Content

Problem

1

Unnecessary retention programs

2

New marketing and acquisition costs

The main objective is to maximize the bank's revenue by minimizing the **unnecessary retention programs** and by **minimizing the marketing and acquisition costs** for new customers by improving the churn model prediction

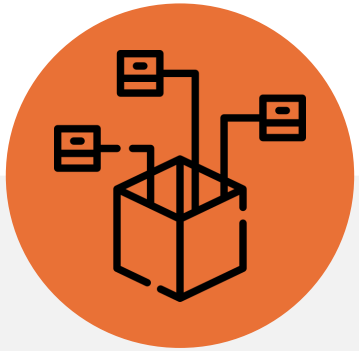
Solution

Data Preparation

Collect and Clean Data: Assemble historical customer data and clean for quality

Feature Engineering: Develop features indicative of churn, such as account activity patterns

Data Labeling: Define and apply labels for 'churned' or 'retained' status



Model Deployment

Algorithm Selection: Choose predictive algorithms suited to the data and problem

Model Training: Train the model on historical data, validating with a separate test and validation set

Performance Evaluation: Assess the model using metrics like F1 score and ROC-AUC

Implementation & Monitoring

Deployment: Integrate the model into the bank's operational systems

Monitoring: Continuously monitor the model's performance and update as needed

Refinement: Use model insights to refine and target retention strategies effectively

Bank Customer Churn Dataset

Target variable: churn
1 if the client has left the bank during some period or 0 if he/she has not

Customer ID: a unique identifier for each customer

Numerical variables	
Credit score	From 350 to 850
Age	From 18 to 92
Tenure	From 0 to 10
Balance	Max: 251k
Product numbers	From 1 to 4
Estimated salary	Max: 200k

Categorical variables	
Country	Spain, France, Germany
Gender	55% male, 45% female
Credit card	70,55% have a credit card
Active membership	51,51% are active members

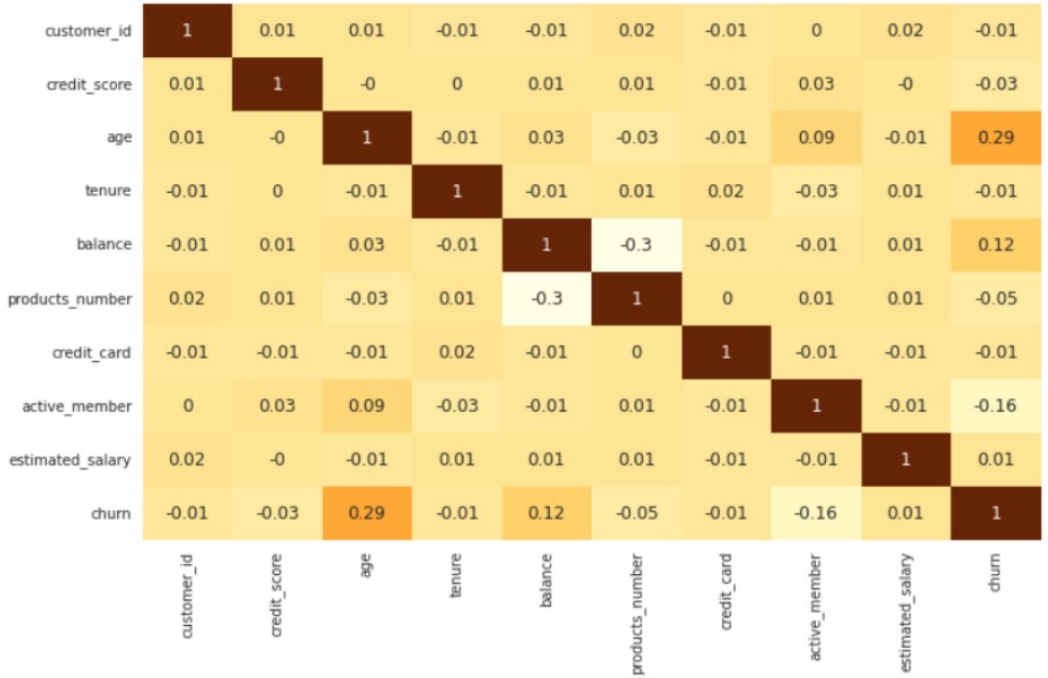
Most of the variables have bell-shaped distribution, except for tenure and estimated salary, which are more uniform-like

Data Cleaning

There are no **missing values**, so no imputation methods are required.

Customer_id is irrelevant for churn prediction, indicating it will be **removed** before model training.

The correlation matrix indicates a lack of significant correlations between features, negating the need for feature selection or dimensionality reduction.

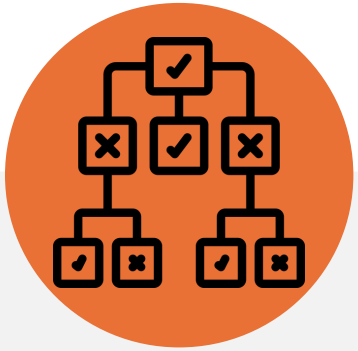


Model Approach

For the ML model a **XGBoost** model was implemented

XGBoost is an **advanced implementation of gradient boosted decision trees** designed for speed and performance

It utilizes **regularized boosting and advanced tree learning** algorithms to improve accuracy and prevent overfitting



Why XGBoost?

Superior Results
Prevents overfitting, ensuring reliable churn predictions

Speed and Scale
Efficiently processes large customer datasets

Insightful Analytics
Offers clear feature importance for targeted strategies

Model Deployment

Load the data into a python notebook

Pre-process the data

Implement the XGBoost model

Upload the Notebook into AWS SageMaker

Deployment

Train the model in AWS SageMaker

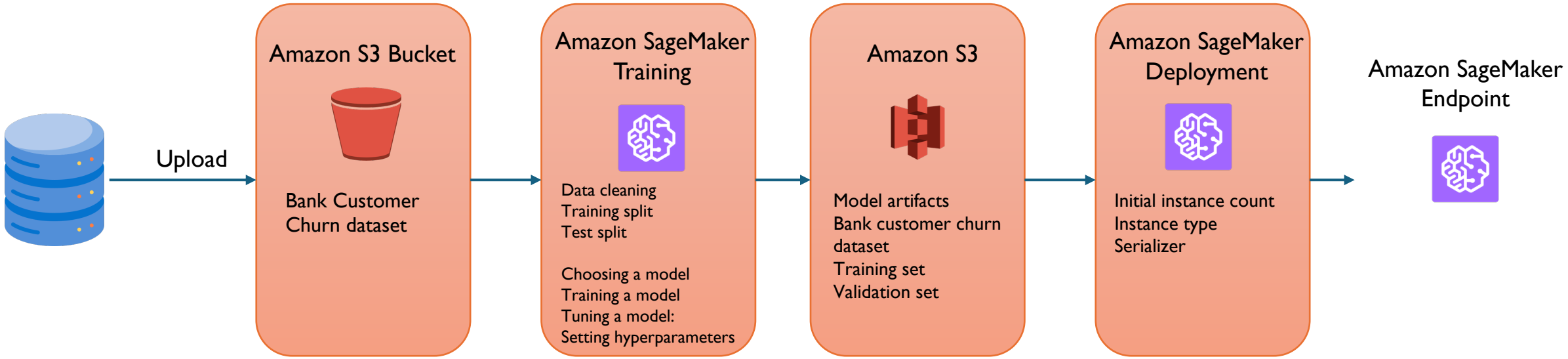
Evaluate and check the metrics of the trained model

Configure and launch a hyperparameter tuning job

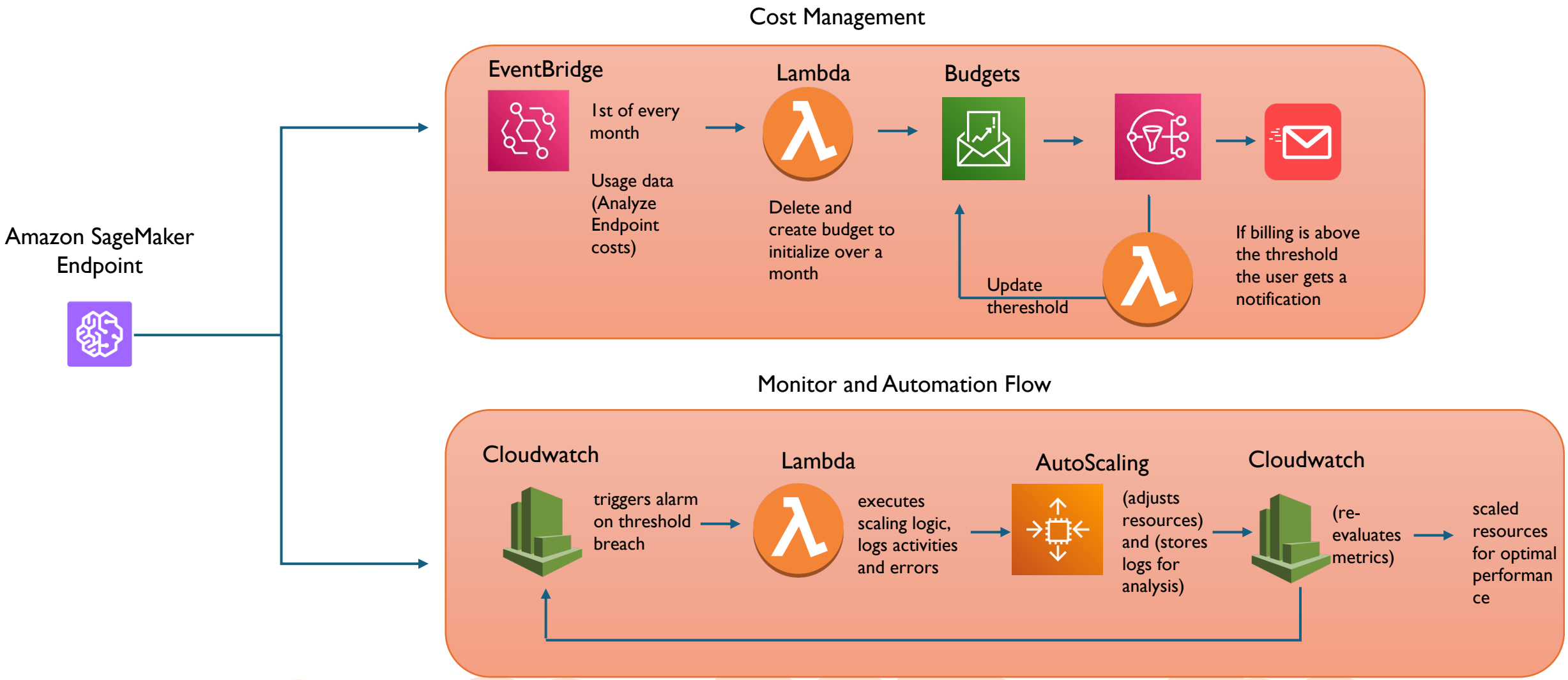
Find the optimal hyperparameter values that maximize the area under the ROC curve (AUC) on the validation dataset

Evaluate the results of the tuned model and deploy it

Architecture Diagram



Next Steps


Cost Management
EventBridge
1st of every month
Usage data (Analyze Endpoint costs)
Lambda
Delete and create budget to initialize over a month
Budgets
Update thereshold
If billing is above the threshold the user gets a notification
Amazon SageMaker Endpoint
Monitor and Automation Flow
Cloudwatch
triggers alarm on threshold breach
Lambda
executes scaling logic, logs activities and errors
AutoScaling
(adjusts resources) and (stores logs for analysis)
Cloudwatch
(re-evaluates metrics)
scaled resources for optimal performance

Results & Evaluation

Train AUC: 0.8983
Validation AUC: 0.8518

Define Key & fixed Hyperparameters
And the hyperparameter range

Train AUC: 0.9084
Validation AUC: 0.8537

Accuracy: High accuracy indicated that the model effectively classifies churn and non-churn cases — **84%**

Precision: High precision indicates the model is reliable in its churn predictions — **72%**

Recall: There's potential to improve the model's detection of actual churn cases — **41%**

F1 Score: Suggests enhancing recall is necessary for better churn identification — **52%**

Insights:

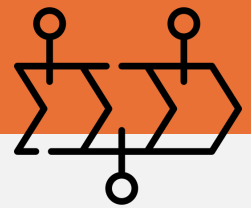
- **Improved accuracy** through effective hyperparameter exploration
- Majority of predicted probabilities are **below the 0.5 threshold**, suggesting low churn likelihood
- The model shows **robust overall performance** but has room for improvement in sensitivity to churn

Conclusion

Churn Model Impact
Implemented a predictive churn model using XGBoost that achieved solid rate of effectively identifying customers at risk of leaving the bank

Data Utilization
Leveraged comprehensive data preparation and model training to highlight the most influential factors affecting customer churn

Efficiency Gain
Streamlined retention programs and reduced unnecessary costs, resulting in a more targeted and financially efficient marketing strategy



Next Steps

Enhanced Automation
Deploy a DynamoDB or RDS in combination with a lambda function to make the database modification more accessible

Model Iteration
Experiment with additional hyperparameter tuning, ensemble methods, or alternative algorithms

Customer Engagement
Integrate the model's insights with customer engagement strategies to proactively address factors contributing to churn

Thank you!