|  |  |
| --- | --- |
|  | Batch No – MDTM28  Bank Risk Controller  Version 1.0        Author:  Sowmya Krishnakumar |

Contents

[Version Details: 2](#_Toc176411096)

[OBJECTIVE: 2](#_Toc176411097)

[Target Applications 2](#_Toc176411098)

[TECHNOLOGIES to be USED 2](#_Toc176411099)

[Process Steps: 2](#_Toc176411100)

# Version Details:

|  |  |  |  |
| --- | --- | --- | --- |
| DATE | VERSION | DESCRIPTION | Documented By |
| 15.03.2025 | 1.0 | Version1.0 (Bank Risk Controller) | Sowmya Krishnakumar |

# OBJECTIVE:

The objective of the Bank Risk Controller is to predict whether the customer will get default or not.

* The Bank Risk Controller System leverages advanced Machine Learning (ML) techniques to evaluate and predict financial risk within a banking environment.
* By analyzing historical transaction data, customer profiles, and external factors, the system aims to identify potential high-risk customers and prevent fraud.
* This project uses various ML models, including Random Forest, XGBoost, and Logistic Regression, to build a predictive framework for assessing the likelihood of defaults or fraudulent activity.
* The results of this analysis demonstrate significant improvements in risk prediction accuracy compared to traditional methods, providing a valuable tool for proactive risk management.

# PROBLEM STATEMENT:

Banks and financial institutions face a big challenge:

How to decide whether a customer applying for a loan will pay it back or fail to do so (default). If customers default on their loans, it can lead to financial losses for the bank. Predicting which customers are likely to default is important for making smarter loan approval decisions and reducing risks.

The goal of this project is to create a machine learning model that analyzes past loan data and predicts whether a new customer will default or not. The prediction is based on information like the customer’s income, loan amount, credit history, and other details. The target column in the dataset is called “TARGET”, where:

* 1 means the customer defaulted (did not repay the loan).
* 0 means the customer successfully repaid the loan.

# Target Applications

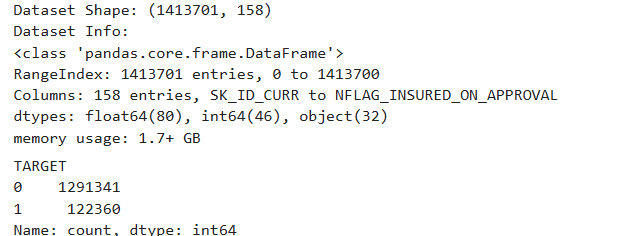
* Juypter Notebook
* Google Colab

# TECHNOLOGIES USED

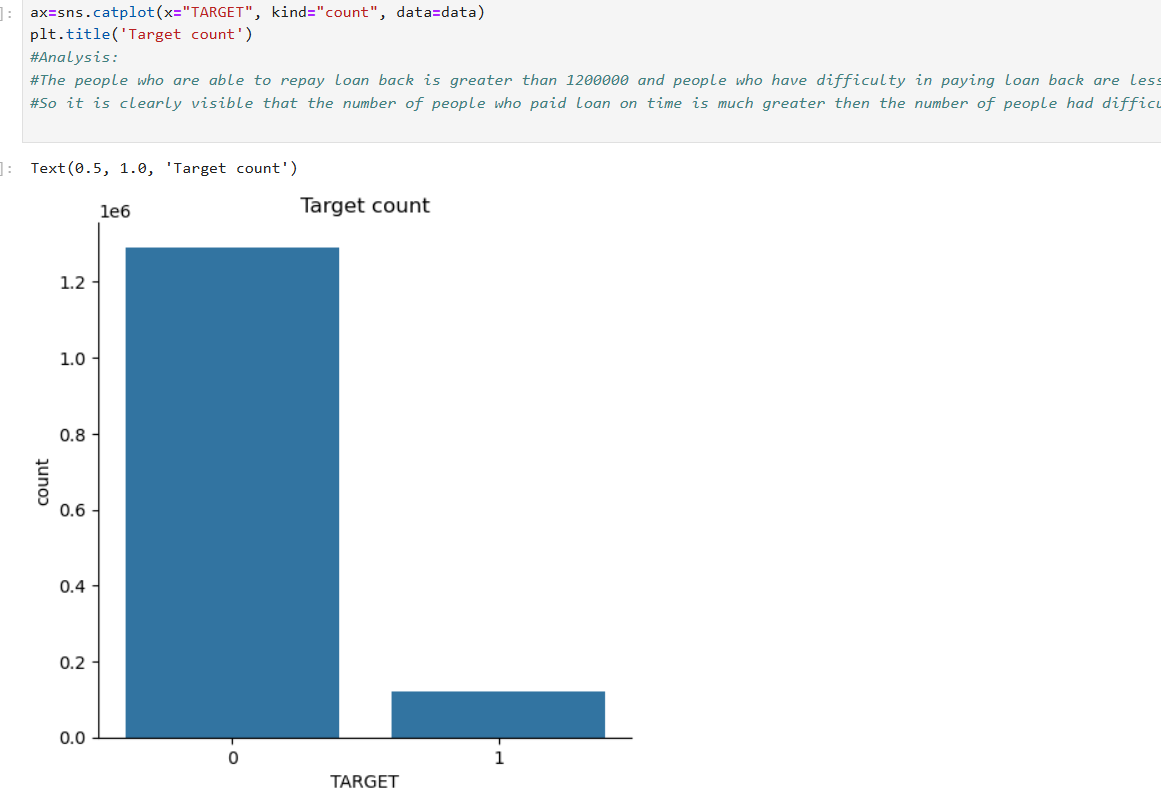
* Python
* Machine Learning
* Deep Learning

# Process Steps:

1. Read file using panda’s library.



1. Data had 1413701 rows and 158 columns.
2. In Target column client with payment difficulties are around 8% and others are 92%.

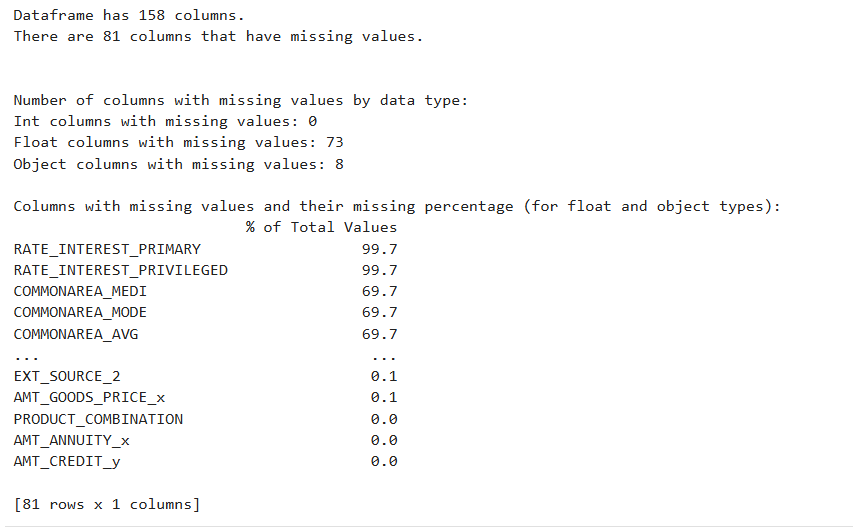


(Scenario 1) - If the model has predicted client will repay loan but actually, he has defaulted.

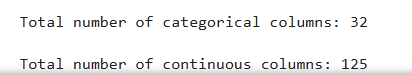
(Scenario 2) - If the model has predicted client will default but he can actually pay loan back.

* But the loss will be much more in Scenario 1 i.e. If the model has predicted client will repay loan but actually, he has defaulted.
* Accuracy cannot be used for imbalanced data, hence not helpful in our case. Precision will tell of all the points that are predicted to be positive how many of them are actually positive. Precision doesn't consider, the points that were actually positive and are predicted positive.
* Recall is important since it will tell of all the points that are actually positive how many of them are predicted positive. So recall is important to identify the Scenario 1. If recall is high means defaulters are correctly predicted as defaulters
* F1-score is geometric mean of precision and recall and precision and is high if recall and precision are high. Since we have no use of precision here, F1-score or micro F1-score or macro F1-score is not important.
* In ROC-AUC score we have True Positive Rate and False Positive Rate; True Positive Rate is same as Recall so Scenario 1 gets covered. False Positive Rate is out of all the points that are actually negative how many are predicted as positive Scenario 2.
* Hence ROC-AUC score is an important metric since it covers both scenarios where Credit can suffer loss. ROC-AUC score should be greater than 0.5 which means your model is doing something right.

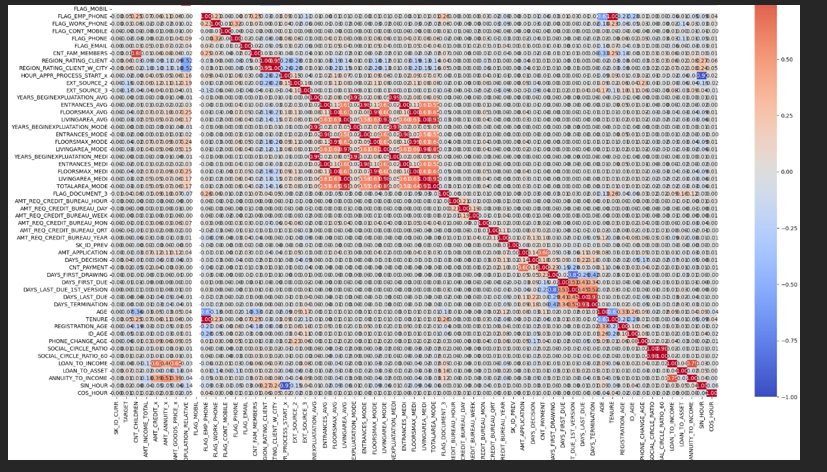
1. Ensure your dataset is complete and free of missing values, as they can skew or prevent accurate model building. Replace missing values using statistical techniques (mean, median, mode) For columns with excessive missing data, consider dropping them.



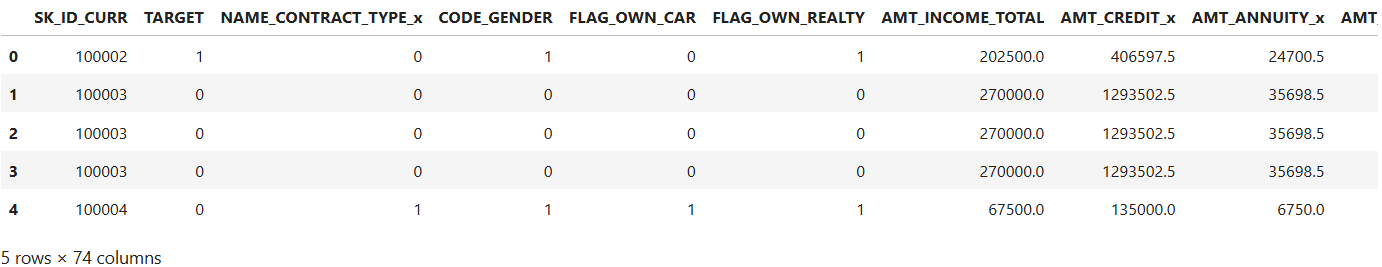
1. Handle categorical variables and numerical columns which may include missing data or require transformation for better model understanding. Replace missing categories with the mode (most frequent value), or use advanced techniques like KNN imputation for categorical variables.



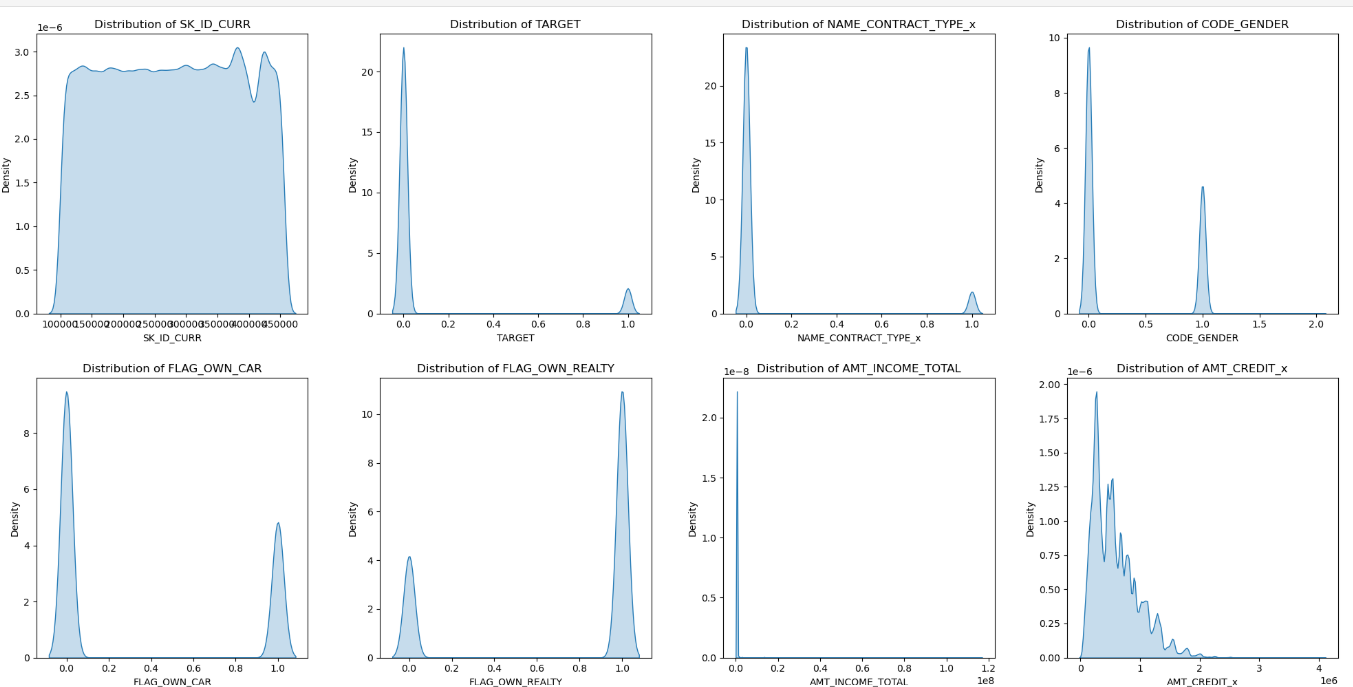
1. Remove duplicate rows to avoid biasing the model with repeated data. Identify duplicate rows based on feature values. Drop exact duplicates while maintaining relevant data for training.
2. Create new features or modify existing ones to improve model performance and reduce dimensionality. Derive new features from existing ones (e.g., ratio features, interaction features). Drop irrelevant or redundant features.
3. Impute missing values for numerical features using the median and for categorical features using the most frequent category.
4. Finally, check and correct data types for numerical columns.
5. Visualize the relationships between features and identify correlations. Generate a heatmap to display correlation coefficients between numerical features. Look for strong correlations between features and the target variable to identify important predictors.

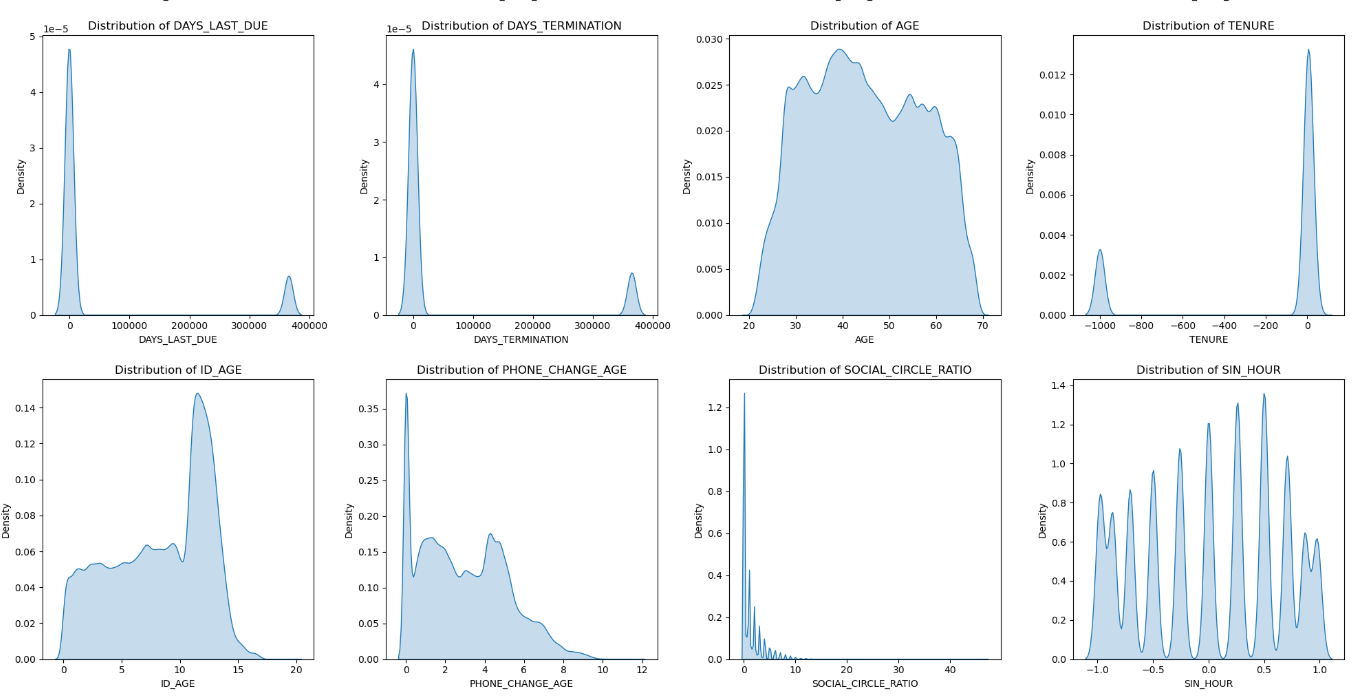


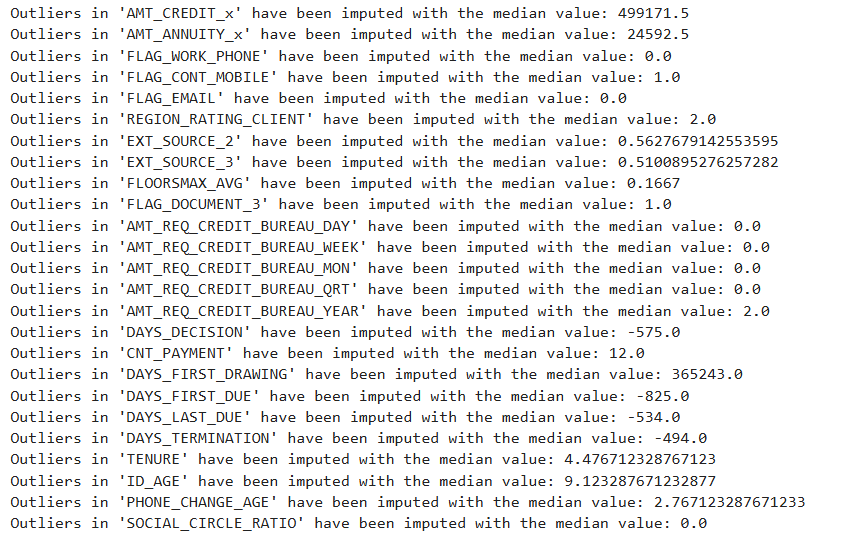
1. Convert categorical features into a numerical format suitable for machine learning models.



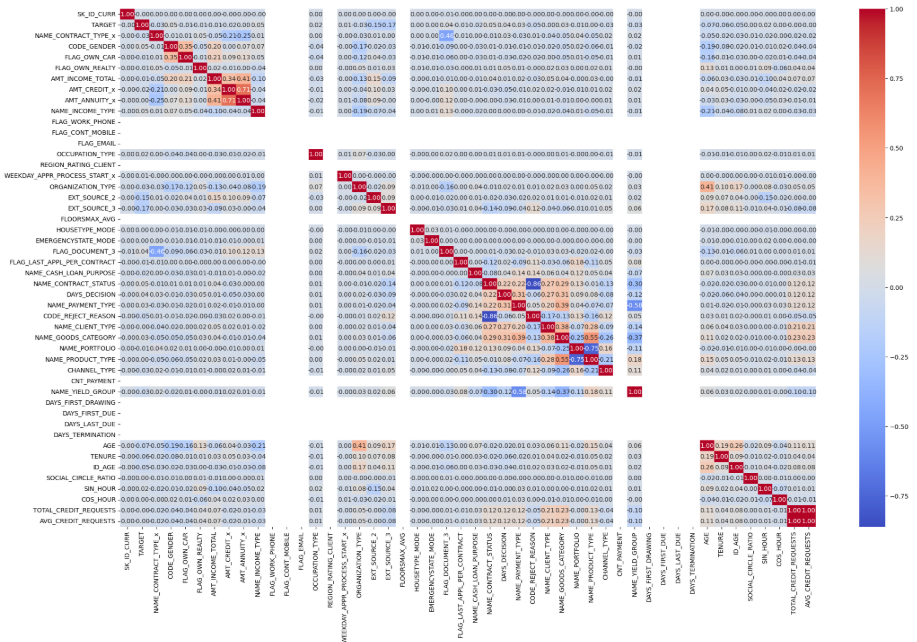
1. Identify and handle outliers that can distort statistical analysis and model training. Use statistical techniques (IQR) to identify outliers. Cap, transform, or remove outliers based on their impact on the model’s performance.



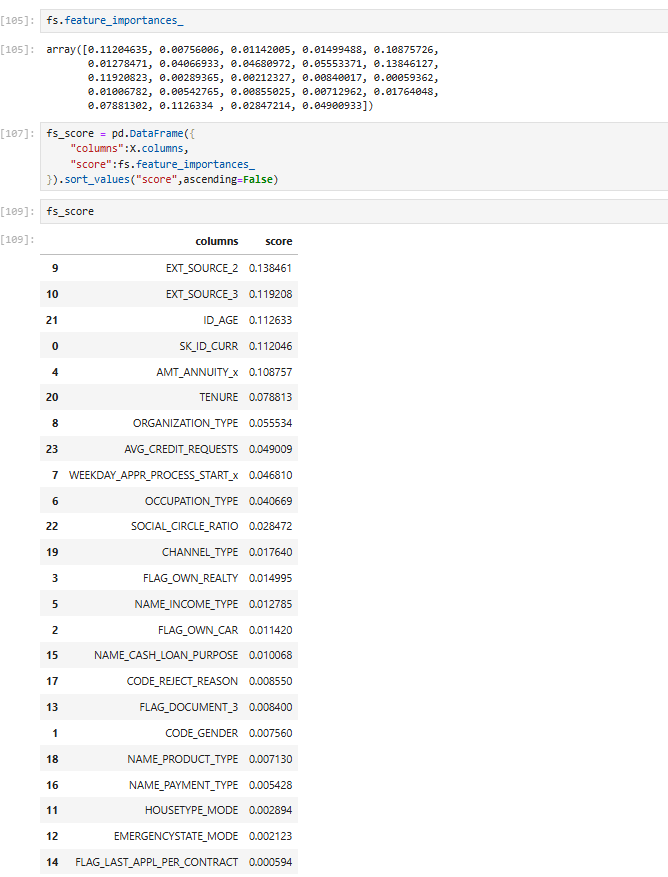




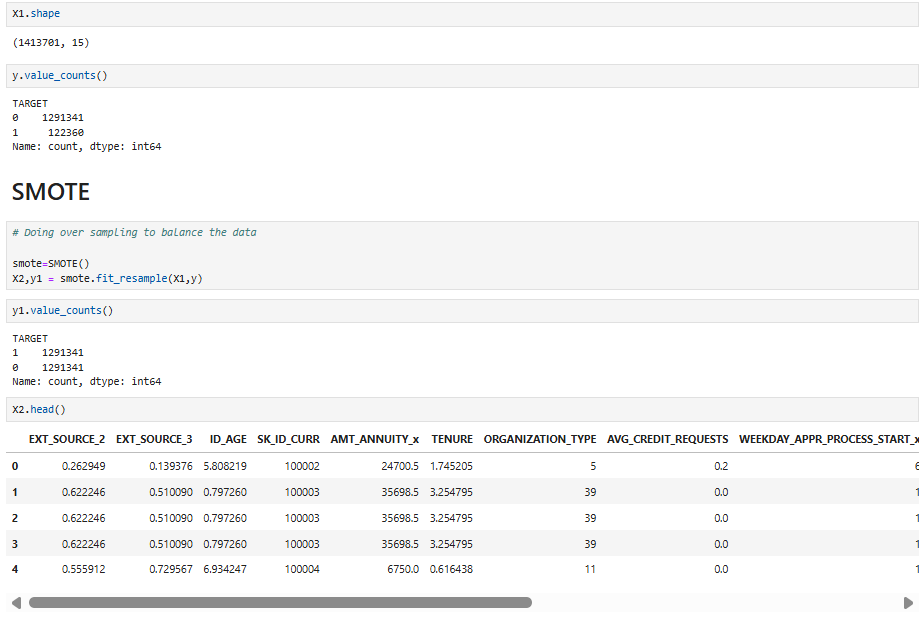
1. Visualize feature relationships after treating outliers and perform feature extraction. Create a new heatmap after handling outliers to reassess the correlation matrix. Focus on columns with the highest correlation with the target. Store the cleaned and processed dataset for model training. Save the dataset in a suitable format (CSV, Excel, etc.) to be used in training the machine learning model.



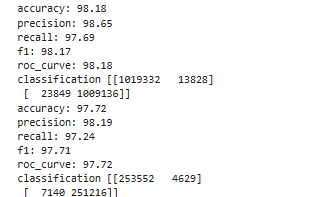
1. Evaluate the importance of features in predicting the target. Use models like Random Forest or statistical tests to assess which features contribute most to the prediction. Retain the most significant features and drop less important ones.



1. Address class imbalance by oversampling the minority class. Generate synthetic data points for the minority class to balance the dataset (SMOTE). Ensure the balance does not introduce noise or overfitting.



1. Divide the dataset into training and testing subsets to evaluate model performance.
2. Train machine learning models to predict the target variable. Use standard splits like 80/20 for training and testing data. Ensure that the split maintains class distribution, especially for imbalanced datasets. Choose appropriate algorithms based on the problem type (classification, regression) and dataset. Tune hyperparameters using techniques like XG Booster or Random Search to optimize performance.



1. Evaluate the performance of the model. Confusion Matrix to Assess the model’s accuracy, precision, recall, and F1-score. Calculate and interpret other metrics such as ROC-AUC for a better understanding of model effectiveness.

