QUANTUM COMPUTING INC

Profiling of CVQBoost Algorithm: Fraud Detection

December 2024

Quantum Computing Inc
quantumcomputinginc.com
(703) 436-2161

Abstract

This study evaluates the performance of CVQBoost, a novel extension of the QBoost algorithm leveraging quadratic optimization solvers like QCi's Dirac-3, for a challenging fraud detection task. Using the imbalanced Kaggle Credit Card Fraud Detection dataset, we benchmark CVQBoost against the state-of-the-art XGBoost algorithm across varying dataset sizes, feature counts, and class imbalance ratios. Our results demonstrate that CVQBoost achieves superior runtime performance, particularly as the size of the training data and the number of features increase, where its scaling advantage becomes increasingly evident.

In terms of accuracy, assessed using the Area Under the Curve (AUC), XGBoost initially exhibits higher performance when balancing is minimal, and the minority-to-majority class ratio remains extremely low. However, as the training data becomes more balanced—using strategies such as ADASYN, SMOTE, SMOTE-SVM, and downsampling—CVQBoost consistently catches up and achieves comparable AUC scores. Notably, with the ADASYN balancing strategy, CVQBoost eventually surpasses XGBoost in accuracy for higher class ratios. These findings highlight CVQBoost's superior scalability and competitive accuracy, making it a promising alternative for large-scale, imbalanced classification problems such as fraud detection.

In summary, benchmarking results show that CVQBoost on Dirac-3 perform superior speed in time as number of features and amount of data increases while giving competitive accuracy.

1 Introduction

The CVQBoost algorithm presented here is our extension of the QBoost algorithm introduced by Neven et al. (2009), is a classification method that leverages quadratic optimization solvers, such as QCi's Dirac machines, to achieve superior speed and power efficiency compared to classical approaches. CVQBoost is an adaptation of the classical boosting algorithm, a powerful machine learning technique that combines the outputs of several weak classifiers to create a strong classifier.

In classical boosting algorithms like AdaBoost, the weights of weak classifiers are iteratively adjusted based on their performance, aiming to minimize the overall classification error. CVQBoost innovates by utilizing quantum computing to solve this optimization problem. It encodes the boosting task as a quadratic optimization problem, leveraging the Dirac machine's capability to explore multiple solutions simultaneously and efficiently escape local minima, surpassing the efficiency of classical algorithms.

In the sections that follow, the runtime of a Dirac-3 implementation of CVQBoost is benchmarked against a state-of-the-art classical method: the XGBoost classification algorithm. Experiments are conducted across varying data sizes and feature counts, with a detailed breakdown of runtimes for different components of the CVQBoost algorithm presented.

2 Formulation

The idea is based on the concept of boosting. Let us assume that we have a collection of N "weak" classifiers h_i where $i = 1, 2, ..., N$. The goal is to construct a "strong" classifier as a linear superposition of these weak classifiers, that is,

$$y = \sum_{i=1}^{N} w_i h_i(\mathbf{x}) \tag{1}$$

where $\mathbf{x}$ is a vector of input features and $y \in \{-1, 1\}$. The goal is find w_i, weights associated with the weak classifiers.

We use a training set $\{(\mathbf{x_s}, y_s)|s = 1, 2, ..., S\}$ of size S. We can determine optimal weights w_i by minimizing,

$$\min_{\mathbf{w}} \sum_{s=1}^{S} |\sum_{i=1}^{N} w_i h_i(\mathbf{x_s}) - y_s|^2 + \lambda \sum_{i=1}^{N} (w_i)^2 \tag{2}$$

where the regularization term $\lambda \sum_{i=1}^{N} (w_i)^2$ penalizes non-zero weights; λ is the regularization coefficient.

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \sum_{j=1}^{N} J_{ij} w_i w_j + \sum_{i=1}^{N} C_i w_i \tag{3}$$

where

$$J_{ij} = \sum_{s=1}^{S} h_i(\mathbf{x_s}) h_j(\mathbf{x_s}) \tag{4}$$

and

$$C_i = -2 \sum_{s=1}^{S} y_s h_i(\mathbf{x_s}) \tag{5}$$

subject to,

$$\sum_{i=1}^{N} w_i = 1 \tag{6}$$

Note that the above algorithm assumes that the total number of weak classifiers, that is N, is less than the number of available qudits on Dirac-3.

2.1 Choices of Weak Classifiers

There are many ways to design a subset of weak classifiers. We have tested CVQBoost using logistic regression, decision tree, naive Bayesian, and Gaussian process classifiers. Each weak classifier is constructed using one or two of the features chosen from all features. This yields a set of weak classifiers that can be used to construct a strong classifier.

3 Use Case

3.1 Dataset

The dataset used in this study is the Kaggle Credit Card Fraud Detection dataset, a widely recognized resource for machine learning research, particularly in anomaly detection.

This dataset comprises transactions made by European credit cardholders over a two-day period in September 2013. It contains 284,807 transactions, of which 492 (approximately 0.172%) are labeled as fraudulent. The significant class imbalance makes it ideal for exploring techniques tailored to imbalanced classification problems.

A total of 38 features are included in the dataset.

3.2 Time Profiling

3.2.1 Impact of Training Data Count on Runtimes

Table 1 presents the runtimes of CVQBoost and XGBoost across varying training dataset sizes, ranging from 1,000 to 150,000 samples. All runtimes are reported in seconds, with each experiment repeated multiple times to ensure reliability. The table provides both the mean and standard deviation of the runtimes, as well as a breakdown of CVQBoost's runtime components. Notably, the Dirac-3 component accounts for a substantial portion of the total runtime for CVQBoost.

A comparison between CVQBoost and XGBoost reveals that XGBoost achieves shorter runtimes on smaller datasets. However, as the training dataset size increases, XGBoost's runtime grows significantly faster than CVQBoost's. This trend is further illustrated in Figure 1, which plots training runtimes against dataset size.

These results suggest that while XGBoost excels in speed for smaller datasets, CVQBoost offers a clear performance advantage as dataset sizes increase. Specifically, CVQBoost begins to outperform XGBoost when the training sample size approaches 50,000 samples, highlighting its scalability for larger datasets.

train data count	cvqboost train time	xgboost train time	dirac3 time
1000	7.3 +/- 1.3	0.9 +/- 0.1	1.3 +/- 0.5
2000	7.9 +/- 1.3	1.8 +/- 0.1	1.7 +/- 0.5
5000	7.4 +/- 1.2	3.7 +/- 0.1	1.4 +/- 0.5
10000	7.3 +/- 1.1	7.2 +/- 0.2	1.1 +/- 0.3
20000	9.0 +/- 1.3	14.6 +/- 0.3	1.7 +/- 0.9
50000	10.5 +/- 3.2	37.4 +/- 1.2	2.3 +/- 3.1
100000	12.3 +/- 1.1	78.0 +/- 2.9	1.2 +/- 0.4
150000	15.1 +/- 1.4	117.4 +/- 4.0	1.3 +/- 0.5

Table 1: Breakdown of CVQBoost and XGBoost runtimes for different training data counts; number of features: 38.

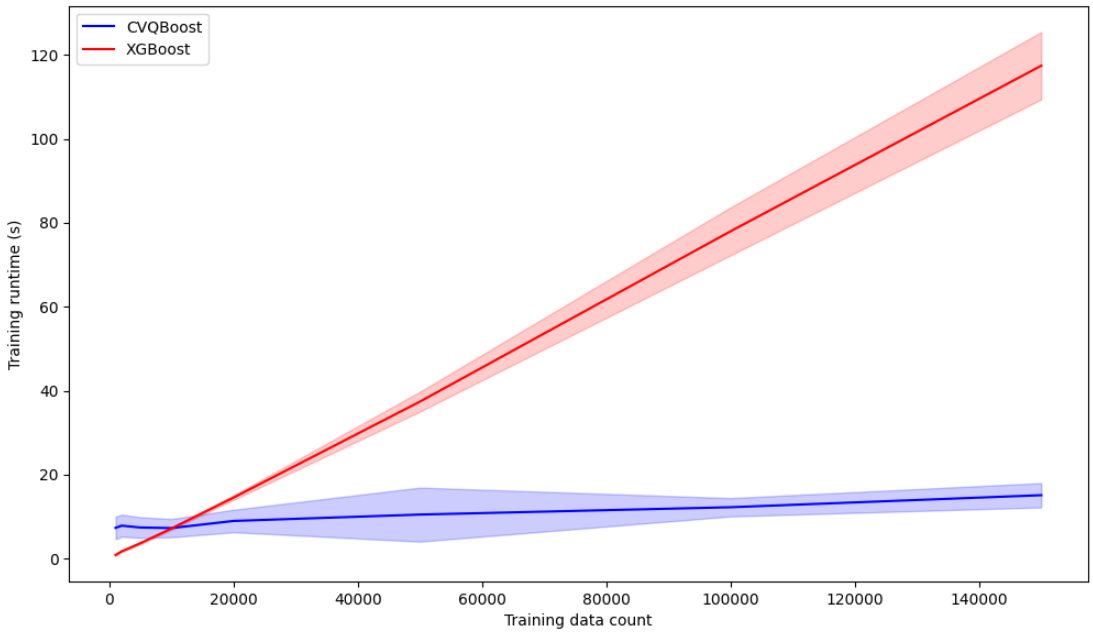

Figure 1: Training runtime of CVQBoost vs. count of training data samples. The 95% confidence intervals are shown.

3.2.2 Number of Features

Table 2 summarizes the runtimes of CVQBoost and XGBoost for varying numbers of features, ranging from 5 to 38. All runtimes are reported in seconds, with each experiment repeated multiple times to ensure reliability. The table includes both the mean and standard deviation of the runtimes, as well as the breakdown of CVQBoost's runtime, where the Dirac-3 component constitutes a significant portion of the total runtime.

As the number of features increases, the runtimes of both CVQBoost and XGBoost grow. However, CVQBoost consistently demonstrates significantly shorter runtimes compared to XGBoost.

Figure 2 further illustrates the relationship between runtime and feature count for both methods. Notably, CVQBoost exhibits a more favorable runtime scaling as the feature dimension increases, making it particularly advantageous for datasets with a larger number of features.

4 Accuracy Profiling

Handling class imbalance is a central challenge in machine learning, particularly in applications such as fraud detection, where the minority class represents rare events. In the original dataset used for this study, fraud cases (the minority class) constitute less than 0.1% of the data. To address

num features	cvqboost train time	xgboost train time	dirac3 time
5	7.6 +/- 1.3	13.7 +/- 0.3	1.2 +/- 0.4
10	7.3 +/- 1.3	18.2 +/- 0.8	1.2 +/- 0.4
20	8.8 +/- 1.3	26.3 +/- 0.5	1.4 +/- 0.5
30	9.5 +/- 1.2	34.4 +/- 3.8	1.2 +/- 0.4
38	10.0 +/- 1.4	41.1 +/- 6.6	1.6 +/- 0.5

Table 2: Breakdown of CVQBoost and XGBoost runtimes for different counts of features; training data count: 50,000.

this extreme imbalance, four widely-used data balancing strategies—ADASYN, SMOTE, SMOTE-SVM, and majority class downsampling—were applied to the training data. The test data remained unchanged to ensure an unbiased evaluation of model performance. The accuracy of CVQBoost and XGBoost was assessed across six levels of class imbalance, characterized by minority-to-majority class ratios of 0.01, 0.02, 0.05, 0.1, 0.5, and 1.0. The Area Under the Curve (AUC), calculated on the test data, was used to measure model accuracy. Table 3 provides a detailed comparison of AUC values across all experiments, while Figures 3–6 illustrate the trends for each balancing strategy.

The results reveal interesting patterns that provide insight into the behavior of the two algorithms under varying levels of class imbalance. When the balancing is minimal—that is, when the minority-to-majority class ratio remains small (0.01 or 0.02)—XGBoost achieves higher AUC scores compared to CVQBoost. This suggests that XGBoost is better able to leverage the heavily skewed class distribution in the early stages of balancing. However, as the class ratio increases and the training data becomes more balanced, CVQBoost's performance improves steadily and catches up to that of XGBoost, eventually achieving comparable AUC scores.

The effect of balancing strategies is particularly noticeable when using ADASYN. ADASYN generates synthetic samples by focusing on regions of the feature space that are prone to misclassification, thereby improving the representation of the minority class. As the minority-to-majority class ratio increases, CVQBoost not only matches but eventually surpasses XGBoost in terms of AUC. This can be seen clearly in both the table and Figure 3, where CVQBoost achieves a slight but consistent improvement over XGBoost at higher class ratios. This suggests that CVQBoost is particularly effective at learning from the diversity introduced by ADASYN-generated synthetic samples.

A similar trend is observed with SMOTE and SMOTE-SVM, where both algorithms benefit as the class ratio increases. However, while XGBoost maintains a slight edge at smaller ratios, CVQBoost consistently narrows the gap as the balancing becomes more pronounced. The superior performance of CVQBoost at higher class ratios reflects its ability to generalize better when sufficient representation of the minority class is provided.

In the case of majority class downsampling, where no synthetic data is introduced and the majority class is simply reduced in size, both algorithms demonstrate gradual improvements in AUC as the

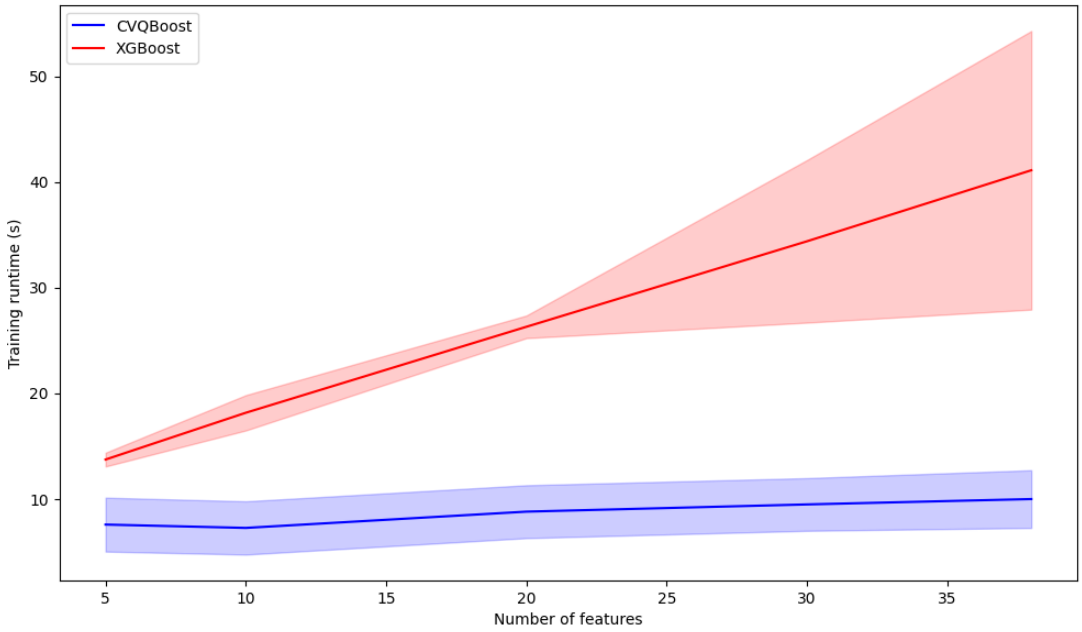

Figure 2: Training runtime of CVQBoost vs. number of features. The 95% confidence intervals are shown.

class ratio increases. However, the rate of improvement is less pronounced compared to the oversampling methods. Even in this scenario, CVQBoost matches and, at times, exceeds the performance of XGBoost as the training data becomes more balanced.

Overall, these results indicate that XGBoost is initially more effective at learning from highly imbalanced data with minimal balancing, as reflected by its superior AUC scores at low class ratios. However, as the minority-to-majority class ratio increases, CVQBoost catches up and often surpasses XGBoost, particularly when using ADASYN. This highlights the adaptability and robustness of CVQBoost, especially in scenarios where synthetic balancing methods enhance the representation of the minority class. Table 3 and Figures 3–6 collectively demonstrate that CVQBoost becomes increasingly advantageous as class imbalance is mitigated, offering superior performance for moderately to highly balanced training data.

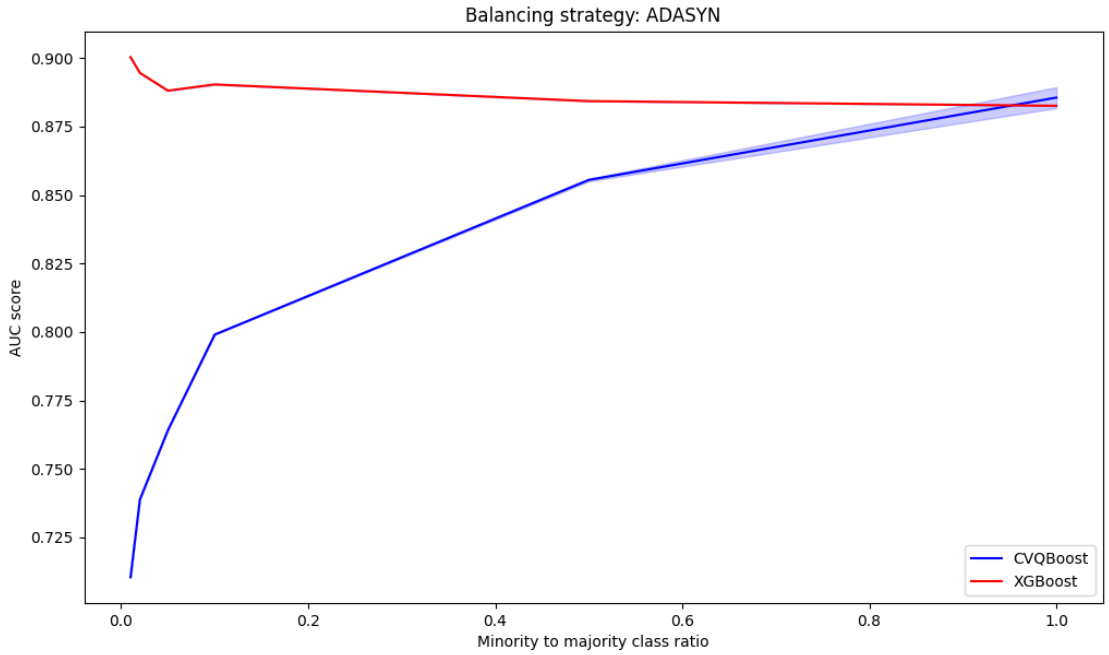

Figure 3: AUC score of CVQBoost vs. minority to majority class ration in training dataset. The balancing strategy used is ADASYN. The 95% confidence intervals are shown.

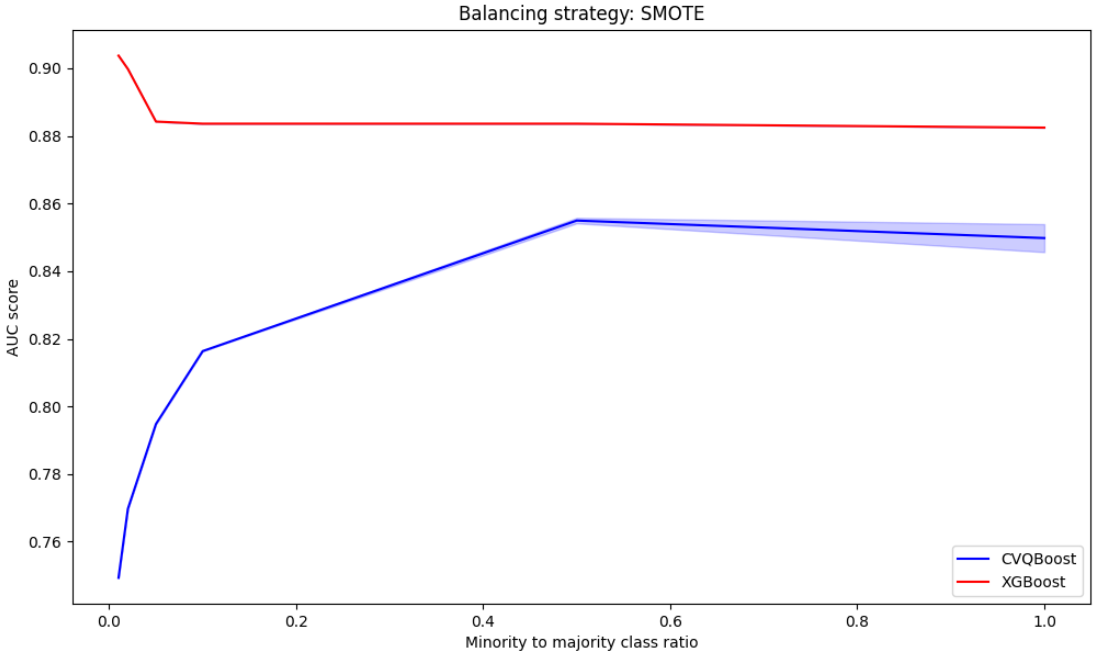

Figure 4: AUC score of CVQBoost vs. minority to majority class ration in training dataset. The balancing strategy used is SMOTE. The 95% confidence intervals are shown.

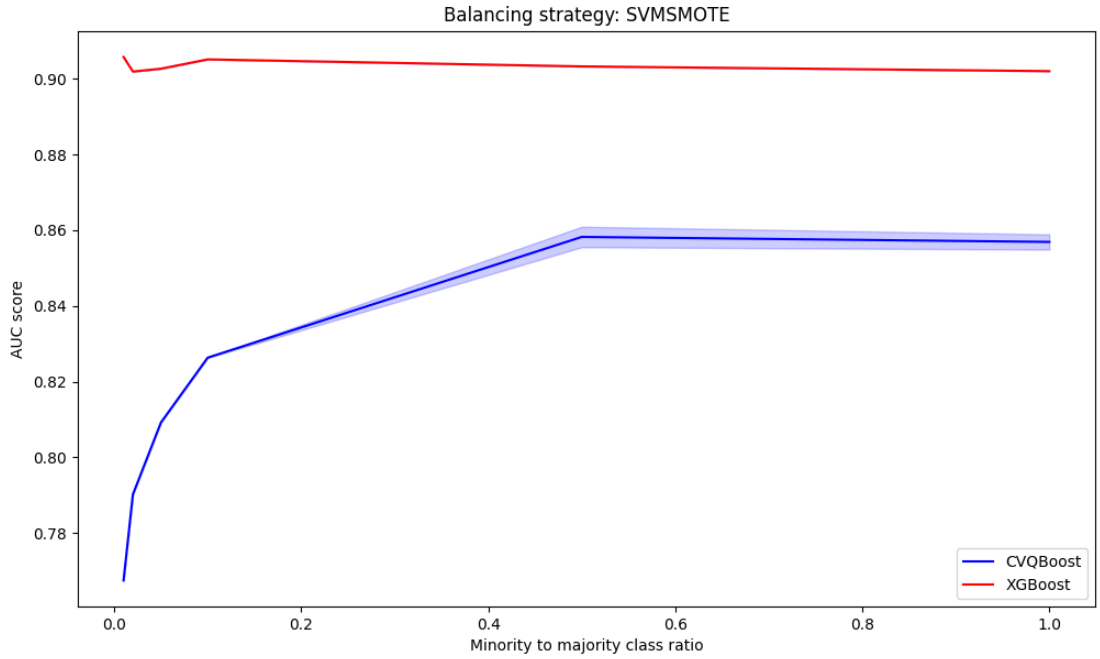

Figure 5: AUC score of CVQBoost vs. minority to majority class ration in training dataset. The balancing strategy used is SVM-SMOTE. The 95% confidence intervals are shown.

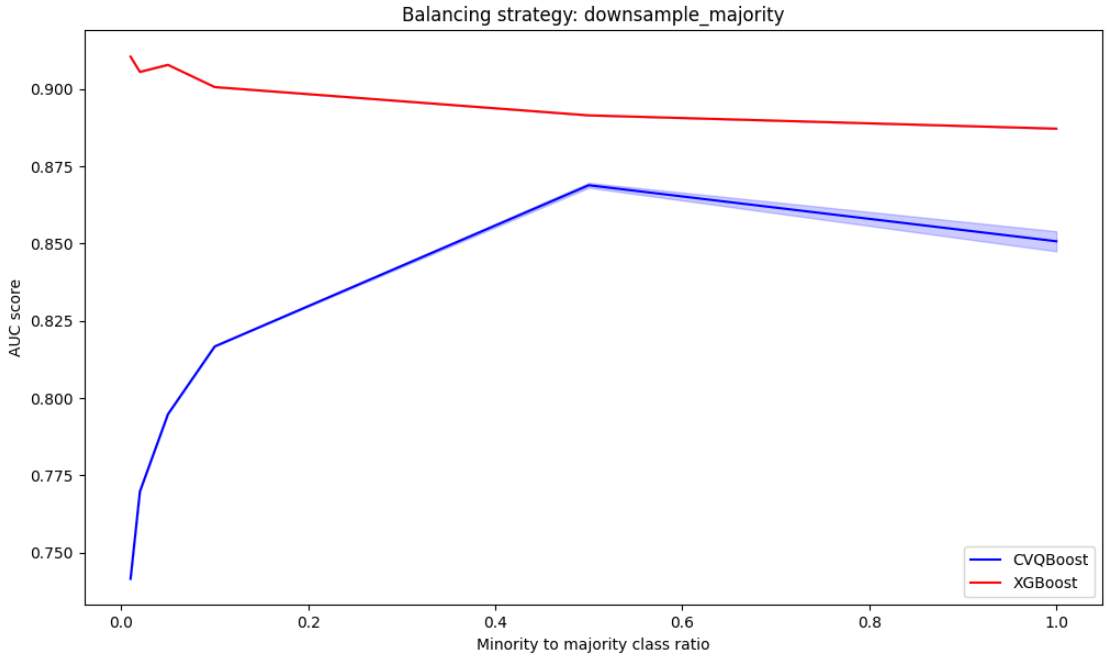

Figure 6: AUC score of CVQBoost vs. minority to majority class ration in training dataset. The balancing strategy used is majority class downsampling. The 95% confidence intervals are shown.

sampling strategy	minority to majority class ratio	cvqboost auc	xgboost auc
ADASYN	0.01	0.7105 +/- 0.0	0.9003 +/- 0.0
ADASYN	0.02	0.7388 +/- 0.0	0.8946 +/- 0.0
ADASYN	0.05	0.7642 +/- 0.0	0.8881 +/- 0.0
ADASYN	0.1	0.799 +/- 0.0	0.8904 +/- 0.0
ADASYN	0.5	0.8555 +/- 0.0002	0.8843 +/- 0.0
ADASYN	1.0	0.8855 +/- 0.0019	0.8826 +/- 0.0
SMOTE	0.01	0.7493 +/- 0.0	0.9037 +/- 0.0
SMOTE	0.02	0.7697 +/- 0.0	0.8998 +/- 0.0
SMOTE	0.05	0.7948 +/- 0.0	0.8842 +/- 0.0
SMOTE	0.1	0.8164 +/- 0.0	0.8836 +/- 0.0
SMOTE	0.5	0.855 +/- 0.0004	0.8836 +/- 0.0
SMOTE	1.0	0.8498 +/- 0.0021	0.8824 +/- 0.0
SVMSMOTE	0.01	0.7674 +/- 0.0	0.9058 +/- 0.0
SVMSMOTE	0.02	0.7902 +/- 0.0	0.9019 +/- 0.0
SVMSMOTE	0.05	0.8092 +/- 0.0	0.9027 +/- 0.0
SVMSMOTE	0.1	0.8263 +/- 0.0	0.9051 +/- 0.0
SVMSMOTE	0.5	0.8582 +/- 0.0014	0.9033 +/- 0.0
SVMSMOTE	1.0	0.8569 +/- 0.001	0.902 +/- 0.0
Downsampling	0.01	0.7415 +/- 0.0	0.9105 +/- 0.0
Downsampling	0.02	0.7698 +/- 0.0	0.9056 +/- 0.0
Downsampling	0.05	0.7948 +/- 0.0	0.9079 +/- 0.0
Downsampling	0.1	0.8167 +/- 0.0	0.9007 +/- 0.0
Downsampling	0.5	0.8689 +/- 0.0004	0.8915 +/- 0.0
Downsampling	1.0	0.8508 +/- 0.0016	0.8872 +/- 0.0

Table 3: AUC scores for CVQBoost and XGBoost using different balancing strategies and minority to majority class rations.