Novel binning-based methods for model fitting and data splitting improved machine learning imbalanced data

Husam Abdulnabi* and J. Timothy Westwood

Department of Cell and Systems Biology, University of Toronto Mississauga, Canada

* Corresponding author

E-mail: husam.abdulnabi@mail.utoronto.com (HA)

Abstract

Machine Learning (ML) models may perform inconsistently on individual classes on nominal outputs or ranges on continuous outputs, collectively referred to here as bins. Models should be assessed through metrics that consider each bin individually, called bin metrics. Inconsistent model performance is often due to model fitting with imbalanced data. Towards improving modelling of imbalanced data, novel model fitting methods are proposed including using bin metrics as loss functions and the use of Epoch sampling. Imbalanced data also poses a challenge for appropriate data splitting. Akin split is a novel method proposed that objectively yields the most appropriate data split(s).

Existing and novel model fitting methods were used to fit models, and the models were assessed by a bin metric in in two case studies. The first case study used synthetically generated datasets with different levels of noise and imbalance. On datasets with noise and greater levels of imbalance, Epoch sampling significantly improved the model performance by up to 23.6% while significantly using less resources (computation and time) by up to 57.7% compared to a standard

model fitting method. The second case study used protein-genome interactions data that are often severely right-skewed. Akin split was used to split the data more appropriately than traditional methods. Model fitting methods were tried on two model configurations. The effects of the model fitting methods varied by the model configuration, but all models were significantly improved by up to 57.7% compared to the standard model fitting.

1 Introduction

Machine learning (ML) models are used for their predictive abilities as well as their learned concepts that ultimately guide and stimulate further exploration. Accordingly, ML models can have large downstream ramifications. ML entails numerous steps, each with several tunable parameters. To ensure an appropriate and effective ML model, it is imperative that each step as well as their tunable parameters are thoughtfully attended to.

ML can be broken into 6 steps: data processing, data splitting, model design, model fitting, model scoring, and model analysis (**Fig. 1**). Data processing involves processing the data into observations which are pairs of inputs and outputs. The outputs can be nominal – which have different classes – or continuous. A ML model processes an input and produces its own output, known as a prediction. Data splitting entails dividing the observations into at least a training and a testing set each serving different purposes. Additional sets may also be necessary (see **Sup. Note 9.1.1)**. Model design involves choosing the ML algorithm and configuring its tunable non-learnable parameters, known as hyperparameters. Model fitting entails exposing the training set data to the model to adjust its learnable parameters. Model scoring involves using the testing set which has not been exposed to the model to assess the model's ability to generalize to

unseen data. Model analysis entails extracting learned concepts which may require simplifying to human-understandable forms.

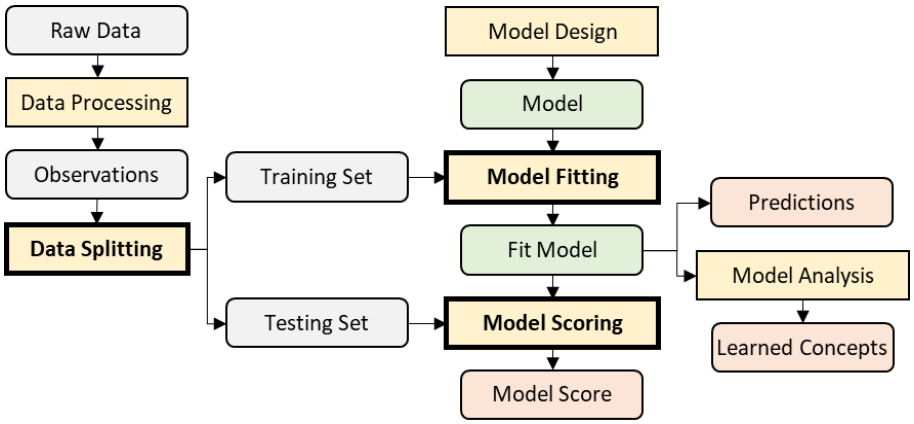


Fig. 1. ML involves 6 steps.

Raw data is processed into observations in data processing which are then divided into training and testing sets in data splitting. The training set is used to fit a model which is constructed in model design. The fit model is scored using the testing set in model scoring. Predictions can be made with the fit model and the learned concepts can be retrieved in model analysis. Flowchart with grey rounded rectangles, yellow rectangles, green rectangles, and red rectangles denoting data, steps, model, and ML products, respectively. Bolded yellow rectangles denote the steps that this study focuses on.

This study focuses on and explores 3 steps of ML: model scoring, model fitting, and data splitting. These steps are significantly challenged by imbalanced data: data where the numbers of outputs belonging to each of the different classes for nominal outputs or ranges for continuous outputs – collectively referred to here as bins – is inconsistent. This study improves the 3 steps by handling the outputs as individual bins instead of an aggregate. Existing methods for model

scoring, model fitting, and data splitting are explored in **Sections 1.1**, **1.2**, **1.3**, respectively. Novel methods for each step are also introduced in those sections and are described in more detail in **Section 2**. Existing and novel model fitting and data splitting methods are experimented in two cases studies. The first case study, **Section 3**, is on synthetically generated data. The second case study, **Section 4**, is on genomics data, specifically protein-genome interactions. Code for methods and case studies are provided at https://github.com/Husam94/BinMeths.

1.1 Introduction for model scoring methods

ML models are scored by metrics that consider the prediction errors which are the differences between the predictions and their corresponding actual output. Most of the commonly used metrics, such as the Mean Squared Error (MSE) and coefficient of determination (also known as r^2) for regression tasks, consider the prediction errors of all predictions together. These metrics do not assess the predictions of a model on individual bins. For continuous values, the number of bins used is arbitrary and the bins are conventionally determined as having equal ranges, referred to here as histogram bins. These metrics can yield identical scores on the predictions for two models that score differently on individual bins, exemplified in **Fig. 2**. To better assess the performance of a model, each bin should be scored individually.

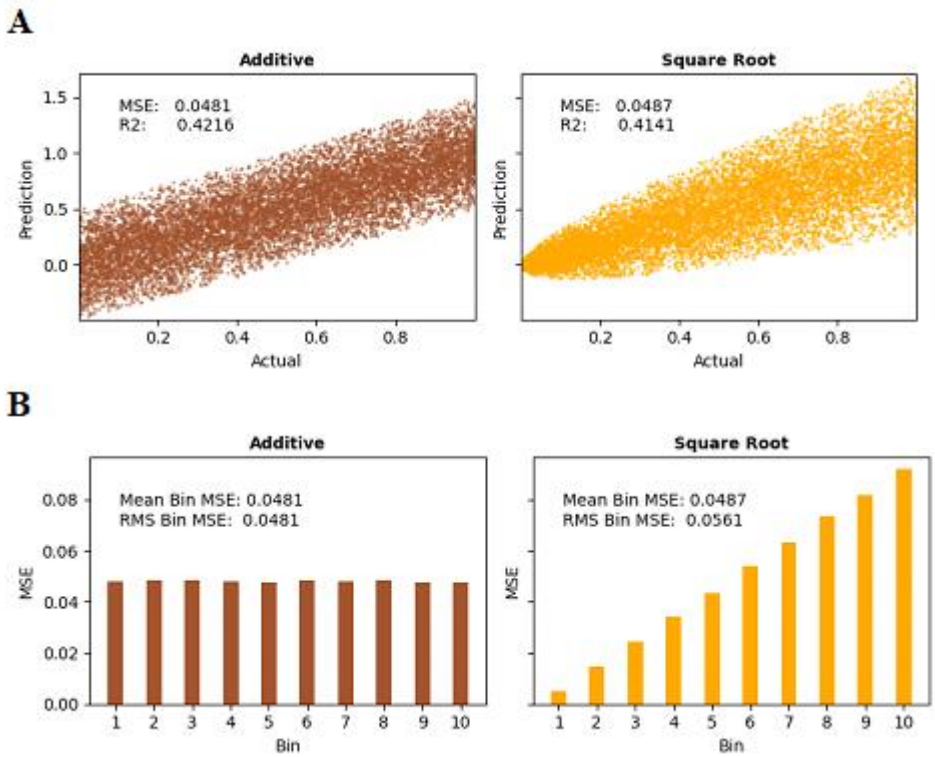


Fig. 2. Models can perform similarly on non-bin metrics but perform differently on different bins.

A) Predictions versus actuals of two simulated models with different types of prediction error called "Additive" and "Square Root". Actual models can have predictions that resemble these prediction errors or can be different.

B) MSE of each histogram bin of the two models, summarized by the mean (Mean Bin MSE) and Root Mean Squared (RMS Bin MSE).

In some ML tasks, some values are more important than others. In other words, the importance, often called relevance, of values is not uniform. For example, in fraud detection, bins with extreme values have more importance than bins with values closer to the mean. Weights are assigned to values based on their importance, referred to here as relevance weights,

and are used in metrics. For regression tasks, this is known as Utility-based regression and can involve the binarization of continuous values (Torgo et al., 2013). For simplicity, in this study, all bins are equally important.

It is preferred that a model performs well on all bins. It is also typically preferred that a model performs consistently on all bins. The performances on the bins can be summarized as the mean but this does not account for the consistency (i.e. variation). To summarize the performances on the bins while accounting for variation for metrics where lower scores are preferred, the Root Mean Squared (RMS) can be used. Extending the previous example, the Mean Squared Error (MSE) of each bin, referred to here as the Bin MSE, is determined and subsequently summarized by the mean (Mean Bin MSE) and the RMS (RMS Bin MSE) (**Fig. 2. B**). While the Mean Bin MSEs are similar for the two models, the RMS Bin MSEs are different, indicating that the "Additive" model has more consistent performance than the "Square Root" model. Metrics that assess the performance on each bin and summarize them (e.g. RMS Bin MSE) are referred to here as **bin metrics**. Assessing the performance on individual bins for regression tasks has been done before (albeit rarely) (e.g. Steininger et al., 2021), but bin metrics is rarely, if ever, done.

Inconsistent model performance on the bins is often due to model fitting with imbalanced data. Imbalanced data consists of bins with proportionally more observations (i.e. over-represented) and bins with proportionally less observations (i.e. under-represented). Generally, over-represented observations have a greater influence on model fitting, resulting in models that are better fit on over-represented observations than under under-represented observations.

The most prevalent example of imbalanced continuous data is *Tailed* data: data with continuous outputs that are increasingly rarer away from the mean. Tailed data includes right-

skewed, left-skewed, and bell-shaped distributions, exemplified in **Fig. 3**. By simply predicting the mean of Tailed data for all observations without fitting to the inputs, models can score deceptively well on non-bin metrics. This is especially dangerous when these models are used for predictions and/or learned concepts that drive later studies. Crucially, histogram binning of Tailed data results in bins away from the mean with increasingly fewer values, often with too few values for adequate model scoring and data splitting. Thus, different binning methods may be necessary for Tailed data.

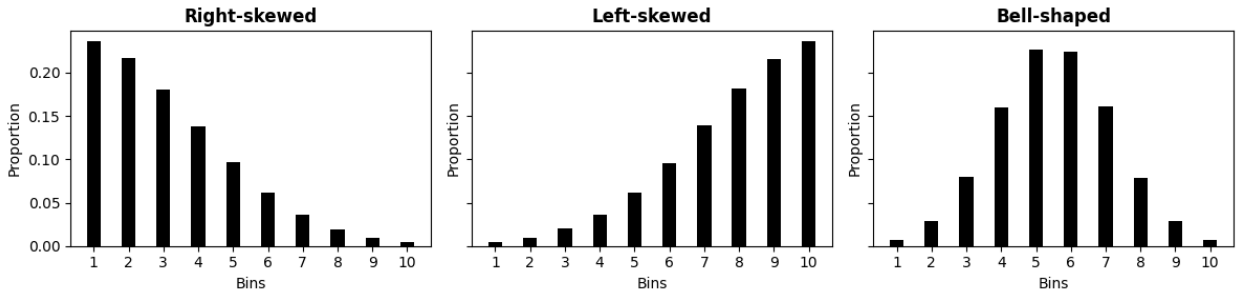


Fig. 3. Tailed data includes data with outputs that are right-skewed, left-skewed, and bell-shaped.

For each set, 10,000 samples were drawn from truncated normal distributions with means of 0 and standard deviations of 1. Right-skewed, left-skewed, and bell-shaped distributions have standard deviation bounds of [0, 3], [-3, 0], and [-3, 3], respectively. For each distribution, the samples are binned into 10 histogram bins and the proportion of samples belonging to each bin were determined.

1.1.1 Binning methods for Tailed data

Instead of using equal ranges for each bin for Tailed data, ranges that are progressively larger away from the mean can be used to capture more values. The bin ranges can be

determined by first transforming the output values by a monotonic transformation that compresses the values away from the mean towards the mean to produce a distribution that better resembles a uniform distribution. Monotonic transformations, simply called transformations hereafter, are functions that do not change the relative order of the values. Bin ranges in the transformation scale are then determined by histogram bins. The transformation is then inversed on the bin ranges, producing bin ranges in normal scale, referred to here as transformation bins. This method of determining bins, referred to here as **transformation binning**, is rarely if ever done.

The transformation bins for Tailed data can be determined by using log-based functions. For right-skewed values, the log function ($\log(1 + x)$) is used. The right-skewed values are first translated by their minimum ($x = x + \min(x)$) to avoid applying the log function to negative numbers. For left-skewed data, the power function (x^q) can be used but the values can be treated as right-skewed values by reflecting them ($x = -x$). For bell-shaped distributions, the log-modules function can be used (John & Draper, 1980). Before applying the log-modules function, the values should be translated by their mean ($x = x - \bar{x}$) such that the mean is 0.

Importantly, inversing log-based functions to determine bin ranges involves the exponential function (e^x), producing bin ranges that exponentially increase away from 0. This can result in bins near 0 having too small of a range that capture too few values. Additionally, this can result in too many bins spanning too small of a total range and too few bins spanning too large of a total range, depending on user preferences. To produce bin ranges that are progressively larger away from 0 but are not exponentially increasing, the $\tanh$ function can be used (**Sup. Eq. 1**, **Sup. Eq. 2**).

There must also be consideration of scaling the values with a scale factor prior to applying a transformation for transformation binning. Scale factors affect the distribution of values in the transformed scale, thereby affecting the bin ranges and number of observations in each bin. Based on user preferences, the minimum and/or maximum bin range proportion as well as the minimum number of observations in a bin could be considered when determining the scale factor. The scale factor can be objectively determined to best fulfil user preferences, referred to here as **objective binning**, but this is rarely, if ever, done. Notably, even with transformation binning, the number of observations in each bin can be inconsistent.

1.2 Introduction for model fitting methods

The following methods are only applied during model fitting and not model scoring and can be used individually or in combination. The methods are categorized into three categories. The first category is using the bin metric that assesses the model performance as the loss function which are metrics used to guide model fitting. Using the same metric for model fitting and scoring may guide model fitting towards better scoring. The other two categories of model fitting methods are output weighting and data balancing.

1.2.1 Output weighting

Output weighting methods assign weights to output values to have different influences on the loss function. To address imbalanced data, greater and lesser weights are assigned to under- and over-represented values, respectively. There are two effective output weighting methods: Density-based Weighting (DenseWeights) and reciprocal weighting, exemplified in **Fig. 4**.

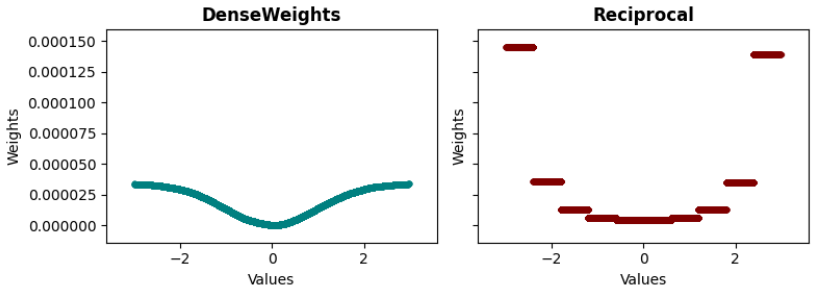


Fig. 4. Different output weighting methods assign different weights.

Data for all examples is the Bell-shaped distribution from **Fig. 3**.

DenseWeights are determined with default parameters from Steininger et al., 2021.

DenseWeights assigns weights to continuous outputs based on the complement of a Kernel Density Estimation (Steininger et al., 2021). DenseWeights was shown to perform well on a variety of tasks compared to other state-of-the-art model fitting methods.

Reciprocal weighting determines weights of values as the reciprocal of the proportion of values belonging to their respective bin (**Sup. Eq. 3**). Reciprocal weighting can be used for both nominal and continuous outputs. Notably, the influence of a value on a bin metric is roughly the reciprocal of the proportion of values belonging to its bin. Thus, using bin metrics as a loss function is nearly equivalent to using non-bin metrics with reciprocal weighting. However, this is not always true for Neural Networks (NNs). NNs are fit to the training data through batches which are subsets that typically range from 32-1024 observations. The batches are generated by a random partition of the training set. Using bin metrics as a loss function on batches is nearly equivalent to using reciprocal weighting based on each batch. Output weights, however, are determined based on all the training data used in the epoch.

1.2.2 Data balancing

Data balancing methods aim to reduce the imbalance of the number of observations in each bin. Importantly, data balancing methods require each observation to belong to a single bin, but observations may have outputs with multiple values thereby belonging to multiple bins. For observations with multiple output values, the bins can be determined as a combination of bins. For example, for observations with 2 output values and each value binned into 10 bins, there are a total of (10*10) 100 combinations of bins. Alternatively, the bins can be determined by clustering the observations by their multiple output values.

Data balancing methods either resample or create synthetic data. Resampling methods either sample and/or oversample (i.e. replicate) observations belonging to over- and under-represented bins, respectively. Synthetic data methods create artificial observations that belong to under-represented bins based on the observations in a dataset through methods such as *k*-nearest neighbors (Branco et al., 2017; Chawla et al., 2002). Oversampling and synthetic data creation methods expand the dataset which requires more computing resources for storage and may require more computing resources for modelling. Sampling, however, results in fewer observations that may require fewer computing resources for modelling. This study only focuses on sampling for data balancing.

Sampling can be done through weighted random sampling. Observations can be weighted by methods mentioned in **Section 1.2.1**. Observations are randomly selected using their weights without replacement until the desired number of total observations is obtained. As this method relies on randomness, the number of sampled observations in each bin can vary considerably in different samples especially with a low number of total observations.

Alternatively, the number of samples from each bin can be determined prior to applying sampling methods. The number of observations in each bin can be that of the bin with the fewest

observations, but this may result in too few total observations to model effectively. Alternatively, the number of observations in each bin can be based on the equal proportions. For example, given 10,000 observations and 10 bins, each bin would have ($\frac{10,000}{10}$) 1,000 observations. Still, under-represented bins are exhausted, resulting in fewer total observations than desired. A novel method called **Harpoon sampling** (see **Section 2.1**) guarantees the desired number of total observations while minimizing the variation of the number of observations in each bin, exemplified in **Fig. 5**.

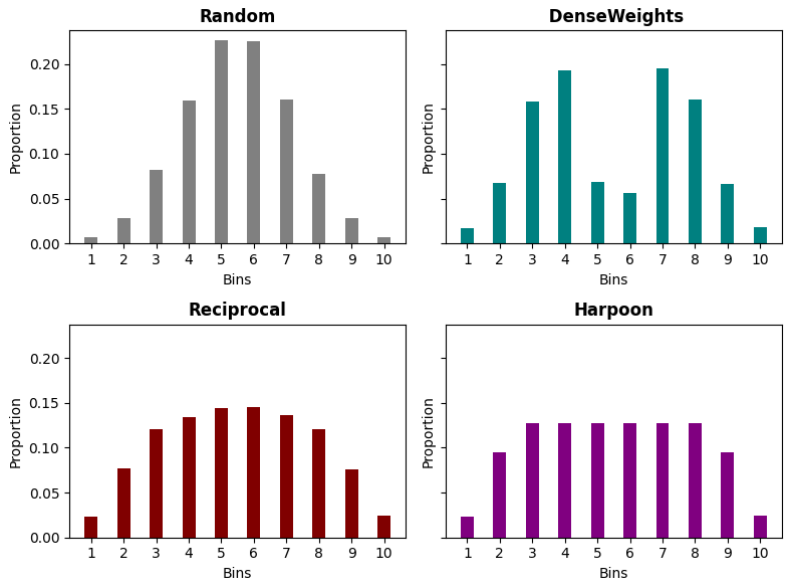


Fig. 5. Harpoon sampling produces the most balanced proportion of observations in each bin compared to other sampling methods without replacement.

Data for all examples is the Bell-shaped distribution from **Fig. 3**.

Sampling with different techniques of random weighted sampling (weighted by DenseWeights, reciprocal weighting), and Harpoon sampling. A sampling proportion of 0.3 is used for all samples, resulting in 3,000 values. The proportions of samples belonging to each bin of each set of values are determined.

After the desired number of observations in each bin are determined, observations can be selected randomly or based on their inputs. The latter involves clustering observations by their inputs through *k*-nearest neighbors (or similar) for each bin and samples from each cluster which reduces the redundancy of the inputs of the observations (Bach et al., 2019; Beckmann et al., 2015). This selection procedure requires additional considerations, computational resources, and time. The following study only selects observations from each bin randomly.

Importantly, sampling methods, even those based on inputs, involve randomness whereby different samples can have different observations. Model fitting with different data can ultimately result in models with substantially different performances. Hence, model fitting with sampling requires multiple samples as an additional level in model scoring. However, this costs additional computational resources and time. NNs provide a unique opportunity to absolve the need for multiple samples. NNs are fit to the training data in cycles, known as epochs. Instead of sampling the training data before modelling, a novel method called **Epoch sampling** (see **Section 2.1**) samples the training data at each epoch thereby introducing new samples at every epoch.

As mentioned, model fitting methods can be used individually or in combination. When using output weighting and sampling, output weights are determined after sampling. When using Epoch Sampling, output weights are determined after sampling within each epoch.

1.3 Introduction for data splitting methods

The observations of a dataset should be divided such that each set contains observations that are independent to the observations of another set and that the output values of the sets have

263 similar distributions (i.e. identically distributed). Non-independent observations that are spread
264 across different sets affect both model fitting and scoring which leads to improper model
265 evaluation, known as data leakage. Moreover, identical distributions are necessary to ensure that
266 the model is fit and scored on comparable data. Scoring models using data that has a different
267 distribution than the data used in model fitting is inappropriate. For example, fitting on Tailed
268 data without any additional model fitting method results in models that better predict values
269 closer to the mean. Scoring a model by only values close to the mean does not account for its
270 performance on values away from the mean and vice versa.

271 The most common method of data splitting is through random sampling, known as
272 Random split. Non-independent observations must be grouped prior to splitting to ensure they
273 are allocated to a single set.

274 Sequential data, such as genomics data, requires additional considerations for
275 independent data splits. Observations from sequential data are windows (i.e. a sub-sequence) of a
276 larger sequence (called the references sequence). Overlapping windows are non-independent and
277 thus must be grouped before splitting. Also, as neighboring windows are likely to be related, it is
278 suggested that neighboring observations also be confined to a single set.

279 For data with multiple reference sequences, overlapping and neighboring observations
280 can be confined to a single set by assigning entire reference sequence(s) to a single set. For
281 example, genomics data is typically based on a genome that contains multiple chromosomes. The
282 prevailing practice is to assign all observations belonging to a specific chromosome(s) to a single
283 set. Typically, the numbers of observations belonging to each reference sequence varies (i.e.
284 different reference sequences have different numbers of observations belonging to them).

Dividing observations such that each set contains a desired proportion of observations is difficult to attain, if possible.

To ensure the desired proportion in each set while limiting data leakage of neighboring interactions, splitting can be done by using larger non-overlapping windows, referred to here as **Window split**. This method first sorts the observations by their location on the reference sequence(s), groups non-independent observations, and uses larger non-overlapping windows that are of sizes corresponding to the desired proportions of each set to split the observations. For example, given 100 observations with no overlap that are sorted by their location, and a desired proportion of observations of 0.7 and 0.3 for the training and testing sets, windows of size 70 and 30 are used, respectively. Different splits can be generated by sliding (i.e. translating) and/or changing the order of the windows on the ordered list.

The methods mentioned thus far (Random and Window split) do not guarantee that the sets are identically distributed. A common method for data splitting for identical distributions is Stratified split which random samples observations based on bins, ensuring each bin is proportionally distributed among the splits. However, Stratified split does not directly accommodate outputs with multiple values, requiring each observation to belong to a single bin which is similar to data balancing methods (see **Section 1.2.2**). More problematic still, Stratified split does not accommodate for non-independent observations and is not possible for sequential data when trying to confine neighboring windows.

Akin split is a novel objective data splitting method that produces a data split(s) that contains sets that are independent, have the most identical output distributions, and have the desired proportions (see **Section 2.3.1**). More specifically, data splits are generated through Random or Window split and scored by the identicalness of the output values of its sets through

a novel method called the **Akin score** (see **Section 2.3**) which can directly handle multiple-output values. The data split(s) with the best Akin score is selected. As Window split can be used to generate splits, Akin split can be used to split sequential data, such as genomics data, appropriately and effectively.

2 Novel methods

2.1 Harpoon sampling

Harpoon sampling is a sampling method that determines the number of samples per bin by iteratively sampling each bin without replacement until a desired number (or proportion of the original) samples is attained (**Fig. 6**). This method aims to capture all under-represented values while sampling over-represented values to produce as balanced a distribution as possible without replacement.

Input:
- b: number of bins
- $q[1..b]$: array of available samples per bin
- n: total number of samples desired

Output:
- $g[1..b]$: array of samples to draw from each bin

Procedure:

```
Initialize g[1..b] ← array of zeroes
while n > 0:
    for bin in Shuffle(1 to b):
        if q[bin] > 0:
            g[bin] ← g[bin] + 1
            q[bin] ← q[bin] − 1
            n ← n − 1
        If n == 0:
```

Break

return g

Fig. 6. Algorithm of Harpoon sampling.

This algorithm describes the sampling logic used to balance bins while ensuring a fixed total sample size.

2.2 Epoch sampling

Epoch sampling performs sampling in each epoch during NN fitting. As different observations are used in each epoch, Epoch sampling absolves the need for modelling multiple samples as an additional level of model scoring, thereby saving resources and time. It is recommended that Harpoon sampling is the sampling method used in Epoch sampling.

2.3 Akin Score

The Akin Score assesses the similarity of the output distributions of a data split. It is essentially a summary of pairwise comparisons of the proportions of values belonging to each bin of each set. For example, given a data split with 3 sets $\{A, B, C\}$ there are 3 pairwise comparisons $\{(A, B), (A, C), (B, C)\}$.

The output of an observation can be a single value or be multiple values. Multi-value outputs can have dependencies with other output values. For example, values in a sequence are dependent on each other. Output values are grouped by dependency, referred to here as output groups. Values that have no dependencies form groups of 1. For observations with multiple output groups, each pairwise comparison of a set involves pairwise comparisons of each matching output group. For example, for observations with 2 output groups $\{H, J\}$ and 3 sets

$\{A, B, C\}$, the pairwise comparisons become

$\{(A_H, B_H), (A_J, B_J), (A_H, C_H), (A_J, C_J), (B_H, C_H), (B_J, C_J)\}$.

The distribution of each output group is represented as the proportion of values belonging to each of their bins, referred to here as bin proportions. Notably, the sum of bin proportions is 1. Bins with lower proportions can be given more influence in later operations by using the reciprocal of the bin proportions but these values must be divided by the total sum to preserve the sum as 1.

The similarity of two bin proportions is scored by the Minkowski distance, producing scores where a lesser score indicates greater similarity. The Minkowski distance can be the Manhattan Distance or the Euclidean Distance when the order of the norm is 1 and 2, respectively. Since the sum of the bin proportions is 1, the minimum (0) and maximum of the Minkowski Distance can be determined and be used in scaling the distance values, producing the Scaled Minkowski Distance (SMD). Notably, the range of possible values of the SMD is [0,1]. The SMD is determined for each pairwise comparison, resulting in multiple values. The SMDs are summarized into a single value by the RMS to account for the average and the variation of the pairwise comparisons.

Notably, the Akin Score assesses the identicalness of the sets of a split regardless of the number of output values, number of output groups, observation grouping, and the type of output values (nominal or continuous).

2.3.1 Akin Split

Akin split selects a split(s) from a pool of splits that has the best (i.e. lowest) Akin Score thereby selecting a split that has the most identical distributions of output values. The splits are

generated by Random or Window Split. Notably, generating and scoring splits is computationally inexpensive and can be parallelized.

3 Case study: synthetic right-skewed data

Models fit on imbalanced data often results in inconsistent model performance. Existing and novel model fitting methods are tried towards producing models with more consistent model performance. The effects of these methods on model performance individually and in combination are not known prior to modelling and may be dependent on the data that is modeled, the metrics used for model scoring, and the ML algorithm used.

ML algorithms contain non-learnable tunable parameters, known as hyperparameters, that must be set before modelling and that substantially influence model performance. The different possible configurations of hyperparameters are a search space that must be searched for optimizing model performance. Some ML algorithms, such as NNs, have many hyperparameters resulting in a search space that may be too large to sufficiently search due to limitations in computational resources and time. Moreover, the performance of the different configurations of hyperparameters may depend on the model fitting methods used. Thus, hyperparameters and model fitting methods may be codependent. The model fitting methods can be searched along with the hyperparameters, but this further expands a search space that may already be too large to adequately search.

To reduce the influence of the hyperparameters of a ML algorithm while determining the effects of the model fitting methods, the following case study generates data and models the data using the same configuration of a ML algorithm. The effects of the methods are explored on datasets that vary by amounts of noise and right-skew. In theory, and from experiments that are

not included in this study, the effects of these methods on right-skewed data are similar to other Tailed data.

3.1 Methods

See https://github.com/Husam94/BinMeths/C1-Synth/ for complete details.

3.1.1 Neural networks

Inspired by Steininger et al., 2021, a NN with 3 Dense layers each with 10 nodes and ReLU activation functions between the Dense layers was used, referred to here as the Base NN. Batch normalization layers were added between the Dense layers as they are found in most current NN architectures.

3.1.2 Data generation

3.1.2.1 Initial data generation

A 10-feature input was generated by sampling uniform distributions with a range [-1, 1]. A total of 1,000,000 10-feature inputs were generated. A Base NN was randomly initialized by Kaiming Initialization and produced corresponding outputs from the inputs. The pairs of input-output are referred to as original data. This procedure was repeated 20 times, generating 20 sets of original data from 20 different initializations of Base NNs, referred to as synthetic repeats. The outputs of the synthetic repeats were visually inspected for bell-shaped distributions. Of the synthetic repeats, 5 were chosen that best resemble bell-shaped distributions for the target and control values.

3.1.2.2 Noise

The following applies to each of the synthetic repeats. Three noise levels were used: $g = 0, 0.5, 1.0$, corresponding to noise conditions N1, N2, N3, respectively. The noise is applied as the product of three components: the noise level g, the standard deviation of the original outputs $\sigma_{original}$, and samples from a truncated normal distribution. The resulting noisy output values are defined by the following equation:

$$y_{noised} = y_{original} + \left(\sigma_{original} * g * \mathrm{TN}(0,1,2)\right)$$

where:

- $y_{original}$ are the original output values
- $\sigma_{original}$ is the standard deviation of the original output
- g is the noise level
- $\mathrm{TN}(0,1,2)$ represents a sample from a truncated normal distribution with mean 0, standard deviation 1, and bounds of ±2 standard deviations.

This application of noise ensures that noise is scaled appropriately and bounded to avoid extreme perturbations. Notably, noise level N1 ($g = 0$) results in no added noise, preserving the original output values. The different levels of noise are exemplified in exemplified in **Fig. 7**.

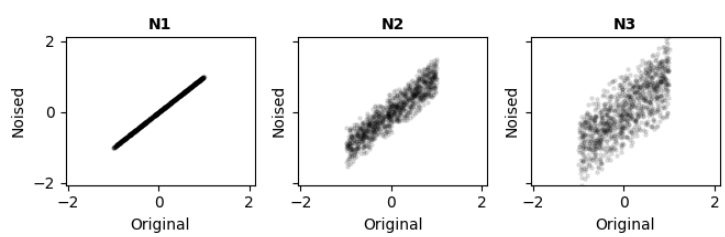


Fig. 7. Examples of levels of noise.

Different levels of noise were added ("Noised") to a random sample of 1,000 values of the outputs of a synthetic repeat ("Original").

3.1.2.3 Right-skew selection

The following applies to each level of noise. Different *levels* of right-skewed data were obtained through weighted random sampling. Observations were weighted by a two-part weighting scheme based on the output values. The weighting function is defined as:

$$w = \frac{\mathrm{BinProps}(y_{norm}, 1000)}{e^{r * y_{norm}}}$$

where:

- y_{norm} are the output values after adding noise, clipping, and min-max normalization
- $\mathrm{BinProps}(y_{norm}, 1000)$ computes the proportion of samples in each of the 1000 histogram bins
- r controls the degree of right-skew with values r$r = 0,2,5,8$ corresponding to skew levels R1 through R4

The exponential term $e^{r * y_{norm}}$ increases the weight decay for higher output values, thereby inducing right-skew. When $r = 0$, the dataset remains balanced (R1). Each skewed dataset contains 10,000 observations sampled according to these weights, exemplified in **Fig. 8**.

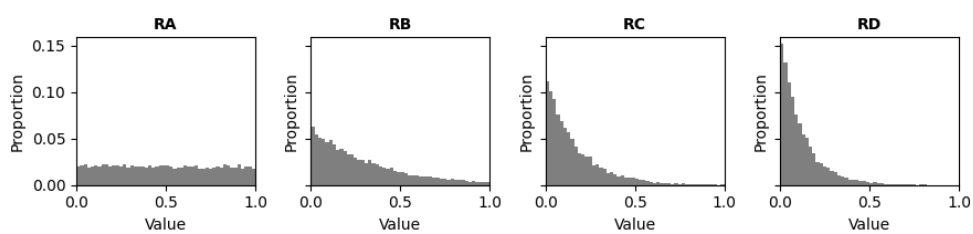


Fig. 8. Examples of levels of right-skew.

Histograms (50 bins) of the output values of the different levels of right-skews generated from one synthetic repeat.

3.1.2.4 Binning and data splitting

For each dataset, the observations were binned into 10 transformation bins by scaling the values by a scaling factor and using the tanh function. The scaling factor was objectively determined as the scaling factor that yielded the lowest variation of the proportions of observations belonging to each bin and having a maximum bin range less than 0.3x the total range of output values. The scaling factors determined ranged from 0 to 1.8 depending on the level of right-skew.

For each dataset, the data was split into training, stoppage (see **Supplementary Note 9.1.1**), and testing sets with proportions of 0.6, 0.2, 0.2, respectively, using Akin split with Random split as the split generator. Each split is referred to here as a split repeat. Binning and split information for different right-skew levels for the first synthetic repeat and noise level N1 are provided in **Sup. Table. 1 – Sup. Table. 4**. Datasets with different synthetic repeats, level of noise, and split repeats have similar binning and splitting for each corresponding level of right-skew.

3.1.2.5 Output weighting

Output weights were determined by the output values of the observations in the training set. For experiments that combined output weighting and Epoch sampling, output weights were determined within each epoch of the sampled training set after Epoch sampling. DenseWeights

were determined using the default parameters described in Steininger et al., 2021. Reciprocal weights were determined as the reciprocal of the proportion of values in each of the bins.

3.1.3 Model scoring

The objective is consistent model performance within each bin and of all bins. For each bin, the corresponding predictions are scored by the Root Mean Squared Error (RMSE). The RMSEs of each bin (Bin RMSEs) are summarized by RMS, yielding the RMS Bin RMSE (RBR). A lower RBR is preferred.

The learnable parameters of NNs are initialized randomly and are fit using stochastic methods which can result in substantially varying model performances. Considering computing budgets, each experimental unit was repeated 15 times, referred to as model repeats. For later statistical assessment, different sets of model repeats were simulated by bootstrapping the existing model repeats (Efron & Tibshirani, 1986). Based on computing budgets, 100 bootstrap sets of model repeats were sampled for each set of model repeats.

Commonly, a single model out of a set of model repeats that best performs on the stoppage set is used for further assessment. This selection criteria, however, lacks reproducibility as a different set of model repeats can yield a substantially different best performing model. Instead of using a single best performing model, the best 3 models were used. The predictions of the best 3 models were averaged, akin to an average ensemble model, simply referred to as an ensemble. The ensemble predictions were scored by the RBR, referred to as a bootstrap score. Each experimental unit had 100 bootstrap scores as there were 100 bootstrap sets of model repeats.

3.1.3.1 Comparisons of models

A subject experimental unit was compared to a corresponding reference experimental unit by the pairwise comparisons of their bootstrap scores, resulting in (100*100) 10,000 comparisons which simulates repeating the modelling of the two experimental units 10,000 times. Each pair of bootstrap scores were compared by percent relative change, referred to here as the pairwise bootstrap score. Negative pairwise bootstrap scores indicate the subject has a lower RBR than the reference which is preferred. For each level of noise, for each level of right-skew, the pairwise bootstrap scores of synthetic and split repeats were pooled, totaling (3*3*10,000) 90,000 pairwise bootstrap scores.

The mean and the standard error of the pooled pairwise bootstrap scores were determined for each dataset and for each experimental unit. For statistical assessment, the 5th (P_5) and the 95th (P_{95}) percentiles of the pairwise bootstrap scores were determined, simulating a one-sided statistical significance level of 0.05 and a two-sided statistical significance level of 0.1 (Efron & Tibshirani, 1986). If $0 > P_{95}$, percentile, the subject model is significantly better than the reference model. If $0 < P_5$ percentile, the subject model is significantly worse than the reference model. If $P_5 < 0 < P_{95}$, the subject model is not significantly different from the reference model.

3.1.4 Model fitting

All modelling used the Base NN, a batch size of 512, RBR as the stoppage metric, and a patience of 10. The choice of patience was informed by preliminary trials (not included here) which indicated that increasing patience did not yield significant performance improvements and this setting reduced computation resources. Also, all models used the Adam optimizer (Kingma & Ba, 2014), selected for its widespread adoption and proven effectiveness across a broad range

of NN applications. The Adam learning rate used for all experiments was determined as the learning rate that best reduces the RBR given the standard model fitting method and a budget of 200 epochs (**Sup. Table. 5**).

The *standard* model fitting method uses RMSE as the loss function, no output weighting, and no sampling. RMSE is chosen as the predictions at bin-level are assessed by RMSE but also because it and the closely related MSE are among the most widely used loss functions for regression tasks.

3.2 Results and discussion

The objective of this case study is to minimize the RMS Bin RMSE (RBR) on modelling datasets with different levels of noise and right-skew. Towards minimizing the RBR, three different categories of model fitting methods of bin metrics as the loss function, output weighting, and data balancing were tried first individually and then in combination.

3.2.1 Model fitting with different loss functions

Compared to models fit with the standard method which used RMSE as the loss function, models fit using RBR as the loss function performed significantly better on datasets with noise (N2 and N3) and right-skew (R2, R3, R4) by up to 23.0% (**Table 1**). On datasets without noise (N1) and those without right-skew (R1), models fit with either method performed similarly. The models fit with RBR as the loss function were not significantly worse than those fit with the standard method for any of the datasets. Using RBR as the loss function on any datasets yielded models that performed as good if not better than those fit using the standard method. Notably,

since the RBR is used as the evaluating metric, it was expected that using RBR as the loss function yields models that perform better on the RBR than using RMSE as the loss function.

Table 1. Models fit with RBR as the loss function performed better than those fit with RMSE on datasets with greater levels of right-skew and noise.

Mean pairwise bootstrap scores comparing models fit with different loss functions ("Loss Func.") to models fit with RMSE as the loss function.

The "Shuffled RMSE" refers to models fit with RMSE as the loss function but with the outputs in the training and validation sets shuffled, serving as a negative control.

See **Sup. Table. 6** for full bootstrap statistics.

Negative values indicate that models fit using the subject configuration performed better than models fit using the reference configuration. Bolded values indicate that the models fit using the subject configuration are significantly better or worse than models fit using the reference configuration.

"Noise ID" and "Right ID" refer to noise level and right-skew level of the dataset, respectively.

Dark to light color scale visualize values from -25 to 0.

Noise ID	**N1**				**N2**				**N3**			
Right ID	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
Loss Function												
RMSE	0.7	1.1	0.8	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RBR	-1.5	12.5	27.7	20.4	0.3	**-5.0**	**-15.4**	**-15.2**	0.0	**-13.1**	**-23.0**	**-19.2**
Shuffled RMSE	**748.0**	**956.5**	**819.1**	**516.4**	**66.3**	**91.4**	**65.2**	**35.1**	**21.2**	**30.1**	**17.1**	**6.7**

3.2.2 Model fitting with different output weighting methods

Compared to models fit with the standard method which does not use output weighting, models fit with reciprocal weighting for output weighting performed similar to those fit with

RBR as the loss function (**Table 2**). They performed significantly better than those fit with the standard method on datasets with noise and right-skew by up to 22.6% and were not significantly worse for any of the datasets. Models fit with DenseWeights also performed significantly better than those fit with the standard method on datasets with noise and right-skew by up to 17.5%. However, models fit with DenseWeights were significantly worse than those fit with the standard method on datasets with noise and without right-skew (R1) by up to 9.1%.

Table 2. Models fit with reciprocal weighting performed better than those fit without output weighting on datasets with greater levels of right-skew and noise.

Mean pairwise bootstrap scores comparing models fit without output weighting ("None") or with output weighting methods of DenseWeights ("DenseW") or reciprocal weighting ("Recip.") to models fit without output weighting.

See **Sup. Table. 7** for full bootstrap statistics.

See **Table 1** for additional descriptions.

Noise ID	**N1**				**N2**				**N3**			
Right ID	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
Weighting												
None	0.7	1.1	0.8	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
DenseW	27.9	10.8	2.7	7.0	**9.1**	**-4.3**	-12.7	-12.7	**5.1**	-12.6	-17.5	-15.2
Recip.	-0.5	11.9	24.6	23.3	0.1	**-5.2**	-15.0	-14.2	0.0	-13.0	-22.6	-19.4

3.2.3 Model fitting with different sampling proportions

Sampling with different proportions – none (1.0), 0.8, 0.4, 0.2, 0.1, 0.05, 0.025, and 0.01 – was tried using Epoch sampling with Harpoon sampling (**Table 3**). These proportions follow a roughly geometric progression, decreasing by approximately a factor of 2 at each step. This design was chosen to systematically explore the effects of increasingly aggressive sampling on

model performance. Harpoon sampling is used instead of random or weighted random sampling as it was previously shown to best yield the most balanced distributions (**Fig. 5**).

Compared to models fit with the standard method which does not use sampling, models fit with a sampling proportion of 0.4 or less performed significantly better on datasets with noise and greater right-skew (R3, R4), by up to 23.6%. Models fit with sampling proportions of 0.8 and 0.4 performed similarly to those fit with the standard method on datasets without noise. Conversely, in general, using a sampling proportion of less than 0.4 yielded models that performed significantly worse than the standard method on datasets without noise by up to 176.9%. On datasets with noise and right-skew, models fit with a sampling proportion of 0.1 performed the best on average.

Table 3. Models fit with sampling generally performed better than models fit without sampling on datasets with noise and right-skew.

Mean pairwise bootstrap scores comparing models fit without sampling ("None") or with different sampling proportions ("Samp. Prop.") ranging from 0.8 to 0.01 to models fit without sampling.

See **Sup. Table. 8** for additional bootstrap statistics.

See **Table 1** for additional descriptions.

Noise ID	**N1**				**N2**				**N3**			
Right ID	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
Samp. Prop.												
None	0.7	1.1	0.8	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.8	6.8	6.7	1.4	6.4	0.2	-2.1	-4.8	**-4.8**	-0.2	**-5.8**	**-5.3**	**-6.1**
0.4	**23.2**	24.5	19.2	19.4	0.7	**-4.8**	-12.8	-11.8	0.0	-12.8	-16.6	-15.8
0.2	**38.8**	**45.1**	**34.0**	25.8	2.0	**-4.5**	-15.9	-15.2	0.1	-13.2	**-21.5**	-18.1
0.1	**69.6**	**72.4**	**53.7**	**47.7**	**3.1**	**-3.4**	-15.8	-15.5	0.5	-12.9	**-23.6**	-20.1
0.05	**113.1**	**111.0**	**98.1**	**92.6**	**5.3**	-1.0	-13.5	-12.8	**1.2**	-12.3	**-22.9**	-19.7
0.025	**135.6**	**142.7**	**121.0**	**111.7**	**7.7**	0.8	-11.7	-12.5	**1.8**	-11.4	**-22.8**	-19.8
0.01	**167.9**	**176.9**	**162.9**	**138.8**	**12.1**	4.5	**-10.4**	**-10.6**	**3.1**	**-10.2**	**-22.5**	-19.8

3.2.4 Model fitting with different combinations of methods

The most effective methods from each category (RBR as the loss function, reciprocal weighting for output weighting, and sampling with a proportion of 0.1) were tried in combination towards further improving model performance (**Table 4**). The methods applied in combination were similarly effective to the methods applied individually but their effects were chiefly dependent on the sampling proportion. Generally, compared to models fit with the standard method (RMSE as the loss function, without output weighting, and without sampling), the models fit with the different combinations of methods performed significantly better on datasets with noise and right-skew by up to 23.6%. On datasets without noise, all combinations with sampling yielded models that were significantly worse than those that were fit with the standard method by up to 72.4%.

Table 4. Models fit with combinations of model fitting methods generally performed similarly on datasets with noise and right-skew.

Mean pairwise bootstrap scores comparing models fit with combinations of RMSE or RBR as the loss function, without output weighting ("None") or with reciprocal weighting (Recip.), and without sampling ("1.0") or with a sampling proportion of 0.1 to models fit with RMSE as the loss function, without output weighting, and without sampling. The combinations are denoted as (loss function, output weighting, sampling proportion).

See **Sup. Table. 9** for additional statistics.

See **Table 1** for additional descriptions.

Noise ID	N1				N2				N3			
Right ID	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4

Combination												
(RMSE, None, 1.0)	0.7	1.1	0.8	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
(RMSE, None, 0.1)	**69.6**	**72.4**	**53.7**	**47.7**	**3.1**	**-3.4**	**-15.8**	**-15.5**	0.5	**-12.9**	**-23.6**	**-20.1**
(RMSE, Recip., 1.0)	-0.5	11.9	24.6	23.3	0.1	**-5.2**	**-15.0**	**-14.2**	0.0	**-13.0**	**-22.6**	**-19.4**
(RMSE, Recip., 0.1)	**61.3**	**71.8**	**58.3**	**50.2**	**3.4**	**-3.0**	**-15.1**	**-14.1**	0.6	**-13.0**	**-23.3**	**-19.9**
(RBR, None, 1.0)	-1.5	12.5	27.7	20.4	0.3	**-5.0**	**-15.4**	**-15.2**	0.0	**-13.1**	**-23.0**	**-19.2**
(RBR, None, 0.1)	**67.5**	**69.8**	**53.2**	**44.9**	**3.4**	-3.1	**-15.0**	**-14.0**	0.6	**-12.9**	**-23.4**	**-20.0**
(RBR, Recip., 1.0)	0.6	15.8	26.1	14.2	0.2	**-4.9**	**-15.9**	**-15.2**	0.0	**-13.0**	**-23.0**	**-19.5**
(RBR, Recip., 0.1)	**64.3**	**66.0**	**55.5**	**47.8**	**3.2**	-3.4	**-14.9**	**-14.0**	0.5	**-12.8**	**-23.5**	**-20.1**

The methods with sampling may use less resources (computation and time) for NN model fitting than those without. Sampling uses fewer observations of the training set in each epoch thereby reducing the resources used per epoch. However, the effect of sampling to the total number of epochs needed for model fitting is not clear. Notably, each epoch involves evaluating the model by the stoppage set which is not sampled – it is the same regardless of sampling the training set. Accounting for the proportion of sampling of the training set and the unsampled stoppage set, the resources used by the different combinations of methods were compared to those used by the standard method (**Table 5**). Model fitting with combinations of methods that used a sampling proportion of 0.1 used significantly fewer resources than model fitting using the standard method for all datasets except for datasets with noise and right-skew R2, by up to 57.7%. Conversely, model fitting with combinations of methods that did not use sampling used similar (i.e. not statistically different) amounts of resources than model fitting with the standard method.

Table 5. Models fit with sampling used substantially less resources than those fit without sampling.

Mean *resource* pairwise bootstrap scores comparing models fit with combinations of RMSE or RBR as the loss function, without output weighting ("None") or reciprocal weighting (Recip.),

and without sampling ("1.0") or sampling with a proportion of 0.1 to models fit with RMSE as the loss function without output weighting and without sampling.

Resource pairwise bootstrap scores are the same as pairwise bootstrap scores except instead of using RBR of the average predictions of a bootstrap ensemble, resources are calculated as the sum of the epochs used by the bootstrap ensemble * (sampling prop. * training set prop. + stoppage set prop.) / (training set prop + stoppage set proportion). Negative values indicate the subject used less computational resources than the reference.

See **Sup. Table. 10** for additional bootstrap statistics.

See **Table 4** for additional descriptions.

	N1				**N2**				**N3**			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
Combination												
(RMSE, None, 1.0)	0.4	0.5	0.7	0.6	0.5	0.4	0.4	0.5	0.4	0.5	0.7	1.0
(RMSE, None, 0.1)	**-49.6**	**-43.4**	**-42.0**	**-44.2**	**-32.6**	-12.5	**-33.4**	**-52.3**	**-26.0**	-14.5	**-45.7**	**-39.4**
(RMSE, Recip., 1.0)	3.8	4.6	7.7	3.8	2.1	24.1	-8.8	-23.3	2.3	9.2	-23.7	-15.8
(RMSE, Recip., 0.1)	**-46.0**	**-45.3**	**-41.7**	**-45.1**	**-33.0**	-11.1	**-37.3**	**-54.7**	-24.9	-15.9	**-46.8**	**-43.5**
(RBR, None, 1.0)	2.7	2.8	0.9	1.4	-4.2	22.8	-4.8	-23.5	3.8	9.4	-20.1	-4.8
(RBR, None, 0.1)	**-47.9**	**-43.3**	**-40.0**	**-41.6**	**-32.1**	-11.0	**-37.8**	**-56.5**	**-25.9**	-14.1	**-47.4**	**-44.4**
(RBR, Recip., 1.0)	0.5	2.5	2.7	5.7	-2.6	**27.4**	-2.2	-18.2	2.6	10.3	-21.3	-7.6
(RBR, Recip., 0.1)	**-48.1**	**-42.8**	**-40.1**	**-45.9**	**-32.7**	-10.3	**-36.5**	**-57.7**	**-23.9**	-12.6	**-47.4**	**-42.0**

As explored, on datasets with noise and greater right-skew, methods with sampling yielded similarly performing models than methods without sampling while using significantly and substantially less resources. Most often, the configuration of hyperparameters of a model for effective modelling is not known prior, requiring a hyperparameter search. While the combinations of methods presented are recommended to be searched alongside the hyperparameters, the hyperparameter search space is often already too large to adequately search. For modelling Tailed datasets with noise, sampling can be used to fit models in hyperparameter search to save computing resources while presumably producing models that

679 perform similarly to those that are fit without sampling. This is beneficial for modelling tasks
680 that have large models, large datasets, and/or limited computing resources. The savings in
681 resources can be used towards more hyperparameter search (i.e. trying more hyperparameter
682 configurations), potentially finding hyperparameter configurations that may ultimately yield even
683 better performing models. Therefore, sampling is recommended in a hyperparameter search on
684 datasets with noise and greater levels of skew. It is still recommended to further optimize the
685 model fitting method after a hyperparameter search.

686 There are two specific caveats of this case study concerning the noise and the NN used.
687 The noise was injected additively (i.e. additive noise) but noise can be in different forms (e.g.
688 multiplicative noise). The methods tried may have different effects on different forms of noise.
689 Moreover, a single configuration of a NN was used. Although the same NN was used to generate
690 and model the data, the methods may behave differently to different NN configurations.
691 Likewise, a single configuration of optimizer and batch size was used for the experiments. The
692 methods tried may have different effects with different NN configurations including the
693 optimizer and batchsize. Together, these caveats are further impetus for experimentation with all
694 model fitting methods for each specific modelling task.
695

696 4 Case study: YY1-genome interactions

697 The interactions of proteins with DNA of the genome, referred to here as protein-genome
698 interactions (PGIs), are central to genomics as they are principal components of gene regulation.
699 The frequency and strength, collectively referred to here as the intensity, of an interaction of a
700 protein to a location in the genome can depend on the underlying DNA sequence as well as the
701 surrounding environment which includes histone post-translational modifications (histone PTMs)

702 (Guertin & Lis, 2010; Slattery et al., 2014; Wang et al., 2012). Successful modelling of PGIs
703 using DNA sequence and the surrounding environment may yield deeper biological insights that
704 are not possible by other methods.

705 Most proteins that interact with their respective genome only substantially interact with a
706 relatively few locations of the genome and interact with progressively greater intensity to
707 progressively fewer locations in the genome. Together, the intensities of PGIs are often severely
708 right-skewed. Early works discretize PGIs using an arbitrary threshold, but this procedure
709 discounts the differences in intensities which are biologically rooted and therefore should be
710 taken into account (Toneyan et al., 2022). While PGIs have been modelled before, data was
711 typically split by chromosome and the models were scored by non-bin metrics that are
712 ineffective for sequential and Tailed data, respectively. The following case study is an example
713 of modelling PGIs and assesses existing and novel methods for appropriate data splitting and
714 consistent model performance.

715 Yin Yang 1 (YY1) is a DNA sequence-specific transcription factor (a protein that affects
716 gene transcription) implicated in human diseases such as cancer and has a significant role in
717 development, specifically neurogenesis (Verheul et al., 2020). The PGIs of YY1 in human
718 cancer cells are modelled using the human genome DNA sequence and 11 histone PTMs.

719 The PGIs of YY1 are modelled using a NN called Oyster (Abdulnabi & Westwood,
720 2025b). There are two major variants of Oyster: Exact and non-Exact. Exact Oyster facilitates
721 model analysis whereas non-Exact may provide performance benefits. While the previous case
722 study used different datasets but a single model configuration, the following case study uses a
723 single dataset but compares the model fitting methods with configurations of Exact and non-

724 Exact Oysters. Model analysis of Exact Oysters modelling YY1-genome interactions is
725 conducted in the (Abdulnabi & Westwood, 2025b).
726

727 4.1 Methods

728 See https://github.com/Husam94/BinMeths/C2-YY1/ for complete details.
729

730 4.1.1 Data processing

731

732 4.1.1.1 Initial data processing

733 PGIs are captured through Chromatin Immunoprecipitation (ChIP) followed by DNA
734 sequencing. Single-end high-throughput sequencing alignment files of ChIP of YY1 and 11
735 histone PTMs in human cancer cell line K562 were obtained from the Encyclopedia of DNA
736 Elements (ENCODE), each having 2-3 repeats and a corresponding control with 3 repeats
737 (ENCODE Project Consortium, 2012) (**Sup. Table. 11**). MACS version 3 (MACS3) was used to
738 estimate the fragment length of the single-end reads for each alignment file (Zhang et al., 2008).
739 The alignments were converted to scores that map over the genome and the resolution was
740 reduced to 50 nucleotides (nts). Only scores in autosomal chromosomes (chr 1-22) were used.
741 Scores in blacklist regions (Amemiya et al., 2019) were zeroed. For each set of repeats, the
742 scores of the repeats were scaled relative to the average total scores and were then merged
743 through averaging.
744 The variation of a score is best explained by a Gamma distribution, explored in
745 (Abdulnabi & Westwood, 2025a). The scores of the ChIP repeats were scaled by the scores of
746 their corresponding control. The ratio of two Gamma distributions is a Beta Prime distribution.

747 The control-scaled scores, referred to as YY1 scores hereafter, are determined as the mode of
748 Beta Prime distribution with an α of the ChIP score and a β of the control score.
749 Windows of size 500 for genomic DNA sequence, 2000 for histone PTM signals, and 50
750 for YY1 signals were used for modelling. The output of each observation is a single value of
751 YY1 score over 50 nts. Only relatively few of the possible locations on the genome, referred to
752 here as sites, outside of the blacklist regions and in autosomal chromosomes can be used for
753 modelling due to computing budgets. Sites selected should have as balanced of a distribution of
754 YY1 values as possible but also be as diverse as possible for effective modelling.
755 ChIP experiments typically result in mound-shaped scores at locations where proteins
756 interact with the genome. MACS3 was used to determine the YY1 peaks (i.e. locations of the
757 genome of significant enrichment) with a q-value of 0.001. YY1 scores extend roughly ±1000
758 nts from the center of the peaks (**Sup. Fig. 1**). Towards diverse sites, the sites containing the
759 highest local values of YY1 score within ±1000 nts and having control scores greater than 0.5
760 corresponding to both YY1 and histone PTM ChIP were determined, totaling 113,540 sites.
761 YY1 scores belonging to the sites were binned by 10,000 histogram bins. Considering
762 computing budgets and towards a balanced distribution of YY1 scores, 20,000 sites were
763 sampled based on their histogram bins using Harpoon Sampling, referred to here as center sites.
764 Towards models that are robust to translational shifts, additional sites shifted ± 150 nts from each
765 of the center sites were also included, totaling 60,000 sites.
766
767 **4.1.1.2 Binning, split generation, and output weighting**
768 The YY1 scores are extremely right-skewed and very noisy (**Fig. 9**). YY1 scores for the
769 selected sites were binned by only using those belonging to center sites to ensure diverse sites in

later bins. They were binned by using the max-scaled YY1 score and using the tanh function and a scaling factor of 2.497 which was determined to produce the bins with the lowest variation with a minimum of 50 observations belong to each bin or a max range of 0.45x the total range, whichever was fulfilled first (**Sup. Table. 12**).

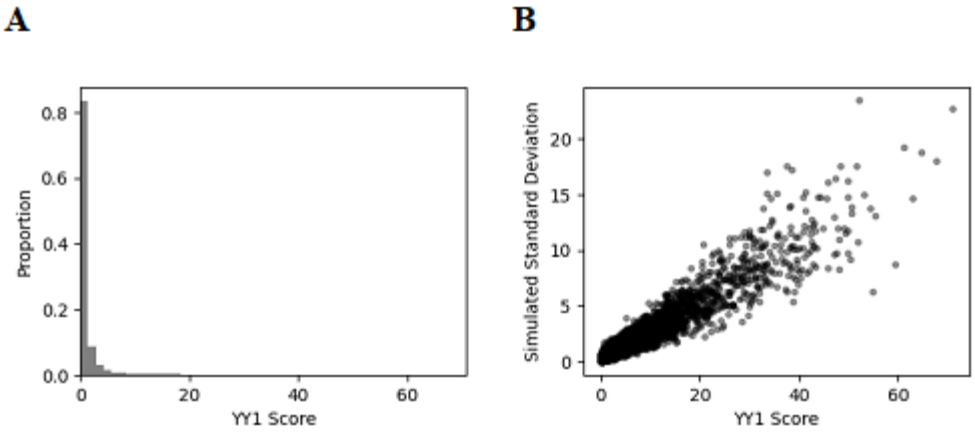


Fig. 9. YY1 scores of selected sites extremely right-skewed and noisy.

A) Histogram (50 bins) of YY1 scores selected for modelling.

B) Scatter plot of YY1 Score and *simulated standard deviation*. For each site, for each of the ChIP and control scores, a Gamma distribution with an α of the score was sampled 3 times and averaged, referred to as simulated scores, mimicking the experiment. For each site, the YY1 score was determined as the Beta Prime mode with α of the simulated ChIP score and a β of the simulated control score, referred to as the simulated YY1 score. This procedure was repeated 100 times, producing 100 simulated YY1 scores for each site. The standard deviation was determined for the set of simulated YY1 scores, referred to as the simulated standard deviation.

The split for modelling was generated using Akin split with Window split as the split generator. Proportions of 0.55, 0.15, 0.15, and 0.15 were used for the training, stoppage, evaluation, and testing sets (see **Supplementary Note 9.1.1**).

Output weights were determined as described in **Section 3.1.2.5**.

4.1.2 Model scoring and model fitting

Methods used for model scoring and model comparisons are the same as the ones used for the previous case study (see **Section 3.1.3**) except for two modifications. First, the Mean Deviation Error (MDE) is used instead of the RMSE as it better assesses predictions for measurements with measurement variation (Abdulnabi & Westwood, 2025a). More specifically, the Deviation Error accounts for each YY1 score to have measurement variation that is explained by a Beta Prime distribution. The RMS of the MDE of each bin is referred to as RBM. Second, there is only a single dataset with a single data split and the evaluation set was used to score and compare the experimental units instead of the testing set which is used to assess the final selected model in (Abdulnabi & Westwood, 2025b).

All modelling used a NN, RBM as the stoppage metric, a patience of 10, and the Adam optimizer. As with the previous case study, the choice of patience was determined based on trials with other patience values (not included here) and considering computing budgets. The *standard* model fitting method used MDE as the loss function without output weighting and without sampling.

4.1.3 Oyster models

Configurations of hyperparameters for the Exact Oyster and the non-Exact Oyster models, including the learning rate for the Adam optimizer and the batch size, were searched. More specifically, for each type of Oyster, 200 configurations were randomly selected from a search space that was based on computing budgets and other experiments not included here, see (Abdulnabi & Westwood, 2025b). All configurations used RBM as the loss function and sampling with a proportion of 0.1 as it is assumed to save resources without impairing performance for data with greater levels of noise and right-skew, as seen in the previous case study.

For each type of Oyster, the top 10 scoring configurations were repeated 10 times. The configuration with the lowest mean pairwise bootstrap scores (see **Section 3.1.3**) was used for further experiments (see (Abdulnabi & Westwood, 2025b) for model configurations).

4.2 Results and discussion

The total proportion of sites belonging to each chromosome and the distribution of values in the bins for each chromosome is not consistent (**Sup. Table. 13**). Even more concerning, some chromosomes do not have any sites belonging to one or more bins (i.e. the proportions of sites belonging to one or more bins is 0). It is not clear which chromosome(s) and/or which part(s) of the chromosome(s) should be used for each division of the data split to obtain the desired proportions and to have identical distributions. Thus, splitting by chromosome is not an adequate data splitting method. Akin split worked well to obtain a data split with the desired proportions and identical distributions (**Sup. Table. 14**). Hence, Akin split is a more appropriate and effective data splitting method than splitting by chromosome.

The objective of this case study is to minimize the RBM (see **Section 4.1.2** for details) and three different categories of model fitting methods of bin metrics as a loss function, output weighting, and sampling were first tried individually and then in combination.

The effectiveness of the model fitting methods applied individually varied and were dependent on the model. Compared to fitting with MDE (see **Section 4.1.2** for details) as the loss function, fitting with RBM as the loss function significantly improved the performance of the Exact model by 54.7% but did not significantly affect the performance of the non-Exact model (**Table 6**). Since the models were evaluated on the RBM, it was expected that using RBM as the loss function yielded models that better performed on the RBM than models fit with MDE as the loss function. Compared to fitting without sample weighting, fitting with reciprocal weighting significantly improved the performance of the Exact model by 56.4% but did not significantly affect the performance of the non-Exact model (**Table 7**). Fitting with DenseWeights did not significantly affect the performance of both models. Compared to fitting without sampling, sampling proportions of 0.8, 0.4, 0.2, 0.025, and 0.01 significantly improved the performance of the non-Exact model by up to 39.2% (**Table 8**). Intriguingly, sampling proportions of 0.1 and 0.05 did not significantly affect non-Exact model performance although sampling proportions of 0.2 and 0.025 had. Conversely, the model performance of the non-Exact model was significantly improved by a sampling proportion of 0.05 by 25.2%. Of the methods applied individually, the fitting with RBM as the loss function, reciprocal weighting for output weighting, and sampling proportions of 0.2 and 0.05 for Exact and non-Exact models, respectively, were the best model fitting methods applied individually.

Table 6. Fitting with RBM as the loss function yielded better performing Exact models but similar performing non-Exact models compared to fitting with the standard method.

Mean pairwise bootstrap scores comparing Oyster models that were fit with different loss functions to models fit with MDE as the loss function.

The "Shuffled MDE" refers to models fit with MDE as the loss function but with the outputs in the training and validation sets shuffled, serving as a negative control.

See **Sup. Table. 15** for full bootstrap statistics.

See **Table 1** for additional descriptions.

Oyster	Exact	non-Exact
Loss Function		
MDE	0.2	1.2
RBM	**-54.7**	-2.9
Shuffled MDE	**322.5**	**592.6**

Table 7. Fitting with reciprocal weighting yielded better performing Exact Oyster models but similar performing non-Exact models.

Mean pairwise bootstrap scores comparing Oyster models that were fit without output weighting ("None") or with output weighting methods of DenseWeights ("Dense") or reciprocal weighting ("Recip.") to models fit without output weighting.

See **Sup. Table. 16** for full bootstrap statistics.

See **Table 1** for additional descriptions.

Oyster	Exact	non-Exact
Weighting		
None	0.2	1.2
DenseW	**-20.7**	-12.4
Recip.	**-56.4**	11.4

Table 8. Fitting with sampling proportion of 0.2 and 0.05 best improved the performance of the Exact and non-Exact models, respectively.

Mean pairwise bootstrap scores comparing Oyster models that were fit without sampling ("None") or with different sampling proportions ("Samp. Prop.") ranging from 0.8 to 0.01 to models fit without sampling.

See **Sup. Table. 17** for additional bootstrap statistics.

See **Table 1** for additional descriptions.

Oyster	Exact	non-Exact
Samp. Prop.		
None	0.2	1.2
0.8	**-23.7**	-8.4
0.4	**-29.7**	3.6
0.2	**-39.2**	-19.1
0.1	-24.6	-24.4
0.05	14.1	**-25.2**
0.025	**-38.5**	-22.0
0.01	**-18.1**	-14.0

The most effective methods from each category were tried in combination towards further improving model performance (**Table 9**). Fitting with RBM as the loss function, with reciprocal weighting, and with sampling best improved the Exact model by 57.7%. Conversely, fitting with RBM as the loss function, without reciprocal weighting, and with sampling best improved the non-Exact model by 32.4%.

Table 9. Using combinations of methods did not further improve the performance of any model.

Mean pairwise bootstrap scores comparing Oyster models fit with combinations of RMSE or RBR as the loss function, without output weighting ("None") or with reciprocal weighting (Recip.), and without sampling ("False") or sampling proportions ("True") of 0.05, 0.025, or 0.05 for Exact, non-Exact A, and non-Exact B models, respectively, to models fit with RMSE as

the loss function, without output weighting, and without sampling. The combinations are denoted as (loss function, output weighting, sampling proportion).

See **Sup. Table. 18** for additional bootstrap statistics.

See **Table 1** for additional descriptions.

Oyster Combination	Exact	non-Exact
('MDE', 'None', False)	0.2	1.2
('MDE', 'None', True)	**-39.2**	**-25.2**
('MDE', 'Recip.', False)	**-56.4**	11.4
('MDE', 'Recip.', True)	**-56.2**	**-29.3**
('RBM', 'None', False)	**-54.7**	-2.9
('RBM', 'None', True)	**-55.8**	**-32.4**
('RBM', 'Recip.', False)	**-47.3**	**57.8**
('RBM', 'Recip.', True)	**-57.7**	**-28.8**

There were model fitting methods that significantly improved the performance of both models. Furthermore, the effects of the model fitting methods on model performance was dependent on the model configuration. The optimal model fitting methods for the model configurations were different. Together, these results further highlight the benefits and importance of experimenting and optimizing with the model fitting methods for each individual model configuration.

5 General discussion

There were model fitting methods that substantially and significantly improved the modelling performance compared to the standard method for both case studies. The effects of the methods on model performance were dependent on the data, specifically the amount of noise and skew, best exemplified in the first case study. In general, the methods were more effective on datasets with noise and skew. Model fitting methods with sampling impaired model performance for datasets without noise and datasets without skew. Still, the model fitting methods with sampling substantially reduced the resources used. For datasets with noise and greater levels of skew, sampling is recommended to be used for hyperparameter search to save resources or enable more search without (presumably) impairing performance. Furthermore, the effects of the methods were also dependent on the model configuration used, best exemplified in the second case study.

There is no general prescription of model fitting methods (or combinations thereof); different methods should be tried for each modelling task. Both case studies explored the methods through a stepwise procedure which first tried the methods individually and then tried the best methods in combination. There may be a combination(s) involving non-best methods that perform better than the combinations explored. An optimal combination of methods is better determined through Random search or an optimization algorithm. As the effects of the combination of methods are dependent on the model configuration used, it is ideal to search the combination of methods along with the model configurations.

There are two major caveats of the case studies presented concerning the metric and the distributions of the data used. First, both case studies assessed the models by the RBR. It is expected that the effects of these methods are dependent on the metric used. As such, the effects

of the methods are expected to be different when assessing the models with different metrics. Second, both case studies only used right-skewed data. Still, it is expected, both in theory and from experimentation not included here, that the methods are similarly effective to similar levels of other Tailed data (left-skewed or bell-shaped data). It is also expected that these methods are effective for imbalanced data in general, including nominal data.

All methods presented in this study including bin metrics, transformation binning, model fitting methods, and Akin split, should be used towards appropriate and effective modelling. Surely, these methods can be used to improve existing works. These methods can be especially beneficial for works that extract the learned concepts of the models to use them to guide future exploration which is common practice in genomics ML. Improved models may yield different and more accurate learned concepts, ultimately better guiding future explorations.

6 Acknowledgements

None.

7 Author contributions

Conceptualization: Husam Abdulnabi

Data curation: Husam Abdulnabi

Formal analysis: Husam Abdulnabi

Investigation: Husam Abdulnabi

Methodology: Husam Abdulnabi

Project administration: Husam Abdulnabi

Software: Husam Abdulnabi

Supervision: J. Timothy Westwood

Writing – original draft: Husam Abdulnabi

Writing – review & editing: J. Timothy Westwood

8 References

Abdulnabi, H., & Westwood, J. T. (2025a). Deviation Error: assessing predictions for replicate measurements in genomics and beyond. *Forthcoming in PLOS ONE*. https://doi.org/10.1101/2025.05.29.656931

Abdulnabi, H., & Westwood, J. T. (2025b). Oyster: a neural network for modelling genomics sequences that enables exact position-specific *k*-mer contributions . *To Be Published*.

Amemiya, H. M., Kundaje, A., & Boyle, A. P. (2019). The ENCODE Blacklist: Identification of Problematic Regions of the Genome. *Scientific Reports*, *9*(1), 9354. https://doi.org/10.1038/s41598-019-45839-z

Bach, M., Werner, A., & Palt, M. (2019). The Proposal of Undersampling Method for Learning from Imbalanced Datasets. *Procedia Computer Science*, *159*, 125–134. https://doi.org/10.1016/j.procs.2019.09.167

Beckmann, M., Ebecken, N. F. F., & Pires de Lima, B. S. L. (2015). A KNN Undersampling Approach for Data Balancing. *Journal of Intelligent Learning Systems and Applications*, *07*(04), 104–116. https://doi.org/10.4236/jilsa.2015.74010

Branco, P., Torgo, L., & Ribeiro, R. P. (2017). SMOGN: a pre-processing approach for imbalanced regression. . *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, 36–60.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Efron, B., & Tibshirani, R. (1986). Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statistical Science*, *1*(1), 54–57.

ENCODE Project Consortium. (2012). An integrated encyclopedia of DNA elements in the human genome. *Nature*, *489*(7414), 57–74. https://doi.org/10.1038/nature11247

Guertin, M. J., & Lis, J. T. (2010). Chromatin landscape dictates HSF binding to target DNA elements. *PLoS Genetics*, *6*(9), e1001114. https://doi.org/10.1371/journal.pgen.1001114

John, J. A., & Draper, N. R. (1980). An Alternative Family of Transformations. *Applied Statistics*, *29*(2), 190. https://doi.org/10.2307/2986305

Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization*.

Slattery, M., Zhou, T., Yang, L., Dantas Machado, A. C., Gordân, R., & Rohs, R. (2014). Absence of a simple code: how transcription factors read the genome. *Trends in Biochemical Sciences*, *39*(9), 381–399. https://doi.org/10.1016/j.tibs.2014.07.002

Steininger, M., Kobs, K., Davidson, P., Krause, A., & Hotho, A. (2021). Density-based weighting for imbalanced regression. *Machine Learning*, *110*(8), 2187–2211. https://doi.org/10.1007/s10994-021-06023-5

Toneyan, S., Tang, Z., & Koo, P. K. (2022). Evaluating deep learning for predicting epigenomic profiles. *Nature Machine Intelligence*, *4*(12), 1088–1100. https://doi.org/10.1038/s42256-022-00570-9

Torgo, L., Ribeiro, R. P., Pfahringer, B., & Branco, P. (2013). *SMOTE for Regression* (pp. 378–389). https://doi.org/10.1007/978-3-642-40669-0_33

Verheul, T. C. J., van Hijfte, L., Perenthaler, E., & Barakat, T. S. (2020). The Why of YY1: Mechanisms of Transcriptional Regulation by Yin Yang 1. *Frontiers in Cell and Developmental Biology*, *8*, 592164. https://doi.org/10.3389/fcell.2020.592164

Wang, J., Zhuang, J., Iyer, S., Lin, X., Whitfield, T. W., Greven, M. C., Pierce, B. G., Dong, X., Kundaje, A., Cheng, Y., Rando, O. J., Birney, E., Myers, R. M., Noble, W. S., Snyder, M., & Weng, Z. (2012). Sequence features and chromatin structure around the genomic regions bound by 119 human transcription factors. *Genome Research*, *22*(9), 1798–1812. https://doi.org/10.1101/gr.139105.112

Zhang, Y., Liu, T., Meyer, C. A., Eeckhoute, J., Johnson, D. S., Bernstein, B. E., Nusbaum, C., Myers, R. M., Brown, M., Li, W., & Liu, X. S. (2008). Model-based Analysis of ChIP-Seq (MACS). *Genome Biology*, *9*(9), R137. https://doi.org/10.1186/gb-2008-9-9-r137

9 Supplementary

9.1 Notes

9.1.1 Additional sets for ML

Neural Network (NNs) are fit in cycles called epochs wherein the NN is exposed to each observation of the training set once. The number of epochs can be specified prior to modelling but the optimal number of epochs is not known prior. Models fit with too few epochs are often underfit, but models fit with too many epochs are often overfit to the training set and do not generalize well to unseen data (e.g. the testing set). An additional set can be used to assess the

model at every epoch for its ability to generalize, called the stoppage set (commonly known as a validation set but renamed for clarity). The performance on the stoppage set deteriorates when the model becomes overfit, indicating that model fitting should be terminated. While the model is not exposed to this data directly, it dictates model fitting and thus is separate to the testing set.

There are several tunable parameters concerned with the ML algorithm and/or the model fitting process, known as hyperparameters. Different configurations of hyperparameters should be tried towards obtaining the best model possible. The testing set needs to be completely unused in this process. An additional set is used to evaluate the model during hyperparameter search, called an evaluation set.

9.2 Equations

Sup. Eq. 1. tanh function.

$$\tanh(x) = \frac{2}{1 + e^{-\alpha x}} - 1$$

Sup. Eq. 2. Inverse tanh function.

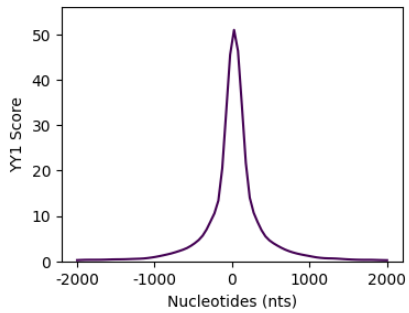
$$\tanh^{-1}(x) = -\frac{\log\left(\left(\frac{2}{(x+1)}\right) - 1\right)}{2}$$

Sup. Eq. 3. Reciprocal weighting for each bin.

$$p = \text{proportion of values belonging to a bin} = \frac{\text{number of values belonging to a bin}}{\text{total number of values}}$$

$$\text{Reciprocal Weighting}\ (p) = \frac{1}{p}, p > 0$$

9.3 Figures



Sup. Fig. 1. YY1 score over YY1 peaks ± 2000 nts from the center.

Peaks were filtered to be outside of Blacklist regions and within autosomal chromosomes.

9.4 Tables

All tables in **BinMeths.Sup.Tables.xlsx**.

9.4.1 Synthetic Tailed data case study

Sup. Table. 1. Tanh Binning and splits for synthetic repeat 1, noise level NA, and right-skew level RA. "Prop." and "Stop." refer to proportion and stoppage, respectively. Select columns are white-grey-black color scaled to visualize low to high values for each respective column.

Sup. Table. 2. Tanh Binning and splits for synthetic repeat 1, noise level NA, and right-skew level RB. Refer to **Sup. Table. 1** for full explanation.

Sup. Table. 3. Tanh Binning and splits for synthetic repeat 1, noise level NA, and right-skew level RC. Refer to **Sup. Table. 1** for full explanation.

Sup. Table. 4. Tanh Binning and splits for synthetic repeat 1, noise level NA, and right-skew level RD. Refer to **Sup. Table. 1** for full explanation.

Sup. Table. 5. Mean pairwise bootstrap scores comparing models fit with Adam learning rates from 0.0003 to 0.01 to models fit with an Adam learning rate of 0.0001.
Bolded values indicate the subject is significantly better or worse than the reference if negative or positive, respectively.
Dark to light color scale visualize values from -25 to 0.

Sup. Table. 6. Statistics of pairwise bootstrap scores comparing models fit with different loss functions to models fit with RMSE as the loss function.
"SE", "Conf. Int.", and "Sig." refer to standard error, confidence interval, and significance, respectively. Bolded rows indicate rows that are significant. Dark to light color scale visualize mean from -25 to 0.

Sup. Table. 7. Statistics of pairwise bootstrap scores comparing models fit without output weighting ("None") or with output weighting methods of DenseWeights ("Dense") or reciprocal weighting ("Recip.") to models fit without output weighting.

See **Sup. Table. 6** for additional descriptions.

Sup. Table. 8. Statistics of pairwise bootstrap scores comparing models fit without sampling (“None”) or with different sampling proportions ranging from 0.8 to 0.01 to models fit without sampling.

See **Sup. Table. 6** for additional descriptions.

Sup. Table. 9. Statistics of pairwise bootstrap scores comparing models fit with combinations of RMSE or RBR as the loss function, without output weighting (“None”) or with reciprocal weighting (Recip.), and without sampling (“None”) or with a sampling proportion of 0.1 to models fit with RMSE as the loss function, without output weighting, and without sampling.

See **Sup. Table. 6** for additional descriptions.

Sup. Table. 10. Statistics of resource pairwise bootstrap scores comparing models fit with combinations of RMSE or RBR as the loss function, without output weighting (“None”) or reciprocal weighting (Recip.), and without sampling (“None”) or sampling with a proportion of 0.1 to models fit with RBR as the loss function without output weighting and without sampling. The combinations are denoted as (loss function, output weighting, sampling proportion).

See **Table 5** for description of resource pairwise bootstrap scores.

See **Sup. Table. 6** for additional descriptions.

9.4.2 YY1-genome interactions case study

Sup. Table. 11. Files obtained from ENCODE. "predictd" column is the predicted fragment length determined using MACS to extend single-end reads.

Sup. Table. 12. Tanh Binning of YY1 values. Refer to **Sup. Table. 1** for full explanation.

Sup. Table. 13. Chromosomes differ in the number and distributions of observations belonging to them.

Proportions of observations of each chromosome belonging to each bin. The Total column indicates the total proportion of observations belonging to the chromosome. Light-dark color scale visualize low to high values for each respective column. Red shading indicates a 0 value.

Sup. Table. 14. Akin Split produces a split with the desired proportion and similar distributions.

Proportions of observations of each set belonging to each bin.

See **Sup. Table. 13** for additional descriptions.

Sup. Table. 15. Statistics of pairwise bootstrap scores comparing Oyster models fit with different loss functions to models fit with MDE as the loss function.

See **Sup. Table. 6** for additional descriptions.

Sup. Table. 16. Statistics of pairwise bootstrap scores comparing Oyster models fit without output weighting (“None”) or with output weighting methods of DenseWeights (“Dense”) or reciprocal weighting (“Recip.”) to models fit without output weighting. See **Sup. Table. 6** for additional descriptions.

Sup. Table. 17. Statistics of pairwise bootstrap scores comparing Oyster models fit without sampling (“None”) or with different sampling proportions ranging from 0.8 to 0.01 to models fit without sampling.
See **Sup. Table. 6** for additional descriptions.

Sup. Table. 18. Statistics of pairwise bootstrap scores comparing Oyster models fit with combinations of MDE or RBM as the loss function, without output weighting (“None”) or with reciprocal weighting (Recip.), and without sampling (“FALSE”) or sampling proportions (“True”) of 0.2 or 0.05 for Exact and non-Exact models, respectively, to models fit with MDE as the loss function, without output weighting, and without sampling. The combinations are denoted as (loss function, output weighting, sampling proportion).
See **Sup. Table. 6** for additional descriptions.