Home Credit Default Risk Recognition

1.0 Project Background

A significant portion of the population struggles to secure home loans due to limited or non-existent credit histories. This can prevent them from purchasing their own homes and may force them to turn to unreliable or high-interest financial sources. On the other hand, financial institutions face challenges in accurately assessing who should be approved for loans. Relying solely on credit history is not always effective—individuals with long credit histories may still default, while others without much history may be reliable borrowers.

To address this, machine learning techniques can be used to build predictive models that go beyond traditional credit scoring. By analyzing a broader set of features, such models can more accurately assess the risk of loan default. This approach can support both borrowers in accessing fair credit and lenders in making more informed decisions.

2.0 Problem Statement and Solution Strategy

The task is a **binary classification problem** where the goal is to predict whether a loan applicant will **repay the loan** or **default**, based on a wide range of financial and behavioral features.

To solve this, the workflow is structured into five key stages:

1. **Data Preparation**
   The datasets are imported, missing values are handled, and categorical variables are encoded to prepare the data for modeling.
2. **Exploratory Data Analysis (EDA)**
   Statistical summaries and visualizations are used to understand the data distribution, detect patterns, and uncover important relationships among variables.
3. **Feature Engineering**
   New features are created by combining existing ones and linking data across multiple related tables to enrich the dataset.
4. **Model Training and Evaluation**
   Several classification algorithms, such as logistic regression, random forest, and gradient boosting, are trained using different libraries. The performance of each model is compared to select the most suitable one.
5. **Hyperparameter Tuning**
   The selected model is further optimized using techniques like K-fold cross-validation and tuning strategies such as grid search, random search, or Bayesian optimization.

This structured approach aims to build an accurate and robust predictive model that can assist financial institutions in making more informed and fair loan approval decisions.

3.0 Datasets and Inputs

The dataset used in this project is sourced from the Kaggle Home Credit Default Risk competition. It consists of multiple CSV files, each representing different aspects of a loan applicant's financial history and behavior.

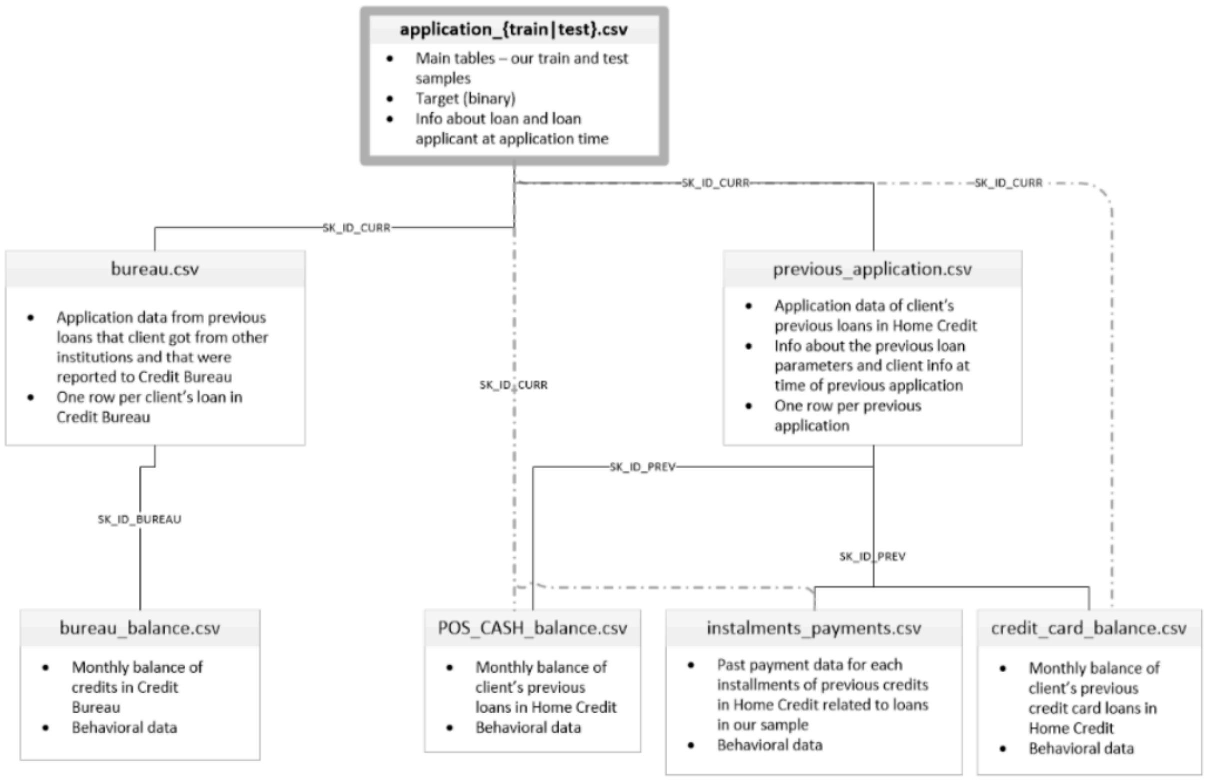
The main file, `application_{train|test}.csv`, contains the core information for each loan application. The training set includes **307,511 records**, and the test set contains **48,744 records**. Each row corresponds to a unique loan, identified by the `SK_ID_CURR` feature. The training data includes a `TARGET` column with binary values — `0` indicates the loan was repaid, and `1` indicates it was not. Since the test set lacks labeled outcomes, a portion of the training data was reserved for validation during model development.

Additional related datasets include:

- `bureau.csv` and `bureau_balance.csv`
- `previous_application.csv`
- `POS_CASH_balance.csv`
- `installments_payment.csv`
- `credit_card_balance.csv`

These supplementary datasets provide detailed information on a client's previous loans, credit card behavior, installment payments, and other financial activities. They are linked to the main application dataset through a shared identifier (`SK_ID_CURR` or related keys). Integrating these files requires Extract, Transform, Load (ETL) processes, treating the data as a relational database to ensure proper merging and consistency.

By leveraging the richness of these multiple data sources, the model can uncover deeper patterns and correlations, potentially improving the accuracy of credit default risk classification.


application_{train|test}.csv
Main tables – our train and test samples
Target (binary)
Info about loan and loan applicant at application time
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
bureau.csv
Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
One row per client's loan in Credit Bureau
previous_application.csv
Application data of client's previous loans in Home Credit
Info about the previous loan parameters and client info at time of previous application
One row per previous application
SK_ID_BUREAU
SK_ID_PREV
SK_ID_PREV
bureau_balance.csv
Monthly balance of credits in Credit Bureau
Behavioral data
POS_CASH_balance.csv
Monthly balance of client's previous loans in Home Credit
Behavioral data
instalments_payments.csv
Past payment data for each installments of previous credits in Home Credit related to loans in our sample
Behavioral data
credit_card_balance.csv
Monthly balance of client's previous credit card loans in Home Credit
Behavioral data

4.0 Benchmark Model and Evaluation Metrics

To evaluate the performance of the binary classification models, **Receiver Operating Characteristic (ROC) Curve** and **Area Under the Curve (AUC)** were used as the primary evaluation metrics. The ROC curve plots the **True Positive Rate** against the **False Positive Rate** across different thresholds, making it well-suited for imbalanced datasets, as it is not biased toward either class.

The **AUC score**, ranging from 0 to 1, provides a single metric to compare model performance across all classification thresholds. Given the importance of handling class imbalance in loan default prediction, AUC was chosen as the main evaluation metric.

In addition to AUC, the following metrics were also considered:

- **Accuracy**: The proportion of correct predictions out of total predictions. However, it can be misleading for imbalanced datasets.
- **F1 Score**: The harmonic mean of **precision** and **recall**, offering a better measure when the class distribution is uneven.

As a benchmark:

- A top-performing solution on the original competition leaderboard achieved an **AUC of 0.807**.
- The best model in this project achieved an **AUC of 0.7860** on the test dataset, which is a strong result given the limitations in domain-specific feature engineering and access to the actual competition test set.
- A baseline model using logistic regression on the raw training dataset achieved an **AUC of 0.7439**, serving as a reference point for improvement.

5.0 Project Design and Solution

The project is developed using **Python 3** within a **Jupyter Notebook** environment. The notebook provides a complete end-to-end workflow for building a **binary classification model** to predict credit default risk.

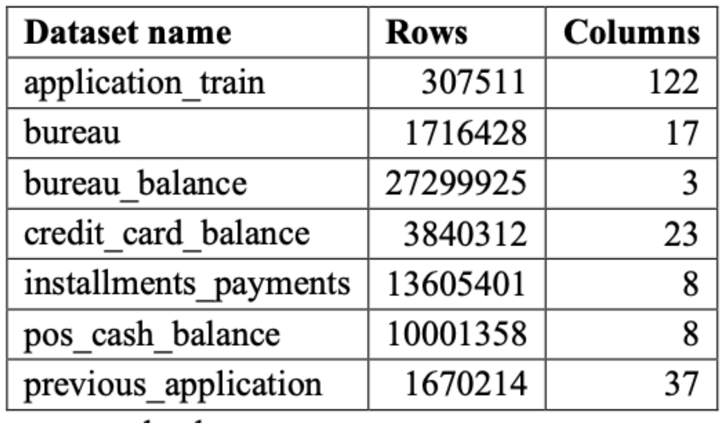
Key components of the solution include:

- **Automated Feature Engineering**: To efficiently connect and process data from multiple relational CSV files, transforming them into a format suitable for machine learning.
- **Classifier Comparison**: Multiple machine learning algorithms, including **XGBoost** and **LightGBM**, are evaluated for performance on the imbalanced dataset.
- **Hyperparameter Tuning**: Advanced techniques like **Gridsearchcv** are used to fine-tune model parameters and improve prediction accuracy.

These steps collectively aim to enhance model performance and identify the most effective classifier for predicting loan defaults. The project is structured into **five main parts**, which are detailed in the following sections.

5.1 Data Preparation

As the first step, the necessary libraries and the datasets are imported. Since there are more than one file, all need to be imported before looking at the feature types and number of rows/columns in eachfile. For the main training data, there are 307511 total samples (each row a separate loan) with 122 features of types 41 integer, 65 float and 16 object data types. Out of these, the feature (SK_ID_CURR)serves as the index and TARGET is the response feature to be predicted. The dataset file names, number of rows and columns is summarized in Table 1.



Dataset name	Rows	Columns
application_train	307511	122
bureau	1716428	17
bureau_balance	27299925	3
credit_card_balance	3840312	23
installments_payments	13605401	8
pos_cash_balance	10001358	8
previous_application	1670214	37

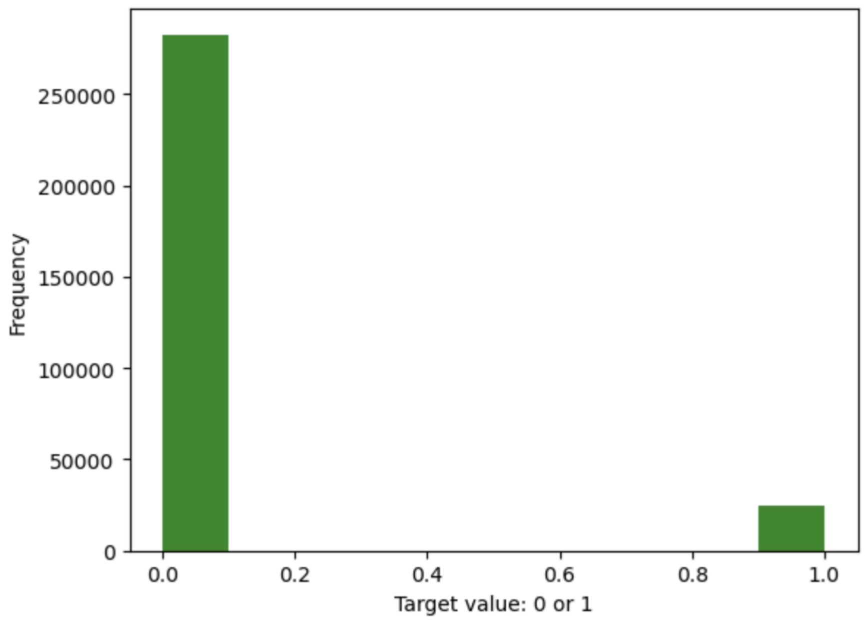
5.2 Exploratory Data Analysis

Once the data is imported, the first step is to assess its quality and explore key patterns. Analyzing the distribution of the target variable reveals a significant class imbalance:

- **Loan Repaid:** 282,686 instances
- **Loan Defaulted:** 24,825 instances

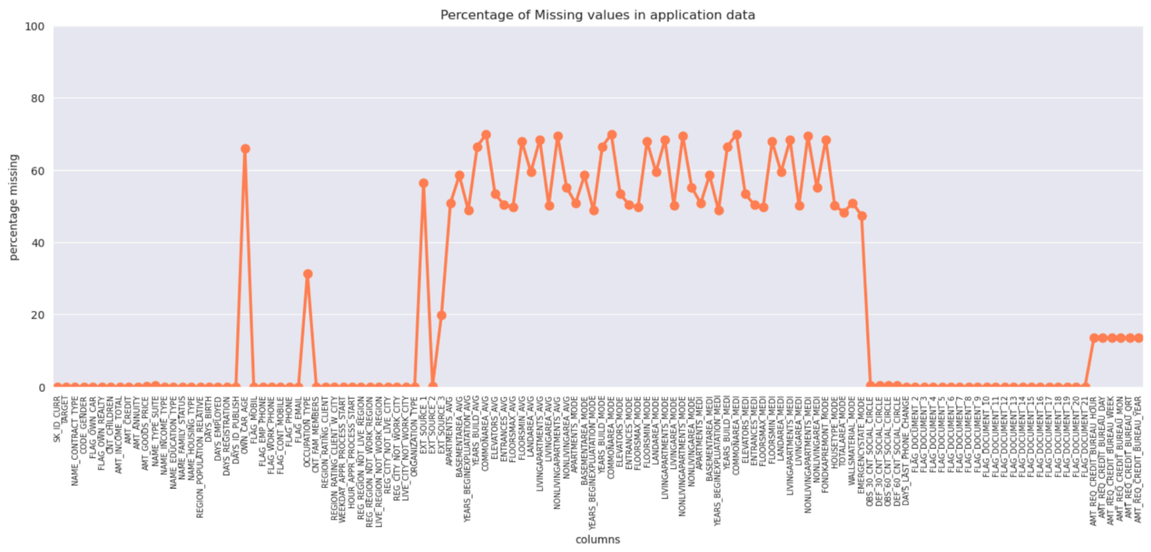
This means the number of repaid loans is more than 10 times higher than defaults. Such an imbalance must be carefully addressed during model training to ensure the classifier doesn't become biased toward the majority class.

```
There are 282686 loans repaid on time (TARGET=0) and 24825 loans defaulted (TARGET=1) in the dataset
```



The **TARGET** variable represents the loan status, where `0` indicates the loan was repaid and `1` indicates default. As shown in Figure , there is a noticeable class imbalance, with significantly more repaid loans than defaults.

An analysis of **missing values** (Figure 3) reveals that several features in the main dataset (`application_train.csv`) have nearly **60% missing data**. Features with excessive missingness are dropped, while those with moderate missing values are **imputed** using appropriate techniques to preserve data integrity. Similar missing data checks and handling steps are applied to the other related datasets to ensure consistent and clean input for model training.



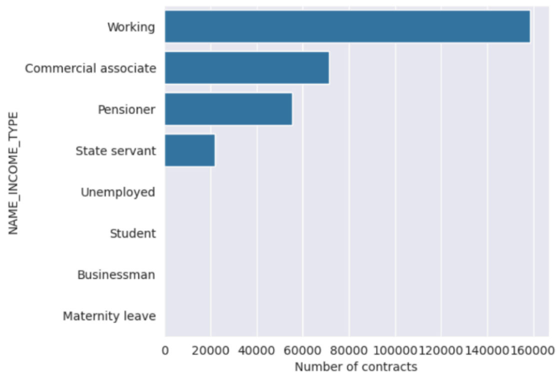
As part of the exploratory data analysis, missing values were examined across all features. Figure 3 visualizes the percentage of missing data in the main dataset, helping to identify which features may require imputation, exclusion, or further attention during preprocessing.

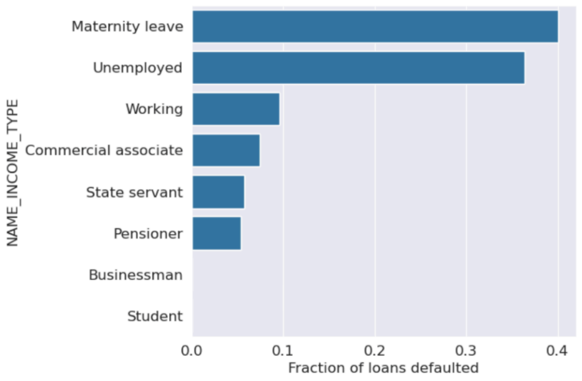
Categorical Feature Analysis

To further understand patterns related to loan repayment and default, categorical features were analyzed using bar plots. Each plot shows:
a) The total number of applicants in each category, and
b) The proportion of defaults within each category.

For instance, in the **NAME_INCOME_TYPE** feature (Figure 4), most loans are taken by applicants who are working. However, the **default rate is noticeably higher** among those who are on **maternity leave** or **unemployed**, indicating a potential risk factor.

A complete set of such categorical comparisons is available in the Jupyter notebook, offering detailed insights for feature selection and risk profiling.





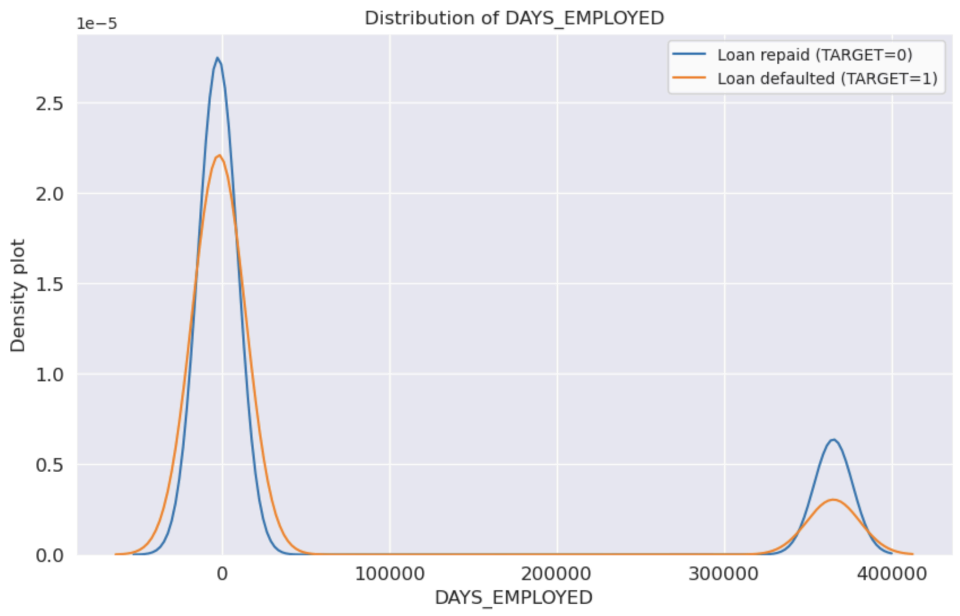
Categorical features were analyzed to observe how loan defaults vary across different applicant profiles. For instance, in the case of **income type**, most loans are sanctioned to working individuals. However, the **default rate is noticeably higher** among applicants who are **unemployed** or on **maternity leave**, indicating these groups may carry higher credit risk.

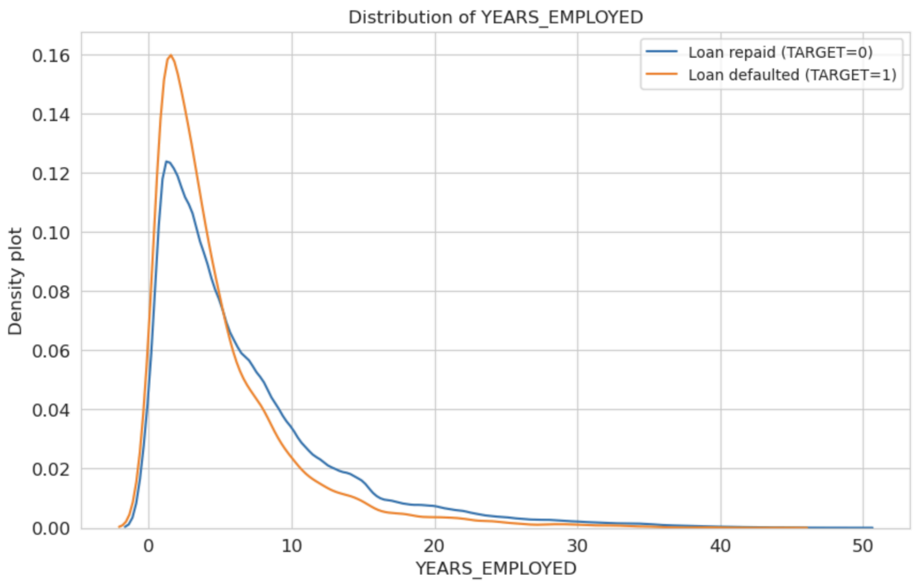
A) Raw distribution of `DAYS_EMPLOYED`
B) Cleaned and converted distribution (in years)

Continuous features were analyzed to compare patterns between repaid and defaulted loans. The `DAYS_EMPLOYED` feature initially showed extreme outliers, with some values exceeding 350,000 days (over 950 years), which are clearly invalid.

After removing these outliers and converting days into years, the revised distribution (Figure 5B) became more interpretable. It was observed that **loan defaults are more common among applicants with fewer years of employment**.

This analysis helps in identifying data quality issues and uncovering trends that can enhance model performance. Additional distribution plots for other continuous features are available in the Jupyter notebook.





5.3 Feature Engineering

After exploring the data, feature engineering was carried out to transform and prepare the dataset for model training. Key steps included:

- **Outlier Treatment:** Outliers beyond three standard deviations (e.g., extreme values in `DAYS_EMPLOYED`) were removed to improve data quality.
- **Missing Value Handling:**
    - Features with over **60% missing data** were dropped.
    - For remaining features, **categorical values** were filled with the **most frequent category**, and **numerical values** were filled with the **median**.
- **Categorical Encoding:**
    - **One-hot encoding** was used over label encoding to avoid misleading ordinal relationships between categories.
- **Feature Scaling:** All numerical features were **rescaled to a 0–1 range** to standardize the input for model training.
- **Domain-Specific Features:** Additional features were engineered using domain knowledge, such as:
    - **Credit-to-income ratio**
    - **Annuity-to-income ratio**
    - **Loan term (credit/annuity)**
    - **Employment duration fraction**
- **Automated Feature Engineering:**
  Using **Featuretools**, automated deep feature synthesis (DFS) was performed across multiple relational datasets.
    - Entities were connected via primary keys (e.g., `SK_ID_CURR`) to form an **entity set**.
    - **Aggregation** and **transformation primitives** like `mean`, `sum`, `difference`, `count`, and `trend` were applied to generate features.
    - This process resulted in **3,082 new features**.
- **Feature Reduction:**
    - Features with **low variance**, **high missing rates**, or **strong correlation** (threshold > 0.8) were removed to avoid redundancy and mitigate the **curse of dimensionality**.

After cleaning, imputing, encoding, and scaling, the final dataset was ready for model training with a refined set of informative features.

5.4 Classifier Models: Training, Prediction, and Comparison

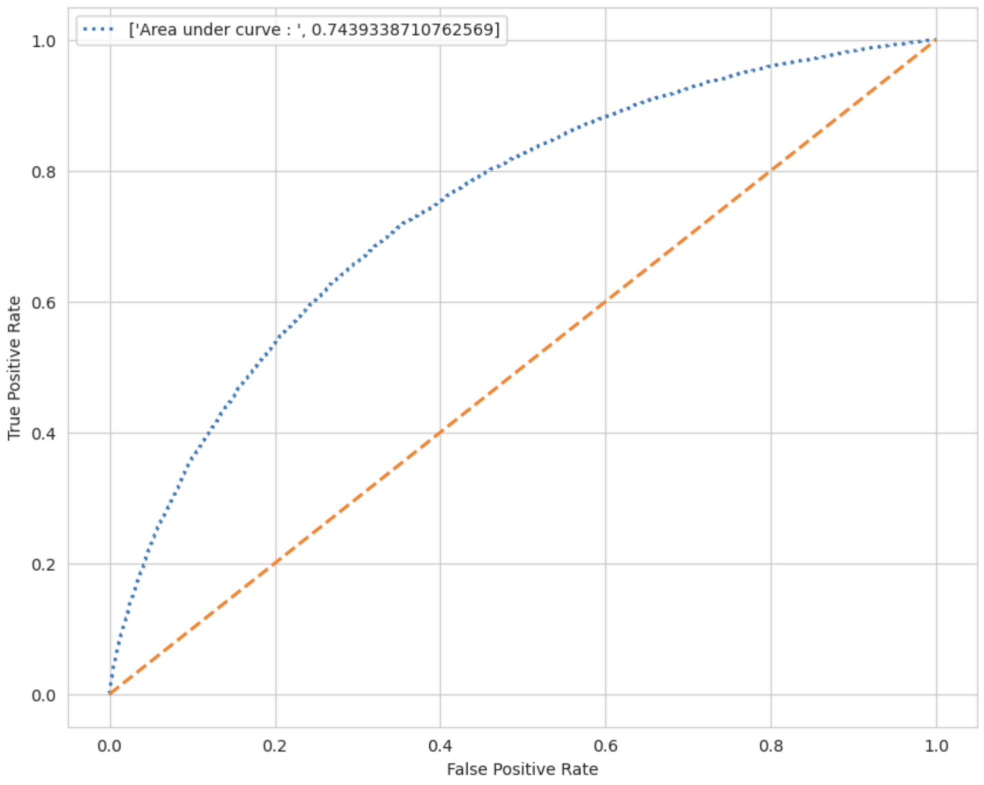
To establish a baseline, a **logistic regression model** was initially trained using only the main application dataset. This base model achieved a **high accuracy of 0.9184** and performed well in predicting loan repayment cases (**F1-score = 0.96**). However, due to the **imbalance in the dataset**, it failed to effectively predict defaults (**F1-score = 0.02**).

To address this limitation, **AUC-ROC** was used as the primary evaluation metric, as it focuses on the quality of prediction probabilities rather than class labels—making it suitable for imbalanced datasets. As shown in **Figure** , the base logistic regression model achieved an **AUC score of 0.7439** on the test data.

The final dataset was then split into **training and testing sets (75:25 ratio)**. To reduce the class imbalance—where non-default cases (TARGET=0) outnumber default cases (TARGET=1) by over 10 times—**random undersampling** was applied to the training data.

- The number of majority class samples was capped at **twice the number of minority class samples**, ensuring enough balance without discarding excessive data.

With this adjusted dataset, multiple classification models were trained and evaluated to identify the best-performing approach for predicting loan defaults more accurately.



5.4 Model Building

This section describes the implementation and performance of various classification algorithms used to predict loan default outcomes. All models were trained using default hyperparameters unless otherwise noted, and their performance was evaluated based on **accuracy**, **F1-score**, and **AUC** metrics to account for class imbalance.

5.4.1 Logistic Regression

Logistic Regression is a supervised classification algorithm that models the probability of a binary outcome using a logistic (sigmoid) function. The linear combination of features is transformed using the sigmoid function as follows:

- Linear equation:
  $t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_{□} X_{□}$
- Sigmoid function:
  $\sigma(t) = 1 / (1 + e^{-t})$

The Logistic Regression model was implemented using the **Scikit-learn** library. When trained on the full dataset using default parameters, it achieved an **accuracy of 0.8413**, **F1-score of 0.3143**, and **AUC of 0.7657**. The poor F1-score and AUC indicate that the model failed to effectively handle the class imbalance, predicting the majority class predominantly.

5.4.2 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees using randomly sampled subsets of data and features. The final prediction is made through majority voting, which enhances accuracy and reduces overfitting.

The Random Forest model was implemented using **Scikit-learn** and achieved an **accuracy of 0.8846**, **F1-score of 0.2656**, and **AUC of 0.7355**. The improvement in F1-score and AUC over Logistic Regression reflects a better capability in capturing minority class instances.

5.4.3 Decision Tree

Decision Tree is a non-parametric algorithm that recursively partitions the dataset into subsets based on feature values to make predictions. While highly interpretable, it is prone to overfitting without proper regularization or pruning.

The Decision Tree model implemented via **Scikit-learn** yielded an **accuracy of 0.6943**, **F1-score of 0.1951**, and **AUC of 0.5850**. The lower accuracy is offset by improved detection of minority classes, albeit still suboptimal.

5.4.4 Gaussian Naïve Bayes

Gaussian Naïve Bayes applies Bayes' Theorem with the simplifying assumption of feature independence and Gaussian-distributed likelihoods. It is computationally efficient and performs well with high-dimensional data.

The model, implemented using **Scikit-learn**, resulted in an **accuracy of 0.2373**, **F1-score of 0.1625**, and **AUC of 0.5859**. Despite the high accuracy, the model failed to identify the minority class, evident from the F1-score of 0.0.

5.4.5 XGBoost

XGBoost is an optimized implementation of gradient boosting that builds sequential decision trees, focusing on correcting the errors of previous models. It supports parallel computation and regularization to improve both speed and performance.

Using the **XGBoost** library, the model achieved an **accuracy of 0.8284**, **F1-score of 0.3117**, and **AUC of 0.7598**. This demonstrates a significant improvement in detecting defaults while maintaining competitive accuracy.

5.4.6 Gradient Boosting

Gradient Boosting builds additive models sequentially using weak learners (typically decision trees) and optimizes a differentiable loss function. It is known for strong performance in structured data problems.

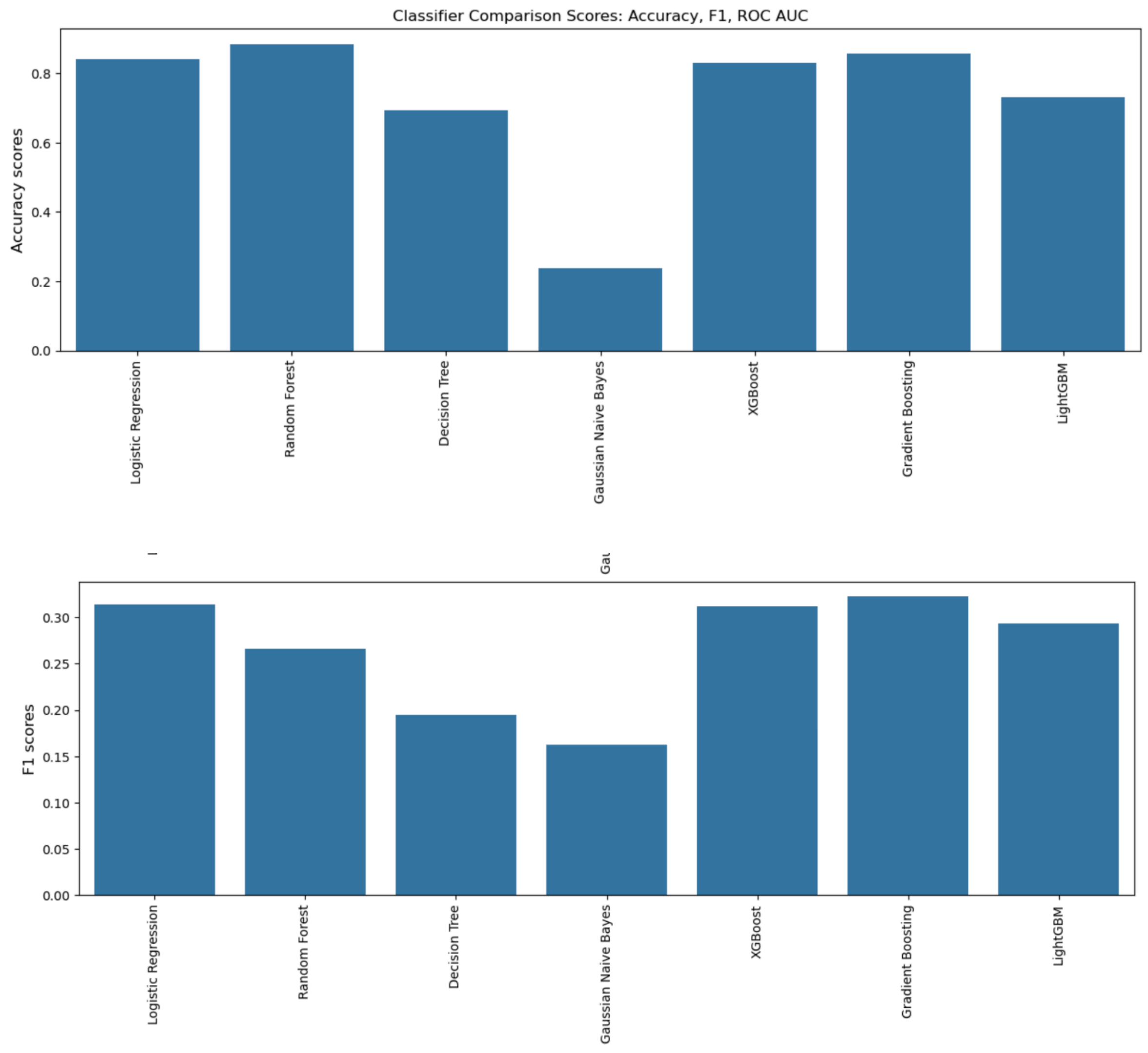
Implemented via **Scikit-learn**, this model achieved an **accuracy of 0.8566**, **F1-score of 0.3224**, and **AUC of 0.7716**, making it one of the better-performing models in terms of balancing accuracy and minority class detection.

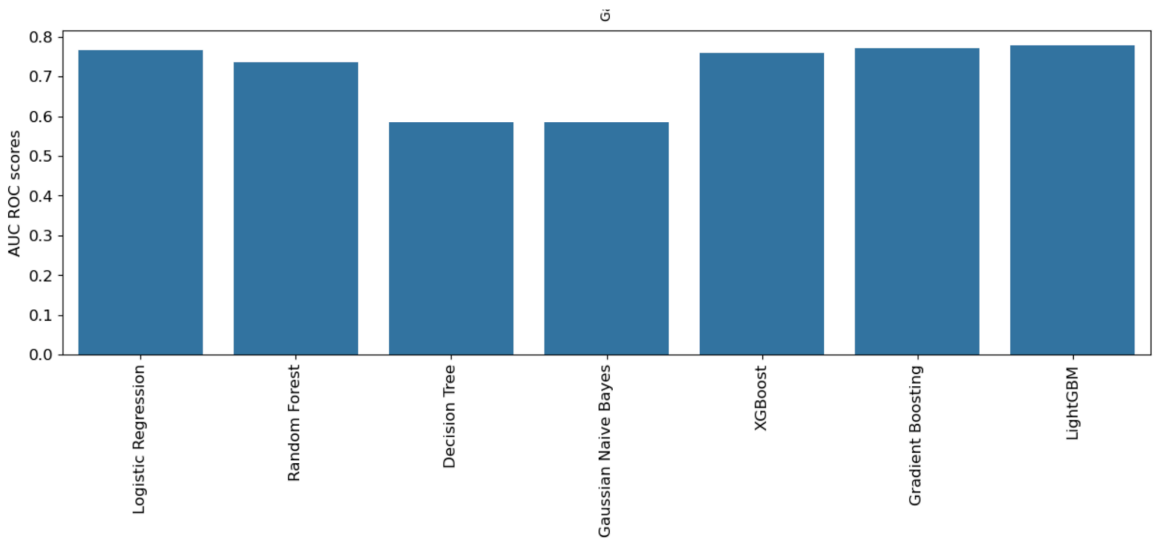
5.4.7 LightGBM

LightGBM is a high-performance gradient boosting framework that uses histogram-based algorithms and leaf-wise tree growth. It is designed for speed and efficiency, especially with large datasets.

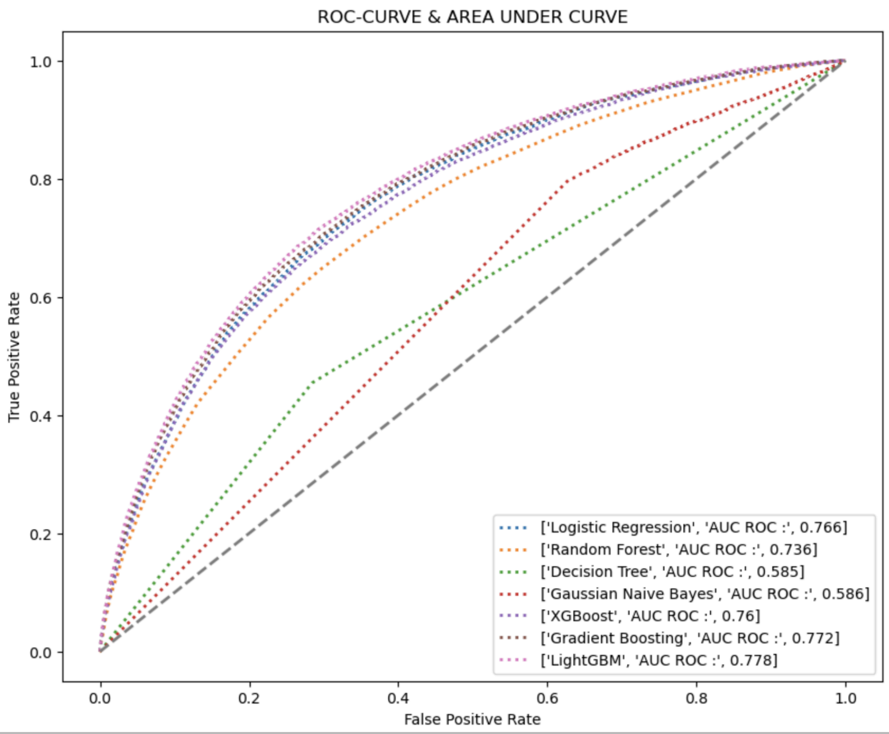
Using the **LightGBM** library with early stopping and validation split, the model produced an **accuracy of 0.7204**, **F1-score of 0.2934**, and **AUC of 0.7781**. Despite a slightly lower accuracy, the AUC and F1-score suggest strong discriminatory power in classifying defaults.

The performance of the different classifiers on the test data can be better compared using metrics like accuracy, F1-score, and ROC AUC (Figure 7). We can see from the comparisons of accuracy,F1 score and AUC ROC scores that all models have different rankings.





Among all classifiers, **LightGBM** shows the best balance across metrics—achieving the **highest F1-score and AUC**. While **Logistic Regression** and **Naive Bayes** have high accuracy, their low F1-scores indicate poor handling of imbalanced data. Models like **XGBoost** and **Random Forest** perform moderately well, but **LightGBM** clearly outperforms others in AUC (0.7781), making it the optimal choice.



After evaluating multiple classification algorithms, I compared their performance using the ROC AUC metric. As seen in the ROC curve above, **LightGBM** achieved the highest AUC score of **0.778**, followed closely by **Gradient Boosting (0.772)** and **Logistic Regression (0.766)**. Other models such as **XGBoost (0.76)** and **Random Forest (0.736)** also performed well, whereas **Decision Tree (0.585)** and **Gaussian Naive Bayes (0.586)** exhibited significantly lower performance.

Based on this comparison, LightGBM was chosen as the most effective classifier. To further validate this choice, I employed **K-fold cross-validation**, a technique that splits the dataset into K subsets, ensuring each subset is used once as a validation set while the rest are used for training. This method provided a robust estimation of model performance without needing a separate validation set. Using this approach, I fine-tuned the parameters and obtained an improved **AUC score of 0.7834**, confirming LightGBM as the optimal model.

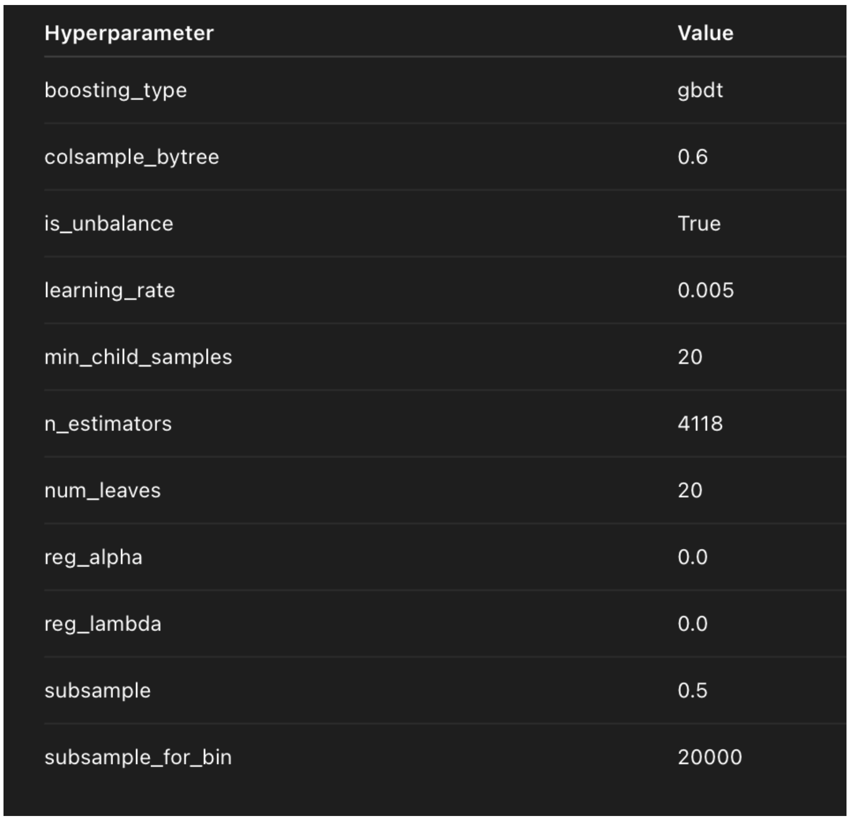
5.5 Hyperparameter Tuning

To further improve model performance, I applied hyperparameter tuning techniques including Grid Search. These methods aim to optimize an objective function over a defined domain space of hyperparameters using search strategies.

5.5.1 Grid Search

Grid Search performs an exhaustive search over all possible combinations of hyperparameters by incrementally testing each value. While this ensures comprehensive coverage, it is computationally expensive. Due to time and resource limitations, I restricted the maximum number of evaluations to **20**. Despite this constraint, Grid Search yielded a model with an improved **AUC score of 0.7837** on the test set.

The hyperparameters for the LightGBM model with the best validation AUC (random search) are summarized in **Table** . These parameters were used to generate the final predictions on the test dataset, including true and predicted labels along with their corresponding indices.



Hyperparameter	Value
boosting_type	gbdt
colsample_bytree	0.6
is_unbalance	True
learning_rate	0.005
min_child_samples	20
n_estimators	4118
num_leaves	20
reg_alpha	0.0
reg_lambda	0.0
subsample	0.5
subsample_for_bin	20000

Proof:

Preprocessing

ModelBuilding

References:

1.Introduction to Automated Feature Engineering. (2018). Kaggle.
https://www.kaggle.com/willkoehrsen/automated-feature-engineering-basics
2.Home Credit Default Risk Competition (2018). Kaggle.
https://www.kaggle.com/c/home-credit-default-risk/overview