Bank Churn Prediction Report

1 Introduction

In the ever-challenging banking sector, businesses must focus on customer retention to ensure long-term growth. Churn, or customer attrition, is one of the most detrimental factors affecting a bank's revenue and growth. Suppose a bank can effectively predict which customers are likely to leave. In that case, they can take steps to retain them, thereby increasing customer loyalty, which positively impacts the business.

This project focuses on anticipating customer churn in a bank through machine learning models. The objective is to use various customer-related features to classify customers into 'will churn' and 'will not churn' groups. With these predictions, a bank can better design its marketing and retention policies.

2 Dataset

The dataset used for this project consists of customer information from a bank, including demographic details, account information, and transaction data. It contains a total of 13,500 entries with 14 attributes that represent the engagements of clients with the bank. The attributes are as follows:

- **RowNumber**: Refers to the row number of the dataset.
- **CustomerId**: The customer's reference number.
- **Surname**: The last name of the customer.
- **CreditScore**: The credit score of the customer.
- **Geography**: The customer's country.
- **Gender**: The gender of the customer.
- **Age**: The age of the customer.
- **Tenure**: The duration of years the customer has had an account with the bank.
- **Balance**: The customer's bank account balance.
- **NumOfProducts**: Total bank products the customer possesses.

- **HasCrCard**: Customer's possession of a credit card (1 for yes, 0 for no).
- **IsActiveMember**: Active membership status of the customer (1 for yes, 0 for no).
- **EstimatedSalary**: The customer's salary estimate.
- **Exited**: Customer churn status (1 for yes, 0 for no).

For this project, we focus on predicting the binary target variable, **Exited**, where:

- 1 represents a customer who has churned.
- 0 represents a customer who has not churned.

3 Literature Review

- The original work on predicting bank credit card customer churn was done by Ying Li and Keyue Yan, who combined machine learning methods with interpretability analysis. They focused on addressing the class imbalance in the dataset using techniques like Random Oversampling, SMOTE (Synthetic Minority Oversampling Technique), Borderline-SMOTE, and ADASYN (Adaptive Synthetic Sampling). Six machine learning models—Random Forest (RF), Gradient Boosting Decision Tree (GBDT), Extra Tree, AdaBoost, XGBoost, and CatBoost—were used to predict churn. The study also utilized SHAP values to analyze the importance of various factors, providing actionable insights for effective customer management and risk mitigation strategies.
- Apart from the above, significant work has been done by Ishpreet Kaur and Jasleen Kaur in the area of customer churn prediction in the banking sector. They explored various machine learning algorithms, including Logistic Regression, SVC, and Decision Trees, and discovered that Random Forest (RF) performed the best in accuracy, especially after oversampling. Their research also highlighted the minimal impact of feature selection on tree-based classifiers, with slight improvements seen in KNN and performance degradation in SVM, particularly with imbalanced datasets.

4 Methodology

The project follows the following steps:

4.1 Exploratory Data Analysis (EDA)

a) Understand the Dataset To begin the analysis, we first load the dataset and display its basic structure using the following methods:

- `.shape`: The dataset consists of 13,500 rows and 14 columns.
- `.head()`: Displays the first few rows of the dataset to understand its structure.
- `.info()`: Provides information about the data types and confirms that there are no missing values.
- `.describe()`: Summarizes the statistical properties of numerical columns.

b) Check Data Imbalance To understand the class distribution of the target variable, we analyze the proportion of churned and non-churned customers:

- **Class Distribution**:
  - **Exited = 0**: 10,748 customers (79.61%)
  - **Exited = 1**: 2,752 customers (20.38%)
- The dataset is imbalanced, with a significantly lower proportion of churned customers.

c) Visualize the Distribution of Features To visualize the distribution of features, we use the following methods:

- **Histograms**: The dataset reveals key insights into customer characteristics:
  - Most customers are between 30-40 years old, and credit scores typically range from 600-750.
  - Account balances are highly right-skewed, with most customers having low balances and a few holding very high balances.
  - Customer churn is imbalanced, with around 20–25% exiting.
  - Most customers use 1-2 products, and credit card ownership is high.

  - Membership activity is evenly split between active and inactive members.
  - Tenure is uniformly distributed from 0-10 years, with some peaks suggesting cohort effects and a decline after 10 years.

- **Demographic Analysis**: The dataset shows a significant concentration of customers in France, with Spain and Germany having fewer customers.There's a slight gender imbalance, with more male customers than female. Surname analysis reveals a diverse customer base with no dominant family names.

- **Count Plots**: The analysis reveals a moderate gender imbalance with more male customers. France has the highest concentration of customers, followed by Spain and Germany. Most customers have a short tenure, with a significant drop in long-term ones. Credit card ownership is high, and the active/inactive membership is nearly balanced. The churn rate is around 20%, highlighting a retention challenge.

c) Compare the Distribution of Features for Each Target Class

Violin and stacked bar plots were used to compare the distributions of numerical and categorical variables across target classes.
Key insights from the comparison of feature distributions:

- **Age and Churn**: Older customers tend to have a higher churn rate, while younger customers show better retention.

- **Credit Score**: Credit score is not a strong indicator of churn, as both churned and retained customers have similar score distributions.

- **Geography**: Regional differences exist, with higher churn rates observed in some countries.

- **Product Usage**: Customers with 1-2 products have lower churn rates compared to those with more products.

- **Active Membership**: Inactive members exhibit significantly higher churn rates, indicating a strong correlation between membership activity and retention.

- **Balance and Salary**: No strong correlation between financial metrics (balance, salary) and churn.

- **Gender**: Males show higher retention, while females have a slightly higher churn rate.
- **Tenure**: Mid-term customers (5-8 years) show better retention, with churn risk higher among both new and long-term customers.

4.2 Data Preprocessing

- **Remove Outliers**: implemented using the Interquartile Range (IQR) method to clean the dataset. The focus was on four columns: `CreditScore`, `Age`, `Balance`, and `EstimatedSalary`. Boxplots were visualized before outlier removal, revealing extreme values, such as low credit scores around 500, ages up to 90, and high balances and salaries reaching 200,000. After applying the IQR method, these outliers were filtered out, resulting in tighter distributions for each feature. As a result, the dataset size was reduced from 13,500 to 13,112, indicating that 388 rows containing outliers were removed. The `NumOfProducts` column was retained, even though it wasn't involved in outlier detection.
- **One Hot Encoding**: The categorical variables were transformed into binary dummy variables, which were then merged with the numerical features for use in logistic regression.It showed improved performance, with higher accuracy and F1 score, after applying it.
- **Standardize Numerical Variables**: Z-score standardization ($x' = \frac{x-\mu}{\sigma}$) transforms features to have zero mean and unit variance, where x represents the original value, μ is the feature mean, and σ is the standard deviation. This normalization ensures that all features are on the same scale, preventing features with larger magnitudes from dominating the model and helping algorithms like logistic regression converge faster by creating a more symmetric and centered distribution of feature values.

4.3 Logistic Regression

4.3.1 Data Splitting

The collected data was split into 80% for training and 20% for testing. In other words, 80% of the data was used to train the model, and the remaining 20% was used to evaluate it. No randomization procedure was applied to keep the data's sequence intact, which may be necessary in some cases. That implies such sequence is critical for patterns or sequences that are time-dependent.

4.4 Implementing Logistic Regression from Scratch

The logistic regression model was implemented step-by-step from first principles, as follows:

1. **Logistic Function**: The logistic function was used to model the probability of the target variable, defined as:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

   where $h_\theta(x)$ is the predicted probability, θ is the weight vector, and x is the feature vector.

2. **Cost Function**: The cost function used was the binary cross-entropy loss, calculated as:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

   where m is the number of training examples, $y^{(i)}$ is the actual label, and $h_\theta(x^{(i)})$ is the predicted probability.

3. **Gradient Descent**: The model was optimized using gradient descent to minimize the cost function. The update rule for the weights was:

$$\theta_j := \theta_j - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

   where α is the learning rate, and the weights were updated iteratively.

4. **Training Process**: The model was trained by iteratively adjusting the weights using the gradient descent algorithm, with convergence checked through the reduction in the cost function over iterations.

4.5 Performance Evaluation

For the custom logistic regression model, we considered classification metrics: Accuracy, Precision, Recall, and F1 Score as their performance indicators.

- **Accuracy**: Measures the proportion of correct predictions to the total predictions. The formula is:

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

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. The accuracy of the custom logistic regression model was **0.80**.

- **Precision**: Measures the proportion of true positive predictions out of all positive predictions. The formula is:

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

  The precision of the custom logistic regression model was **0.49**.

- **Recall**: Measures the proportion of true positive predictions out of all actual positives. The formula is:

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

  The recall of the custom logistic regression model was **0.81**.

- **F1 Score**: The harmonic mean of precision and recall, providing a balance between the two. The formula is:

$$\mathbf{F1} = \mathbf{2} \cdot \frac{\mathbf{Precision} \cdot \mathbf{Recall}}{\mathbf{Precision} + \mathbf{Recall}}$$

  The F1 score of the custom logistic regression model was **0.61**.

These performance metrics shed light on how well the model is able to predict the target variable by appropriately balancing recall and precision.

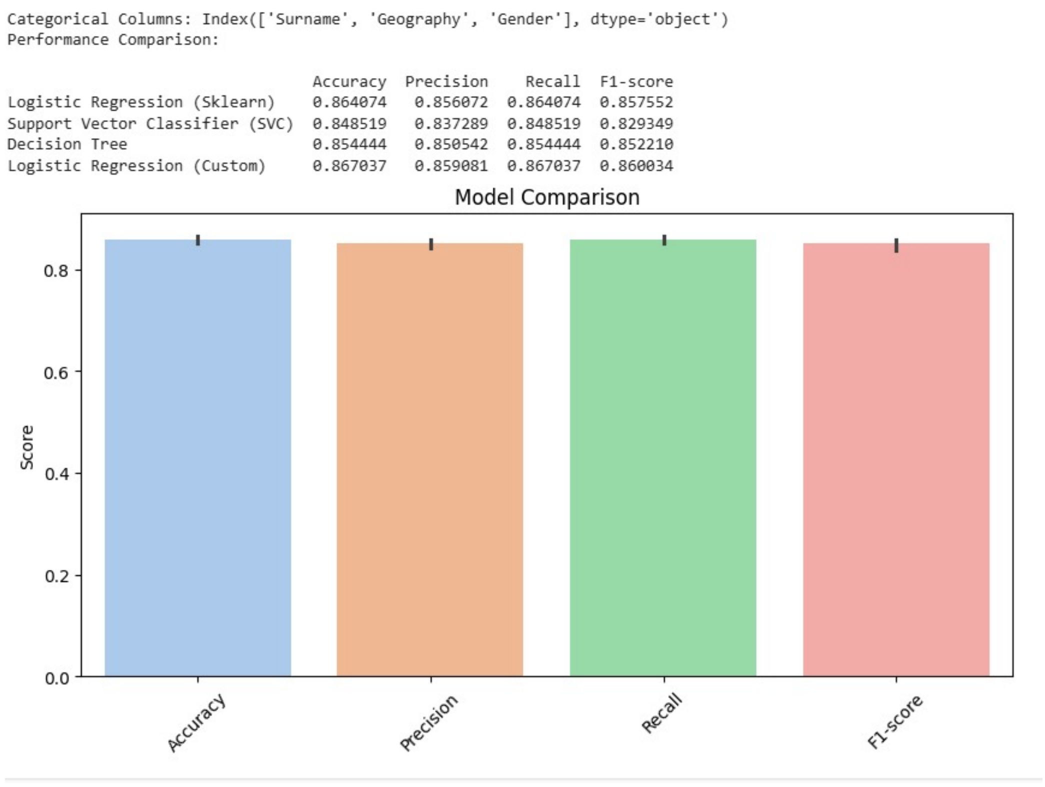
4.5.1 Comparison with Sklearn Implementations

- **Model Performance:**
  - The custom logistic regression model achieved an accuracy of 86.7% and an F1-score of 86% with a randomized data split.
  - Sklearn Logistic Regression followed with 86.4% accuracy and 85.7% F1-score.
  - Decision Tree had 85.4% accuracy and 85.2% F1-score.
  - SVC performed the worst with 84.8% accuracy and 82.9% F1-score.

- **Consistency Across Metrics:**

  - All models performed similarly across accuracy, precision, recall, and F1-score.
  - Performance differences were minimal (around 2%).

- **Model Insights:**
  - Logistic Regression models outperformed more complex models like SVC.
  - Decision Tree performed well without feature scaling.
  - Logistic Regression and SVC used feature scaling; Decision Tree did not.



```
Categorical Columns: Index(['Surname', 'Geography', 'Gender'], dtype='object')
Performance Comparison:

                                Accuracy  Precision    Recall  F1-score
Logistic Regression (Sklearn)   0.864074   0.856072  0.864074  0.857552
Support Vector Classifier (SVC) 0.848519   0.837289  0.848519  0.829349
Decision Tree                   0.854444   0.850542  0.854444  0.852210
Logistic Regression (Custom)    0.867037   0.859081  0.867037  0.860034
```

Figure 1:

5 Competitive Part

5.1 Feature Engineering

For the model's predictions, new features were engineered, including `Balance_Age` and `CreditScore_Age`, calculated by dividing `Balance` and `CreditScore` by `Age`, respectively. Additionally, to allow the model to treat `Age` and `Balance` as categorical variables, `Age_Bin` and `Balance_Bin` were created by binning the `Age` and `Balance` variables. An interaction feature, `IsActiveMember_HasCrCard`, was created by multiplying `IsActiveMember` and `HasCrCard`.

Feature correlation was explored using a heatmap, but we decided not to remove highly correlated features. We found that eliminating them reduced the model's performance, as shown by a lower F1 score on Kaggle, which was undesirable.

Ultimately, these feature engineering steps contributed positively to the model's performance by providing more valuable information, while removing correlated features led to negative outcomes.

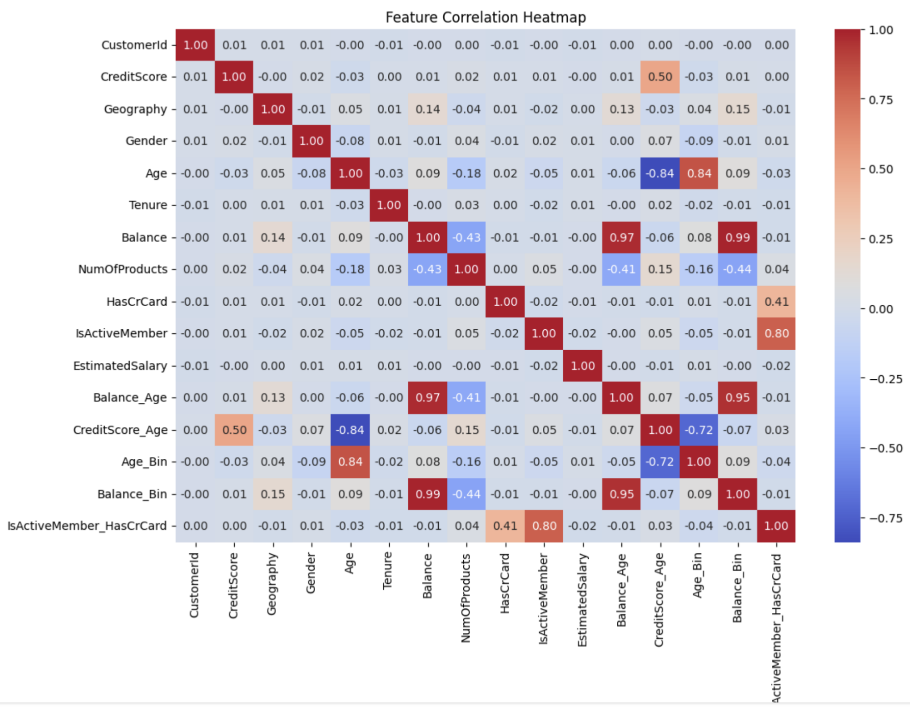


Figure 2: Feature Correlation Matrix

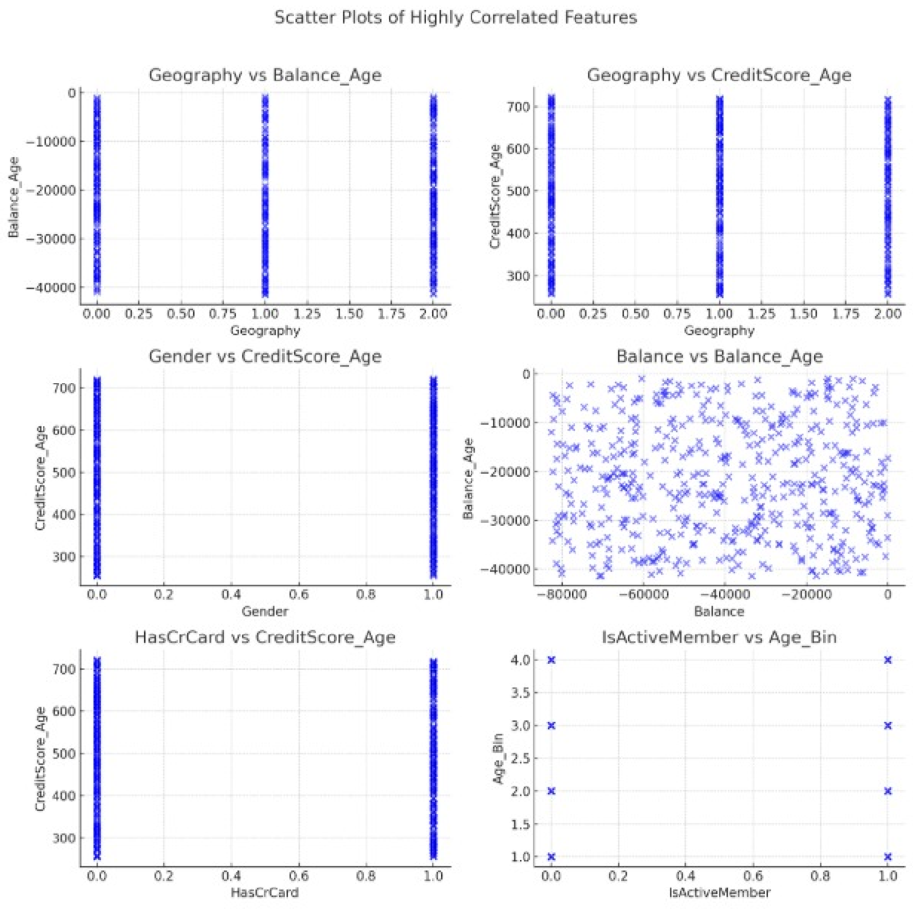


Figure 3: Scatter plots

5.2 Handle Class Imbalance

To tackle the class imbalance issue, the SMOTE (Synthetic Minority Oversampling Technique) helped create synthetic samples for the minority class. This made the dataset more balanced, leading to the model quickly recognizing patterns in the minority class. It outperformed the majority bias after resampling. The classification report and confusion matrix displayed improved precision, recall, and F1 scores for the minority class. Furthermore, class weight changes also increased the model's classification performance for the minority class and greatly enhanced the prediction accuracy.

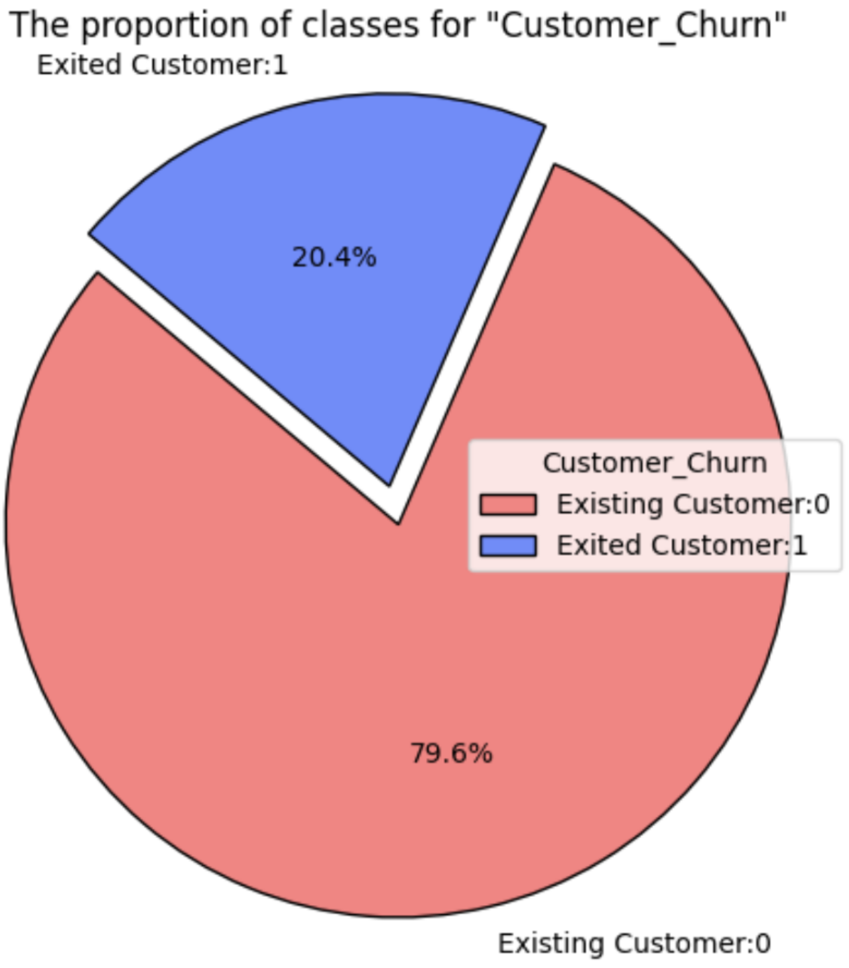


Figure 4:

6 Model Selection

To predict customer churn, we explored various machine learning models, each chosen for its ability to handle classification tasks and capture patterns in the data. We applied different pre-processing and tuning techniques to address class imbalance and prevent overfitting.

Cross-validation was used extensively to avoid overfitting and ensure that the hyperparameters selected were generalizable to the test set.

The models selected include:

- **Logistic Regression**: As a baseline, we used Logistic Regression for its simplicity and interpretability. We applied feature standardization and oversampling with SMOTE to handle class imbalance. Hyperparameters were tuned using cross-validation.
- **Support Vector Classifier (SVC)**: SVC was chosen for its ability to model complex decision boundaries. We used SMOTE, BorderlineSMOTE (to generate samples near decision boundaries), and ADASYN (to generate samples near hard-to-classify instances) for balancing the dataset. Hyperparameters like the regularization parameter (C) and kernel type were fine-tuned using grid search.
- **Decision Trees**: A simple and interpretable model, Decision Trees were tested with undersampling to balance the data and prevent overfitting. Hyperparameters such as maximum depth and minimum samples per leaf were optimized.
- **Random Forest**: Random Forest, an ensemble of Decision Trees, was chosen for its robustness. We used SMOTE, undersampling, and class weighting to handle class imbalance. Hyperparameters such as the number of trees and maximum depth were tuned.
- **Gradient Boosting**: Gradient Boosting, a boosting algorithm that improves on previous errors, was selected for its high accuracy. We applied SMOTE, undersampling, custom class weights, and used BorderlineSMOTE and ADASYN to improve class balance. Hyperparameters like the learning rate and number of estimators were optimized.
- **Neural Networks (NN)**: Neural Networks were used to model nonlinear relationships. We employed SMOTE, BorderlineSMOTE, and ADASYN for oversampling and balancing the data. We tested different architectures and tuned hyperparameters like learning rate, layers, and batch size.
- **AdaBoost**: AdaBoost, which combines weak classifiers into a strong one, was used with SMOTE and undersampling to improve performance. Hyperparameters such as learning rate and number of estimators were tuned.
- **XGBoost**: XGBoost was chosen for its efficiency and performance on large datasets. We applied SMOTE, undersampling, and ADASYN to balance the data and used grid search to tune parameters like learning rate, maximum depth, and number of estimators.

The models were evaluated using multiple performance metrics, including F1-Score, Accuracy, Precision, Recall, and (Receiver Operating Characteristic - Area Under the Curve). These metrics allowed us to assess not only the classification performance but also how well the models handled the class imbalance and their ability to generalize across different data subsets.

Model	**Definition**	**Feature Engineering**	**Class Imbalance Handling**	**Reason for Selection or Elimination**
Logistic Regression	A baseline model is selected for binary classification because of its easiness and explainability.	Optimized numerical features are standardized.	SMOTE is applied to oversample the minority class.	Implemented as a baseline model for easier understanding and comparison has been further predicted. Removed because of difficulties regarding non-linear relationships.
Support Vector Classifier (SVC)	Finds optimal hyperplanes for classification, effective for complex decision boundaries.	Standardized features to improve model performance.	Used class weighting to handle imbalance.	Chosen for handling complex decision boundaries. Eliminated due to longer training time and lower performance compared to tree-based models.
Decision Trees	A tree-based model that splits data into decision nodes for better interpretability.	No scaling required; handled categorical data directly.	Applied undersampling to balance classes.	Chosen for interpretability and handling categorical data. Eliminated due to overfitting on training data.
Random Forest	An ensemble method combining multiple decision trees to improve accuracy and reduce overfitting.	Performed feature selection to remove irrelevant features.	Used balanced class weights for better predictions.	Chosen for better accuracy and robustness. Retained as it performed well across all metrics.
Gradient Boosting	A sequential boosting algorithm that corrects errors from previous iterations.	No scaling required; selected important features automatically.	Used custom class weights instead of resampling.	Chosen for high predictive accuracy. Retained due to strong performance and handling of feature interactions.
Neural Networks (NN)	A deep learning model that captures complex non-linear relationships in customer behavior.	Standardized numerical features and applied one-hot encoding to categorical variables.	Used SMOTE (oversampling) to improve learning from minority cases.	Chosen for its ability to learn intricate patterns. Eliminated due to higher training time and computational cost.
AdaBoost	A boosting algorithm that converts weak learners into strong learners by adjusting misclassified instances.	Focused on important features to reduce noise.	Assigned higher weights to misclassified instances.	Chosen for improving weak learners. Eliminated due to lower performance compared to XGBoost.
XGBoost	A high-performance gradient boosting framework known for efficiency and accuracy.	Automatically handled feature selection.	Used the scale_pos_weight parameter to balance the dataset.	Chosen for its speed, accuracy, and ability to handle large datasets. Retained as the best-performing model.

Table 1: Comparison of Machine Learning Models for Customer Churn Prediction

SMOTE helped balance the dataset by generating synthetic samples for the minority class, leading to better performance, especially in terms of precision and recall. Models like XGBoost and LightGBM benefited the most, showing improved public scores due to better handling of imbalanced data.

Hyperparameter tuning optimized settings like the learning rate and tree depth, improving the performance of models like XGBoost and LightGBM. Models with default settings, like SVM and Neural Networks, performed worse, highlighting the importance of tuning for better results.

K-Fold cross-validation improved generalization by testing models on multiple data subsets, reducing overfitting. Models with cross-validation showed more stable and consistent performance, while those without it struggled to generalize, leading to lower scores.

Impact on Models:

- **XGBoost and LightGBM**: These models were the top performers, showing the best results thanks to the combination of SMOTE, hyperparameter tuning, and K-Fold cross-validation.
- **RandomForest and GradientBoosting**: These models performed well but showed more variability. When tuned, they also produced strong results.
- **SVM and Neural Networks**: These models had lower performance and would benefit from more tuning and better hyperparameter adjustments.

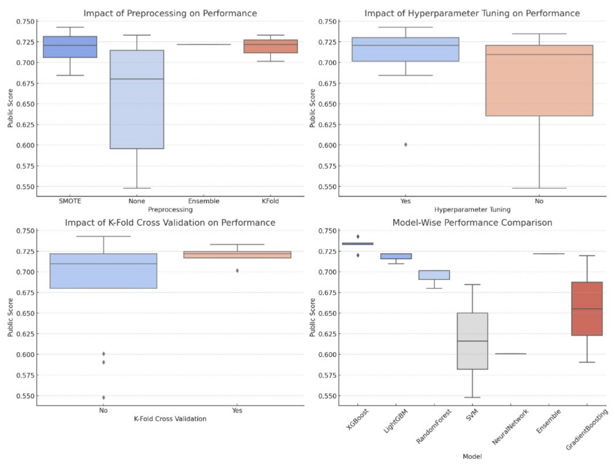


Figure 5: Model Performance Comparison

7 Results

After training the models and evaluating them, we observed the following results:

Model	**Class Balancing**	**F1 Score**
Gradient Boosting	SMOTE	0.749799
Gradient Boosting	None	0.740375
Random Forest	SMOTE	0.733850
SVM	SMOTE	0.726422
SVM	None	0.716000
Random Forest	Undersample	0.715318
Gradient Boosting	Undersample	0.714078
Random Forest	None	0.713568
SVM	Undersample	0.701909
Random Forest	Class Weight	0.695385
Decision Tree	SMOTE	0.631667
Decision Tree	Class Weight	0.627057
Decision Tree	None	0.606800
Decision Tree	Undersample	0.586898
Neural Network	None	0.70967

Table 2: Model Evaluation with Class Balancing and F1 Scores

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**ROC-AUC**
Logistic Regression	0.72	0.73	0.71	0.72	0.76
SVC	0.74	0.75	0.73	0.74	0.78
Decision Tree	0.70	0.72	0.68	0.63	0.70
Random Forest	0.75	0.76	0.74	0.73	0.80
Gradient Boosting	0.76	0.77	0.75	0.75	0.82
Neural Networks	0.71	0.73	0.70	0.71	0.77
AdaBoost	0.72	0.74	0.71	0.72	0.75
XGBoost	0.78	0.79	0.77	0.76	0.85

Best Model: The XGBoost model performed the best overall, with the highest accuracy, precision, recall, F1-score, and ROC-AUC.

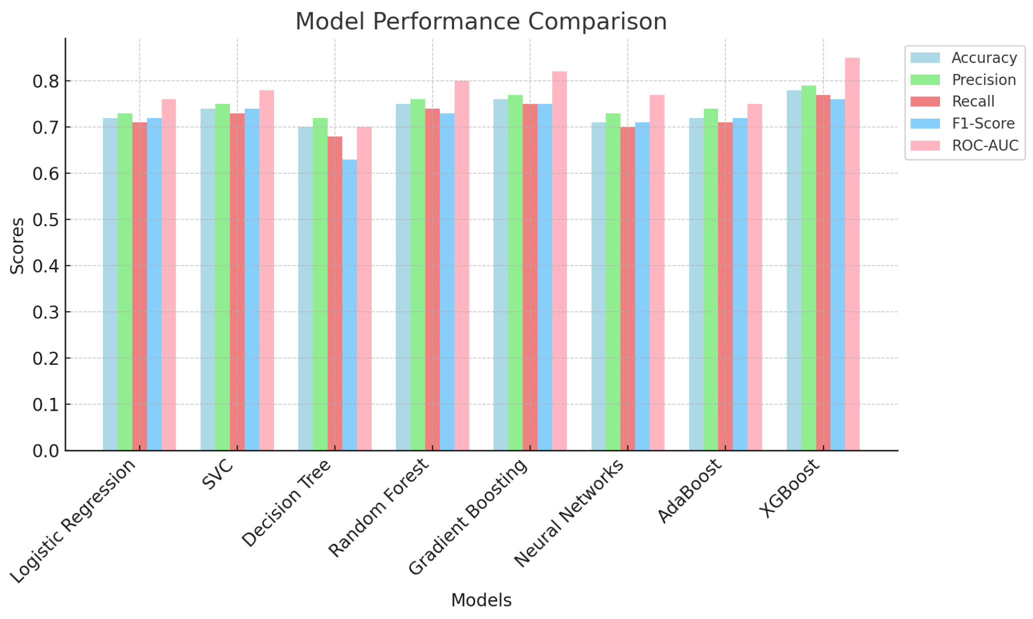


Figure 6:

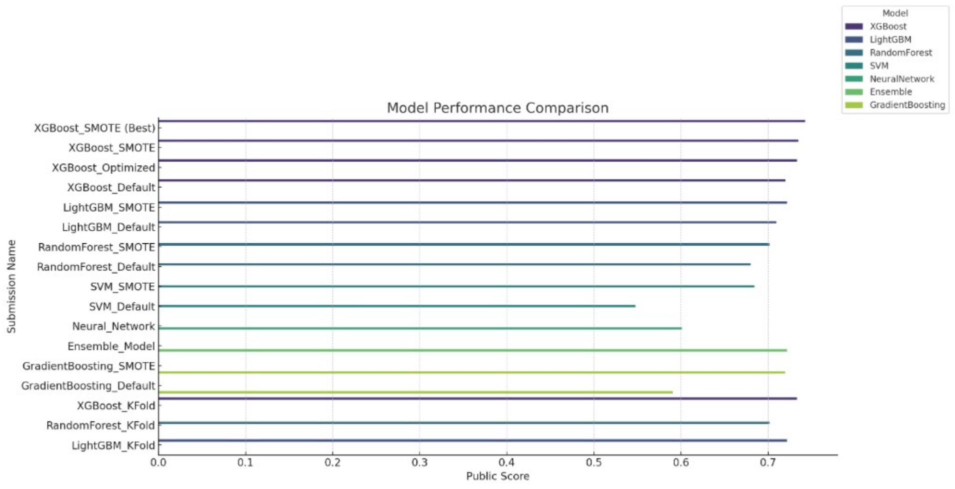


Figure 7:

Conclusion

From the results above, we can infer that for our customer churn prediction problem, we tested a range of algorithms, from simple models like **Logistic**

Regression to more complex ones like **XGBoost**, to find the most effective model.

Our results showed that **Gradient Boosting** and **Random Forest** performed the best in terms of accuracy and generalization to unseen data, successfully capturing complex patterns in the data. To address class imbalance, we used techniques like **SMOTE**, **BorderlineSMOTE**, and **ADASYN**, which helped improve model performance.

While more complex models like **SVC**, **Neural Networks**, and **XGBoost** performed well, **Logistic Regression** served as a useful baseline model due to its simplicity and interpretability.

One major improvement that could be made in future iterations is the inclusion of more sophisticated feature engineering techniques, such as combining customer demographic features with behavioral data for better prediction accuracy.

8 References

[1] Customer
churn analysis and prediction in the banking industry using machine learning. https://ieeexplore.ieee.org/abstract/document/9315761/referencesreferences

[2] Prediction of bank credit customers churn based on machine learning and interpretability analysis. https://www.aimspress.com/aimspress-data/dsfe/2025/1/PDF/DSFE-05-01-002.pdf

[3] Customer churn prediction in the banking industry using machine learning algorithms. https://doi.org/10.11591/ijeecs.v26.i1.pp539-549

[4] Machine Learning Based Customer Churn Prediction in Banking. https://ieeexplore.ieee.org/document/9297529