**CREDIT RISK SUMMARY REPORT**

**Kanini Kagendo (149810)**

**Allan Matano (094946)**

**Billy Otieno (078375)**

**Richard Waithera (150684)**

**Njeri Macharia (078624)**

# **INTRODUCTION**

This technical report focuses on a case study that provides insights into solving real-world business problems through exploratory data analysis (EDA) and machine learning techniques. The case study revolves around LendingClub, a prominent peer-to-peer lending company headquartered in San Francisco, California. Notably, LendingClub holds the distinction of being the first peer-to-peer lender to register its offerings as securities with the Securities and Exchange Commission (SEC) and facilitate loan trading on a secondary market. With its position as the world's largest peer-to-peer lending platform, LendingClub presents an ideal subject for our analysis.

By undertaking this case study, we aim to gain practical experience in addressing business challenges using EDA and machine learning approaches. Additionally, the study provides an opportunity to enhance our understanding of risk analytics within the banking and financial services domain. We will explore how data-driven techniques can be leveraged to mitigate the risk of financial losses while lending to customers. This analysis will equip us with valuable insights and strategies applicable to the broader field of risk management in the financial sector.

# **DATA UNDERSTANDING**

We began the data understanding process by examining the initial and final records from both the accepted and rejected loan applications datasets. This provided an overview of the data's general structure and content.

The rejected\_df dataset, containing data on rejected loan applications, comprised 27,648,741 records spread across 9 columns. These columns were 'Amount Requested', 'Application Date', 'Loan Title', 'Risk\_Score', 'Debt-To-Income Ratio', 'Zip Code', 'State', 'Employment Length', and 'Policy Code'. The dataset had a mix of numerical (float64) and categorical (object) datatypes. Upon inspection, we found that the 'Loan Title', 'Risk\_Score', 'Zip Code', 'Employment Length', and 'Policy Code' columns had missing values.

The accepted\_df dataset, which had data on accepted loan applications, was made up of 2,260,701 records across 151 columns. This dataset was significantly more intricate, with numerous numerical and categorical features. Just like the rejected applications dataset, several columns in this dataset also had missing data. Notably, the 'member\_id' column was entirely missing.

Additionally, we analyzed the uniqueness of our datasets. In the rejected applications dataset, for instance, the 'Risk\_Score' feature had 692 unique values, whereas the 'Debt-To-Income Ratio' feature had 126,145 unique values. In the accepted applications dataset, the 'loan\_amnt' (loan amount) feature exhibited 1,572 unique values, and 'funded\_amnt\_inv' (funded amount investor) feature presented 10,057 unique values.

This initial understanding of the shape, structure, and content of our data guided us through the data cleaning and feature selection stages. The primary focus was to handle missing data effectively and ensure that the chosen features were relevant and beneficial to our machine learning models. The uniqueness of certain features also indicated their potential as potent differentiators in our models.

# **DATA CLEANING**

The cleaning process for our data was comprehensive, starting with the removal of null or redundant data. From the `accepted\_df` dataframe, the 'member\_id' column was dropped because it contained only null values. In both the `accepted\_df` and `rejected\_df` dataframes, any columns that had more than 50% missing values were removed, helping to ensure the remaining data was more reliable and complete.

Next, we tackled the issue of data types. In the `rejected\_df` dataframe, the 'Application Date' was initially in the 'object' data type. It was converted to the 'datetime' data type to facilitate any subsequent time-series analysis or operations.

With our dataframes now reduced to more manageable sizes and structures, we trimmed down the `accepted\_df` dataframe even further to contain just 20 key features. These were chosen based on their perceived relevance to the loan acceptance process. This trimmed dataframe, `accepted\_trim`, contained information such as the borrower's state, annual income, debt-to-income ratio, employment length, FICO scores, loan amount, and status, amongst others.

Both `accepted\_trim` and `rejected\_df` still contained null values in their data. After determining which rows had null values, we dropped these rows from both dataframes. In the `accepted\_trim` dataframe, this resulted in a drop of 197,263 rows, while in `rejected\_df`, it led to a drop of 953,770 rows.

Next, we sought to identify and remove outliers from our numerical features. A function was created to calculate the Interquartile Range (IQR) for each numerical feature in the dataframe. Any values below Q1 - 1.5 \* IQR or above Q3 + 1.5 \* IQR were classified as outliers and replaced with NaN. This function was then applied to both `accepted\_trim` and `rejected\_df`, successfully removing outliers.

Given that some of our features were in a format that was not conducive to analysis, we created functions to extract useful data. For instance, in the 'term' column of the `accepted\_trim` dataframe, a function was created to split the term into two components - the term number and the term unit. These two components were then stored as new columns in the dataframe.

Similarly, for the 'Employment Length' column in the `rejected\_df` dataframe, a function was created to extract both the number of years of employment and the context ('less than a year', 'ten or more years', or 'exact years').

Additionally, the 'Debt-To-Income Ratio' column in `rejected\_df` was cleaned to remove the '%' sign and convert the remaining value into a float data type.

Finally, any remaining rows with null values after these transformations were dropped from both `accepted\_trim` and `rejected\_df`. This left us with clean, transformed, and more useful dataframes for our subsequent data analysis and modeling.

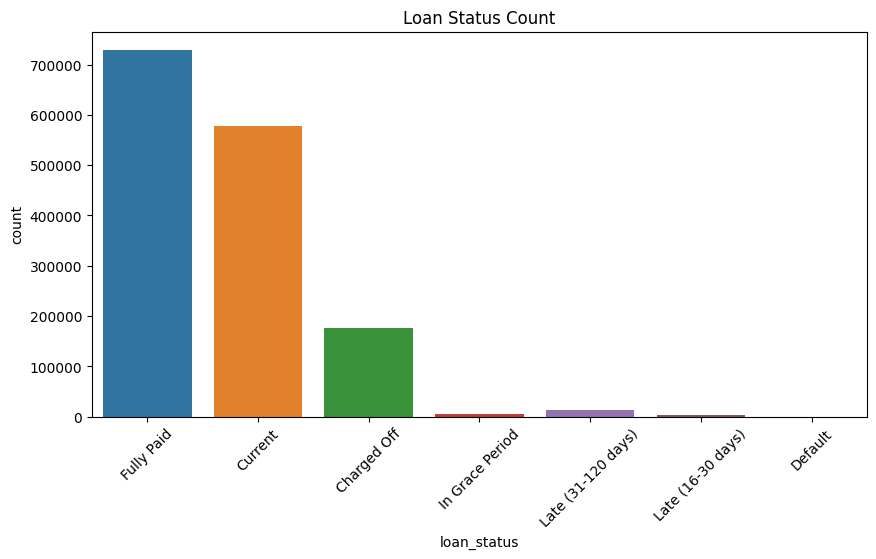
# **EXPLORATORY DATA ANALYSIS**

The Exploratory Data Analysis (EDA) played a crucial role in our analysis by providing valuable insights into the dataset. Through EDA, we identified the variables that hold significance for our analysis, allowing us to focus our attention on the most relevant factors. Additionally, we gained a comprehensive understanding of the data by examining summary statistics, such as measures of central tendency and variability. The EDA also allowed us to visualize the data through various graphical representations, facilitating the identification of patterns, trends, and potential outliers.

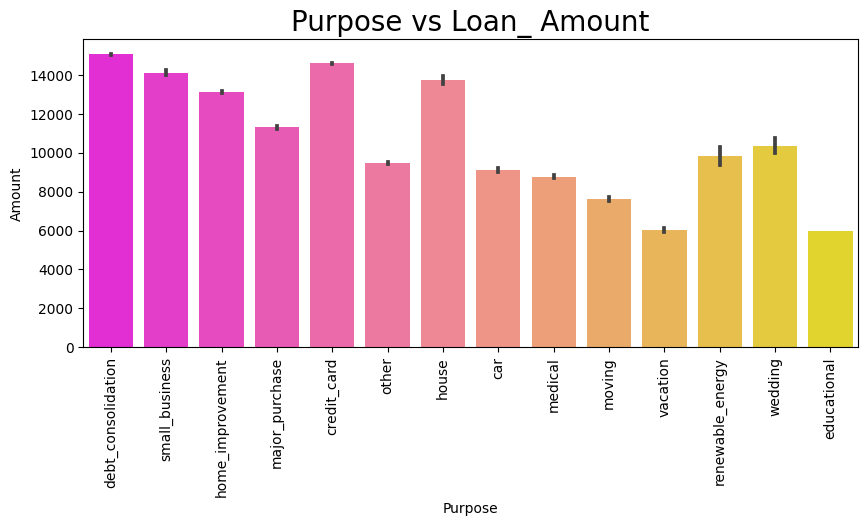
We notice that, there are broadly two types of features:

* Features related to the applicant (demographic variables such as occupation, employment details etc.),
* Features related to loan characteristics (amount of loan, interest rate, purpose of loan etc.)

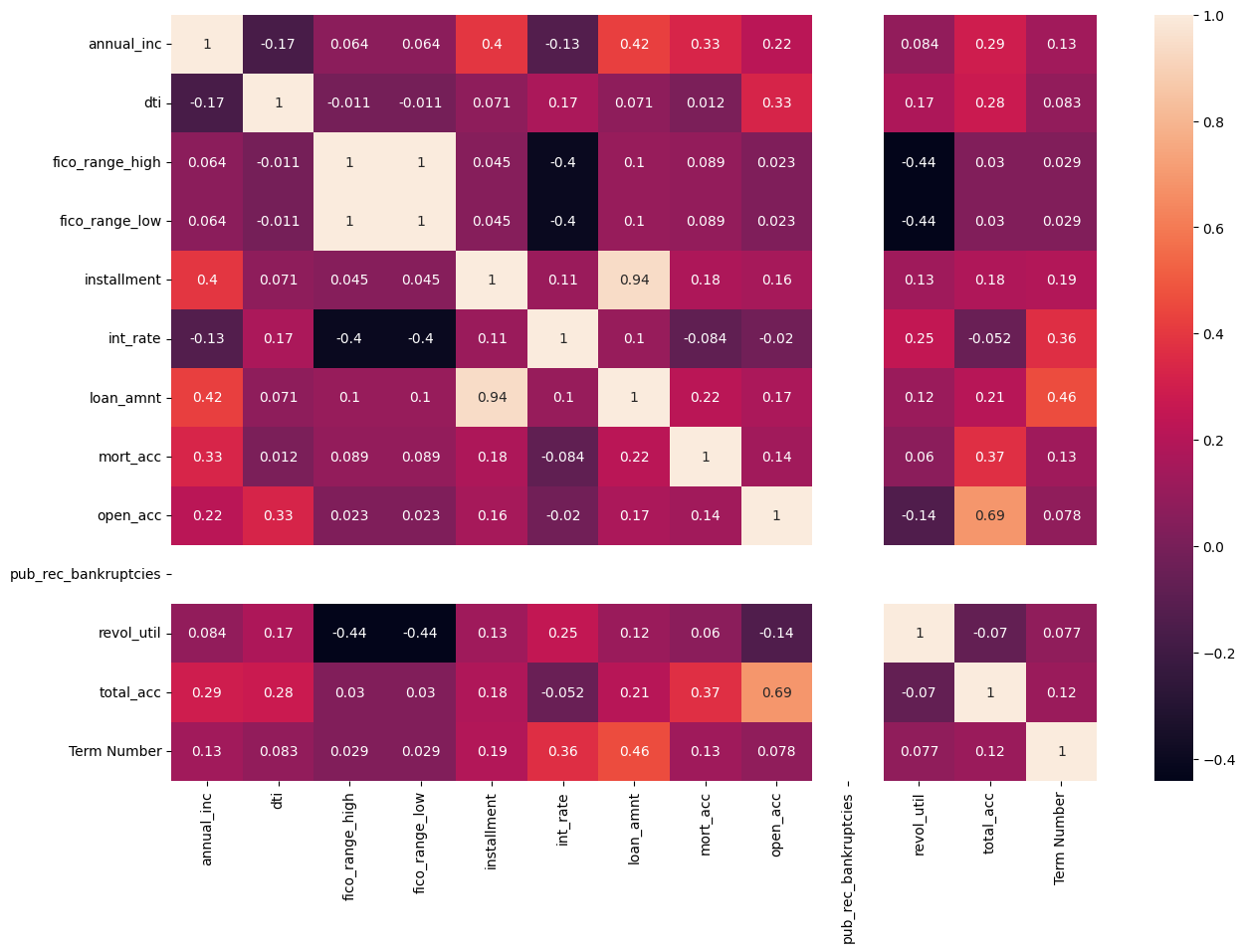
Below is a visual representation of the Loan status count, which shows that the majority of the loans were actually fully paid and very few defaults. This also informed the scaling of our data for further analysis.



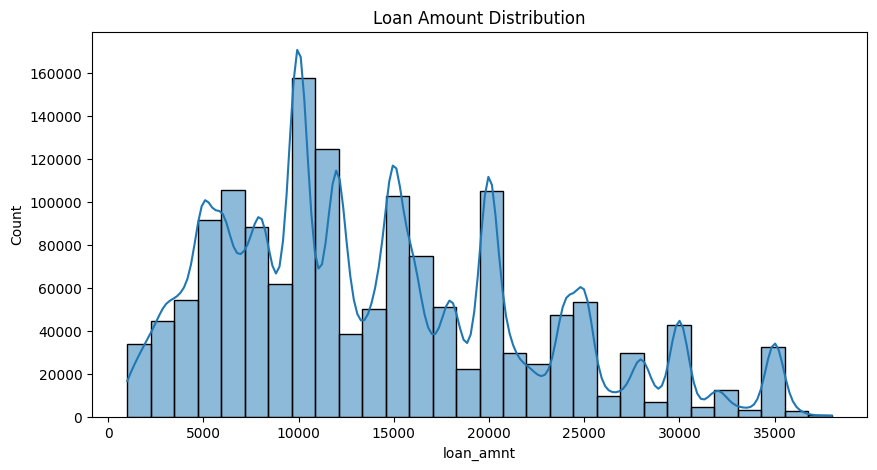
We were able to deduce the various reasons why people take loans and the respective amounts.



We also noticed an almost perfect correlation between "loan\_amnt" and the "installment" feature.

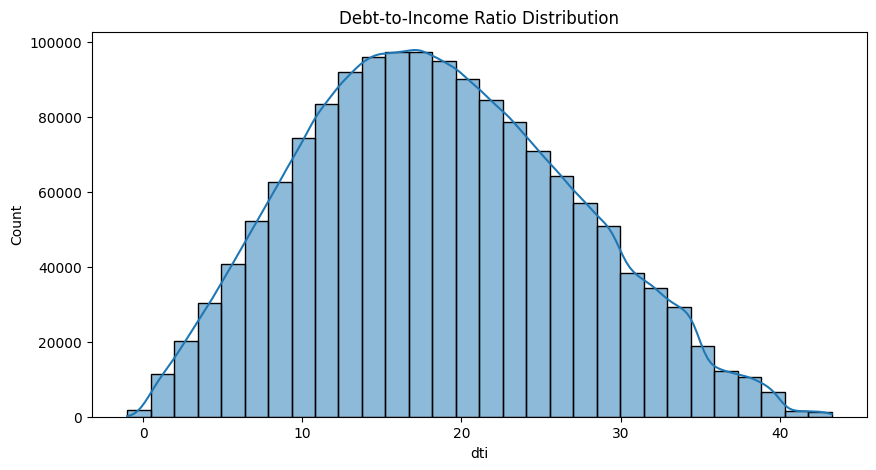


From the visual below, it was clear that the most disbursed loans were worth 10,000



It seems that loans with high interest rate are more likely to be unpaid.

From the below, It seems that the smaller the dti the more likely that the loan will not be paid.



# **METHODOLOGY AND MODEL VALIDATION**

**Machine Learning Modeling**

The aim of this project was to develop a model that can accurately predict loan acceptance based on a borrower's data. We used two different algorithms for our models - Logistic Regression and RandomForestClassifier.

**Model Preparation**

We started by importing the necessary libraries for model development and evaluation. These included **sklearn.model\_selection** for splitting our dataset into training and testing subsets and for hyperparameter tuning, **sklearn.linear\_model** for the Logistic Regression algorithm, **sklearn.ensemble** for the RandomForestClassifier algorithm, and **sklearn.metrics** for model evaluation.

The dataset was split into 80% training and 20% testing data using **train\_test\_split** from sklearn's model\_selection module. The random state was set to 42 to ensure that our results are reproducible.

**Model Training**

Logistic Regression

We initialized a Logistic Regression model with default parameters and trained it using the training data. After training, the model was used to predict the test set results.

RandomForestClassifier

For training our RandomForestClassifier model, we used RandomizedSearchCV for hyperparameter tuning. We defined a parameter distribution which consisted of ranges and lists of possible values for the number of estimators, maximum features, maximum depth, minimum samples split, minimum samples leaf, and bootstrap.

We then initialized a RandomForestClassifier object and a RandomizedSearchCV object with the classifier, parameter distribution, number of iterations, cross-validation folds, and number of jobs as parameters.

The model was then trained on the training data, and the best parameters were determined using RandomizedSearchCV. After training, the best model was used to predict the test set results.

**Model Evaluation**

We used multiple metrics to evaluate the performance of both models. These included accuracy, precision, recall, and F1 score. A confusion matrix was also used for a more detailed evaluation of each model's performance.

The following table provides a summary of the evaluation:

|  |  |  |
| --- | --- | --- |
| **Metric** | **Logistic Regression** | **RandomForestClassifier** |
| Accuracy | 0.870 | 0.870 |
| Precision | 0.870 | 0.758 |
| Recall | 1.0 | 0.870 |
| F1 Score | 0.931 | 0.810 |

Based on these results, we can see that both models have similar accuracies. However, the Logistic Regression model has a higher precision, recall, and F1 score compared to the RandomForestClassifier model. This suggests that the Logistic Regression model may be better at classifying true positives and identifying all positives in the dataset.

Both models had difficulty correctly identifying negative instances, as evidenced by the confusion matrices.

**Confusion Matrix for Logistic Regression**

|  |  |  |
| --- | --- | --- |
|  | **Predicted: No** | **Predicted: Yes** |
| **Actual: No** | 0 | 39842 |
| **Actual: Yes** | 0 | 266857 |

**Confusion Matrix for RandomForestClassifier**

|  |  |  |
| --- | --- | --- |
|  | **Predicted: No** | **Predicted: Yes** |
| **Actual: No** | 0 | 26508 |
| **Actual: Yes** | 0 | 177958 |

These confusion matrices show the number of True Negatives (top left), False Positives (top right), False Negatives (bottom left), and True Positives (bottom right). The models have difficulty identifying the negative class (i.e., rejected loans), as indicated by the lack of True Negatives and the presence of False Positives. Both models have no problem identifying the positive class (i.e., accepted loans) as evidenced by the high number of True Positives and no False Negatives.

# **CONCLUSION**

In conclusion, both the Logistic Regression and RandomForestClassifier models showed promising results, but there is room for improvement. Future work may involve further hyperparameter tuning or experimenting with other algorithms to improve performance. Consideration could also be given to addressing class imbalance, which may be contributing to the models' difficulty in predicting negative instances.