Fraud Detection Systems:

Evaluating XGBoost for Balanced and Highly Imbalanced Data

Gissel Velarde, Ph.D

Senior Expert Data Scientist,

Vodafone GmbH

Gissel Velarde, Ph.D, MSc. Ing.

- I started my career in 2000
- Ph.D. in Computer Science and Engineering, Aalborg University (2017)
- Pattern Discovery Algorithms ranked high at the MIREX (2014,2015,2016)
- Teaching award from UPB, Lecture "Selected Topics in Artificial Intelligence." (2021)
- Vodafone’s Star Award (2022)

A. Vaidya

K. Sharma

P. Asawara

M. Weichert

P. Gulpa

A. Sundhir

R. Niegoth

G. Velarde

A. Deshmukh

S. Deshmane

V. Joshi

S. Jadhav

A. B. Mazi

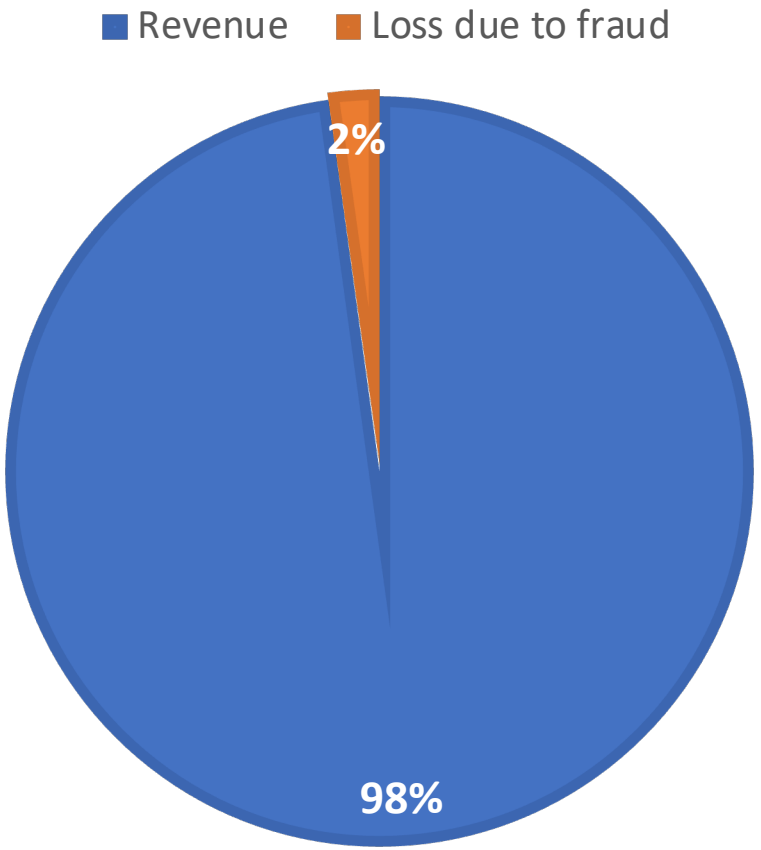
S. Kapoor

This talk is for you if you…

- Are interested in Fraud Detection
- You care about accelerated data science
- Faced the Curse of Imbalanced Data
- Want to know more about ML Boosting Systems's performance

At the end of the talk…

- You will better understand
  - how good can Boosting Systems be depending on your Data
- I'll share some useful resources for further learning



Estimations 2021

- Global telecom revenues USD 1.8 Trillion
- Loss due to fraud USD 39.89 Billion
  - Equipment theft USD 3.11 Billion
  - Commissions Fraud USD 2.15 Billion
  - Device Reselling USD 1.67 Billion

Jacob Howell , 2021. Telecom Fraud on the rise: 2021 CFCA Global Telecommunications Fraud Loss Survey. https://www.subex.com/blog/2021-cfca-global-telecommunications-fraud-loss-survey/

Images taken from Google search

Why is Fraud detection hard?

- Fraudsters continuously change their behaviour,
- They may represent rare cases,
- Fraud patterns may even be unseen during training,
- Time delay until Fraud is identified

Industries affected by fraud

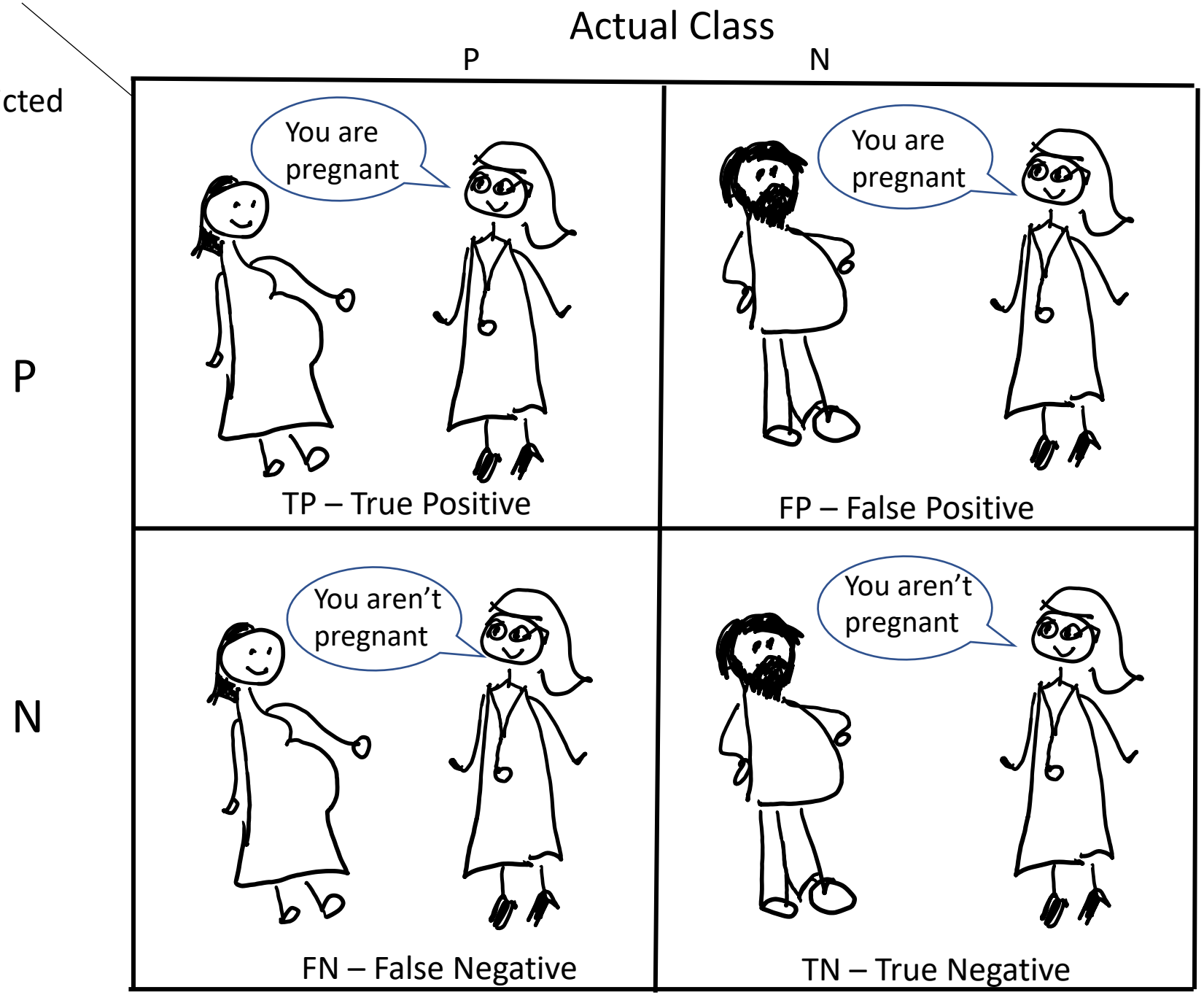
- Telecommunications
- Finance
- E-Commerce
- Automotive

Fraud Detection

Is treated as a binary classification problem

The positive class is of extreme interest

Evaluation
In Detection Systems


Actual Class
P
N
Predicted
P
N
You are pregnant
TP – True Positive
You are pregnant
FP – False Positive
You aren't pregnant
FN – False Negative
You aren't pregnant
TN – True Negative

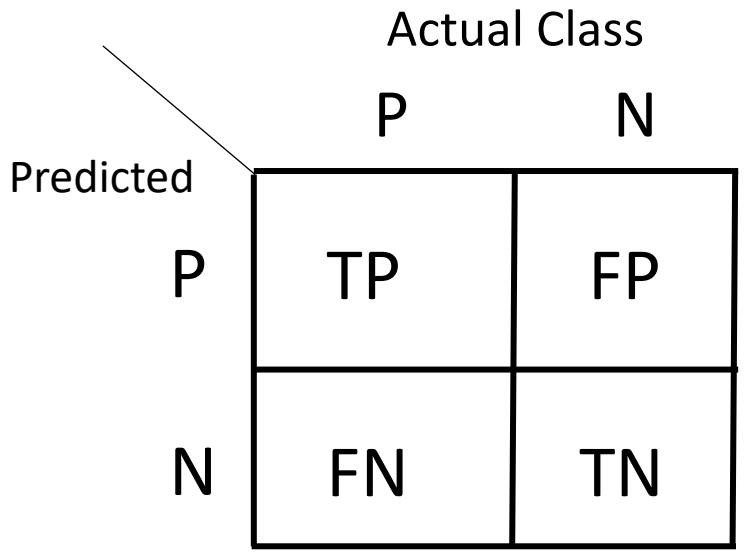
Evaluation
In Detection Systems

TP – True Positive

TN – True Negatives

FP – False Positives

FN – False Negatives

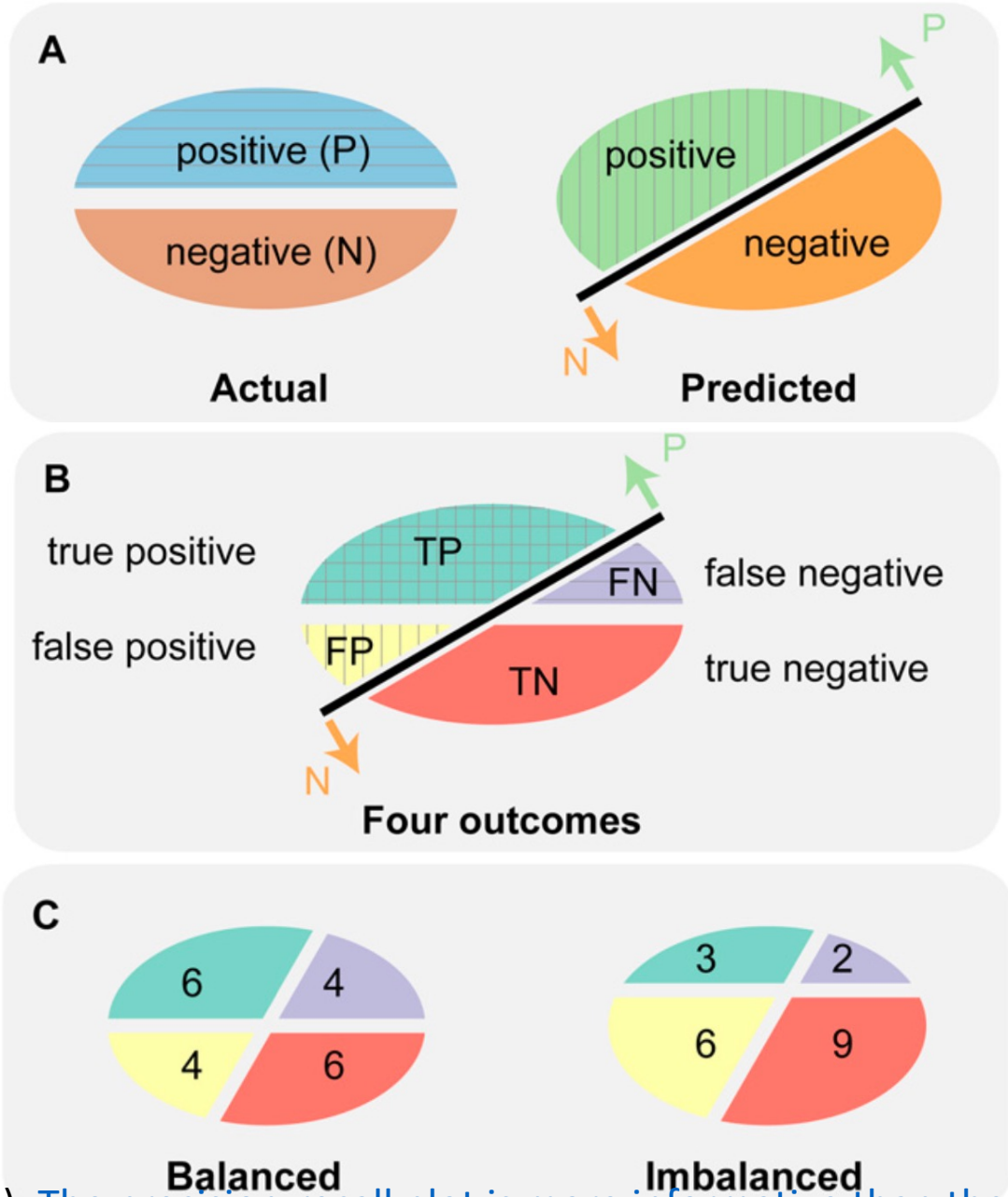


Is your data Imbalanced?

Evaluation

Class distribution of the data matters

Evaluation measures behave differently under balanced and imbalanced datasets



Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, *10*(3), e0118432.

Key Performance Indicators

- Confusion Matrix
- Area Under Precision-Recall Curve – AUC-PR (instead of ROC)
- Precision@n
- F1 Score (Depending on your case F0.5 or F2)
- Matthews Correlation Coefficient (MCC)
- False Positive Rate, False Negative Rate
- Revenue or Cost
- Time

- Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, *10*(3), e0118432.
- Chicco, D., Tötsch, N., & Jurman, G. (2021). 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.

Key Performance Indicators

- Precision-Recall – AUC-PR (instead of ROC)
- F1 Score (Depending on your case F0.5 or F2)

- Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, *10*(3), e0118432.
- Chicco, D., Tötsch, N., & Jurman, G. (2021). 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.

$$\text{Baseline PRC} = \frac{P}{P + N}$$

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

Predicted \ Actual Class	P	N
P	TP	FP
N	FN	TN

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

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

$F_{0.5}$ gives more weight to precision
F_2 gives more weight to recall

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

	Example 1	Example 2
	Classifier makes an equal number of mistakes for FN and FP	Classifier makes an equal number of mistakes for FN and FP
N	**500**	**900**
P	500	100
TN	500	850
TP	168	50
FN	166	50
FP	166	50
Precision	0.50	0.50
Recall	0.50	0.50
F1	0.50	0.50
F0.5	0.50	0.50
F2	0.50	0.50
Accuracy	0.67	0.90
Baseline PRC	0.50	0.10

Example 1. Balanced Data
Example 2. Imbalanced Data

	Example 3	Example 4	Example 5
	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	**900**	**900**	**900**
P	100	100	100
TN	900	0	900
TP	0	100	5
FN	100	0	95
FP	0	900	0
Precision	⚠	0.10	**1.00**
Recall	0.00	**1.00**	0.05
F1	⚠	0.18	0.10
F0.5	⚠	0.12	0.21
F2	⚠	0.36	0.06
Accuracy	0.90	0.10	0.91
Baseline PRC	0.10	0.10	0.10

All examples with Imbalanced data

eXtreme Gradient Boosting (XGBoost)

Why XGBoost?

Based on Literature review

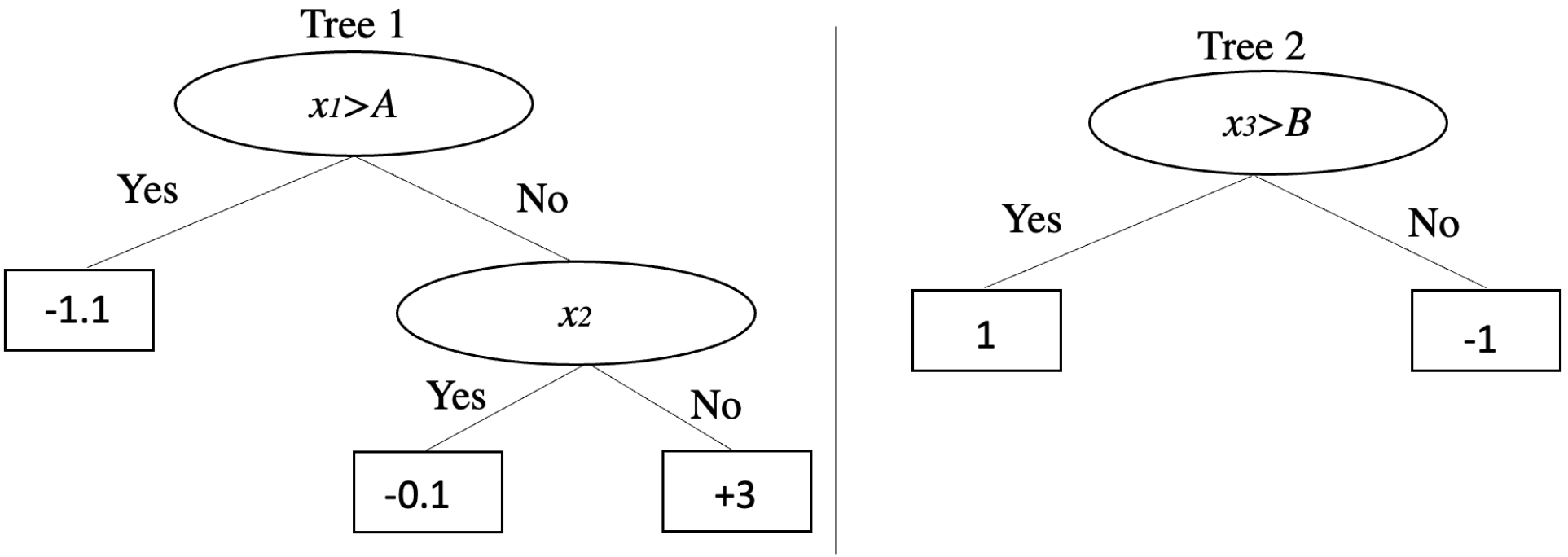
Empirical evidence

Time

Ensemble of decision trees

1. Create a simple decision tree
2. Iterate over *M* number of trees

    Build a tree selecting samples that were misclassified by the previous tree



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.

PaySim Dataset (Synthetic)

9 Attributes

6.32 M samples

Train – Test -> 75% -25%

Train with 5-fold Cross Validation

Hajek, P., Abedin, M. Z., & Sivarajah, U. (2022). Fraud Detection in Mobile Payment Systems using an XGBoost-based Framework. *Information Systems Frontiers*, 1-19.

Supervised

Method	*F1*	*Precision*	*Recall*	*Execution time [s]*
k-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

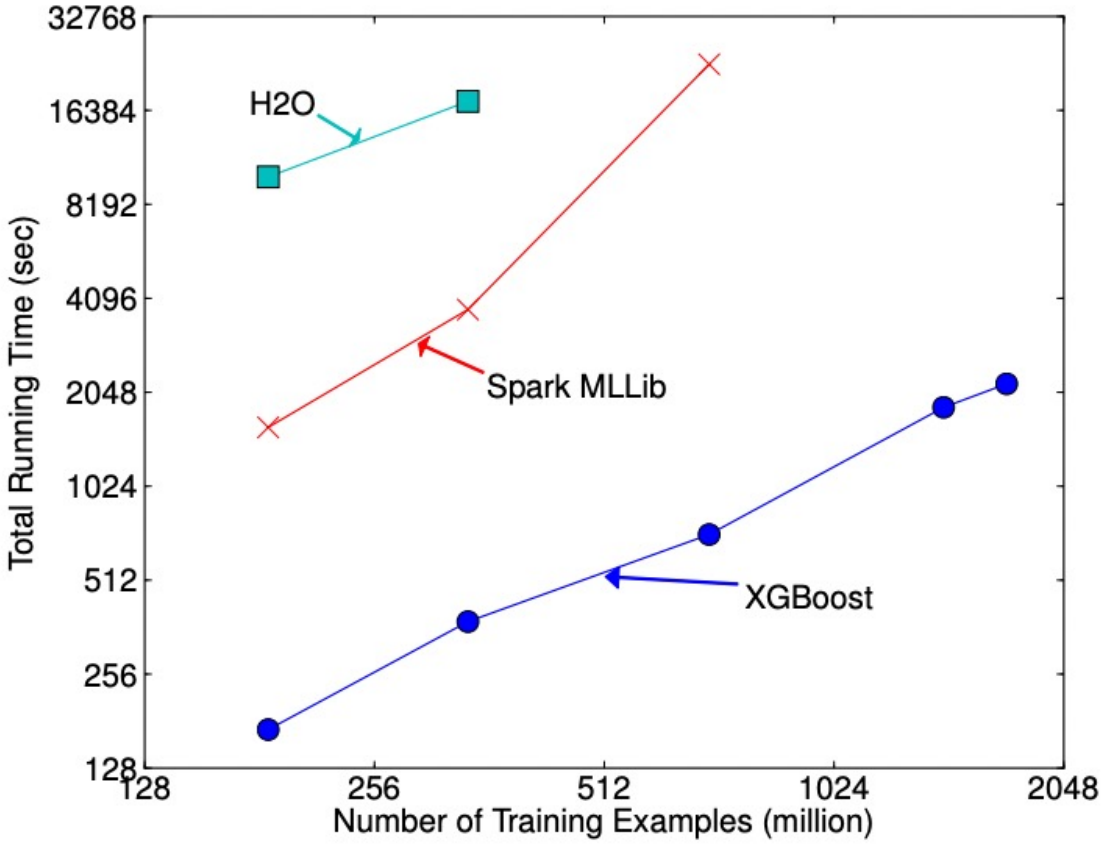
Method	*F1*	*Precision*	*Recall*	*Execution time [s]*
ABOD	0.0680	0.0675	0.0685	2,646.5
CBLOF	0.0822	0.0829	0.0822	41.3
HBOS	0.0077	0.0078	0.0076	4.1
LODA	0.1060	0.1026	0.1096	14.8
Isolation Forest	0.0189	0.0307	0.0137	189.9
KNN	0.1260	0.1288	0.1233	1,948.5
MCD	0.1084	0.1087	0.1081	127.4
OCSVM	0.0273	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
XGBOD	**0.8737**	**0.9942**	**0.7793**	4,256.3

Requires labels

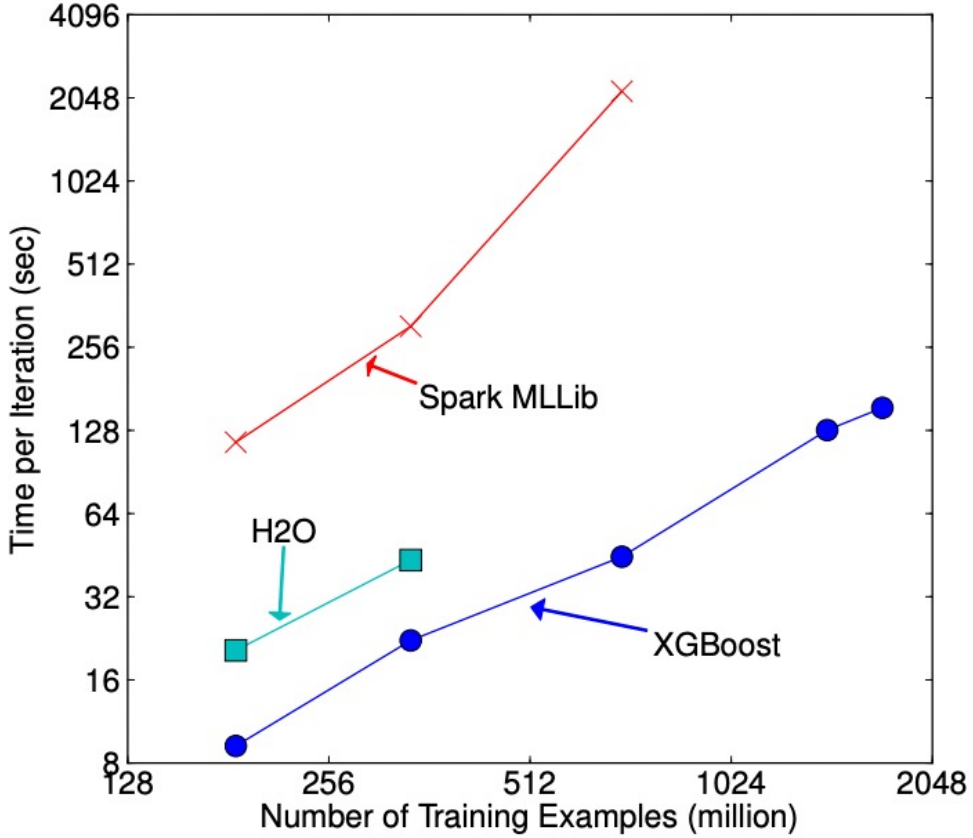
The best results are in bold, # The experiments were performed on GPU NVIDIA GeForce GTX 1060 6GB, 1280 cores on a Windows 10 oper. system in the Python library PyOD

Hajek, P., Abedin, M. Z., & Sivarajah, U. (2022). Fraud Detection in Mobile Payment Systems using an XGBoost-based Framework. *Information Systems Frontiers*, 1-19.

Which boosting system?



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



(b) Per iteration cost exclude data loading

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data min ing* (pp. 785-794). https://dl.acm.org/doi/pdf/10.1145/2939672.2939785

Table 1: Comparison of major tree boosting systems.

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

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data min ing* (pp. 785-794). https://dl.acm.org/doi/pdf/10.1145/2939672.2939785

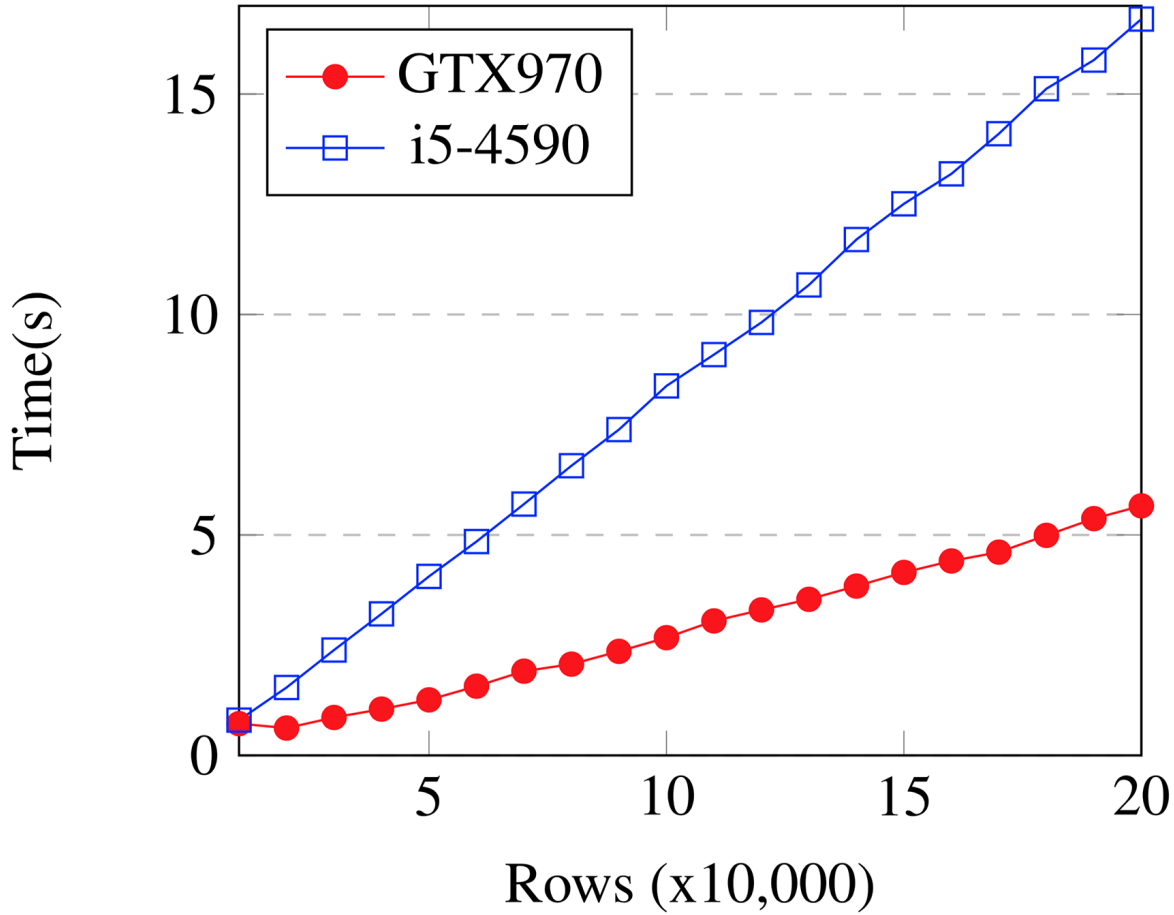


Fig. Time vs Problem size

Mitchell, R., & Frank, E. (2017). Accelerating the XGBoost algorithm using GPU computing. *PeerJ Computer Science*, *3*, e127.

Datasets.

Dataset	Training instances	Test instances	Features
YLTR[a]	473,134	165,660	700
Higgs[b]	10,500,000	500,000	28
Bosch[c]	1,065,373	118,374	968

DOI: 10.7717/peerj.cs-127/table-20

Notes:

a https://webscope.sandbox.yahoo.com/catalog.php?datatype=c.

b https://archive.ics.uci.edu/ml/datasets/HIGGS.

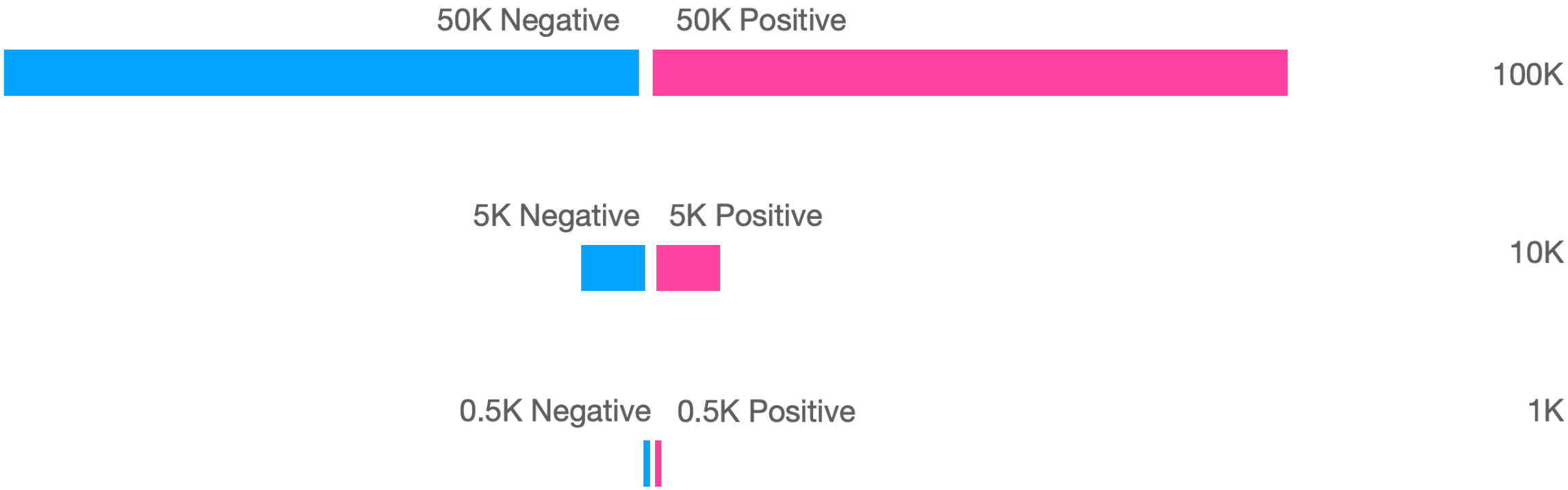
c https://www.kaggle.com/c/bosch-production-line-performance/data.

Dataset	Subset	CPU time (s)	GPU time (s)	Speedup
YLTR	1.0	877	277	3.16
Higgs	1.0	14,504	3,052	4.75
Bosch	1.0	3,294	591	5.57

Mitchell, R., & Frank, E. (2017). Accelerating the XGBoost algorithm using GPU computing. *PeerJ Computer Science*, *3*, e127.

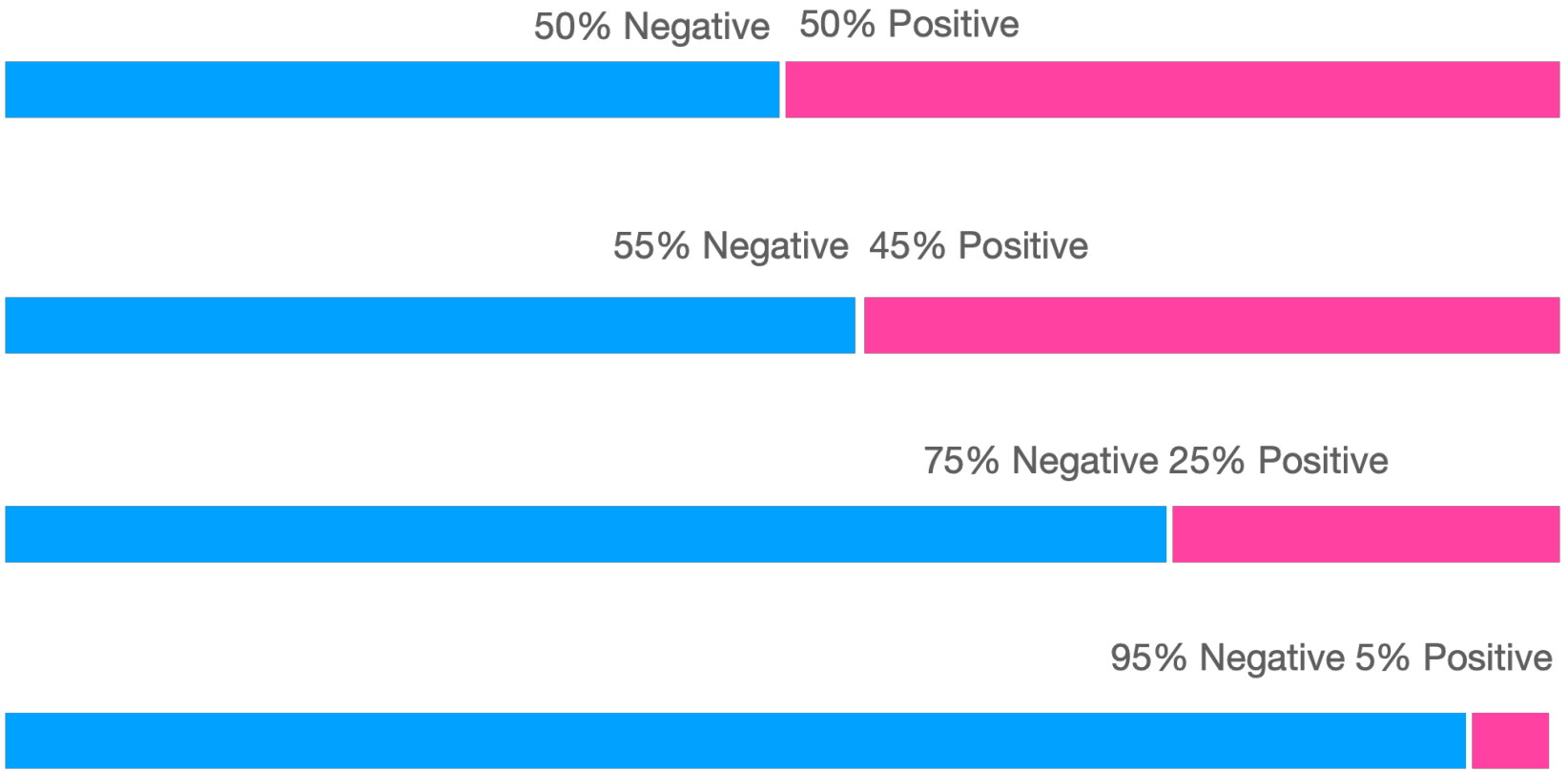
Experiments

Experiment 1



On a private dataset with more than 150 features

For 1K, 10k and 100K samples



Balanced

Imbalanced

On a private dataset with more than 150 features

Pipeline

Numerical data scaled between 0 and 1

Categorical data under Ordinal Encoder, such that values unseen during training received a reserved value

Vanilla XGBoost

```
from xgboost import XGBClassifier


model = XGBClassifier(
    objective='binary:logistic’,
    missing=1,
    random_state = 100,
    eval_metric='aucpr’,
    tree_method='gpu_hist')
```

Vanilla XGBoost

Train – Test:
80% - 20%

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

Randomized search over following parameters

```
s_parameters = {
        'XGBoostcl__max_depth': [3, 6, 12, 20],
        'XGBoostcl__learning_rate': [0.02, 0.1, 0.2],
        'XGBoostcl__subsample': [0.4, 0.8, 1],
        'XGBoostcl__colsample_bytree': [0.4, 0.6, 1],
        'XGBoostcl__n_estimators': [100,1000,5000]
         }
```

Vanilla parameters

```
'max_depth': 6
'learning_rate': 0.3
'subsample': 1
'colsample_bytree': 1
'n_estimators': 100
```

Why searching over these parameters?

McDonald and Deotte. Leveraging Machine Learning to Detect Fraud: Tips to Developing a Winning Kaggle Solution (2021)

RS-Tuned XGBoost

Train – Test:
80% - 20%

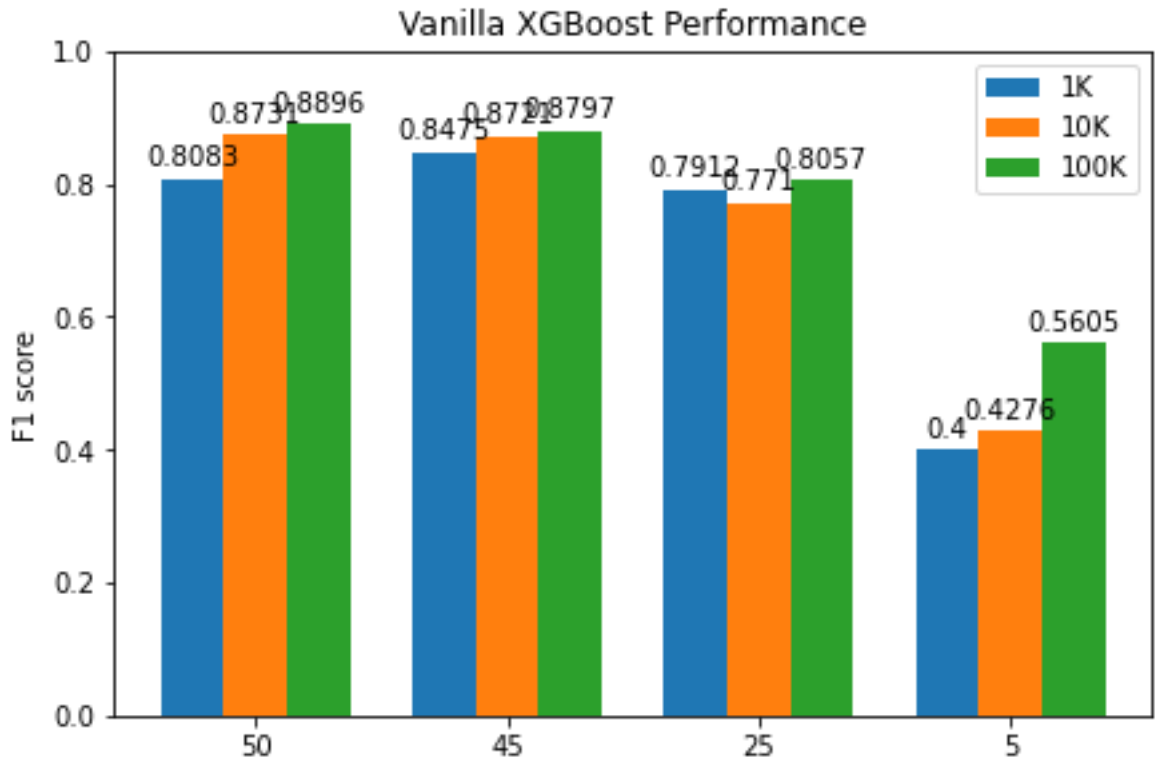
1K*	Positive %	Precision	Recall	F1	F05
	50	0.8593	0.7933	0.8249	0.8452
	45	0.8501	0.8185	0.8339	0.8435
	25	0.8582	0.6867	0.7612	0.816
	5	0.3595	0.2000	0.2542	0.3068
10K	**Positive %**	**Precision**	**Recall**	**F1**	**F05**
	50	0.8739	0.8870	0.8804	0.8765
	45	0.8713	0.8722	0.8717	0.8714
	25	0.8162	0.7640	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

Comparison

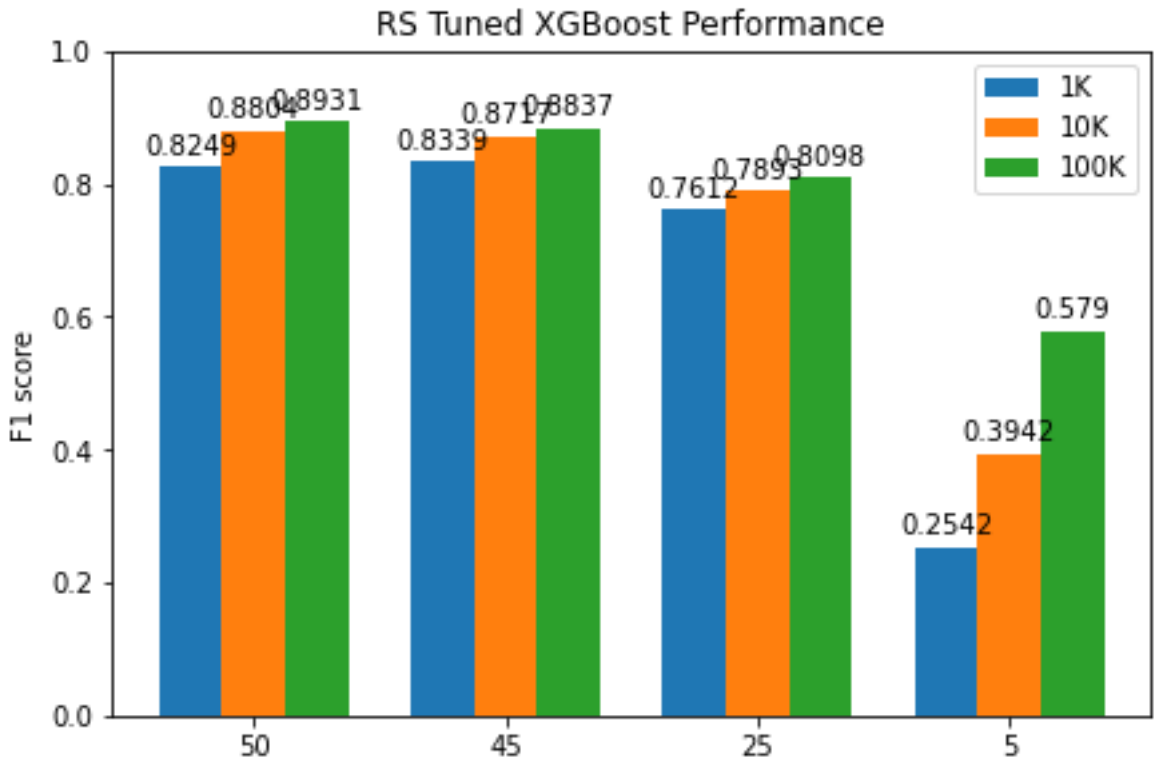
		Vanilla	RS-Tuned parameters
1K*	**Positive %**	**F1**	**F1**
	50	0.8083	**0.8249±0.00**
	45	**0.8475**	0.8339±0.00
	25	**0.7912**	0.7612±0.03
	5	**0.4000**	0.2542±0.11
10K	**Positive %**	**F1**	**F1**
	50	0.8731	**0.8804**
	45	**0.8721**	0.8717
	25	0.7710	**0.7893**
	5	**0.4276**	0.3942
100K	**Positive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

*Run 3 times.

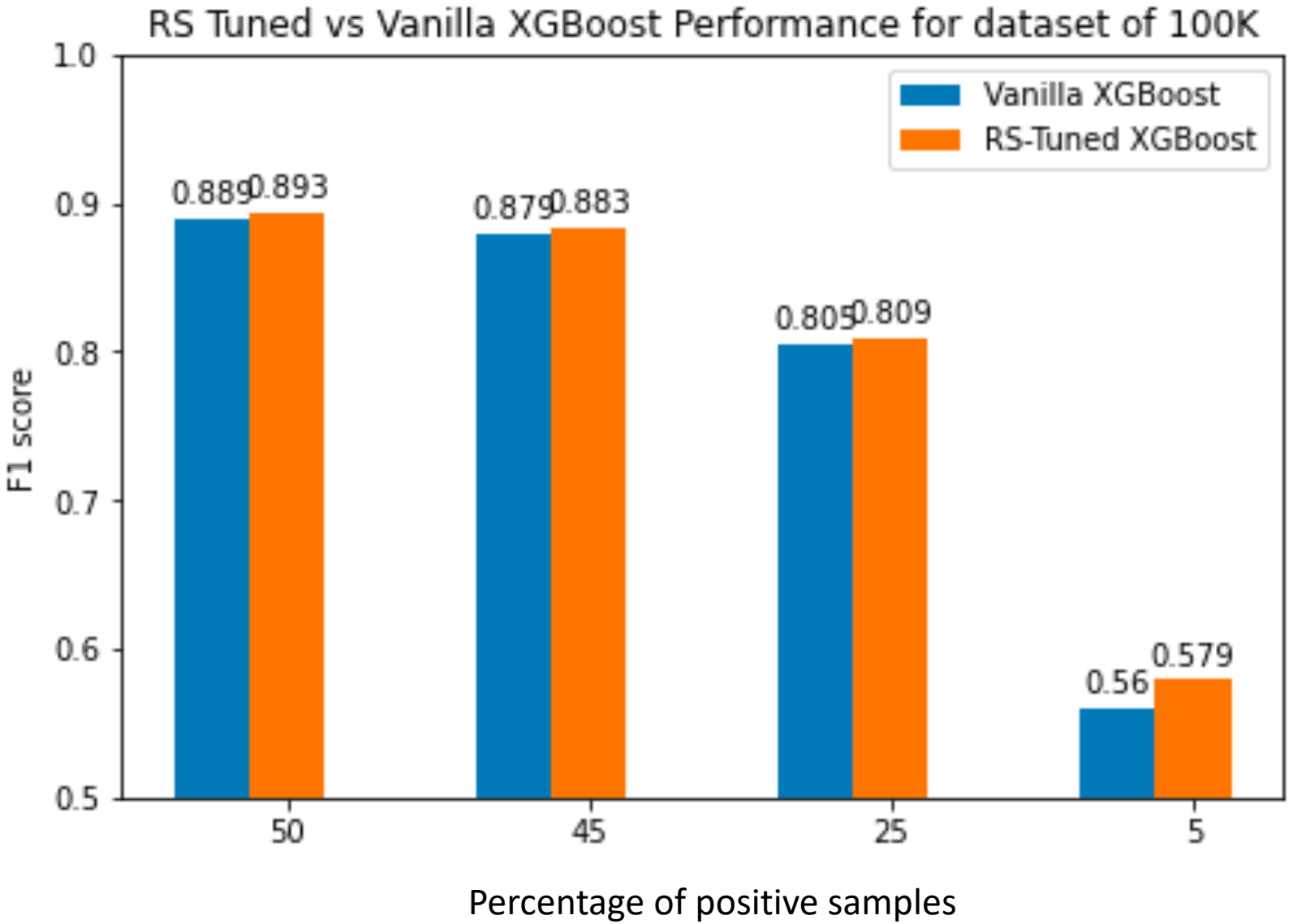
There is only a benefit when the dataset is large



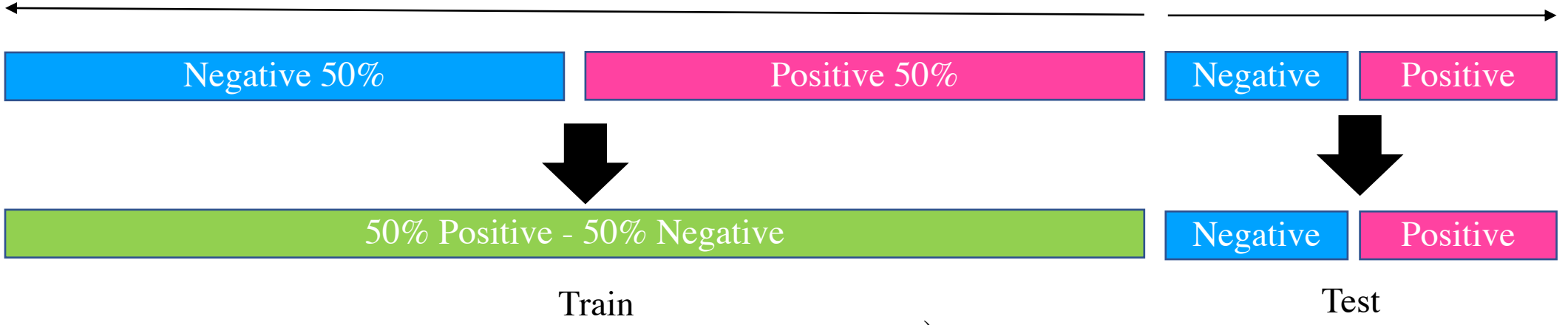
Percentage of positive samples



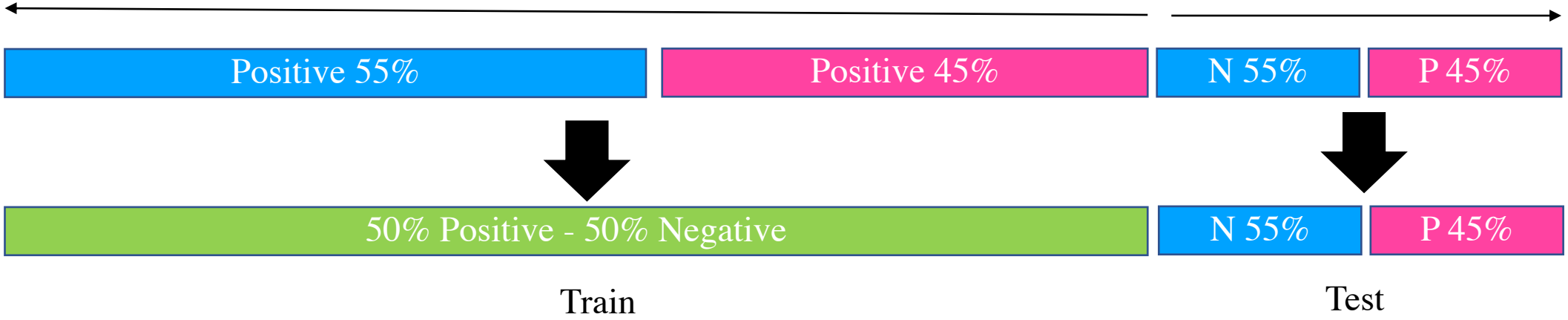
Percentage of positive samples


RS Tuned vs Vanilla XGBoost Performance for dataset of 100K
Vanilla XGBoost
RS-Tuned XGBoost
F1 score
1.0
0.9
0.8
0.7
0.6
0.5
0.889
0.893
0.879
0.883
0.805
0.809
0.56
0.579
50
45
25
5
Percentage of positive samples

Experiment 2



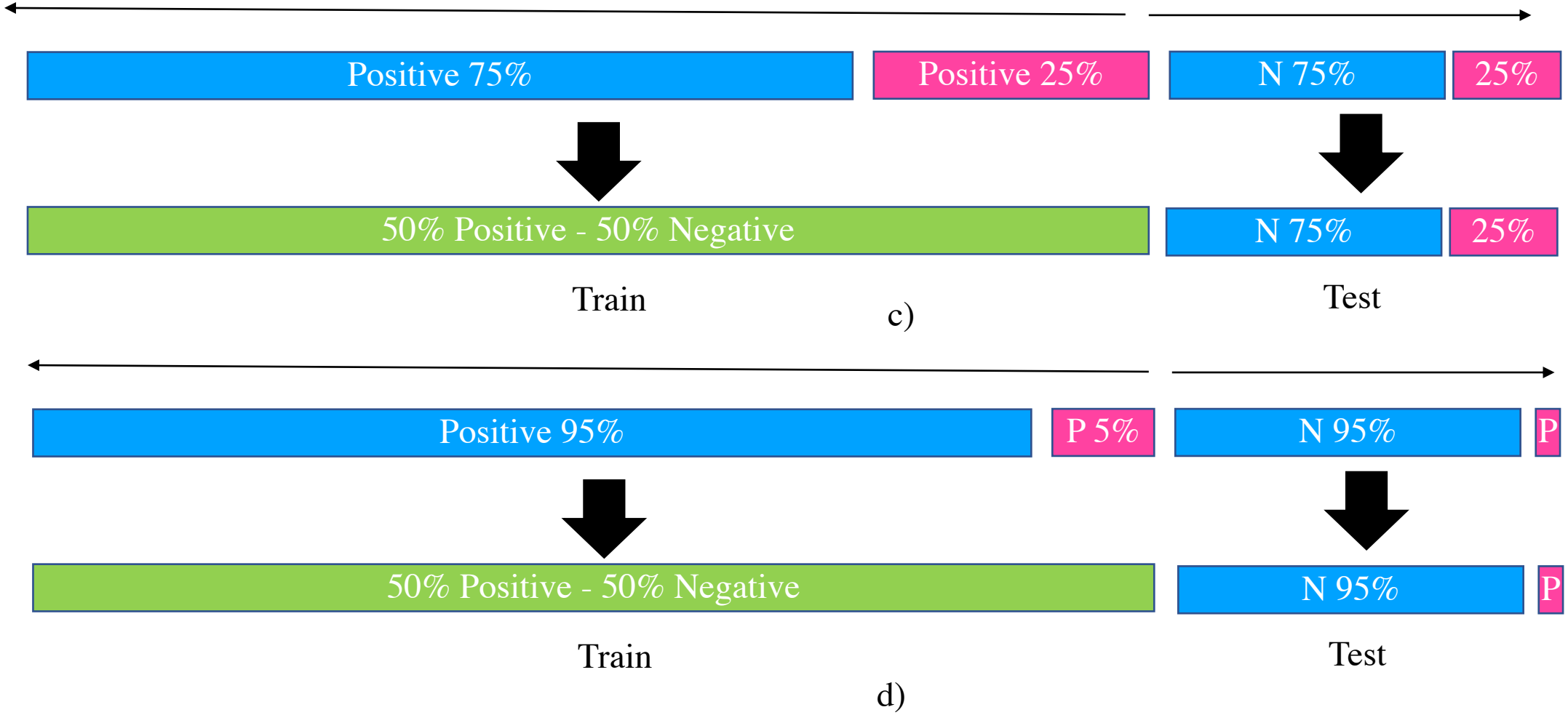
a)



b)

On a private dataset with more than 150 features

Experiment 2



On a private dataset with more than 150 features

Random Sampling to train with a balanced dataset RS-tuned XGBoost

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

The estimated F1 is unrealistic!

Recall improves but Precision worsens

Comparison

		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**

Next steps

- Graphs seem promising
- Review methods for imbalanced learning
- Deep Learning and Representation Learning
- Triton and Morpheus

Where to look now…

XGBoost Documentation

RecSys 2020 Tutorial: Feature Engineering for Recommender Systems

Gissel.Velarde@Vodafone.com