.
- Shape of the dataset - Shape attribute tells us number of observations and variables we have in the data set. It is used to check the dimension of data. The Loan Delinquent dataset has 11548 observations and 8 variables in the data set.
- Info of the dataset - `info()` is used to check the Information about the data and the datatypes of each respective attributes & to find count.
- Summary of the dataset – The describe method will help to see how data has been spread for the numerical values. We can clearly see the minimum value, mean values, different percentile values and maximum values.

Check for Duplicate records

- Since there are no Duplicate values in any of the columns in the dataset. No need for treating the missing values.

Check for missing value in any column

- Since there are no missing values in any of the columns in the dataset. No need for treating the missing values.

Converting the categorical variables to numerical value

- Since the term, gender, purpose, homeownership, age and FICO are object datatype so for the analytics purpose the object variables are been converted into numerical values such as 0 's and 1 's using one-hot encoding.

DATA VISUALIZATION

UNIVARIATE:

Univariate analysis of numerical values – distance plot

1. `sns.distplot(x=loan['purpose_cat'])`

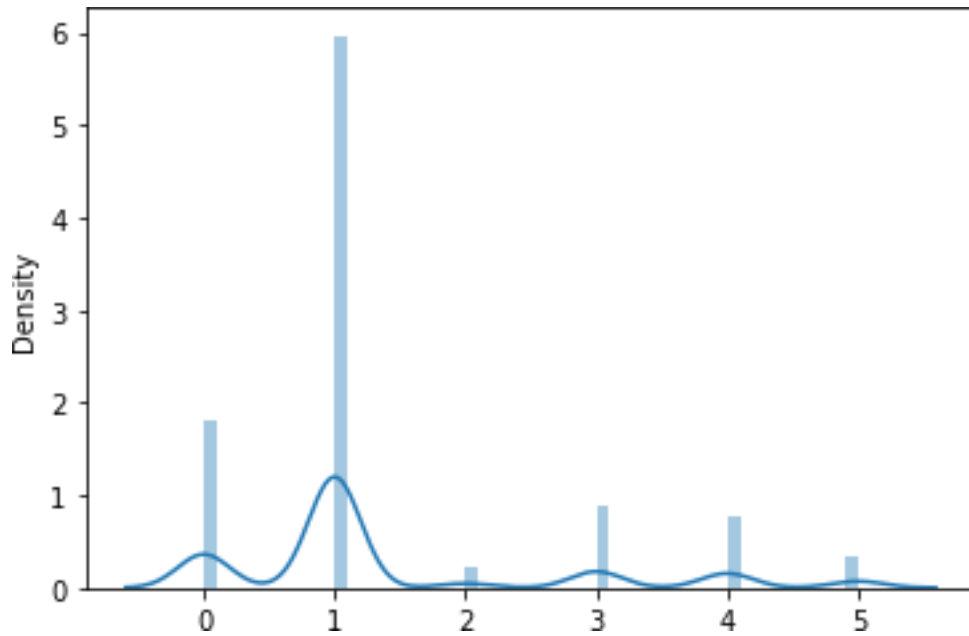


Fig.no: 1

Inference:

most loan applicants are for house loans (59%) followed by car loans

(18%) Univariate analysis of categorical values – bin, pie chart, count plot

2. `bin_values = np.arange(start = 0 , stop = 3, step = 1)`
`loan['age'].hist(bins=bin_values)`

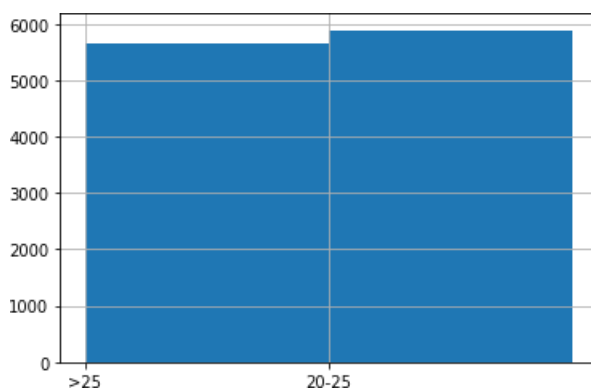


Fig.no: 2

Inference:

almost an equal percentage of people aged 20 - 25 and >25 have applied for loan

```
3. x = plt.figure().subplots()
```

```
loan.home_ownership.value_counts().plot(x=x,  
kind='pie',autopct='%1.1f%%')x.set_ylabel("")  
x.set_title("home_ownership")plt.show()
```

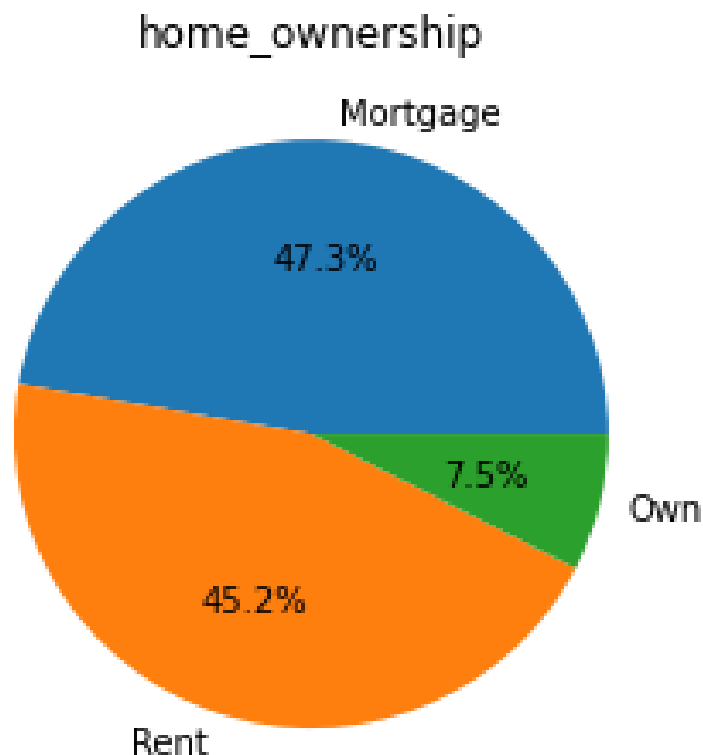


Fig.no: 3

Inference:

very few applicants own their house

```
4. x = plt.figure().subplots()  
loan.term.value_counts().plot(x=x,  
kind='pie',autopct='%1.1f%%')x.set_ylabel("")  
x.set_title("term")plt.show()
```

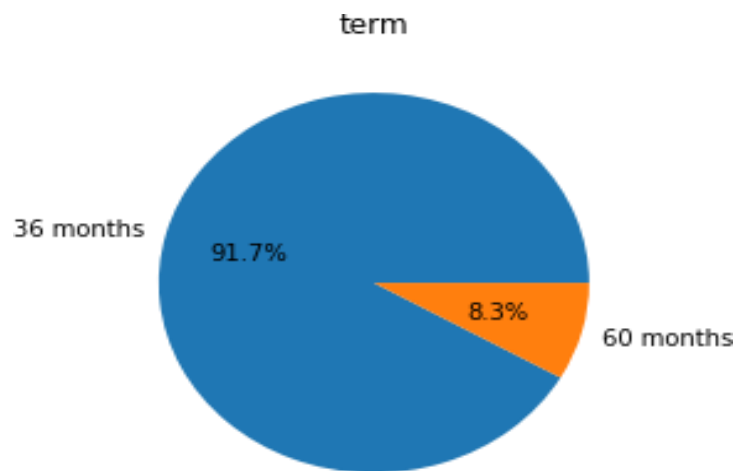


Fig.no: 4

Inference:

91.7% of the loans are for a 36-month term

```
5. x = plt.figure().subplots()
   loan.purpose.value_counts().plot(x=x,
   kind='pie',autopct='%1.1f%%')x.set_ylabel("")
   x.set_title("purpose of
   loan")plt.show()
```

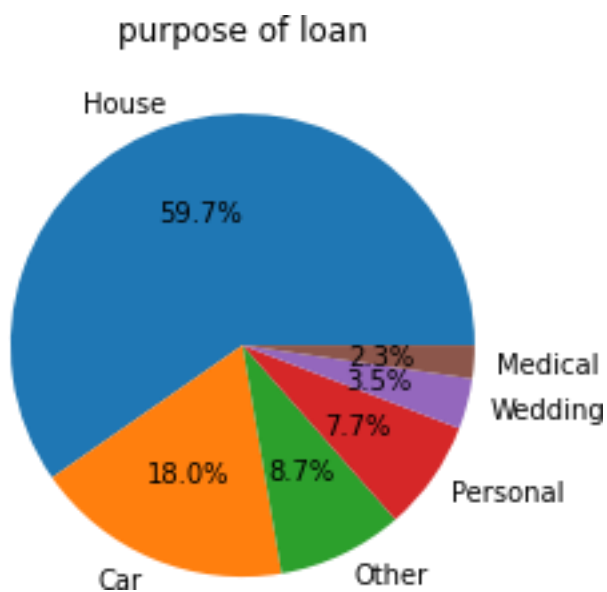


Fig.no: 5

Inference:

most loan applicants are for house loans (59%)

```
6. sns.countplot(x=loan['gender'])plt.show()
```

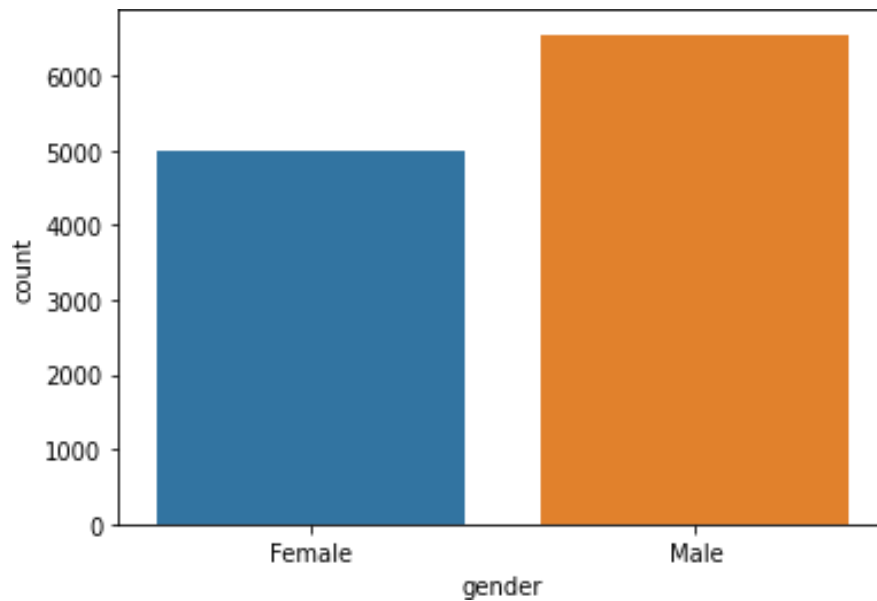


Fig.no: 6

Inference:

There are more male applicants (56.8%) than female applicants (43.2%)

```
7. sns.countplot(x=loan['FICO'])plt.show()
```

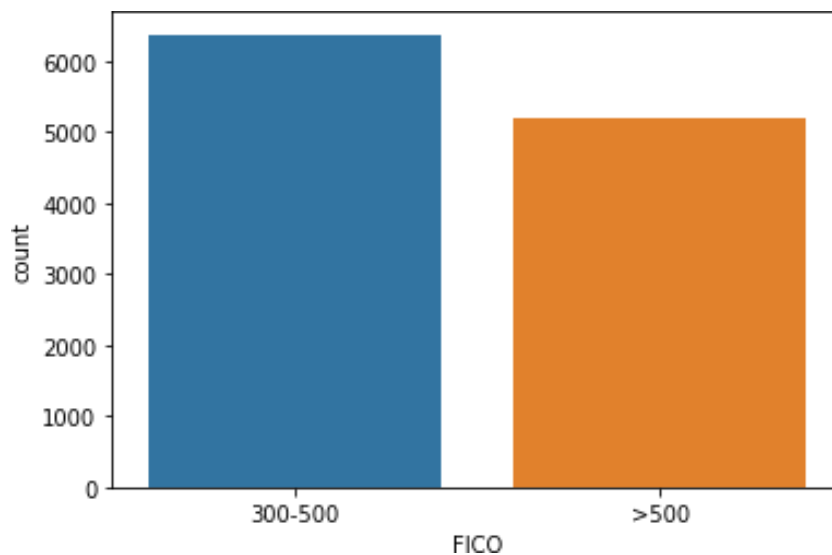


Fig.no: 7

Inference:

most customers have a FICO score between 300 and 500 (55.2%) followed by a score of greater than 500 (44.8%)

```
8. sns.countplot(x=loan['isDelinquent'])plt.show()
```

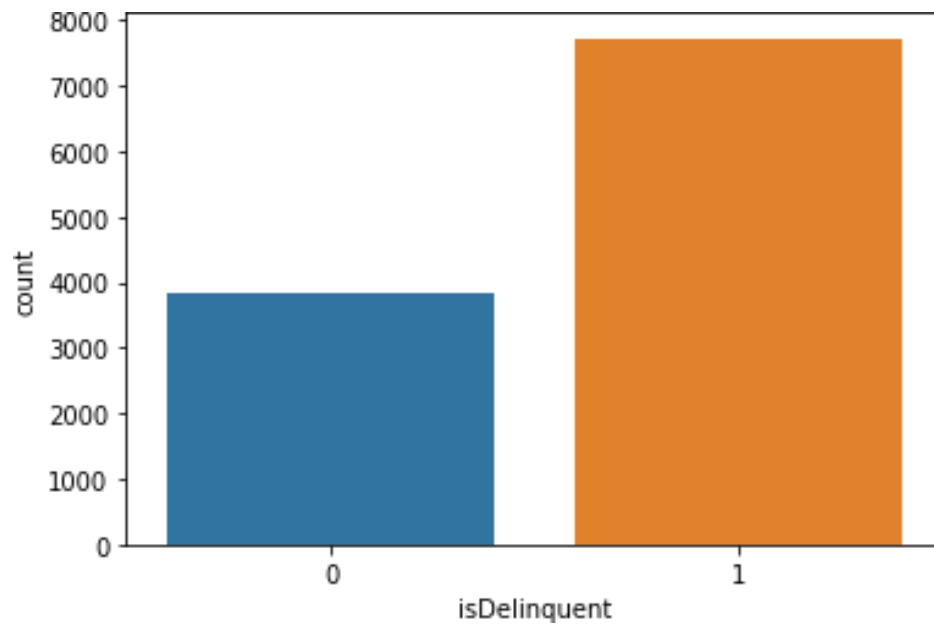


Fig.no: 8

Inference:

66% of the customers are delinquent

BIVARIATE ANALYSIS:

Bivariate analysis of categorical values – bar plot

1. `sns.barplot(x='age_cat',y='isDelinquent',data=loan)`

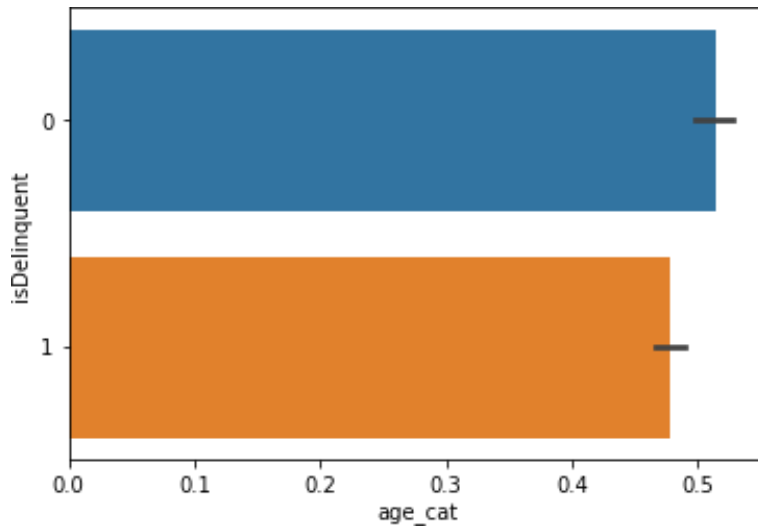


Fig.no: 9

Inference:

almost an equal percentage of people aged 20 - 25 and >25 have

applied for loan

2. `sns.jointplot(x='purpose_cat',y='isDelinquent',data=loan)`
`plt.show()`

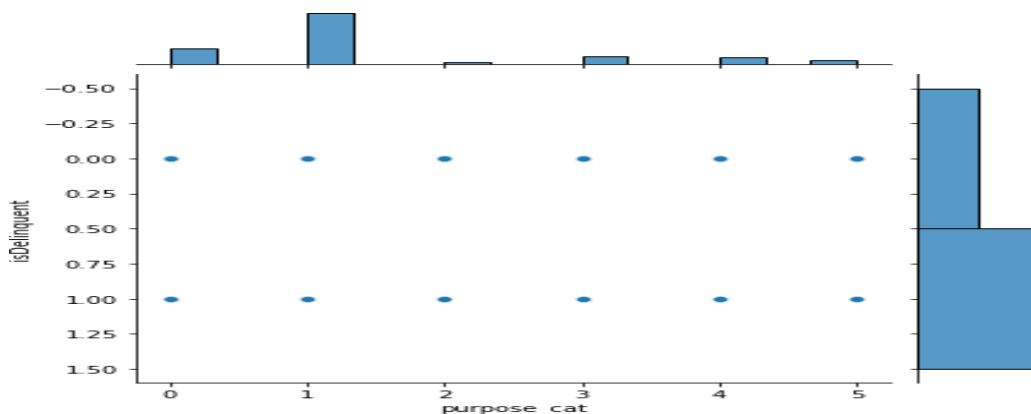


Fig.no: 10

Inference:

most loan applicants are for house loans (59%) followed by car loans (18%) and 66% of the customers are delinquent

3. `sns.kdeplot(x='home_ownership_cat',hue='isDelinquent',data=loan)`

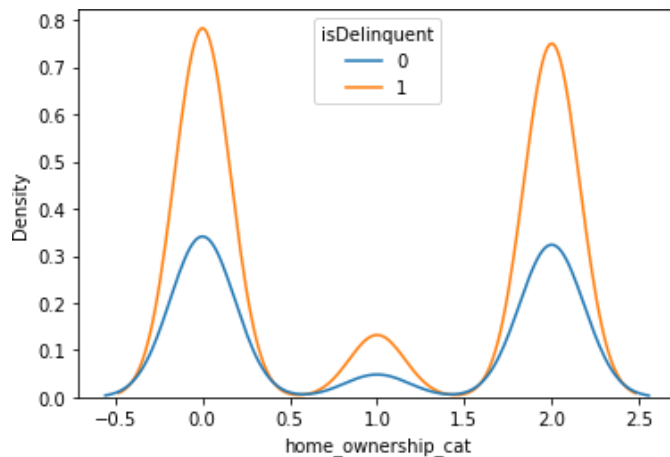


Fig.no: 11

Inference:

very few applicants <10% own their house, most customers have either mortgaged their houses or live on rent and mortgaged have delinquent

bivariate analysis of categorical values – crosstab, count plot

4. `pd.crosstab(loan.term, loan.purpose, margins=True).style.background_gradient(cmap='copper')`

purpose	Car	House	Medical	Other	Personal	Wedding	All
term							
36 months	1922	6339	188	928	830	382	10589
60 months	158	553	78	82	62	26	959
All	2080	6892	266	1010	892	408	11548

Fig.no: 12

Inference:

91.7% of the loans are for a 36-month term and most loan applicants are for house loans(59%) followed by car loans (18%)

5. `pd.crosstab(loan.isDelinquent, loan.age, margins=True).style.background_gradient(cmap='copper')`

	age	20-25	>25	All
isDelinquent				
0		1858	1969	3827
1		4030	3691	7721
All		5888	5660	11548

Fig.no: 13

Inference

:

almost an equal percentage of people aged 20 - 25 and >25 have applied for loan and cause for delinquent

```
6.pd.crosstab([loan.gender, loan.home_ownership],
margins=True).style.background_gradient(cmap='copper')
```

home_ownership	Mortgage	Own	Rent	All
gender				
Female	2461	372	2160	4993
Male	3000	499	3056	6555
All	5461	871	5216	11548

Fig.no: 14

Inference:

very few applicants own their house and there are more male applicants (56.8%) than female applicants (43.2%)

```
7. sns.countplot(x=loan[("purpose")],hue=loan[("isDelinquent")])
```

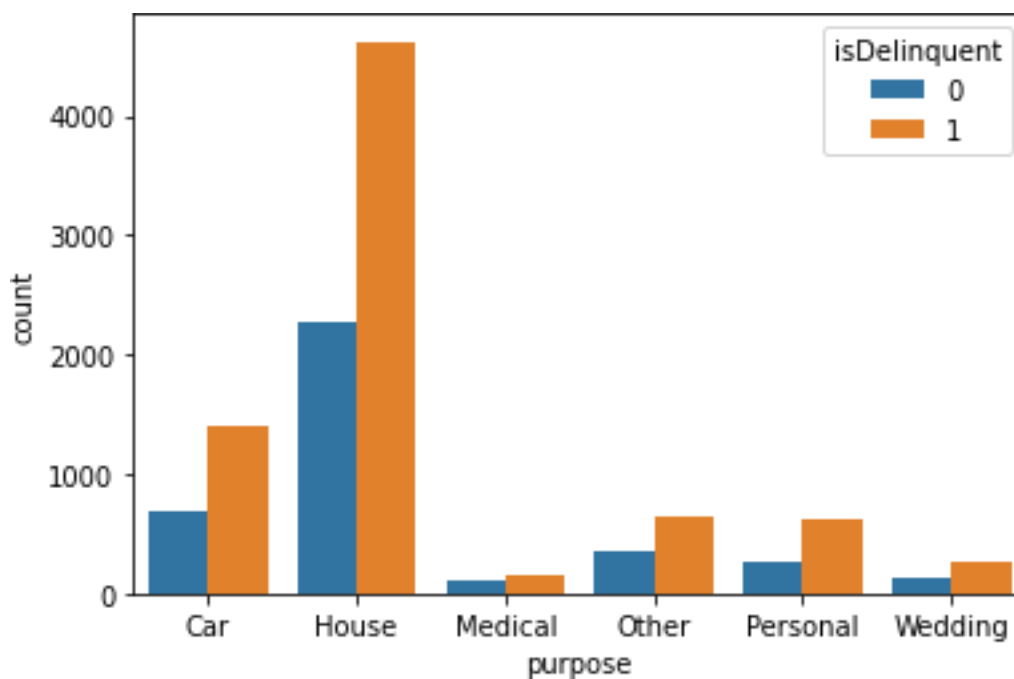


Fig.no: 15

Inference:

most loan applicants are for house loans (59%) and risk of delinquent

8. `sns.countplot(x=loan[("home_ownership")],hue=loan["isDelinquent"])`

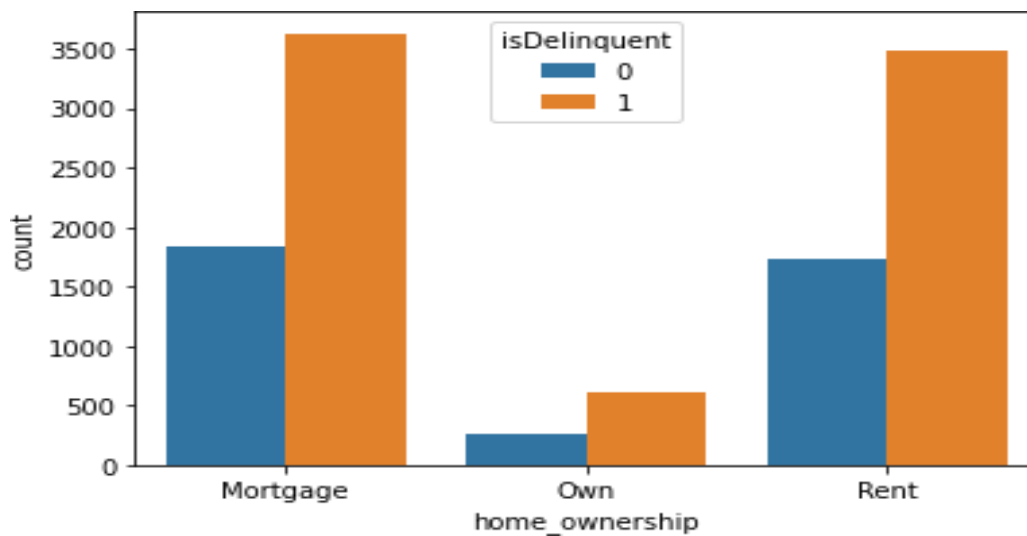


Fig.no: 16

Inference:

very few applicants own their house and likely less causing delinquent

9. `sns.countplot(x=loan[("term")],hue=loan["isDelinquent"])`

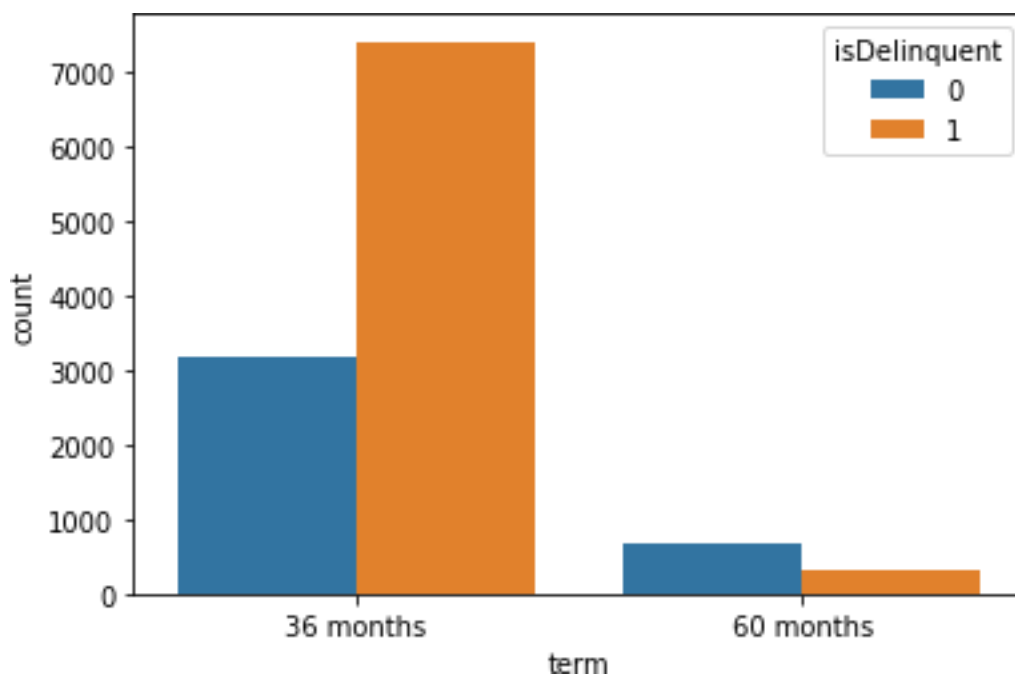


Fig.no: 17

Inference:

91.7% of the loans are for a 36-month term and causing delinquent

```
10. sns.countplot(x=loan["FICO"],hue=loan["isDelinquent"])
```

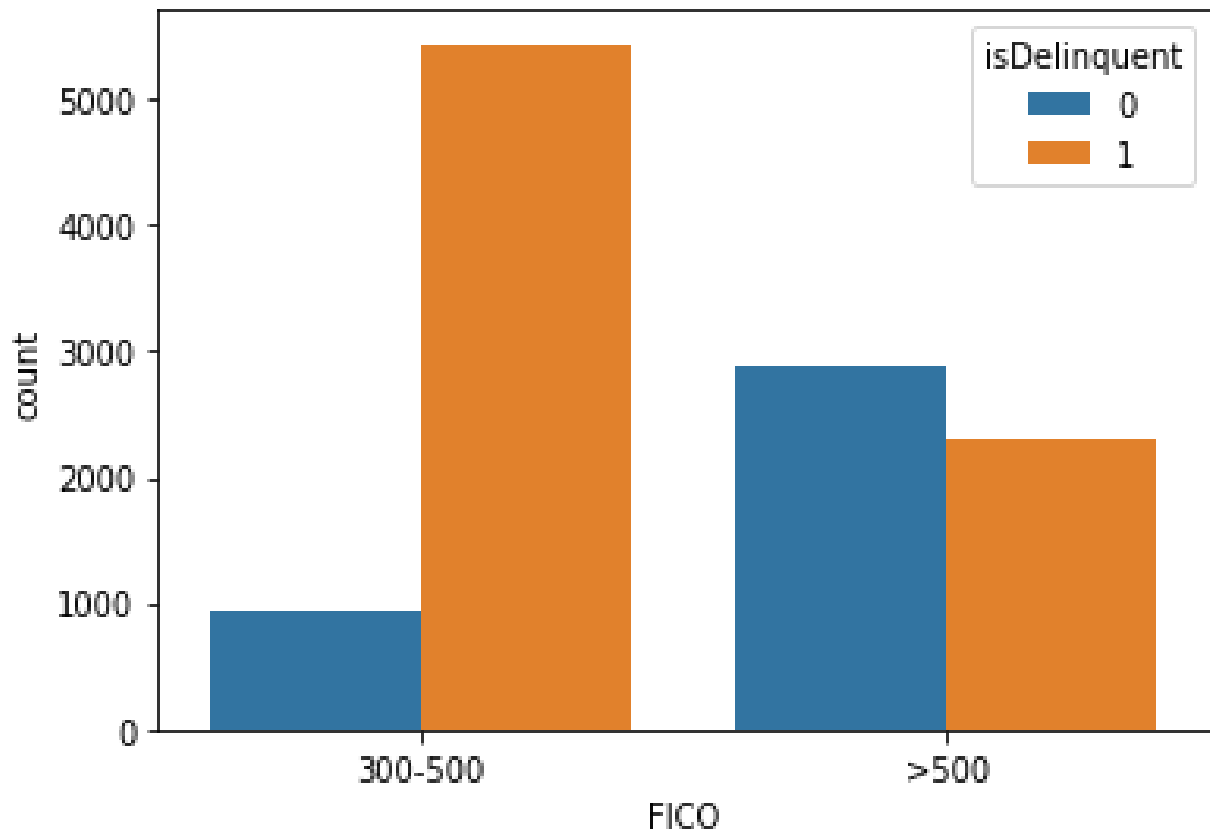


Fig.no: 18

Inference:

most customers have a FICO score between 300 and 500 (55.2%) causing delinquent followed by a score of greater than 500 (44.8%)

MULTIVARIATE ANALYSIS:

Heat map for checking correlation, pairplot

```
1. f = plt.figure()
   f.set_figwidth(25)
   )
   f.set_figheight(25)
   dataplot = sns.heatmap(loan.corr(), cmap='YlGnBu',
                           annot=True)plt.show()
```

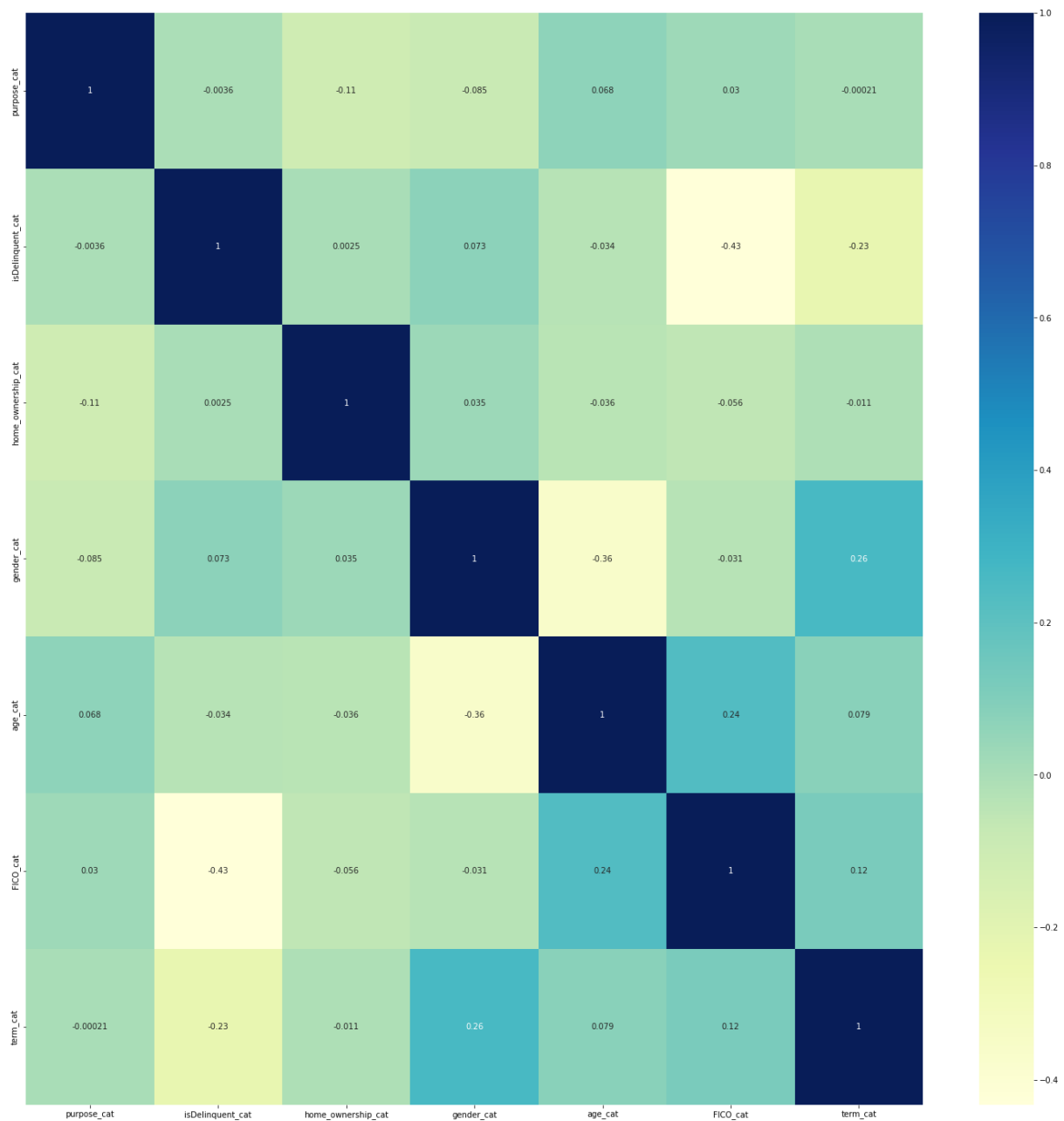


Fig.no: 19

Inference:

There is no strong correlation between variables

2. sns.pairplot(loan)

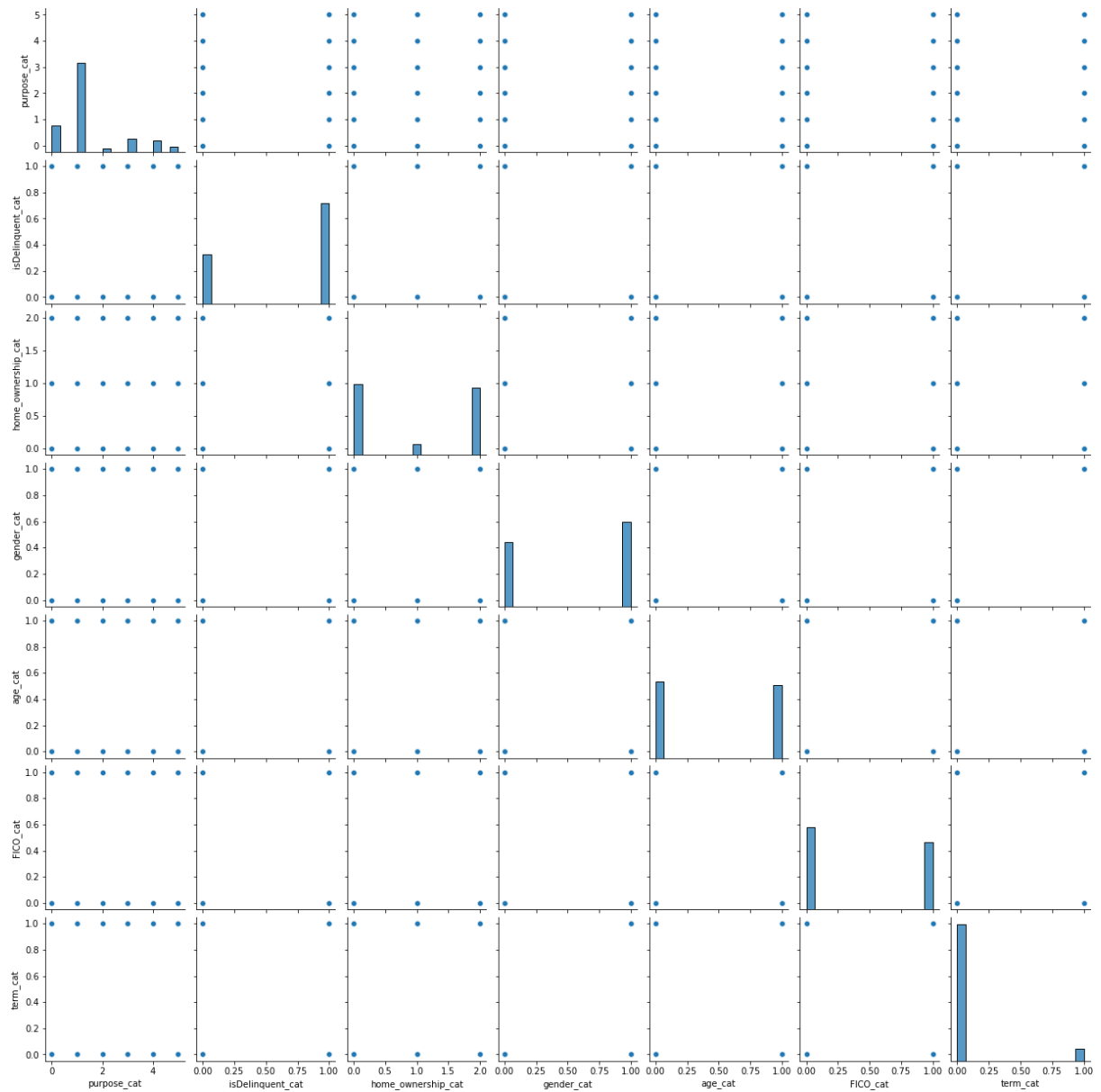


Fig.no: 20

Inference:

They are good in determining the values and are good indicators

3. sns.pairplot(loan, hue = 'isDelinquent')

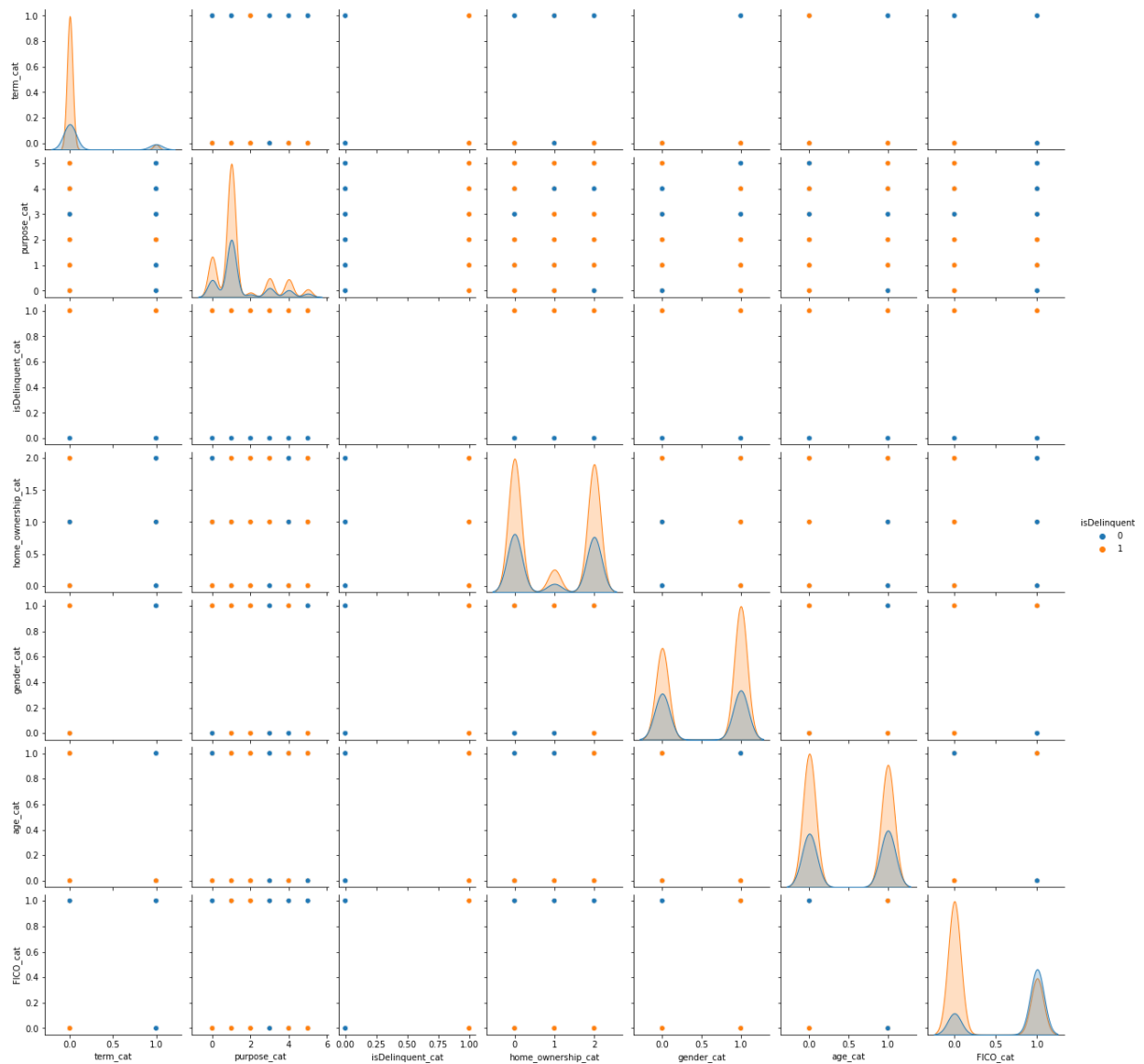


Fig.no: 21

Inference:

They are good in determining the values and are good indicators

2.6 MODEL BUILDING

For the loan delinquent dataset, the following three models can be built.

- Logistic Regression
- Decision Tree
- Random Forest

Train and Test Split:

The independent and dependent variables are names as x and y respectively. The data has been split into train and test data.

```
x = loan.drop(["isDelinquent"],  
axis=1)y = loan["isDelinquent"]
```

```
#train and split data  
from sklearn.model_selection import train_test_split  
xtrain,xtest,ytrain,ytest = train_test_split(x,y,train_size=0.7,random_state=2859)
```

Logistic Regression:

Logistic regression is a supervised learning method in machine learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring. It is a significant machine learning algorithm because it can provide probabilities and classify new data using continuous and discrete datasets. It can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification

```
[ ] from sklearn.linear_model import LogisticRegression  
log = LogisticRegression()  
log.fit(xtrain,ytrain)  
  
[ ] log  
LogisticRegression  
LogisticRegression()  
  
[ ] ypred = log.predict(xtrain)  
ypred  
array([1, 1, 0, ..., 0, 1, 1])  
  
[ ] ytrain.values  
[1, 1, 0, 0, 1, ..., 0, 1, 1, 1]  
Length: 8083  
Categories (2, Int64): [0, 1]  
  
[ ] ypred4 = log.predict(xtest)  
ypred4  
array([1, 1, 0, ..., 1, 1, 1])  
  
[ ] ytest.values  
[1, 1, 0, 0, 1, ..., 1, 1, 1, 1]  
Length: 1465  
Categories (2, Int64): [0, 1]
```

Fig.no: 22

Decision Tree:

A decision tree is a type of supervised machine learning used to categorize or make predictions based on how a previous set of questions were answered. The model is a form of supervised learning, meaning that the model is trained and tested on a set of data that contains the desired categorization. Decision trees are used as an approach in machine learning to structure the algorithm.

```
[ ] from sklearn.tree import DecisionTreeClassifier
[ ] Decision_Tree_Classifier=DecisionTreeClassifier(max_depth=6,random_state=59)
[ ] Decision_Tree_Classifier.fit(xtrain,ytrain)
[ ] Decision_Tree_Classifier
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=6, random_state=59)
[ ] ypred=Decision_Tree_Classifier.predict(xtrain)
ypred
array([1, 1, 0, ..., 0, 1, 1])
[ ] ypred1=Decision_Tree_Classifier.predict(xtest)
ypred1
array([1, 1, 0, ..., 1, 1, 1])
[ ] print(Decision_Tree_Classifier.score(xtrain,ytrain))
0.8491896573054559
[ ] print(Decision_Tree_Classifier.score(xtest,ytest))
0.8551226551226552
```

Fig.no: 23

Random Forest:

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm

```
[ ] from sklearn.ensemble import RandomForestClassifier
rand_reg=RandomForestClassifier(n_estimators=500,max_leaf_nodes=16,random_state=59)
[ ] rand_reg.fit(xtrain,ytrain)
[ ] rand_reg
RandomForestClassifier
RandomForestClassifier(max_leaf_nodes=16, n_estimators=500, random_state=59)
[ ] ypred2=rand_reg.predict(xtrain)
ypred2
array([1, 1, 0, ..., 0, 1, 1])
[ ] print(rand_reg.score(xtrain,ytrain))
0.8467153284671532
[ ] ypred3=rand_reg.predict(xtest)
ypred3
array([1, 1, 0, ..., 1, 1, 1])
[ ] print(rand_reg.score(xtest,ytest))
0.8528138528138528
```

Fig.no: 24

2.7 Model Evaluation:

Confusion Matrix:

A confusion matrix presents a table layout of the different outcomes of the prediction and results of a classification problem and helps visualize its outcomes. It plots a table of all the predicted and actual values of a classifier. It is used to determine the performance of the classification models for a given set of test data.

	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

True Negative: Model has given prediction No, and the real or actual value was also No. True Positive: The model has predicted yes, and the actual value was also true.

False Negative: The model has predicted no, but the actual value was Yes. It is also called as Type-II error.

False Positive: The model has predicted Yes, but the actual value was No. It is also called as Type-I error.

Classification Report:

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report. The classification report visualizer displays the precision, recall, F1, and support scores for the model.

Precision - Accuracy of positive predictions.

Recall - Fraction of positives that were correctly identified.

F1 score - Percentage of positive predictions were correctly identified.

Support score - Number of actual occurrences of the class in the specified dataset.

Confusion Matrices and Classification Reports for Test and Train Data

Logistic Regression:

Confusion Matrix of Logistic Regression Train Data

```
[[1700  957]
 [ 685 4741]]
```

```
True Negative 1700
False Positives: 957
False Negatives: 685
True Positives: 4741
```

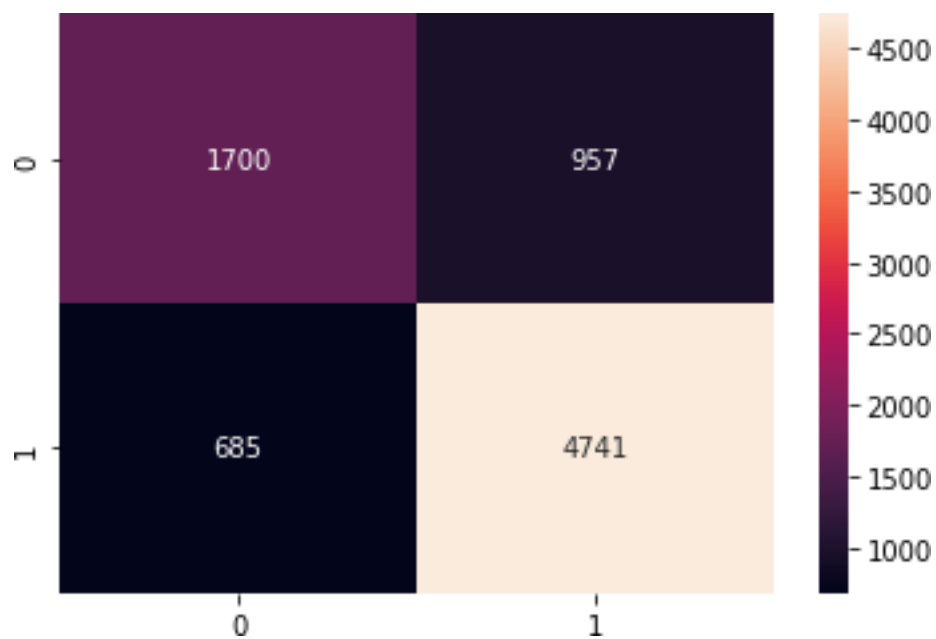


Fig.no: 25

Classification Report for Logistic Regression Train Data

	precision	recall	f1-score	support
0	0.71	0.64	0.67	2657
1	0.83	0.87	0.85	5426
accuracy			0.80	8083
macro avg	0.77	0.76	0.76	8083
weighted avg	0.79	0.80	0.79	8083

Confusion Matrix of Logistic Regression Test Data

```
[[ 742  428]
 [ 297 1998]]
```

True Negative 742
False Positives: 428
False Negatives: 297
True Positives: 1998

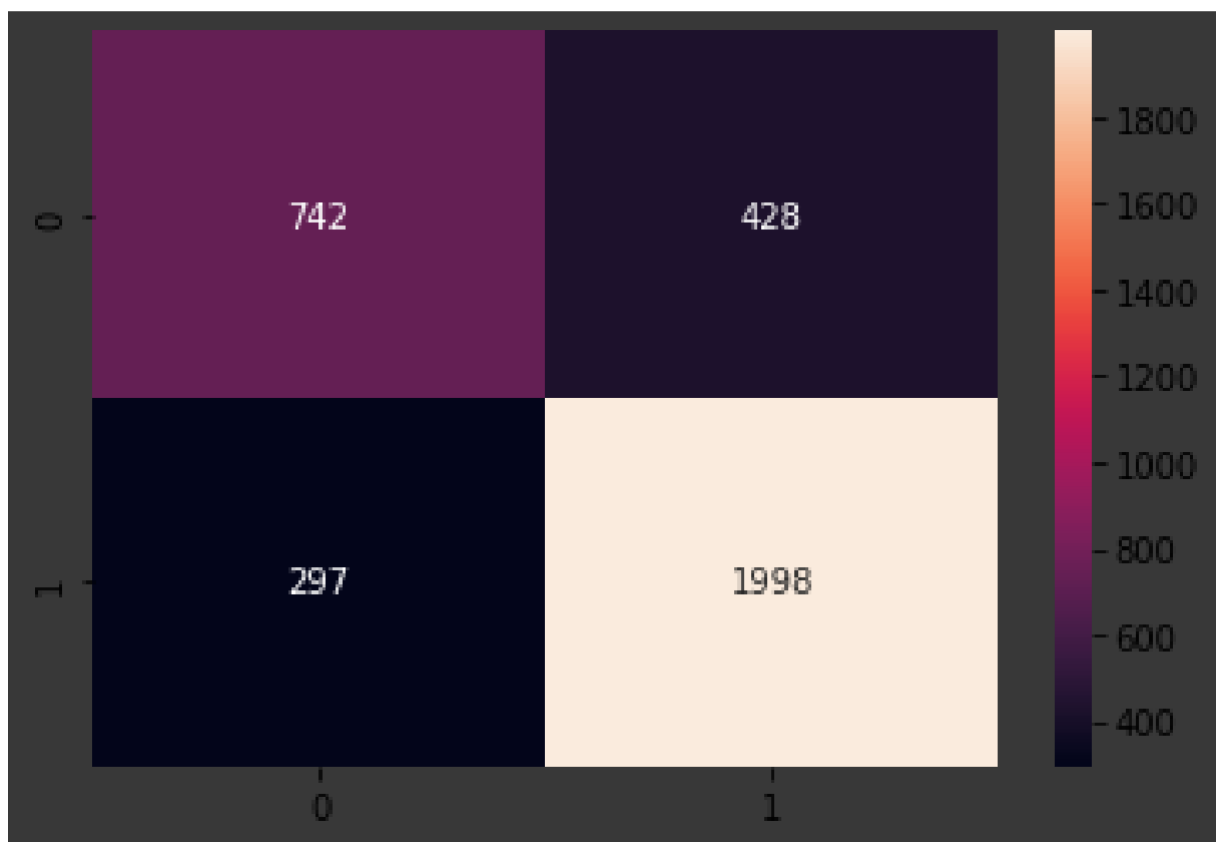


Fig.no: 26

Classification Report for Logistic Regression Test Data

	precision	recall	f1-score	support
0	0.71	0.63	0.67	1170
1	0.82	0.87	0.85	2295
accuracy			0.79	3465
macro avg	0.77	0.75	0.76	3465
weighted avg	0.79	0.79	0.79	3465

Decision Tree:

Confusion Matrix of Decision Tree Train Data

[[1953 704]
[515 4911]]

True Negative
1953
False
Positives: 704
False Negatives: 515
True Positives: 4911

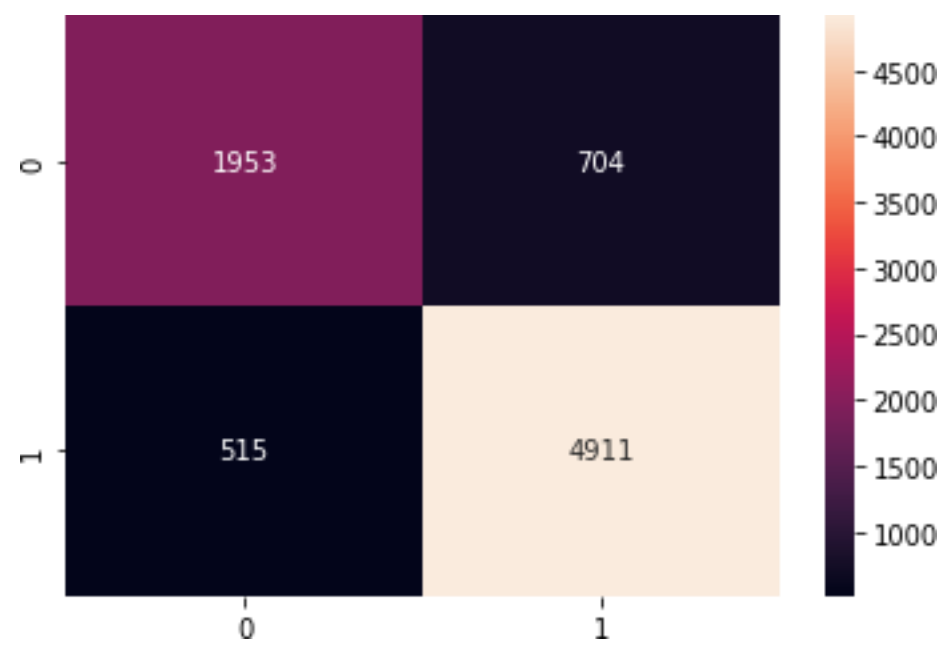


Fig.no: 27

Classification Report for Decision Tree Train Data

	precision	recall	f1-score	support
0	0.79	0.74	0.76	2657
1	0.87	0.91	0.89	5426
accuracy			0.85	8083
macro avg	0.83	0.82	0.83	8083
weighted avg	0.85	0.85	0.85	8083

Confusion Matrix of Decision Tree Test Data

```
[[ 891 279]
 [ 223 2072]]
```

True Negative 891

False Positives: 279

False Negatives: 223

True Positives: 2072

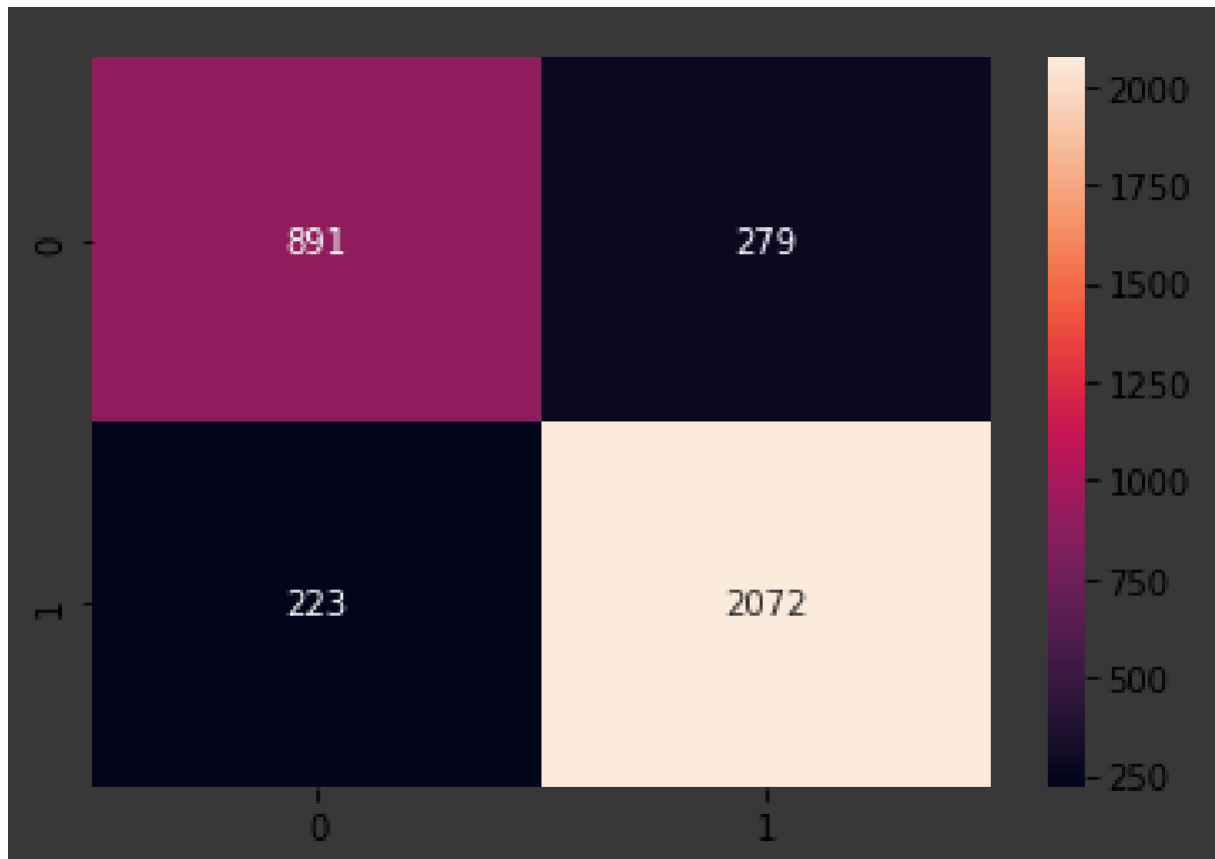


Fig.no: 28

Classification Report for Decision Tree Test Data

	precision	recall	f1-score	support
0	0.80	0.76	0.78	1170
1	0.88	0.90	0.89	2295
accuracy			0.86	3465
macro avg	0.84	0.83	0.84	3465
weighted avg	0.85	0.86	0.85	3465

Random Forest:

Confusion Matrix of Random Forest Train Data

[[1898 759]
[480 4946]]

True Negative
1898
False
Positives: 759
False Negatives: 480
True Positives: 4946

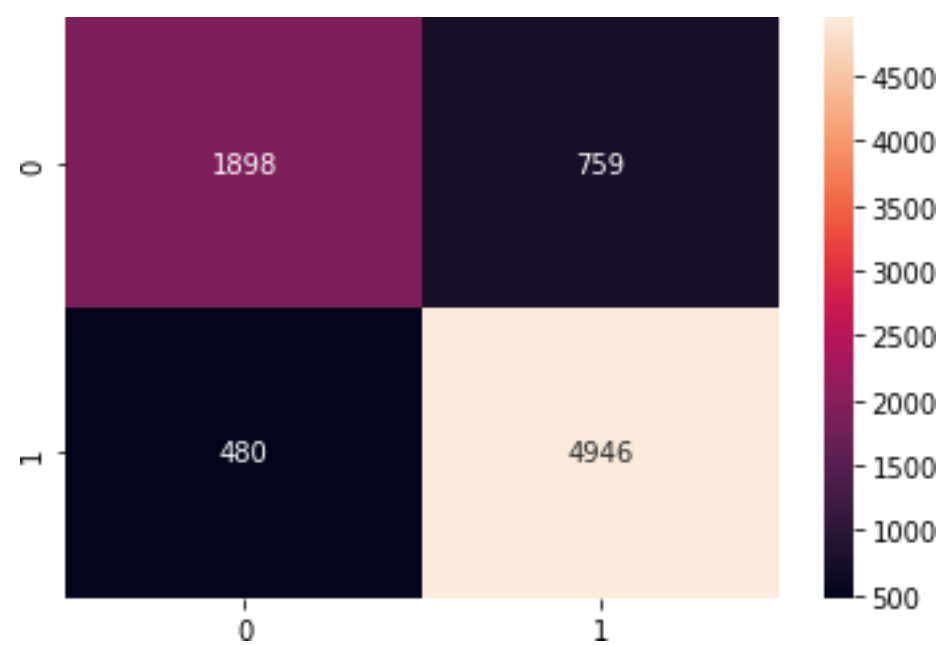


Fig.no: 29

Classification Report for Random Forest Train Data

	precision	recall	f1-score	support
0	0.80	0.71	0.75	2657
1	0.87	0.91	0.89	5426
accuracy			0.85	8083
macro avg	0.83	0.81	0.82	8083
weighted avg	0.84	0.85	0.84	8083

Confusion Matrix of Random Forest Test Data

```
[[ 873 297]
 [ 213 2082]]
```

True Negative 873

False Positives: 297

False Negatives: 213

True Positives: 2082

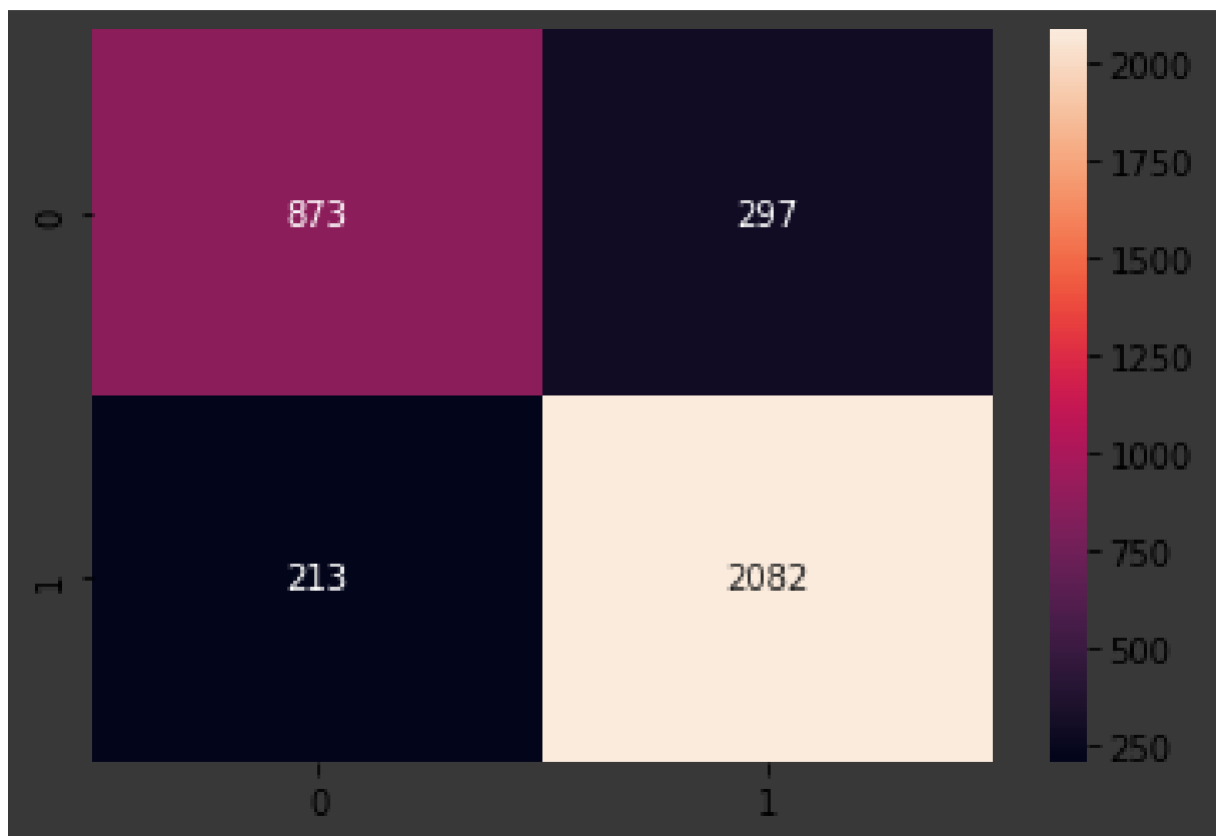


Fig.no: 30

Classification Report for Random Forest Test Data

	precision	recall	f1-score	support
0	0.80	0.75	0.77	1170
1	0.88	0.91	0.89	2295
accuracy			0.85	3465
macro avg	0.84	0.83	0.83	3465
weighted avg	0.85	0.85	0.85	3465

2.8 Model Inference:

Since the dataset has almost unbalanced target variable with 66.86% delinquent customers and 33.13% not delinquent customers. So, precision, re-call, f1-score are the Classification Report can be used to derive at the best model. The best performing model was a Decision Tree classifier, which achieved a precision, re-call and f1-score on macro avg is about 84%, 83% and 84% on the test set and 83%, 82% and 83% on the train data. It was followed by the Random Tree Classifier, which achieved an precision, re-call, f1-score on macro avg of 84%, 83%, and 83% on the test data and 83%, 81% and 82% on the train data. The Logistic Regression has shown precision, re-call and f1-score on macro avg of 77%, 75% and 76% on the test data and 77%, 76% and 76% on the train data.

Chapter 3: BUSINESS INSIGHTS AND RECOMMENDATIONS

3.1 Business Insights:

- FICO, term and gender are the important variables in determining if a borrower will get into a delinquent stage.
- No borrower shall be given a loan if they are applying for a 36-month term loan and have a FICO score in the range of 300-500.
- Female borrowers with a FICO score greater than 500 should be our target customers.
- Criteria to approve loan according to decision tree model should depend on three main factors - FICO score, duration of loan and gender that is - If the FICO score is less than 500 and the duration of the loan is less than 60 months then the customer will not be able to repay the loans. If the customer has a greater than 500 FICO score and is a female higher chance that they will repay the loans.

3.2 Recommendations:

- I would recommend analysing the loan delinquency data to identify patterns and trends that may be contributing to the delinquency.
- This analysis can help identify potential risk factors and inform strategies for reducing delinquency rates.
- Additionally, implementing automated payment reminders and offering flexible payment options may help reduce delinquency rates.
- It is also important to regularly review and update loan policies and procedures to ensure they are effective in preventing delinquency.

Chapter 4: References and Bibliography

1. "Loan Delinquency and Default: Implications for Risk Management" by John Krainer and Stephen F. LeRoy. This article discusses the causes and consequences of loan delinquency and default, and provides recommendations for risk management.
2. "Predicting Loan Delinquency: A Comparative Study of Machine Learning Algorithms" by S. S. S. Kumar and S. S. Kumar. This study compares the performance of different machine learning algorithms in predicting loan delinquency.
3. "The Impact of Loan Delinquency on Credit Scores" by Donghoon Lee and Wilbert van der Klaauw. This article examines the impact of loan delinquency on credit scores, and provides insights into the credit scoring process.
4. "Loan Delinquency and Default: A Macro Perspective" by Mark Doms and Frederic S. Mishkin. This paper provides a macroeconomic perspective on loan delinquency and default, and discusses the implications for monetary policy.
5. "Loan Delinquency and Default: A Review of the Literature" by Sumit Agarwal, Souphala Chomsisengphet, and Chunlin Liu. This article provides a comprehensive review of the literature on loan delinquency and default, and identifies gaps in the research.

5. Appendix

#Importing Libraries

```
import numpy as
npimport pandas
as pd
import matplotlib.pyplot as
pltimport seaborn as sns
from sklearn.model_selection import
GridSearchCVimport scipy.stats as stats
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import
LogisticRegressionfrom sklearn.model_selection
import train_test_split
from sklearn.metrics import
confusion_matrix,precision_score,recall_score,f1_score,accurac
y_score,classification_report
from sklearn.preprocessing import
StandardScalerimport warnings
warnings.filterwarnings("ignore")
```

#Loading Data

```
data = pd.read_csv("/content/Loan Delinquent Dataset.csv")
```

#Data Overview

```
data.shape

# let's create a copy of the
dataloan = data.copy()

# let's view the first 5 rows of the
dataloan.head()

# let's view the last 5 rows of the
dataloan.tail()

# let's check the data types of the columns in the
datasetloan.info()

# let's check for duplicate values in the
dataloan.duplicated().sum()

# let's check for missing values in the
```

```
dataloan.isnull().sum()
```

```
# let's view the statistical summary of the numerical columns in  
the dataloan.describe(include="all")
```

#Data Pre-processing

```
#checking for unique values in the ID
columnloan["ID"].nunique()

loan.drop(["ID"], axis=1, inplace=True)

# checking unique value in
purpose
loan["purpose"].unique()

# We can merge the purpose - 'other' and 'Other' together
loan["purpose"].replace("other", "Other", inplace=True)

loan["purpose"].unique()
```

#Encoding

One-hot encoding

```
x = loan.drop(["isDelinquent"],
axis=1)y = loan["isDelinquent"]

# encoding the categorical variables
x = pd.get_dummies(x,
drop_first=True)x.head()
```

#EDA

Univariate analysis

```
sns.distplot(x=loan['purpose_cat'])
```

most loan applicants are for house loans (59%) followed by car loans (18%)

```
bin_values = np.arange(start = 0 , stop = 3,
step = 1)loan['age'].hist(bins=bin_values)
```

almost an equal percentage of people aged 20 - 25 and >25 have applied for loan

```
x = plt.figure().subplots()
loan.home_ownership.value_counts().plot(x=x,
kind='pie', autopct='%1.1f%%')x.set_ylabel("")
x.set_title("home_ownership")plt.show()
```

very few applicants own their house

```
x = plt.figure().subplots()
loan.term.value_counts().plot(x=x,
kind='pie', autopct='%1.1f%%')x.set_ylabel("")
x.set_title("term")plt.show()
```

91.7% of the loans are for a 36-month term

```
x = plt.figure().subplots()
loan.purpose.value_counts().plot(x=x,
kind='pie', autopct='%1.1f%%')x.set_ylabel("")
x.set_title("purpose of loan")plt.show()
```

most loan applicants are for house loans (59%)

```
sns.countplot(x=loan['gender'])plt.show()
```

There are more male applicants (56.8%) than female applicants (43.2%)

```
sns.countplot(x=loan['FICO'])plt.show()
```

most customers have a FICO score between 300 and 500 (55.2%) followed by a score of greater than 500 (44.8%)

```
sns.countplot(x=loan['isDelinquent'])plt.show()
```

66% of the customers are delinquent

Bivariate analysis

```
sns.barplot(x='age_cat',y='isDelinquent',data=loan)
```

almost an equal percentage of people aged 20 - 25 and >25 have applied for loan

```
sns.jointplot(x='purpose_cat',y='isDelinquent',data=loan).plt.show()
```

most loan applicants are for house loans (59%) followed by car loans (18%) and 66% of the customers are delinquent

```
sns.kdeplot(x='home_ownership_cat',hue='isDelinquent',data=loan)
```

very few applicants <10% own their house, most customers have either mortgaged their houses or live on rent and mortgaged have delinquent

```
pd.crosstab(loan.term, loan.purpose, margins=True).style.background_gradient(cmap='copper')
```

91.7% of the loans are for a 36-month term and most loan applicants are for house loans (59%) followed by car loans (18%)

```
pd.crosstab(loan.isDelinquent, loan.age, margins=True).style.background_gradient(cmap='copper')
```

almost an equal percentage of people aged 20 - 25 and >25 have applied for loan and cause for delinquent

```
pd.crosstab(loan.gender, loan.home_ownership, margins=True).style.background_gradient(cmap='copper')
```

very few applicants own their house and there are more male applicants (56.8%) than female applicants (43.2%)

```
sns.countplot(x=loan[("purpose")],hue=loan[("isDelinquent")])
```

most loan applicants are for house loans (59%) and risk of delinquent

```
sns.countplot(x=loan[("home_ownership")],hue=loan[("isDelinquent")])
```

very few applicants own their house and likely less causing delinquent

```
sns.countplot(x=loan[("term")],hue=loan["isDelinquent"])
```

91.7% of the loans are for a 36-month term and causing

delinquent

```
sns.countplot(x=loan[("FICO")],hue=loan["isDelinquent"])
```

most customers have a FICO score between 300 and 500 (55.2%) causing delinquent followed by a score of greater than 500 (44.8%)

Multivariate analysis

```
f = plt.figure()
f.set_figwidth(25
)
f.set_figheight(2
5)
dataplot = sns.heatmap(loan.corr(), cmap='YlGnBu',
annot=True)plt.show()
```

There is no strong correlation between

variables

```
sns.pairplot(loan)
```

They are good in determining the values and are good indicators

```
sns.pairplot(loan, hue = 'isDelinquent')
```

They are good in determining the values and are good indicators

#Building the model

```
x = loan.drop(["isDelinquent"],
axis=1)y = loan["isDelinquent"]
```

```
#train and split data
```

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,train_size=0.7,random_state=2859)
```

#Logistic Regression

```
from sklearn.linear_model import
LogisticRegressionlog = LogisticRegression()
log.fit(xtrain,ytrain)

LogisticRegression
LogisticRegression
()

ypred =
log.predict(xtrain)ypred

ytrain.values

ypred4 =
log.predict(xtest)ypred4

ytest.values

score =
accuracy_score(ytrain,ypred)
score

score =
accuracy_score(ytest,ypred4)
score

sns.heatmap(confusion_matrix(ytrain,
ypred),annot=True,fmt='.4g');print('Confusion Matrix of
Logistic Regression Train Data \n')
print(metrics.confusion_matrix(ytrain, ypred),'\n')
tn, fp, fn, tp = metrics.confusion_matrix(ytrain, ypred).ravel()
print('True Negative',tn,'\n"False Positives:',fp,'\n"False Negatives:',fn,'\n"True
Positives:',tp,'\n')

sns.heatmap(confusion_matrix(ytest,
ypred4),annot=True,fmt='.4g');print('Confusion Matrix of
Logistic Regression Test Data \n')
print(metrics.confusion_matrix(ytest, ypred4),'\n')
tn, fp, fn, tp = metrics.confusion_matrix(ytest, ypred4).ravel()
print('True Negative',tn,'\n"False Positives:',fp,'\n"False Negatives:',fn,'\n"True
Positives:',tp,'\n')

print('Classification Report for Logistic Regression Train
```

```
Data \n')print(classification_report(ytrain,ypred))

print('Classification Report for Logistic Regression Test
Data \n')print(classification_report(ytest,ypred4))
```

#Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

Decision_Tree_Classifier=DecisionTreeClassifier(max_depth=6,random_state=59)Decision_Tree_Classifier.fit(xtrain,ytrain)

☑ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=6,
random_state=59)

ypred=Decision_Tree_Classifier.predict(xtrain)ypred

ypred1=Decision_Tree_Classifier.predict(xtest)ypred1

print(Decision_Tree_Classifier.score(xtrain,ytrain))

print(Decision_Tree_Classifier.score(xtest,ytest))

sns.heatmap(confusion_matrix(ytrain,
ypred),annot=True,fmt='.4g');print('Confusion Matrix of
Decision Tree Train Data \n')
print(metrics.confusion_matrix(ytrain, ypred),'\n')
tn, fp, fn, tp = metrics.confusion_matrix(ytrain, ypred).ravel()
print('True Negative',tn,'\n"False Positives:',fp,'\n"False Negatives:',fn,'\n"True
Positives:',tp,'\n')

from sklearn.metrics import
accuracy_score,classification_report,confusion_matrixfrom sklearn import
metrics
sns.heatmap(confusion_matrix(ytest,
ypred1),annot=True,fmt='.4g');print('Confusion Matrix of
Decision Tree Test Data \n')
print(metrics.confusion_matrix(ytest, ypred1),'\n')
tn, fp, fn, tp = metrics.confusion_matrix(ytest, ypred1).ravel()
print('True Negative',tn,'\n"False Positives:',fp,'\n"False Negatives:',fn,'\n"True
Positives:',tp,'\n')

print('Classification Report for Decision Tree Train
Data \n')print(classification_report(ytrain,ypred))

print('Classification Report for Decision Tree Test
Data \n')print(classification_report(ytest,ypred1))
```

#Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rand_reg=RandomForestClassifier(n_estimators=500,max_leaf_nodes=16,random_
state=59)

rand_reg.fit(xtrain,ytrain)

n)

RandomForestClassifier

er
RandomForestClassifier(max_leaf_nodes=16, n_estimators=500, random_state=59)

ypred2=rand_reg.predict(xtrain)
ypred2

print(rand_reg.score(xtrain,ytrain))

ypred3=rand_reg.predict(xtest)
ypred3

print(rand_reg.score(xtest,ytest))

from sklearn.metrics import
accuracy_score,classification_report,confusion_matrixfrom sklearn import
metrics
sns.heatmap(confusion_matrix(ytrain,
ypred2),annot=True,fmt='.4g');print('Confusion Matrix of
Random Forest Train Data \n')
print(metrics.confusion_matrix(ytrain, ypred2),'\n')
tn, fp, fn, tp = metrics.confusion_matrix(ytrain, ypred2).ravel()
print('True Negative',tn,'\n"False Positives:',fp,'\n"False Negatives:',fn,'\n"True
Positives:',tp,'\n')

sns.heatmap(confusion_matrix(ytest,
ypred3),annot=True,fmt='.4g');print('Confusion Matrix of
Random Forest Test Data \n')
print(metrics.confusion_matrix(ytest, ypred3),'\n')
tn, fp, fn, tp = metrics.confusion_matrix(ytest, ypred3).ravel()
print('True Negative',tn,'\n"False Positives:',fp,'\n"False Negatives:',fn,'\n"True
Positives:',tp,'\n')
print('Classification Report for Random Forest Train
Data \n')print(classification_report(ytrain,ypred2))
print('Classification Report for Random Forest Test
Data \n')print(classification_report(ytest,ypred3))
```