COMP9417 Group Project - Built Diff

[redacted]

[redacted]

Connor Schicht

[redacted]

[redacted]

1 Introduction

Despite decades of progress, supervised learning on tabular data is still undoubtedly challenging. As opposed to image and text data, tabular datasets often consist of heterogeneous features, intricate non-linear relationships, and comparatively limited sample sizes. This makes it challenging for standard deep learning approaches to generalize effectively. Consequently, tree-based ensemble methods such as XGBoost and Random Forest remain the optimal approaches on account of their robustness and strong empirical performance across a variety of tasks.

Recent studies on Recursive Feature Machines (RFMs) delineate that kernel methods are capable of providing strong performance on tabular datasets by using gradient-based feature learning. However, expressiveness and scalability is limited by their dependence on a single global representation. The xRFM framework couples adaptive partitioning with localized feature learning in order to bridge the gap between modern tree-based models and classical kernel methods. This method is particularly useful for tabular data where feature interactions are not necessarily uniform, and where global models may be unable to capture subtle, local structure. In this way, xRFM preserves the flexibility of kernel methods while improving adaptability and scalability.

This report evaluates xRFM against XGBoost and Random Forest over a diverse collection of datasets, namely Adult Census Income, Abalone, Wine Quality, Student Performance, and TunaDromd. The selection of these datasets was predicated upon ensuring a span of range of sizes, feature types, and problem settings were covered. This allowed for a thorough comparison across both regression and classification tasks. Overall, results indicate that the performance gap across the three models is marginal, though xRFM can achieve competitive, and in some instances better, predictive performance. However, the key distinctions are noted in computational efficiency, with XGBoost providing quicker training and inference. This, in turn, implies that for several tabular sets, efficiency may be the primary consideration in model choice rather than gains in predictive accuracy.

2 Methodology

2.1 The Average Gradient Outer Product (AGOP)

2.1.1 Definition

The core object underlying xRFM is the Average Gradient Outer Product (AGOP). Given a trained model $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ and data $S = \{x^{(1)}, \ldots, x^{(n)}\} \subset \mathbb{R}^d$, the AGOP is defined as

$$\text{AGOP}(\hat{f}, S) = \frac{1}{n} \sum_{i=1}^{n} \nabla \hat{f}(x^{(i)}) \nabla \hat{f}(x^{(i)})^T \in \mathbb{R}^{d \times d}.$$

When training labels, are multi-dimensional, we use the Jacobian instead of the gradient. [2]

2.1.2 Geometric interpretation

Geometrically, the AGOP aggregates the outer products of gradients across the training data, producing a matrix that summarises the directions along which the model is most sensitive to input perturbations. The AGOP's diagonal indicates coordinates relevant to predictions. As such, a large diagonal entry e.g. $[i, i]$ indicates that perturbing coordinate i leads to large changes in the model's output, making it a natural measure of feature importance. Conversely, small diagonal entries identify coordinates to which the model is insensitive. Beyond individual coordinates, investigating the top eigenvector allows us to identify the effects of a combination of feature perturbations on the prediction. Namely, if entry i is positive and entry j is negative in that eigenvector, the model is said to be sensitive to the contrast between the two features.[2]

These two strategies examining diagonal entries, and examining top eigenvectors are mathematically related. Suppose the eigendecomposition of the AGOP is given by $Q\Lambda Q^T$ (because AGOP matrix is symmetric). Noting that Λ is diagonal, the decomposition of the AGOP matrix is,

$$AGOP = Q\Lambda Q^T = \sum_{k=1}^{n} \lambda_k q_k q_k^T,$$

where q_k is the kth eigenvector. So then, the $[i, i]$th entry is given by $AGOP[i, i] = \sum_{k=1}^{n} \lambda_k q_k[i]^2$. As we can see, each diagonal entry is a weighted sum of contributions from all eigenvectors simultaneously. Thus, the diagonal conflates all directions of model sensitivity into a single number per feature, highlighting its importance. However, it is unable to provide specific information of the combination of features which drives its importance. Whereas examining eigenvectors separately provides a finer decomposition of the sources of model sensitivity.

2.1.3 Comparison to simpler feature importance methods

PCA computes the eigenvectors of the input covariance matrix $\Sigma = \frac{1}{n}\sum_i x^{(i)}(x^{(i)})^T$, identifying directions of maximum variance in the inputs. This is entirely unsupervised i.e. the label y plays no role, so a direction can dominate simply because inputs vary strongly along it, regardless of whether that variation is predictive. [1]The AGOP has an identical mathematical structure but replaces raw inputs with gradients, making it a supervised analogue. Its eigenvectors identify directions of maximum variance in the model's sensitivity rather than the inputs. As a consequence, a feature with high input variance but near-zero gradient will be flagged by PCA but ignored by the AGOP, and vice versa. [2]

Mutual information $I(X_i;Y) = \sum_{x_i}\sum_y p(x_i,y)\log\frac{p(x_i,y)}{p(x_i)p(y)}$ measures the reduction in uncertainty about Y given feature X_i, and is in principle capable of detecting arbitrary nonlinear dependencies. However, it requires estimating the joint density. This introduces significant estimation error in practice. More fundamentally, MI is defined marginally for each feature individually and is a scalar, meaning it cannot detect joint effects of multiple features, nor does it carry any directional information about how features combine to drive predictions.[6] The AGOP addresses both limitations: its eigenvectors directly encode which linear combinations of features jointly influence the model output, with explicit sign and magnitude information.[2]

2.2 xRFM Training Pipeline

Standard kernel methods usually fail on tabular data because they rely on a single global kernel. This creates a major bottleneck when trying to manage the complex and varied feature interactions that naturally occur across different parts of a dataset. The xRFM framework solves this issue by using a binary tree to partition the input space. Instead of one massive model, it fits a localised Recursive Feature Machine at each leaf so every subregion of the data can learn a kernel specifically optimised for its own characteristics.

The training process at each node begins by fitting an RFM to extract the Average Gradient Outer Product. The leading eigenvector of this AGOP serves as the primary splitting tool, as it identifies the axis of maximum output variation. Once this direction is established, the node is thresholded along the axis, and the entire process repeats recursively until a predefined stopping point is reached. To predict a new observation x^*, the tree routes the point through each split until it lands in a specific leaf. Once the point is assigned to a subregion, the model applies the corresponding localised RFM to generate the final output.

The overall training complexity for xRFM is $O(L \cdot T \cdot n_\ell^2 \cdot d)$, where L is the number of leaves, T the iterations per leaf, and n_ℓ the average leaf size. By breaking the dataset into these smaller local patches, the model avoids the $O(n^3)$ cost of a standard global RFM.

2.3 Experimental Setup

All experiments used a 60/20/20 train/validation/test split and random seed 42.

2.3.1 Adult Census Income

The Adult Census Income dataset (48,842 records) was used to benchmark binary classification of incomes exceeding \$50K. Instances with missing values were removed, leaving 45,222 records. Categorical features were label-encoded and all features normalised via a training-only `StandardScaler`. The dataset was chosen for its size ($n > 10{,}000$) and mix of continuous and categorical features. Accuracy and AUC-ROC were recorded.

Hyperparameters were selected by maximising validation AUC-ROC. The final configurations were: XGBoost (`n_estimators=400`, `max_depth=6`, `learning_rate=0.05`); Random Forest (`n_estimators=400`, `max_depth=20`); xRFM (`n_tree_iters=10`, kernel `l2`, bandwidth 5.0, regularisation 10^{-3}, RFM iterations per leaf fixed at 3).

2.3.2 Abalone

The Abalone dataset[7] (4,177 records) was used to benchmark regression of abalone age measured in rings. The single categorical feature (Sex: M/F/I) was encoded using a training-only `OneHotEncoder`, with all continuous features normalised via a training-only `StandardScaler`. Numerical and categorical blocks were concatenated prior to model input. Feature selection was performed using mutual information with a threshold of 0.1. The dataset was chosen for its mix of continuous and categorical features and its complex, non-linear relationships between physical measurements and the target (age). RMSE and R^2 were recorded. Hyperparameters were selected by minimising validation RMSE. The final configurations were: XGBoost (`n_estimators=400`, `max_depth=4`, `learning_rate=0.01`); Random Forest (`n_estimators=200`, `max_depth=20`, `min_samples_split=2`); xRFM (kernel `l2`, bandwidth 10.0, regularisation 10^{-3}, RFM iterations per leaf fixed at 3).

2.3.3 Wine Quality

The Wine Quality dataset (6,499 records) was used to benchmark the regression performance of predicting wine quality from various physiochemical properties. Despite all features being continuous, for ensuring consistency with the modelling pipeline, the preprocessing kept both numerical scaling through a training-only `StandardScaler` and a categorical encoding stage using `OneHotEncoder`, with the outputs being concatenated prior to model input. This ensures alignment with other datasets while isolating behavior on purely numerical inputs.

In comparison to mixed-type datasets, this setting provides a more controlled environment to evaluate how models assess subtle, weakly non-linear relationships in the absence of the added complexity of categorical interactions. In addition, the target variable is known to exhibit noise and limited dynamic range, making it a difficult regression task. RMSE, MAE, and R^2 were recorded, with the hyperparameters being selected by minimizing validation RMSE. The final configurations are given as follows: Random Forest (`n_estimators = 400`), XGBoost (`n_estimators = 400`, `max_depth = 6`, `learning_rate = 0.1`), and xRFM (kernel = 12, bandwidth ≈ 5 , regularization 10^{-3}, iterations = 3).

2.3.4 TUNADROMD

The TUNADROMD dataset (4465 instances) was used to benchmark classification with large features. This dataset has 241 features to be used to classify malware vs good ware. All features contain only binary data with no null values, with the only preprocessing completed being the normalisation of the training data with `StandardScaler` for purposes of consistency across all models and datasets.

Accuracy and AUC-ROC were both used to evaluate validation and test models, with the hyperparameters selected for each model being: Random Forest (`min_samples_split`: 2, `n_estimators`: 300); XGBoost (`learning_rate`: 0.1, `max_depth`: 5, `n_estimators`: 200); xRFM (kernel: 12, bandwidth: 1.0 , regularization 10^{-3}, iterations = 3).

2.3.5 Student Performance

The Student's Performance dataset (2392 instances) was used to benchmark multiclass classification of student's grades ranging from A to F. This dataset was chosen due to containing both categorical and numerical features, while also testing the models' ability to handle multiclass classification. The categorical features were label-encoded with `OneHotEncoder`, and the numerical features were normalized with `StandardScaler` on the xRFM model.

Hyperparameters were selected by maximizing validation AUC-ROC. The final configuration were: Random Forest (`n_estimators=500`, `max_depth=None`, `min_samples_leaf=5`); XGBoost (`n_estimators=750`, `max_depth=3`, `learning_rate=0.01`); xRFM (`n_tree_iters=10`, kernel `l2`, bandwidth 10.0, regularisation 0.05 RFM iterations per leaf fixed at 3).

3 Results

Table 1: Regression results summary. Best result per dataset and metric is **bolded**.

	xRFM				XGBoost				Random Forest			
Dataset	**RMSE**	R^2	**Tr(s)**	**Inf**	**RMSE**	R^2	**Tr(s)**	**Inf**	**RMSE**	R^2	**Tr(s)**	**Inf**
Abalone	**2.23**	**0.541**	0.35	0.0013	2.24	0.538	**0.07**	**0.0008**	2.24	0.535	0.22	0.0794
Wine Quality	**0.63**	**0.463**	0.38	**0.0028**	0.65	0.426	**0.24**	0.0031	0.65	0.431	1.40	0.0299

Table 2: Classification results summary. Best result per dataset and metric is **bolded**.

	xRFM				XGBoost				Random Forest			
Dataset	**Acc**	**AUC**	**Tr(s)**	**Inf**	**Acc**	**AUC**	**Tr(s)**	**Inf**	**Acc**	**AUC**	**Tr(s)**	**Inf**
Adult Census	0.8265	0.7440	26.13	0.0030	**0.8679**	**0.9238**	**0.33**	**0.0003**	0.8624	0.9150	0.71	0.0097
TUNADROMD	0.9933	**0.9997**	0.38	0.0065	0.9922	0.9991	**0.10**	**0.0011**	**0.9944**	0.9995	0.34	0.0213
Student Perf.	**0.7140**	**0.8753**	0.85	**0.004**	0.6743	0.8622	**0.71**	0.023	0.6952	0.8617	0.50	0.107

3.1 Scaling Analysis: Adult Census Income

Figure 1 tracks test AUC-ROC and training time as a function of training set size. Both XGBoost and Random Forest showed consistent performance gains as the training set grew, reaching final AUC values of 0.924 and 0.915

respectively. In contrast, xRFM's performance remained stagnant, with the AUC fluctuating between 0.74 and 0.75 regardless of the number of samples provided.

The training efficiency also showed a significant gap between the models. XGBoost and Random Forest both completed training in under one second at full scale. xRFM, however, required nearly 25 seconds to train on the full 27,132 sample.

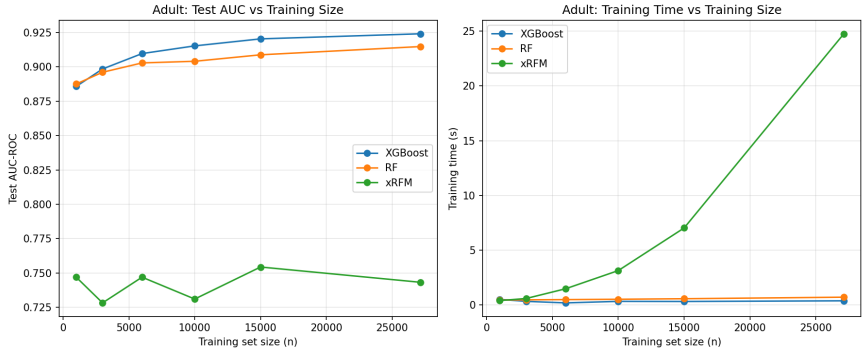

Figure 1: Test AUC-ROC (left) and training time in seconds (right) versus training set size on the Adult Census Income dataset.

3.2 Abalone

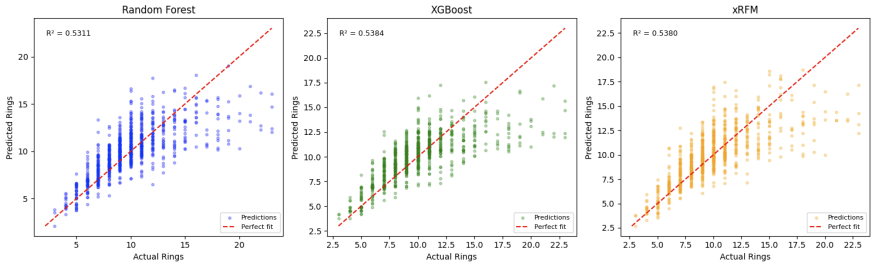

Figure 2: Predicted vs Actual - Abalone Rings

The results are summarised in Table 1. The performed feature selection showed that correlation between male or female and age were very low and dropped the M and F features leaving us with our final models. xRFM achieved the best test RMSE and R^2, narrowly outperforming XGBoost and Random Forest. All three models exhibit a small increase in RMSE from validation to test, consistent with standard generalisation variance rather than model-specific overfitting. The performance gap between models is small, suggesting that on this dataset the choice of model has limited impact on predictive accuracy.

The most significant differences emerged in computational cost. XGBoost the fastest to train at 0.07s, compared to 0.22s for Random Forest and 0.35s for xRFM. xRFM performed close to XGBoost in terms of inference time and Random Forest had the slowest time. Random Forest's slower inference is attributable to averaging across 400 deep trees, while xRFM's overhead comes from its kernel computation during prediction.

Figure 3. confirms these findings visually. All three models produce predictions that track the diagonal well for rings between 5 and 15 but diverge for older abalones (rings > 15), where data is sparse and all models tend to underpredict.

3.3 Wine Quality

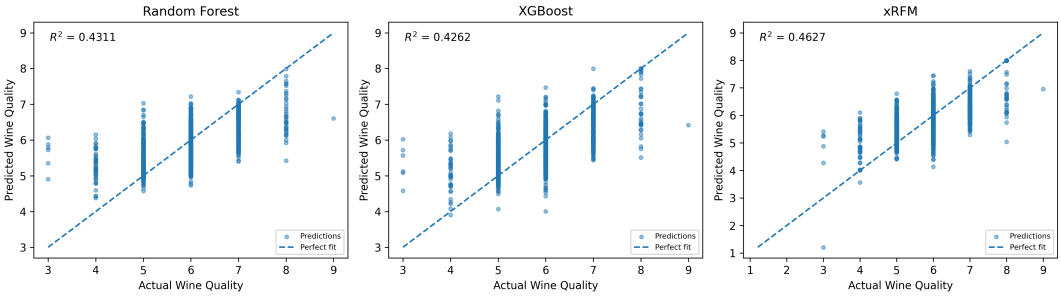

Figure 3: Wine Quality: Predicted vs. Actual

As shown in Table 1, the three models achieve a similar predictive performance. This is particularly highlighted by test RMSE in the range $0.63 - 0.65$ and R^2 scores between $0.43 - 0.46$. xRFM performed the best overall, achieving the lowest test RMSE and highest R^2 of 0.63 and 0.463, respectively, indicating a small yet consistent advantage. The minor validation-test gap suggests good generalization with minimal overfitting.

With reference to Figure 3, all models closely follow the diagonal in the mid-range (quality 5-7), but exhibit regression-to-the-mean behavior. This causes under-predicting of high quality wines and slight over-predicting of lower quality wine. This underscores data imbalance and inherent noise in subjective quality scores. In regards to computation, XGBoost is the fastest, Random Forest is the slowest, and xRFM lies in the middle being slightly slower than XGBoost. xRFM achieves the best accuracy, but performance differences are marginal. The differences between them are driven more by computational efficiency than predictive power.

3.4 TUNADROMD

Table 2 shows all models performed very similarly in terms of accuracy and AUC-ROC, with a respective 0.002 and 0.0006 range. This is evidence of the discriminative nature of the TUNADROMD dataset that favours tree models and makes classification tasks very easy. The major difference between the three models is the training and inference time. XGBoost performs the best here in both situations, being about 20x faster than Random Forest in inference and 6x faster than xRFM.

On this dataset, an interpretability comparison was performed for each xRFM leaf, evaluating the difference in the AGOP, PCA loadings, mutual information scores, and permutation importance methods.

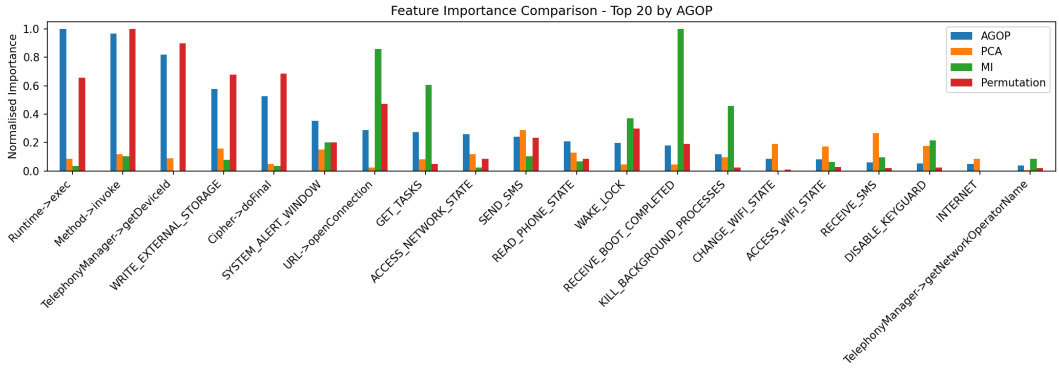

Figure 4: TUNADROMD: Interpretability Comparison

Fig. 4 visualises the difference in various interpretability measures on the top 20 most important features when determined by AGOP. Generally, we can interpret that AGOP and permutation importance agree across the features, however the PCA loadings and mutual information scores differ greatly to AGOP across all of the features. In particular we can see that mutual information places a high importance on `RECEIVE_BOOT_COMPLETED` and `URL->openConnection`,

whereas AGOP places much less importance on those features. This would be a result of mutual information's use of correlation for importance, which fails to recognise that some features are also very important for good ware.

3.5 Student Performance

Table 2 represents the results obtained from creating each model on the Students Performance dataset. xRFM achieved the best AUC-ROC and accuracy, slightly outperforming both the XGBoost and Random Forest models.

The training efficiency of all the models also showed a slight gap between the three models. Random Forest completed its training the fastest with xRFM being the slowest. Illustrating the tradeoff between predictive power with efficiency.

4 Discussion

We utilized 5 tabular datasets to capture the performance of xRFM, to compare it against XGBoost and Random Forest. From the results, we see the empirical advantage of xRFM in regression tasks, as it was able to outperform both XGBoost and Random Forest in all cases. When dealing with classification tasks, we see that xRFM's was able to outperform all cases except when attempting to model Adult Census data. This difference in predictive power when dealing with regression and classification tasks is to be somewhat expected. Previous research highlights that xRFM usually outperforms all models in regression tasks, while still being somewhat competitive with classification tasks.

xRFM generally performs better at regression tasks compared to classification tasks theoretically, as AGOP has the opportunity to obtain more meaningful local gradients and geometry from continuous variables. Thus, resulting in it performing better at regression tasks than classification tasks. AGOP paired with tree partitioning allows xRFM to generally outperform XGBoost and Random Forest as its local feature learning surpasses their capabilities. Furthermore, when understanding the comparison of AGOP to other feature importance methods, we see it supports our theoretical arguments made previously. AGOP is generally able to pick on features that PCA was unable to.

In evaluating xRFM, the Adult Census data must be discussed as it is the case where xRFM performs noticeably worse than XGBoost and Random Forest. We understand the drawbacks of xRFM in classification; however, this difference in AUC-ROC score is something to note. One possible hypothesis is that the comparatively larger dataset used could limit the performance of xRFM as the Adult Census was the only large dataset that we had utilized to model. However, looking further into Test AUC-ROC on varying training set size, we see that xRFM performs considerably worse in all cases of sample size. This, paired with previous research on this model having no issue with dataset size affecting its performance as well, leads us to believe this hypothesis to be incorrect.

Another explanation is that xRFM's local feature learning struggled to exploit the Adult Census dataset as effectively as the other models. This could be a factor of many high cardinal categorical features paired with sparse label encoded features. We know from theory that xRFM's local feature learning works better with continuous, smooth data compared to discrete data. Therefore, when dealing with the adult census, which contains a variety of categorical features with high cardinality, xRFM would not be able to perform its best. Furthermore, XGBoost would be able to perform well under these circumstances due to its tree splitting capabilities and its general robustness to categorically encoded sparsity.

This ultimately leads us to believe that to improve our xRFM model, we would require more time and thought into how we process the high cardinal categorical features. Understanding a better way for us to preprocess the data and build our model on would be a crucial step. Reducing cardinality in categorical features with little or no relevance or utilizing a better categorical encoding method could help increase the performance of the model.

5 Conclusion

This report examined how xRFM, Random Forest, and XGBoost worked on distinct types of tabular datasets for both regression and classification tasks. All models had largely similar performances, with slight differences in RMSE, R^2, and accuracy, although xRFM provided a slight advantage on the Wine Quality dataset. The most evident differences were in computational efficiency, where XGBoost was consistently faster, xRFM incurred higher training costs as datasets got larger, and Random Forest was discernibly slower due to ensemble averaging. Visual analyses also underscored the comparable prediction behavior of the three models, specifically regression-to-the-mean effects and lowered accuracy in sparse regions. xRFM is unequivocally competitive in accuracy, however, its practical benefits rely heavily on the dataset and are often times overshadowed by efficiency prioritizations.

References

[1] Aishwarya. (2026) Principal Component Analysis (PCA). GeeksforGeeks. Accessed 24 April 2026. Available at: https://www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/

[2] Beaglehole, D., Holzmüller, D., Radhakrishnan, A., & Belkin, M. (2026). xRFM: Accurate, scalable, and interpretable feature learning models for tabular data. arXiv. https://arxiv.org/abs/2508.10053

[3] Becker, B. & Kohavi, R. (1996). Adult [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5XW20.

[4] Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). Wine Quality [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C56S3T.

[5] El Kharoua, R. (2024). Students performance dataset. Kaggle. https://www.kaggle.com/datasets/rabieelkharoua/students-performance-dataset

[6] Mishra, P. (2025). Information gain and mutual information for machine learning. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/information-gain-and-mutual-information-for-machine-learning/

[7] Nash, W., Sellers, T., Talbot, S., Cawthorn, A., & Ford, W. (1994). Abalone [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C55C7W.

[8] Ren, W., Zhao, T., Huang, Y., & Honavar, V. (2025). Deep learning within tabular data: Foundations, challenges, advances and future directions. arXiv. https://arxiv.org/abs/2501.03540

[9] TUNADROMD [Dataset]. (2021). UCI Machine Learning Repository. https://doi.org/10.1109/CICT51604.2020.9312053.