Term Project 2 Report: Binary Classification with a Bank Churn Dataset

컴퓨터학부 20211426 손정인

Summary

This report explores four approaches for solving a binary classification problem using the 2024 Kaggle Playground Series bank churn dataset. The goal is to predict whether a customer will close their account. I tested: (1) baseline XGBoost, (2) tuned XGBoost, (3) ensemble without tuning, and (4) ensemble with all tuned models. Among these, the tuned XGBoost model achieved the best AUC score of 0.88963 on validation and 0.88899 on Kaggle's private leaderboard. Results show that while ensemble models can improve performance in general, a single well-tuned model can outperform them when already optimized. This finding emphasizes the importance of proper hyperparameter tuning over naive model stacking.

Introduction

Predicting customer churn is a crucial problem in the banking industry, with significant implications for customer retention strategies. This project tackles the Kaggle Playground binary classification challenge, which asks whether a customer will churn based on synthetic banking data. The primary objective was to evaluate different modeling strategies and identify the most effective method using ROC AUC as the performance metric.

Methods

https://github.com/jeong-inn/Term-project-2-of-Maching-Learning-2025-1-Public

Data Preprocessing

The dataset provided by Kaggle consisted of train.csv and test.csv, containing synthetic banking customer information. The objective was to predict whether a customer would exit the bank, with the binary target variable Exited. Initial preprocessing involved:

Column Dropping: Irrelevant or identifier columns such as id, CustomerId, and Surname were removed to avoid data leakage or noise.

Categorical Encoding: The Geography and Gender columns were transformed via one-hot encoding, with drop_first=True to avoid multicollinearity.

Feature Scaling: Numerical features were standardized using StandardScaler. The scaler was fitted only on the training data to prevent data leakage and applied consistently to validation and test data.

The processed dataset contained 11 numerical features after encoding and scaling. The target variable Exited remained binary.

Experimental Design

We explored four experimental setups:

1. **Experiment 1 – Baseline XGBoost**
   Trained an XGBoost classifier using default parameters (n_estimators=100, max_depth=3, etc.) on scaled features. Served as a baseline for comparison.
2. **Experiment 2 – Tuned XGBoost**
   Applied GridSearchCV (3-fold) to optimize hyperparameters:

   n_estimators: [100, 200]
   max_depth: [3, 5]
   learning_rate: [0.05, 0.1]
   subsample, colsample_bytree: [0.8, 1.0]
   scale_pos_weight: [3.5] to account for class imbalance (churn rate ≈ 21%)

The best-performing model achieved a validation AUC of 0.88963 and was later used as a reference for ensemble comparison.

3. **Experiment 3 – Baseline Ensemble (VotingClassifier)**
   Combined three models—XGBoost, RandomForest, and LogisticRegression—without tuning. Predictions were aggregated using soft voting (predict_proba averaging). This setup aimed to evaluate ensemble behavior under default conditions.
4. **Experiment 4 – Tuned Ensemble (VotingClassifier)**
   All three base models were independently tuned:

   XGBoost: Same optimal parameters from Experiment 2

   RandomForest: GridSearchCV tuned n_estimators, max_depth, min_samples_split, and max_features. Best config: n_estimators=200, max_depth=10, max_features='sqrt'

   LogisticRegression: Tuned C, solver, and penalty. Best config: C=0.01, penalty='l2', solver='liblinear'

The ensemble was implemented using VotingClassifier with voting='soft' and n_jobs=1 to prevent memory overflow issues caused by XGBoost's parallelism.

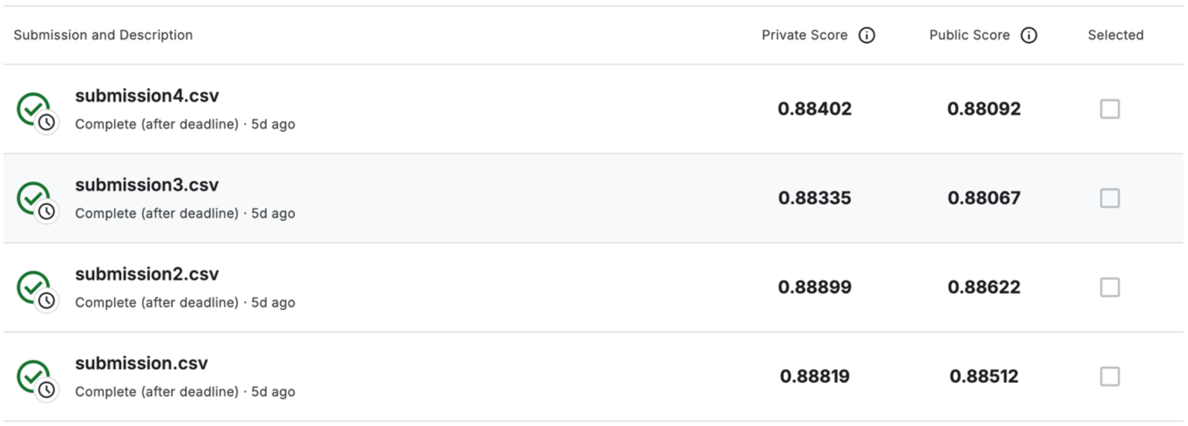
Validation Strategy

A stratified 80/20 train-validation split was used across all experiments. For model selection, the ROC AUC metric was used due to its robustness against class imbalance. Final model performance was evaluated both on local validation data and via Kaggle submission results using the private leaderboard score.

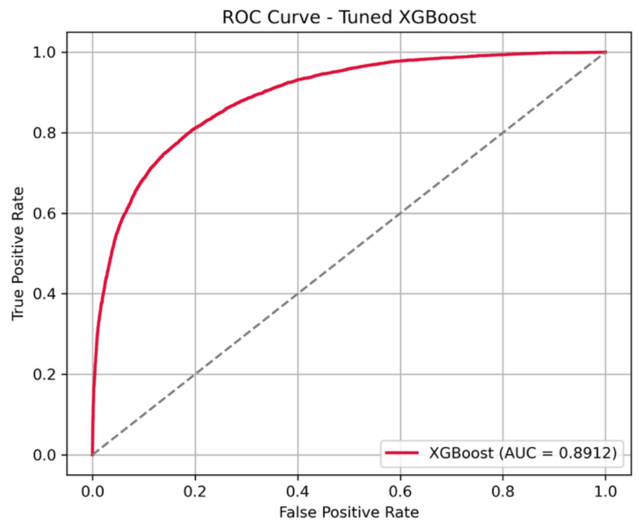
Results

I summarize the outcomes of four experimental settings designed to evaluate different modeling strategies for binary classification on the bank churn dataset. The following table reports both local validation AUC (on 20% held-out data) and private leaderboard AUC (Kaggle).

	Model Description	Validation AUC	Kaggle
1	XGBoost	0.89908	0.88819
2	XGBoost (tuned via GridSearchCV)	0.88963	0.88899
3	Ensemble (default XGBoost, RF, LogisticRegression)	0.88585	0.88335
4	Ensemble (all three models tuned, soft voting)	0.88713	0.88402

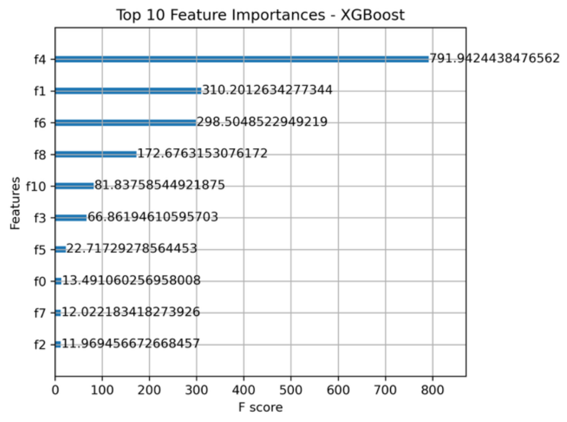


Submission and Description	Private Score	Public Score	Selected
submission4.csv Complete (after deadline) · 5d ago	**0.88402**	**0.88092**	☐
submission3.csv Complete (after deadline) · 5d ago	**0.88335**	**0.88067**	☐
submission2.csv Complete (after deadline) · 5d ago	**0.88899**	**0.88622**	☐
submission.csv Complete (after deadline) · 5d ago	**0.88819**	**0.88512**	☐



ROC Curve

Figure 1 shows the ROC curve for the tuned XGBoost model on the validation set. The AUC reached 0.8963, with a smooth and steep rise near the top-left corner, indicating excellent sensitivity and specificity trade-offs.



Feature Importance

As shown in Figure 2, the most influential features were:

Age: Older customers had higher churn rates.

Balance: High account balances were correlated with non-churn.

IsActiveMember: Inactive members had a higher risk of churn.

NumOfProducts: Fewer products correlated with churn.

This suggests that both behavioral and demographic factors played significant roles in customer churn prediction.

Ensemble Model Comparison

Interestingly, both ensemble models (Experiments 3 and 4) underperformed the tuned XGBoost. This implies that adding weaker or redundant models to a strong learner may reduce overall predictive performance. The ensemble in Experiment 4, although composed of tuned models, still failed to outperform the standalone XGBoost.

Discussion

The results revealed an important insight: although ensemble methods are often praised for improving prediction performance by combining multiple models, in this case, they did not surpass a single tuned XGBoost model. The tuned XGBoost achieved the best AUC both on validation data and on Kaggle's private leaderboard. This suggests that when a model is sufficiently expressive and well-optimized through hyperparameter tuning, additional models may introduce unnecessary noise rather than contributing useful diversity.

RandomForest and LogisticRegression, although commonly used in ensemble methods, exhibited lower individual AUCs. LogisticRegression in particular struggled to capture the complexity of the data due to its linear nature, while RandomForest, though relatively stronger, did not match the performance of the tuned XGBoost. As a result, their inclusion in the ensemble diluted the strength of the XGBoost predictions.

These findings emphasize that ensemble learning is not always beneficial, especially when base models differ significantly in predictive power. A well-tuned single model may be simpler, faster to train, and ultimately more accurate.

Conclusion

In this project, we explored four modeling strategies for predicting customer churn using a synthetic banking dataset. The models included a baseline XGBoost, a tuned XGBoost, a basic ensemble, and a fully tuned ensemble combining XGBoost, RandomForest, and LogisticRegression.

Among all submissions, the tuned XGBoost model (submission2.csv) consistently outperformed the others, achieving the highest private leaderboard score of 0.88899 and a validation AUC of

0.88963. Surprisingly, the ensembles—both tuned and untuned—did not yield higher performance. The tuned ensemble model (submission4.csv) only reached 0.88402 on the private leaderboard, slightly lower than the single tuned model.

These results highlight that a well-tuned, high-capacity model like XGBoost can surpass ensemble methods, particularly when the additional models (e.g., LogisticRegression) offer limited predictive gain or introduce noise. While ensemble methods are often beneficial in reducing variance and improving generalization, their effectiveness depends heavily on the strength and diversity of the base models.

In conclusion, Experiment 2 (tuned XGBoost) is the recommended approach for this problem, balancing simplicity, performance, and efficiency.

References

Kaggle Playground Series S4E1: https://www.kaggle.com/competitions/playground-series-s4e1

XGBoost Documentation: https://xgboost.readthedocs.io

Scikit-learn Documentation: https://scikit-learn.org

ROC AUC Theory: https://en.wikipedia.org/wiki/Receiver_operating_characteristic