Loan Default Prediction: A Machine Learning Approach

Project Report

Group 5

Vaishnavee Manivannan

Naveen Dhanasekaran

979-837-9191

617-331-9411

manivannan.va@northeastern.edu

dhanasekaran.n@northeastern.edu

Percentage of Effort Contributed by Student 1: 50%

Percentage of Effort Contributed by Student 2: 50%

Signature of Student 1: Vaishnavee M

Signature of Student 2: Naveen D

Submission Date: 14th August 2023

Table of Contents

Problem Setting:

Loans are a lucrative product in the banking sector due to their potential for high revenue, but they also entail a proportionate level of risk. Despite banks' stringent assessments of an individual's loan repayment capacity, there are instances when they still experience failures. Hence, it becomes imperative to have a robust technique to select only low-risk applicants before lending loans. However, it should not be done at the cost of classifying many low-risk applicants as high-risk. Banks need to strike a balance between both. In the past, banks have hired highly trained credit analysts to manually calculate the credit score of customers. The credit score is a metric used to assess the creditworthiness of a customer. The credit score considered various factors such as payment history, credit utilization, length of credit history, types of credit accounts, and recent credit inquiries. However, with the advancement of technology, credit score calculations have transitioned to automated processes that utilize statistical models to filter eligible customers with low credit risks. These automated models also leverage historical data to provide more accurate predictions of creditworthiness. This project aims to assess various machine learning techniques to predict loan defaults. This will help approve the loans of low-risk customers only.

Problem Definition:

Predicting whether an applicant will default or not is a binary classification problem. In this project, we will build various supervised classification models using logistic regression, random forest, and boosting that classify each record in the dataset into ‘defaulter’ or ‘non-defaulter’. We then compare the classification models to select the best-performing one. The model primarily tries to answer the following questions.

(i) What is the level of risk associated with the borrower?

(ii) Considering the borrower's risk level, will they repay the loan or not, and what could be the best ML model to predict it? This is in fact the primary objective(prediction) of the model.

(iii) Along the way, we will try to answer questions or identify patterns related to the distribution of loan purpose, how loan purpose and the loan amount are related, the

distribution of interest rate, and the relationship between loan default and home ownership/employment through exploratory data analysis.

Data Sources:

We used the 'Lending Club Issued Loans' dataset from Kaggle. It contains the lending club's complete loan data from 2007-2015.

Dataset link: https://www.kaggle.com/datasets/husainsb/lendingclub-issued-loans (we have used the "lc_loan.csv" file)

LendingClub is one of the largest and most well-known online peer-to-peer lending platforms that virtually connects borrowers with investors in the US. Through LendingClub individuals or businesses in need of loans can borrow money directly from investors. The platform assesses the creditworthiness of loan applicants and helps investors in decision-making. Lending Club makes its loan performance data publicly available while protecting the identity and privacy of its users at the same time.

Data Description:

The dataset contains all accepted loan applications along with their current repayment status from 2007-2015. It has 74 features and 887379 data points. Such large datasets will be helpful in building different kinds of models, fine-tuning the parameters, and testing their efficiency.

The table below shows a sample of features from the dataset.

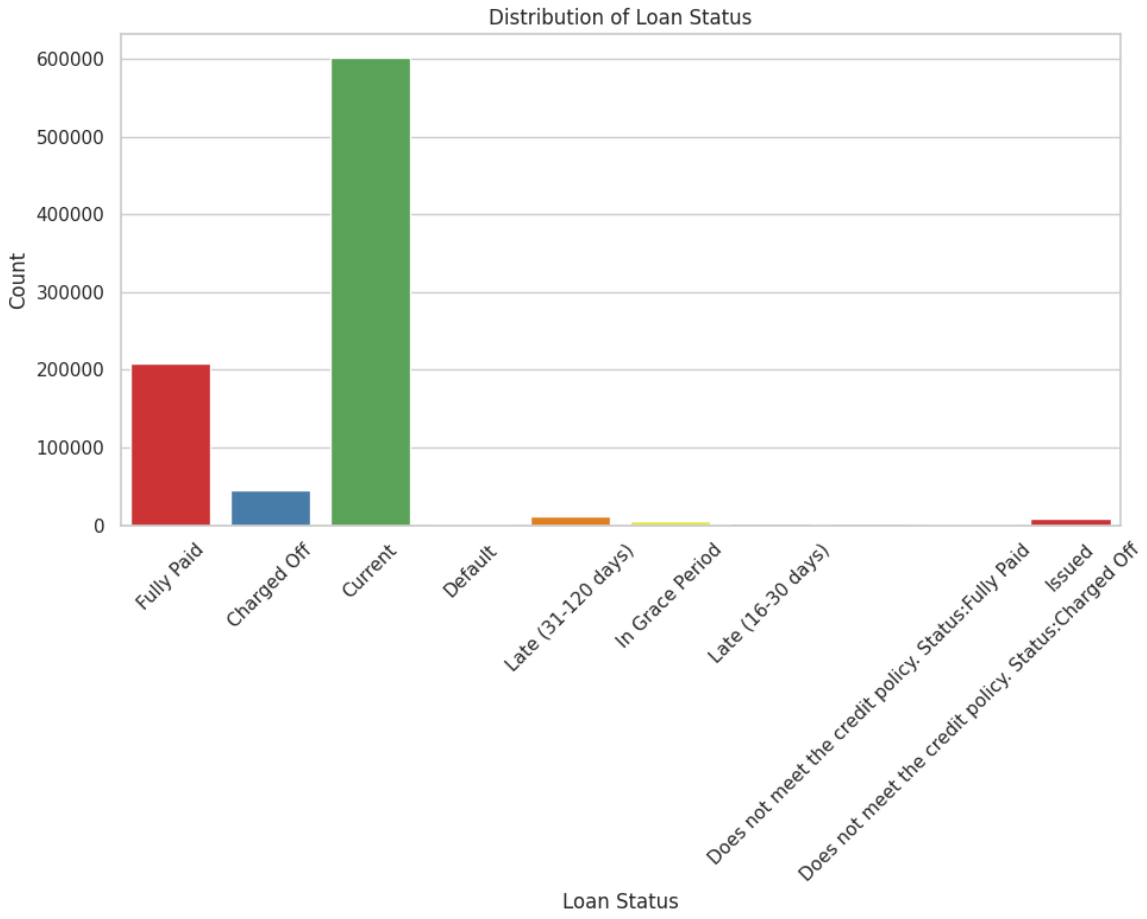
No.	Variable Name	Description
1	id	A unique LC assigned ID for the loan listing.
2	member_id	A unique LC assigned Id for the borrower member.
3	loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
4	funded_amnt	The total amount committed to that loan at that point in time.

5	funded_amnt_inv	The total amount committed by investors for that loan at that point in time.
6	term	The number of payments on the loan. Values are in months and can be either 36 or 60.
7	int_rate	Interest Rate on the loan
8	loan_status	Current status of the loan

In the dataset, the variable "**loan_status**" tells the current status of the loan, whether it is fully paid, defaulted, or in the grace period, etc. This column will be used as the target variable and all other remaining columns will be used as predictor variables.

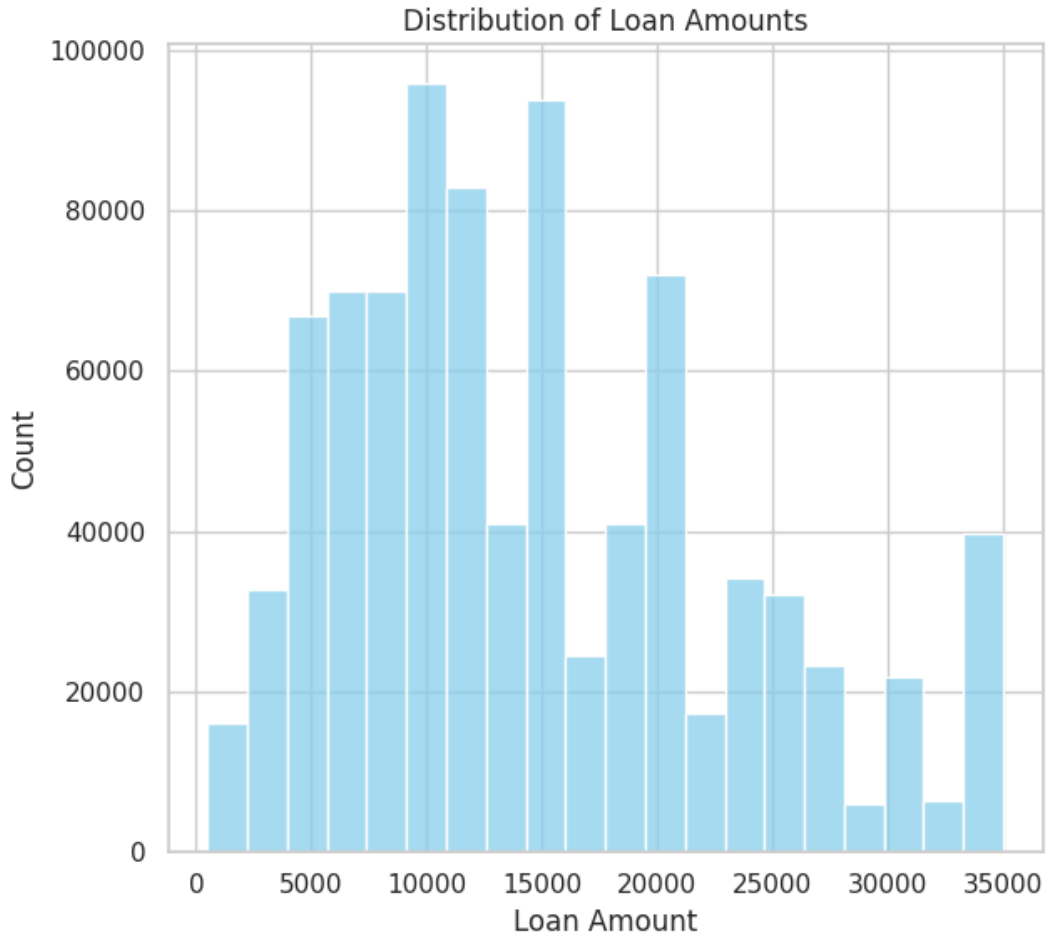
Data Exploration:

First, we looked at the target variable, and we plot a distribution of the "loan_status" column.

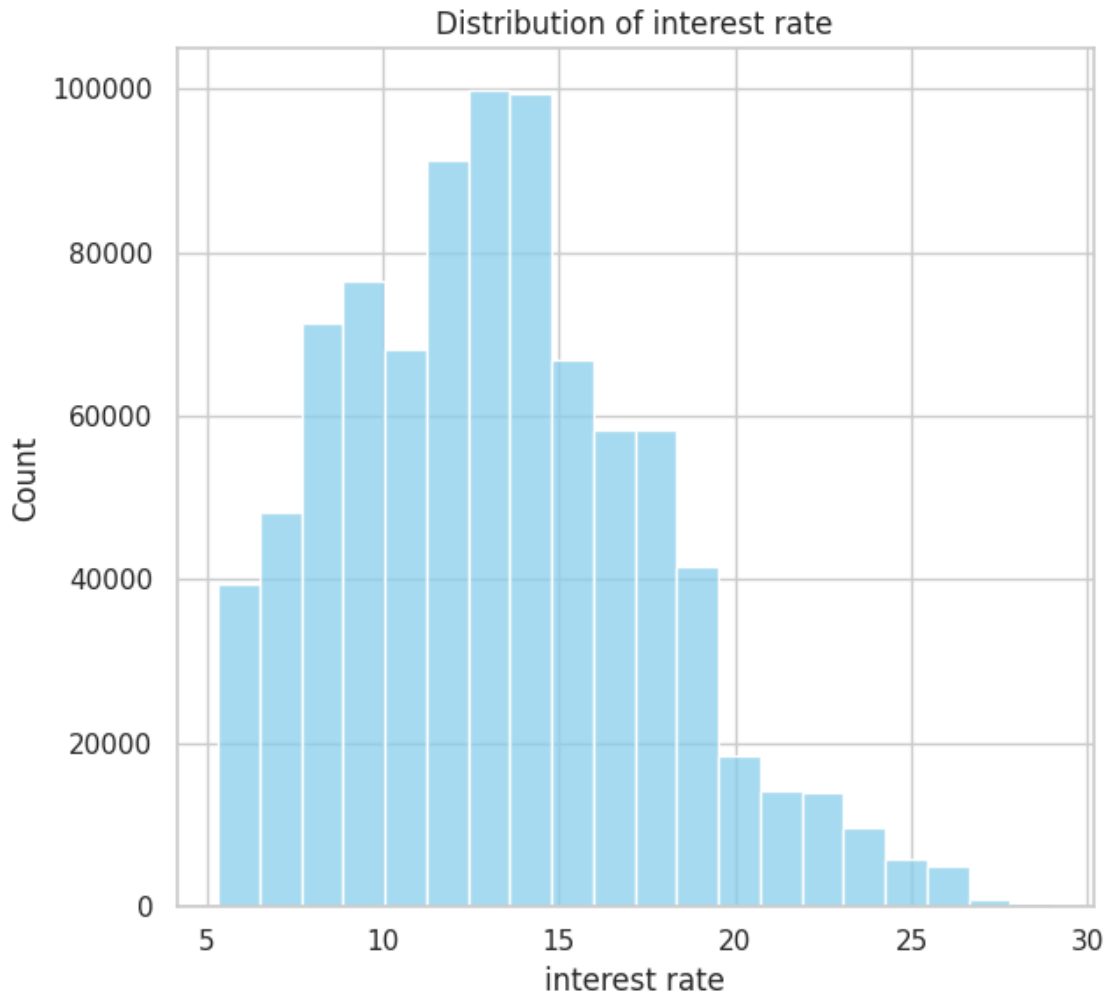


The majority of the loans are under the "current" category. But for our predictions, we will be mainly using the "Fully Paid" and "Charged Off" categories.

After that, we look at the distribution of other important columns like loan amount and interest rate.



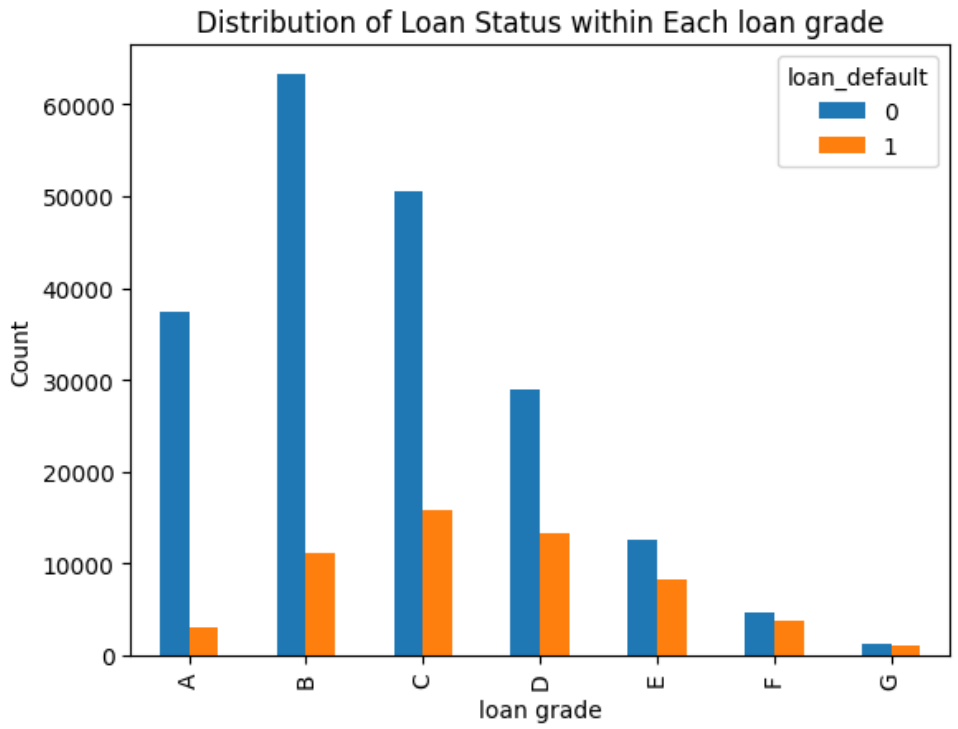
Distribution of loan amount doesn't follow any pattern. It's more randomly distributed.



Interest rate is more right-skewed indicating that the majority of loans might have relatively lower interest rates, with fewer loans having higher interest rates.

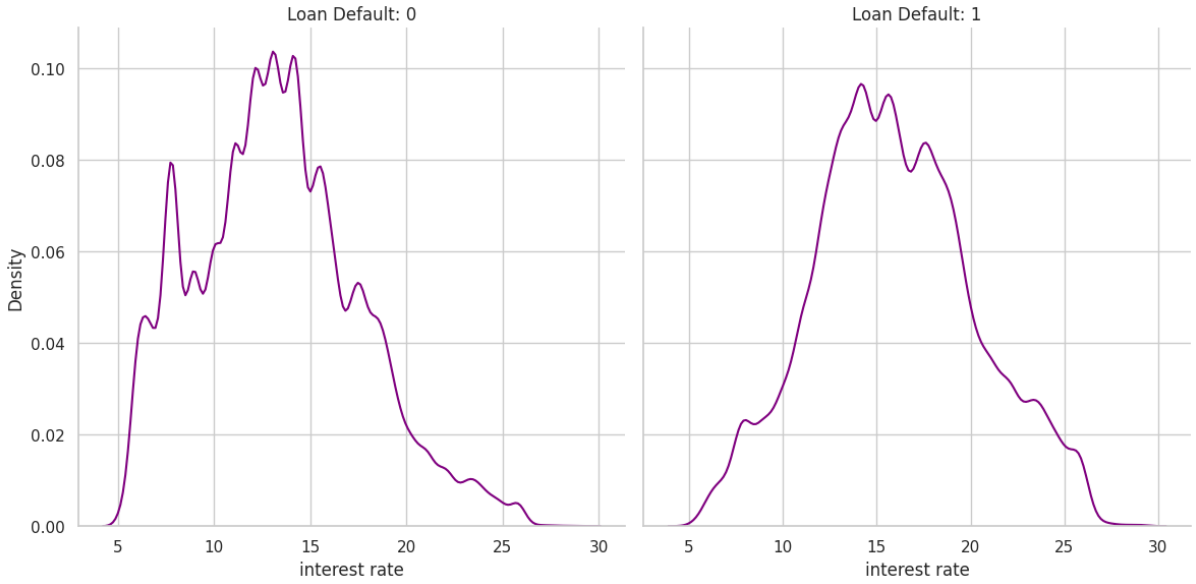
We use binary encoding to convert the target variable column (loan_status) with 0 as fully paid and 1 as charged off/default/late.

plot of distribution of loan status within each loan grade.



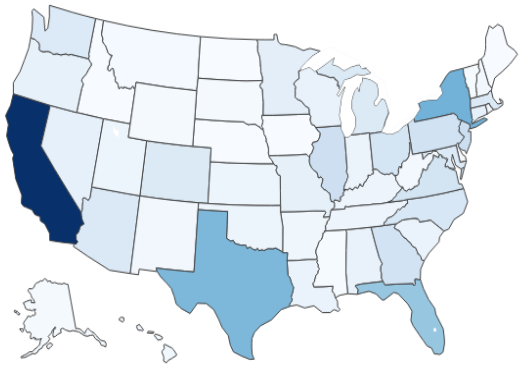
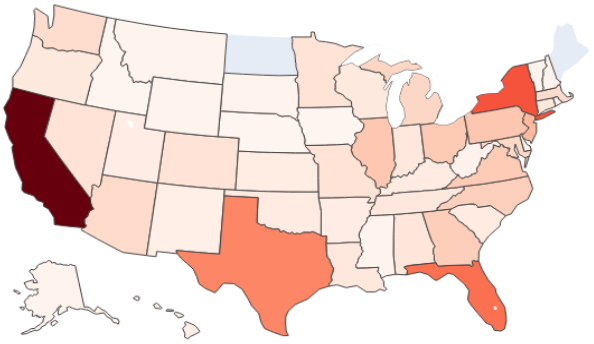
LendingClub's grading system typically ranges from A to G, with A being the least risky and G being the riskiest. Therefore, a Grade G borrower on LendingClub's platform would be considered to have the highest risk profile among the grades offered by the platform. This means that borrowers with a Grade G rating might have lower credit scores, less stable financial histories, or other factors that make them less likely to repay their loans compared to borrowers with higher grades.

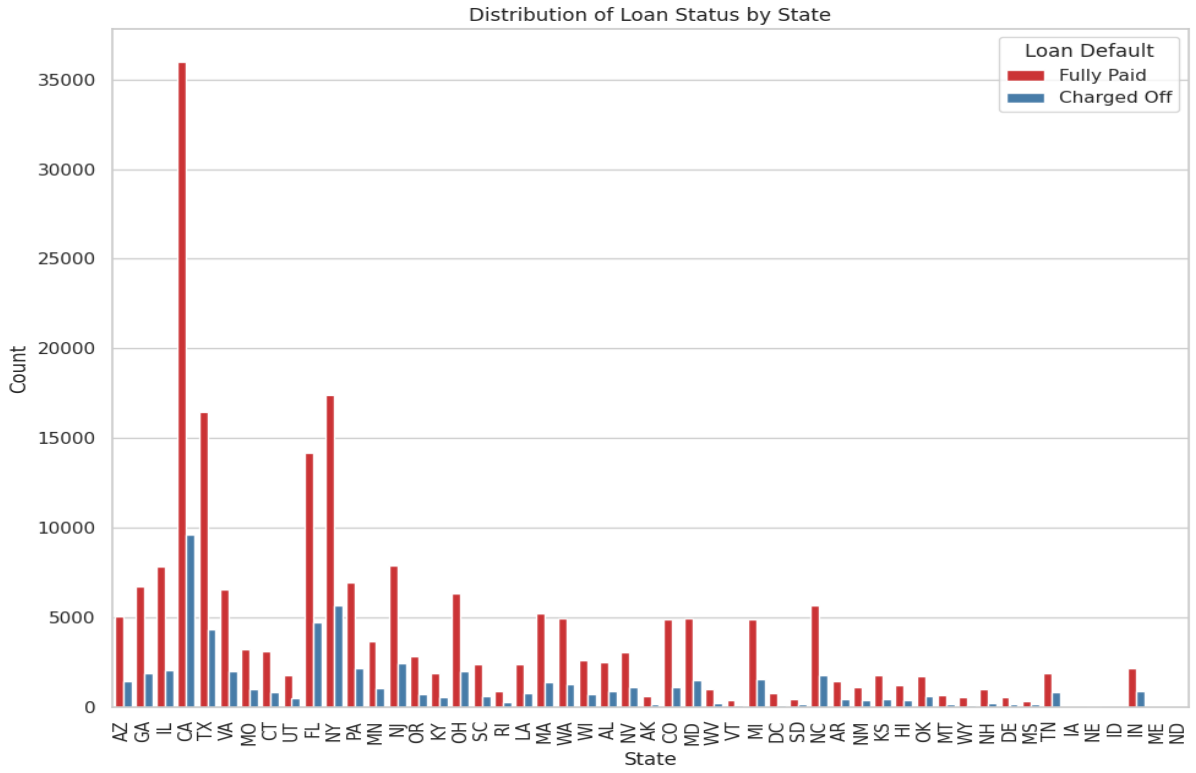
The distribution curve of interest rates for each loan status.



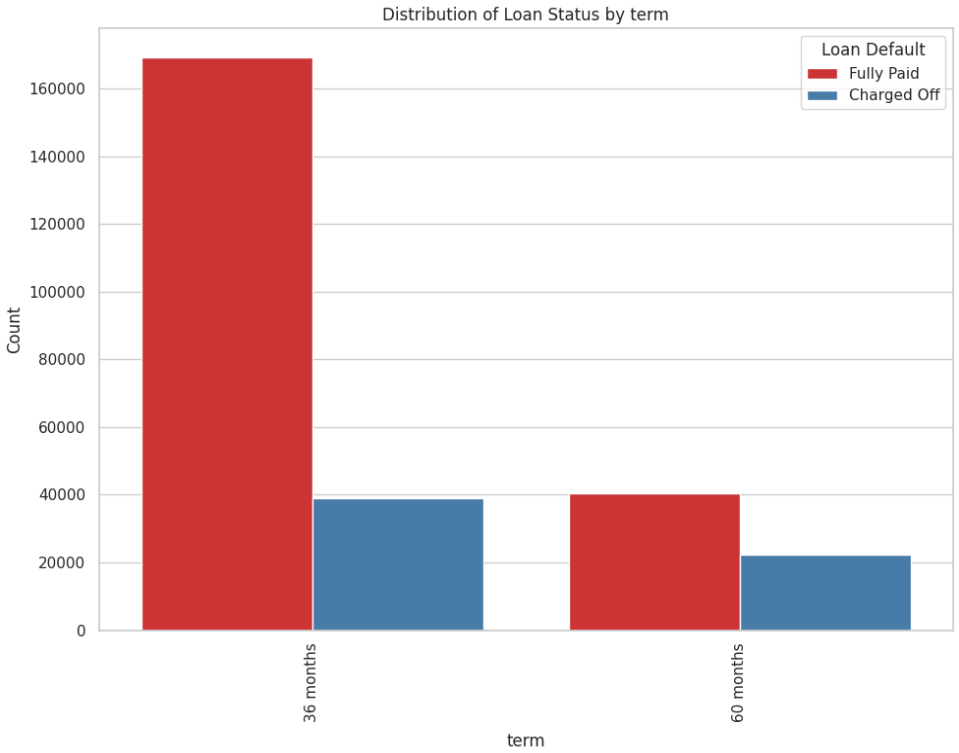
The density plot for non-default loans(left chart) is more skewed towards right as compared to the default loans indicating that non-default loans have lower interest rates as compared to the default loans.

distribution of the number of loans by location for both status types:

Fully Paid	Charged Off
	



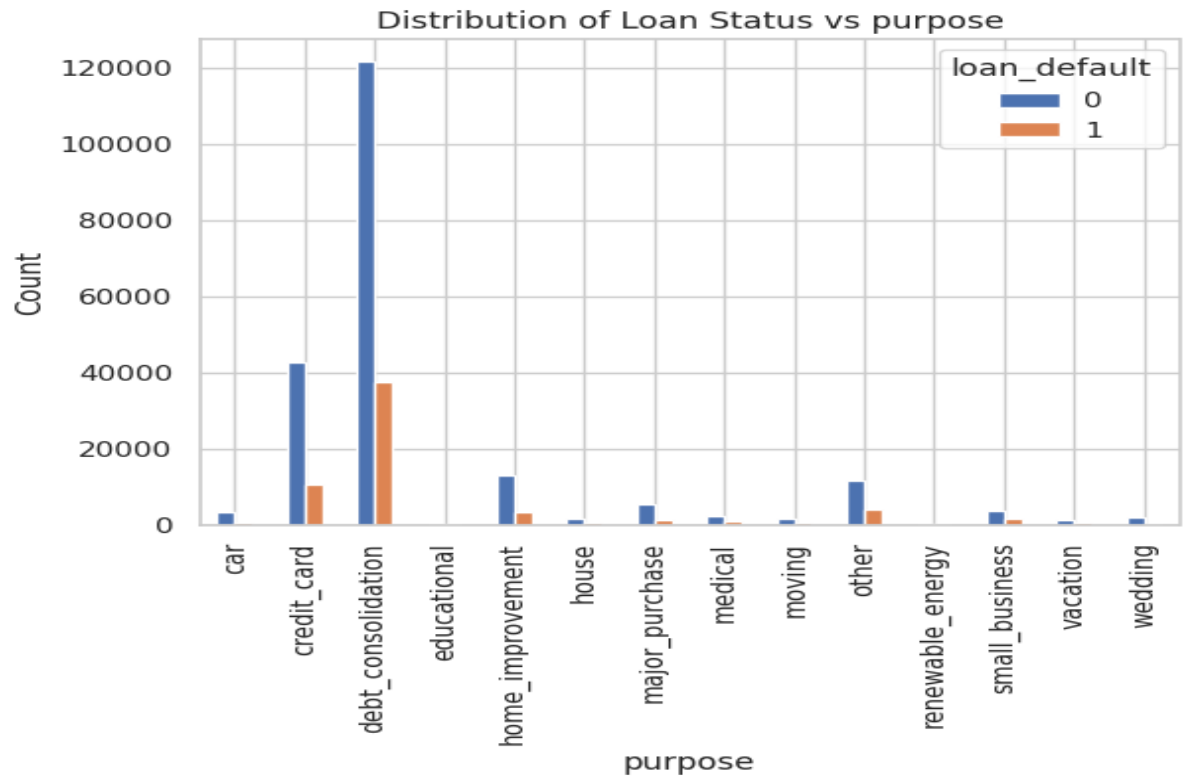
The heatmap and bar plot indicate that CA, NY, TX, FL have the highest count of both fully paid and charged off loans. It may be due to the fact that California (CA), New York (NY), Texas (TX), and Florida (FL) are among the most populous states in the United States. Because the number of borrowers are high, the number of defaulters are also high.



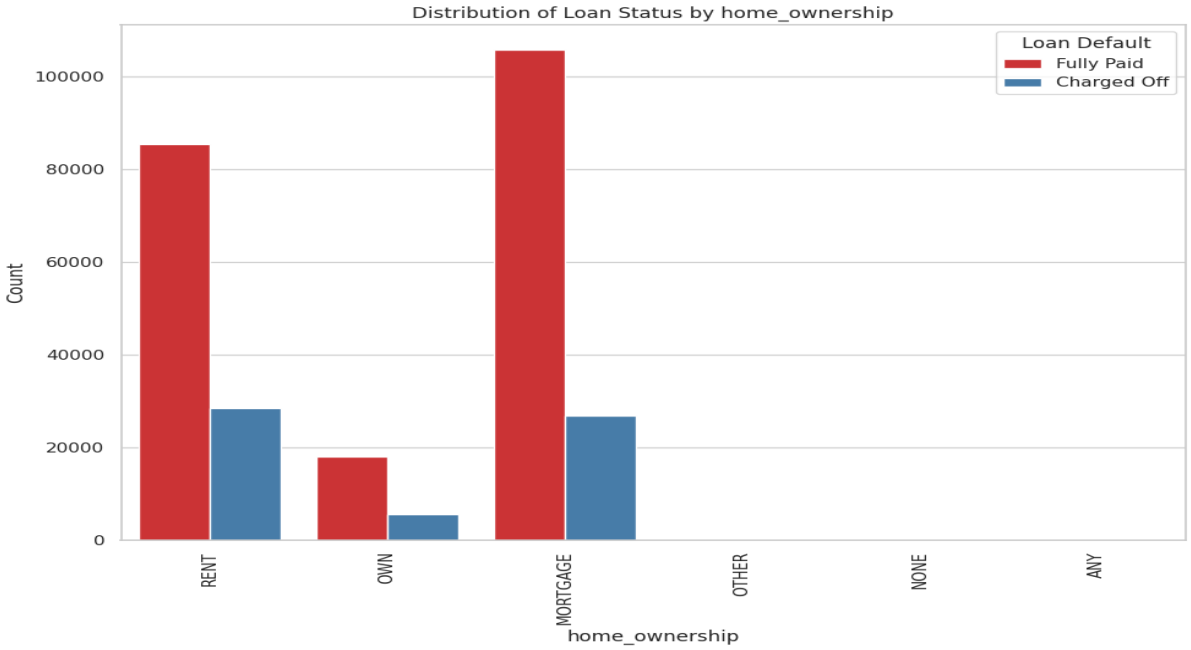
The plot illustrates the loan distribution based on the loan term. It is evident that the majority of the people prefer the shorter 36 month term loan. Additionally the default rate for the 36-month loan term is significantly lower compared to the 60- month term.

plot of the target variable “loan_status” distribution vs other categorical features

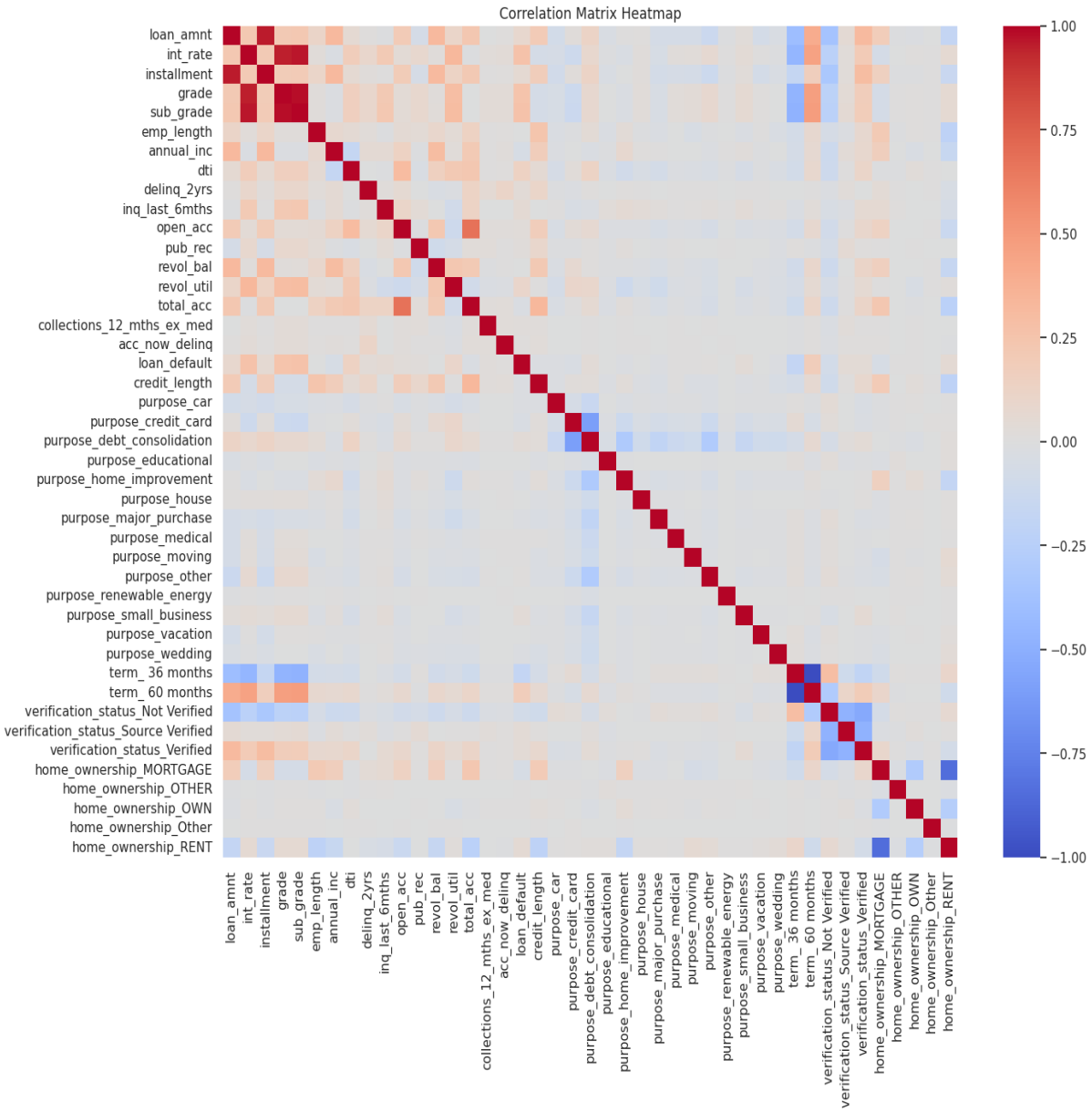
Majority of the borrowers belong to the employment length category of 10+ years and the majority of repaid loans also belong to the same category.



It is evident that the majority of the borrowers take out loans for the purpose of debt_consolidation or credit_card.



Borrowers with house mortgages and are in rental houses take a higher number of loans and thus higher number of fully paid and charged off loans.



Columns like grade, subgrade, interest rate were highly correlated. Dropping columns based on the correlation heatmap will be discussed in the next section.

Data Mining Tasks:

The entire data cleaning task can be broadly divided into three steps.

Step 1: In this step, we aimed to drop the columns that were very evidently not useful/suitable for the analysis. Then we also dropped the columns with more than 70% of null values.

The dataset has 887379 rows and 74 columns, representing different information provided by the lending club. The first step was to drop the features with a high number of null values as it will be difficult to do any kind of analysis with such columns. We fixed the threshold as 70%. About 20 columns were dropped. The list of columns with more than 70% missing data is as follows:

['desc', 'mths_since_last_record', 'mths_since_last_major_derog', 'annual_inc_joint', 'dti_joint', 'verification_status_joint', 'open_acc_6m', 'open_il_6m', 'open_il_12m', 'open_il_24m', 'mths_since_rcnt_il', 'total_bal_il', 'il_util', 'open_rv_12m', 'open_rv_24m', 'max_bal_bc', 'all_util', 'inq_fi', 'total_cu_tl', 'inq_last_12m']

In this project, the assumption is that our model will run before the loan is approved. Thus, there should be no information about the user's payment behavior for the loan. Also, columns like ‘zipcode’, ‘addr_state’, ‘Id’, ‘member_id’, and ‘URL’ are also dropped because they will not be useful to the analysis as both Id and member_id are just randomly generated unique numbers, whereas URL is a link to the user profile. "addr_state" and “zip_code” just talk about the location of the applicant. Hence, in this sub-step, we further dropped around 21 columns that are listed below.

- **funded_amnt:** The total amount funded by investors
- **funded_amnt_inv:** The total amount funded by investors (independent of LC)
- **pymnt_plan:** Indicates if the loan payment plan has been put on hold
- **out_prncp:** Remaining outstanding principal for total amount funded
- **out_prncp_inv:** Remaining outstanding principal for the portion of total amount funded by investors

- **total_pymnt:** Payments received to date for the total amount funded
- **total_pymnt_inv:** Payments received to date for the portion of total amount funded by investors
- **total_rec_prncp:** Principal received to date
- **policy_code:** Publicly available policy code for the product
- **total_rec_int:** Interest received to date
- **total_rec_late_fee:** Late fees received to date
- **recoveries:** Post charge-off gross recovery
- **collection_recovery_fee:** Post charge-off collection fee
- **last_pymnt_d:** The last month payment was received
- **next_pymnt_d:** The next scheduled payment date
- **last_pymnt_amnt:** The last total payment amount received
- **last_credit_pull_d:** The most recent month LC pulled credit for this loan
- **zip_code:** The first 3 numbers of the zip code provided by the borrower
- **member_id:** A unique LC assigned ID for the borrower member
- **id:** A unique LC assigned ID for the loan listing
- **url:** URL for the LC page with listing data

We will also drop the column “policy_code” as it only has only one 1 distinct value.

These steps have left us with 887379 rows and 33 columns.

Step 2: In this step, we did EDA and data cleaning in parallel. Through EDA we realized some of the features in the data set may be redundant. So we dropped those columns.

The target variable “loan_status” has the following values:

```
Current                                                601779
Fully Paid                                             207723
Charged Off                                             45248
Late (31-120 days)                                      11591
Issued                                                   8460
In Grace Period                                          6253
Late (16-30 days)                                        2357
Does not meet the credit policy. Status:Fully Paid       1988
Default                                                  1219
Does not meet the credit policy. Status:Charged Off       761
Name: loan_status, dtype: int64
```

For this project, we will exclude loans that are currently in the "Current" status as these loans are still being repaid by the borrowers. Our focus will be on loans that have either been fully repaid or have defaulted. Specifically, we will classify loans as belonging to the non-default category if they are marked as "Fully Paid," and as belonging to the default category if they are in the following statuses: "Charged Off," "Default," or "Late" (with a delay of 16-30 days or 31-120 days). Loans that are in the "Grace Period" or "Issued" status will be removed from our dataset due to the uncertainty associated with their repayment status.

After subsetting the data, we once again checked for missing values. It was observed that column 'mths_since_last_delinq' had 55% null values and the columns 'tot_coll_amt', 'tot_cur_bal', and 'total_rev_hi_lim' had about 25% null values. As the null value percentages are high, it is difficult to impute the missing values using mean or median. Hence, we decided to drop those four columns. Finally, we dropped rows with any null value using dropna(). We lost about 5% of the records after dropna().

We dropped the ‘title’ column which has too many categories. The column tells about the purpose of taking the loan. Encoding it will not be a good choice further down the pipeline. Alternatively, we used another column ‘purpose’ for our analysis. It has a well-defined set of categories and captures similar information as loan titles.

The column “emp_title” was dropped as it has around 139000 unique values. Encoding this will not be a good idea as it will result in high-dimensional sparse data.

We also dropped ‘application_type’ whose value_counts are as follows:

```
INDIVIDUAL 255297

JOINT 3
```

As most of its values correspond to individuals, this column is not of much use as a predictor, Hence, this column was dropped as well.

Step 3: Performed one-hot encoding/ordinal encoding for categorical features. The final features of the data were again checked for any multicollinearity and cleaned before proceeding for analysis.

The columns ‘grade’ and ‘sub_grade’ represent the overall creditworthiness of the borrower and risk level within any particular grade respectively. We did label encoding of these columns preserving the order of values in these columns.

The column ‘employee_title’ had too many categories. Encoding this will not be a good idea as it will result in high-dimensional sparse data. Hence, we dropped it.

We then mapped the employment length to corresponding numeric values as shown below.

```
emp_length_map = {'< 1 year': 0,
                  '1 year': 1,
                  '2 years': 2,
                  '3 years': 3,
                  '4 years': 4,
                  '5 years': 5,
                  '6 years': 6,
                  '7 years': 7,
                  '8 years': 8,
                  '9 years': 9,
                  '10+ years': 10}
```

‘home_ownership’ column tells whether the applicant is in a rental house, own house or the house is under mortgage and so on. We reduced the number of categories in this column by clubbing ‘ANY’ and ‘NONE’ categories and did one-hot encoding of the feature.

We also encoded the ‘verification_status’ column which tells whether the income information of the borrower has been verified.

Next, we derived a column called ‘credit_length’ from the columns ‘issue_d’ and ‘earliest_cr_line’. ‘issue_d’ is the date on which the loan was issued. This column in conjunction with the ‘earliest_cr_line’ column(date of earliest credit history available for the applicant) was used to calculate the approximate credit history length.

```
df3['credit_length'] =
df3['issue_d'].str.split('-').str[1].astype(int) -
df3['earliest_cr_line'].str.split('-').str[1].astype(int)
```

We used the one-hot encoding of categorical features “emp_length”, “purpose”, “term”, “verification_status”, and “home_ownership”. We used binary encoding to convert the target variable “loan_status” column where 0 represents ‘fully paid’ and 1 represents ‘charged off’, ‘default’, or ‘late’ categories. We standardized all the numeric columns in the dataset.

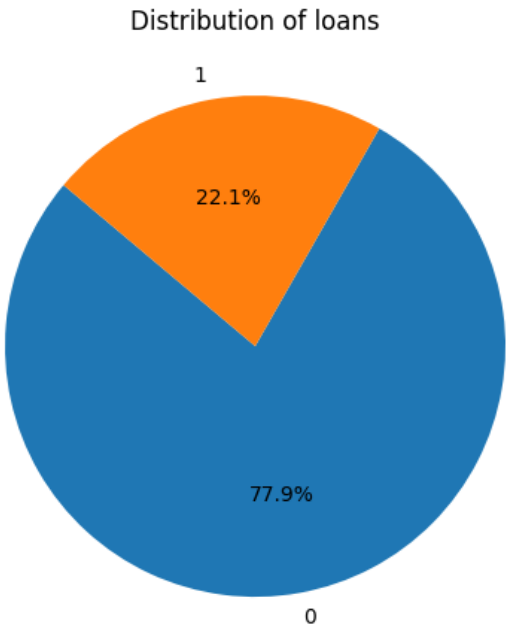
We then created a heat map for the correlation coefficient. It makes sense to check for correlation afteron-hot encoding inorder to find correlation among encoded categorical columns as well. It was observed that the following pairs of columns were highly correlated(>0.8).

Pairs of highly correlated columns:[('loan_amnt', 'installment'), ('int_rate', 'grade'), ('int_rate', 'sub_grade'), ('grade', 'sub_grade'), ('term_ 36 months', 'term_ 60 months'), ('home_ownership_MORTGAGE', 'home_ownership_RENT')]

We removed the columns “installments”, “grade”, “sub_grade”, 'term_ 36 months', and 'home_ownership_RENT'.

- Loan amount and installment are highly correlated. This is natural as a high loan amount will result in a high installment. Hence, the installment column was dropped.
- Interest rate, grade, and sub-grades seem to be all correlated with one another. This is also natural as grade and subgrade represent the risk of lending a loan to the applicant. High-risk loans will have high-interest rates. We will keep the interest rate and drop off all grade and subgrade columns as they were encoded and not originally present like the interest rate column.
- term_36 months and term_60 months are correlated because the column term has only two unique values. So, whenever one of the columns has a value of 0, the other one takes the value of 1. Let's drop the term_36 months from the data.
- Let's also drop home_ownership_RENT as m-1 columns are enough to represent m classes during one-hot encoding.

The final cleaned data had **255300 records with 38 features**. The chart below shows the final distribution of default(class 1) and non-default loans.



There is a class imbalance in the dataset for the obvious reason that most borrowers will pay back the loan successfully. We split the above data into 70% train and 30% test datasets for modeling and evaluation. Numeric features of train and test using StandardScaler() and p Initially, the sampling method was defined as 'stratify' in order to preserve the original data distribution in the test and train set. However, it was observed, even without stratified sampling, random sampling resulted in a distribution of train and valid data same as test data.

Data Mining Models and Performance Evaluation:

In this project, we used three different models for binary classification based on:

1) Logistic regression
2) Random forest
3) LightGBM Classifier

We also employed various resampling techniques along with these ML models to address the class imbalance. There are various techniques to address the class imbalance. Oversampling, undersampling, adjusting class weights, and bootstrapping are some of them. In this project, we explored class weights, bootstrapping, and SMOTE oversampling.

Evaluation metrics:

For evaluation we looked at recall and precision.

Recall (Sensitivity or True Positive Rate): TP / (TP + FN)

Recall measures the proportion of actual positive cases that were correctly predicted by the model. It is also known as sensitivity or the true positive rate.

Precision: TP / (TP + FP)

Precision measures the proportion of positive predictions that were actually correct. It is a measure of how well the model avoids making false positive predictions.

where,

TP: True Positives (correctly predicted positive cases)

FP: False Positives (negative cases that were incorrectly predicted as positive)

In this analysis, we will emphasize on recall value inorder to maximize the number of defaulters identified as the misclassification cost due to not identifying defaulter is higher than identifying a non-defaulter as defaulter.

We also looked at **ROC AUC score** which provides a single value that summarizes the ability of a binary classification model to discriminate between the positive and negative classes across different probability thresholds.

Accuracy may not be a good metric in this case due to class imbalance. High accuracy doesn't always mean positive classes are predicted well.

Logistic regression: Logistic regression is one of the most popular supervised classification algorithms used for binary classification problems. It predicts the class of an instance by calculating its probability of belonging to a particular class. In this case, it will allow us to predict whether the borrower will default on the loan or not. The algorithm is known for its mathematical simplicity, interpretability, and its computational efficiency. We built a simple logistic regression model with 'liblinear' solver (baseline) and added various customizations to it to address the class imbalance to improve model performance.

First, we introduced balanced weights to give equal importance to the majority and minority classes. Class weights are known to be useful in handling imbalanced data and avoids overfitting.It resulted in improved model performance. We then increased the weights to the ratio 1:5 (1 for majority class and 5 for minority class) which further improved the model performance. Different weights were assessed and 1:5 seemed to be the best in performance. The model's generalizability was also assessed using stratified five fold cross-validation.

Next, we used the SMOTE oversampling technique with logistic regression. SMOTE oversampling is a technique that creates synthetic examples of the minority class by interpolating between existing minority class samples. It selects a minority class instance at random and finds its k-nearest minority class neighbors. Then, it generates synthetic samples using the combination of that random minority class instance and one of its k-nearest neighbors. It performs better than random oversampling technique which randomly replicates the minority class samples. We found the logistic regression with SMOTE oversampling performed slightly worse than logistic regression with a balanced weight of 1:5.

The dataset had 38 features. We tried performing feature selection using K-Best and selected the best 15 or 20 features and build a logistic regression model. There was not much improvement in the performance. In Fact it had a lower recall value than the previous model with class weights.

The table shows the metrics for various logistic regression(LR) models created.

No	Model	Acc	Prec	Rec	F-1	AUC
1	Simple LR	0.78	0.54	0.10	0.17	0.71
2	LR +balanced weights	0.65	0.35	0.64	0.45	0.70
3	LR + weights (1:5)	0.55	0.30	0.80	0.44	0.70
4	LR + weights (1:5) + stratified CV	0.50	0.28	0.72	0.44	0.64
5	LR + SMOTE	0.65	0.35	0.64	0.45	0.65
6	LR + SMOTE + stratified CV	0.64	0.33	0.63	0.44	0.63
7	LR + feature selection with K-Best	0.66	0.35	0.63	0.45	0.71

Random Forest: Random Forest is a powerful machine learning algorithm used for both classification and regression tasks. It's an ensemble learning method that combines the predictions of multiple individual decision trees to produce a more accurate and robust overall prediction.

We first build a simple random forest with n_estimators as 100 and no other parameter. As expected it overfit the data due to both class imbalance and full-grown tree.

Next, to the plain random forest, we set bootstrap parameters to 'True', with tree depth as 10. It resulted in a very poor recall(0.06) and f-1 score(0.1). When the class_weight was set to 'balanced' along with bootstrapping, the model performance increased quiet well(recall 0.6)

Like logistic regression, different weight ratios were tested. Bootstrapping in conjunction with class weights beyond 'balanced' decreased the model performance. This could be due to overfitting as we are already using bootstrapping to reduce the effect of class imbalance on model performance.

All these optimizations would have been ideal to do using Hyperparameter tuning using GridSearchCV. But Hyperparamter tuning couldn't be done with the computation resources available with the free Colab account. Hence, we did a trial and error approach to get somewhat optimized parameters for the trees. Tree depth=10 and number of estimators=100 were used in some literature. Hence, we used those parameters. However, we have tried hyperparameter tuning with less computationally expensive LightGBM and it is discussed in the next section.

Now that random forest with bootstrapping and balanced weights is the best performing tree model so far, we added feature selection using K-best to it and evaluated the model performance. It resulted only in a slight increase (by a unit of ~0.03-0.05) in accuracy, precision, recall and F-1. This is because feature selection removes features that introduce noise into the ensemble preventing the trees from using them for decision splits.

Lastly, we tried adding SMOTE resampling to the previous random forest model. Model performance actually decreased due to overfitting.

No	Model	Acc	Prec	Rec	F-1	AUC
1	RF+Bootstrap+balanced weights	0.67	0.31	0.60	0.41	0.71
2	RF + Bootstrap + balanced weights + K-Best	0.66	0.35	0.63	0.45	0.70
3	RF + Bootstrap + balanced weights +SMOTE	0.79	0.45	0.15	0.23	0.70

LightGBM Classifier: LightGBM stands for "Light Gradient Boosting Machine". LightGBM is a powerful gradient-boosting framework to build high-performance machine learning models at scale. It supports parallel, distributed data processing and GPU learning. LightGBM performs leaf-wise split making the tree grow vertically as opposed to boosting methods that perform split level-wise. This results in high accuracy of LightGBM models. LightGBM also demonstrates remarkable efficiency in handling large datasets.

First, we build the LighGBM model with parameters optimized using GridSearchCV.

```
param_grid = {
    'learning_rate': [0.1, 0.05, 0.01],
    'max_depth': [3, 5, 7],
    'num_leaves': [15, 31, 63],
    'n_estimators': [50, 100, 200],
}
```

The number of folds was set to 5, with binary_logloss metric. The model had good precision but poor(0.1) recall and F-1 score(0.17).

Next, LightGBM with balanced class weight was modeled. It has improved the model recall and F-1 quite well.

Next, we use LightGBM with SMOTE oversampling. The model performance was decent, almost the same as the balanced weight GBM model and comparable with the logistic regression's best model but with a slightly lower recall value. However, LightGBM is more computationally intensive than logistic regression.

No	Model	Acc	Prec	Rec	F-1	AUC
1	LightGBM + GridSearchCV	0.78	0.54	0.10	0.17	0.71
2	LightGBM + GridSearchCV+balanced weight	0.66	0.36	0.64	0.46	0.71
2	LightGBM+SMOTE	0.65	0.35	0.64	0.45	0.70

Project Results and Future Scope:

- Overall, Logistic regression with weights ratio of 1:5 performed the best out of all the models with a recall of 0.8 and precision of 0.3. It also achieved an AUC score of 0.7.
- The model does very well on identifying default loans. Out of all loan defaults, the model was able to capture 80% of them but at the cost of misclassifying non-default loans as default. The chosen model is conservative as it has a low precision of 0.3. It means out of all the instances predicted as default, only 30% of them are actually default loans.
- Simple logistic regression has outperformed more complex random forest and LightGBM models.
- Tweaking class weights seem to be a better option than sophisticated oversampling techniques like SMOTE for this particular dataset. Models with assigned class weights have equal or better model performance than SMOTE resampled models.
- In this analysis, Recall was used as a primary metric with some importance to precision. From a business point of view, to maximize the revenue, it is important for the model to balance both the metrics. That is, to minimize the number of non-defaulter incorrectly classified as defaulters(low false positives or high precision) and find the maximum number of defaulters(high recall). This is because the lender wouldn't want to lose potential customers while trying to find potential defaulters. Hence, a custom cost function needs to be defined with misclassification cost for each class.
- In this study, Hyperparameter tuning was not done in an extensive manner due to computational complexity. With proper tuning, random forest and LightGBM may yield better model results.

Impacts of Project Outcomes:

Following are some of the impacts of predicting the loan default.

Lower Default Rates and improved risk assessments: By using predictive models to approve loans more selectively, lenders can experience lower default rates, resulting in better loan performance.

Enhanced Profitability: Fewer defaults and improved loan performance can lead to higher profitability for the lending institution and investors.

Better terms and interest: By assessing the risk of default, investors can offer more suitable terms and interest rates to low-risk borrowers, enhancing borrower satisfaction and retention.

Increased Investor Participation: Accurate default prediction models can attract more investors to invest through the lending club platform, as they gain confidence in the risk management practices.

Efficient Resource Allocation: Lenders can allocate resources more efficiently by focusing their attention on high-risk borrowers and optimizing collection efforts.

Appendix

Data Dictionary:

No.	Variable Name	Description
1	id	A unique LC assigned ID for the loan listing.
2	member_id	A unique LC assigned Id for the borrower member.
3	loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
4	funded_amnt	The total amount committed to that loan at that point in time.
5	funded_amnt_inv	The total amount committed by investors for that loan at that point in time.
6	term	The number of payments on the loan. Values are in months and can be either 36 or 60.
7	int_rate	Interest Rate on the loan

8	installment	The monthly payment owed by the borrower if the loan originates.
9	grade	LC assigned loan grade
10	sub_grade	LC assigned loan subgrade
11	emp_title	The job title supplied by the Borrower when applying for the loan.
12	emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
13	home_ownership	The home ownership status provided by the borrower during registration. Our values are: RENT, OWN, MORTGAGE, OTHER.
14	annual_inc	The self-reported annual income provided by the borrower during registration.
15	verification_status	Indicates if the borrowers income was verified by LC, not verified, or if the income source was verified
16	issue_d	The month which the loan was funded
17	loan_status	Current status of the loan
18	pymnt_plan	Indicates if a payment plan has been put in place for the loan
19	url	URL for the LC page with listing data.
20	desc	Loan description provided by the borrower
21	purpose	A category provided by the borrower for the loan request.
22	title	The loan title provided by the borrower
23	zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.
24	addr_state	The state provided by the borrower in the loan application
25	dti	A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower’s self-reported monthly income.
26	delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
27	earliest_cr_line	The month the borrower's earliest reported credit line was opened
28	inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)

29	mths_since_last_delinq	The number of months since the borrower's last delinquency.
30	mths_since_last_record	The number of months since the last public record.
31	open_acc	The number of open credit lines in the borrower's credit file.
32	pub_rec	Number of derogatory public records
33	revol_bal	Total credit revolving balance
34	revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
35	total_acc	The total number of credit lines currently in the borrower's credit file
36	initial_list_status	The initial listing status of the loan. Possible values are – W, F
37	out_prncp	Remaining outstanding principal for total amount funded
38	out_prncp_inv	Remaining outstanding principal for portion of the total amount funded by investors
39	total_pymnt	Payments received to date for the total amount funded
40	total_pymnt_inv	Payments received to date for a portion of the total amount funded by investors
41	total_rec_prncp	The principal received to date
42	total_rec_int	Interest received to date
43	total_rec_late_fee	Late fees received to date
44	recoveries	post charge off gross recovery
45	collection_recovery_fee	post charge off collection fee
46	last_pymnt_d	Last month payment was received
47	last_pymnt_amnt	Last total payment amount received
48	inq_last_12m	Number of credit inquiries in past 12 months
49	next_pymnt_d	Next scheduled payment date

50	last_credit_pull_d	The most recent month LC pulled credit for this loan
51	collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
52	mths_since_last_major_derog	Months since most recent 90-day or worse rating
53	policy_code	publicly available policy_code=1, new products not publicly available policy_code=2
54	application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
55	annual_inc_joint	The combined self-reported annual income provided by the co-borrowers during registration
56	dti_joint	A ratio calculated using the co-borrowers' total monthly payments on the total debt obligations, excluding mortgages and the requested LC loan, divided by the co-borrowers' combined self-reported monthly income
57	verification_status_joint	Indicates if the co-borrowers' joint income was verified by LC, not verified, or if the income source was verified
58	acc_now_delinq	The number of accounts on which the borrower is now delinquent.
59	tot_coll_amt	Total collection amounts ever owed
60	tot_cur_bal	Total current balance of all accounts
61	open_acc_6m	Number of open trades in last 6 months
62	open_il_6m	Number of currently active installment trades
63	pen_il_12m	Number of installment accounts opened in past 12 months
64	open_il_24m	Number of installment accounts opened in past 24 months
65	mths_since_rcnt_il	Months since most recent installment accounts opened
66	total_bal_il	Total current balance of all installment accounts
67	il_util	Ratio of total current balance to high credit/credit limit on all install acct

68	open_rv_12m	Number of revolving trades opened in past 12 months
69	open_rv_24m	Number of revolving trades opened in past 24 months
70	max_bal_bc	Maximum current balance owed on all revolving accounts
71	all_util	Balance to credit limit on all trades
72	total_rev_hi_lim	Total revolving high credit/credit limit
73	inq_fi	Number of personal finance inquiries
74	total_cu_tl	Number of finance trades