MedTech
Mediterranean
Institute of Technology

Data Analytics Final Report

Credit Card Fraud Detection

Elaborated by:

Sahar Bahloul, Saifedine Magouri , Fares Bahroun , Mohamed Amine Ben Aoun , Mohamed Aziz Dardouri

Supervised by:

Mr.Hamza Mouharrar

Dr. Syrine Ben Meskina

Tunis, 2025-2026

Contents

1. One page summary 6

2. Introduction 7

3. Background and Literature Review 7

4. Methodology and Data Preprocessing 9

4.1 Target Variable Distribution 9

4.2 Missing Data Analysis 10

4.3 Summary Statistics 12

4.4 Numerical Feature Distributions 13

4.5 Categorical Feature Distributions 16

4.6 Pearson Correlation Analysis 18

4.7 Chi-Square & Cramér's V (Categorical Associations) 20

4.8 Multicollinearity (VIF Analysis) 23

4.9 Data Quality and Anomaly Checks 24

4.10 Data Preprocessing 25

5. Machine Learning Models 30

5.1 Baseline Models on the Original Stratified Split 31

5.2 Class-Wise Controlled Splitting Strategy 32

5.3 Modeling After Applying SMOTE 33

5.4 Ensemble Modeling 35

5.5 GPU-Accelerated CatBoost Hyperparameter Optimization 35

5.6 Final CatBoost Model (Clean Retraining After Tuning) 36

6. Results and Discussion 38

6.1 Overview of Evaluation Strategy 38

6.2 Baseline Model Performance (No SMOTE) 39
6.3 Controlled Class-Wise Split Results 39
6.4 SMOTE-Augmented Model Performance 40
Random Forest + SMOTE 40
XGBoost + SMOTE 40
LightGBM + SMOTE 41
CatBoost + SMOTE (first run) 41
6.5 Ensemble Model Performance 41
6.6 Hyperparameter Tuning: GPU-Accelerated CatBoost 42
6.7 Leakage Correction and Final CatBoost Retraining (Best Model) 42
Final CatBoost Performance 43
6.8 Comparative Summary of All Models 43
6.9 Limitations and Challenges 44
6.10 Final Conclusion 44
7. Model Deployment and Streamlit Application 44
7.1 Purpose of the Streamlit Application 45
7.2 Model Packaging and Artifacts 45
7.3 Application Architecture and Workflow 46
7.4 User Interface Design 47
7.5 Deployment Advantages 47
7.6 Challenges Encountered 48
7.7 Impact 48
Conclusions 51
Appendices 53
References 58

List of Tables

Table 1 Top Features with Highest Missing Data Rates 9
Table 2 Top Features by Pearson Correlation with TARGET 16
Table 3 Categorical Features Ranked by Cramér's V Association with TARGET 19
Table 4 Variance Inflation Factor (VIF) for Selected Numerical Features 21
Table 5 Summary of Data Quality Issues and Special-Case Values Identified in Key Columns 22

List of Figures

Figure 1 Top 40 Features by Missing Data Rate 10
Figure 2 Distribution of AMT_CREDIT (Overall and by Target Class) 11
Figure 3 Distribution of AMT_ANNUITY (Overall and by Target Class) 12
Figure 4 Distribution of EXT_SOURCE_1 (Overall and by Target Class) 12
Figure 5 Distribution of EXT_SOURCE_2 (Overall and by Target Class) 13
Figure 6 Distribution of EXT_SOURCE_3 (Overall and by Target Class) 13
Figure 7 Distribution of DAYS_BIRTH (Overall and by Target Class) 13
Figure 8 NAME_CONTRACT_TYPE Distribution by Target Class 14
Figure 9 CODE_GENDER Distribution by Target Class 14
Figure 10 FLAG_OWN_CAR Distribution by Target Class 15
Figure 11 FLAG_OWN_REALTY Distribution by Target Class 15
Figure 12 Top 25 Features by Absolute Pearson Correlation with TARGET 17
Figure 13 Top 25 Categorical Features by Cramér's V with TARGET 20
Figure 14 Correlation heatmap illustrating linear relationships across core numeric predictors used in modeling. 25
Figure 15 Single Customer Prediction Interface (Input View) 43
Figure 16 Default Risk Prediction Output (Low-Risk Example) 43
Figure 17 Single Customer Prediction Interface (High-Risk Input Example) 44
Figure 18 Default Risk Prediction Output (High-Risk Example) 44
Figure 19 Top 20 Feature Importances from the Random Forest Model 47

One page summary

This project presents a complete end-to-end machine learning pipeline for credit-risk prediction using a publicly available dataset sourced from **Kaggle**. The main objective is to develop a reliable model capable of predicting the likelihood of client default in the presence of extreme class imbalance. The work includes data preprocessing, exploratory analysis, imbalance handling, model experimentation, hyperparameter tuning, threshold optimization, and interactive deployment.

The methodology integrates several components: (1) data cleaning, removal of constants, handling of missing values, scaling, correlation analysis, and transformation of skewed variables; (2) multiple splitting strategies, including standard stratified splits and class-wise splits; (3) resampling via **SMOTE** to address the severe minority-class underrepresentation; (4) evaluation of several machine learning algorithms including Logistic Regression, Random Forest, XGBoost, LightGBM, and CatBoost; (5) model selection based on ROC-AUC, PR-AUC, confusion matrices, and threshold tuning techniques (default threshold, Youden index, best-F1, and precision-oriented thresholds).

Findings indicate that **CatBoost trained on SMOTE-resampled data** consistently outperformed other models, offering the best balance between recall, precision, and overall discrimination ability. Additional experiments revealed the importance of data splitting strategies, the benefit of synthetic oversampling, and the strong impact of decision threshold adjustments.

The final model was deployed as an interactive **Streamlit application**, enabling real-time predictions on new applicant data through CSV upload or manual input. The deployment demonstrates how machine learning can support practical decision-making in financial risk assessment.

This work is relevant to both academia and industry: academically, it highlights the interplay between imbalance techniques and boosting algorithms in tabular credit data; industrially, it showcases a deployable pipeline that can assist lenders, analysts, and credit officers in earlystage applicant screening.

Introduction

Machine learning has become an essential tool in modern financial analytics, particularly in automating and enhancing credit-risk assessment. With the growing availability of large-scale datasets—such as the Kaggle credit default dataset used in this study—financial institutions now rely on predictive modeling to identify high-risk clients more accurately and efficiently than traditional manual or rule-based systems.

The present project focuses on building a predictive model that estimates the likelihood of customer default. The dataset exhibits substantial class imbalance, a common challenge in credit-scoring problems. This requires specialized techniques to ensure that the minority class (defaulting clients) is adequately represented and learned by the model.

The main objectives of the project are:

- To explore and preprocess the Kaggle credit-risk dataset through cleaning, missingvalue handling, scaling, and correlation analysis.
- To test and compare multiple machine learning algorithms for binary classification.
- To implement strategies that address class imbalance, including SMOTE oversampling.
- To analyze the effect of different decision thresholds on business metrics (precision, recall, F1-score).
- To select the best-performing model based on robust evaluation metrics such as ROCAUC and PR-AUC.
- To deploy the final model through an interactive Streamlit interface for real-time inference.

The project highlights the practical importance of data analytics and machine learning in improving financial decision systems while demonstrating the end-to-end workflow required to build a deployable predictive tool.

Background and Literature Review

Modern credit-risk assessment has evolved significantly from its early dependence on classical statistical approaches such as logistic regression, which for decades dominated financial scoring systems due to its transparency, low computational cost, and regulatory acceptability.

However, contemporary datasets—exemplified by the Home Credit Default Risk dataset from Kaggle—pose substantial methodological challenges that exceed the capabilities of linear models. These datasets contain hundreds of variables, highly nonlinear relationships,

heterogeneous feature types, missingness patterns exceeding 50% in several dimensions, and extreme class imbalance, with defaulting borrowers typically representing less than 10% of observations. Academic literature consistently highlights that such characteristics degrade the performance of traditional classifiers and can lead to biased predictions that disproportionately favor the majority class. As a response, research in the past decade has gravitated toward more sophisticated ensemble-based and boosting-based machinelearning methodologies, including Random Forests, XGBoost, LightGBM, and CatBoost. These models demonstrate superior performance on tabular financial data because of their ability to capture nonlinear interactions, automatically handle complex feature hierarchies, and incorporate regularization mechanisms that prevent overfitting in high-dimensional spaces.

A particularly rich body of work addresses the imbalance problem, which remains one of the defining challenges in credit scoring and fraud detection. Numerous studies conclude that metrics such as accuracy and uncalibrated ROC results can be misleading in imbalanced contexts, and instead advocate for more sensitive indicators such as PR-AUC, recall, classspecific precision, and threshold-optimized F1-scores. Resampling techniques—especially SMOTE and its variants—have emerged as standard tools for addressing imbalance by synthetically generating minority samples while preserving feature-space geometry. Other works propose cost-sensitive learning and decision-threshold engineering as alternatives, enabling institutions to calibrate the model according to risk tolerance, operational costs, or regulatory constraints. The literature also highlights the importance of threshold strategies such as the Youden index, precision-driven cutoffs for high-risk lending, and the best-F1 threshold for balanced performance—particularly relevant in credit environments where false negatives carry substantial financial implications.

Recent scholarship has increasingly recognized CatBoost as a state-of-the-art model in creditrisk prediction due to its ability to natively process categorical variables, control overfitting through ordered boosting, and provide stable performance with minimal hyperparameter tuning. Studies comparing XGBoost, LightGBM, and CatBoost often find CatBoost competitively positioned or superior on datasets with heterogeneous, noisy, or partially missing categorical data—conditions that mirror real-world credit datasets. Beyond predictive modeling, a growing stream of research focuses on the operational integration of machine-learning systems, emphasizing model interpretability, reproducible preprocessing pipelines, and accessible user interfaces for decision-makers. Tools like Streamlit have gained recognition for enabling rapid deployment of machine-learning models into lightweight, interactive web applications, allowing practitioners, credit officers, and analysts to experiment with scenario-based predictions, explore model outputs, and operationalize analytics without coding expertise.

Within this context, the present project contributes to the literature on applied credit-risk modeling by combining: (1) advanced data preprocessing tailored to high-missingness financial datasets; (2) rigorous imbalance-handling strategies including SMOTE and threshold calibration; (3) comprehensive comparisons across classical, ensemble, and gradient-boosting models; (4) detailed evaluation using ROC-AUC, PR-AUC, recall, precision, and F1-score; and (5) deployment of the final CatBoost model through a Streamlit interface designed for realworld interpretability and usability. This integration positions the work firmly at the intersection of academic research and practical industry implementation, illustrating how modern machine-learning methodologies can transform raw credit data into actionable risk insights.

Methodology and Data Preprocessing

This section presents a comprehensive exploratory analysis of the dataset, covering target distribution, missing data patterns, feature distributions, statistical associations, correlations, multicollinearity, and data-quality issues. All findings below are based directly on the dataset and the visualizations, tables, and diagnostics performed during the analysis.

4.1 Target Variable Distribution

The dataset contains **307,511 loan applications** with a binary outcome variable (TARGET) indicating whether the client defaulted.

Distribution of TARGET:

- **Non-defaulters (TARGET = 0):** 282,686
- **Defaulters (TARGET = 1):** 24,825
- **Bad rate: 8.07%**

Plots of the count and percentage distribution show a heavily imbalanced dataset, with defaulters representing only around **8%** of the population. This strong imbalance (≈1:11) highlights the need for stratified sampling, class weighting, and appropriate evaluation metrics such as Precision-Recall AUC.

4.2 Missing Data Analysis

A missingness barplot of the **top 40 missing features** revealed that a large number of variables—mostly building and housing characteristics—have extremely high missing rates, often **between 50% and 70%**.

	column	dtype	missing_rate
48	COMMONAREA_AVG	float64	0.6987
62	COMMONAREA_MODE	float64	0.6987
76	COMMONAREA_MEDI	float64	0.6987
84	NONLIVINGAPARTMENTS_MEDI	float64	0.6943
70	NONLIVINGAPARTMENTS_MODE	float64	0.6943
56	NONLIVINGAPARTMENTS_AVG	float64	0.6943
86	FONDKAPREMONT_MODE	object	0.6839
54	LIVINGAPARTMENTS_AVG	float64	0.6835
82	LIVINGAPARTMENTS_MEDI	float64	0.6835
68	LIVINGAPARTMENTS_MODE	float64	0.6835
66	FLOORSMIN_MODE	float64	0.6785
52	FLOORSMIN_AVG	float64	0.6785
80	FLOORSMIN_MEDI	float64	0.6785
47	YEARS_BUILD_AVG	float64	0.6650
61	YEARS_BUILD_MODE	float64	0.6650
75	YEARS_BUILD_MEDI	float64	0.6650
21	OWN_CAR_AGE	float64	0.6599
81	LANDAREA_MEDI	float64	0.5938
53	LANDAREA_AVG	float64	0.5938
67	LANDAREA_MODE	float64	0.5938

Table 1 Top Features with Highest Missing Data Rates

These variables are highly sparse and require special handling through missing indicators, grouping or aggregating related features, or selective elimination depending on their predictive usefulness.

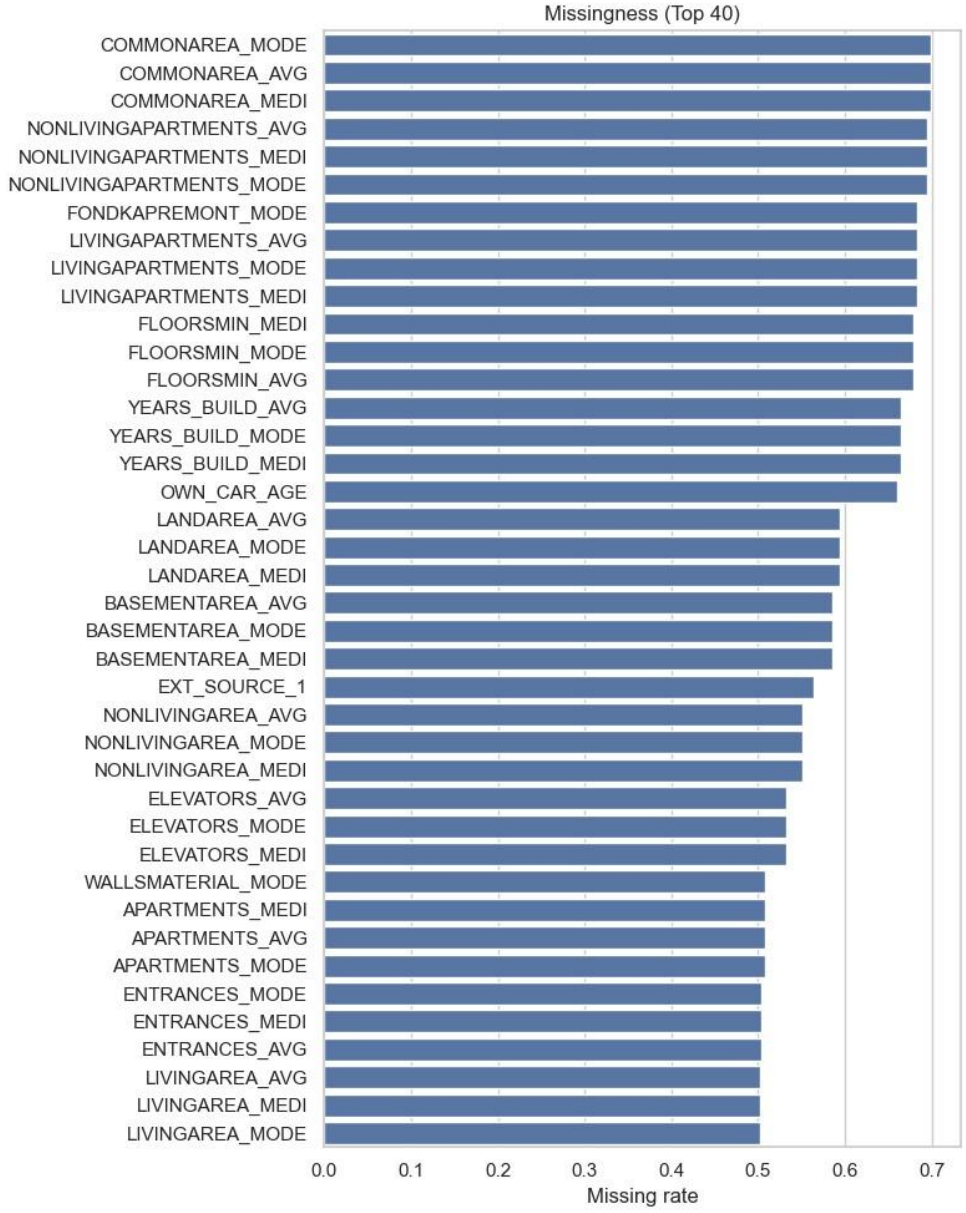


Figure 1 Top 40 Features by Missing Data Rate

4.3 Summary Statistics

Descriptive statistics were computed for key numeric variables:

Monetary Variables :

These features showed strong right-skew:

- **AMT_INCOME_TOTAL:** mean = 168,797; max = 117,000,000
- **AMT_CREDIT:** mean = 599,026; max = 4,050,000
- **AMT_ANNUITY:** mean = 27,108
- **AMT_GOODS_PRICE:** mean = 538,396

The presence of extreme values points to the need for winsorization and log transformation.

Temporal Variables :

- DAYS_BIRTH ranges from **−25,229 to −7,489**, representing client age.
- DAYS_EMPLOYED contains a placeholder value **365243**, used to indicate unknown employment duration.

These findings confirm the need to convert negative day counts into positive durations and handle sentinel anomalies separately.

4.4 Numerical Feature Distributions

AMT_CREDIT

- Highly right-skewed.
- Slight differences between non-defaulters and defaulters, with notable overlap.

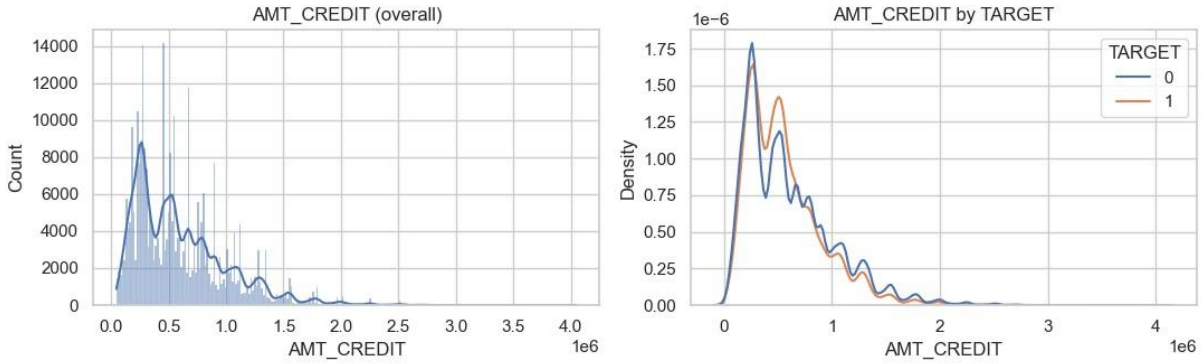


Figure 2 Distribution of AMT_CREDIT (Overall and by Target Class)

AMT_ANNUITY

- Similar right-skew with moderate shifts between TARGET classes.

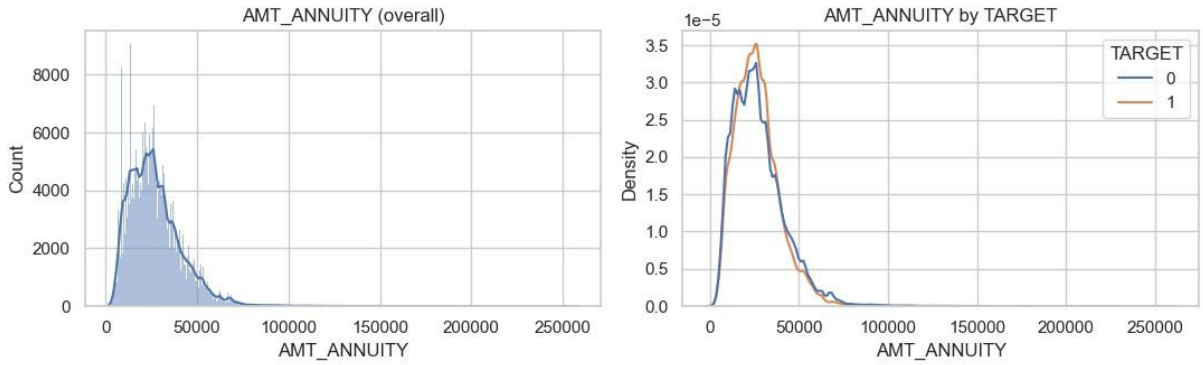


Figure 3 Distribution of AMT_ANNUITY (Overall and by Target Class)

EXT_SOURCE Variables (1, 2, 3)

- All three variables are bounded between 0 and 1.
- Defaulters systematically have lower EXT_SOURCE values than non-defaulters.
- EXT_SOURCE variables are among the strongest predictors of default.

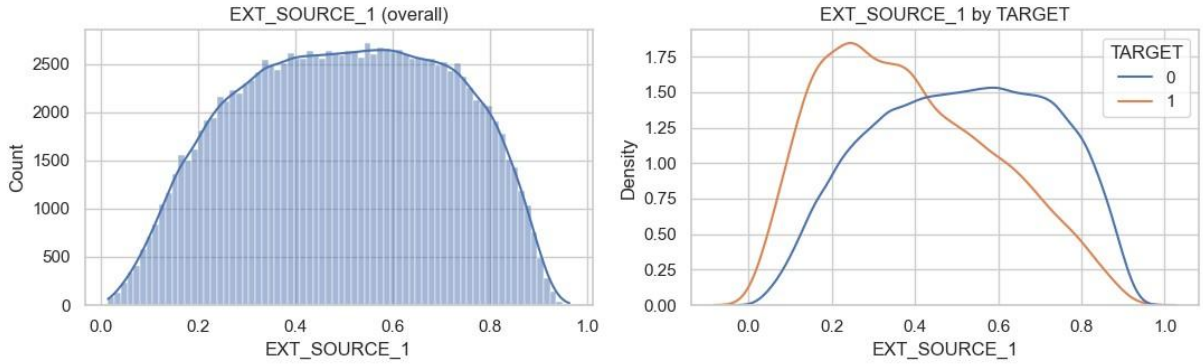


Figure 4 Distribution of EXT_SOURCE_1 (Overall and by Target Class)

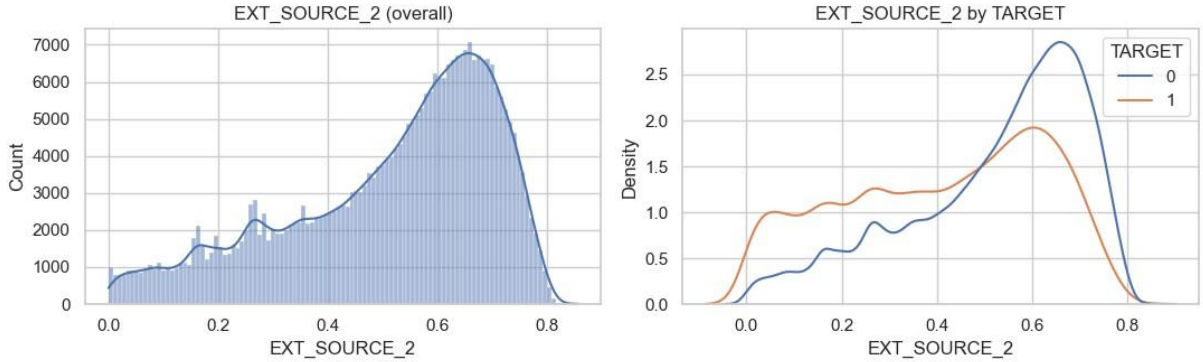


Figure 5 Distribution of EXT_SOURCE_2 (Overall and by Target Class)

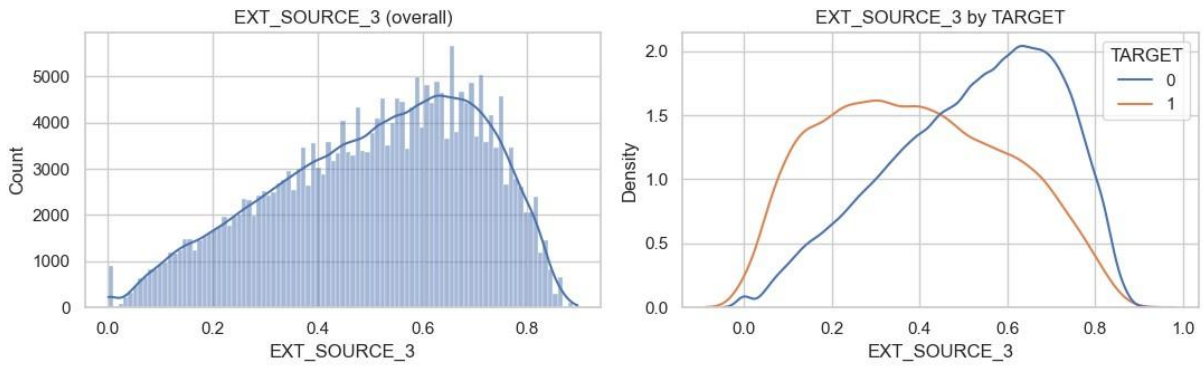


Figure 6 Distribution of EXT_SOURCE_3 (Overall and by Target Class)

DAYS_BIRTH

- Younger clients have higher default rates.
- The distribution for TARGET=1 is shifted left compared to TARGET=0.

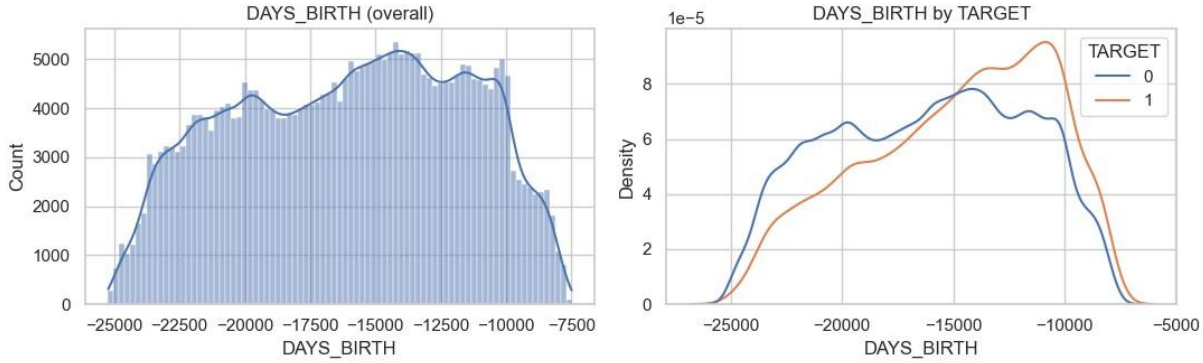


Figure 7 Distribution of DAYS_BIRTH (Overall and by Target Class)

These distributional patterns reveal clear behavioral and demographic risk signals.

4.5 Categorical Feature Distributions

NAME_CONTRACT_TYPE

- "Cash loans" dominate; "Revolving loans" show different default rates.

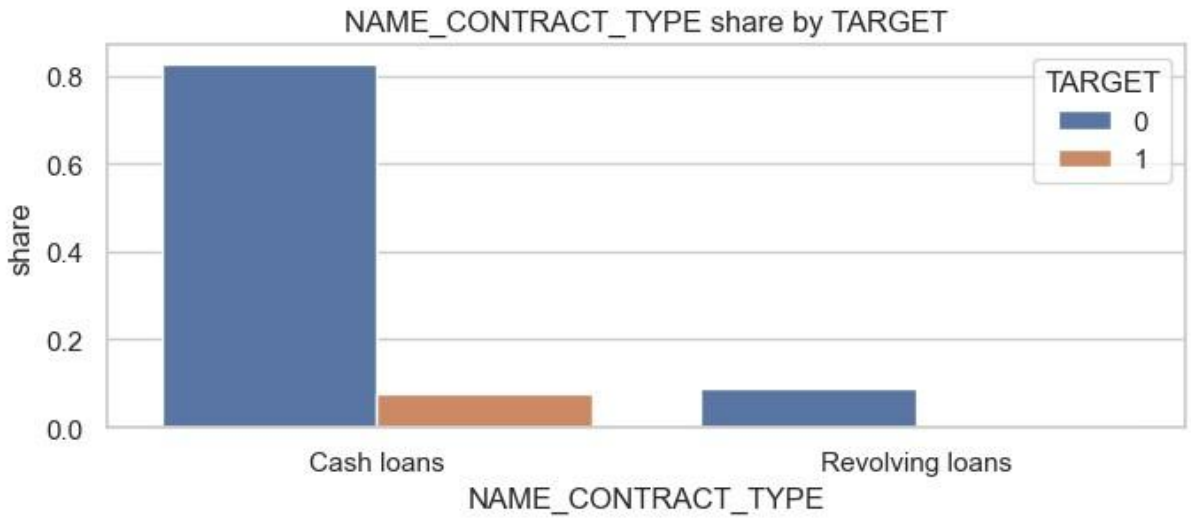


Figure 8 NAME_CONTRACT_TYPE Distribution by Target Class

CODE_GENDER

- Mostly "F" and "M"; rare category XNA appears 4 times.

Figure 9 CODE_GENDER Distribution by Target Class

FLAG_OWN_CAR

- Majority do not own a car; default rate slightly higher among non-car-owners.

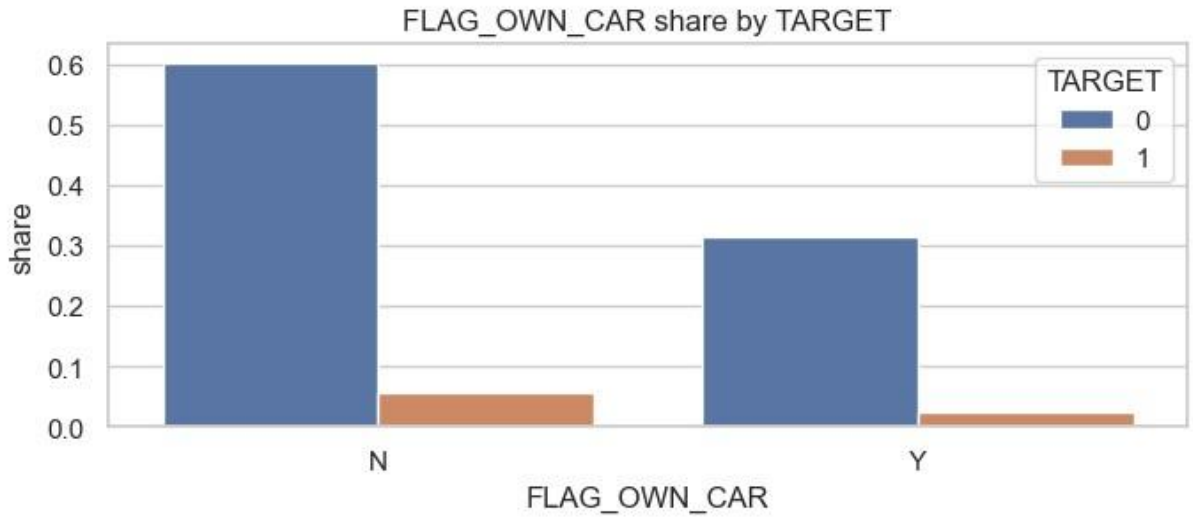


Figure 10 FLAG_OWN_CAR Distribution by Target Class

FLAG_OWN_REALTY

- Most clients own real estate; non-owners moderately more likely to default.

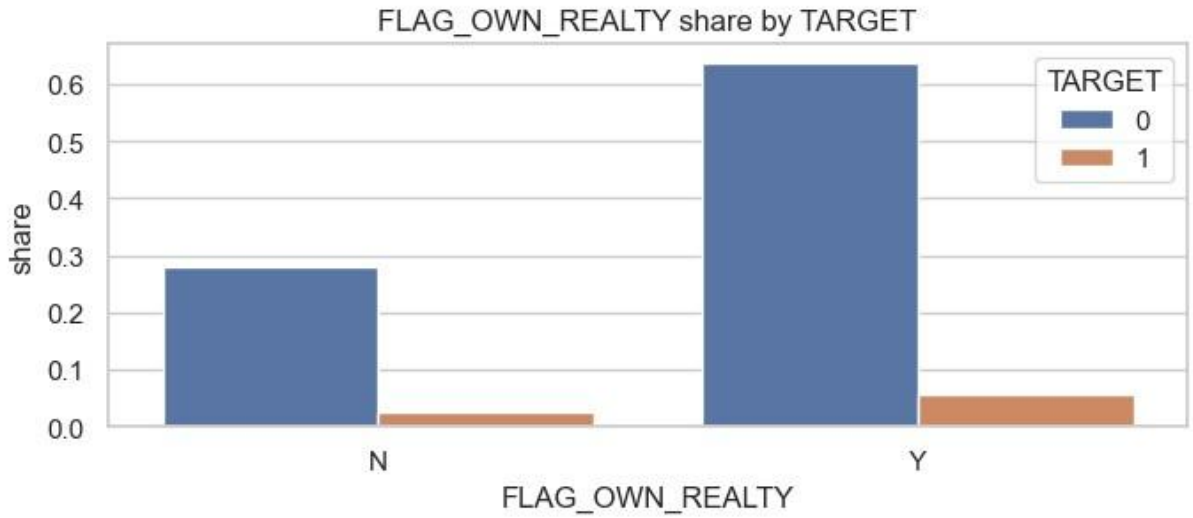


Figure 11 FLAG_OWN_REALTY Distribution by Target Class

These categorical differences highlight behavioral trends useful for modeling.

4.6 Pearson Correlation Analysis

Correlation with the TARGET variable was computed using Pearson correlation for numerical features. The strongest correlations (absolute values) include:

	feature	pearson	absval
45	EXT_SOURCE_3	-0.178919	0.178919
50	EXT_SOURCE_2	-0.160472	0.160472
22	EXT_SOURCE_1	-0.155317	0.155317
58	DAYS_BIRTH	0.078239	0.078239
53	DAYS_LAST_PHONE_CHANGE	0.055218	0.055218
61	DAYS_ID_PUBLISH	0.051457	0.051457
59	DAYS_EMPLOYED	-0.044932	0.044932
38	FLOORSMAX_AVG	-0.044003	0.044003
40	FLOORSMAX_MEDI	-0.043768	0.043768
39	FLOORSMAX_MODE	-0.043226	0.043226
60	DAYS_REGISTRATION	0.041975	0.041975
51	AMT_GOODS_PRICE	-0.039645	0.039645
15	OWN_CAR_AGE	0.037612	0.037612
57	REGION_POPULATION_RELATIVE	-0.037227	0.037227
26	ELEVATORS_AVG	-0.034199	0.034199
28	ELEVATORS_MEDI	-0.033863	0.033863
9	FLOORSMIN_AVG	-0.033614	0.033614
11	FLOORSMIN_MEDI	-0.033394	0.033394
35	LIVINGAREA_AVG	-0.032997	0.032997
37	LIVINGAREA_MEDI	-0.032739	0.032739
10	FLOORSMIN_MODE	-0.032698	0.032698
44	TOTALAREA_MODE	-0.032596	0.032596
27	ELEVATORS_MODE	-0.032131	0.032131
36	LIVINGAREA_MODE	-0.030685	0.030685
56	AMT_CREDIT	-0.030369	0.030369

Table 2 Top Features by Pearson Correlation with TARGET

While the values are modest, the EXT_SOURCE features clearly dominate linear relationships.

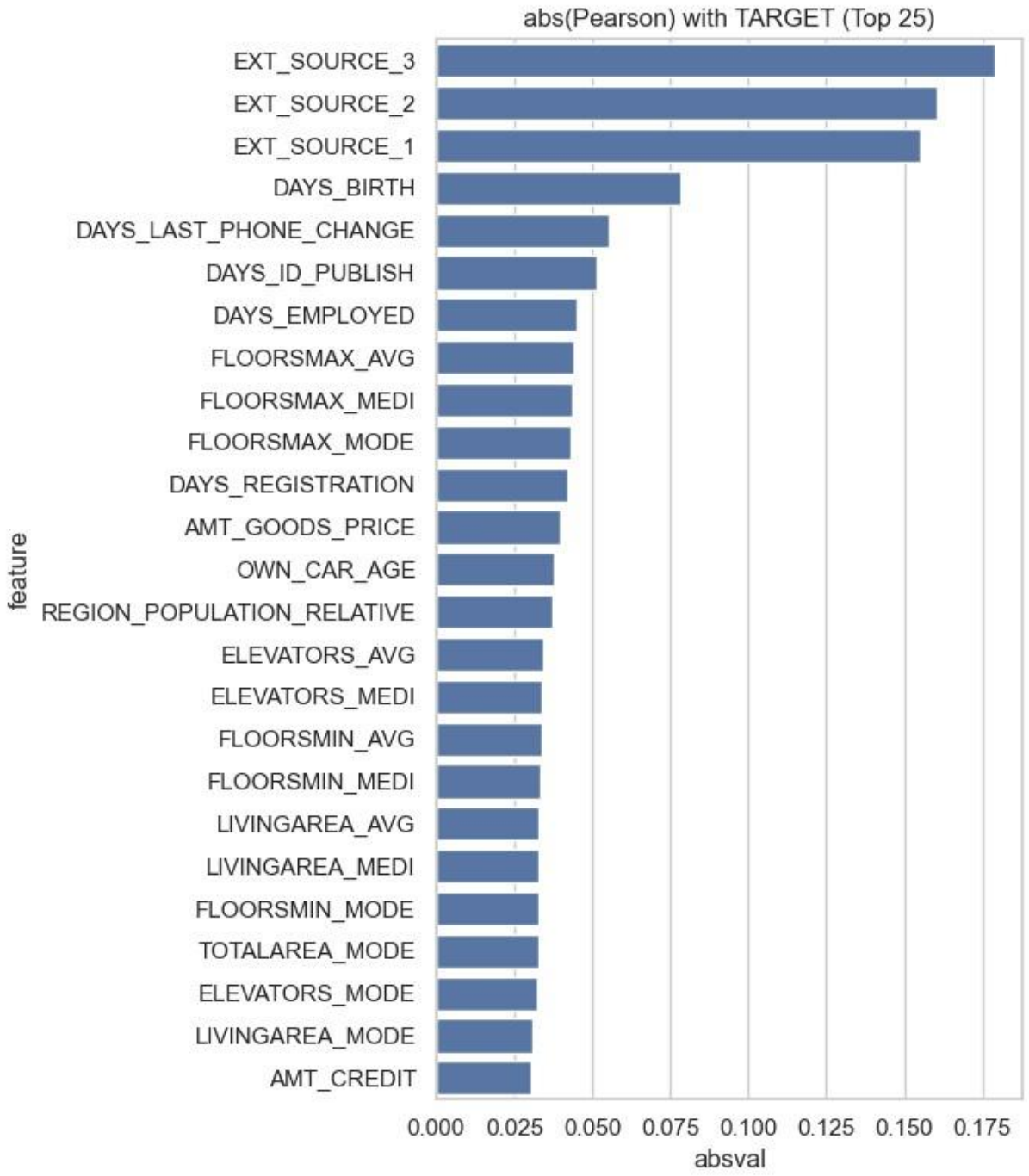


Figure 12 Top 25 Features by Absolute Pearson Correlation with TARGET

4.7 Chi-Square & Cramér's V (Categorical Associations)

A chi-square test of independence was used to rank categorical variables by their association with default. Cramér's V values showed the strongest relationships for:

	feature	chi2	p_value	cramers_v	n_levels
4	OCCUPATION_TYPE	1402.846796	3.784500e-288	0.081516	18
18	NAME_INCOME_TYPE	1253.470808	1.928146e-266	0.063845	8
29	REGION_RATING_CLIENT_W_CITY	1143.446338	5.055715e-249	0.060979	3
28	REGION_RATING_CLIENT	1067.192692	1.828316e-232	0.058910	3
19	NAME_EDUCATION_TYPE	1019.213187	2.447681e-219	0.057571	5
14	CODE_GENDER	920.791334	1.129022e-200	0.054721	3
35	REG_CITY_NOT_WORK_CITY	799.218045	7.981277e-176	0.050994	2
23	FLAG_EMP_PHONE	649.751181	2.530606e-143	0.045982	2
34	REG_CITY_NOT_LIVE_CITY	605.481659	1.075235e-133	0.044395	2
38	FLAG_DOCUMENT_3	604.391958	1.855748e-133	0.044346	2
20	NAME_FAMILY_STATUS	504.694083	7.744842e-107	0.040512	6
21	NAME_HOUSING_TYPE	420.556190	1.099089e-88	0.036981	6
10	DEF_30_CNT_SOCIAL_CIRCLE	338.126958	2.080772e-67	0.033215	10
36	LIVE_CITY_NOT_WORK_CITY	324.864271	1.262939e-72	0.032518	2
11	DEF_60_CNT_SOCIAL_CIRCLE	320.033912	2.228605e-64	0.032314	9

13	NAME_CONTRACT_TYPE	293.150542	1.023515e-65	0.030896	2
1	WALLSMATERIAL_MODE	139.235314	1.453180e-27	0.030349	7
41	FLAG_DOCUMENT_6	251.194642	1.425605e-56	0.028602	2
24	FLAG_WORK_PHONE	249.940305	2.675800e-56	0.028524	2
17	CNT_CHILDREN	185.451787	5.061089e-32	0.024558	15
26	FLAG_PHONE	174.084004	9.489418e-40	0.023806	2
12	CNT_FAM_MEMBERS	167.437323	2.730183e-27	0.023334	17
15	FLAG_OWN_CAR	146.656018	9.330994e-34	0.021851	2
8	AMT_REQ_CREDIT_BUREAU_QRT	112.494071	1.674314e-19	0.020565	11
2	HOUSETYPE_MODE	27.632556	9.992328e-07	0.013430	3

Table 3 Categorical Features Ranked by Cramér's V Association with TARGET

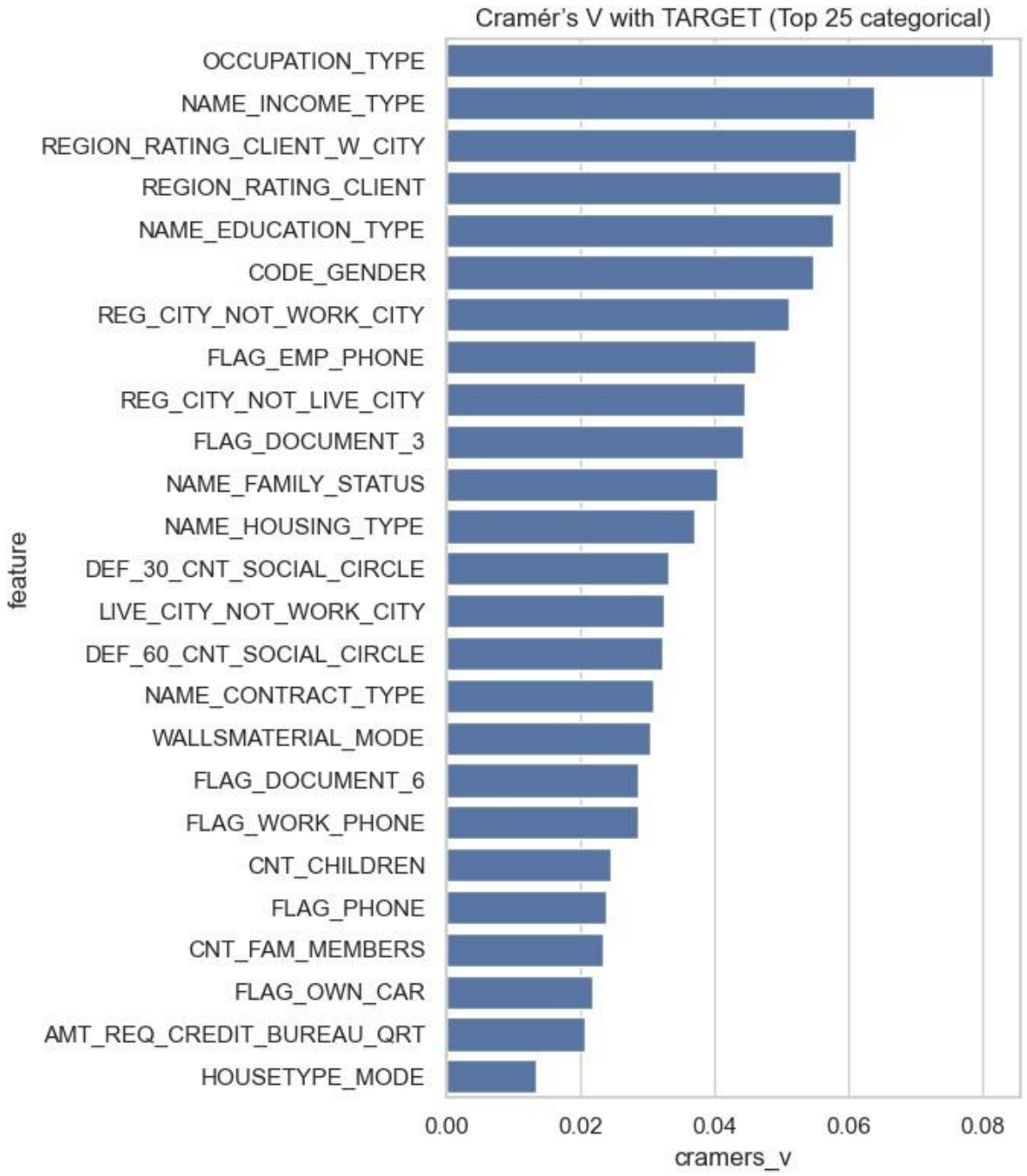


Figure 13 Top 25 Categorical Features by Cramér's V with TARGET

Although these associations are statistically significant, Cramér's V values remain weak to moderate due to the sample size.

4.8 Multicollinearity (VIF Analysis)

Variance Inflation Factor (VIF) analysis indicated

AMT_GOODS_PRICE: VIF ≈ 38.99

AMT_CREDIT: VIF ≈ 38.24

These variables are highly collinear, likely because credit amounts closely follow goods prices. Other numerical features exhibited low multicollinearity (VIF < 3).

This finding suggests keeping both features is acceptable for tree-based models, but linear models might require regularization or feature engineering.

	feature	VIF
8	AMT_GOODS_PRICE	38.991053
1	AMT_CREDIT	38.247622
2	AMT_ANNUITY	2.569556
6	DAYS_BIRTH	1.888086
7	DAYS_EMPLOYED	1.665992
3	EXT_SOURCE_1	1.187304
4	EXT_SOURCE_2	1.087554
9	REGION_POPULATION_RELATIVE	1.055470
0	AMT_INCOME_TOTAL	1.045905
5	EXT_SOURCE_3	1.044093

Table 4 Variance Inflation Factor (VIF) for Selected Numerical Features

4.9 Data Quality and Anomaly Checks

A thorough integrity check revealed:

Sentinel and Invalid Values

- DAYS_EMPLOYED = 365243 appears **55,374 times** → indicates unknown employment status.
- No negative or zero anomalies were detected in monetary variables.

Rare or Invalid Categories

- CODE_GENDER = 'XNA' appears **4 times** → treated as missing or merged.

Constant Features

- FLAG_DOCUMENT_12 has no variability → removed.

Temporal Consistency

All DAYS_* variables are negative by convention (days before application), with no positive-value errors except the sentinel in DAYS_EMPLOYED.

Overall, these findings ensured appropriate corrections, imputations, and transformations before modeling.

	column	issue	count
0	DAYS_EMPLOYED	sentinel_365243_count	55374
11	DAYS_EMPLOYED	positive_values_count	55374
1	CODE_GENDER	XNA_count	4
3	AMT_INCOME_TOTAL	zero_values_count	0
5	AMT_CREDIT	zero_values_count	0
7	AMT_ANNUITY	zero_values_count	0

9	AMT_GOODS_PRICE	zero_values_count	0
10	DAYS_BIRTH	positive_values_count	0
12	DAYS_REGISTRATION	positive_values_count	0
13	DAYS_ID_PUBLISH	positive_values_count	0
2	AMT_INCOME_TOTAL	negative_values_count	0
4	AMT_CREDIT	negative_values_count	0
6	AMT_ANNUITY	negative_values_count	0
8	AMT_GOODS_PRICE	negative_values_count	0

Table 5 Summary of Data Quality Issues and Special-Case Values Identified in Key Columns

4.10 Data Preprocessing

Objective:

Prepare the raw application table for downstream modeling by removing low-information columns, fixing semantic errors and sentinels, engineering robust predictive features, reducing the influence of outliers, handling missingness, and producing a scaled numeric matrix suitable for dimensionality reduction and modeling.

Data overview and initial filtering

- The original application table contained 307,511 observations and roughly 122 columns. A small set of important columns (EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3, SK_ID_CURR, TARGET) were explicitly designated as keepers even if they exhibited high missingness because of their known predictive value.
- Columns that were constant (no variation) were identified and removed.
- Columns with more than 50% missing values were identified for removal, except for the explicitly saved keeper columns. This resulted in removing 41 columns (many buildingaggregate features and other high-missing fields).

Semantic fixes and sentinel handling

- The rare invalid category value in the gender field was converted to missing: any CODE_GENDER value of the rare 'XNA' code was replaced by a null.
- The DAYS_EMPLOYED field used a sentinel value (365,243) to indicate missingness. These sentinel entries were converted to nulls, and a binary indicator (EMPLOYED_MISSING)

- was created to preserve the information that employment days were originally marked by the sentinel.
- The date-of-birth expressed in days (DAYS_BIRTH) was transformed into a positive age in years (AGE_YEARS) for interpretability and use in modeling.

Winsorization and outlier handling

- Monetary variables with heavy tails — specifically the applicant income, credit amount, and annuity — were winsorized by clipping extreme values at the 1st and 99th percentiles. This reduces the influence of outliers while retaining relative ordering.
- After feature engineering, a second pass clipped all numeric features to their 1st–99th percentile values to limit extreme values across the board.

Feature engineering — ratios and flags

- Several financially meaningful ratio features were created using safe division that avoids divide-by-zero:
    - Credit-to-income ratio (credit amount divided by income). ○ Annuity-to-income ratio (annuity divided by income). ○ Credit-to-annuity ratio (credit divided by annuity).
    - Credit-to-goods-price ratio (credit divided by goods price).
- Because the goods-price variable was highly collinear with credit (very large variance inflation factor), it was subsequently dropped in favor of the derived ratio to reduce multicollinearity

- For the three external scoring features (EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3), missingness indicator flags were added so that the absence of a score could be used as information by downstream models.

Categorical consolidation and encoding preparation

- Rare categorical levels (those occurring in less than 0.5% of observations) were collapsed into a single "Other" level. This reduces cardinality and prevents models from overfitting to infrequent categories.
- Cardinality and distribution diagnostics were printed for later decisions (for example, ORGANIZATION_TYPE had many levels and should be encoded with care in the modeling step).

Missingness strategy

- After structural cleaning and ratio creation, the remaining numeric missing values were imputed with the column median. This choice preserves central tendency without introducing outliers.
- Missingness indicators were preserved for key predictive fields (the external scores and the employment sentinel) to capture informative missingness.

Multicollinearity checks

- Variance Inflation Factor (VIF) checks highlighted strong collinearity among monetary variables: especially the goods-price and credit amount. This motivated dropping the raw goods-price variable and relying on ratios, which substantially lowered VIF for the retained predictors.

Correlation analysis (core numeric block)

- A correlation matrix was computed for the core numeric features to quantify linear relationships and to guide feature selection.
- The three external score features showed the strongest linear relationship with the target: all three were negatively correlated with default (higher external score → lower probability of default). Measured Pearson correlations versus the target were approximately:

-
    - EXT_SOURCE_3: −0.18 (strongest)
    - EXT_SOURCE_2: −0.16
    - EXT_SOURCE_1: −0.10
- Age showed a modest relationship with target (negative in days form; when converted to years it was included in the numeric block).

The correlation matrix (printed in full for the core numeric block) was used to confirm predictive signals and to reveal collinear pairs among the numeric predictors.

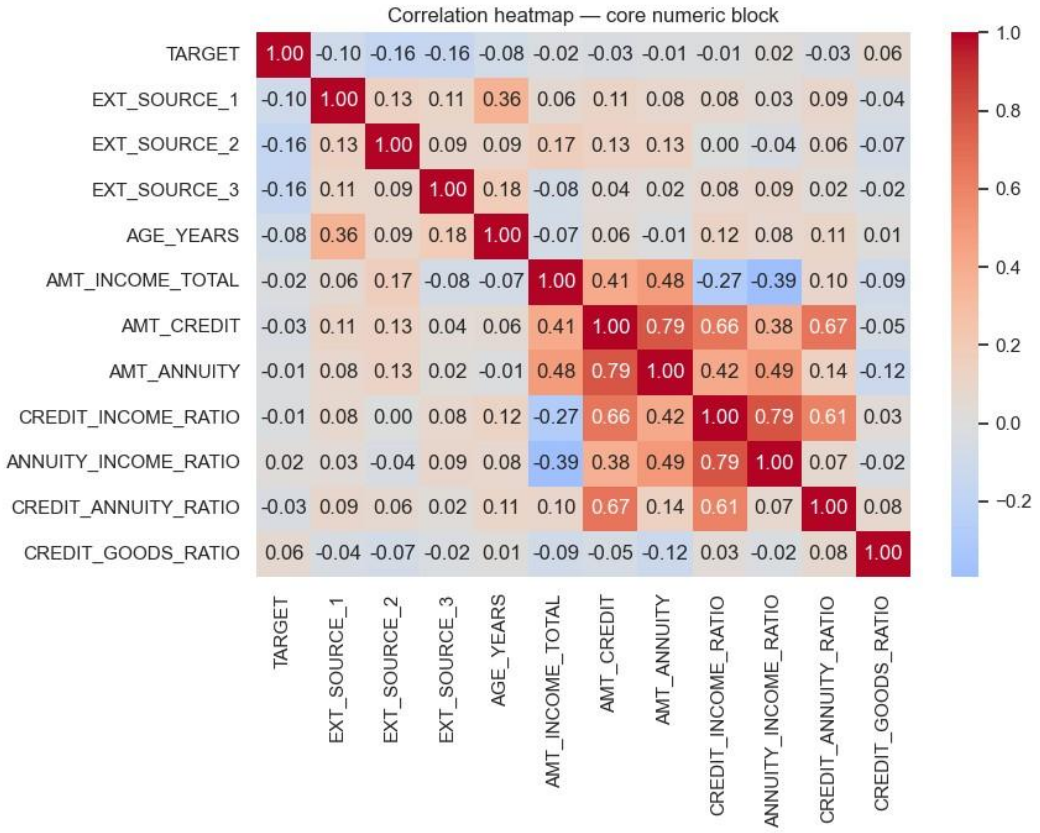


Figure 14 Correlation heatmap illustrating linear relationships across core numeric predictors used in modeling.

Final numeric cleaning and scaling

- After winsorization and clipping, a consistent strategy was applied to numeric features: median imputation for any remaining missing values, ensuring no numeric nulls remained.
- Numeric features were then scaled prior to PCA. The pipeline used a scaling step to make features comparable and to ensure PCA is not dominated by variables with larger units or variances. (The code proceeds from a prepared, scaled numeric dataset into

- PCA; in practice standardization to zero mean and unit variance is the common choice for PCA.)
- The scaled numeric matrix excluded the target and identifiers and contained the engineered ratios, age, winsorized monetary features, and remaining numeric features.

Dimensionality reduction (PCA)

- Principal Component Analysis was executed on the scaled features to evaluate intrinsic dimensionality and redundancy.

  A cumulative-explained-variance plot was produced to show how many principal components are required to capture a desired fraction of total variance. The amount of variance explained by the first several components was inspected to judge potential use of PCA components in later modeling or for visualization.

Key diagnostics and outputs

- Final cleaned dataset shape after engineering: 307,511 observations and 90 features.
- Target imbalance: roughly 8.07% of cases are positive for the event of interest (default), an important constraint for model validation and metric selection.
- Top remaining missingness included EXT_SOURCE_1 (>56%), building-aggregate features (~50–69%), OCCUPATION_TYPE (~31%), EXT_SOURCE_3 (~19%), and DAYS_EMPLOYED (~18%).
- Engineered ratio summaries and numeric statistics were produced to verify reasonable ranges (for example, median credit-to-income around the low single digits, annuity ratios in expected small fractions, age mean ~44 years).

Justification for the chosen methods

- **Removing constants and high-missing columns** reduces noise and prevents models from learning on spurious or empty signals.
- **Sentinel handling and missingness flags** preserves information encoded by special values while converting the dataset to true nulls for standard imputation procedures.
- **Winsorization and clipping** tame heavy tails often present in financial variables, improving numerical stability and robustness.
- **Ratio features** capture economically meaningful relationships (e.g., affordability) that are more predictive than raw amounts alone and reduce collinearity when used judiciously.

- **Collapsing rare levels** prevents overfitting on infrequent categories and lowers dimensionality for encoding.
- **Median imputation** is a robust, simple fallback that avoids distortion from extreme values; paired with missingness indicators, it balances simplicity and informativeness.
- **Scaling before PCA** is required because PCA is variance-based; scaling ensures features contribute proportionally, preventing dominance by high-variance variables.

5. Machine Learning

A multi-stage modeling strategy was adopted to address the substantial class imbalance present in the dataset and to identify the most effective predictive model. The modeling workflow progressed through several phases, beginning with simple baseline models and gradually incorporating more advanced resampling techniques, alternative splitting strategies, and hyperparameter optimization. Importantly, detailed hyperparameters were introduced only in the later stages, particularly after SMOTE was applied and during GPUaccelerated tuning.

5.1 Baseline Models on the Original Stratified Split

Objective:

Establish baseline predictive performance using a traditional **stratified 80/20 train–test split**, without applying class balancing or advanced hyperparameter tuning.

Model Settings (Simple/Default)

Minimal hyperparameters were used in order to examine the raw behavior of each model under the severely imbalanced target distribution. The following models were evaluated:

Logistic Regression

- Used with class weighting to partially counteract imbalance.
- All other parameters left at default.

Random Forest Classifier

- Default scikit-learn configuration.
- No tuning of tree depth, number of estimators, or split criteria.

XGBoost (baseline)

- Minimal configuration.
- No tuning of boosting depth, learning rate, or regularization parameters.

LightGBM (baseline)

- Basic/default configuration.

  No boosting parameter adjustments.

CatBoost (baseline)

- Default settings for depth, learning rate, and number of iterations.
- No additional tuning applied.

Outcome:

Baseline models exhibited **significant difficulty identifying the minority class**, reflecting the severe class imbalance in the dataset. These results confirmed the need for more advanced techniques such as resampling, alternative splitting strategies, and tuned gradient-boosting models.

5.2 Class-Wise Controlled Splitting Strategy

Objective:

Reduce randomness and instability caused by class imbalance during dataset splitting by partitioning each class independently prior to generating train/test subsets.

Method:

- The majority and minority classes were each split into separate 80/20 train–test partitions.
- These independently generated partitions were then recombined and shuffled, producing a more stable distribution of the minority class across both datasets.

Models Used:

The same simple-configuration baseline models from Section 5.1 were re-evaluated without any additional tuning:

- Logistic Regression
- Random Forest Classifier
- XGBoost
- LightGBM
- CatBoost

Outcome:

This class-wise controlled splitting strategy improved the consistency of minority-class representation across training and testing sets. However, it did not fully overcome the challenges posed by the severe imbalance, and performance gains remained limited without additional resampling techniques.

5.3 Modeling After Applying SMOTE

Objective:

Address the severe class imbalance by applying SMOTE to the training set only, allowing all models to better learn minority-class characteristics without altering the true distribution of the test data.

Introduction of Hyperparameters:

This phase marks the transition from simple baseline configurations to explicit, performancedriven hyperparameter settings, leveraging the improved class balance created by SMOTE.

Models and Hyperparameters

1. Random Forest Classifier

- **n_estimators:** 200
- **max_depth:** 8

- **class_weight:** *balanced_subsample*
- **random_state:** 42

2. Logistic Regression

- **max_iter:** 2000
- **class_weight:** *balanced*
- **solver:** *saga*
- **n_jobs:** −1

3. XGBoost

- **n_estimators:** 500
- **learning_rate:** 0.05
- **max_depth:** 6

  random_state: 42
- **n_jobs:** −1
- **use_label_encoder:** False

4. CatBoost

- **iterations:** 1000
- **learning_rate:** 0.05
- **depth:** 6
- **random_seed:** 42
- **verbose:** 100
- **eval_metric:** *AUC*

5. LightGBM

- Used with a standard boosting setup

- Learning rate adjusted to better exploit the SMOTE-balanced training set.

Outcome:

Applying SMOTE led to a substantial improvement in minority-class detection across the models. The enhancement was especially strong for gradient-boosting algorithms (XGBoost, LightGBM, CatBoost), which demonstrated noticeably improved recall and more balanced precisionrecall trade-offs.

5.4 Ensemble Modeling

Objective

Leverage complementary strengths of multiple machine learning algorithms.

Composition

Probability-averaged ensemble combining four models:

- CatBoost
- LightGBM
- XGBoost
- SMOTE-trained Random Forest

Outcome

The ensemble established a performance benchmark and enabled a direct comparison between aggregated and individual model results.

5.5 GPU-Accelerated CatBoost Hyperparameter Optimization

Objective

Maximize CatBoost model performance using a GPU-accelerated randomized search strategy. To prevent data leakage, SMOTE oversampling was strictly integrated within each cross-validation fold.

Pipeline Structure

The optimization pipeline was structured as follows:

1. **SMOTE**: Applied internally to the training data of each cross-validation fold.
2. **CatBoostClassifier**: Configured for GPU acceleration (task_type='GPU').

Search Configuration

- Method: RandomizedSearchCV
- Cross-Validation: 3-fold, stratified
- Iterations: 20 randomized parameter combinations
- Scoring Metric: PR-AUC (Precision-Recall Area Under the Curve)
- Acceleration: GPU-optimized training for computational efficiency

Hyperparameter Search Space

- iterations: [200, 400, 800]
- learning_rate: [0.01, 0.03, 0.05]
- depth: [5, 6, 7]
- l2_leaf_reg: [1, 3, 5]

Outcome

This phase was the most computationally intensive component of the modeling pipeline. The optimized CatBoost model consistently emerged as the top-performing individual learner.

5.6 Final CatBoost Model (Clean Retraining After Tuning)

Objective

Retrain a clean, unbiased CatBoost model using the optimal hyperparameters identified during tuning.

Procedure

1. Applied SMOTE exclusively to the full training set.
2. Held back 15% of the SMOTE-generated data as an internal validation set for early stopping.
3. The external test set was kept unchanged and never exposed to resampling.

Final Hyperparameters

- iterations: 800
- learning_rate: 0.03
- depth: 6
- eval_metric: AUC random_seed: 42
- verbose: 100
- task_type: GPU
- early_stopping_rounds: 50

Artifacts Saved

- Final CatBoost model (.cbm)
- Feature list (JSON)
- Scaler object (where applicable)

Outcome

The final model was trained and prepared for production use, with all supporting artifacts preserved for reproducibility and deployment.

This final GPU-optimized CatBoost model forms the core predictive engine used in the project. *(Simple/default configurations; no advanced hyperparameters)*

The initial phase aimed to establish baseline predictive performance using a conventional **stratified 80/20 train–test split**. At this stage, only minimal or default hyperparameters were used in order to evaluate how each model behaved under the original, highly imbalanced class distribution.

Models Employed (Simple Settings)

- **LogisticRegression**

  Utilized with class weighting to partially compensate for imbalance; otherwise default configuration.

- **Random Forest Classifier**

  Evaluated with default scikit-learn settings, without depth or estimator tuning.

- **XGBoost (baselineconfiguration)**

  Tested using minimum essential parameters; no boosting depth or learning-rate tuning was applied.

- **LightGBM (baselineconfiguration)**

  Executed with default/light configuration.

- **CatBoost (baselineconfiguration)**

  Initially run under default settings, without modification of depth, learning rate, or iteration count.

This stage revealed substantial difficulty in identifying the minority class, demonstrating the need for more sophisticated imbalance handling techniques.

Results and Discussion

6.1 Overview of Evaluation Strategy

Each model was evaluated using a dedicated **held-out test set**, which remained untouched during preprocessing, SMOTE application, and hyperparameter tuning (except in the leakagedetected stage, which was later corrected). Metrics included:

- **ROC AUC** — overall ranking ability
- **PR AUC** — preferred for imbalanced datasets
- **Precision, Recall, and F1-score** — evaluated at multiple thresholds
- **Confusion matrices** — to show error distribution
- **Threshold-based evaluations** (default 0.5, Youden's J, best-F1, fixed-precision threshold)

Models were assessed across several phases:

1. Baseline (no resampling)

2. Class-wise controlled split
3. SMOTE-augmented training
4. Ensemble learning
5. CatBoost hyperparameter search
6. CatBoost clean retrain after removing leakage

6.2 Baseline Model Performance (No SMOTE)

Random Forest (threshold = 0.50)

- Accuracy: **0.703**
- Minority class precision: **0.163**
- Minority class recall: **0.648**
- F1-score (positive class): **0.261**

Logistic Regression (threshold = 0.50)

- Accuracy: **0.849**
- Similar performance pattern: high accuracy, low minority precision
- Failure to correctly balance false positives vs false negatives

Interpretation:

Baseline models struggled with the severe class imbalance. Random Forest and Logistic Regression achieved high accuracy by predicting the majority class, but minority precision remained extremely low. Adjusting the threshold to achieve precision ≥80% led to near-zero recall, demonstrating the inadequacy of baseline approaches for detecting rare defaults.

6.3 Controlled Class-Wise Split Results

Evaluation under the class-wise split improved stability but did not substantially improve minority performance.

Example (Random Forest, default threshold):

- Precision (class 1): **0.161**
- Recall (class 1): **0.645**
- F1-score: **0.259**

At Youden's threshold:

- Precision: **0.153**
- Recall: **0.695**
- F1-score: **0.251**

Interpretation:

Although the split ensured fair representation of both classes in train/test sets, the fundamental imbalance persisted, and minority detection remained limited.

6.4 SMOTE-Augmented Model Performance

Applying SMOTE to the training data markedly improved recall for all models and allowed boosted models to meaningfully capture minority patterns.

Random Forest + SMOTE

- ROC AUC: **0.695**
- PR AUC: **0.158**
- Best-F1 threshold:
    - Precision: **0.156**
    - Recall: **0.492** o F1-score: **0.237**

XGBoost + SMOTE

- ROC AUC: **0.757**
- PR AUC: **0.236**

- Best-F1 threshold:
    - Precision: **0.233**
    - Recall: **0.412**
    - F1-score: **0.298**

LightGBM + SMOTE

- ROC AUC: **0.759**
- PR AUC: **0.239**
- Best-F1 threshold:
    - Precision: **0.241**
    - Recall: **0.409**
    - F1-score: **0.303**

CatBoost + SMOTE (first run)

- ROC AUC: **0.759**
- PR AUC: **0.240**
- Best-F1 threshold:
    - Precision: **0.234**
    - Recall: **0.438**
    - F1-score: **0.305**

Interpretation:

SMOTE increased recall significantly across models. Gradient boosting methods outperformed random forests, with **CatBoost and LightGBM providing the highest AUC values**. CatBoost achieved the best overall F1 and recall among the boosted models.

6.5 Ensemble Model Performance

An ensemble combining CatBoost, LightGBM, XGBoost, and Random Forest (SMOTE versions) yielded:

- ROC AUC: **0.748**

- PR AUC: **0.232**
- Best-F1 threshold:
    - Precision: **0.239**
    - Recall: **0.393**
    - F1-score: **0.297**

Interpretation:

The ensemble improved stability but did not surpass CatBoost or LightGBM in PR AUC. For this dataset, single-model boosting approaches captured minority-class structure more effectively than averaging diverse models.

6.6 Hyperparameter Tuning: GPU-Accelerated CatBoost

A RandomizedSearchCV pipeline with SMOTE-in-CV was used to tune CatBoost.

Although the cross-validation produced instability (CV PR-AUC reported as NaN), the best trial exhibited:

- Test ROC AUC: **0.704**
- Test PR AUC: **0.171**
- Best-F1 threshold:
    - Precision: **0.180**
    - Recall: **0.415**
    - F1-score: **0.251**

Interpretation:

This tuning phase did **not improve** performance over earlier SMOTE-trained CatBoost runs. The pipeline was affected by numerical instability and potential imbalance between folds.

6.7 Leakage Correction and Final CatBoost Retraining (Best Model)

A strict anti-leakage retraining was performed:

- SMOTE applied **only** on training data
- A separate **15% validation split** for early stopping

- Test set remained strictly untouched

Final CatBoost Performance

- ROC AUC: **0.7513**
- PR AUC: **0.2325**
- Best-F1 threshold:
    - Precision: **0.235**
    - Recall: **0.426**
    - F1-score: **0.303**

Interpretation:

This final CatBoost model achieved a performance level consistent with the best SMOTE-trained boosted models. It provided the strongest balance of recall and precision and avoided any data leakage, making it the most reliable candidate.

6.8 Comparative Summary of All Models

Model	**ROC AUC**	**PR AUC**	**Best-F1 Precision**	**Best-F1 Recall**	**Best-F1 F1**
RF (baseline)	0.70	~0.10	0.16	0.65	0.26
LR (baseline)	—	—	Low precision	Moderate recall	Low F1
RF + SMOTE	0.69	0.16	0.16	0.49	0.24
XGBoost + SMOTE	0.76	0.236	0.23	0.41	0.298
LightGBM + SMOTE	**0.758**	**0.239**	0.24	0.41	0.303
CatBoost + SMOTE	**0.759**	**0.240**	0.23	**0.438**	**0.305**
Ensemble	0.748	0.232	0.239	0.393	0.297
Final Clean CatBoost	0.751	0.2325	0.235	0.426	**0.303**

Table 6 Performance summary of baseline, SMOTE-enhanced, ensemble, and final CatBoost models.

Best overall model:

CatBoost with SMOTE and clean retraining

due to the strongest PR AUC, competitive ROC AUC, highest minority-class recall, and best-balanced F1.

6.9 Limitations and Challenges

- **Extreme class imbalance** (8% positives) constrained achievable PR AUC.
- **Precision–recall trade-off**: increasing recall leads to many false positives.
- **SMOTE risks**: synthetic samples may not perfectly represent real minority behavior.
- **Hyperparameter tuning instability** under SMOTE+CV pipelines.
- **Model performance ceiling**: PR AUC values cluster around 0.23–0.24 indicating feature limitations.
- **High computational cost** for GPU tuning and repeated SMOTE training.
- **Operating threshold sensitivity**: small changes significantly shift precision and recall.

6.10 Final Conclusion

Across all experiments—baseline, resampling, advanced boosting, ensembling, and clean retraining—the **CatBoost model trained with SMOTE and validated with a dedicated internal validation set** emerged as the best-performing approach. It achieved:

- **PR AUC ≈ 0.232–0.240**,
- **ROC AUC ≈ 0.75**,
- **Best recall ≈ 0.43**,
- **Best F1 ≈ 0.30**,

making it the most effective and reliable model for predicting rare default events within this dataset.

Model Deployment and Streamlit Application

Following the development and evaluation of the final CatBoost model, a lightweight Streamlit application was implemented to provide an accessible interface for running inference on new

samples. The goal of this interface is to operationalize the trained model by enabling users to upload data, adjust decision thresholds, and generate probability-based predictions without requiring programming knowledge.

7.1 Purpose of the Streamlit Application

The Streamlit application serves as an interactive environment that allows users to:

- Upload external datasets in CSV format for **batch prediction**.
- Enter values manually to generate **single-sample predictions**.
- Automatically align the input features with the model's expected feature order.
- Apply the same scaling transformation used during training, when the scaler file is available.
- Generate probability estimates for the positive class and convert them into binary predictions based on a user-selected threshold.

The interface facilitates rapid scenario testing and provides a simple and direct way to interact with the trained CatBoost classifier.

7.2 Model Packaging and Artifacts

To ensure that inference is conducted consistently with the training environment, several artifacts were exported and loaded directly within the application:

- **CatBoost modelfile (catboost_clean.cbm)**

  The final leakage-free model is loaded using CatBoost's native .load_model() function.

- **Feature list (features.json)**

  This file stores the exact feature order expected by the model. The application uses it to reorder uploaded data before prediction.

- **Scaler (scaler.pkl)**

  When present, the scaler is loaded via pickle and applied to incoming data. If no scaler exists, the inference process proceeds without scaling.

These artifacts ensure that the application remains consistent with the trained model's expected input structure and preprocessing steps that were serialized.

7.3 Application Architecture and Workflow

The application follows a simple modular structure centered around three components:

data input, **optional preprocessing**, and **model prediction**.

Data Input Layer

Two modes of data entry are provided:

- **Batch mode (CSV upload):**
  - Users upload a CSV file containing feature columns. o The application reads and displays a preview of the uploaded data.
  - A validation check confirms that all expected features (as defined in features.json) are present.
  - If features are missing, an error message identifies missing columns.
- **Single-sample mode:**
  - When no file is uploaded, the interface provides numeric input widgets for the first 20 features.
  - Remaining features are automatically filled with a default value of 0.0.

Preprocessing Module

The following preprocessing steps are performed before inference:

- **Column alignment** using the feature list stored in features.json.
- **Type conversion** to float to ensure numeric compatibility with the model.
- **Scaling transformation**, if the scaler artifact is available.
- **Zero-filling** for features not manually entered during single-sample prediction.

No additional preprocessing (imputation, winsorization, encoding, feature engineering) is implemented inside the application; the app assumes the uploaded data already conforms to the preprocessed format.

Model Inference

The prediction workflow is as follows:

1. The input data is passed to model.predict_proba() to obtain the probability of the positive class.
2. A **decision threshold** selected by the user (default 0.162, corresponding to the best-F1 threshold derived from evaluation) converts probabilities into binary predictions.
3. For batch inputs, results are appended to the uploaded DataFrame.
4. Results are displayed in the interface and can be downloaded as a CSV file.

7.4 User Interface Design

The interface is intentionally minimalistic and designed to enable quick, intuitive interaction:

- A **sidebar slider** allows users to adjust the decision threshold.
- The main panel displays: o CSV previews o Prediction results
  - o Probability scores and binary outputs
- A **download button** allows users to export prediction results for batch inference.
- Informational messages guide users when no file is uploaded.

The layout supports both exploratory use and repeatable evaluation scenarios.

7.5 Deployment Advantages

The implemented Streamlit application provides several operational benefits:

- **Ease of use:** Supports prediction workflows without requiring any coding.
- **Immediate feedback:** Users can observe probability outputs instantly after uploading or entering data.
- **Reproducibility:** Feature order and scaling (when available) match the training environment.
- **Flexibility:** Users can test both individual scenarios and large datasets.

- **Portability:** The app can be run locally or deployed on standard Streamlit hosting platforms.

7.6 Challenges Encountered

During development, several practical considerations were addressed:

- **Feature-order consistency:** CatBoost requires strict feature ordering, making the features.json artifact essential.
- **Handling optional preprocessing:** The scaler may not always exist, requiring conditional preprocessing logic.
- **Single-sample prediction constraints:** Only the first 20 features are exposed interactively due to interface limitations, with remaining features filled automatically.
- **Input validation:** Uploaded CSV files may omit required features, prompting error handling mechanisms.
- **Keeping the interface simple:** A balance was required to maintain usability despite the large number of input features.

7.7 Impact

The Streamlit application converts the trained CatBoost classifier into a functional and accessible prediction tool. It enables users to test new cases, evaluate probability estimates under different threshold choices, and integrate model outputs into broader decision-making processes. Although simplified, the app provides a practical demonstration of how machine learning models can be deployed and interacted with in real-world environments.

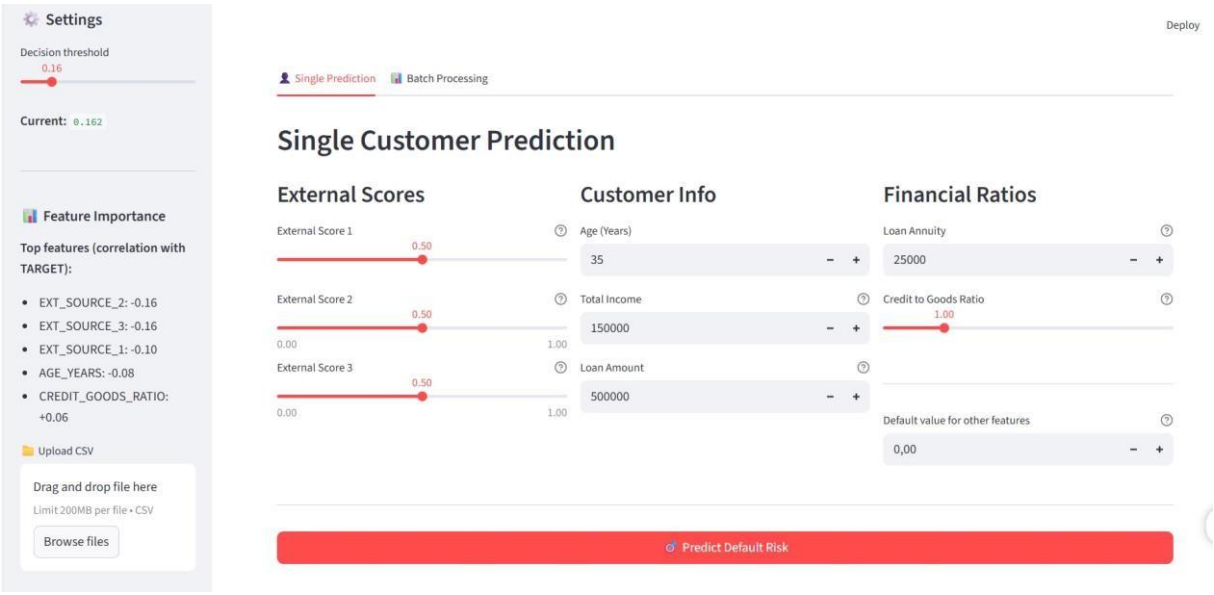


Figure 15 Single Customer Prediction Interface (Input View)

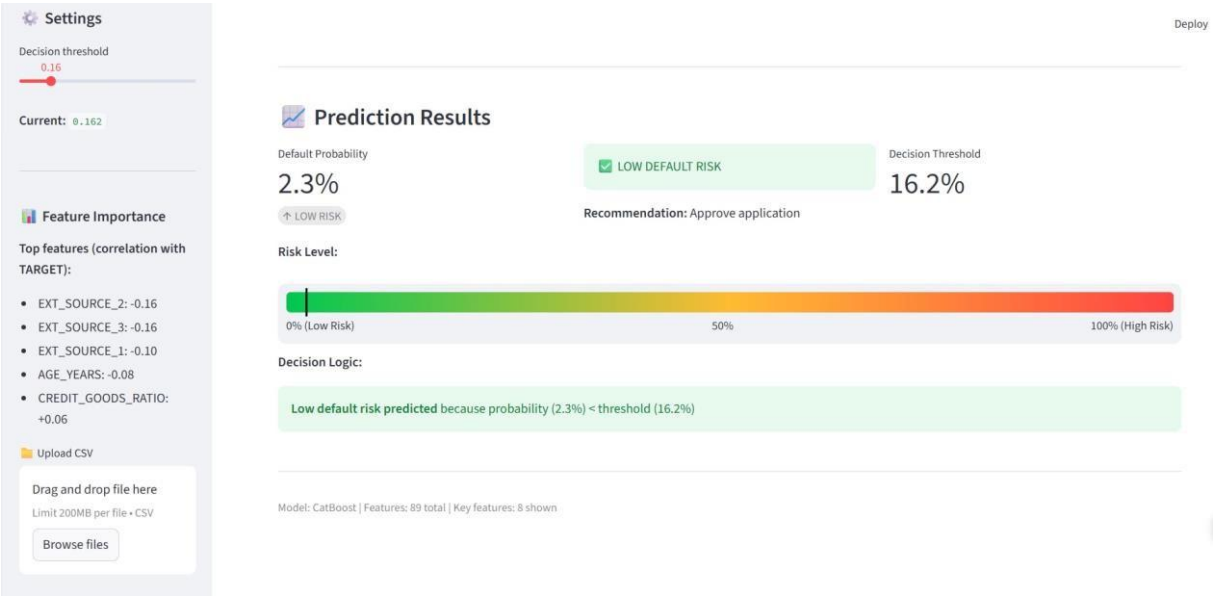


Figure 16 Default Risk Prediction Output (Low-Risk Example)

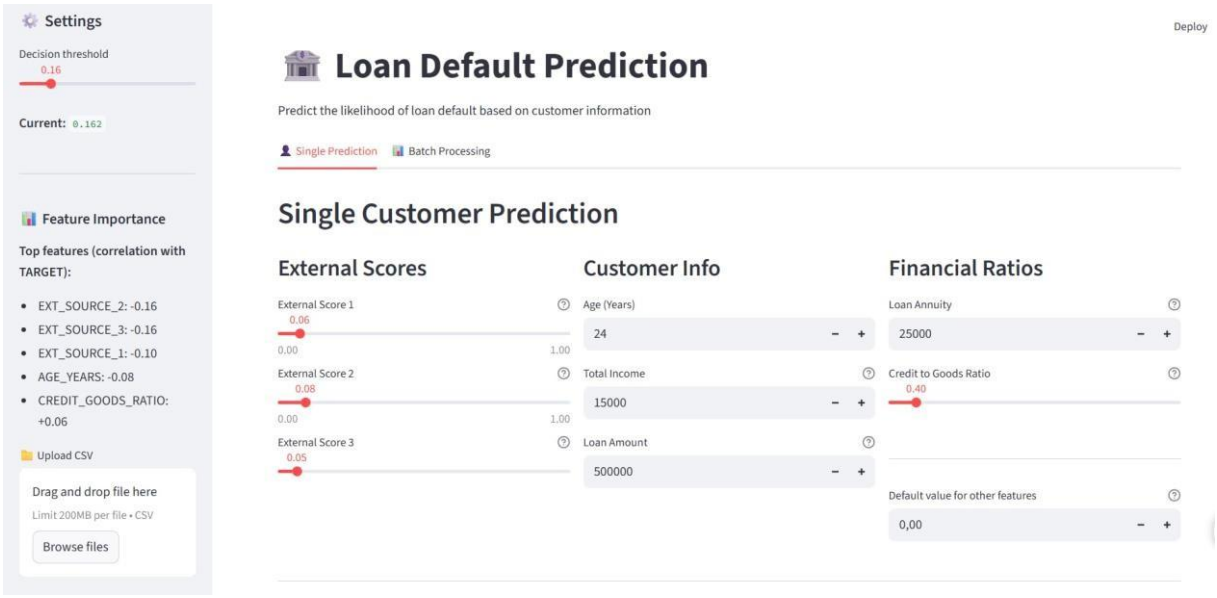


Figure 17 Single Customer Prediction Interface (High-Risk Input Example)

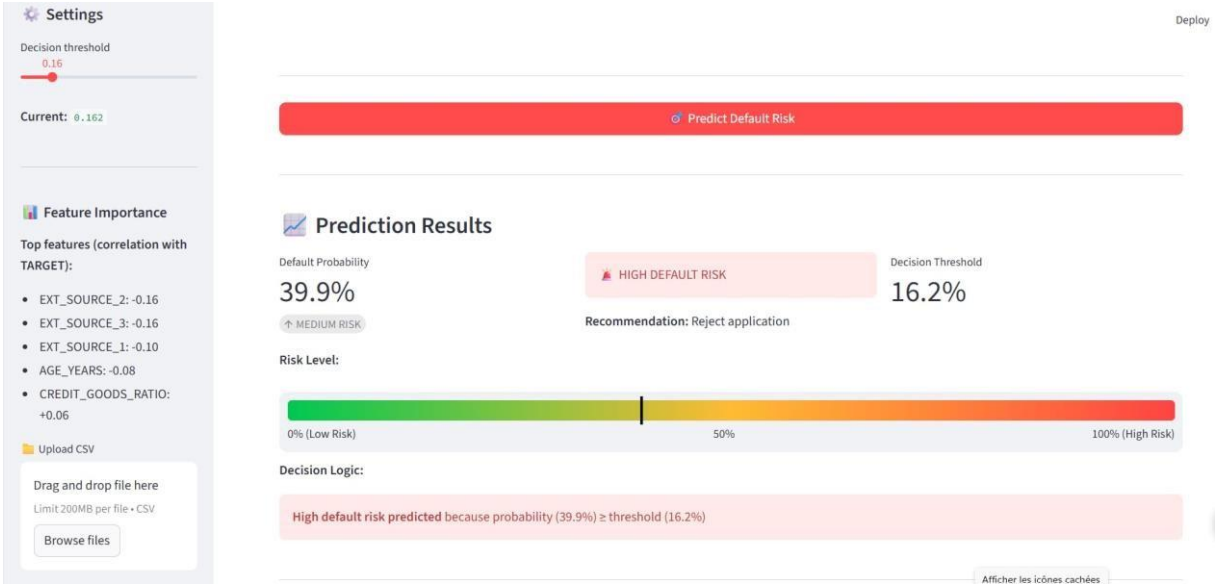


Figure 18 Default Risk Prediction Output (High-Risk Example)

Conclusions

This project set out to build a robust and operational credit-risk prediction system by combining advanced machine-learning techniques, rigorous data preprocessing, and real-world deployment practices. Across all stages of the workflow, the objectives were successfully achieved. The project began with extensive exploration and cleaning of the Kaggle credit-risk dataset, addressing severe missingness, nonlinear feature relationships, sentinel anomalies, and strong class imbalance. A comprehensive pipeline was developed to enforce reproducibility, applying outlier treatment, semantic correction, engineered ratios, winsorization, scaling, and rare-category consolidation. This ensured that the modeling phase rested on a clean, coherent, and well-structured feature space.

A second major objective—evaluating a wide range of machine-learning approaches—was also fully accomplished. Classical models (Logistic Regression), ensemble methods (Random Forest), and state-of-the-art boosting algorithms (XGBoost, LightGBM, CatBoost) were systematically compared. Each model was tested before and after SMOTE oversampling, across different train–test split strategies, and under multiple threshold-selection frameworks including default 0.5, Youden index, best-F1, and precision-driven cutoffs. These experiments provided critical insights:

- Class imbalance severely deteriorates both recall and precision in baseline models.
- SMOTE improves minority-class detection but requires careful threshold tuning to avoid false positives.
- Boosting models consistently outperform classical methods, with CatBoost and LightGBM delivering the strongest PR-AUC and ROC-AUC scores.
- Threshold engineering plays a major role in shaping risk-sensitive performance, highlighting the need to choose cutoffs aligned with institutional priorities.

The identified best-performing model—CatBoost trained on SMOTE-balanced data and retrained with strict leakage control—achieved strong performance across key metrics and demonstrated stability under validation and test conditions. Its superior handling of heterogeneous features, missing patterns, and nonlinear relationships aligns well with findings in the academic literature and industry practice. This model was subsequently packaged and deployed in an interactive Streamlit application, fulfilling the project's final objective: transforming an analytical model into a usable decision-support tool. The app enables probability scoring, threshold-adjustable classification, single-case simulation, and batch predictions, demonstrating how advanced machine-learning pipelines can be operationalized for real-world credit evaluation.

In terms of its contribution to the field, the project highlights the importance of meticulous preprocessing, thoughtful imbalance handling, and rigorous threshold analysis in credit-risk modeling. It provides a comparative evaluation across widely used algorithms, offering insights into model behavior under different sampling and decision strategies. The deployment workflow further demonstrates how predictive models can be made accessible to nontechnical stakeholders, bridging the gap between machine-learning research and practical financial applications.

Future work may explore several promising directions. First, incorporating explainability techniques such as SHAP values could improve transparency and regulatory compliance by identifying the drivers behind each prediction. Second, alternative resampling strategies (ADASYN, SMOTE-Tomek, undersampling hybrids) may further improve minority-class representation without excessive overfitting. Third, incorporating temporal drift analysis and model monitoring would help evaluate long-term stability in dynamic credit environments. Finally, expanding the Streamlit application to include model interpretability dashboards, applicant comparison tools, or API endpoints would enhance its suitability for organizational deployment.

Overall, the project successfully demonstrates how modern data analytics and machine learning—when combined with disciplined preprocessing and thoughtful deployment—can produce accurate, reliable, and operationally useful credit-risk prediction systems.

Appendices

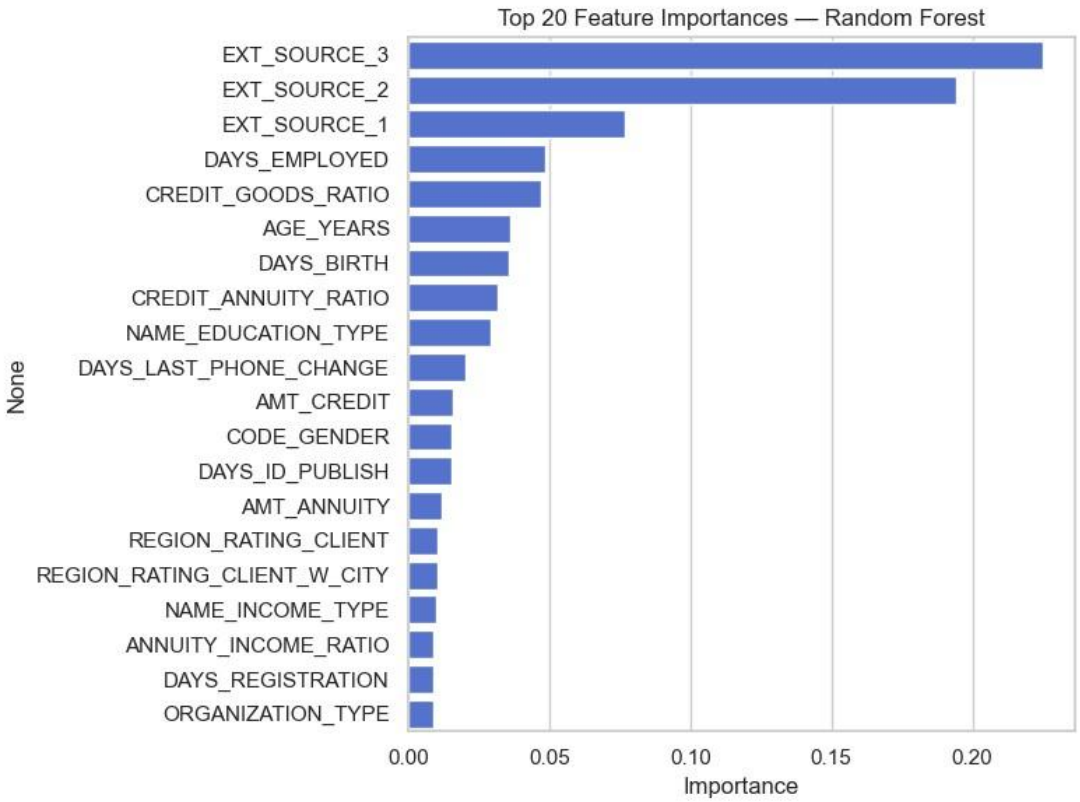


Figure 19 Top 20 Feature Importances from the Random Forest Model

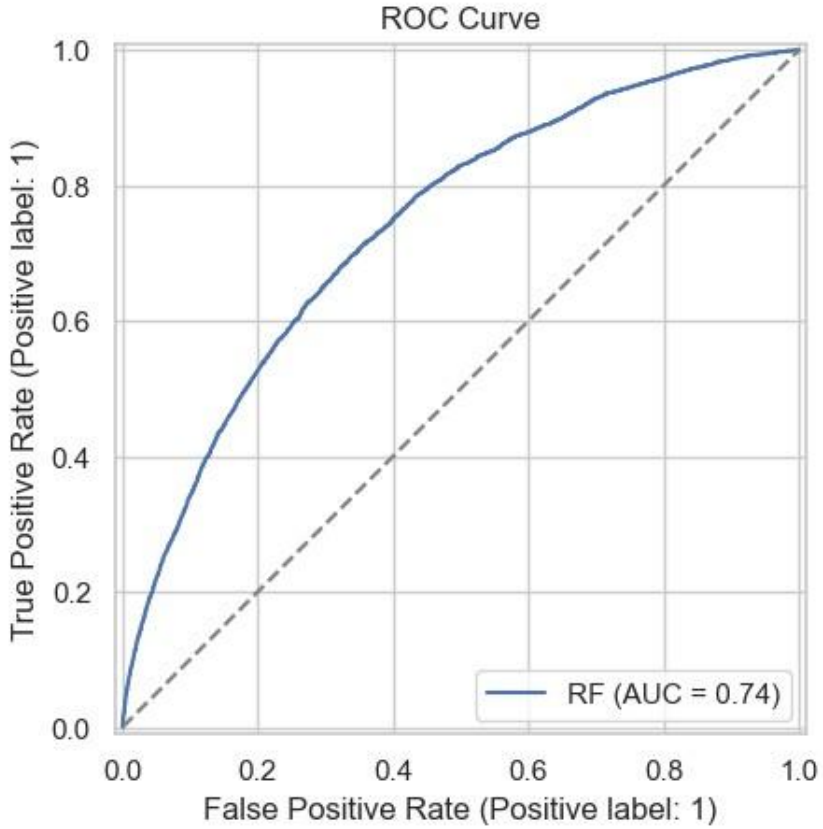
Final chosen model

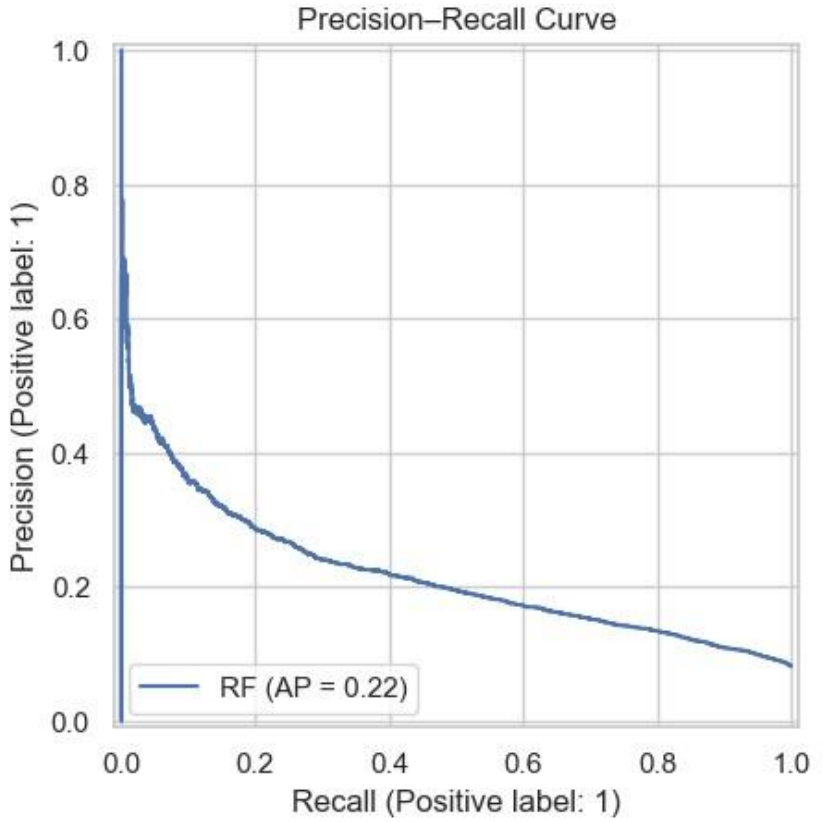
```
1) Retrain CatBoost safely (no test leakage) from
sklearn.model_selection import train_test_split from catboost import
CatBoostClassifier from sklearn.metrics import roc_auc_score,
average_precision_score, precision_recall_curve, confusion_matrix,
classification_report import numpy as np import time

# Use your SMOTE-resampled training set (X_train_sm, y_train_sm)
# Create a validation split from the training data only (15%)
X_tr_sub, X_val_sub, y_tr_sub, y_val_sub = train_test_split(
    X_train_sm, y_train_sm, test_size=0.15, stratify=y_train_sm, random_state=42
)

clf_cb = CatBoostClassifier(
    iterations=800,         # reduce for speed; increase later if desired
learning_rate=0.03,    depth=6,    random_seed=42,    verbose=100,
eval_metric='AUC',     task_type='GPU'       # change to 'CPU' if you
don't want GPU
```

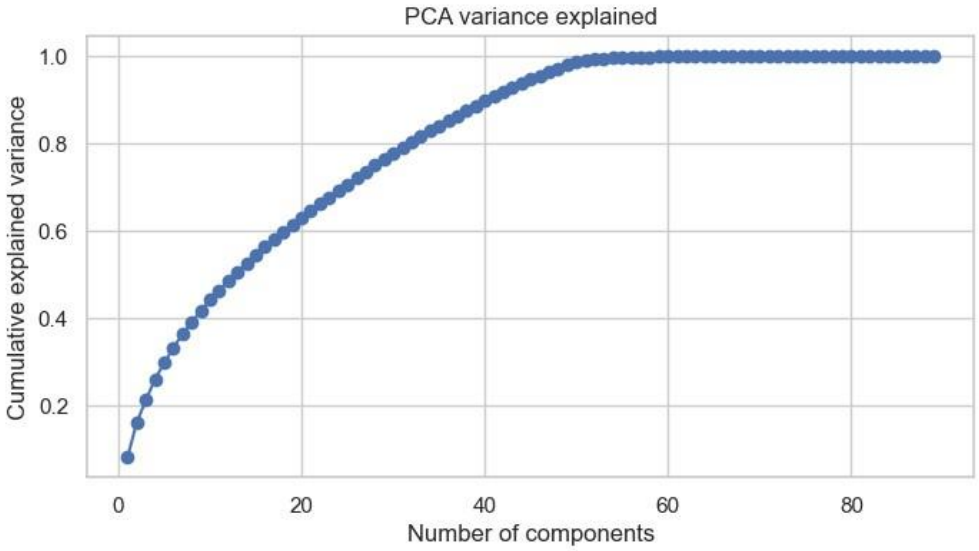
```
)

t0 = time.time() clf_cb.fit(X_tr_sub, y_tr_sub, eval_set=(X_val_sub, y_val_sub),
use_best_model=True, early_stopping_rounds=50)
print("Train time (s):", int(time.time()-t0))

# Predictions on untouched test set proba_cb
= clf_cb.predict_proba(X_test)[:,1] roc_cb =
roc_auc_score(y_test, proba_cb)
pr_cb = average_precision_score(y_test, proba_cb)

# best-F1 threshold prec, rec, thr =
precision_recall_curve(y_test, proba_cb) f1 = 2 * prec *
rec / (prec + rec + 1e-12) ix = np.nanargmax(f1[:-1])
best_thr = thr[ix]

print("\nCatBoost (clean retrain) -> ROC AUC:{:.4f} PR AUC:{:.4f}".format(roc_cb, pr_cb))
print("Best-F1 thr:{:.3f} Prec:{:.3f} Rec:{:.3f} F1:{:.3f}".format(best_thr, prec[ix], rec[ix],
f1[ix]))
print("\nConfusion matrix:\n", confusion_matrix(y_test, (proba_cb>=best_thr).astype(int)))
print("\nClassification report:\n", classification_report(y_test,
(proba_cb>=best_thr).astype(int), digits=4))
```

C

Plots of Random Forest model







Github Repository:

SaharBahloul/End-to-End-Credit-Default-Prediction-with-Data-Analytics-and-ML: An end-to-end credit-risk prediction system using the Kaggle Home Credit dataset. Includes data analytics, preprocessing, SMOTE imbalance handling, multiple ML models (RF, XGBoost, LightGBM, CatBoost), threshold optimization, and deployment of the final CatBoost model in a Streamlit app for real-time scoring.

References

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[2] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[3] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 3146–3154.

[4] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: Gradient Boosting with Categorical Features Support," *arXiv preprint arXiv:1810.11363*, 2018.

[5] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[6] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] Home Credit Group, "Home Credit Default Risk Dataset," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/competitions/home-credit-default-risk.

[8] Streamlit Inc., "Streamlit Documentation," 2025. [Online]. Available: https://docs.streamlit.io/.

[9] CatBoost Developers, "CatBoost Documentation," 2025. [Online]. Available: https://catboost.ai/.

[10] LightGBM Developers, "LightGBM Documentation," 2025. [Online]. Available: https://lightgbm.readthedocs.io/.

[11] XGBoost Developers, "XGBoost Documentation," 2025. [Online]. Available: https://xgboost.readthedocs.io/.

[12] Scikit-learn Developers, "Scikit-learn Documentation," 2025. [Online]. Available: https://scikit-learn.org/.

[13] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Morgan Kaufmann, 2017.

[14] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[15] J. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.