Predictive Analytics for Customer Retention: Designing and Evaluating a Machine Learning–Based Churn Prediction System

BLERINA ÇELIKU *, MARIJON PANO
Department of Informatics,
University "Fan S. Noli" Korçë,
Bulevardi Rilindasit 11, Korçë,
ALBANIA

*Corresponding Author

Abstract: - Customer churn represents a critical challenge for organizations in highly competitive sectors, particularly banking, where retaining existing clients is more effective than acquiring new ones. This study develops a machine learning–based system for churn prediction in the banking sector. The system was designed in Python, integrating modules for data preprocessing, feature engineering, and model training and evaluation. Within this system, five models are trained using five different algorithms (one training algorithm produces one trained model): Logistic Regression, Linear Discriminant Analysis (LDA), Decision Tree, Random Forest, and Gradient Boosting, based on an initial dataset of 10,000 clients, stratified prior to use, obtained from a public Kaggle dataset. For the evaluation of model performance, metrics such as accuracy in churner identification, precision, recall, and F1-score were employed, along with supplementary procedures such as ROC–AUC curve analysis and bootstrapping for model stability assessment. Results demonstrate that ensemble models, particularly those trained using the Random Forest and Gradient Boosting algorithms, outperform baseline approaches across all selected evaluation metrics. Moreover, the 95% confidence intervals, which are narrower than those of the other trained models, along with the ROC–AUC curve analysis, indicate stable predictive performance for these two models. As a result, these algorithms are recommended for identifying potential churners across various business and industrial use cases. Their reliable performance indicates that they can effectively support customer retention campaigns.

Key-Words: - churn prediction, banking sector, machine learning, Random Forest, ensemble learning, customer retention.

Received: January 13, 2026. Revised: March 19, 2026. Accepted: May 22, 2026. Published: August 3, 2026.

1 Introduction

Customer churn refers to the phenomenon of clients discontinuing their relationship with a service provider, whether by closing accounts, switching to competitors, or ceasing to purchase products. In the banking sector, churn directly affects profitability and long-term stability, since acquiring new clients is estimated to be five times more costly than retaining existing ones, [1]. High churn rates not only impact revenues but also undermine customer trust and brand reputation.

In recent years, advances in data analytics and machine learning have provided financial institutions with tools to predict churn and implement targeted interventions. By analyzing customer behavior, demographic factors, and transaction histories, banks can identify individuals at risk of leaving and apply retention strategies, such as loyalty programs or personalized offers, [2]. Predictive models thus transform churn analysis from a reactive process into a proactive, data-driven strategy.

The objective of this research is to design, implement, and evaluate a churn prediction system by training and evaluating 5 models using 5 different machine learning algorithms. Building upon publicly available datasets, the system makes use of Python libraries for the pre-processing stage, including: "data loading and cleaning", "Stratified Train–Test Partitioning", "feature encoding and engineering", model training, and trained models' performance evaluation. The main Research Question, which serves as a central axis of this study, is:

RQ1- *Under the same training datasets and evaluation metrics, which machine learning algorithms can be used to train models that*

provide superior discrimination performance and stability for banking churn prediction ?

To fully address this Research Question, we have to practically implement a full model training and evaluation pipeline for the sake of developing a churn prediction system. During this exploratory journey, the answer to the Research Question may be seen as a puzzle composed of the answers to 3 smaller-scale, yet interconnected sub-research questions:

RQ.1.1- *What steps have to be taken during the training pipeline, and what are the characteristics of a good representative dataset that should be chosen for the training purpose ?*

RQ.1.2- *What evaluation metrics should be used to best measure the performance of trained models?*

RQ.1.3- *What instruments can be used to verify the robustness of findings and the stability of the system among datasets of different sizes?*

These Research Questions guide the direction of this study and provide the foundation for the analyses and discussions presented in the following sections.

2 Research Methodology

In modern financial entities, clients are not stationary. They may silently leave one bank for the competitive offers of another one. This phenomenon, called "churn", may cause a major disruption in these clients' spending in the original bank over time, quantified by the "Customer Lifetime Value" (CLV) metric, [3]. As a result, prediction models are increasingly being used to prioritize clients based on their likelihood of churn. These models continuously feed the consumer retention campaign, with the focus expanding and increasing the CLV of "possible-churner" clients over time, and in that way preserving the institution's profitability.

Algorithms used to train "churn prediction models" are essential for the successful identification of churners in real client datasets. Logistic Regression is a commonly used baseline algorithm due to the interpretability of its probabilistic outputs, [4]. The more recent literature shows a progression toward the creation of models based on advanced Machine Learning algorithms. Specifically, models trained on Random Forest and Gradient Boosting are reported to achieve both high precision in churners identification and high discrimination abilities, [5], [6]. In addition, hybrid models have been trained, with composite algorithms, where each one of them has been optimized to target and evaluate a specific feature within the client behaviour and their consumption pattern, [7].

An entry, yet important point in the evaluation of the trained churn-prediction models is the choice of evaluation metrics that will lead the comparative analysis between models. The general consensus in the literature is that no single metric is the best representative of the overall trained model performance. Since performance is a multifaceted concept, the approach toward it should involve the analysis of many individual, yet interconnected, performance metrics. The evaluation metrics, in this case, include Accuracy of the model, Recall, Precision, and F1-Score, [8].

Complementary procedures may be obtained to improve the explainability of crucial aspects of the model analysis, such as the discriminative capacity of the model that may be visualized via ROC-AUC curve illustration. Especially, the AUC value demonstrates this capacity independent of the threshold-defined probability value for churners vs non-churners.Moreover, since human decisions to churn/not churn are prone to emotions and the impact of relatives, the churn risk may be inherited through connections of the susceptible client with other ones, [9]. In that way, churn- related risk can be further mapped to graph nodes and their relationships within graphs.

On the same line as the choice of evaluation metrics, the literature focuses even on models' performance across different data conditions, including, but not limited to, the dataset size. The analysis of models' performance under varying dataset sizes and configurations is key for gaining a complete understanding of the models' behaviour in real-world scenarios and supports their application in these environments. To better understand the models' behaviour, bootstrap resampling and confidence intervals for targeted metrics have been widely used, especially with a focus on analysing the model stability and robustness, [10], [11].

Finally, the use of controlled environments for model training and evaluation is vital in order to have reproducible and auditable results for researchers in the same field. Python libraries and their implementation IDEs create a good ecosystem where researchers can experiment, train and evaluate models in a controlled and reproducible manner,

[12]. From our standpoint, it is vital for the sake of reproducibility and comparison of results, where multiple models are trained and assessed under the same conditions. Overall, from the literature review, the roadmap for the successful implementation of churn prediction models is clear and passes through these major milestones (as confirmed by several studies):

1. The selection of the model training and evaluation environment, including the choice of programming language, IDE, and related implementation libraries. For us, as authors, Replit IDE offers a lightweight, yet very comprehensive stack for the training and evaluation pipeline. It has also been used in many other related studies.
2. The selection of training algorithms. *We chose 5 algorithms for our models' training, evolving progressively from a traditional and statistical one ("Logistic Regression") to ML ensemble algorithms ("Random Forest" and "Gradient Boosting").*
3. The implementation of the training pipeline, including the preprocessing stage (Data Loading and Cleaning, Stratified Train–Test Partitioning, Feature Extraction and Encoding) and the model training stage.
4. The selection of evaluation metrics (Accuracy, Recall, Precision, F1-Score, ROC- AUC Curve, Stability analysis via Bootstrapping method).
5. The implementation of the evaluation procedure via evaluation datasets of different size.
6. The selection of the models with the best performance for incorporation in bigger explainable churn prediction systems via the GUI inference of the trained models.This roadmap also serves as the leading one for the methodology of our experimental study.

3 Methodology

3.1 The Objectives of the Methodology and Experimental Framework

The aim of the study is to evaluate five churn prediction models, trained with algorithms of different complexity, in a comparative and structured procedure within a controlled Replit Python environment. We would like to mention that in the literature on the evaluation and comparison of ML algorithms, even small differences in data handling and preprocessing procedures may have serious effects on the behaviour of those algorithms, [13], [14]. For that reason, the architecture of the experiment was structured as a modular and controlled pipeline, where the preprocessing of training data was done only once, at first, before all the model training took place.

All our experimental work was done in Replit IDE, with a standard free-tier plan. All individual tasks needed for the full model training and evaluation pipeline were programmed in the Python language (v. 3.11). In this context, we made use of the "*scikit-learn*" Python library to ensure that identical feature transformations were applied to the training dataset for all models prior to the training phase.

3.2 Training Pipeline

Five Machine Learning algorithms, belonging to the class of supervised learning, were considered for model training, with a one-to-one ratio (one algorithm trains one model) : Logistic Regression, Linear Discriminant Analysis (LDA), Decision Tree, Random Forest, and Gradient Boosting. To ensure a shared training framework, each algorithm used the same Python "scikit-learn" preprocessing pipeline, ensuring that identical feature transformations were applied across all models.

3.2.1 The Choice of the Training Dataset

The chosen dataset consisted of 10,000 banking customers described by demographic and financial attributes, with a binary churn indicator, [15]. The following script examined the target distribution to quantify class proportions:

```
import pandas as pd
df = pd.read_csv("Bank_Customer_Churn_Prediction.csv")
df["churn"].value_counts(normalize=True)
```

We used the Microsoft Excel built-in function "Sum" to calculate the total number of "churners" and "non-churners" in the chosen dataset, prior to the preprocessing of data. As expected, given that churn datasets are naturally composed of more "non-churners" than "churners," our calculation showed that "non-churn" instances made up about 79.6% (n1= 7960 clients), while churn instances accounted for 20.4% (n2 = 2040 clients) of the chosen training dataset.

A very important point to share is that within the "scikit-learn" functionalities, there is also an option to adjust the parameter "class_weight" to the value "balanced" ("class_weight = 'balanced'"). We made use of this feature because it assigns weights based on how often each class appears, giving more

importance to less frequent classes and increasing the penalty when those instances are misclassified during optimization.

3.2.2 Data Loading and Cleaning

At the initial stage, the dataset was imported to Replit IDE from CSV file using the call of the Python function *load_data(),* within *"scikit-learn"* library. The target was defined by the binary churn variable, and predictor attributes were separated accordingly:

```
y = df["churn"]
X = df.drop(columns=["churn", "customer_id"])
```

The attribute "*customer_id",* represents a unique identifier without predictive meaning, therefore was excluded from modeling. Such identifiers, when included, may cause information leakage by enabling instance-level memorization rather than learning generalized structural relationships between explanatory variables and churn behavior.

3.2.3 Stratified Train – Test Partitioning

The evaluation of representative performance was conducted through dataset partitioning into training and test subsets using an 80/20 split. Stratified sampling was also utilized to preserve the observed class proportions. Stratification maintains the evidence-based class distribution across partitions, minimizing sampling- caused variability in performance metrics and stabilizing discrimination estimates in imbalanced classification environments, [16]. By maintaining class proportions across these subsets, the evaluation reflects the distribution of the original data:

```
from sklearn.model_selection import
train_test_split
def split_dataset(X, y):
    return train_test_split(
        X, y,
        test_size=0.2,
        stratify=y,
        random_state=42 )
```

Stochastic components such as data partitioning and model initialization were controlled through a value that is supposed to be fixed and pseudo-random (random_state=42).

Using a fixed random value guarantees deterministic reproducibility of experimental outcomes and removes variability attributed only to random sampling effects.

One common and important recommendation in empirical machine learning research is the controlling of stochasticity, which improves replicability and facilitates consistent comparative evaluation across used models, [17].

3.2.4 Feature Engineering and Encoding

The next step in the training pipeline was to identify, target, and represent the training features from the available datasets. It is important to note that we did not create new features (neither categorical nor numerical) or merge existing attributes into larger features for the purpose of model training. Instead, we applied a standardized "data inspection and encoding" procedure which, based on the type of feature, followed one of the following two strategies:

1. For categorical features, we used one-hot encoding to generate binary indicator variables. We also excluded the reference category in order to avoid redundancy and model misinterpretation.
2. For numerical features (including the column "will churn/will not churn"), feature values were normalized using z-score normalization, defined as: $x^* = \frac{x-\mu}{\sigma}$, implemented using `StandardScaler()` Python library, where μ and σ are estimated exclusively from the training subset.

We chose the feature encoding to follow the two procedures above because standardization ensures stable optimization for models such as Logistic Regression and LDA, whose parameter estimation depends on the magnitude of the training features, [18]. Tree-based models are less sensitive to scaling; however, within a shared preprocessing and training framework, their results become more structurally aligned with those of other models and easier to compare, [19]. Since encoding was an integral early part of the training pipeline, transformation parameters were learned only from the training data and then separately applied to the test subset. This ensures that no information leakage occurs between the training and test subsets.

3.2.5 The Training Stage (Model Construction)

The pipeline object was then serialized using the .joblib format, as illustrated in Figure 1.

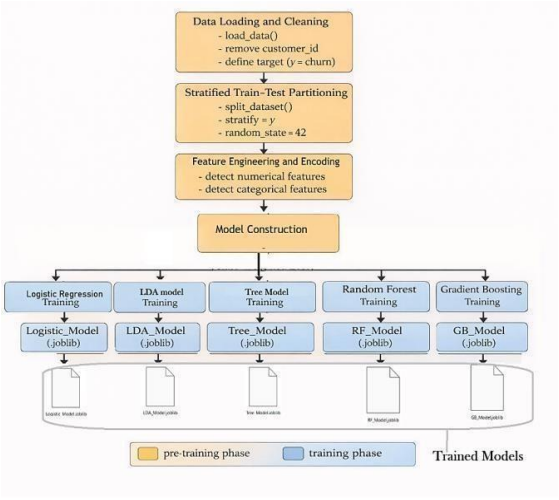

Fig. 1: The training pipeline
Source: Created by the authors

The code snippets from the shared training pipeline, along with the structure of the unified training pipeline, are provided below:

```
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.pipeline import Pipeline
import joblib

models = {
    "Logistic_Model.joblib": LogisticRegression(
        class_weight="balanced",
        random_state=42,
        max_iter=1000
    ),
    "LDA_Model.joblib": LinearDiscriminantAnalysis(),
    "Tree_Model.joblib": DecisionTreeClassifier(
        class_weight="balanced",
        random_state=42
    ),
    "RF_Model.joblib": RandomForestClassifier(
        class_weight="balanced",
        n_estimators=200,
        random_state=42
    ),
    "GB_Model.joblib": GradientBoostingClassifier(
        n_estimators=200,
        random_state=42
    )
}
```

Once the training pipeline implemented in Python, reached its final stage, it produced five trained models in .joblib format: *name_of_used_training_algorithm_for_that_model.joblib.*

3.3 Evaluation Strategy for Trained Models

3.3.1 Customer Churn Probability Estimation and Decision-Making

After training in the unified pipeline using the dataset of 10,000 clients, the five trained models were able to produce, for each client in the test dataset, a probability value ranging from 0 (representing a 0% chance of churn) to 1 (representing a 100% chance of churn). To convert this predicted churn probability into a concrete and operational decision (since the client effectively has only two outcomes: *will churn or will not churn).* For the sake of model comparison, we applied the following decision rule to the final evaluated datasets:

Probability of churning : $\begin{cases} >= 0.5\ (p \geq 50\%),\ \textit{“churner”} \\ < 0.5\ (p<5\%),\ \textit{“not churner”} \end{cases}$

```
y_prob = model.predict_proba(X_test)[:, 1]
y_pred = (y_prob >= 0.5).astype(int)
```

Using this rule, each prediction made by the trained models falls into one of four categories : *True Positives (TP)* –A churner is correctly identified: the model predicts “churn,” and the client leaves the bank.
True Negatives (TN) – A non-churner is correctly identified: the client continues using the bank’s services, and the model predicts “non-churn.”
False Negatives (FN) – A churner is incorrectly classified as a non-churner: the client leaves the bank, but the model predicts “non-churn,” preventing inclusion in retention strategies.
False Positives (FP) – A non-churner is incorrectly classified as a churner: the client remains with the bank, but the model predicts “churn,” potentially leading to unnecessary retention efforts and additional costs.

3.3.2 Evaluation Metrics

Based on the classification outcomes of each model, which fall into one of the four categories described above, we selected the following representative metrics for the evaluation and comparison of the trained models. While these metrics are not the only ones available for assessing the effectiveness of machine learning models, they are widely used in the literature to provide valuable insight into model performance and comparability.
Accuracy of the model : This metric is used to evaluate the effectiveness of the model by measuring the total number of correct predictions (true positive churners + true negative non-churners) over the total number of customers in the evaluation dataset.

$$Accuracy = \frac{TP + TN}{TP+TN+FP+FN}$$

```
from sklearn.metrics
import accuracy_score
accuracy =
accuracy_score(y_test,
y_pred)
```

Accuracy provides a general overview of how effective a model is in correctly identifying churners versus non-churners. Ideally, if the numbers of churners and non-churners were balanced within the evaluation dataset, accuracy alone would be one of

the strongest indicators of model effectiveness. However, in practice, non-churners typically greatly outnumber churners. As a result, a model that classifies most clients as non- churners may still achieve artificially high accuracy. For this reason, accuracy must be interpreted alongside more targeted metrics.

Recall of the trained model : Recall measures the model's ability to correctly identify "actual churners". It answers the question *"Out of all clients who truly churn, how many did the model correctly detect? "*.Recall as a parameter focuses on coverage of the "churners subset" – the ability to capture as many real churners as possible.

$$Recall = \frac{TP}{TP + FN}$$

A high recall means that only a few real churners are missed by the model, which demonstrates high effectiveness. Since the primary goal of churn modeling is to identify customers before they leave, recall is often one of the most important performance indicators. Low recall means that many customers leave without being detected in advance.

Precision of the trained model : Precision measures how accurate the model is when it predicts churn. It answers the question : *"Out of all clients predicted as churners, how many actually churn?"*

$$Precision = \frac{TP}{TP + FP}$$

```
from sklearn.metrics import
precision_score
precision = precision_score(y_test,
y_pred)
```

High precision means that most customers flagged as high risk of leaving actually do leave. Low precision means that many flagged customers would have stayed anyway, leading to unnecessary intervention costs. Precision, in contrast to recall, focuses on the efficiency of positive predictions.

F1-Score of the trained model: Since recall and precision focus on different aspects (coverage of churners versus efficiency in identifying churners) of the trained models, there is a natural need to evaluate performance using a metric that captures both coverage and precision in churner identification. This metric is the F1-score of the trained model:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Since the F1-score takes into consideration both precision and recall, it is used to provide a balanced and comprehensive evaluation of model performance based on both metrics.

The Receiver Operating Characteristic (ROC) curve and its summary measure, the Area Under the Curve (AUC) for the trained model : The evaluation metrics discussed previously are based on a fixed decision threshold (0.5). However, in real-world business scenarios, this threshold may vary depending on strategic objectives. For instance, a bank may lower the threshold to identify more potential churners or increase it to reduce unnecessary retention actions. Therefore, it is essential to assess the model's ability to rank customers by churn risk independently of any specific cutoff. The ROC curve evaluates two key quantities:

1- The True Positive Rate (TPR), which is equivalent to recall, represents the proportion of actual churners correctly identified by the model.

2- The False Positive Rate (FPR) represents the proportion of non-churners incorrectly classified as churners, indicating the rate of false alarms.

By varying the classification threshold from 0 to 1, the ROC curve captures how TPR and FPR change and provides a comprehensive view of model performance across all possible thresholds. The Area Under the Curve (AUC) summarizes the ROC curve into a single value between 0 and 1. It can be interpreted as the probability that the model assigns a higher churn risk score to a randomly selected churner than to a randomly selected non-churner, making it a threshold-independent measure of the model's discriminative ability. For example, an ROC- AUC value of 0.87 means that in 87% of randomly selected churner–non-churner pairs, the churner receives a higher predicted probability.

```
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_test, y_prob)
```

Unlike accuracy, ROC-AUC Value does not depend on a fixed threshold. It evaluates how well the model ranks customers by risk. This is particularly useful when businesses prioritize customers based on risk ranking rather than binary decisions.

3.3.3 Assessing Stability of Trained Models

A single ROC-AUC score provides a snapshot of model performance; however, it depends on the specific evaluation dataset used. If the evaluation dataset were slightly different, the result might change. To better assess the reliability of the trained

models, it is necessary to examine the variability of this performance. For this reason, nonparametric bootstrap resampling was applied to the evaluation datasets. By repeatedly drawing samples with replacement (for the datasets of 500, 650, 1200, 1500, and 2000 customers from Kaggle) [20], [21], [22], [23] and recalculating the ROC-AUC value each time, a distribution of ROC-AUC values is obtained rather than a single estimate. This approach provides insight into how stable the model's performance is across different evaluation datasets.

Through the Python code provided below, we calculated the 95% confidence interval to assess the stability of the trained models. This interval represents the range within which the model's true discriminative performance is likely to fall under repeated sampling:

```
import numpy as np
B = 1000
rng = np.random.default_rng(42)
boot_aucs = []
for _ in range(B):
    idx = rng.integers(0, len(y_test), len(y_test))
        if len(np.unique(y_test.iloc[idx])) < 2:
        continue
    auc_b = roc_auc_score( y_test.iloc[idx], y_prob[idx] )
    boot_aucs.append(auc_b)
ci_lower = np.percentile(boot_aucs, 2.5)
ci_upper = np.percentile(boot_aucs, 97.5)
```

If the interval is narrow, the model's performance is considered stable; if it is wide, the performance is more sensitive to variations in the data. Taking a broader evaluation across the five datasets, models that consistently achieve higher ROC-AUC values and narrower 95% confidence intervals demonstrate better performance in identifying churner clients than those with lower AUC values and wider confidence intervals obtained through the bootstrapping method.

The section "Results and Discussion" evaluates the performance of the five trained models for churn prediction, each trained using a different algorithm (Logistic Regression, LDA, Decision Tree, Random Forest, and Gradient Boosting), based on the metrics discussed in this methodological section: accuracy, recall, precision, F1-score, ROC–AUC Value analysis, and the 95% confidence interval for model stability obtained via bootstrapping.

4 Results and Discussion

4.1 The Evaluation of 5 Trained Models with the N= 2000 Client Dataset

The evaluation of the trained models, using metrics: Accuracy of the model, Recall, Precision, F1-Score, was firstly conducted in a Replit Enivronment ,with code mentioned in the previous section, in Python Language. SciPy and Scikit-Learn were used for bootstrapping and the identification of 95% CI Intervals for each trained models, with the sake of evaluating the model stability among different datasets.

The corresponding quantitative performance metrics for 5 trained models for the N=2000 evaluation dataset are summarized in Table 1 (Appendix). The churn prediction model trained with the Gradient Boosting algorithm demonstrates the strongest overall performance among the four other models. Its accuracy of 88% indicates that it achieves the largest proportion of correct classifications (churners vs. non-churners) across all clients. Its leading recall value of 0.84 confirms that it is the most effective model in capturing actual churners, which is an essential requirement for proactive retention strategies.

At the same time, its F1-score is the highest compared to the other trained models, showing that this strong churn detection capacity is achieved without a significant increase in incorrect churn predictions. In 87% of randomly selected churner–non-churner pairs, the model assigns a higher probability of churn to the actual churner (compared to 85% in the case of the Logistic Regression and Random Forest models, 81% for LDA, and a lower value of 69% in the case of the model trained with the Decision Tree algorithm).

Churn prediction models trained with Random Forest and Logistic Regression algorithms demonstrate a bit lower, yet considerable accuracy level (86% and 82%, respectively), which from our perspective is satisfactory enough to be used in real business use cases and successful client-retention campaigns.

On the other side, models trained with the Decision Tree and LDA algorithms demonstrate weaker model performance metrics among all evaluation metrics. The lower-than-other-models accuracy of 75% for the model trained with the Decision Tree algorithm, combined with its weaker discrimination capacity, indicates that this model is less likely to identify potential churners, and as a consequence should not be considered a first-line choice for practical implementations versus other models.

As per the overall time of each trained model to give its evaluation on the 2000-client dataset, we made use of the timeit library, developed in Python, inside the Replit development environment.

We would like to mention that the Python timeit library is an open-source, easy-to-use utility that can also be employed to evaluate the execution time of

machine learning–based models, [24]. All five models' execution time was measured under the same testing conditions and the same Replit cloud container processing capacity (free tier: approximately 1 virtual CPU and 2 GB of RAM per execution environment).

Logistic Regression and LDA models provided their responses with probability values for client churn within 1.35 s and 1.3 s, respectively. Since models trained with Random Forest and Gradient Boosting inherently rely on more complex internal data structures to generate individual client churn predictions, their execution times were 4.3s and 5.1 s, respectively.

We would like to emphasize that, in this trade-off between model execution time and performance metrics, we strongly advocate the use of Gradient Boosting and Random Forest instead of LDA and Logistic Regression, as performance in identifying potential churners (model accuracy and discriminative capability) is crucial for client retention decisions. In addition, since the evaluation procedure with real banking datasets is typically performed only once or twice per client retention campaign (i.e., the generation of churn probabilities is more of a "snapshot" procedure rather than a repetitive one), execution time does not pose a bottleneck for the performance of the system in which it is integrated.

4.2 A Graphical Representation of ROC-AUC Curve for the 2000 Client Evaluation Dataset

To complement this quantitative analysis, visual comparative graphs of ROC-AUC curve of the 5 trained models were conducted using Matplotlib and Seaborn Python under the same evaluation framework. Figure 2 presents the ROC-AUC curves, in the same graph, for all 5 trained models in correspondence to the evaluation of churning probabilities of the 2000 client dataset.

The horizontal axis represents the False Positive Rate (FPR), while the vertical axis represents the True Positive Rate (TPR). The diagonal line (y = x trajectory) corresponds to Random Classification performance.

ROC-AUC Curves demonstrate a clear advantage in precision of the churners identification and higher discrimination capacity for the Gradient Boosting trained model, followed closely by Random Forest and Logistic Regression. This advantage is present throughout the entire FPR range. LDA occupies an intermediate position, while the Decision Tree remains clearly separated below the others. An important insight that can be viewed by the graph is that the low intersection of curves indicates that relative model ranking does not depend on threshold selection (the 0.5 threshold we selected at the initial stage of evaluation; where clients with possibility greater than 50% of churning were considered "churners", otherwise "non-churners") and that the trained models preserve their performance metrics throughout all threshold conditioning.

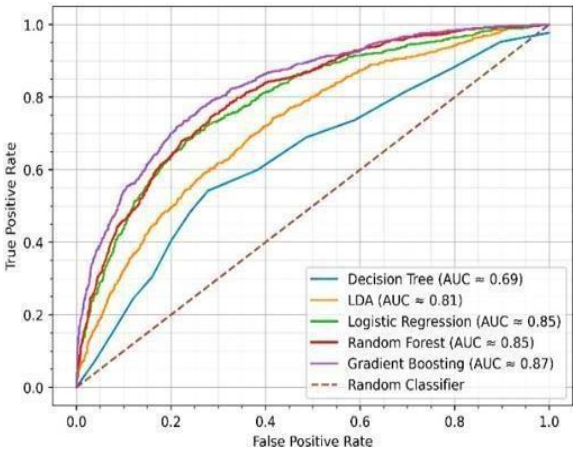

Fig. 2: ROC-AUC Curves for All Trained Models
Source: Created by the authors.

The analysis in Section 4.1 and 4.2 focused on the evaluation of the 5 trained models within the 2000-client dataset. In order to capture the "full, comprehensive view of model behaviours" and to find whether they provide consistent performance metrics across different dataset sizes, we repeated the ROC- AUC Value evaluation on four more evaluation datasets containing 500, 650, 1200, and 1500 clients. Table 2 (Appendix), shows how the ROC-AUC value of the trained models evolves across all five evaluation datasets of increasing size (those four datasets plus the 2000-client dataset mentioned in 4.1).

If we take into consideration how the AUC value evolves, we can draw two important conclusions:

1. For all trained models, regardless of which one of the five training algorithms was used, the discrimination capacity gradually improved (as the AUC value increases) with the increase in the dataset sizes used in the evaluation procedure.
2. The increase of AUC value for models trained with Gradient Boosting and Random Forest algorithms clearly demonstrates the robustness and stability of findings. Specifically speaking, as the evaluation dataset gradually evolved from a 500 client dataset to a 2000 client dataset, in correlation with that process, the AUC of the trained model evolved from a value of 0.82 (stating that in random pairs of churners and non-churners the

model correctly identifies the churner client in 82% of cases) to 0.87 (87% of cases).

Following the above two projections, we can argue that, in larger client datasets derived from real financial institutions, the AUC of the trained models algorithms offers good performance, stability, and reliability. As a result, models trained with these two algorithms can be incorporated as parts of larger software systems aiming to target clients who are likely to churn in the near future and provide them with extra dedicated and personalized services as part of targeted client retention campaigns.

4.3 The Development of an Explainable GUI Layer on top of the Trained Models

The set of probabilistic values for individual client churner prediction, obtained from the trained models with the evaluation datasets, should turn into practical and operational decisions. Since we performed the models training and evaluation procedure inside the Replit environment, we made use of the Replit AI Agent for Coding to create a composite and explainable Graphical User Interface layer on top of our models evaluation, where we could follow up with the probability of churning for specific clients and the overall metrics of the evaluation datasets.

Figure 3 an instance of this GUI layer, visualizes the metrics of the trained LDA model for the 2000-client evaluation dataset in its upper part. In the lower part, it displays, in tabular form, the churning probability of individual clients present in the evaluation dataset.

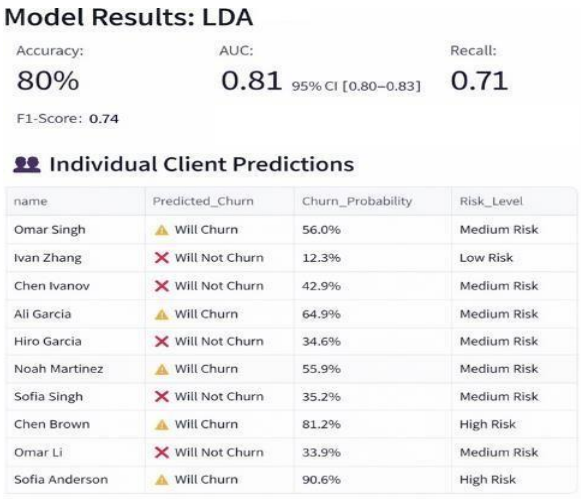

Fig. 3: GUI for Model Inference (Case of LDA with Gradient Boosting and Random Forest trained model, with 2000 clients evaluation dataset)
Source: created by the authors.

We have set the "Predicted_Churn" column to display one of these two messages based on the previously defined threshold of the churn probability of the client (for the client with less than 50% probability for churning, the message "Will Not Churn" is displayed; meanwhile, for the client with probability of churning equal to 50% or more, "Will Churn" is displayed).

The third column, *"Risk_Level"*, is an add-on to the explainability of the system, involving for each client in the evaluation datasets one of three risk levels for churning: *probability less than 30% for churning – "Low Risk"; 30% to 70% – "Medium Risk";* and *above 70% – "High Risk"*.

Despite the main objective of this study, that was to evaluate which training algorithm performed more efficiently for churn prediction analysis, we would like to emphasize that the trained model inference and future reuse pose no significantly added complexity. This is especially true if all stages, such as feature pre- processing, model training and evaluation, and finally model GUI inference, are configured and initiated in the same controlled development environment (case of Replit IDE).

The GUI of the trained models is an easy-to-develop software component nowadays with the help of assistive coding technologies; yet it is used to provide detailed information regarding specific clients and to make informed decisions in client retention campaigns.

5 Conclusion & Recommendations

This study developed and evaluated a machine learning–based churn prediction system within a controlled and unified framework, where five models were trained under identical conditions to allow a fair comparison. Ensemble models, particularly Gradient Boosting and Random Forest, provide the strongest performance across all evaluation metrics, combining higher accuracy with a stronger ability to identify churners and more stable results across different datasets.

At the same time, simpler models such as Decision Tree and LDA demonstrate lower discrimination capacity and reduced robustness, which limits their use in real-world applications where reliability is essential. Another important aspect emerges from the evaluation: as the dataset size increases, model performance improves gradually. This leads us to the conclusion that, in real-world datasets containing much more clients than in our evaluation datasets, the performance metrics of the trained models would be within a very satisfactory range.

In practical terms, the slightly higher computational cost of ensemble models is justified by their stronger predictive performance. Since identifying potential churners is a key objective and the evaluation procedures are not of a repetitive nature (they may need to be repeated only a small number of times within a campaign, or, for example, on a quarterly or yearly basis), in this trade-off between model execution time and performance metrics, we strongly advocate the use of models trained with "Gradient Boosting" and "Random Forest" ML algorithms.

These models can be integrated into customer retention systems to precisely identify clients at risk of churn and support targeted actions to increase their "Customer Lifetime Value."

Future work can extend this study by training and evaluating hybrid models in the context of churn prediction systems. These hybrid models may take into consideration behavioral and time-based features that better capture customer patterns. The integration and study of the explainability layer on top of existing churn prediction systems would make the system not only predictive, but also more useful for decision-making.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

The author wrote, reviewed and edited the content as needed and verifies that none utilized artificial intelligence (AI) tools were used. The author takes full responsibility for the content of the publication.

References:

[1] Zhang, Q., Predicting customer churn through lifetime value: Analysis of multidimensional influencing factors and their interactions, *Proceedings of the 2nd Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Digital Economy and Artificial Intelligence (DEAI)*, 2025, pp. 182–187. https://doi.org/10.1145/3745238.3745270.

[2] Verbeke, W., Dejaeger, K., Martens, D., Hur, J., Baesens, B., New insights into churn prediction in the telecommunication sector: A profit driven data mining approach, *European Journal of Operational Research*, Vol. 218, 2012, pp. 211–229. doi: 10.1016/j.ejor.2011.09.031.

[3] Moro, S., Cortez, P., Rita, P., A data-driven approach to predict the success of bank telemarketing, *Decision Support Systems*, Vol. 62, 2014, pp. 22–31. https://doi.org/10.1016/j.dss.2014.03.001.

[4] Pan, H., Logistic Regression for Customer Churn Analysis: The Role of Data Proportion and Class Balancing in Model Performance, Finance & Economics, Vol. 1, 2025, Article FE007392. https://doi.org/10.61173/x28wbd81.

[5] Văduva, A. G., Oprea, S. V., Niculae, A. M., Bâra, A., Andreescu, A. I., Improving churn detection in the banking sector: A machine learning approach with probability calibration techniques, Electronics, Vol.13, 2024, Article 4527. https://doi.org/10.3390/electronics13224527.

[6] Singh, P. P., Anik, F. I., Senapati, R., Sinha, A., Sakib, N., Hossain, E., Investigating customer churn in banking: A machine learning approach and visualization app for data science and management, *Data Science and Management*, Vol. 7, 2024, pp. 7– 16. https://doi.org/10.1016/j.dsm.2023.09.002.

[7] Rouhani, S., Mohammadi, A., A Novel hybrid Forecasting Approach for Customer Churn in Banking Industry, *Journal of Information and Knowledge Management*, Vol. 21, 2022, Article 2250089. https://doi.org/10.1142/S0219649222500897.

[8] Ashraf, R., Bank customer churn prediction using machine learning framework, *Journal of Applied Finance and Banking,* Vol. 14, 2024, pp. 65–109. https://doi.org/10.47260/jafb/1445.

[9] Lima Lemos, R. A., Silva, T. C., Tabak, B. M., Propension to customer churn in a financial institution: A machine learning approach, *Neural Computing and Applications*, Vol. 34, 2022, pp. 11751–11768. https://doi.org/10.1007/s00521-022-07067-x.

[10] De Bock, K. W., Van den Poel, D., Reconciling performance and interpretability in customer churn prediction using ensemble learning based on generalized additive models, *Expert Systems with Applications*, Vol. 39, 2012, pp. 6816–6826. https://doi.org/10.1016/j.eswa.2012.01.014.

[11] Sana, J. K., Abedin, M. Z., Rahman, M. S., Rahman, M. S., A novel customer churn prediction model for the telecommunication industry using data transformation methods and feature selection, *PLOS ONE*, Vol. 17, 2022, Article e0278095. https://doi.org/10.1371/journal.pone.0278095.

[12] Teixeira, A. R., Brito-Costa, S., Gomes, A., Repeated measures study within subjects with randomizations, *WSEAS Transactions on Computers*, Vol. 13, 2025, pp.99-102. https://doi.org/10.37394/232018.2025.13.10.

[13] Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Larochelle, H., Improving reproducibility in machine earning research (A Report from the NeurIPS 2019 Reproducibility Program), *Journal of Machine Learning Research*, Vol. 22, 2021, Article 164, pp. 1–20. https://doi.org/10.48550/arXiv.2003.12206.

[14] Kaufman, S., Rosset, S., Perlich, C., Stitelman, O., Leakage in data mining: Formulation, detection, and avoidance, *ACM Transactions on Knowledge Discovery from Data*, Vol. 6, 2012, pp. 1–21. https://doi.org/10.1145/2382577.2382579.

[15] Gaurav Topre, *Bank Customer Churn Dataset*, Kaggle, 2024, [Online]. https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset (Accessed Date: January 2, 2026).

[16] Gundersen, O. E., Gil, Y., Aha, D. W., On reproducible AI:Towards reproducible research, open science, and digital scholarship in AI publications, *AI Magazine*, Vol. 39, 2018, pp. 56–68. https://doi.org/10.1609/aimag.v39i3.2816.

[17] Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., Herrera, F., Learning from Imbalanced Data Sets, *Springer*, Cham, 2018. https://doi.org/10.1007/978-3-319-98074-4.

[18] Krawczyk, B., Learning from imbalanced data: Open challenges and future directions, Progress in Artificial Intelligence, Vol. 5, 2016, pp. 221–232. https://doi.org/10.1007/s13748-016-0094-0.

[19] Bottou, L., Curtis, F. E., Nocedal, J.,Optimization methods for large-scale machine learning, *SIAM Review*, Vol. 60, 2018, pp. 223–311. https://doi.org/10.1137/16M1080173.

[20] Muhammad Shahid Azeem, Customer Churn Dataset, Kaggle, 2024, [Online]. https://www.kaggle.com/datasets/muhammadshahidazee m/customer-churn-dataset (Accessed Date: January 3, 2026).

[21] Rangala Mahesh, Bank Churn, Kaggle, 2024, [Online]. https://www.kaggle.com/datasets/rangalamahesh/bank-churn (Accessed Date: January 4, 2026).

[22] Severino Silva, *Bank Churn Dataset*, Kaggle Notebook, 2024, [Online]. https://www.kaggle.com/code/severinosilva/bank-churn-dataset (Accessed Date: January 6, 2026).

[23] Dharun4772, *Bank Churn Dataset Classification*, Kaggle Notebook, 2024, [Online]. https://www.kaggle.com/code/dharun4772/bank-churn-dataset-classification (Accessed Date: January 6, 2026).

[24] Python Software Foundation, timeit — Measure execution time of small code snippets, *Python 3 Documentation*, [Online]. https://docs.python.org/3/library/timeit.html. (Accessed Date: January 21, 2026).

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed to the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

The authors would like to thank "Fan S. Noli" University, Korçë, Albania, for funding this research paper.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

APPENDIX

Table 1. Quantitative performance metrics for 5 trained models for the N=2000 evaluation dataset

Evaluation metrics / Model trained with the corresponding algorithm :	Accuracy of the trained model in identifying churner clients (%)	ROC-AUC value	95% CI (AUC)	Recall of the trained model	F1-Score of the trained model	Overall Time (s) needed for the trained model to produce results for the evaluation dataset
Logistic Regression	82%	0.85	[0.84–0.86]	0.72	0.76	1.35
LDA	80%	0.81	[0.80–0.83]	0.71	0.74	1.30
Decision Tree	75%	0.69	[0.65–0.73]	0.65	0.66	2.70
Random Forest	86%	0.85	[0.84–0.86]	0.82	0.84	4.30
Gradient Boosting	88%	0.87	[0.86–0.88]	0.84	0.86	5.10

Source: created by the authors based on the models' evaluation results

Table 2. The ROC-AUC Value of the trained models across all five evaluation datasets.

No. of clients present in the evaluation dataset / Models trained with the algorithm	500	650	1200	1500	2000
Logistic Regression	0.81	0.82	0.83	0.84	0.85
LDA	0.80	0.81	0.82	0.83	0.81
Decision Tree	0.62	0.63	0.64	0.65	0.69
Random Forest	0.81	0.84	0.85	0.86	0.85
Gradient Boosting	0.82	0.83	0.85	0.86	0.87

Source: created by the authors based on the models' evaluation results