


Presented at *NVIDIA GTC, The Conference for the Era of AI and the Metaverse*,

March 23, 2023. [S51129]

Evaluating XGBoost for Balanced and Imbalanced Data Application to Fraud Detection

Gissel Velarde*, , Anindya Sudhir, Sanjay Deshmane, Anuj Deshmunkh,
Khushboo Sharma, Vaibhav Joshi

Vodafone,
Ferdinand Platz 1, Germany
*gissel.velarde@vodafone.com
<http://www.vodafone.com>

Abstract. This paper evaluates XGboost’s performance given different dataset sizes and class distributions, from perfectly balanced to highly imbalanced. XGBoost has been selected for evaluation, as it stands out in several benchmarks due to its detection performance and speed. After introducing the problem of fraud detection, the paper reviews evaluation metrics for detection systems or binary classifiers, and illustrates with examples how different metrics work for balanced and imbalanced datasets. Then, it examines the principles of XGBoost. It proposes a pipeline for data preparation and compares a Vanilla XGBoost against a random search-tuned XGBoost. Random search fine-tuning provides consistent improvement for large datasets of 100 thousand samples, not so for medium and small datasets of 10 and 1 thousand samples, respectively. Besides, as expected, XGBoost recognition performance improves as more data is available, and deteriorates detection performance as the datasets become more imbalanced. Tests on distributions with 50, 45, 25, and 5 percent positive samples show that the largest drop in detection performance occurs for the distribution with only 5 percent positive samples. Sampling to balance the training set does not provide consistent improvement. Therefore, future work will include a systematic study of different techniques to deal with data imbalance and evaluating other approaches, including graphs, autoencoders, and generative adversarial methods, to deal with the lack of labels.

Keywords: Balanced and Imbalanced Data, XGBoost, Fraud Detection, Performance Evaluation

1 Introduction

Classification is a widely applied machine learning task in industrial setups. Outside laboratories, there might be few cases where class distribution is balanced, since most real-world problems deal with imbalanced datasets. Binary classification systems are evaluated on their ability to correctly identify negative and positive samples. Often, detecting positive samples is critical.

In fraud detection, positive class samples may represent substantial business losses. At the same time, negative samples are essential, and therefore, flagging a negative sample as positive, is a lost business opportunity. Furthermore, the challenges are the following:

- fraudsters continuously change their behavior,
- they may represent rare cases,
- fraud patterns may even be unseen during training, and
- there might be a considerable delay until fraud is identified.

In 2021, estimations in the telecommunications sector report that loss due to fraud accounts for USD 39.89 Billion, representing over two percent of the global revenue of USD 1.8 Trillion [6]. From the several types of fraud, we are most concerned about equipment theft, commissions fraud, and device reselling, in which global losses were estimated USD 3.11 Billion, USD 2.15 Billion, and USD 1.67 Billion, respectively [6].

In recent years, eXtreme Gradient Boosting (XGBoost) has gained attention since it proved highly competitive in machine learning contests for its recognition performance and speed. In this study, XGBoost is systematically evaluated on small, medium, and large datasets presenting different class distributions. In the first experiment, XGboost is evaluated given perfectly balanced data up to highly imbalanced data, where the positive cases represent only 5 percent of all samples.

The contributions of this paper are the following:

- It provides examples to illustrate how different evaluation measures are to be interpreted for detection systems or binary classifiers.
- It reviews the principles of Boosting Trees and the advantages of XGBoost as the selected boosting system.
- It explains a pipeline for a Vanilla XGBoost and a random search-tuned XGBoost.
- It demonstrates empirically that XGboost performance increases its detection performance as the dataset size increases.
- It shows that XGBoost’s performance decreases as the data becomes more imbalanced.
- It tests sampling to balance training set to deal with data imbalance.

The following section reviews evaluation in binary detection systems or binary classifiers. Section 3, reviews XGBoost. The experimental section can be found in section 4. Finally, conclusions are drawn in section 5.

2 Evaluation in Detection Systems

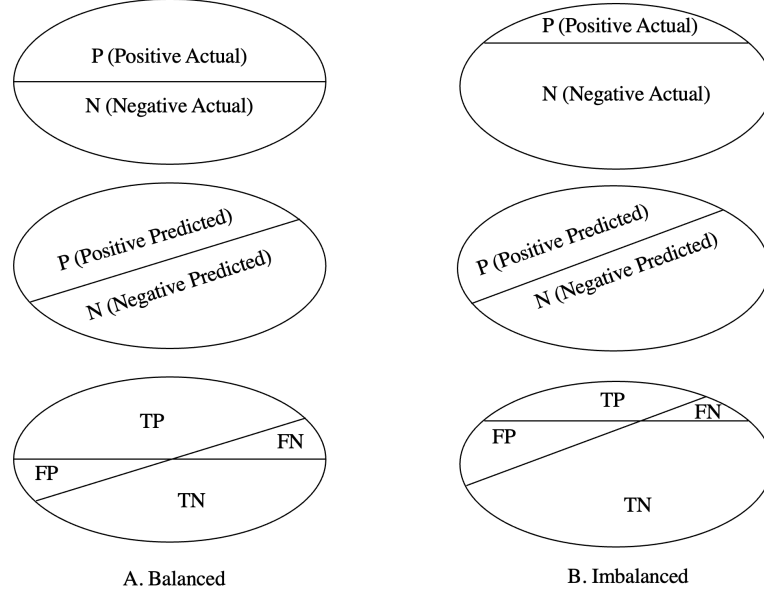


Fig. 1. Examples of possible distributions for balanced and imbalanced datasets.

In detection systems or binary classifiers, we deal with Negative (N) and Positive (P) samples, where the total number of samples is equal to $N + P$. Detection systems are evaluated considering detection performance, observing the number of True Positive (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) samples. See Fig. 1.

Detection systems or classifiers are evaluated considering the following [9],[3]:

- Confusion Matrix,
- Area Under Precision-Recall Curve (AUPRC) also known as Precision/Recall Curve (PRC).
- Precision@n,
- F scores, depending on the case F_1 , $F_{0.5}$ or F_2 ,
- Matthews Correlation Coefficient (MCC),
- False Positive Rate, False Negative Rate,
- Revenue, or Costs, and
- Execution time, among other measures.

In this paper, we will focus on the Confusion Matrix, Precision, Recall, and F scores as relevant measures to evaluate detection systems. Although, Receiver Operating Characteristic (ROC) curve, is still widely used for evaluation, it is

Table 1. Confusion Matrix

		Actual Class	
Predicted Class		P	N
	P	True Positive (TP)	False Positive (FP)
	N	False Negative (FN)	True Negative (TN)

only a powerful tool for balanced datasets, and not recommended for imbalanced datasets [9]. Likewise, Accuracy is deceiving when the dataset is imbalanced. Examples are provided later on.

The confusion matrix shown in Table 1 allows us to compute [9]:

$$BaselinePRC = \frac{P}{P + N}, \quad (1)$$

$$Precision = \frac{TP}{TP + FP}, \quad (2)$$

$$Recall = \frac{TP}{TP + FN}, \quad (3)$$

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}, \quad (4)$$

where F_1 gives same weight to Precision and Recall, $F_{0.5}$ gives more weight to Precision, and F_2 gives more weight to Recall.

In addition, we can compute:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}, \quad (5)$$

however, as mentioned before, Accuracy is not recommended when datasets are imbalanced. For instance, see Table 2, which presents five examples with 1000 samples each. Example 1 has an equal number of Positive and Negative samples, 500 each, respectively. Examples 2 to 5 have 900 Negative and 100 Positive samples. The Baseline PRC is 0.50 for Example 1 and 0.10 for Examples 2 to 5.

Examples 1 and 2. The classifiers make an equal number of mistakes for FP and FN, such that Precision, Recall, F_1 , $F_{0.5}$, F_2 are equal to 0.50. Accuracy is 0.67 for Example 1 and 0.90 for Example 2, although the classifier in the second example still makes the same amount of mistakes, that is, the same number of FP and FN.

Examples 3. It showcases a classifier that flags everything as Negative. In this case, only Recall and Accuracy can be computed, and again Accuracy gives a misleading score of 0.90.

Examples 4. The classifier in this example flags everything as Positive. In this case, Recall is 1. The rest of the measures reflect better the detection ability of such a classifier. As expected, $F_{0.5}$ is worse than F_1 and F_2 , because $F_{0.5}$ gives more weight to Recall.

Examples 5. It showcases a classifier with high Precision but low Recall. Because this classifier makes no mistakes, it is highly precise, but it identifies only five Positive samples out of 100, and therefore its Recall is low. F scores behave as expected.

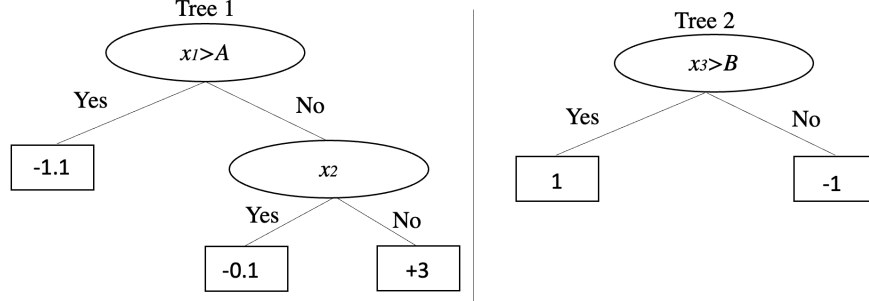
Table 2. Performance for five detection systems (classifiers) when a dataset is balanced (Example 1) and imbalanced (Examples 2 to 5).

	Example 1	Example 2	Example 3	Example 4	Example 5
	Classifier makes an equal number of mistakes for FN and FP	Classifier makes an equal number of mistakes for FN and FP	Classifier Flags everything as Negative	Classifier flags everything as positive, and has high Recall but low Precision	Classifier has high Precision but low Recall
N	500	900	900	900	900
P	500	100	100	100	100
TN	500	850	900	0	900
TP	168	50	0	100	5
FN	166	50	100	0	95
FP	166	50	0	900	0
Precision	0.50	0.50	⚠️	0.10	1.00
Recall	0.50	0.50	0.00	1.00	0.05
F1	0.50	0.50	⚠️	0.18	0.10
F0.5	0.50	0.50	⚠️	0.12	0.21
F2	0.50	0.50	⚠️	0.36	0.06
Accuracy	0.67	0.90	0.90	0.10	0.91
Baseline PRC	0.50	0.10	0.10	0.10	0.10

The previous examples teach us that while it is important to look at Precision and Recall separately, F scores summarise the performance of classifiers. In addition, Accuracy is deceiving when datasets are imbalanced, and therefore, is not recommended for evaluation.

Likewise, the ROC curve, although being a widely used tool to evaluate detection systems or classifiers, is a strong tool only when datasets are balanced and misleading when they are imbalanced [9].

Fig. 2. Example of a tree ensemble model with two trees. Decision nodes are oval and leaf nodes are rectangular. Each tree contributes to the final prediction.



3 XGBoost Review

Extreme Gradient Boosting (XGBoost) [2] is a powerful tree boosting method [4] that:

- First, creates a decision tree,
- Then, iterates over M number of trees, such that
 - It builds a tree likely selecting samples that were misclassified by the previous tree.

See Figure 2. On tabular data, XGBoost has been reported to win several machine learning competitions [2]. Related work shows that XGBoost outperforms several machine learning algorithms for fraud detection in mobile payment [5]. The authors used a synthetic dataset called *Paysim* containing more than 6 million samples with nine attributes or features, where only 0.13 percent of samples are positive. The study evaluated several supervised and unsupervised learning algorithms, reporting that Random Forest (RF) obtains the second best F_1 for supervised learning algorithms, being XGBoost much faster than RF. Besides, XGBOD, which is presented as a semi-supervised learning method performs best, but to train XGBOD, labels are required just like in a supervised setting, and XGBOD is very slow compared to XGBoost. Multi-Objective Generative Adversarial Active Learning (MO-GAAL) returns the fourth best score and it is the slowest of all algorithms. See Table 3.

Table 4 and Figure 3 present comparisons between XGBoost and other boosting systems [2]. XGBoost is the fastest and possesses several characteristics, like sparsity awareness and parallel computing, that made it our choice for experimentation. The Exact Greedy Algorithm deals with finding the best splits and enumerates all of them over all features, being computationally expensive. Therefore, XGBoost implements approximate local (per split) and global (per tree) solutions to the problem of finding the best splits [2].

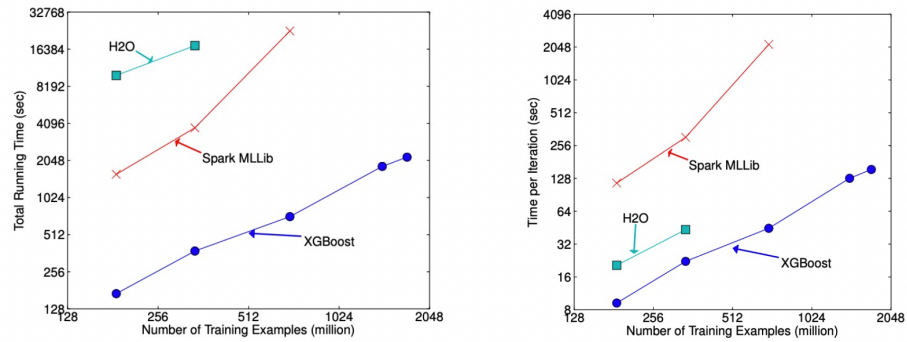
Table 3. Detection performance for various supervised and unsupervised methods on *Paysim* dataset. Taken from [5].

Supervised	Method	<i>F1</i>	<i>Precision</i>	<i>Recall</i>	<i>Execution time [s]</i>
	<i>k</i> -NN	0.1588	0.0873	0.8744	4,581.4
	SVM	0.4655	0.9474	0.3086	12,082.9
	RF	0.8394*	0.9146	0.7756	1,196.2
	XGBoost	0.8410*	0.8794	0.8059	207.0
Unsupervised	Method	<i>F1</i>	<i>Precision</i>	<i>Recall</i>	<i>Execution time [s]</i>
	ABOD	0.0680	0.0675	0.0685	2,646.5
	CBLOF	0.0822	0.0829	0.0822	41.3
	HBOS	0.0077	0.0078	0.0076	4.1
	LODA	0.1060	0.1026	0.1096	14.8
	Isolation Forest	0.0189	0.0307	0.0137	189.9
	KNN	0.1260	0.1288	0.1233	1,948.5
	MCD	0.1084	0.1087	0.1081	127.4
	OCSVM	0.0273	0.0272	0.0274	802.9
	AE#	0.0869	0.0870	0.0868	931.1
	VAE#	0.0869	0.0870	0.0868	2,922.9
	MO-GAAL	0.6059	0.5902	0.6225	13,184.4
Requires labels	XGBOD	0.8737	0.9942	0.7793	4,256.3

The best results are in bold, # The experiments were performed on GPU NVIDIA GeForce GTX 1060 6GB, 1280 cores on a Windows 10 oper. system in the Python library PyOD

Table 4. Comparison of tree boosting systems considering several characteristics. Taken from [2].

System	exact greedy	approximate global	approximate local	out-of-core	sparsity aware	parallel
XGBoost	yes	yes	yes	yes	yes	yes
pGBRT	no	no	yes	no	no	yes
Spark MLlib	no	yes	no	no	partially	yes
H2O	no	yes	no	no	partially	yes
scikit-learn	yes	no	no	no	no	no
R GBM	yes	no	no	no	partially	no

Fig. 3. Comparison of tree boosting systems considering time. Taken from [2].

(a) End-to-end time cost include data loading

(b) Per iteration cost exclude data loading

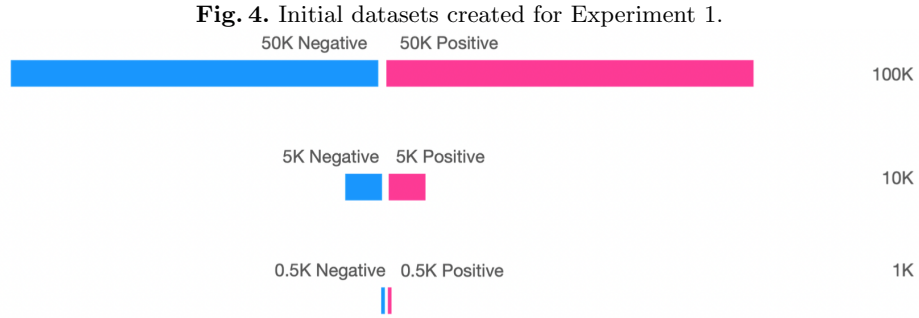
4 Experiments

Two experiments are presented in this section for private data (non-synthetic) containing more than 150 features.

4.1 Experiment 1

The first experiment aims at studying XGBoost performance in relation to dataset size and class distribution. In addition, it aims at understanding the impact of parameter fine-tuning with random search, given that random search is more efficient than grid search [1].

Datasets. Three datasets were created with 100, 10 and 1 thousand (K) samples, respectively, with an equal number of negative and positive samples in each dataset, as illustrated in Figure 4. Then each subset was sampled such that it had four distributions from balanced to highly imbalanced (N%-P%), such that the positive class was under-sampled from: 50%-50%, 55%-45%, 75%-25%, to 95%-5%. See Figure 5.



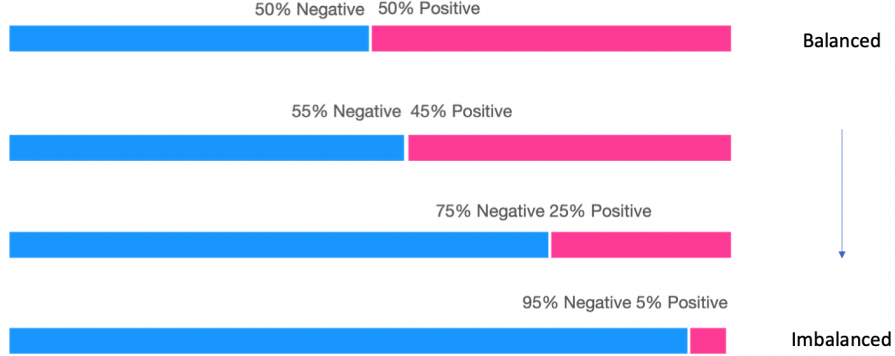
Pipeline The datasets were partitioned using 80-20% for training-testing. Data was prepared as follows:

- Numerical data was scaled between 0 and 1.
- Categorical data was encoded with an ordinal encoder, such that values unseen during training received a reserved value.

A Vanilla XGboost, with most default parameters [10], was tested with the following setup:

- Binary logistic objective function,
- Handling of missing values,

Fig. 5. Datasets created from the partition shown in Figure 4, for 100K, 10K and 1K samples.



- Random state equal to 100 for reproducibility,
- Evaluation metric: AUC-PR,
- Tree method: GPU histogram
- Maximum depth of a tree equal to 6,
- Learning rate equal to 0.3,
- Subsample ratio of training instances before growing trees equal to 1,
- Subsampling of columns by tree equal to 1, and
- Number of trees equal to 100.

In addition, Random Search (RS)-Tuned XGBoost models were obtained in cross-validation over the following space:

- Maximum depth of a tree in values equal to 3, 6, 12, and 20,
- Learning rate in values equal to 0.02, 0.1, and 0.2,
- Subsample ratio of training instances before growing trees equal to 0.4, 0.8, and 1,
- Subsampling of columns by a tree in values equal to 0.4, 0.6, and 1, and
- Number of trees equal to 100, 1000, and 5000.

This set of parameters were tested over random search, given that a winning Kaggle entry for fraud detection use them [8].

4.2 Results of Experiment 1

Table 5 presents the results obtained by the Vanilla XGboost configuration. As expected, XGBoost improves as the dataset increases in size. Focusing on F_1 score, we observe that for the balanced distribution and up to 75%-25%, there is a larger improvement from 1K to 10K, and a smaller increase from 10K to 100K. In contrast, for the most imbalanced case of 95%-5%, the most considerable improvement occurs between 10K and 100K samples.

Table 6 presents the results obtained by RS-Tuned XGBoost, and Table 7 shows a comparison of F_1 scores taken from Table 5 and Table 6, that is a comparison between Vanilla XGBoost and RS-Tuned XGBoost. Random Search fine-tuning did not present improvement for 1K and 10K samples. Moreover, random search fine-tuning deteriorated the performance for the highly imbalanced scenario of 95%-5%. In contrast, the benefit of random search fine-tuning can be observed for the larger dataset of 100K samples. These results indicate that XGBoost default parameters are the recommended choice for small and medium datasets in comparison to the selected parameters tested for RS-Tuned XGBoost. For larger datasets, random search fine-tuning improves detection performance at the cost of being computational expensive. A Summary of the results can be seen in Figure 6.

Table 5. Performance for Vanilla XGBoost.

1K	Positive %	Precision	Recall	F1	F05
	50	0.8387	0.7800	0.8083	0.8263
	45	0.8621	0.8333	0.8475	0.8562
	25	0.8780	0.7200	0.7912	0.8411
	5	0.4000	0.4000	0.4000	0.4000
10K	Positive %	Precision	Recall	F1	F05
	50	0.8723	0.8740	0.8731	0.8726
	45	0.8765	0.8678	0.8721	0.8748
	25	0.7887	0.7540	0.7710	0.7815
	5	0.6889	0.3100	0.4276	0.5536
100K	Positive %	Precision	Recall	F1	F05
	50	0.8847	0.8945	0.8896	0.8866
	45	0.8792	0.8802	0.8797	0.8794
	25	0.8263	0.7860	0.8057	0.8179
	5	0.7147	0.4610	0.5605	0.6439

4.3 Experiment 2

Since XGBoost performs best when dealing with balanced datasets and is strongly affected for the highly imbalanced dataset containing only 5% positive samples,

Table 6. Performance for RS-Tuned XGBoost. For 1K, random search run three times. Reported mean values.

1K*	Positive %	Precision	Recall	F1	F05
	50	0.8593	0.7933	0.8249	0.8452
	45	0.8501	0.8185	0.8339	0.8435
	25	0.8582	0.6867	0.7612	0.816
	5	0.3595	0.2000	0.2542	0.3068
10K	Positive %	Precision	Recall	F1	F05
	50	0.8739	0.8870	0.8804	0.8765
	45	0.8713	0.8722	0.8717	0.8714
	25	0.8162	0.7640	0.7893	0.8052
	5	0.7297	0.2700	0.3942	0.5444
100k	Positive %	Precision	Recall	F1	F05
	50	0.8884	0.8979	0.8931	0.8903
	45	0.8836	0.8839	0.8837	0.8837
	25	0.8345	0.7866	0.8098	0.8245
	5	0.7618	0.4670	0.5790	0.6764

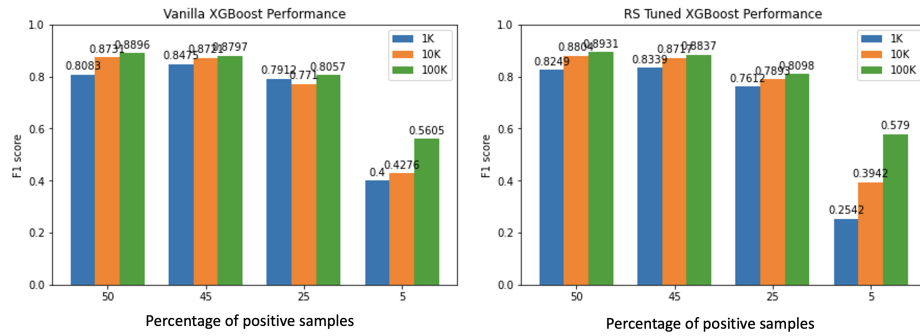
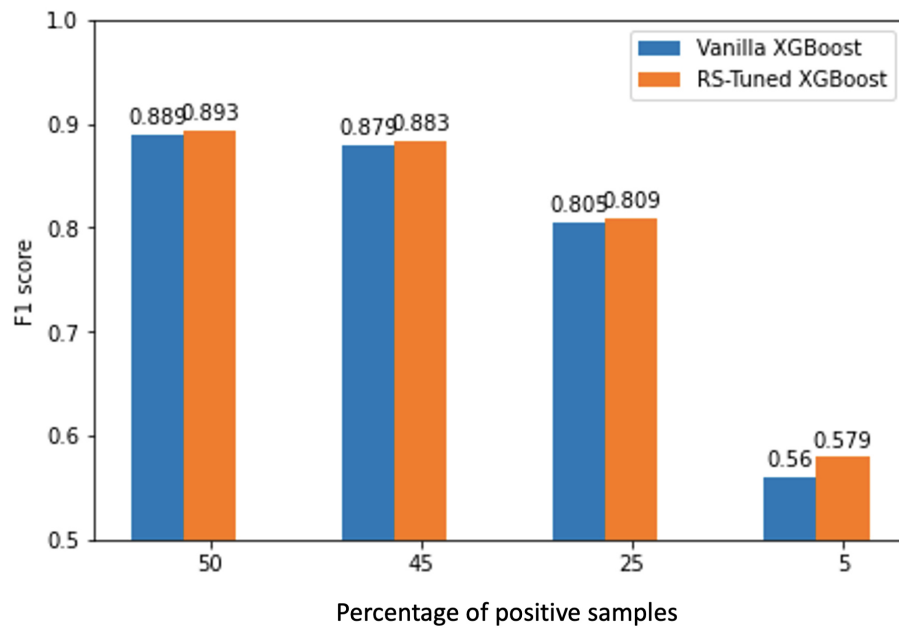
Fig. 6. F1 scores obtained by Vanilla XGBoost and RS-Tunned XGBoost from Tables 5 and 6.

Table 7. Comparison between Vanilla XGBoost and RS-Tuned XGboost. Best F_1 scores in bold. For 1K, random sampling run three times. Fine tuning did not provide consistent improvement for small and medium datasets. In addition, the standard deviation for 95%-5% is high. In contrast, fine tuning with random search consistently improve performance for the dataset of 100K samples.

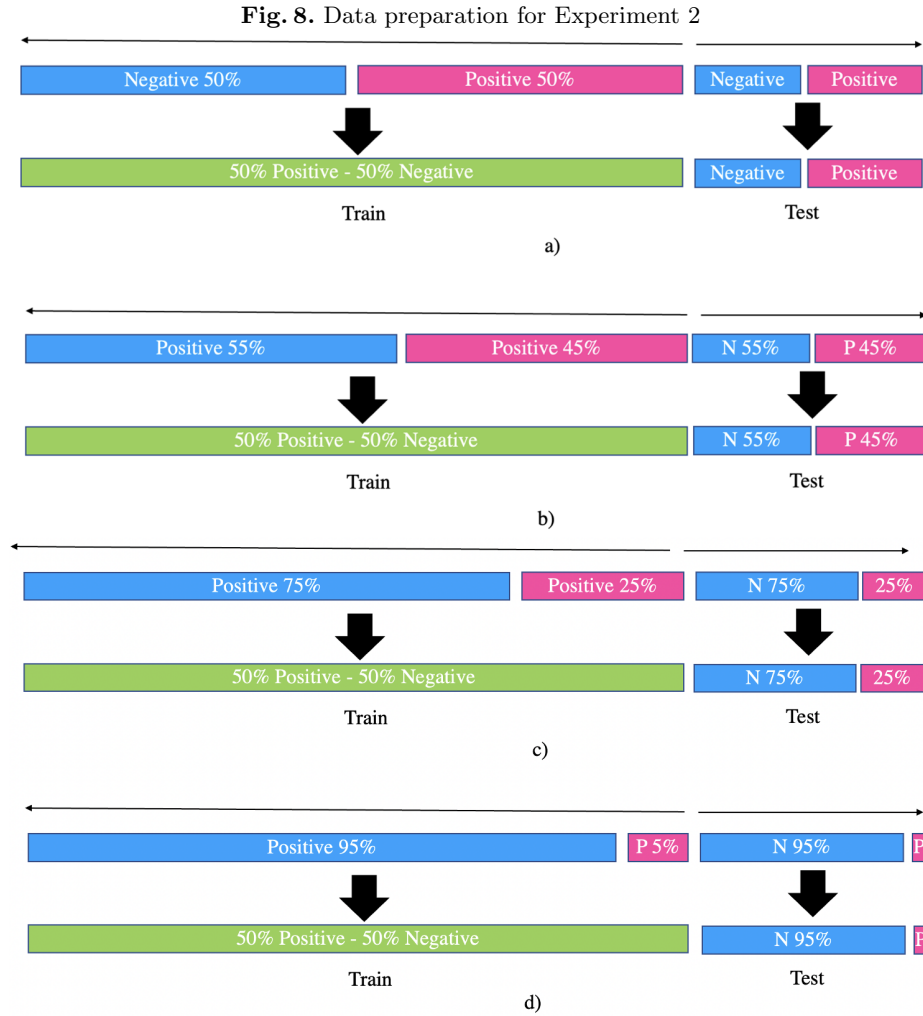
		Vanilla	RS-Tuned parameters
1K*	Positive %	F1	F1
	50	0.8083	0.8249±0.00
	45	0.8475	0.8339±0.00
	25	0.7912	0.7612±0.03
	5	0.4000	0.2542±0.11
10K	Positive %	F1	F1
	50	0.8731	0.8804
	45	0.8721	0.8717
	25	0.7710	0.7893
	5	0.4276	0.3942
100K	Positive %	F1	F1
	50	0.8896	0.8931
	45	0.8797	0.8837
	25	0.8057	0.8098
	5	0.5605	0.5790

Fig. 7. F1 scores obtained by Vanilla XGBoost and RS-Tunned XGBoost for the dataset of 100K samples, results from Table 7.



the second experiment was designed to study the effect of sampling as a technique to improve detection for imbalanced distributions.

Datasets. Figure 8 illustrates how data was sampled, such that each training set had equal number of positive and negative samples, while the test set reflected four distributions: 50%-50%, 55%-45%, 75%-25%, and 95%-5%. A time point was selected to split the data on train and test sets. Then, sampling followed to obtain the desired distribution for each set and partition. This procedure was repeated for 1K and 10K samples, with 80%-20%, train-test partition.



4.4 Results Experiment 2

Table 8 shows the results when training set is sampled to have equal number of positive and negative samples and XGBoost is tuned using random search. In this case, the estimated F_1 becomes unrealistic for distributions of 75%-25% and 95%-5%. Besides, recall improves but precision worsens. Since we observed that random search fine tuning did not provide consistent improvement for datasets of 1K and 10K, a vanilla XGBoost was also tested. The comparison between Vanilla and RS-Tuned XGBoost can be seen in Table 9. Sampling on training set did not produce consistent improvement on Vanilla or RS-Tuned XGBoost.

Table 8. Effect of sampling to 50%-50% (Negative - Positive) on training set for datasets of size 1K and 10K. For RS-Tuned XGBoost, recall improves but precision worsens. The estimated F_1 score becomes unrealistic for 75%-25% and 95%-5% distributions.

1K	Positive %	Precision	Recall	F1	F05	F1 estimated
	50	0.8000	0.9600	0.8727	0.8276	0.8584
	45	0.7456	0.9444	0.8333	0.7784	0.8655
	25	0.5341	0.9400	0.6812	0.5846	0.8522
	5	0.1860	0.8000	0.3019	0.2198	0.8296
10K	Positive %	Precision	Recall	F1	F05	F1 estimated
	50	0.8544	0.8920	0.8728	0.8617	0.8743
	45	0.8255	0.8833	0.8535	0.8365	0.8751
	25	0.6934	0.9000	0.7833	0.7267	0.8788
	5	0.2451	0.8700	0.3824	0.2862	0.8745

5 Conclusions

This work focused on evaluating XGBoost on balanced and imbalanced datasets. It explained how evaluation measures are to be interpreted in general for detection systems or binary classifiers. The provided examples showed that accuracy

Table 9. RS-Tuned XGboost and Vanilla XGboost under the effect of sampling to 50%-50% (Negative - Positive) on training set for datasets of size 1K and 10K. There is no consistent improvement from using any of the configurations.

		RS-Tuned XGBoost and sampling train set	Vanilla XGBoost and sampling train set
1K	Positive %	F1	F1
	50	0.8727	0.8818
	45	0.8333	0.8235
	25	0.6812	0.6667
	5	0.3019	0.2540
10K	Positive %	F1	F1
	50	0.8728	0.8648
	45	0.8535	0.8473
	25	0.7833	0.7940
	5	0.3824	0.3973

is deceiving when datasets become imbalanced. Besides, it showed that while it is essential to observe precision and recall values, F scores summarise performance. Depending on the application, F_1 should be considered if precision and recall are equally weighted, $F_{0.5}$ if precision is preferred over recall, and F_2 if recall is more important than precision.

In addition, this work reviewed XGBoost, which stands out in various benchmarks as a recommended boosting system [2],[5]. The proposed pipeline scales numerical values between 0 and 1, and encodes categorical data, giving a reserved value when unseen categories appear in test. Preliminary experiments showed that scaling numerical values does not have an impact for small and medium dataset, but improves performance when a dataset reaches 100K samples. Therefore, scaling numerical values has been used in the pipeline thought out the experiments. Besides, as expected, this report empirically demonstrated that XGboost increases its detection performance as the dataset size increases. Moreover, the experiments showed that XGBoost’s performance decreases as the data becomes more imbalanced.

Sampling was tested as a solution to overcome the problem of decreased performance when imbalance increases. Sampling to balance the training set did not provide consistent improvement. Similarly, related work found that random under-sampling deteriorates XGBoost’s performance [5]. There are various methods implemented to deal with the problem of decreased performance for imbalanced datasets [7]. These methods will be tested in future work. In addition, forthcoming tests will involve fine-tuning XGBoost weights that observe the ratio between positive and negative samples.

Finally, future directions include comparing XGBoost against Graphs, autoencoders, and generative adversarial approaches such as Multi-Objective Generative Adversarial Active Learning (MO-GAAL) to deal with the lack of labels.

Author contributions

G.V. wrote the paper, created the dataset partitions, developed the pipeline, train and tested XGBoost, and evaluated the results. A.S., S.D. and A.D. provided insights on previous implementations. A.S., K.S., and V.J. collected the dataset for the experiments. K.S. started initial experiments with graphs.

Acknowledgment

We would like thank Michael Weichert for his feedback on drafts of this report, and Rafael Niegoth for providing business knowledge and assigning the task of developing and evaluating a detection system. We would like thank Praveen Maurya and Steffen Wenzel for their support with the servers. In addition, we would like to thank the NVIDIA reviewers and organisers Lilac Ilan, Krystian Garbaciak, Melanie Mangum, and Bridget Johnson for their feedback and support.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. *Journal of machine learning research* **13**(2) (2012)
2. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. pp. 785–794 (2016)
3. Chicco, D., Tötsch, N., Jurman, G.: The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData mining* **14**(1), 1–22 (2021)
4. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. *Annals of statistics* pp. 1189–1232 (2001)
5. Hajek, P., Abedin, M.Z., Sivarajah, U.: Fraud detection in mobile payment systems using an xgboost-based framework. *Information Systems Frontiers* pp. 1–19 (2022)
6. Howell, J.: Telecom fraud on the rise: 2021 cfca global telecommunications fraud loss survey. (2021), <https://www.subex.com/blog/2021-cfca-global-telecommunications-fraud-loss-survey/>, accessed 6-3-2023
7. Lemaître, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research* **18**(1), 559–563 (2017)
8. McDonald, C., Deotte, C.: Leveraging machine learning to detect fraud: Tips to developing a winning kaggle solution (2021), <https://developer.nvidia.com/blog/leveraging-machine-learning-to-detect-fraud-tips-to-developing-a-winning-kaggle-solution/>, accessed 13-2-2023
9. Saito, T., Rehmsmeier, M.: The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one* **10**(3), e0118432 (2015)
10. xgboost developers: Xgboost parameters (2022), <https://xgboost.readthedocs.io/en/stable/parameter.html>, accessed 7-2-2023