National Research
Tomsk
State
University

Лаборатория
анализа данных
физики высоких энергий
Томского
государственного
университета

Measurement of differential cross-sections of a single top quark produced in association with a *W* boson with ATLAS at $\sqrt{s}$ = 13 TeV

Progress Report

Neda Firoz

Goal: separate **tW (top+anti-top)** from **$t\bar{t}$** in the **1j1b dilepton** region

•**Inputs (9 variables used):**
bdt_centrality_ll_recalc_NOSYS,
bdt_delta_pT_ll_MET_recalc_NOSYS,
bdt_delta_pT_llb_MET_recalc_NOSYS,
bdt_eta_llMetB_recalc_NOSYS,
bdt_m_l1b_recalc_NOSYS,
bdt_m_l2b_recalc_NOSYS,
bdt_pT_llMetB_recalc_NOSYS,
bdt_pT_llb_recalc_NOSYS,
bdt_sum_ET_recalc_NOSYS.

•**Samples / tree:** all files' tree name is analysis

Signal: tW (top) + $\bar{t}$W (anti-top)
Background: $t\bar{t}$ (non-all-had)

•**Event weights:** auto-resolved to weight_mc_NOSYS * weight_pileup_NOSYS * globalTriggerEffSF_NOSYS.

•**Bad values:** any variable ≤ **−990** or non-finite is masked per event.

Reference Article's Report on BDT

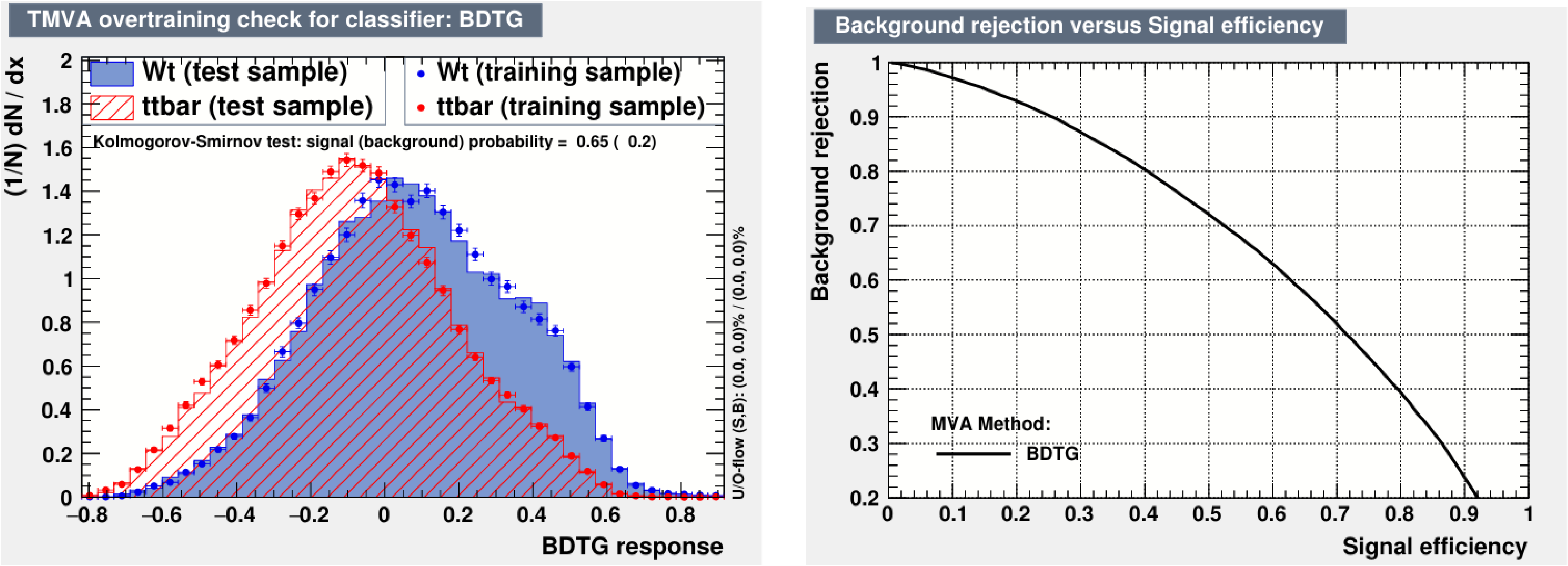


Figure 9: Comparison of test/training sample distributions and background rejection factor versus signal efficiency.

New Results for Python after Hyper parameter optimization by random search (since grid search was taking too long for condor)

Automated hyperparameter optimization and unbiased model comparison (16 algorithms)
Slide content:

- Load weighted signal and background events from ROOT
- Use identical input features for all algorithms (9)
- Define algorithm-specific hyperparameter search spaces
- Perform random-search HPO in inner CV using weighted AUC
- Evaluate selected model in outer CV for unbiased performance
- Aggregate out-of-fold predictions across all events
- Compare algorithms using AUC, weighted accuracy, S/B, and $S/\sqrt{S+B}$

Hyperparameter optimization strategy:
Search space defined per algorithm
Random search using ParameterSampler
Inner 3-fold CV selects best hyperparameters by weighted AUC
Outer 5-fold CV evaluates tuned model on unseen data
Out-of-fold predictions used for final ROC and physics plots
HPO history saved for reproducibility and auditability

"The optimization is automated, but constrained within expert-defined hyperparameter ranges."

Algorithms	overall_oof_auc	overall_oof_wacc	mean_fold_auc	std_fold_auc	mean_fold_wacc	std_fold_wacc	max_significance_tmva_like	optimal_cut_tmva_like	cpu_sec_nested	wall_sec_nested	n_iter	outer_k	inner_k
CatBoost	0.677358	0.623951	0.67777	0.009907	0.623949	0.009367	22.53319	-0.37517	4597.23	2295.79	25	5	3
GradientBoosting	0.67482	0.62379	0.675134	0.01051	0.623787	0.0108	22.39555	-0.36898	8060.265	8100.366	25	5	3
ExtraTrees	0.673411	0.619695	0.673515	0.010477	0.619693	0.009589	22.47794	-0.2761	3092.425	1677.636	25	5	3
RandomForest	0.67238	0.620409	0.673045	0.010076	0.620407	0.010415	22.52224	-0.29845	14168.95	7175.542	25	5	3
SVC	0.671871	0.622698	0.672906	0.011466	0.622696	0.014309	22.30438	-0.35871	2126.425	2136.84	12	5	3
HistGradientBoosting	0.668984	0.61906	0.670825	0.011496	0.619058	0.013581	22.33833	-0.27489	203.7338	107.3245	25	5	3
MLP	0.660661	0.51298	0.660391	0.024628	0.512979	0.009094	22.50213	-0.83959	736.7484	393.3603	25	5	3
AdaBoost	0.660288	0.620915	0.667745	0.008661	0.620913	0.008934	22.15322	-0.03786	1994.595	2005.079	25	5	3
KNN	0.655979	0.50016	0.656032	0.009047	0.50016	0.000391	22.40944	-0.88251	259.921	261.1354	25	5	3
SGDClassifier	0.654416	0.576342	0.654547	0.011232	0.576341	0.004277	22.02061	-0.62842	28.44219	28.59732	25	5	3
Bagging	0.653939	0.505979	0.654744	0.008789	0.50598	0.00354	22.37251	-0.8938	72.2342	2875.612	25	5	3
QDA	0.651292	0.520597	0.6514	0.006987	0.520597	0.001775	22.32001	-0.89	2.364426	1.200313	25	5	3
GaussianNB	0.639867	0.612395	0.6407	0.006545	0.612393	0.00922	22.32842	-0.22929	0.49265	0.498028	25	5	3
LogisticRegression	0.639509	0.603746	0.639506	0.011228	0.603744	0.013164	22.24308	-0.2384	9.065713	4.57647	25	5	3
LinearSVC	0.637216	0.600697	0.637195	0.011291	0.600695	0.011075	22.23599	-0.06938	4.114012	4.143002	25	5	3
LDA	0.631748	0.501218	0.631761	0.010358	0.501218	0.000734	22.2711	-0.82242	2.817813	1.448922	25	5	3

Python Results Analysis

1. Best algorithm by key metric

A) Best OOF AUC
Ranking from the table:
CatBoost = 0.677358
GradientBoosting = 0.674820
ExtraTrees = 0.673411
RandomForest = 0.672380
SVC = 0.671871

CatBoost is the top performer in discrimination power, but the margin over GradientBoosting and ExtraTrees is **small**, so the top tree-based models are all competitive.

B) Best by max significance
Top values:
CatBoost = 22.53319
RandomForest = 22.52224
MLP = 22.50213
ExtraTrees = 22.47794
KNN = 22.40944

CatBoost is again best. Interesting point: **MLP is not among the best in** AUC, but still gives high max significance after a cut, so it may produce a useful signal-enriched region despite weaker overall classification quality.

Python Results Analysis

2. Best by computational performance

A) Fastest by CPU time

Fastest models:
GaussianNB = 0.493 s
QDA = 2.364 s
LDA = 2.818 s
LinearSVC = 4.114 s
LogisticRegression = 9.066 s

B) Fastest by wall time

Fastest models:
GaussianNB = 0.498 s
QDA = 1.200 s
LDA = 1.449 s
LinearSVC = 4.143 s
LogisticRegression = 4.576 s

The fastest models are **GaussianNB, QDA, and LDA**, but they are **not the strongest in AUC**.
So, they are computationally efficient baselines, not the best final classifiers.

Python Results Analysis

3. Best balance of performance and speed:

HistGradientBoosting is the best compromise.
Why:
OOF AUC = 0.668984, which is still reasonably strong
CPU = 203.7 s
Wall = 107.3 s

Best absolute performance
CatBoost
Best practical tradeoff between accuracy and runtime
HistGradientBoosting

Python Results Analysis

4. Computationally Expensive algorithms

A) By CPU time
RandomForest = 14168.95 s
GradientBoosting = 8060.27 s
CatBoost = 4597.23 s
ExtraTrees = 3092.43 s
SVC = 2126.43 s

B) By wall time
GradientBoosting = 8100.37 s
RandomForest = 7175.54 s
Bagging = 2875.61 s
CatBoost = 2295.79 s
SVC = 2136.84 s
AdaBoost = 2005.08 s
ExtraTrees = 1677.64 s

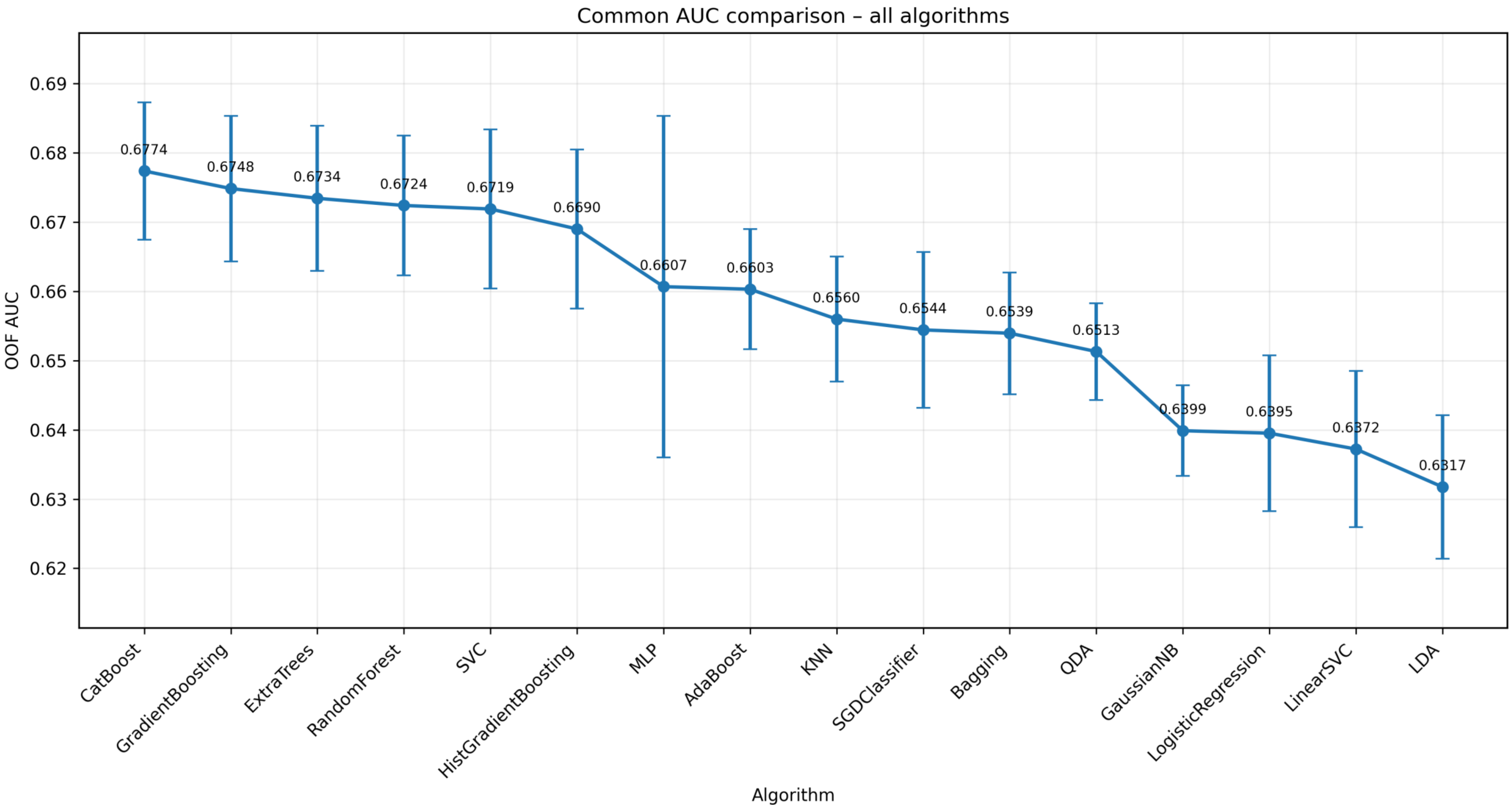
Conclusion:
RandomForest and **GradientBoosting** are very expensive.
Bagging is also expensive relative to its modest performance.
CatBoost is costly, but its performance justifies the cost much more than Bagging or RandomForest.

Python Results Analysis

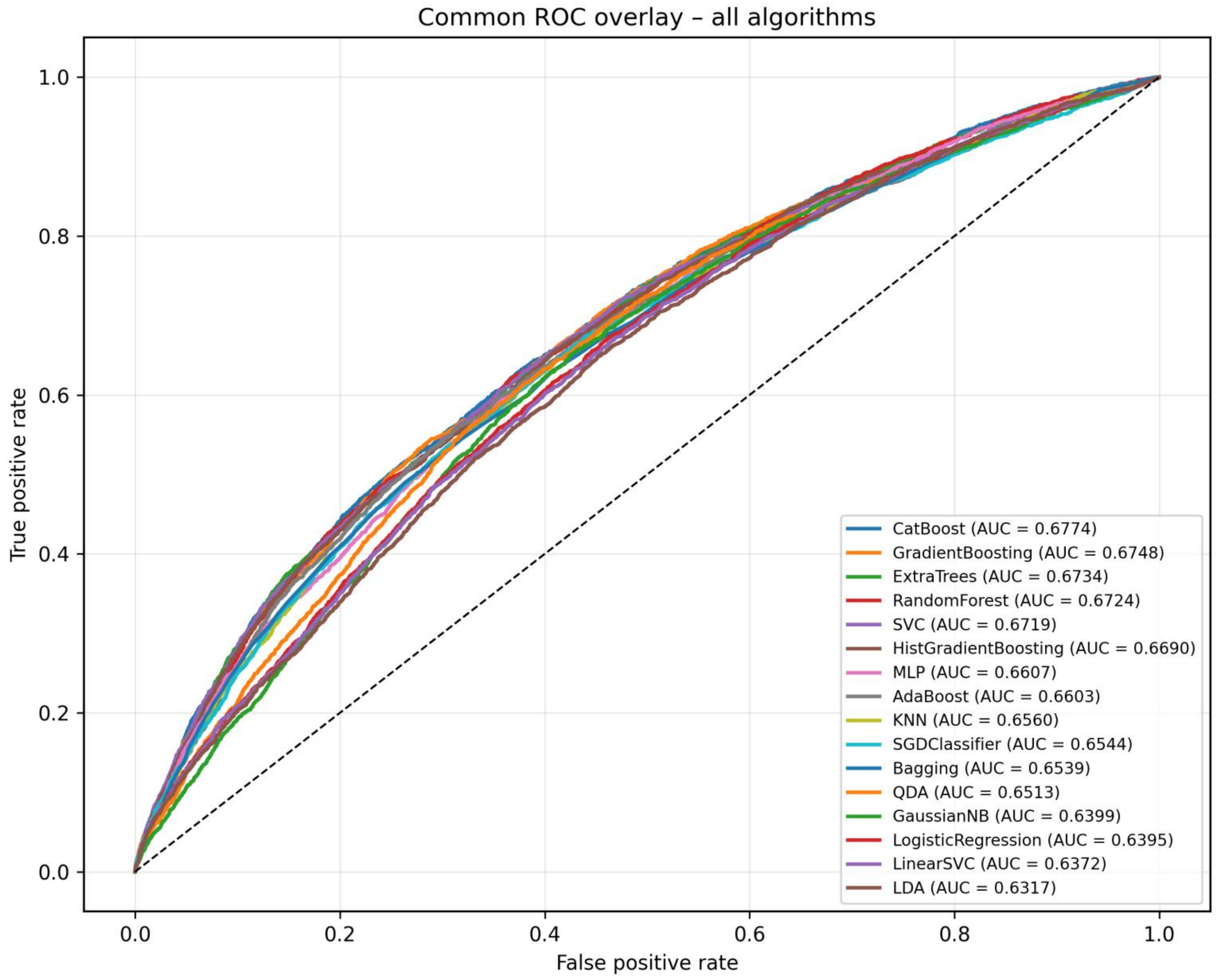
Stability across folds

A lower std_fold_auc means more stable performance across CV folds.
Among strong models:
CatBoost = **0.009907**
ExtraTrees = **0.010477**
GradientBoosting = **0.010510**
RandomForest = **0.010076**
SVC = **0.011466**

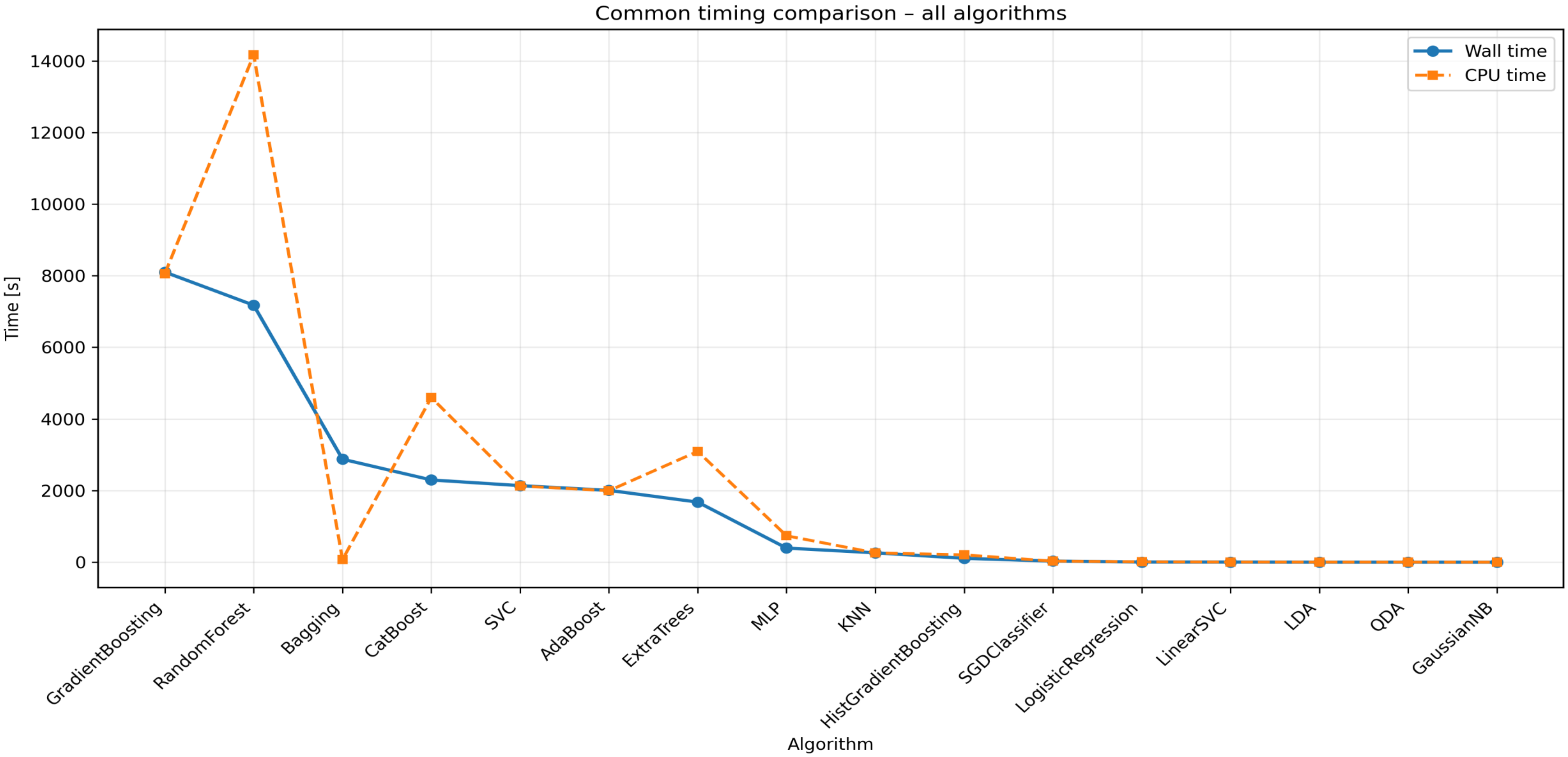
The top models are all reasonably stable. CatBoost is not only best in AUC, it is also **stable across folds**, which strengthens confidence in the result.

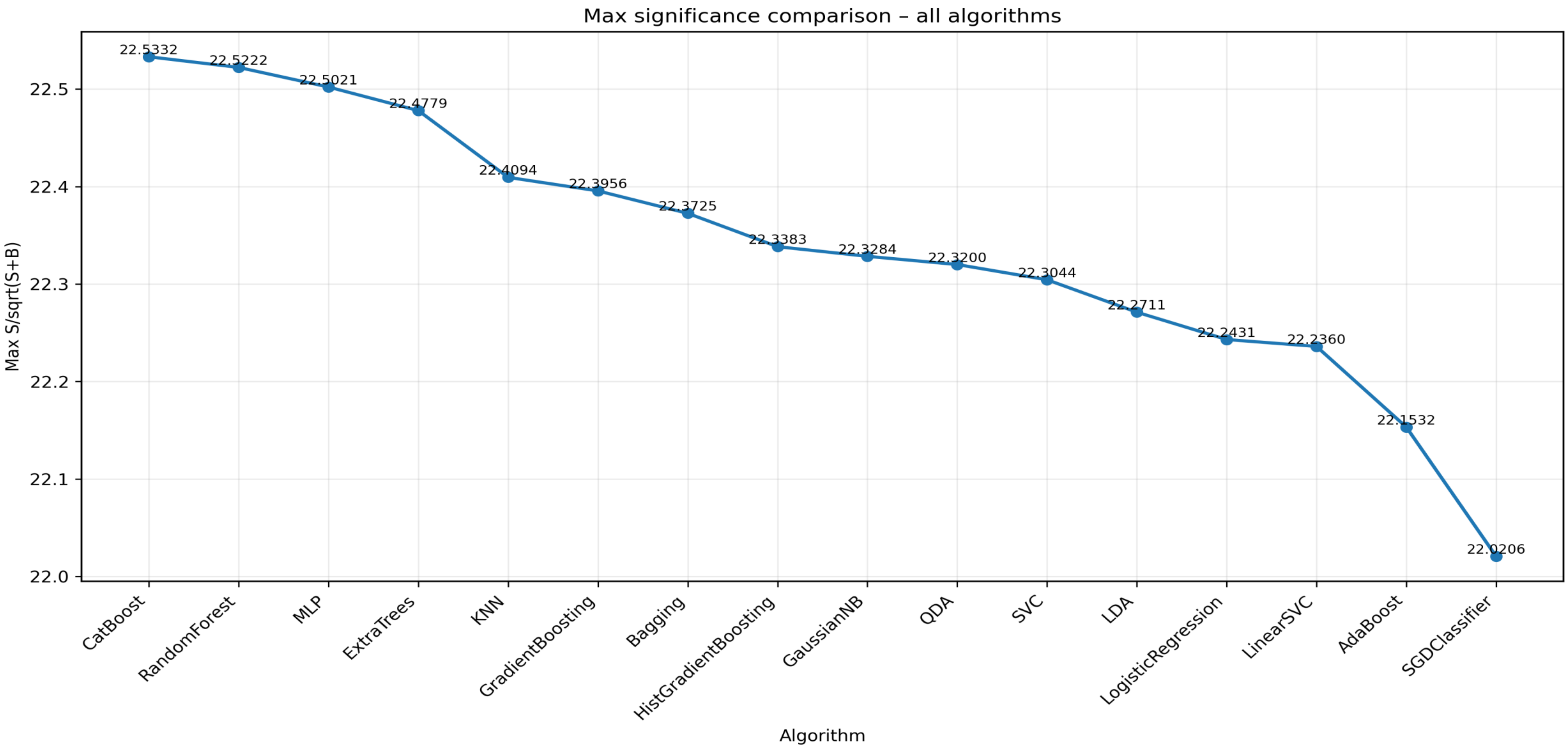


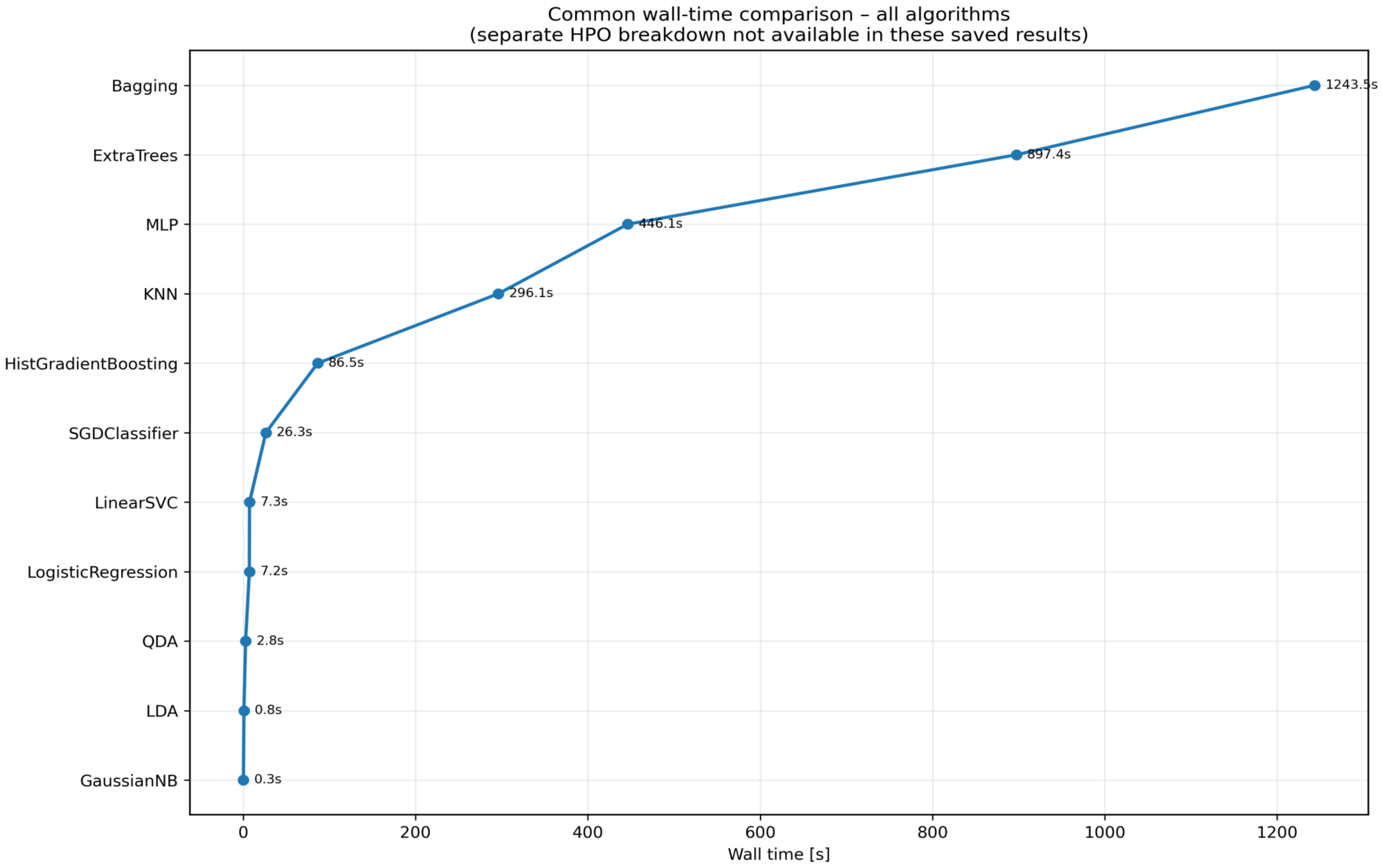
Common AUC comparison – all algorithms
OOF AUC
0.69
0.68
0.67
0.66
0.65
0.64
0.63
0.62
0.6774
0.6748
0.6734
0.6724
0.6719
0.6690
0.6607
0.6603
0.6560
0.6544
0.6539
0.6513
0.6399
0.6395
0.6372
0.6317
CatBoost
GradientBoosting
ExtraTrees
RandomForest
SVC
HistGradientBoosting
MLP
AdaBoost
KNN
SGDClassifier
Bagging
QDA
GaussianNB
LogisticRegression
LinearSVC
LDA
Algorithm

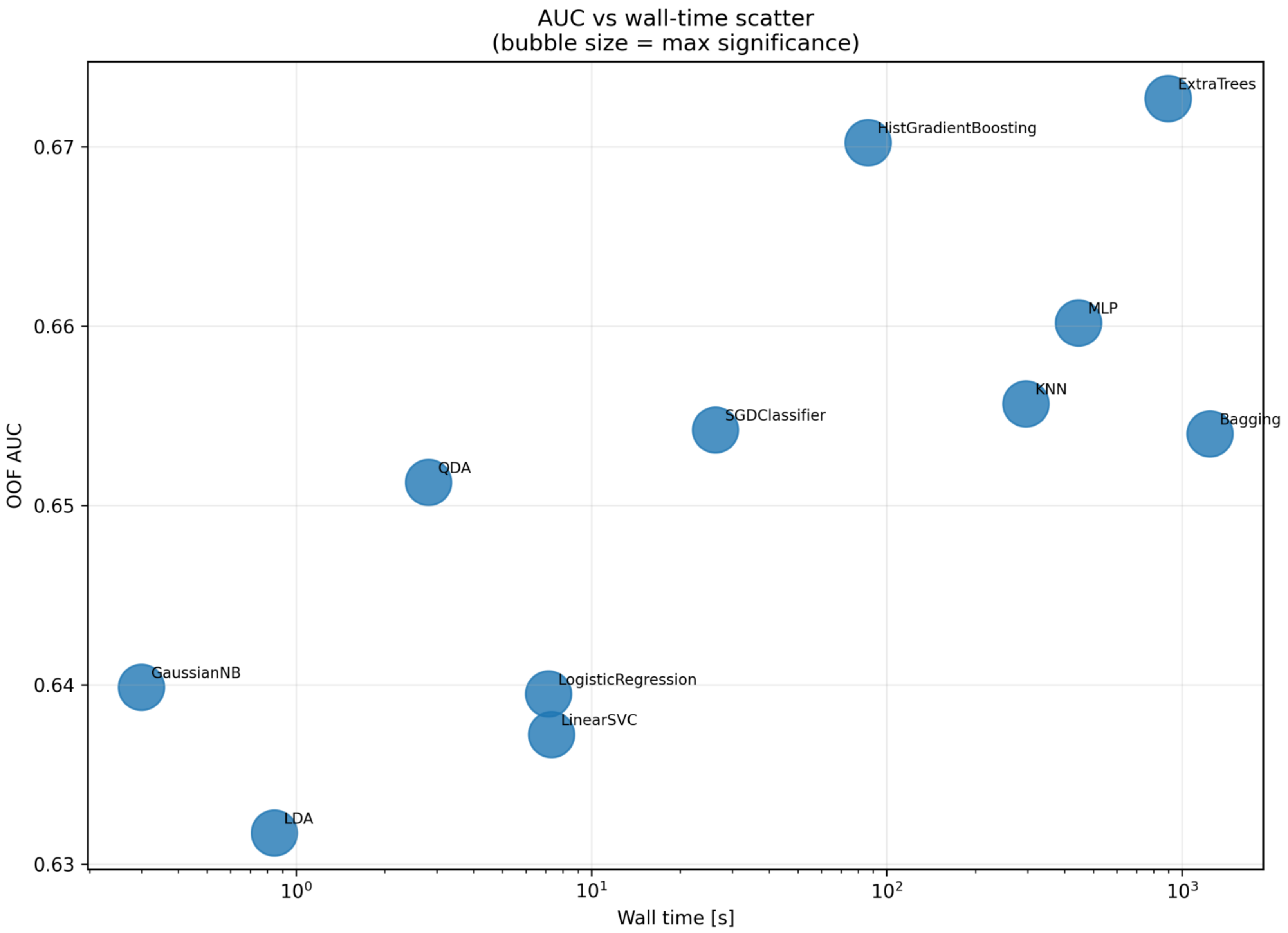


Common ROC overlay – all algorithms
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
CatBoost (AUC = 0.6774)
GradientBoosting (AUC = 0.6748)
ExtraTrees (AUC = 0.6734)
RandomForest (AUC = 0.6724)
SVC (AUC = 0.6719)
HistGradientBoosting (AUC = 0.6690)
MLP (AUC = 0.6607)
AdaBoost (AUC = 0.6603)
KNN (AUC = 0.6560)
SGDClassifier (AUC = 0.6544)
Bagging (AUC = 0.6539)
QDA (AUC = 0.6513)
GaussianNB (AUC = 0.6399)
LogisticRegression (AUC = 0.6395)
LinearSVC (AUC = 0.6372)
LDA (AUC = 0.6317)


Common timing comparison – all algorithms
Wall time
CPU time
Time [s]
14000
12000
10000
8000
6000
4000
2000
0
GradientBoosting
RandomForest
Bagging
CatBoost
SVC
AdaBoost
ExtraTrees
MLP
KNN
HistGradientBoosting
SGDClassifier
LogisticRegression
LinearSVC
LDA
QDA
GaussianNB
Algorithm

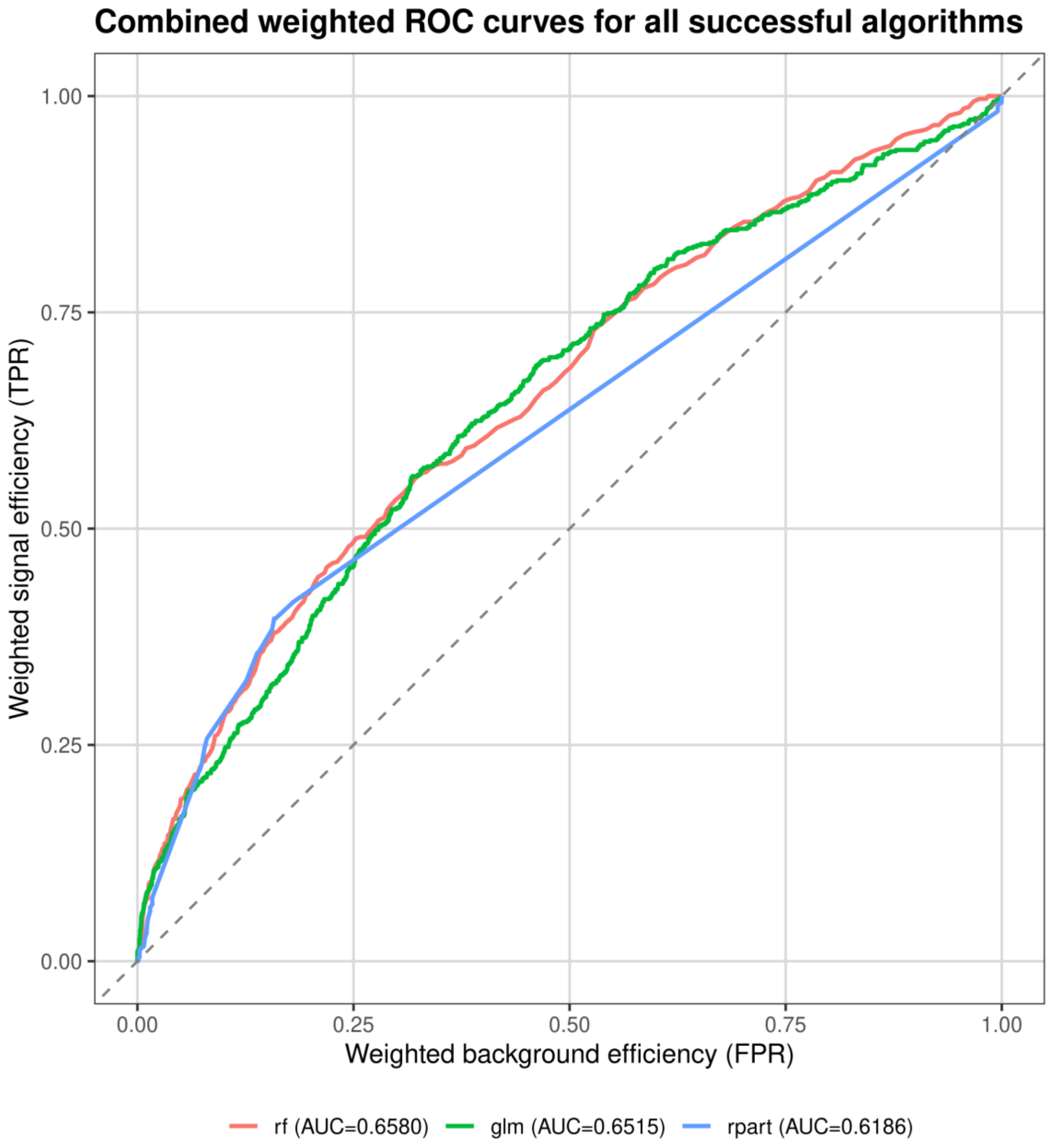

Max significance comparison – all algorithms
Max S/sqrt(S+B)
Algorithm
22.0
22.1
22.2
22.3
22.4
22.5
22.5332
22.5222
22.5021
22.4779
22.4094
22.3956
22.3725
22.3383
22.3284
22.3200
22.3044
22.2711
22.2431
22.2360
22.1532
22.0206
CatBoost
RandomForest
MLP
ExtraTrees
KNN
GradientBoosting
Bagging
HistGradientBoosting
GaussianNB
QDA
SVC
LDA
LogisticRegression
LinearSVC
AdaBoost
SGDClassifier


Common wall-time comparison – all algorithms
(separate HPO breakdown not available in these saved results)
Bagging
ExtraTrees
MLP
KNN
HistGradientBoosting
SGDClassifier
LinearSVC
LogisticRegression
QDA
LDA
GaussianNB
1243.5s
897.4s
446.1s
296.1s
86.5s
26.3s
7.3s
7.2s
2.8s
0.8s
0.3s
0
200
400
600
800
1000
1200
Wall time [s]

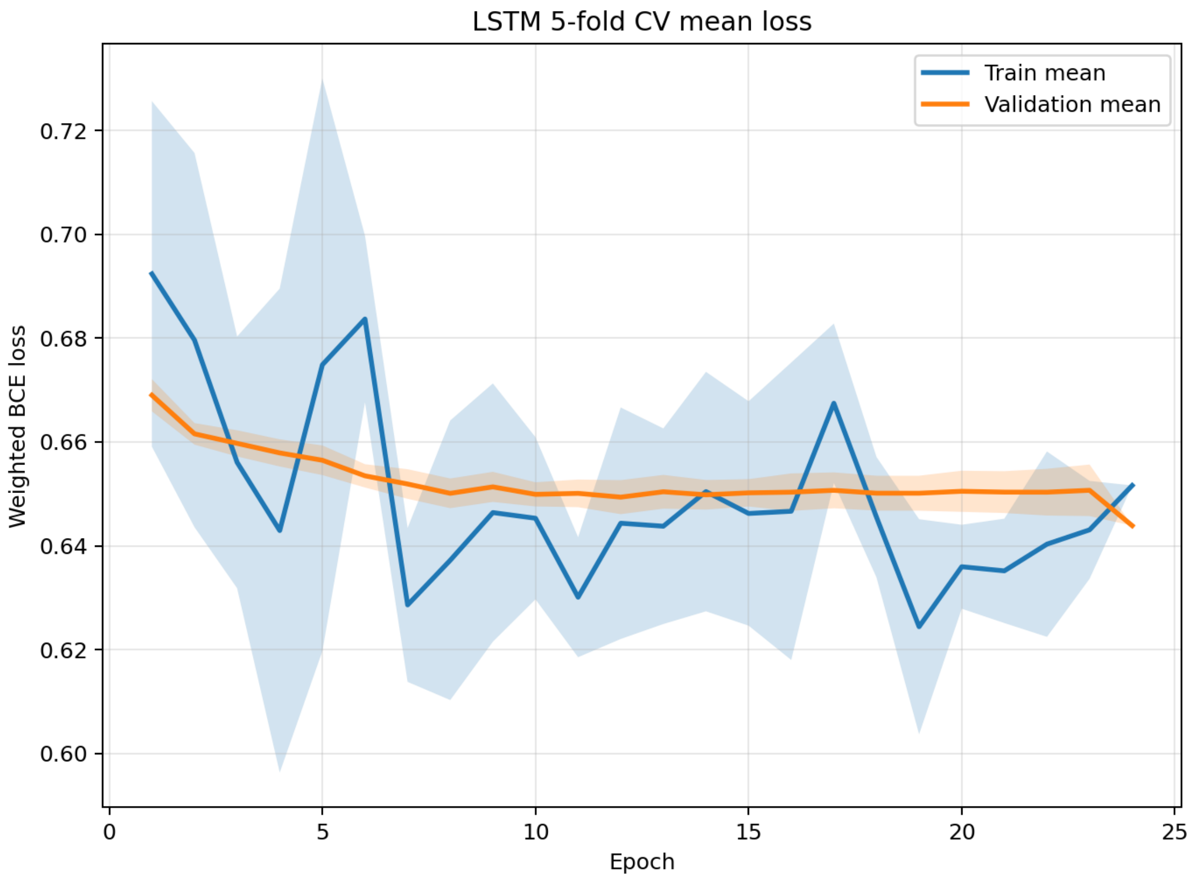

AUC vs wall-time scatter
(bubble size = max significance)
OOF AUC
Wall time [s]
0.67
0.66
0.65
0.64
0.63
10^0
10^1
10^2
10^3
ExtraTrees
HistGradientBoosting
MLP
KNN
Bagging
SGDClassifier
QDA
GaussianNB
LogisticRegression
LinearSVC
LDA

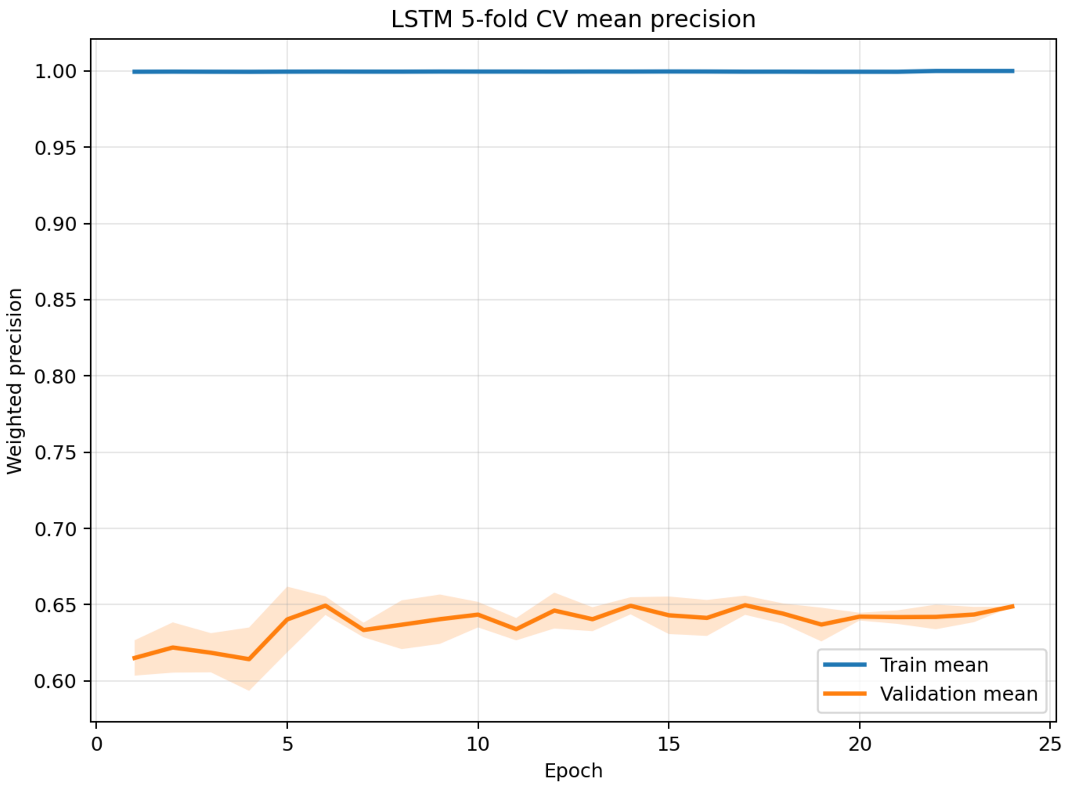
R

The package installation is under progress


Combined weighted ROC curves for all successful algorithms
1.00
0.75
0.50
0.25
0.00
Weighted signal efficiency (TPR)
0.00
0.25
0.50
0.75
1.00
Weighted background efficiency (FPR)
rf (AUC=0.6580)
glm (AUC=0.6515)
rpart (AUC=0.6186)

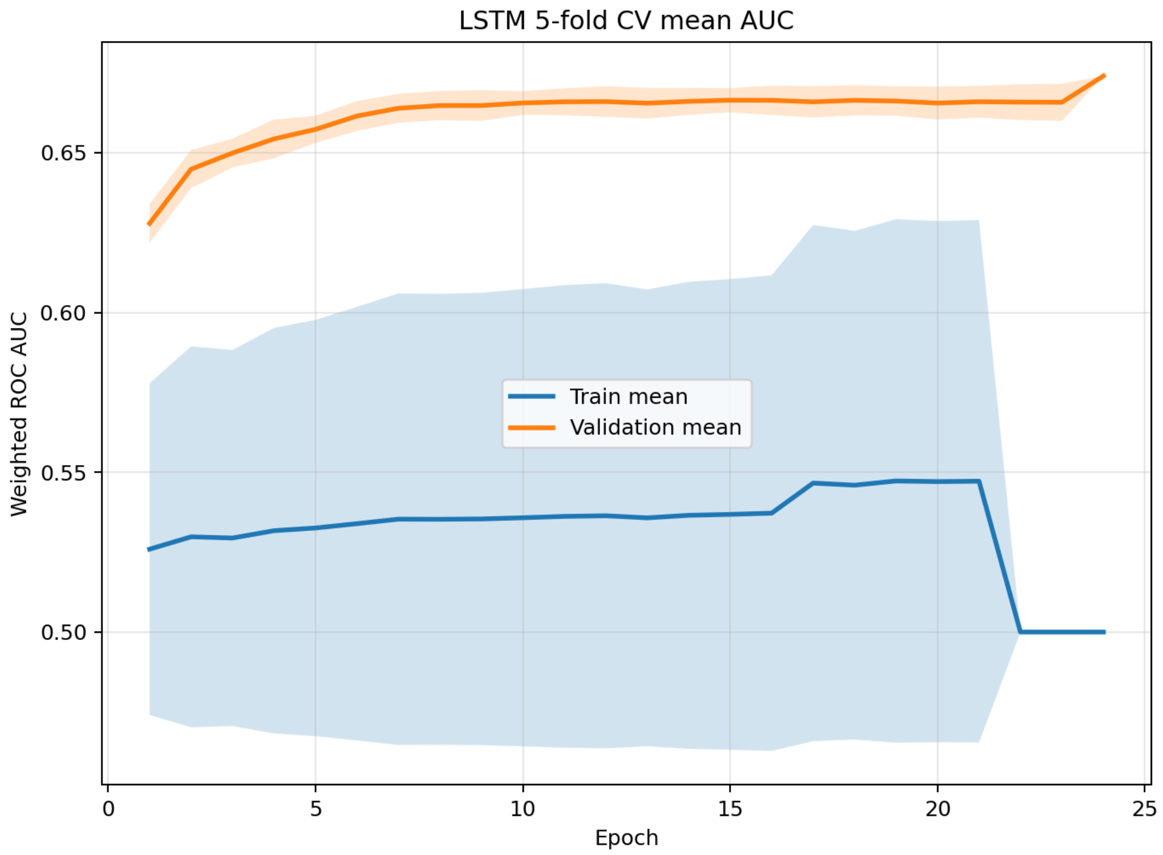
NEW LSTM
(4 layer denser)

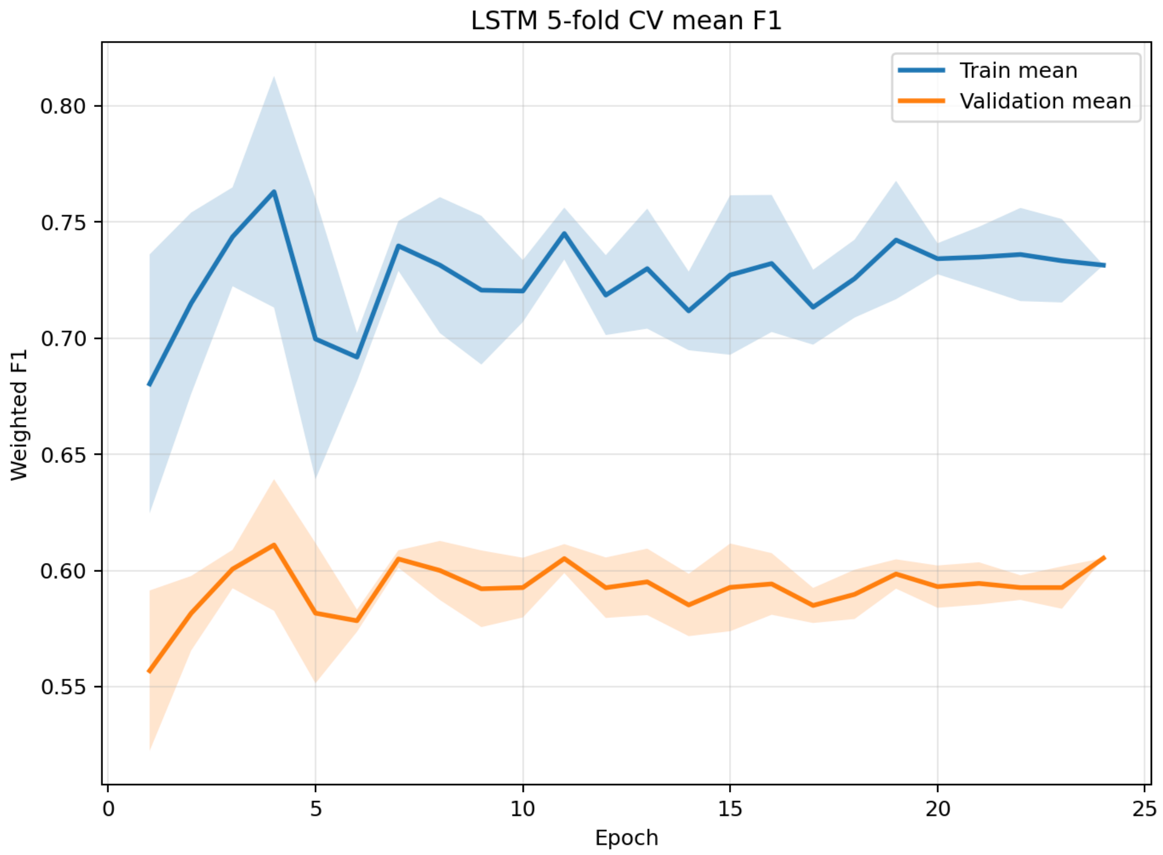


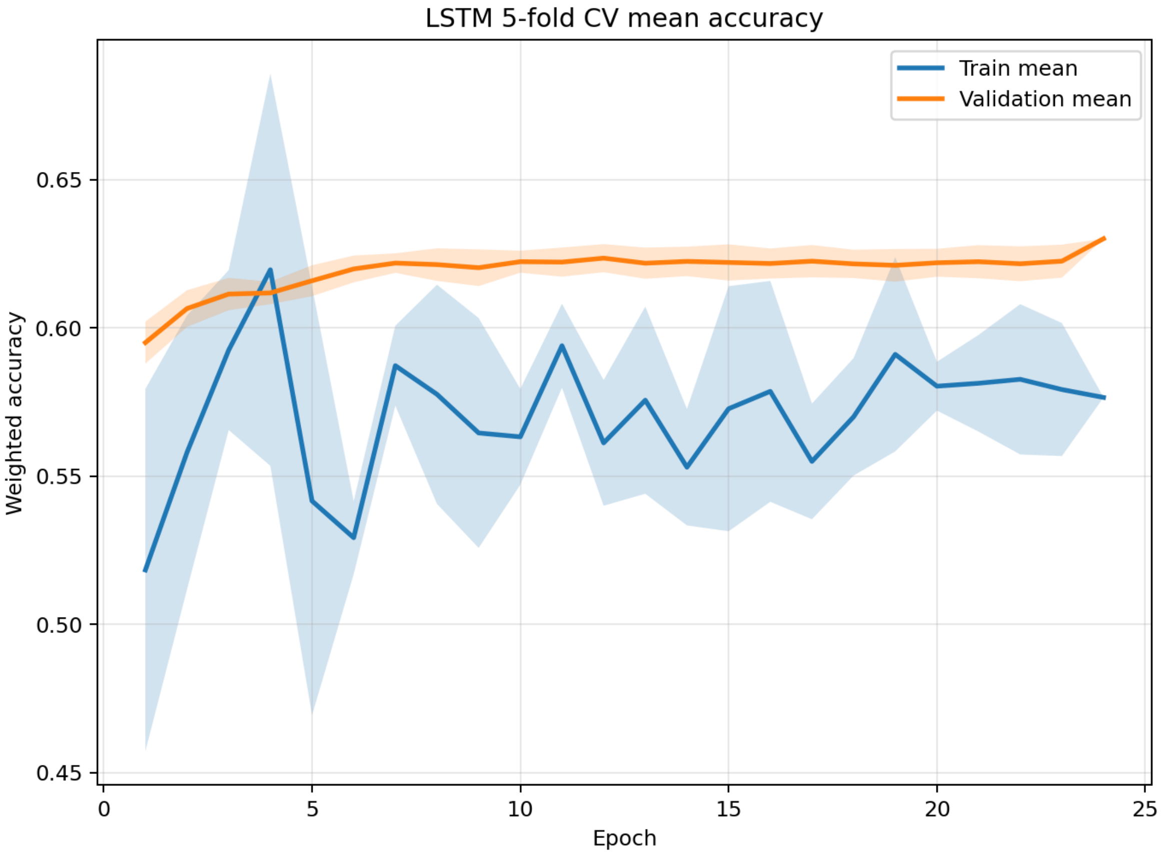


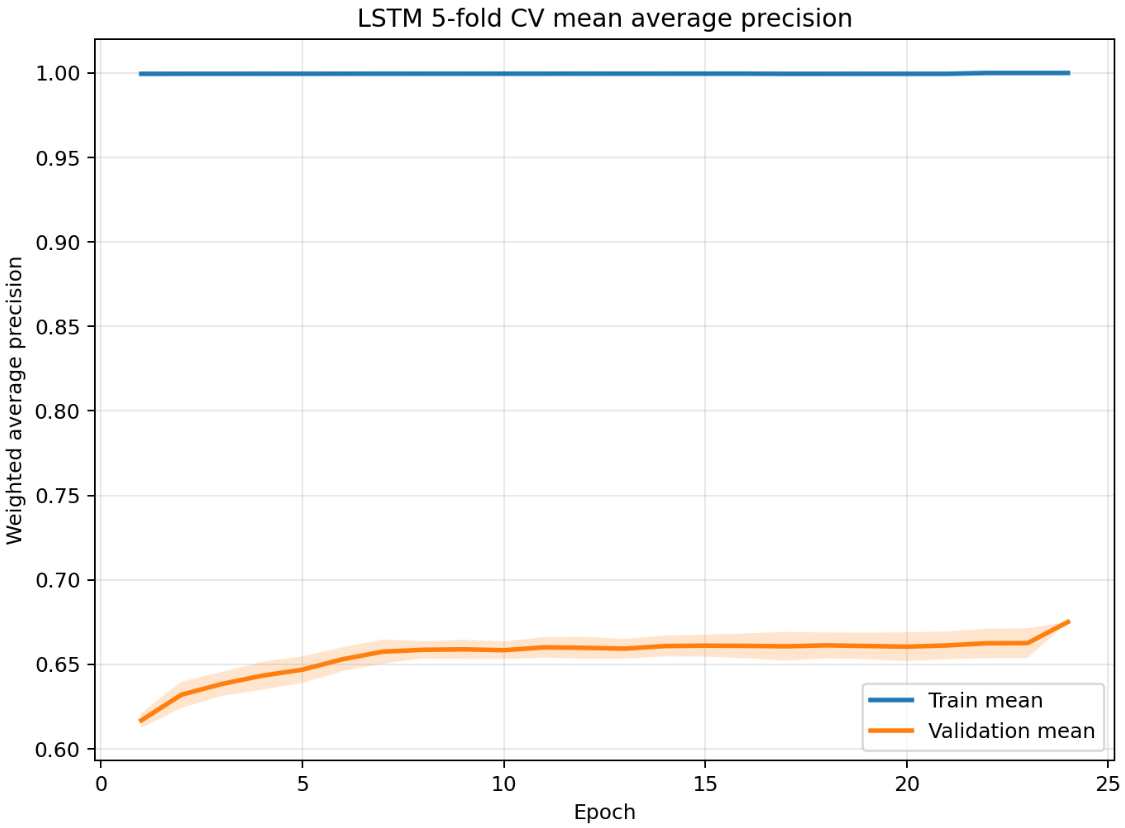
fold	runtime_sec	best_inner_auc	train_loss	val_loss	train_auc	val_auc	train_acc	val_acc	train_precision
0	12012.44	0.667684	0.643263	0.651301	0.677472	0.66755	0.629839	0.621597	0.661582
1	14619.3	0.66522	0.634585	0.651799	0.690159	0.663689	0.638558	0.61662	0.64502
2	11674.56	0.658742	0.646331	0.646985	0.672648	0.671132	0.626255	0.628862	0.65494
3	13968.54	0.666895	0.639957	0.652761	0.681403	0.661919	0.631842	0.618224	0.644922
4	13715.41	0.665507	0.641221	0.643443	0.680146	0.67439	0.629824	0.626357	0.641265

val_precision	train_recall	val_recall	train_f1	val_f1	val_best_threshold	val_best_significance	val_acc_bestthr	val_precision_bestthr	val_recall_bestthr	val_f1_bestthr
0.65275	0.531572	0.519808	0.589494	0.578743	0.235	35.44028	0.518026	0.509332	0.986795	0.671877
0.625399	0.616231	0.581833	0.630297	0.60283	0.1825	35.4925	0.515866	0.508156	0.991997	0.67205
0.65819	0.53372	0.536029	0.588149	0.590861	0.2625	35.38728	0.508977	0.504537	0.993595	0.66924
0.633744	0.586752	0.560048	0.614463	0.594621	0.22	35.38295	0.511871	0.506044	0.990392	0.669834
0.638004	0.589354	0.584067	0.614214	0.609845	0.23	35.49294	0.516752	0.508579	0.991593	0.672327

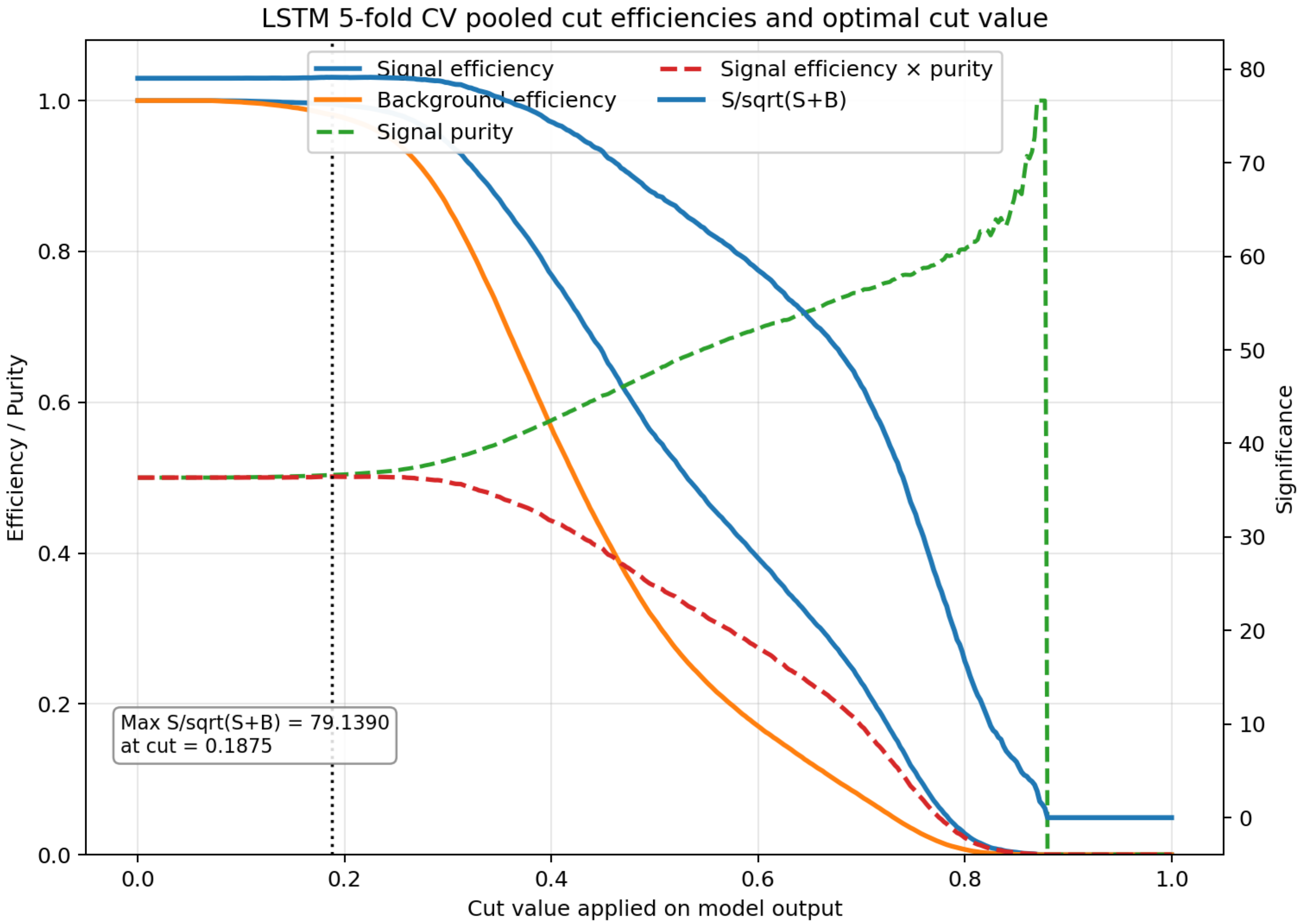

LSTM 5-fold CV mean AUC
Weighted ROC AUC
Epoch
Train mean
Validation mean
0.65
0.60
0.55
0.50
0
5
10
15
20
25


LSTM 5-fold CV mean F1
Weighted F1
Epoch
Train mean
Validation mean
0.80
0.75
0.70
0.65
0.60
0.55
0
5
10
15
20
25

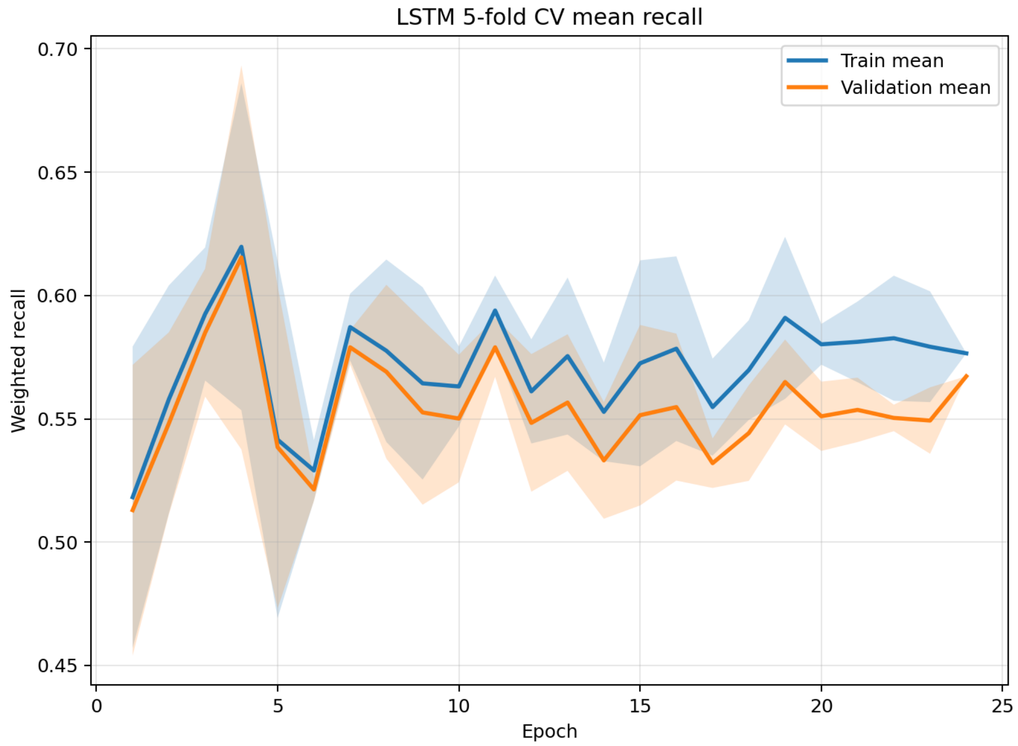

LSTM 5-fold CV mean accuracy
Train mean
Validation mean
Weighted accuracy
0.45
0.50
0.55
0.60
0.65
0
5
10
15
20
25
Epoch

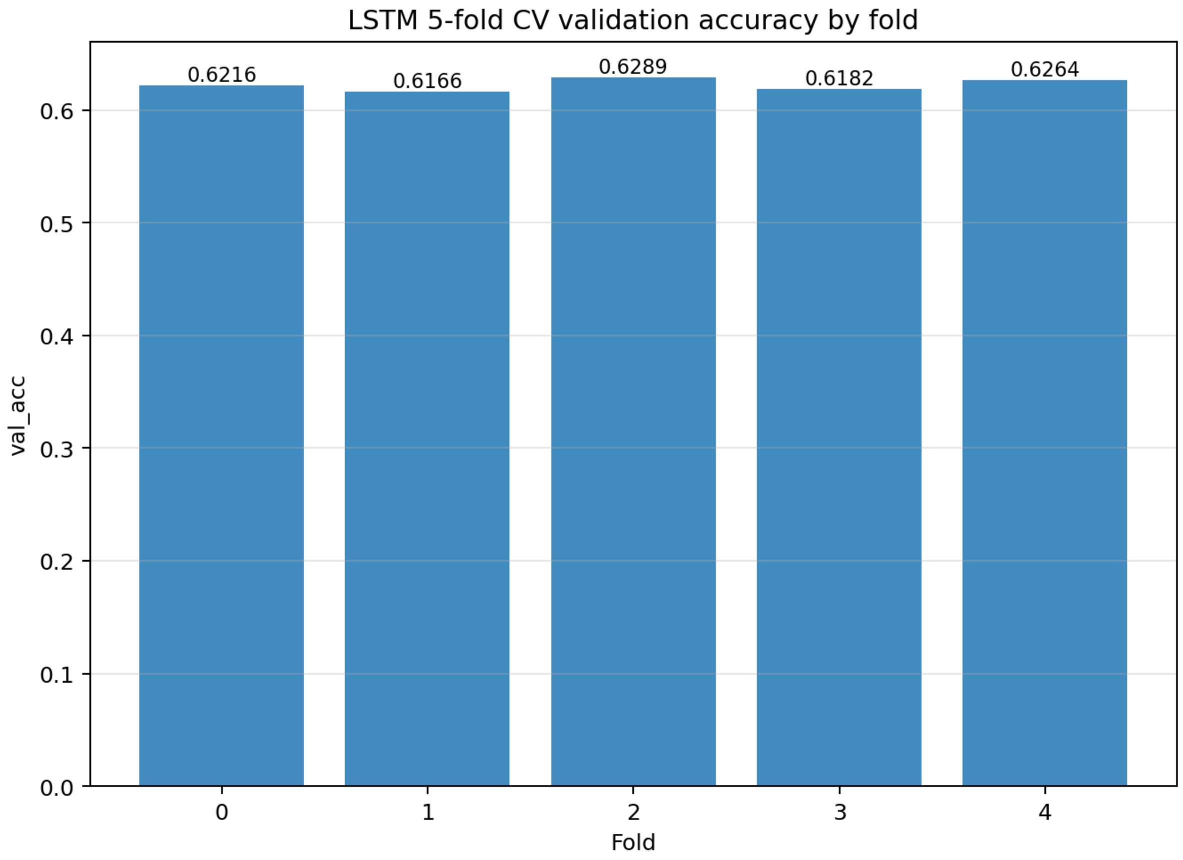

LSTM 5-fold CV mean average precision
Train mean
Validation mean
Weighted average precision
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00
0
5
10
15
20
25
Epoch

LSTM

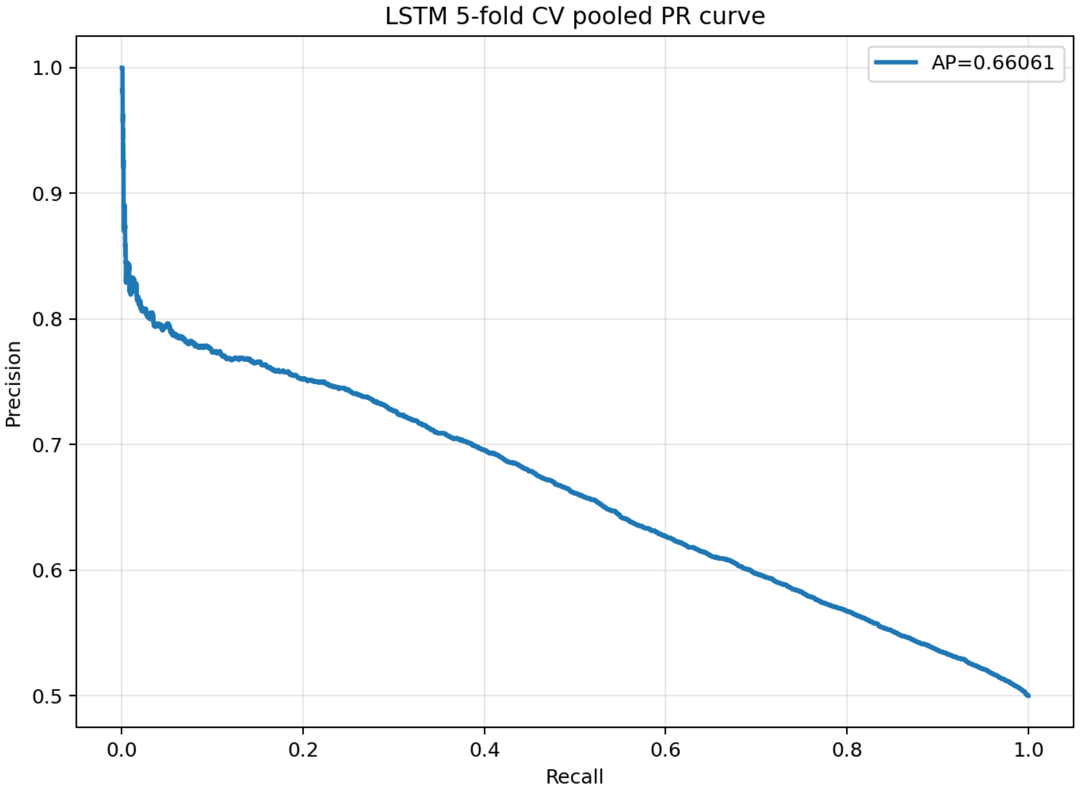

LSTM 5-fold CV pooled cut efficiencies and optimal cut value
Signal efficiency
Background efficiency
Signal purity
Signal efficiency × purity
S/sqrt(S+B)
Max S/sqrt(S+B) = 79.1390
at cut = 0.1875
Efficiency / Purity
Significance
Cut value applied on model output
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
70
80

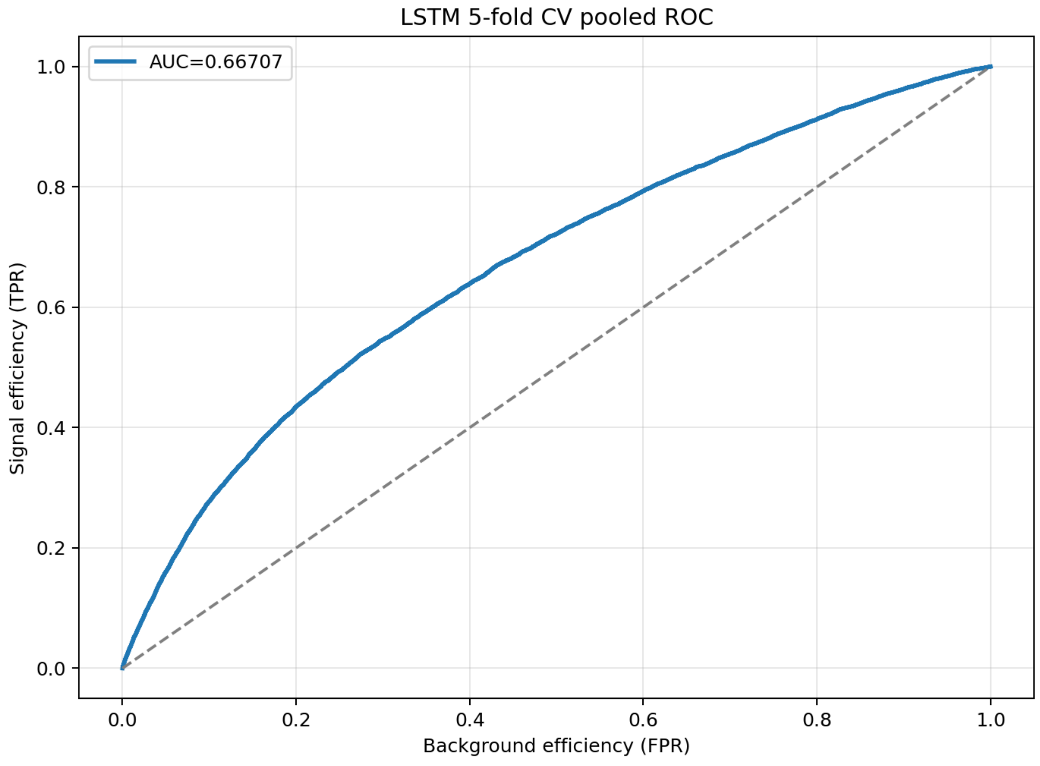
LSTM


LSTM 5-fold CV mean recall
Train mean
Validation mean
Weighted recall
Epoch
0.70
0.65
0.60
0.55
0.50
0.45
0
5
10
15
20
25

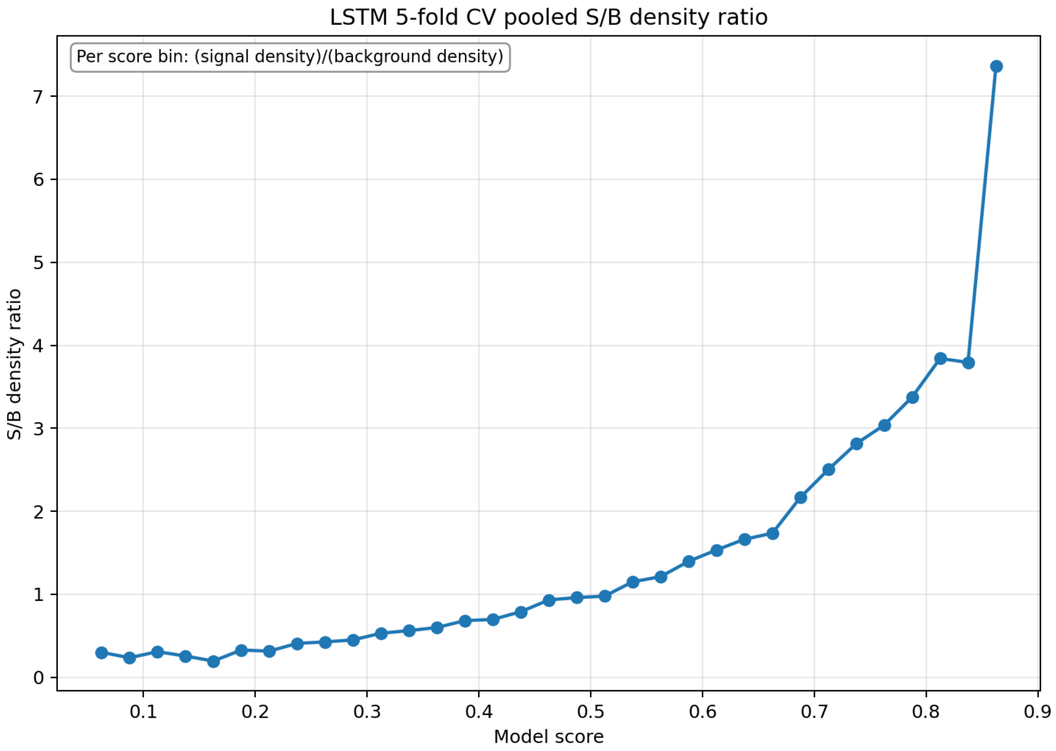

LSTM 5-fold CV validation accuracy by fold
0.6216
0.6166
0.6289
0.6182
0.6264
val_acc
Fold
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
1
2
3
4

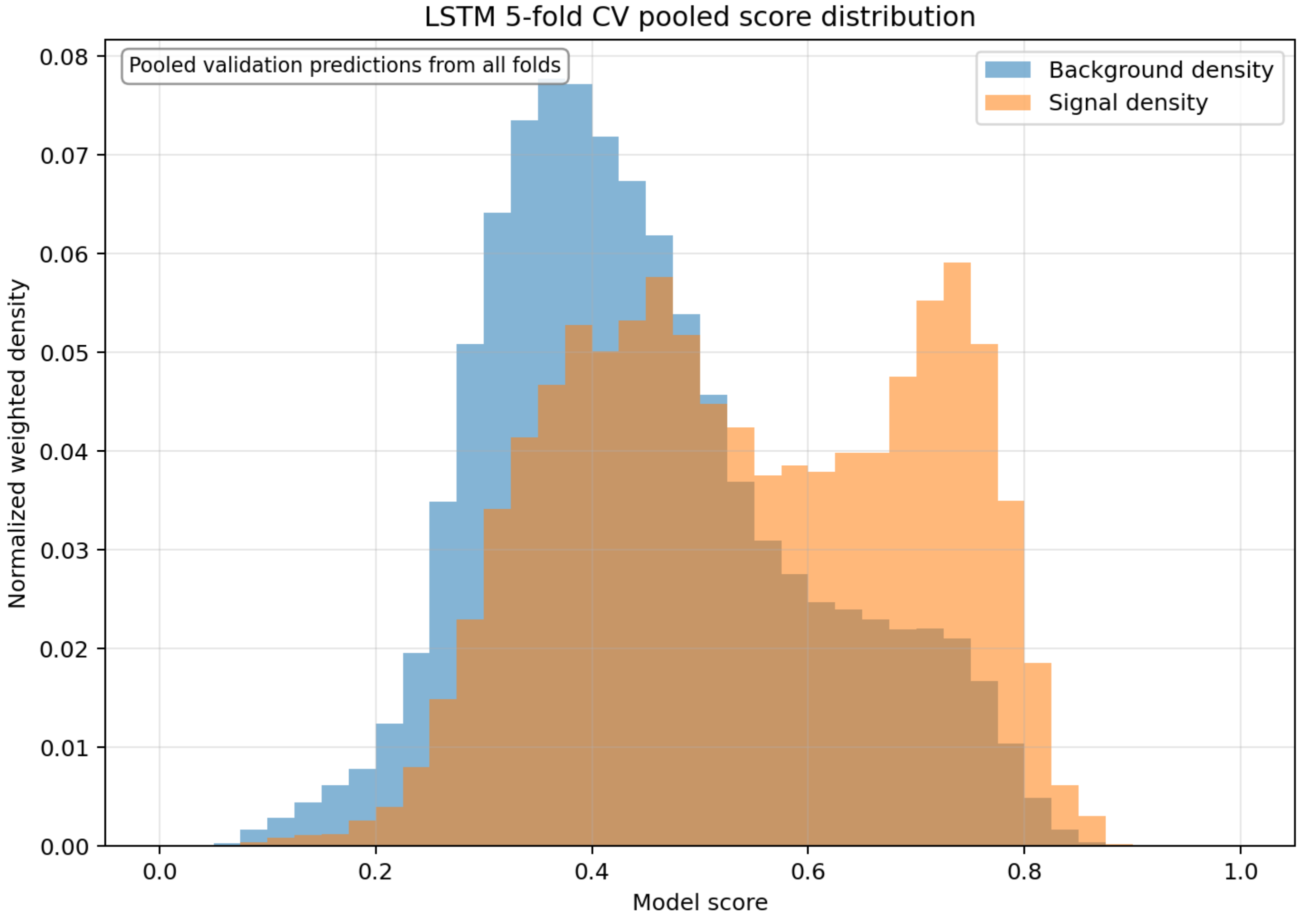
LSTM


LSTM 5-fold CV pooled PR curve
AP=0.66061
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

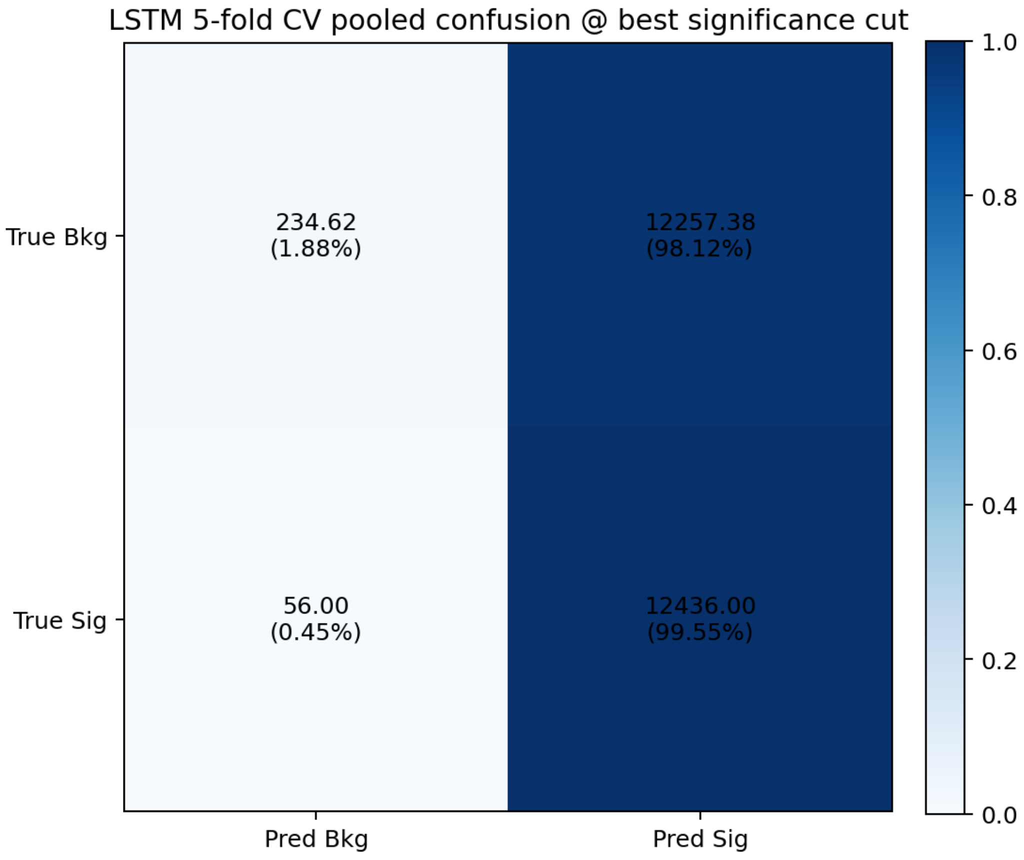

LSTM 5-fold CV pooled ROC
AUC=0.66707
Signal efficiency (TPR)
Background efficiency (FPR)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

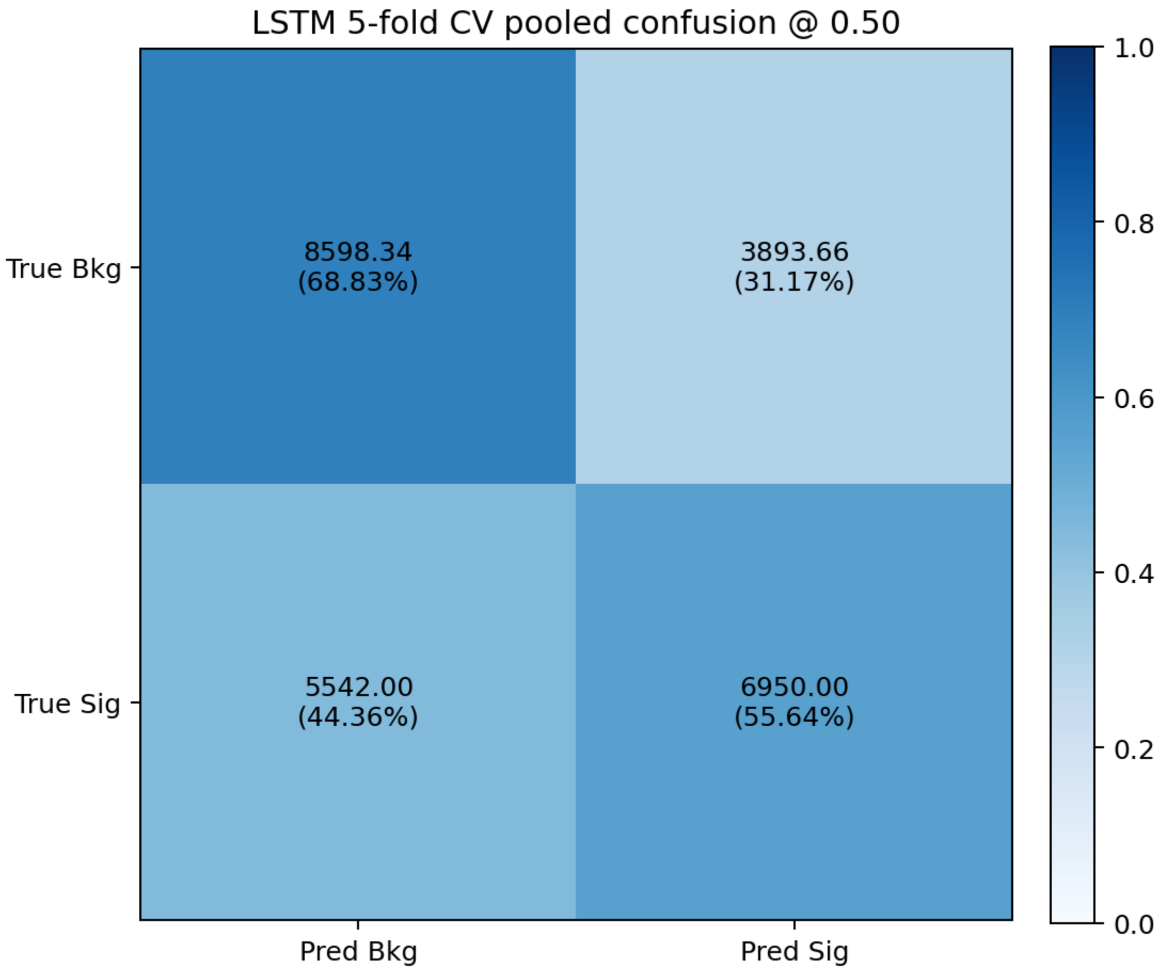
LSTM


LSTM 5-fold CV pooled S/B density ratio
Per score bin: (signal density)/(background density)
S/B density ratio
Model score
0
1
2
3
4
5
6
7
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


LSTM 5-fold CV pooled score distribution
Pooled validation predictions from all folds
Background density
Signal density
Normalized weighted density
Model score
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.0
0.2
0.4
0.6
0.8
1.0

LSTM


LSTM 5-fold CV pooled confusion @ best significance cut
True Bkg
True Sig
Pred Bkg
Pred Sig
234.62
(1.88%)
12257.38
(98.12%)
56.00
(0.45%)
12436.00
(99.55%)
1.0
0.8
0.6
0.4
0.2
0.0


LSTM 5-fold CV pooled confusion @ 0.50
True Bkg
True Sig
Pred Bkg
Pred Sig
8598.34
(68.83%)
3893.66
(31.17%)
5542.00
(44.36%)
6950.00
(55.64%)
1.0
0.8
0.6
0.4
0.2
0.0

Transformers

Results from New Denser Transformer Model, Earlier it was 3 layered, now it is 4 layers and it took longer to run on condor.

Model Description

Reads ROOT ntuples and extracts a fixed set of physics features.

Builds event weights from the supplied weight expression.

Trains a Transformer -based classifier on weighted signal and background events.

Scans the classifier cut to find the threshold that gives the best **signal-to-background ratio** or best **approximate significance** $S/\sqrt{S+B}$.

After the Transformer, the [CLS] representation is passed to a **deep dense head** made of four dense blocks:

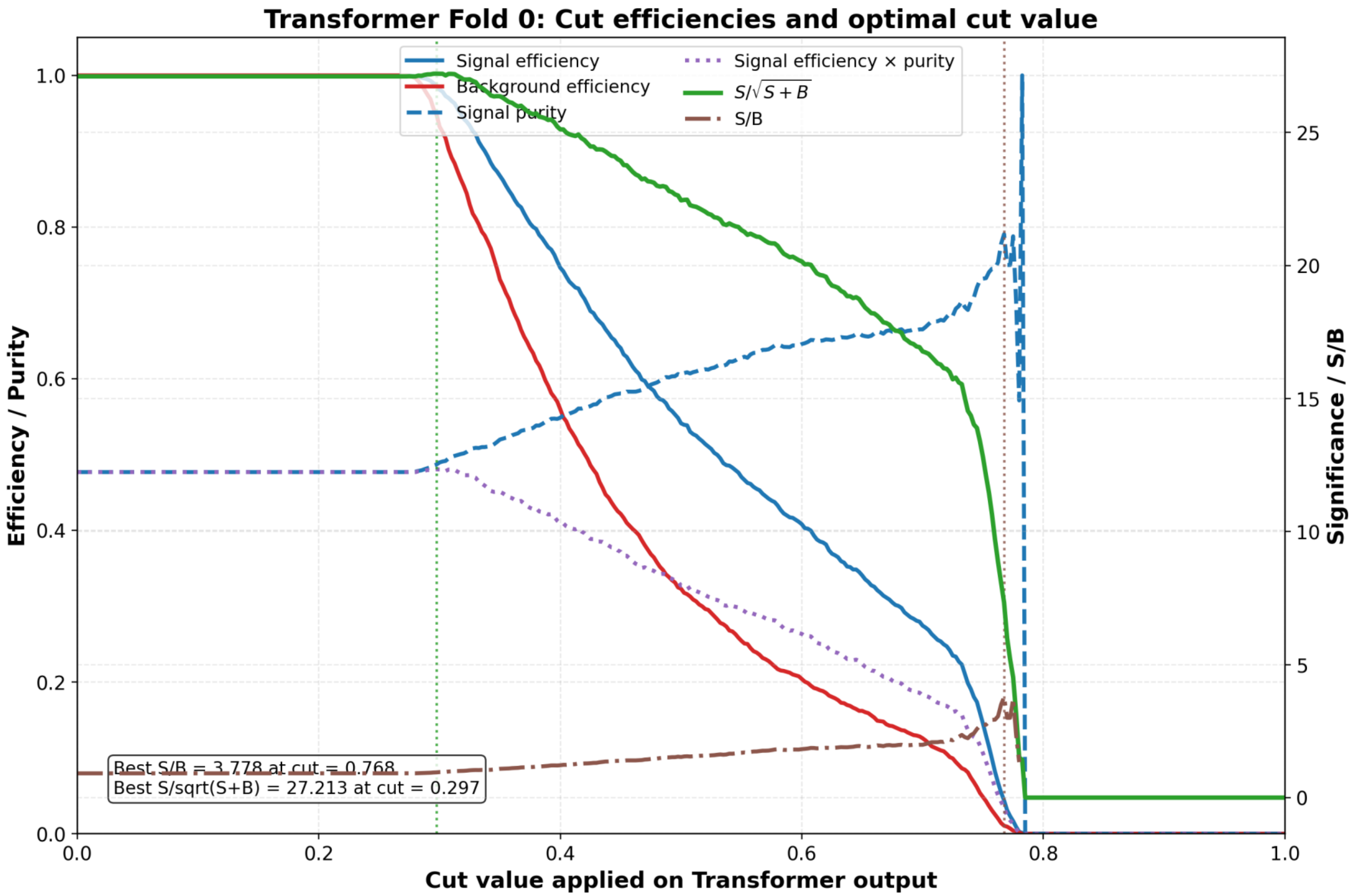
Linear

LayerNorm

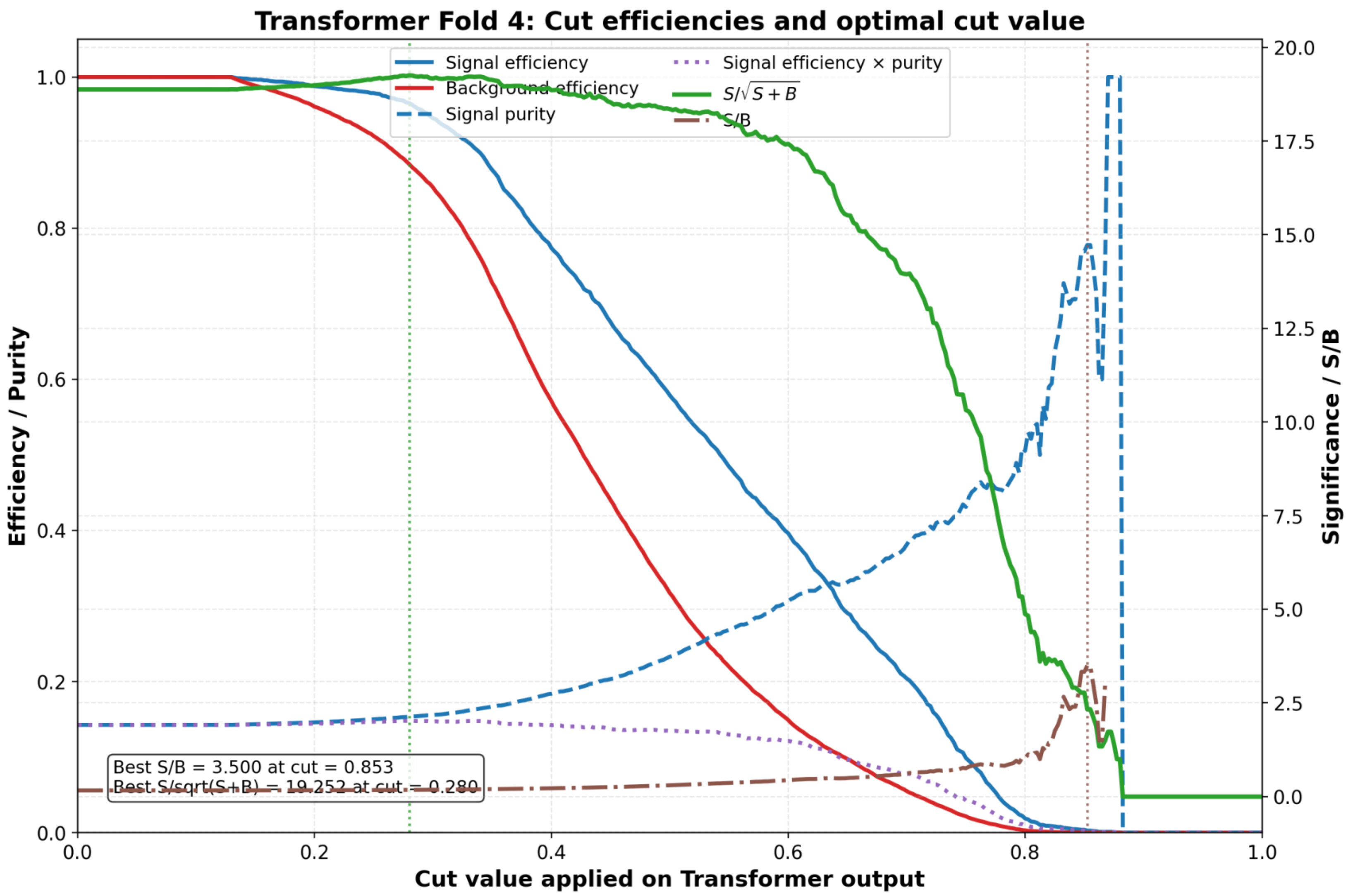
GELU

Dropout

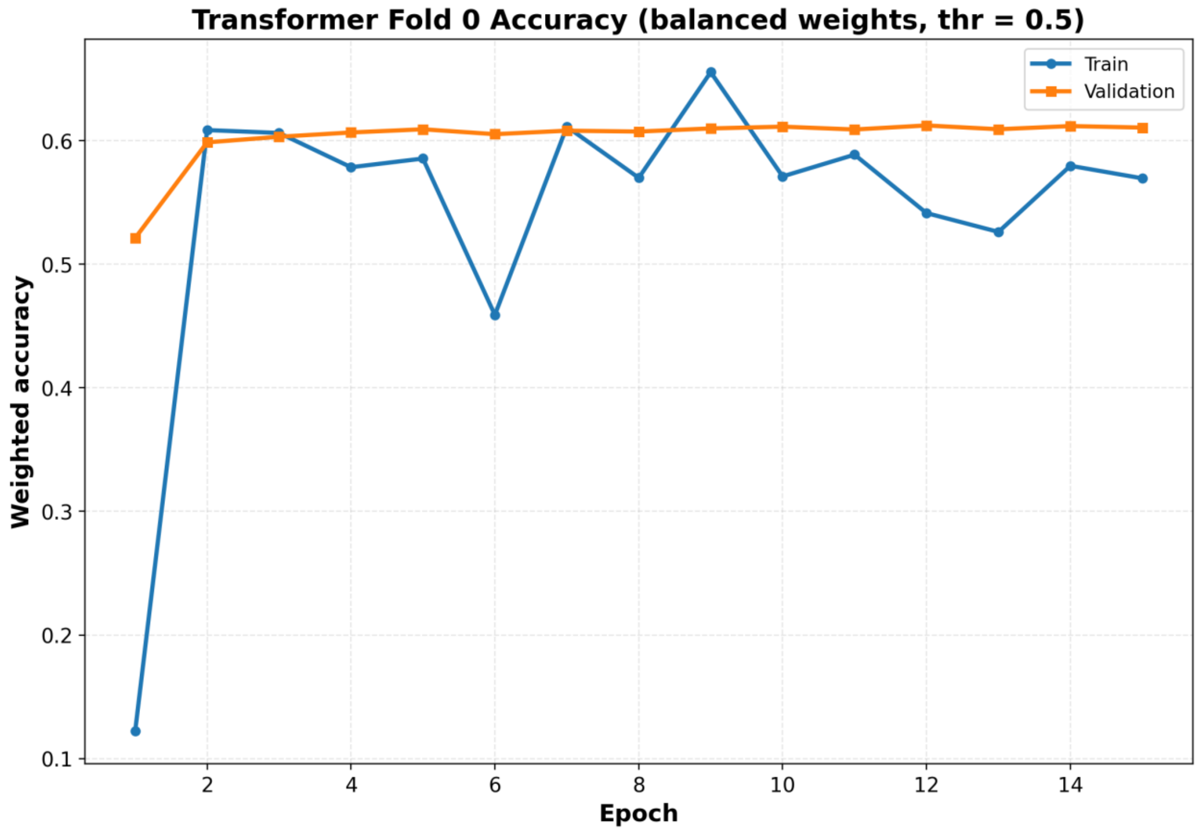
Input features→token embeddings→[CLS] token + Transformer encoder→dense head→sigmoid score

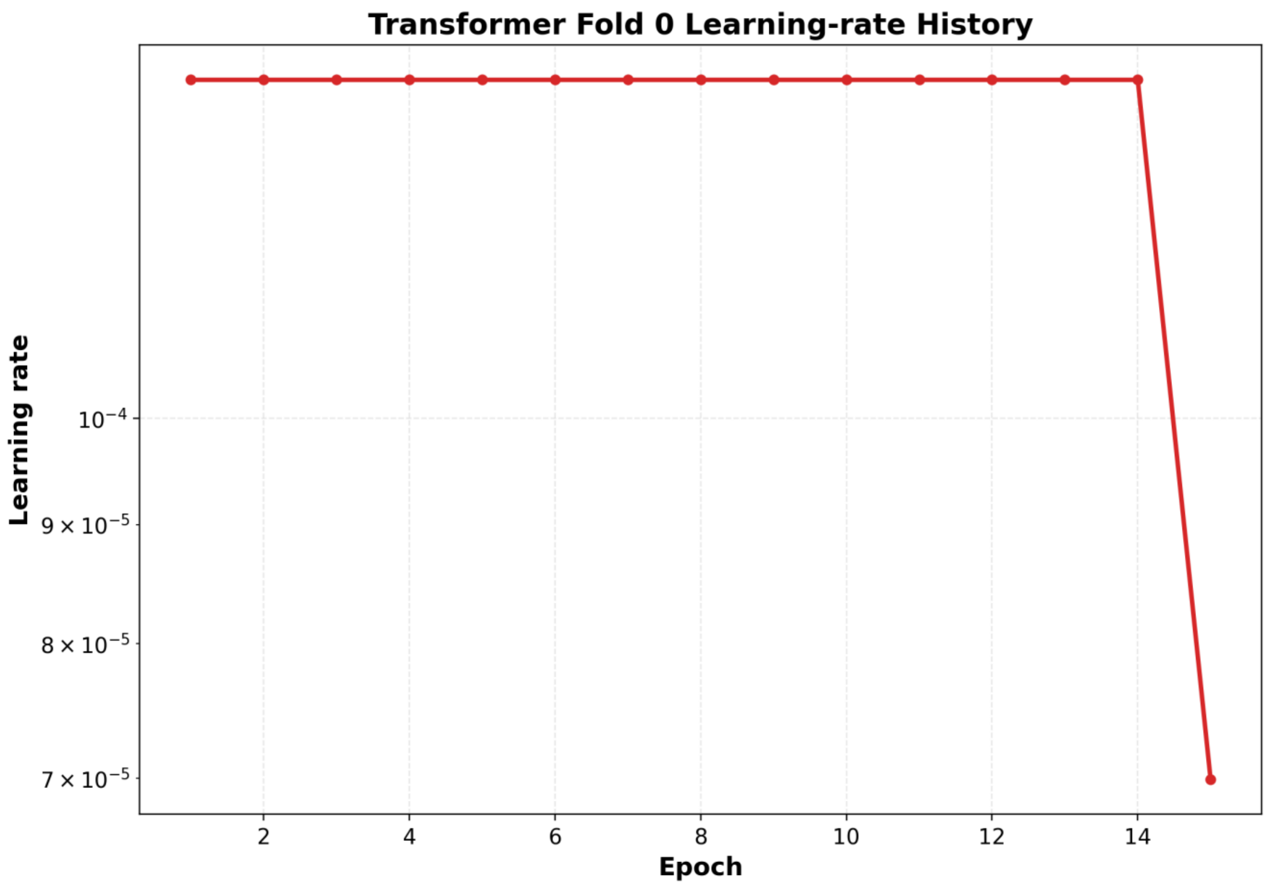


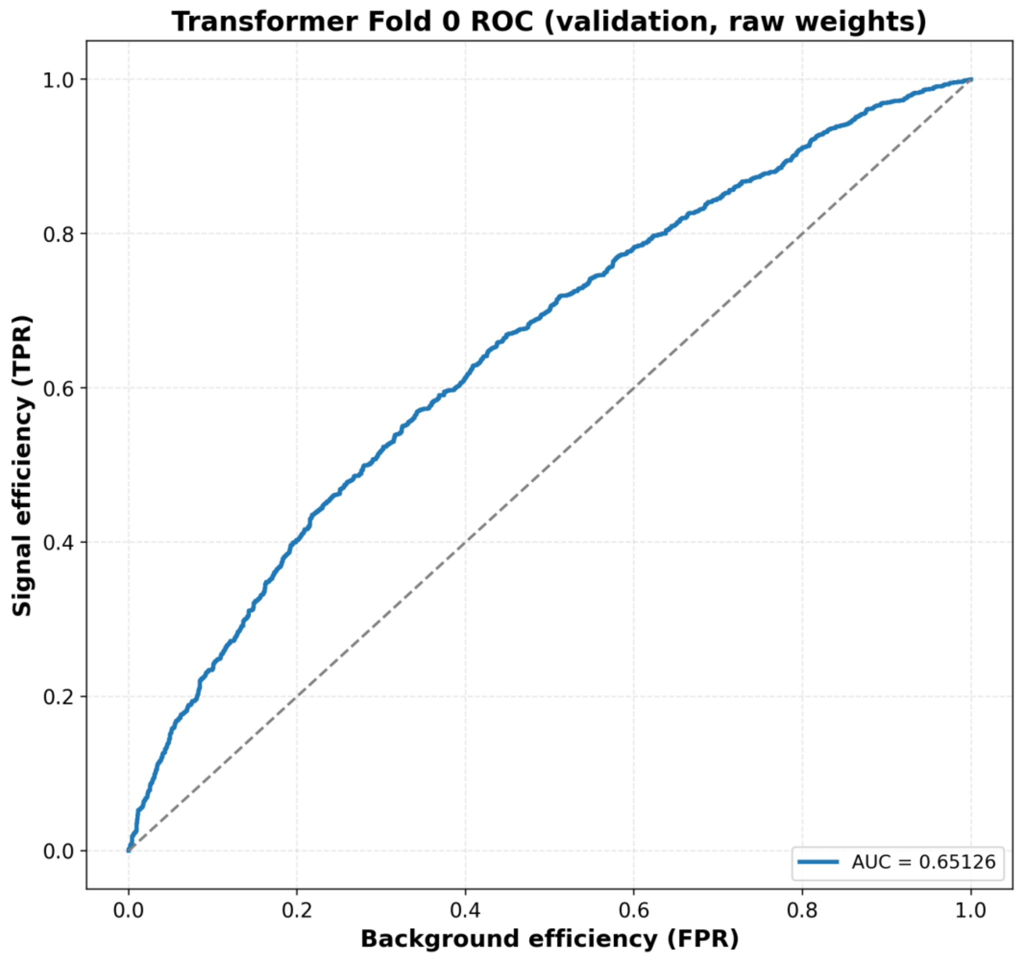
Transformer Fold 0: Cut efficiencies and optimal cut value
Signal efficiency
Background efficiency
Signal purity
Signal efficiency × purity
$S/\sqrt{S+B}$
S/B
Best S/B = 3.778 at cut = 0.768
Best S/sqrt(S+B) = 27.213 at cut = 0.297
Efficiency / Purity
Significance / S/B
Cut value applied on Transformer output

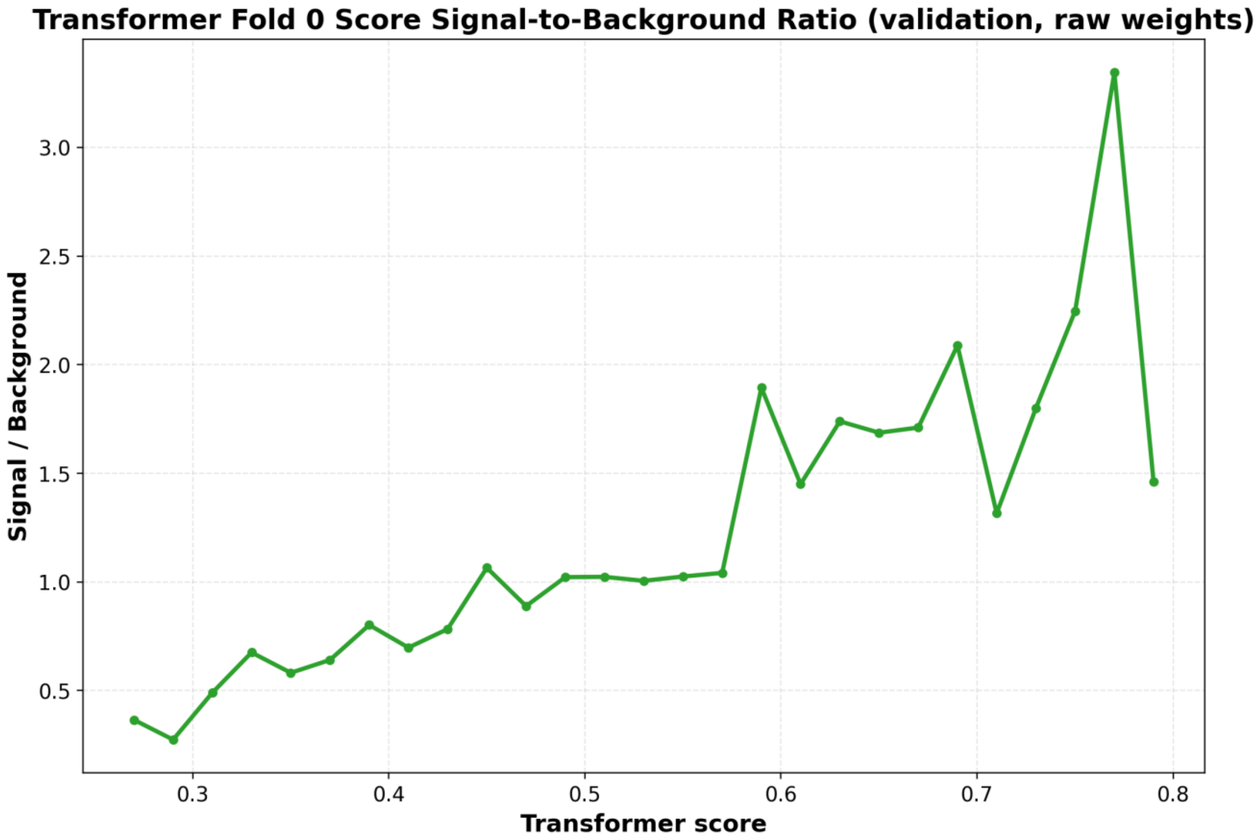

Transformer Fold 4: Cut efficiencies and optimal cut value
Signal efficiency
Background efficiency
Signal purity
Signal efficiency × purity
$S/\sqrt{S+B}$
S/B
Best S/B = 3.500 at cut = 0.853
Best S/sqrt(S+B) = 19.252 at cut = 0.280
Efficiency / Purity
Significance / S/B
Cut value applied on Transformer output
0.0
0.2
0.4
0.6
0.8
1.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

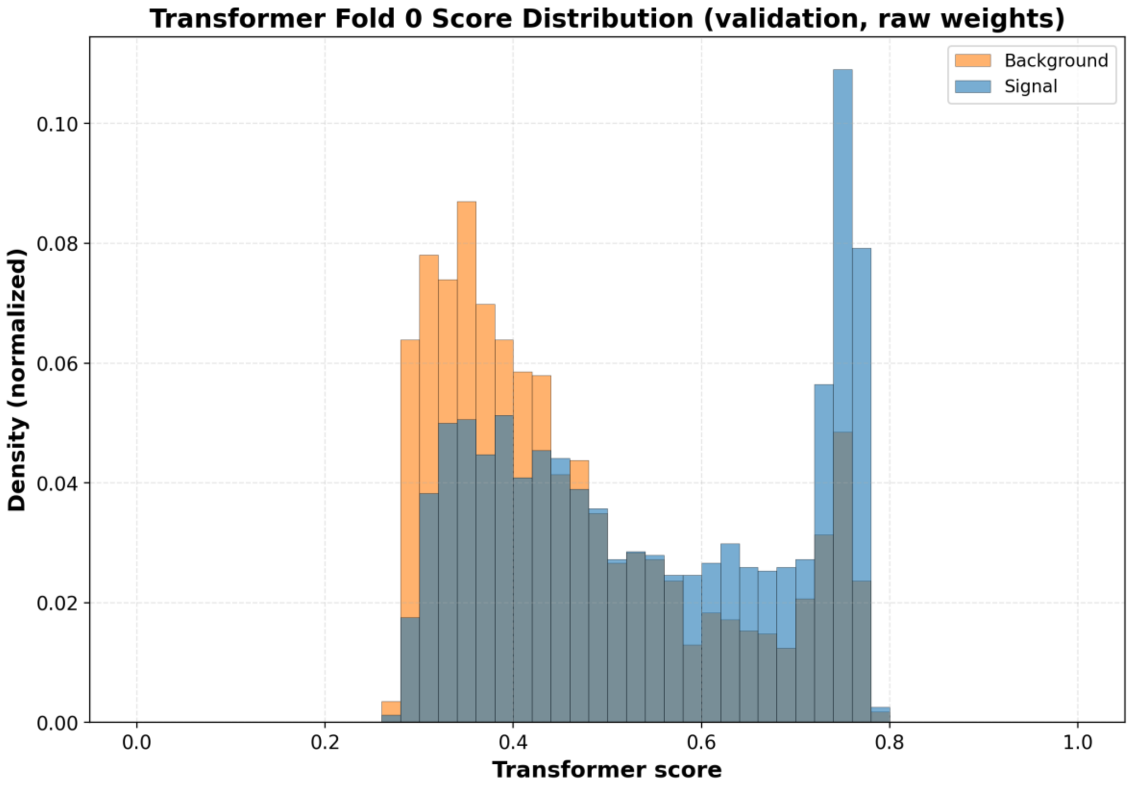

Transformer Fold 0 Loss (early stop at epoch 11)
Train
Validation
Weighted BCE loss
0.75
0.70
0.65
0.60
2
4
6
8
10
12
14
Epoch


Transformer Fold 0 Accuracy (balanced weights, thr = 0.5)
Train
Validation
Weighted accuracy
0.6
0.5
0.4
0.3
0.2
0.1
2
4
6
8
10
12
14
Epoch

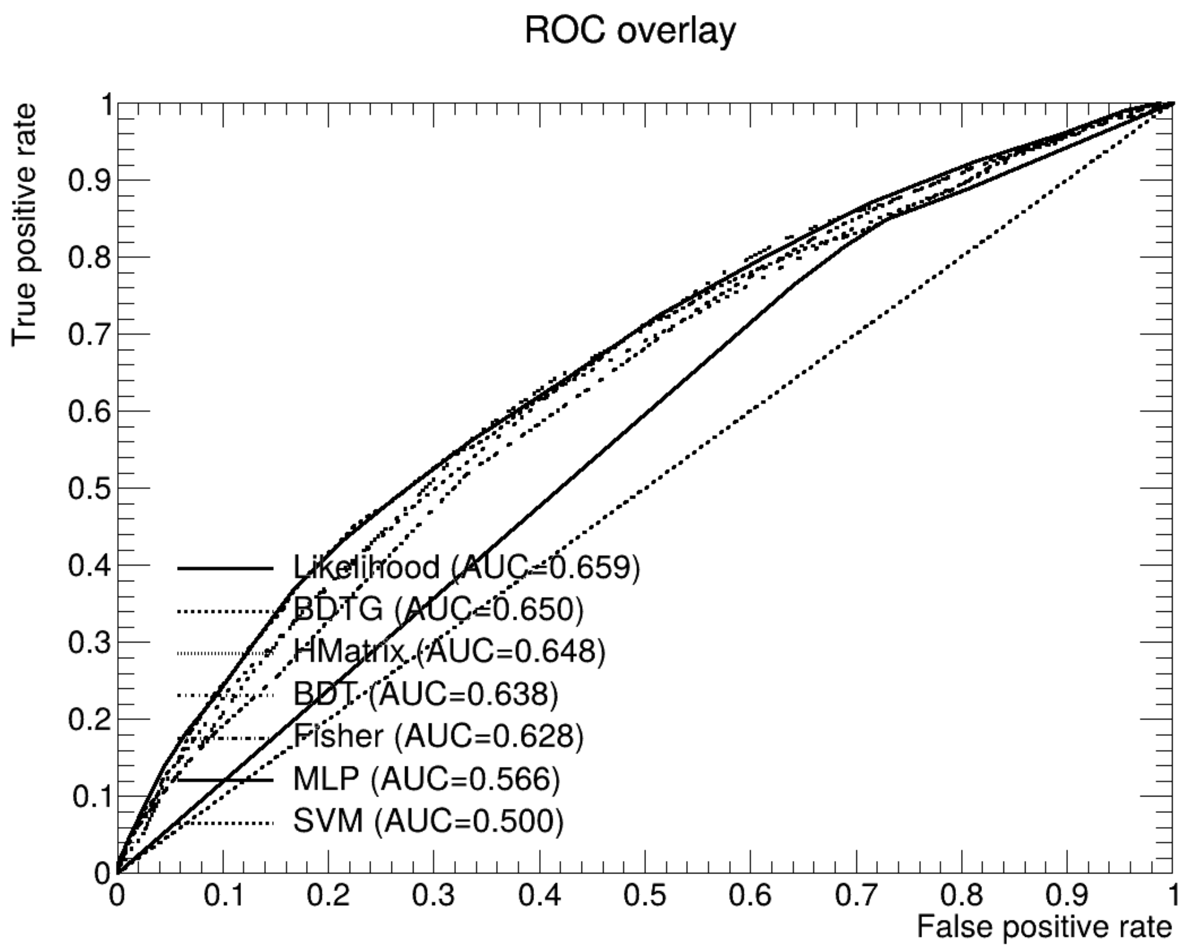

Transformer Fold 0 Learning-rate History
Learning rate
10^{-4}
9×10^{-5}
8×10^{-5}
7×10^{-5}
2
4
6
8
10
12
14
Epoch


Transformer Fold 0 ROC (validation, raw weights)
Signal efficiency (TPR)
Background efficiency (FPR)
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.65126

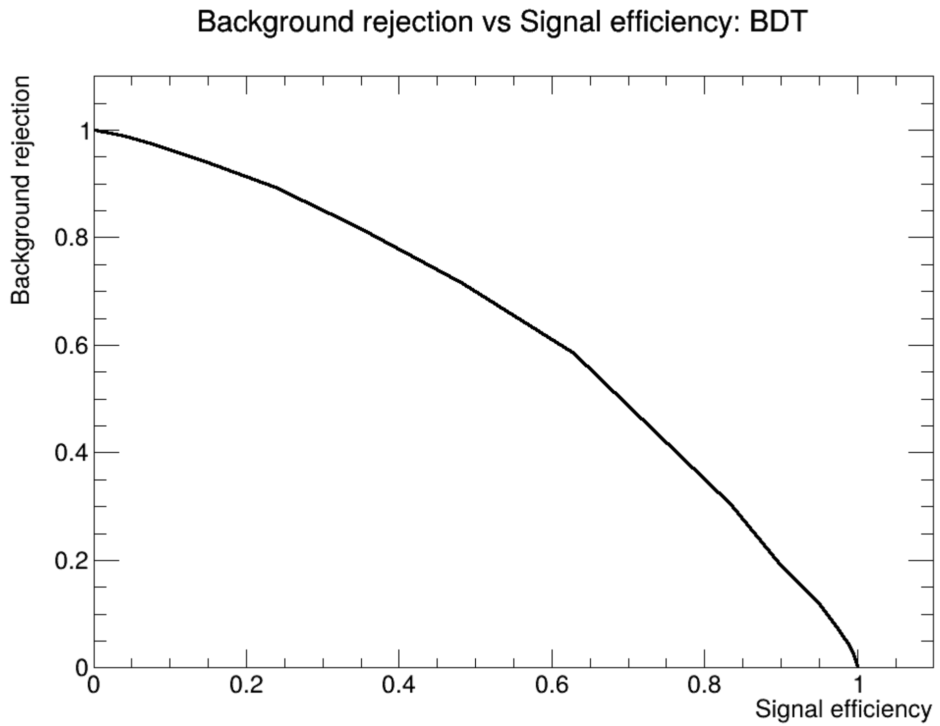

Transformer Fold 0 Score Signal-to-Background Ratio (validation, raw weights)
Signal / Background
Transformer score
0.5
1.0
1.5
2.0
2.5
3.0
0.3
0.4
0.5
0.6
0.7
0.8

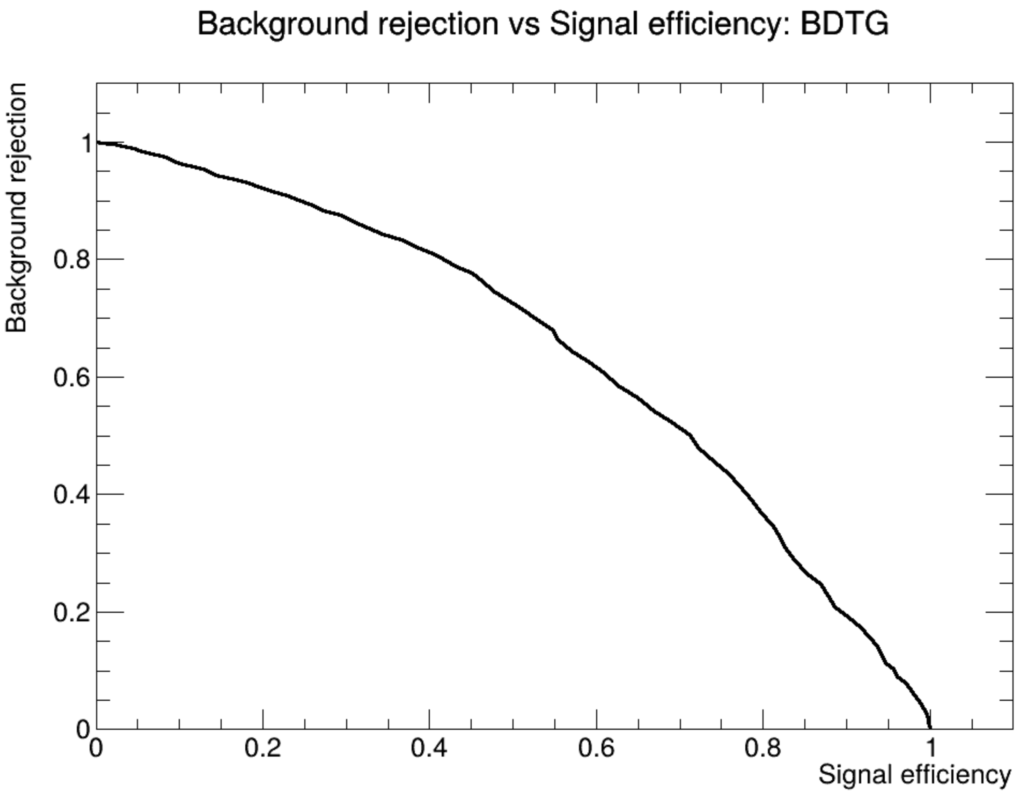

Transformer Fold 0 Score Distribution (validation, raw weights)
Background
Signal
Density (normalized)
Transformer score
0.00
0.02
0.04
0.06
0.08
0.10
0.0
0.2
0.4
0.6
0.8
1.0

TMVA (Still need to organize all results for publication)

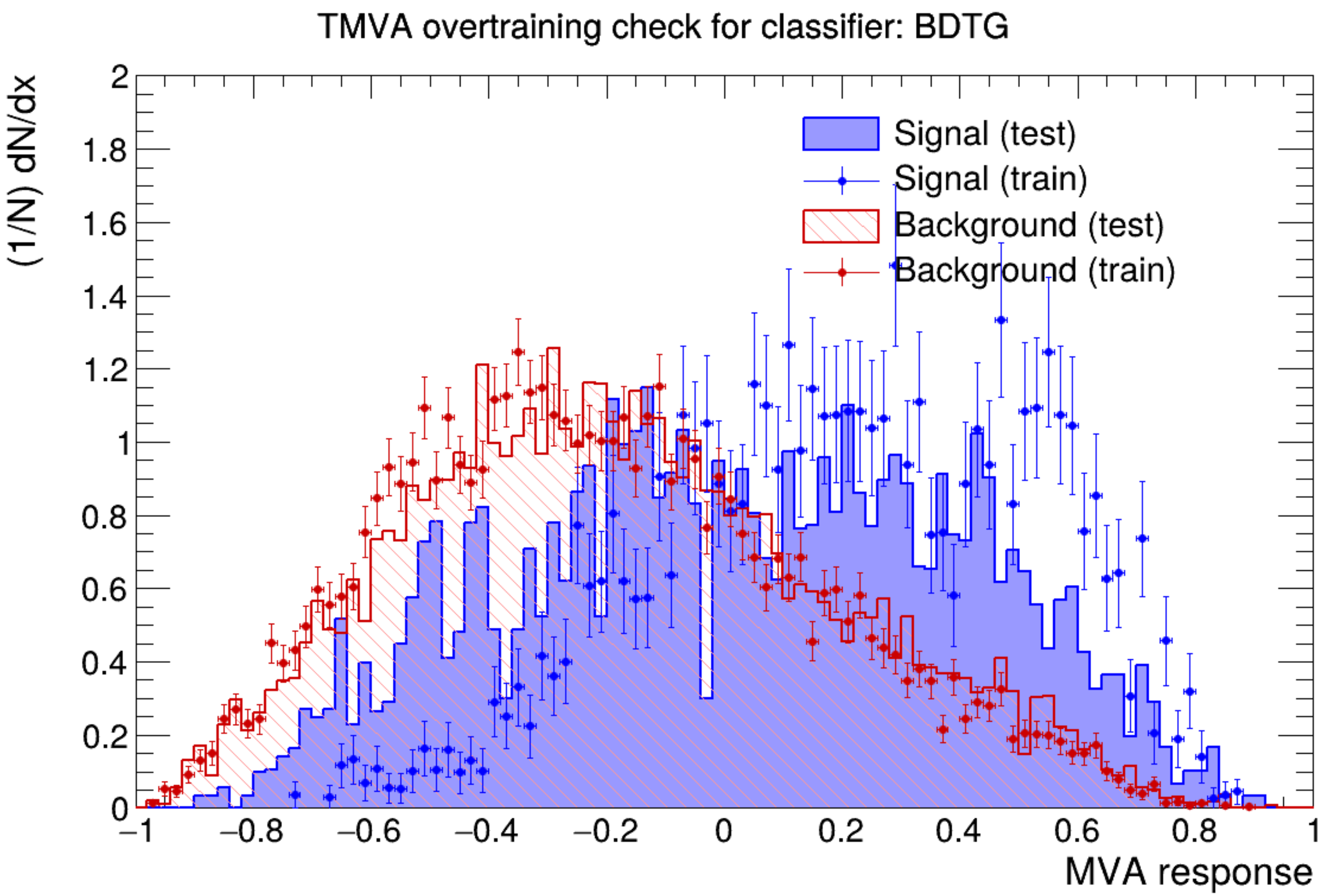


TMVA

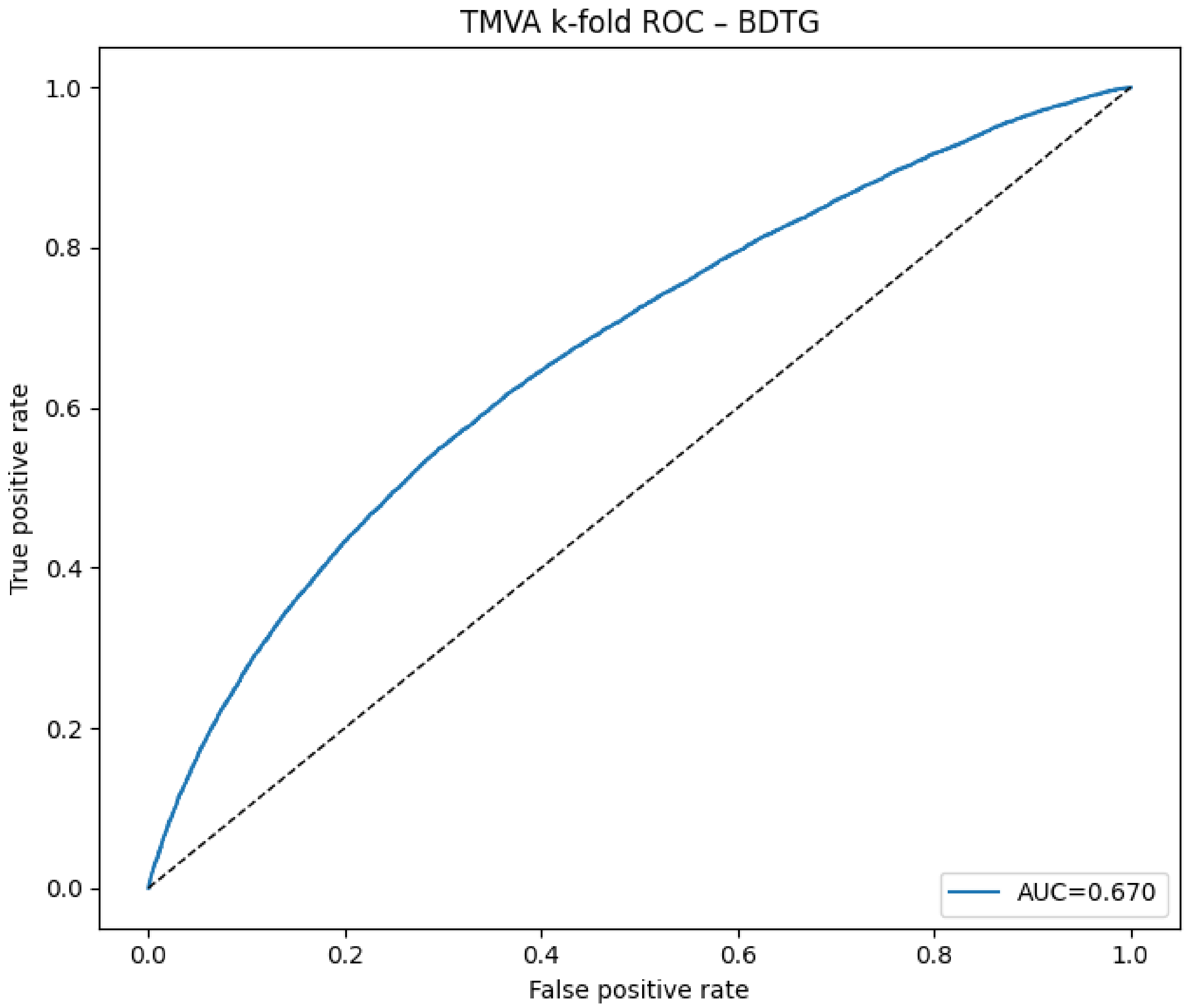

Background rejection vs Signal efficiency: BDT
Background rejection
Signal efficiency
0
0.2
0.4
0.6
0.8
1


Background rejection vs Signal efficiency: BDTG
Background rejection
Signal efficiency
0
0.2
0.4
0.6
0.8
1

TMVA


TMVA overtraining check for classifier: BDTG
(1/N) dN/dx
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1
MVA response
Signal (test)
Signal (train)
Background (test)
Background (train)

TMVA



TMVA k-fold ROC – BDTG
1.0
0.8
0.6
0.4
0.2
0.0
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
AUC=0.670

National Research Tomsk State University

Лаборатория анализа данных физики высоких энергий Томского государственного университета

Thank you for your attention!!!