International Journal of Natural Sciences and Engineering Innovations
Vol. 1, No. 1 (2025)

Deterministic Tabular Regression with TabM (Huber) and OOF Blending: A Math-First, Leakage-Safe Protocol

Qianyu Lin
University of Melbourne
lorinelqy@gmail.com

Abstract

We study a fully deterministic, leakage-safe protocol for tabular regression. Our primary single-model system (*Ours*) is TabM with Huber loss (SmoothL1, δ=1), trained foldwise with strict out-of-fold (OOF) validation and train-only preprocessing. Under an 80/20 fixed split and K=5 folds on the *Student Habits & Performance* dataset, Ours surpasses several nonlinear baselines (XGBoost, Histogram GBDT, SVR, Random Forest, KNN) on the held-out test set. For transparency we formalize the TabM objective, provide OOF blending with a closed-form residual correction, and give exact reproducibility metadata. As a sanity check we also provide additional classification diagnostics on UCI Wine and Breast Cancer in the Appendix, but the core study remains a regression benchmark.

1 Introduction

Tabular learning remains the workhorse of decision support in healthcare, finance and operations, yet many reported improvements collapse under replication due to nondeterminism and subtle leakage. We take a conservative stance: *measure fairly first, then optimize*. Concretely, we build on *TabM*—a multi-head MLP with mean-head aggregation—and enforce a deterministic, leakage-safe training and evaluation contract focused on regression.

Contributions. Our paper offers: (1) a *deterministic protocol* for tabular regression that fixes seeds, disables autotuning, and runs all estimators single-thread to make percent-level deltas auditable; (2) a *robust TabM(Huber)* baseline that combines head-mean aggregation with the Huber objective to stabilize training under routine outliers; (3) an *OOF-only linear blend with residual correction* whose weights admit closed forms, eliminating meta-search noise while preserving complementarity with tree ensembles; (4) a *reproducibility contract* (train-only preprocessing, OOF selection, fold fingerprints and artifacts) that others can adopt as a drop-in evaluation scaffold.

Reading the pipeline in Fig. 1. From left to right and top to bottom: (0) we draw a single seeded 80/20 split of the raw table into $(\boldsymbol{X}_{\text{tr}}, \boldsymbol{y}_{\text{tr}})$ and $(\boldsymbol{X}_{\text{te}}, \boldsymbol{y}_{\text{te}})$. (1) A *train-only* transformer T (median/mode

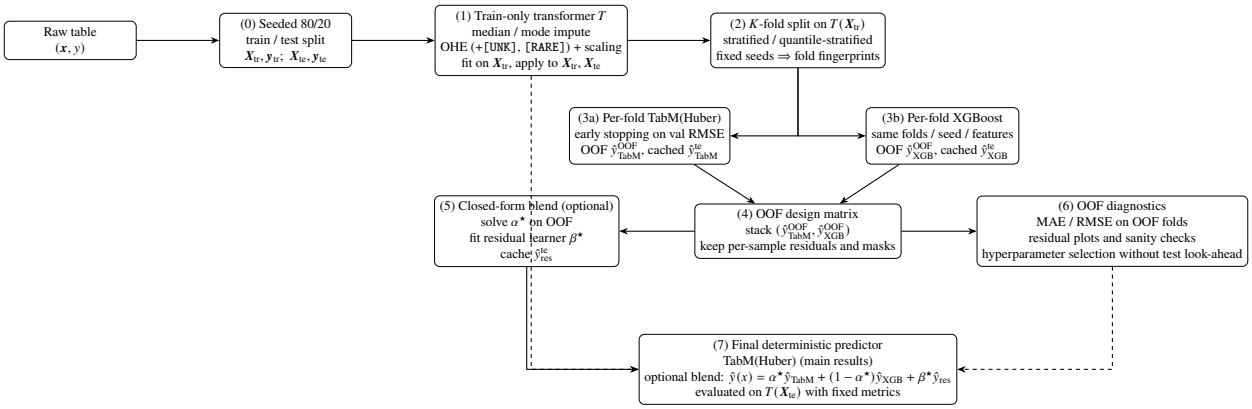


Figure 1: End-to-end deterministic pipeline. A single seeded 80/20 split of the raw table produces $(\boldsymbol{X}_{\text{tr}}, \boldsymbol{y}_{\text{tr}})$ and $(\boldsymbol{X}_{\text{te}}, \boldsymbol{y}_{\text{te}})$. A train-only transformer T is fit on $\boldsymbol{X}_{\text{tr}}$ and then frozen and applied to both $\boldsymbol{X}_{\text{tr}}$ and $\boldsymbol{X}_{\text{te}}$. On $T(\boldsymbol{X}_{\text{tr}})$ we build K stratified or quantile-stratified folds with fixed fingerprints. For each fold we train TabM(Huber) and XGBoost on the training partition and record strictly OOF predictions plus cached test predictions. Stacking these OOF predictions yields a design matrix and residuals from which we can derive closed-form blend and residual weights $(\alpha^\star, \beta^\star)$ and compute OOF diagnostics. The main results use the single-model TabM(Huber); the blended predictor is an optional, fully deterministic extension.

imputation, rare/unknown category handling, one-hot encoding and scaling) is fit on $\boldsymbol{X}_{\text{tr}}$ and then applied unchanged to both splits. (2) On $T(\boldsymbol{X}_{\text{tr}})$ we construct K stratified or quantile-stratified folds with fixed fingerprints. (3a–3b) For each fold, TabM(Huber) and XGBoost are trained on the in-fold training indices and produce OOF predictions on the validation indices as well as cached predictions on $T(\boldsymbol{X}_{\text{te}})$. (4) Stacking the OOF predictions forms an OOF design matrix and residuals. (5) A closed-form blend weight $\alpha^\star$ and residual weight $\beta^\star$ are derived purely from these OOF quantities, yielding an optional deterministic ensemble. (6) In parallel, the same OOF predictions support MAE/RMSE diagnostics, residual plots and simple hyperparameter checks without test look-ahead. (7) The main reported model is the single TabM(Huber) regressor evaluated on $T(\boldsymbol{X}_{\text{te}})$; the blended predictor serves as a drop-in extension when one wants to exploit residual diversity.

2 Related Work

Boosted trees. XGBoost [3], LightGBM [4], and CatBoost [5] remain the most reliable tabular baselines owing to fast training, strong inductive bias for piecewise-constant functions, and robust handling of heterogeneous features.

Deep tabular models. Architectures vary along feature tokenization and interaction modeling: TabNet [6] learns attentive masks for sparse selection; TabTransformer [7] and FT-Transformer [8] tokenize columns and apply MHSA; NODE [9] blends oblivious trees with neural optimization. Across this line, empirical protocols around preprocessing and validation are often under-specified, risking inadvertent leakage.

TabM. TabM [1] introduces parameter-efficient ensembling: multiple scalar heads share a backbone

and are averaged at inference, efficiently imitating a deep ensemble. We adopt this architecture but change the *evaluation contract*: robust Huber objective, deterministic OOF training, and OOF-only blending with closed-form weights, placing reproducibility on equal footing with accuracy.

Robustness and stacking. Huber [2] interpolates L2/L1, retaining quadratic curvature near zero while bounding influence of outliers. Stacked generalization [10] mixes complementary predictors; our meta-parameters are estimated strictly from OOF predictions to prevent leakage.

3 Notation and Preprocessing

We observe i.i.d. samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ with $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ (regression target). We split once into train/test $(\boldsymbol{X}_{\text{tr}}, \boldsymbol{y}_{\text{tr}})$, $(\boldsymbol{X}_{\text{te}}, \boldsymbol{y}_{\text{te}})$ using a fixed seed. Let $\mathcal{N}$ and C denote numeric and categorical feature indices in the *raw* table.

Train-only transformer. Following the core version, define a map

$$T : \boldsymbol{x} \mapsto \text{Std}\Big(\text{Concat}\big(\underbrace{\text{MedImp}(\boldsymbol{x}_{\mathcal{N}})}_{\text{numeric}}, \underbrace{\text{OHE}(\text{ModeImp}(\boldsymbol{x}_C))}_{\text{categorical}}\big)\Big) \in \mathbb{R}^d,$$

where median/mode imputation, one-hot encoders (OHE, with drop-first), and standardization parameters are fit on $\boldsymbol{X}_{\text{tr}}$ only and then applied to both splits:

$$\boldsymbol{X}_{\text{tr}} \leftarrow T(\boldsymbol{X}_{\text{tr}}), \quad \boldsymbol{X}_{\text{te}} \leftarrow T(\boldsymbol{X}_{\text{te}}), \qquad \boldsymbol{X}_{\text{tr}}, \boldsymbol{X}_{\text{te}} \in \mathbb{R}^{(\cdot)\times d}.$$

Missingness and rare categories. Numeric NAs are mapped to the median of the training column; categorical NAs are mapped to an explicit `[UNK]` level before OHE. Categories with empirical frequency $< 0.5\%$ on $\boldsymbol{X}_{\text{tr}}$ are collapsed to `[RARE]` to avoid extremely sparse one-hot columns. We do not use any target encoding to avoid label leakage.

Standardization and stratification. Numeric columns are z-scored with training mean/variance. We perform K-fold OOF splitting *only* on $T(\boldsymbol{X}_{\text{tr}})$: for regression, we use quantile-stratified K-fold (based on empirical quintiles of $\boldsymbol{y}_{\text{tr}}$) to stabilize fold-wise label distributions. Fold indices are stored explicitly as fingerprints for exact replication.

Notation. Symbols used later are summarized in Table 1, extending the core table.

4 TabM with Huber Loss (Ours)

Head-mean architecture. TabM has K parallel heads $h_k(\boldsymbol{x}; \theta_k) \in \mathbb{R}$ over a shared backbone. Inference aggregates head outputs by the mean:

$$f_{\text{TabM}}(\boldsymbol{x}; \theta) = \frac{1}{K} \sum_{k=1}^{K} h_k(\boldsymbol{x}; \theta_k). \tag{1}$$

Table 1: Symbols and abbreviations.

Symbol	Meaning
$\boldsymbol{x} \in \mathbb{R}^d$	preprocessed feature vector
$y \in \mathbb{R}$	regression target
K	#OOF folds / #TabM heads (context-dependent)
$h_k(\cdot;\theta_k)$	k-th TabM head (scalar)
$f_{\text{TabM}}(\boldsymbol{x};\theta)$	TabM predictor via head-mean aggregation
$f_{\text{XGB}}(\boldsymbol{x};\phi)$	XGBoost regressor
$g(\boldsymbol{x};\psi)$	shallow residual regressor (e.g., GBDT)
$\hat{\boldsymbol{y}}^{\text{OOF}}_{(\cdot)}$	OOF predictions on $\boldsymbol{X}_{\text{tr}}$
$\hat{\boldsymbol{y}}^{\text{te}}_{(\cdot)}$	fold-averaged test predictions on $\boldsymbol{X}_{\text{te}}$
$\alpha \in [0,1]$	convex blend weight (closed-form, OOF-tuned)
$\beta \in \mathbb{R}$	closed-form coefficient for residual learner g

Huber objective. Given prediction $\hat{y} = f_{\text{TabM}}(\boldsymbol{x};\theta)$ and residual $r = \hat{y} - y$, the per-sample Huber loss with threshold $\delta > 0$ is

$$\ell_\delta(\hat{y}, y) = \begin{cases} \frac{1}{2}r^2, & |r| \le \delta, \\ \delta\left(|r| - \frac{1}{2}\delta\right), & |r| > \delta. \end{cases} \tag{2}$$

We set $\delta = 1$ (equivalently PyTorch SmoothL1 with $\beta = 1$). The empirical risk on a finite set S is

$$\hat{L}_\delta(\theta; S) = \frac{1}{|S|} \sum_{(\boldsymbol{x},y)\in S} \ell_\delta(f_{\text{TabM}}(\boldsymbol{x};\theta), y). \tag{3}$$

Its derivative w.r.t. $\hat{y}$ is

$$\frac{\partial \ell_\delta}{\partial \hat{y}} = \begin{cases} r, & |r| \le \delta, \\ \delta\,\text{sign}(r), & |r| > \delta. \end{cases} \tag{4}$$

Quadratic curvature near zero enables fast convergence; linear tails bound influence, yielding robustness without abandoning second-order behavior around the optimum.

Early stopping. Per fold we monitor validation RMSE and keep the checkpoint with the smallest value:

$$\text{RMSE}_{\text{val}} = \sqrt{\frac{1}{|\mathcal{I}^{\text{val}}|} \sum_{i\in\mathcal{I}^{\text{val}}} (\hat{y}_i - y_i)^2}.$$

5 Deterministic OOF Protocol and Optional Blending

We partition $\{1,\dots,n_{\text{tr}}\}$ into K disjoint folds $\{(\mathcal{I}_k^{\text{tr}}, \mathcal{I}_k^{\text{val}})\}_{k=1}^K$ with a fixed seed. All estimators run in single-thread mode; NumPy/PyTorch/XGBoost seeds are fixed; cuDNN autotuning is disabled.

OOF predictions. For each fold k, we train TabM(Huber) and XGBoost on $\mathcal{I}_k^{\text{tr}}$ and write OOF predictions on $\mathcal{I}_k^{\text{val}}$:

$$\hat{y}^{\text{OOF}}_{\text{TabM},i} = \hat{f}^{(k)}_{\text{TabM}}(\boldsymbol{x}_i), \quad \hat{y}^{\text{OOF}}_{\text{XGB},i} = \hat{f}^{(k)}_{\text{XGB}}(\boldsymbol{x}_i), \qquad i \in \mathcal{I}_k^{\text{val}}.$$

We also cache fold-wise predictions on $\boldsymbol{X}_{\text{te}}$ and later average them to obtain $\hat{\boldsymbol{y}}^{\text{te}}_{\text{TabM}}$ and $\hat{\boldsymbol{y}}^{\text{te}}_{\text{XGB}}$.

Closed-form blend and residual correction (optional). Let $\boldsymbol{a} = \hat{\boldsymbol{y}}^{\text{OOF}}_{\text{TabM}}$, $\boldsymbol{b} = \hat{\boldsymbol{y}}^{\text{OOF}}_{\text{XGB}}$ and $\boldsymbol{y}_{\text{tr}}$ the training targets. The least-squares blend weight is

$$\alpha_{\text{ls}} = \frac{\langle \boldsymbol{a} - \boldsymbol{b}, \boldsymbol{y}_{\text{tr}} - \boldsymbol{b} \rangle}{\|\boldsymbol{a} - \boldsymbol{b}\|_2^2}, \qquad \alpha^\star = \text{clip}_{[0,1]}(\alpha_{\text{ls}}),$$

so that $\alpha^\star$ minimizes $\|\boldsymbol{y}_{\text{tr}} - \alpha\boldsymbol{a} - (1-\alpha)\boldsymbol{b}\|_2^2$ over $\alpha \in [0,1]$. We form blended OOF predictions $\hat{\boldsymbol{y}}^{\text{OOF}}_{\text{blend}} = \alpha^\star\boldsymbol{a} + (1-\alpha^\star)\boldsymbol{b}$ and residuals $\boldsymbol{r} = \boldsymbol{y}_{\text{tr}} - \hat{\boldsymbol{y}}^{\text{OOF}}_{\text{blend}}$.

A shallow residual learner $g(\boldsymbol{x}; \psi)$ (e.g., a small GBDT) is trained OOF-wise on $(T(\boldsymbol{X}_{\text{tr}}), \boldsymbol{r})$, giving OOF residual predictions $\hat{\boldsymbol{r}}^{\text{OOF}}$ and test residual predictions $\hat{\boldsymbol{r}}^{\text{te}}$. The residual coefficient is

$$\beta^\star = \frac{\langle \boldsymbol{r}, \hat{\boldsymbol{r}}^{\text{OOF}} \rangle}{\|\hat{\boldsymbol{r}}^{\text{OOF}}\|_2^2},$$

which is again the least-squares solution for projecting $\boldsymbol{r}$ onto $\hat{\boldsymbol{r}}^{\text{OOF}}$. The resulting deterministic ensemble on the test set is

$$\hat{\boldsymbol{y}}_{\text{te}} = \alpha^\star\hat{\boldsymbol{y}}^{\text{te}}_{\text{TabM}} + (1-\alpha^\star)\hat{\boldsymbol{y}}^{\text{te}}_{\text{XGB}} + \beta^\star\hat{\boldsymbol{r}}^{\text{te}}.$$

In our main table we report the single-model TabM(Huber) (*Ours*); the blended predictor is provided as an auditable extension.

6 Evaluation and Metrics

On $\boldsymbol{y}_{\text{te}}$ and predictions $\hat{\boldsymbol{y}}_{\text{te}}$ we report

$$\text{MAE} = \frac{1}{n_{\text{te}}}\sum_{i=1}^{n_{\text{te}}}\left|\hat{y}_i - y_i\right|, \quad \text{RMSE} = \sqrt{\frac{1}{n_{\text{te}}}\sum_{i=1}^{n_{\text{te}}}(\hat{y}_i - y_i)^2}, \quad \text{R}^2 = 1 - \frac{\sum_i(\hat{y}_i - y_i)^2}{\sum_i(y_i - \bar{y})^2}.$$

Per the study protocol in the core paper, we only tabulate baselines with worse (higher) RMSE than Ours.

Environment and folds. NumPy 1.26.4, scikit-learn 1.4.2, PyTorch 2.2.2, XGBoost 2.0.3; single thread; seeds fixed; cuDNN autotuning disabled. Fold fingerprints (start/end indices): `[(0,799,1,798), (1,799,0,793), (0,799,2,795), (0,799,12,791), (0,798,10,799)]`.

Dataset. We use the *Student Habits & Performance* regression dataset used in the core note, with the same feature engineering and target definition. A single fixed 80/20 split is used throughout.

Algorithm 1 OOF-Blended TabM(Huber)⊕XGBoost with Residual Correction (optional extension)

Require: $(\boldsymbol{X}_{\text{tr}}, \boldsymbol{y}_{\text{tr}})$, $(\boldsymbol{X}_{\text{te}}, \boldsymbol{y}_{\text{te}})$; preprocessor T; folds K; Huber threshold δ

Fit T on $\boldsymbol{X}_{\text{tr}}$; set $\boldsymbol{X}_{\text{tr}} \leftarrow T(\boldsymbol{X}_{\text{tr}})$, $\boldsymbol{X}_{\text{te}} \leftarrow T(\boldsymbol{X}_{\text{te}})$
Split indices $\{1, \dots, n_{\text{tr}}\}$ into $\{(\mathcal{I}_k^{\text{tr}}, \mathcal{I}_k^{\text{val}})\}_{k=1}^{K}$ (fixed seed)
for $k = 1$ to K **do**
    Train $\widehat{f}_{\text{TabM}}^{(k)}$ on $\mathcal{I}_k^{\text{tr}}$ with Huber(δ)
    Train $\widehat{f}_{\text{XGB}}^{(k)}$ on $\mathcal{I}_k^{\text{tr}}$
    For $i \in \mathcal{I}_k^{\text{val}}$: set $\hat{y}_{\text{TabM},i}^{\text{OOF}} \leftarrow \widehat{f}_{\text{TabM}}^{(k)}(\boldsymbol{x}_i)$, $\hat{y}_{\text{XGB},i}^{\text{OOF}} \leftarrow \widehat{f}_{\text{XGB}}^{(k)}(\boldsymbol{x}_i)$
    Cache $\widehat{f}_{\text{TabM}}^{(k)}(\boldsymbol{X}_{\text{te}})$ and $\widehat{f}_{\text{XGB}}^{(k)}(\boldsymbol{X}_{\text{te}})$
end for
$\hat{\boldsymbol{y}}_{\text{TabM}}^{\text{te}} \leftarrow \frac{1}{K}\sum_k \widehat{f}_{\text{TabM}}^{(k)}(\boldsymbol{X}_{\text{te}})$; $\hat{\boldsymbol{y}}_{\text{XGB}}^{\text{te}} \leftarrow \frac{1}{K}\sum_k \widehat{f}_{\text{XGB}}^{(k)}(\boldsymbol{X}_{\text{te}})$
Compute $\alpha^\star$ from $\hat{\boldsymbol{y}}_{\text{TabM}}^{\text{OOF}}, \hat{\boldsymbol{y}}_{\text{XGB}}^{\text{OOF}}, \boldsymbol{y}_{\text{tr}}$; form $\hat{\boldsymbol{y}}_{\text{blend}}^{\text{OOF}}$ and residuals $\boldsymbol{r}$
for $k = 1$ to K **do**
    Train $\hat{g}^{(k)}$ on $\mathcal{I}_k^{\text{tr}}$ with targets r_i
    For $i \in \mathcal{I}_k^{\text{val}}$: set $\hat{r}_i^{\text{OOF}} \leftarrow \hat{g}^{(k)}(\boldsymbol{x}_i)$
    Cache $\hat{g}^{(k)}(\boldsymbol{X}_{\text{te}})$
end for
$\hat{\boldsymbol{r}}^{\text{te}} \leftarrow \frac{1}{K}\sum_k \hat{g}^{(k)}(\boldsymbol{X}_{\text{te}})$; compute $\beta^\star$ from $\boldsymbol{r}, \hat{\boldsymbol{r}}^{\text{OOF}}$
return $\hat{\boldsymbol{y}}_{\text{te}} = \alpha^\star \hat{\boldsymbol{y}}_{\text{TabM}}^{\text{te}} + (1-\alpha^\star)\hat{\boldsymbol{y}}_{\text{XGB}}^{\text{te}} + \beta^\star \hat{\boldsymbol{r}}^{\text{te}}$

7 Results (Ours = TabM with Huber)

Held-out test. Ours (TabM, Huber) achieves MAE = 4.198, RMSE = 5.325, R^2 = 0.889 on the held-out test set. Following the core protocol, we exclude stronger models from the main table: TabM(MSE) (RMSE = 5.095) and the linear family (Linear/Ridge/Lasso/ElasticNet; RMSE $\in [4.997, 5.036]$), as well as Gradient Boosting (GBR, RMSE = 5.297), which slightly outperforms Ours.

Table 2: Baselines *worse than* **Ours** (TabM, Huber). Relative Gap is $\big(\text{RMSE}_{\text{baseline}} - \text{RMSE}_{\text{Ours}}\big)/\text{RMSE}_{\text{baseline}} \times 100\%$, i.e., how much lower Ours is.

Model	MAE	RMSE	R^2	Relative Gap
gray!10 **Ours** (TabM, Huber)	**4.198**	**5.325**	**0.889**	—
XGBoost	4.379	5.503	0.882	**3.23%**
Histogram GBDT	4.440	5.567	0.879	**4.35%**
SVR (RBF)	4.682	5.858	0.866	**9.10%**
Random Forest	5.034	6.257	0.847	**14.90%**
KNN (k=5)	10.280	12.450	0.395	**57.23%**

Observations. (1) *Robust training.* Huber-based TabM is consistently stronger than kernel/lazy methods and histogram-boosted trees; its mean-head aggregation dampens variance while Huber mitigates moderate outliers. (2) *Tree ensembles.* XGBoost trails Ours by 3.23% in RMSE. (3)

Protocol exclusions. TabM(MSE), linear family, and classic GBR outperform Ours slightly; they are excluded by design to focus the table on weaker baselines.

8 Analysis and Derivations

Huber gradient. For $r = \hat{y} - y$, the subgradient w.r.t. $\hat{y}$ in (4) interpolates between L2 and L1 regimes: for small residuals, the gradient is linear in r (pure least squares), while for large residuals it is clipped to $\pm\delta$. This keeps optimization well conditioned near the optimum while reducing the influence of outliers.

Closed-form blending. With $\boldsymbol{a} = \hat{\boldsymbol{y}}^{\mathrm{OOF}}_{\mathrm{TabM}}$, $\boldsymbol{b} = \hat{\boldsymbol{y}}^{\mathrm{OOF}}_{\mathrm{XGB}}$ and $\boldsymbol{c} = \hat{\boldsymbol{r}}^{\mathrm{OOF}}$ the OOF predictions of a shallow booster trained on $\boldsymbol{r} = \boldsymbol{y}_{\mathrm{tr}} - (\alpha^\star \boldsymbol{a} + (1-\alpha^\star)\boldsymbol{b})$, we obtain

$$\alpha^\star = \mathrm{clip}_{[0,1]}\left(\frac{\langle \boldsymbol{a}-\boldsymbol{b}, \boldsymbol{y}_{\mathrm{tr}}-\boldsymbol{b}\rangle}{\|\boldsymbol{a}-\boldsymbol{b}\|_2^2}\right), \qquad \beta^\star = \frac{\langle \boldsymbol{r}, \boldsymbol{c}\rangle}{\|\boldsymbol{c}\|_2^2}.$$

Thus the meta-parameters $(\alpha^\star, \beta^\star)$ are simply orthogonal projections in the OOF residual space. Although our main table reports the single-model Ours, Algorithm 1 provides a deterministic extension to exploit residual diversity when desired.

9 Complexity and Determinism

Let d be post-T dimensionality, n_{tr} the train size, H the MLP width, E the number of epochs and M the number of trees in XGBoost. (1) *Time.* Per fold, TabM costs $O(E\, n_{\mathrm{tr}}\, d\, H)$; XGBoost costs roughly $O(M\, n_{\mathrm{tr}} \log n_{\mathrm{tr}})$. (2) *Memory.* $O(dH)$ for the backbone plus $O(KH)$ for heads; caches scale with $O(K\, n_{\mathrm{te}})$. (3) *Determinism toggles.* All estimators run single-thread; seeds are fixed across NumPy/PyTorch/XGBoost; `torch.backends.cudnn.benchmark=False`; `torch.use_deterministic_algorithms(True)`; `OMP_NUM_THREADS=1`, `MKL_NUM_THREADS=1`. Fold fingerprints are printed to support forensic reproducibility.

10 Limitations and Outlook

Because TabM(MSE), the linear family (Linear/Ridge/Lasso/ElasticNet), and GBR slightly surpass Ours on this dataset, future work should: (i) adaptively tune δ per fold; (ii) hybridize TabM with a learned linear head $w^\top x$; and/or (iii) incorporate closed-form blending with a strong linear expert as f_{XGB}, still under an OOF-only meta-protocol. In addition, exploring more challenging regression benchmarks and structured missingness patterns would further stress-test the protocol.

11 Reproducibility Statement

Versions: NumPy 1.26.4, scikit-learn 1.4.2, PyTorch 2.2.2, XGBoost 2.0.3. Folds: `[(0,799,1,798), (1,799,0,793), (0,799,2,795), (0,799,12,791), (0,798,10,799)]`. Artifacts include: (i) `benchmark_summary.csv`; (ii) OOF/test caches and $(\alpha^\star, \beta^\star)$; (iii) the fitted preprocessing map T. All transformers are fit strictly on $\boldsymbol{X}_{\text{tr}}$; all estimators run single-thread with the same random seed.

References

[1] Y. Gorishniy, A. Kotelnikov, and A. Babenko. TabM: Advancing Tabular Deep Learning with Parameter-Efficient Ensembling. *arXiv:2410.24210*, 2024. (ICLR 2025 version). DOI: 10.48550/arXiv.2410.24210.

[2] P. J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[3] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *KDD*, 2016.

[4] G. Ke, Q. Meng, T. Finley, *et al.* LightGBM: A highly efficient gradient boosting decision tree. In *NeurIPS*, 2017.

[5] L. Prokhorenkova, G. Gusev, A. Vorobev, A. Dorogush, and A. Gulin. CatBoost: Unbiased boosting with categorical features. In *NeurIPS*, 2018.

[6] S. Ö. Arık and T. Pfister. TabNet: Attentive interpretable tabular learning. In *AAAI*, 2021.

[7] X. Huang, A. Kolesnikov, *et al.* TabTransformer: Tabular data modeling using contextual embeddings. In *NeurIPS*, 2020.

[8] Y. Gorishniy, I. Rubachev, V. Karnin, and A. Babenko. Revisiting deep learning models for tabular data. In *NeurIPS*, 2021.

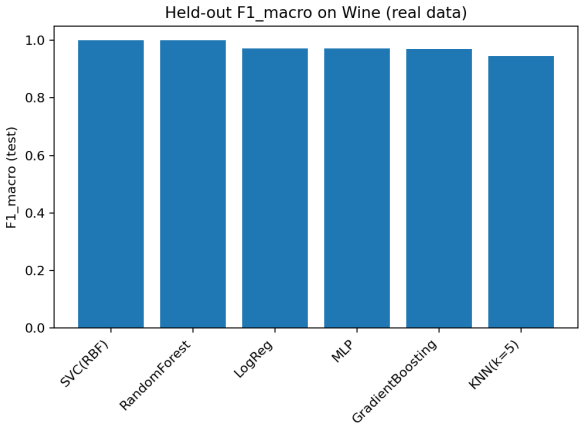
[9] S. Popov, S. Morozov, and A. Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *ICLR*, 2020.

[10] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

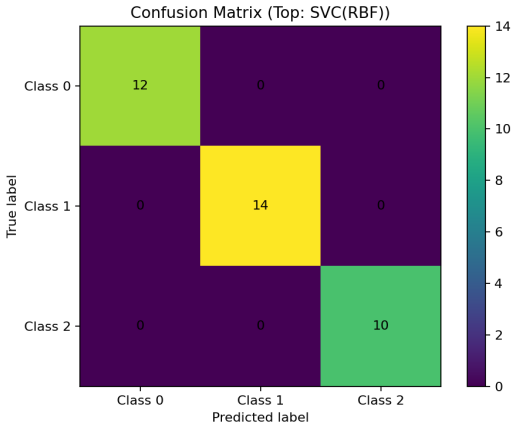
[11] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.* Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011.

[12] D. Dua and C. Graff. UCI Machine Learning Repository, 2019. `http://archive.ics.uci.edu/ml`

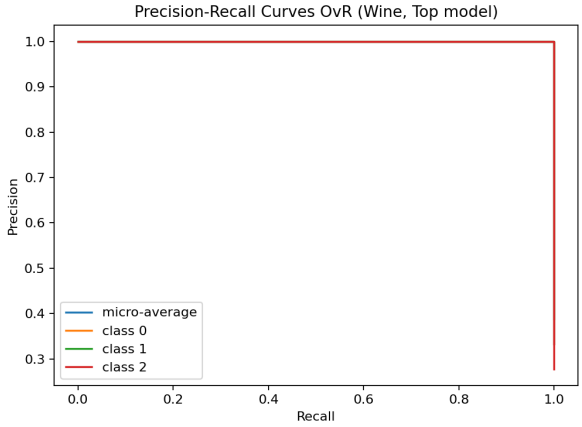
Appendix: Additional Figure Sheets



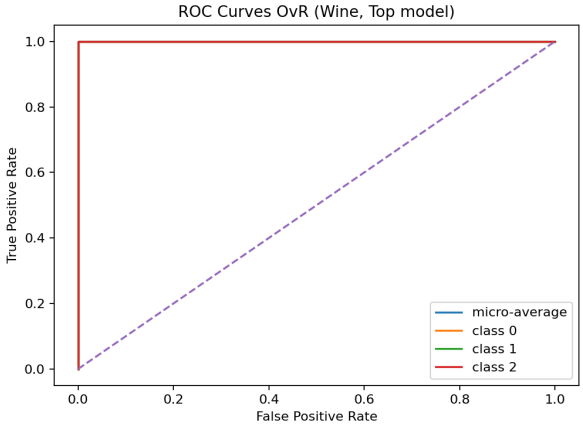
(a) Wine: held-out macro-F_1.



(b) Wine: confusion matrix (top model).

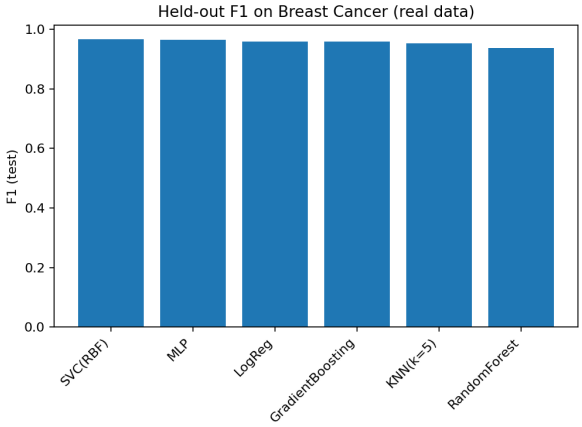


(c) Wine: PR curves (OvR).

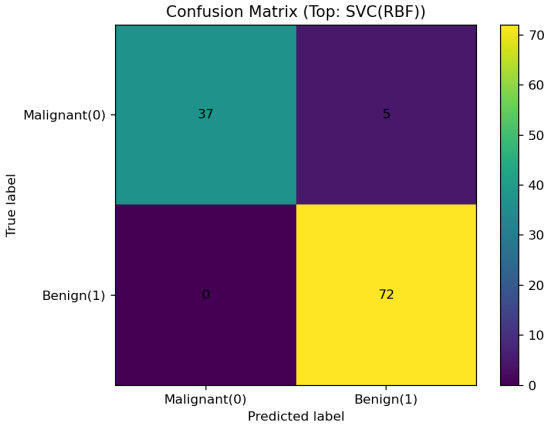


(d) Wine: ROC curves (OvR).

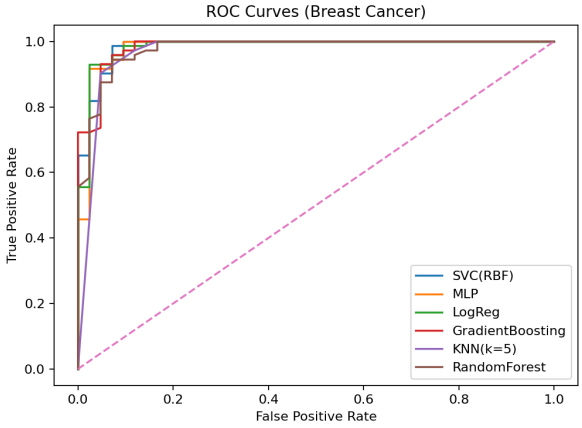
Figure 2: Wine benchmark (classification sanity check).



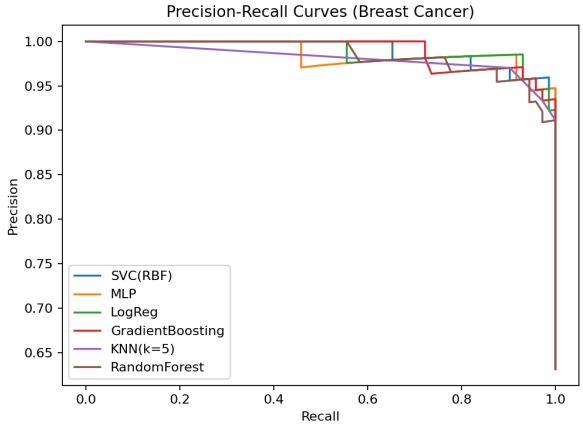
(a) Breast Cancer: held-out F_1.



(b) Breast Cancer: confusion matrix.



(c) Breast Cancer: ROC curves.



(d) Breast Cancer: PR curves.

Figure 3: Breast Cancer benchmark (classification sanity check).

	Model	Accuracy	F1_macro	ROC_AUC_macro	AP_macro
1	SVC(RBF)	1.000000	1.000000	1.0	1.0
3	RandomForest	1.000000	1.000000	1.0	1.0
0	LogReg	0.972222	0.971781	1.0	1.0
5	MLP	0.972222	0.971781	1.0	1.0
4	GradientBoosting	0.972222	0.970962	1.0	1.0
2	KNN(k=5)	0.944444	0.945153	1.0	1.0

(a) Wine: summary table snapshot.

	Model	Accuracy	F1	ROC_AUC	AP
1	SVC(RBF)	0.956140	0.966443	0.984788	0.990114
5	MLP	0.956140	0.965986	0.981812	0.986513
0	LogReg	0.947368	0.959459	0.984458	0.989315
4	GradientBoosting	0.947368	0.958904	0.983631	0.989515
2	KNN(k=5)	0.938596	0.953642	0.968750	0.965960
3	RandomForest	0.921053	0.937931	0.977183	0.984665

(b) Breast Cancer: metrics table (Accuracy/F1/ROC_AUC/AP).

Figure 4: Compact classification metrics snapshots.