Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **505,** 4847–4856 (2021) https://doi.org/10.1093/mnras/stab1513
Advance Access publication 2021 May 29

Benchmarking and scalability of machine-learning methods for photometric redshift estimation

Ben Henghes,[1★] Connor Pettitt,[2] Jeyan Thiyagalingam,[2★] Tony Hey[2] and Ofer Lahav[1]
[1]Department of Physics & Astronomy, University College London, Gower Street, London WC1E 6BT, UK
[2]Scientific Computing Department, Rutherford Appleton Laboratory, Science and Technology Facilities Council (STFC), Harwell Campus, Didcot OX11 0QX, UK

Accepted 2021 May 14. Received 2021 April 29; in original form 2021 April 5

ABSTRACT

Obtaining accurate photometric redshift (photo-z) estimations is an important aspect of cosmology, remaining a prerequisite of many analyses. In creating novel methods to produce photo-z estimations, there has been a shift towards using machine-learning techniques. However, there has not been as much of a focus on how well different machine-learning methods scale or perform with the ever-increasing amounts of data being produced. Here, we introduce a benchmark designed to analyse the performance and scalability of different supervised machine-learning methods for photo-z estimation. Making use of the Sloan Digital Sky Survey (SDSS – DR12) data set, we analysed a variety of the most used machine-learning algorithms. By scaling the number of galaxies used to train and test the algorithms up to one million, we obtained several metrics demonstrating the algorithms' performance and scalability for this task. Furthermore, by introducing a new optimization method, time-considered optimization, we were able to demonstrate how a small concession of error can allow for a great improvement in efficiency. From the algorithms tested, we found that the Random Forest performed best with a mean squared error, MSE = 0.0042; however, as other algorithms such as Boosted Decision Trees and k-Nearest Neighbours performed very similarly, we used our benchmarks to demonstrate how different algorithms could be superior in different scenarios. We believe that benchmarks like this will become essential with upcoming surveys, such as the Vera C. Rubin Observatory's Legacy Survey of Space and Time (LSST), which will capture billions of galaxies requiring photometric redshifts.

Key words: methods: data analysis – galaxies: distances and redshifts – cosmology: observations.

1 INTRODUCTION

Calculating distances to cosmological objects remains one of the most important steps required for probing cosmology. These distances are given by the distance–redshift relation, and hence one needs very accurate measures of redshift to be confident in the inferred distances. Ideally, high-resolution spectra would be obtained for every object enabling for a precise measurement of the redshift. However, with current and future surveys such as the Dark Energy Survey (DES) (DES Collaboration 2005, 2016), Euclid (Amendola et al. 2018), and the Vera C. Rubin Observatory's Legacy Survey of Space and Time (Tyson et al. 2003; Ivezić et al. 2019), even with large spectroscopic surveys such as the Dark Energy Spectroscopic Instrument (Flaugher & Bebek 2014; Martini et al. 2018), only tens of millions of the galaxies will have spectroscopy performed, despite hundreds of millions of galaxies being observed.

In the absence of real spectroscopic measurements, obtaining photometric redshifts (photo-z) estimations is the only viable route available for scientists. There are two major techniques used for photometric redshift estimation, template flitting (e.g. Benitez 2000), and machine learning (ML) (e.g. Collister & Lahav 2004). Both methods rely on the photometric information produced by the survey, usually given as magnitudes in different colour bands. These magnitudes act as approximate measures of the underlying spectral energy distribution (SED) of the observed object, and by appropriately reconstructing the SED, a corresponding redshift can be inferred (Bolzonella, Miralles & Pelló 2000).

Template-fitting methods use a small and fixed set of template spectra for the estimations and inherently rely on the assumption that the best-fitting SED template provides the true representation of the observed SED. There are benefits of template methods, such as, the ability to incorporate physical information, like dust extinction, into the model. However, embedding such physical constraints requires very precise calibration and an accurate model (Benitez 2000).

ML techniques, on the other hand, do not have any explicit model for capturing the physical information of the objects or of the estimation process. Instead, ML techniques rely on a training data set with spectroscopic redshifts from observed or simulated (or a combination of both) data for inferring an estimation model. More specifically, supervised learning models rely on a guided principle that with sufficient examples of input–output pairs an estimation model can be inferred by understanding the latent variables of the process. In other words, ML methods derive a suitable functional mapping between the photometric observations and the corresponding redshifts.

★ E-mail: ben.henghes.13@ucl.ac.uk(BH); t.jeyan@stfc.ac.uk(JT)

The learning process relies on a labelled data set consisting of a set of magnitudes in each wavelength band (the inputs) and corresponding true values of the spectroscopic redshifts (the output labels or ground truth). The learning model, such as a Random Forest or Neural Network, learns the mappings that can be non-linear. It has been shown that the functional mapping learned through the supervised learning can for some science goals outperform the template-based methods (Abdalla et al. 2011).

Although the usage of ML in handling this problem has become very common (Hoyle 2016; D'Isanto & Polsterer 2018; Pasquet et al. 2019), and some studies are beginning to investigate the efficiency of different models (Euclid Collaboration 2020; Schmidt et al. 2020), there is still no comprehensive study outlining the benchmarking process and how it changes our overall understanding of how different ML methods handle the photo-z problem. In fact, this is a common problem across all domains of sciences, and as such, the notion of artificial intelligence (AI) benchmarking is an upcoming challenge for the AI and scientific community. This is particularly true in light of recent developments in the ML and AI domains, such as the deep learning revolution (Sejnowski 2018), technological development on surveys (Dewdney et al. 2009), the ability to generate or simulate synthetic data (Springel 2005), and finally the progress in computer architecture space, such as the emergence of GPUs (Kirk 2007).

The notion of benchmarking (Dongarra, Luszczek & Petitet 2003) has conventionally been about how a given architecture (or an aspect of a given architecture) performs for a given problem, such as the LINPACK challenge (Dongarra et al. 1979). However, in our case, the focus is broader than just performance. Our motivation here is many-fold, including understanding how different ML models compare when estimating the redshifts, how these techniques perform when the available training data are scaled, and finally how these techniques scale for inference. Furthermore, one of the key challenges here is the identification of appropriate metrics or figures of merit for comparing these models across different cases.

We intend to answer some of these questions in this paper by introducing this as a representative AI benchmarking problem from the astronomical community. The benchmarks will include several baseline reference implementations covering different ML models and address the challenges outlined above. The rest of this paper is organized as follows: In Section 2, we describe the data set used and include discussions on the features selected. In Section 3, we briefly describe the ML models that were evaluated in the study, followed by the descriptions of the optimization and benchmarking processes and the different metrics that are part of our analyses. The results are then presented in Section 4 along with our observations, and we conclude the paper in Section 5 with directions for further work.

2 DATA

The data used in our analysis come entirely from the Sloan Digital Sky Survey (SDSS) (York et al. 2000). Using its dedicated 2.5-m telescope at Apache Point Observatory (Gunn et al. 2006), SDSS is one of the largest public surveys with over 200 million photometric galaxies and three million useful galaxy spectra as of data release 12 (DR12) (Alam et al. 2015).

In this work, we downloaded 1 639 348 of these galaxies with spectroscopic data available to be used by the ML algorithms. The spectroscopic redshift was required as it was taken to be the ground truth for the redshift that the algorithms were trying to predict using the magnitudes of each galaxy. SDSS took images using five different optical filters (u, g, r, i, and z), and as a result of these different wavelength bands, there were five magnitudes for each observed galaxy (Eisenstein et al. 2011).

The 1.6-million galaxies used in this investigation were from a cleaned data set where it was a requirement for all five magnitudes to have been measured. In many cases for observations of galaxies, there could be a missing value in one of the filters, which would negatively impact its redshift estimation. By only using galaxies with complete photometry, we ensured that our comparison of methods was not also being affected by the kind of galaxies within the different-sized data sets.

Furthermore, the redshift range of the galaxies used was constrained to have only galaxies with a redshift, $z < 1$. While this greatly simplified the task of obtaining photo-z estimations and meant that the specific models tested would not be useful outside of this range, there are far fewer galaxies with measured spectroscopic redshifts greater than 1, and we kept within this range to ensure that the training set would be representative and allow for reliable estimates to be generated inside of this range. This meant that the benchmarking performed could be carried out without also having to take into account the effects that an unclean data set might have had on the different ML algorithms.

The main features of the data used by ML algorithms were the five magnitudes that could also be combined to give the four colours that are simply the difference in magnitudes between neighbouring wavelength bands (u–g, g–r, r–i, and i–z). There were additional feature columns contained in the SDSS data that could have been added such as the subclass of galaxy or the Petrosian radius (Petrosian 1976; Soo et al. 2017). However; adding these additional features would not have had a large impact on the results and could have added more issues due to incompleteness if the feature was not recorded for every galaxy. Instead, it was decided to use only the information from the five magnitudes as features that we knew to be complete.

Finally, we also scaled the features by subtracting their mean and dividing by their standard deviation to give unit variance. This ensured that the ML algorithms used were not being influenced by the absolute size of the values, where a difference in a feature's variance could result in it being seen as more important than other features. And by randomly splitting the full data set to form the training and testing sets, the subsets created kept the same distribution of redshift and were representative of the overall data frame.

3 METHODOLOGY

With the data prepared, the first step of the ML process was to split the entire data set to create a training set, testing set, and validation set, whereby the test and validation sets were kept unseen by the ML algorithms until after they had been trained using the training data. As part of the benchmarking process, the ML algorithms (described in Section 3.1) were trained and tested on many different sizes of data sets, and to do this, the data were split randomly for each size of training and testing set required.

During training, the algorithms were also optimized by changing the hyperparameters. These are the parameters of the models that control how the ML algorithms create their mappings from the features to the redshift. The most complete way of optimizing would be to perform brute force optimization where every combination of a defined grid of hyperparameters would be tested. However, this is far more computationally intensive than random optimization that instead tests a random subset of the hyperparameter grid and provides a good estimate of the best hyperparameters. The grids of hyperparameters tested for each algorithm are given in Table 1 along

Table 1. Grids of hyperparameters that were searched to test and compare each machine learning algorithm, along with the hyperparameters that were selected by the random optimization. The arrays of hyperparameters were chosen to give a good overview of different possible configurations of the algorithms, and by changing the parameters that had the greatest impact on the algorithms, we ensured finding a good representation of the 'best' performing algorithms.

Classifier	Hyperparameter	Array of values searched	Selected value
LR	'fit intercept'	[True, False]	**True**
	'normalize'	[True, False]	**True**
kNN	'no. neighbors'	[1, 200]	**21**
	'weights'	['uniform', 'distance']	**'distance'**
	'leaf size'	[10, 100]	**27**
	'p'	[1, 4]	**2**
DT	'max. features'	[1, 5, 'auto']	**'auto'**
	'min. samples split'	[2, 100]	**38**
	'min. samples leaf'	[1, 100]	**64**
	'min. weight fraction leaf'	[0, 0.4]	**0**
	'criterion'	['mse', 'mae']	**mse**
BDT	'no. estimators'	[1, 200]	**88**
	'loss'	['ls', 'lad', 'huber', 'quantile']	**'lad'**
	'max. features'	[1, 5]	**4**
	'max. depth'	[1, 20]	**17**
	'min. samples split'	[2, 100]	**46**
	'min weight fraction leaf'	[0, 0.4]	**0**
RF	'no. estimators'	[1, 200]	**94**
	'max. features'	[1, 5]	**4**
	'min. samples leaf'	[1, 100]	**8**
	'min. samples split'	[2, 100]	**13**
	'min weight fraction leaf'	[0, 0.4]	**0**
	'criterion'	['mse', 'mae']	**mae**
ERT	'no. estimators'	[1, 200]	**147**
	'max. features'	[1, 5]	**4**
	'min. samples leaf'	[1, 100]	**3**
	'min. samples split'	[2, 100]	**87**
	'min weight fraction leaf'	[0, 0.4]	**0**
	'criterion'	['mse', 'mae']	**mse**
MLP	'hidden layer sizes'	[(100, 100, 100), (100, 100), 100]	**(100, 100, 100)**
	'activation'	['tanh', 'relu']	**'tanh'**
	'solver'	['sgd', 'adam']	**'adam'**
	'alpha'	[0.00001, 0.0001, 0.001, 0.01]	**0.01**
	'tol'	[0.00001, 0.0001, 0.001, 0.01]	**0.00001**
	'learning rate'	['constant','adaptive']	**'constant'**

with the selected parameters.

To be able to optimize the algorithms, the decision first had to be made of which metric would be optimized for. There are three main metrics used for regression problems such as this: mean squared error (MSE), mean absolute error (MAE), and R-squared score (R^2). The formulae for calculating each of these metrics are given below where for the i-th sample within a total of n samples, $\hat{z}_i$ is the predicted value, and z_i is the true value.

$$\mathrm{MSE}(z, \hat{z}) = \frac{1}{n_{\mathrm{samples}}} \sum_{i=0}^{n_{\mathrm{samples}}-1} (z_i - \hat{z}_i)^2 \tag{1}$$

$$\mathrm{MAE}(z, \hat{z}) = \frac{1}{n_{\mathrm{samples}}} \sum_{i=0}^{n_{\mathrm{samples}}-1} |z_i - \hat{z}_i| \tag{2}$$

$$R^2(z, \hat{z}) = 1 - \frac{\sum_{i=1}^{n}(z_i - \hat{z}_i)^2}{\sum_{i=1}^{n}(z_i - \bar{z})^2}. \tag{3}$$

There are three additional metrics defined below that are commonly used to determine the performance of photometric redshift estimations: bias (the average separation between prediction and true value), precision [also 1.48 × median absolute deviation, which gives the expected scatter as given by Ilbert et al. (2006)], and outlier fraction (the fraction of predictions where the error is greater than a set threshold, here chosen to be >0.10). Each of these metrics was also calculated and the results are given in Table 2.

$$\mathrm{Bias} = <z_{\mathrm{pred}} - z_{\mathrm{spec}}> \tag{4}$$

$$\mathrm{Precision} = 1.48 \times \mathrm{median}\left(\frac{|z_{\mathrm{pred}} - z_{\mathrm{spec}}|}{1 + z_{\mathrm{spec}}}\right) \tag{5}$$

$$\mathrm{Outlier\ fraction} = \frac{N(\Delta z) > 0.10}{N_{\mathrm{total}}}. \tag{6}$$

As well as deciding which metric to optimize for, we introduced an extra stage included in the optimization that allowed for a time-considered optimization (see Section 3.2). We optimized the ML algorithms for MSE (aiming to minimize the MSE) and used a random optimization with 1000 iterations to ensure a good estimate of the best hyperparameters for each algorithm. Furthermore, we used a threefold cross validation (Breiman & Spector 1992) to ensure

Table 2. Results of testing the seven machine-learning algorithms described in Section 3.1. Each algorithm was trained using 10 000 galaxies and tested using fivefold cross validation to obtain the quoted standard deviation.

	Linear regression (LR)	k-Nearest neighbours (kNN)	Decision tree (DT)	Boosted decision tree (BDT)	Random forest (RF)	Extremely randomized trees (ERT)	Multilayer perceptron (MLP)
MSE	0.005714 ± 0.000577	0.004438 ± 0.000417	0.004631 ± 0.000407	0.004277 ± 0.000394	0.004221 ± 0.000423	0.004327 ± 0.000419	0.004701 ± 0.000499
MAE	0.050931 ± 0.001679	0.040881 ± 0.001626	0.041827 ± 0.001452	0.038757 ± 0.001514	0.038504 ± 0.001484	0.040459 ± 0.001537	0.051260 ± 0.008874
R^2	0.865198 ± 0.009009	0.895208 ± 0.007215	0.890677 ± 0.006415	0.899017 ± 0.006822	0.900366 ± 0.007373	0.897861 ± 0.007198	0.871507 ± 0.014329
Bias	0.039742 ± 0.000920	0.031109 ± 0.000977	0.032030 ± 0.000895	0.029428 ± 0.000893	0.029334 ± 0.000854	0.030927 ± 0.000947	0.034577 ± 0.002209
Precision	0.043421 ± 0.000578	0.031836 ± 0.000845	0.032895 ± 0.001137	0.028986 ± 0.000264	0.029279 ± 0.000766	0.031945 ± 0.000609	0.040837 ± 0.003310
Outlier Fraction	0.060500 ± 0.005187	0.034800 ± 0.007033	0.033400 ± 0.007439	0.029300 ± 0.005183	0.029700 ± 0.004389	0.033400 ± 0.006304	0.037600 ± 0.002709

that the algorithms were not overfitting (which could mean that the algorithms were able to perform well for the training data used but then failed to generalize), and that the results would be valid for any given data set. Once optimized, each algorithm was then retrained and tested to give the final results given in Section 4, along with the benchmarking results, where the benchmarking process used is described in Section 3.3.

3.1 Descriptions of machine-learning algorithms tested

The following algorithms were selected for testing as they are some of the most widely used ML algorithms and all available through the PYTHON package Scikit–Learn (Pedregosa et al. 2011). While a simple neural network (Multilayer Perceptron) was included, we did not include any other examples of deep learning. This decision was made as deep learning algorithms perform best with many features (often thousands), and there is only so much information that the photometry could provide with the five magnitude features. Furthermore, it has been shown by Hoyle (2016) that 'traditional' algorithms can perform equally well as deep learning methods, and that it might be beneficial to use more computationally expensive deep learning models only when directly using images as the training data (Pasquet et al. 2019).

3.1.1 Linear regression

Linear regression, or ordinary least squares regression, fits a linear model to the data with coefficients that act to minimize the sum of the squared residuals between the observations and the predictions from the linear approximation. The linear model requires independent features, as features that are correlated will give estimates that are very sensitive to random errors in the observations, resulting in a large variance (Hastie, Tibshirani & Friedman 2009).

3.1.2 k-Nearest Neighbours

k-Nearest Neighbours uses a predefined number of data points in the training sample, *k*, which are closest in Euclidean distance to the new point whose value is then predicted based off those. This is an example of an instance-based algorithm, where there is no general model used to make predictions but which stores the training data. Although one of the simplest methods, being non-parametric can make it very successful especially in cases with an irregular decision boundary. Increasing the value of *k* acts to reduce the effects of noise; however, it also makes the decision boundary less distinct and could result in overfitting (Altman 1992).

3.1.3 Decision Trees

Decision Trees (Breiman et al. 1983) are non-parametric algorithms whereby the data features are used to learn simple decision rules. The decision rules are basic if-then-else statements and are used to split the data into branches. The tree is then trained by recursively selecting the best feature split, which is taken to be the split that gives the highest information gain, or greatest discrepancy between the two classes. Typically, Decision Trees can produce results with a high accuracy; however, they are bad at generalizing the data as they are often complex and overfitted. Instead, they can be combined in an ensemble such as Boosted Decision Trees, Random Forests, or Extremely Randomized Trees.

3.1.4 *Boosted Decision Trees*

Boosted Decision Trees were the first ensemble method we considered. The process of boosting can be described whereby the ML algorithm is repeatedly fitted to the same data set, but each time the weights of objects with higher errors are increased. This aims to result in an algorithm that can better handle the less common cases than a standard decision tree. The boosting can be generalized by using an arbitrary differentiable loss function, which is then optimized (Friedman 2002), and we found that for this problem, using the least absolute deviation loss function produced the best results.

3.1.5 *Random Forests*

Random Forests also take many decision trees to build an ensemble method, averaging the predictions of the individual trees to result in a model with lower variance than in the case of a single decision tree. This is done by adding two elements of randomness, the first of which is using a random subset of the training data that is sampled with replacement (Breiman 1996). Secondly, the feature splits are found from a random subset of the features rather than using the split that results in the greatest information gain. This randomness can yield decision trees with higher errors; however, by averaging the predictions, the errors cancel out and the variance reduction yields a greatly improved model as well as removing the typical overfitting that occurs with single decision trees (Breiman 2001).

3.1.6 *Extremely Randomized Trees*

Extremely Randomized Trees (Geurts, Ernst & Wehenkel 2006) is an algorithm very similar to Random Forests but with an additional step to increase randomness. The feature splits are not only found from a random subset of the features. It also uses thresholds that are picked at random for each feature before the best of these random thresholds are then used for the decision rules, instead of simply using the thresholds that result in the greatest information gain. This acts to further reduce the variance compared to a Random Forest; however, it also results in a slightly greater bias.

3.1.7 *Multilayer Perceptron*

The Multilayer Perceptron is an example of a fully connected neural network with at least three layers of nodes. It consists of the input node, output node, and a minimum of one hidden layer, although more can be added, which makes the MLP the most simple instance of deep learning. In the way that the perceptron learns how to map the input node to the target vector, it is similar to logistic regression; however, it differs with the addition of one or more non-linear hidden layers that allow it to approximate any continuous function (Werbos 1988; LeCun et al. 2012).

3.2 Time-considered optimization

In the normal process of optimizing ML algorithms, a single metric is chosen to minimize. If brute force optimization is used, this produces an algorithm configured with the hyperparameters from the defined grid that gives the best result for the metric (e.g. the lowest MSE). Although this algorithm by definition would have the best result, it does not necessarily result in the most useful or suitable algorithm. The hyperparameters selected to minimize the error likely also act to increase the computational time required both in training and inference, resulting in a much slower model.

Rather than minimizing a single metric, in time-considered optimization, we also consider the time taken by the models both in training and inference. By setting an error tolerance, we allow for the model selection to suggest an alternative to the 'best' model (the model that minimizes the error metric), instead providing a model that will have a higher error, while kept below the tolerance, but in return will also have faster training and inference times. This was done by optimizing for a combination of MSE and time, taking the model that ran the fastest and that had an error below the set tolerance level. In certain cases, such as training the Decision Tree, it was possible to achieve a two magnitude increase in efficiency while increasing the error by <10 per cent.

For the purpose of benchmarking the ML algorithms in this paper, we set the error tolerance to machine precision (which is usually 10^{-16}) resulting in the 'best' model in terms of error. This decision was made as these optimized algorithms would result in the algorithms most commonly used in other ML studies where time-considered optimization has not been implemented.

3.3 Benchmarking

The benchmarking performed was achieved by recording the system state (described by the time, CPU usage, memory usage, and disk I/O) throughout the process of running the ML algorithms. This allowed us to compare the efficiency of both training and predicting performance of the ML models and when combined with the regression errors obtained, allowed for a complete description of the performance of the different methods.

Our main focus of the benchmark was to investigate how training and testing times varied with different sizes of data frames, and how the final redshift estimations would be affected. As such, we incrementally changed both the training and testing data sets and recorded the times taken, which allowed us to produce the plots shown in Figs 1–5.

4 RESULTS

The results given in Table 2 show how the seven ML algorithms performed at producing photometric redshift estimations. Furthermore, Fig. 6 displays the true spectroscopic redshifts plotted against the photometric redshift estimates for each ML algorithm. We also plotted the distributions of the redshift estimations for each of the algorithms as well as the true spectroscopic redshift in a violin plot in Fig. 7 to quickly see which algorithms were able to capture the correct distribution.

From these results, we saw that all algorithms were able to successfully provide photometric redshift estimations. Using the violin plots from Fig. 7, we could see that the rough distribution was recovered by each algorithm, with the Multilayer Perceptron (MLP) producing a slightly more similar shape to the true redshifts. However, from simply looking at the outputs shown in Figs 6 and 7, it would be very difficult to determine which algorithm would be best to use. While the Decision Tree (DT) might be excluded due to the estimates being put into bands at set redshifts, its errors were still found to be quite low and it outperformed both the Linear Regression (LR) and MLP algorithms.

Looking at the metrics in Table 2 alone, the Random Forest (RF) performed best having the lowest errors with an $\mathrm{MAE} = 0.0385$ and $\mathrm{MSE} = 0.0042$; however, the other algorithms *k*-Nearest Neighbours (kNN), Boosted Decision Tree (BDT), and Extremely Randomized Trees (ERT) all performed incredibly similarly with $\mathrm{MAE} < 0.042$ and $\mathrm{MSE} < 0.0046$. Indeed, the BDT was almost

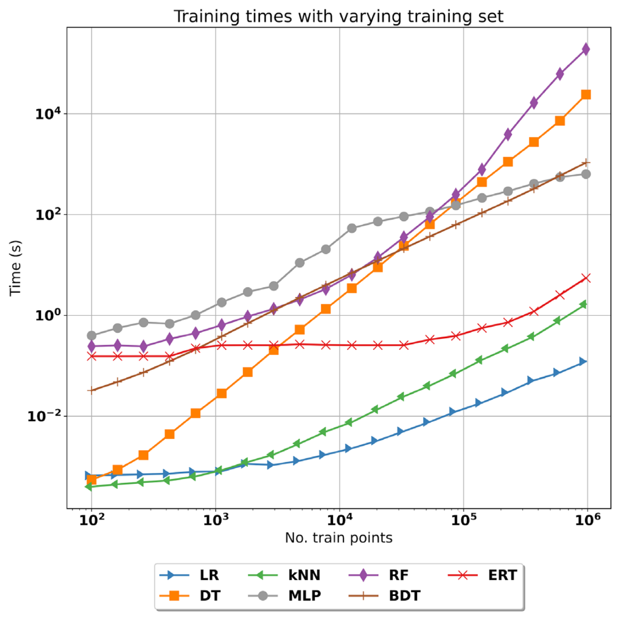

Figure 1. Graph of the training time plotted against the number of galaxies used in the training set to show how each algorithm scales with different sizes of training data sets. We saw that the simpler LR, kNN, and DT algorithms all begin as the fastest to train; however, the DT had terrible scaling and for large training sets became one of the slowest algorithms. Conversely, the ERT and MLP algorithms began as two of the slowest algorithms to train but scaled much better than the rest and could be more useful for massive training data sets.

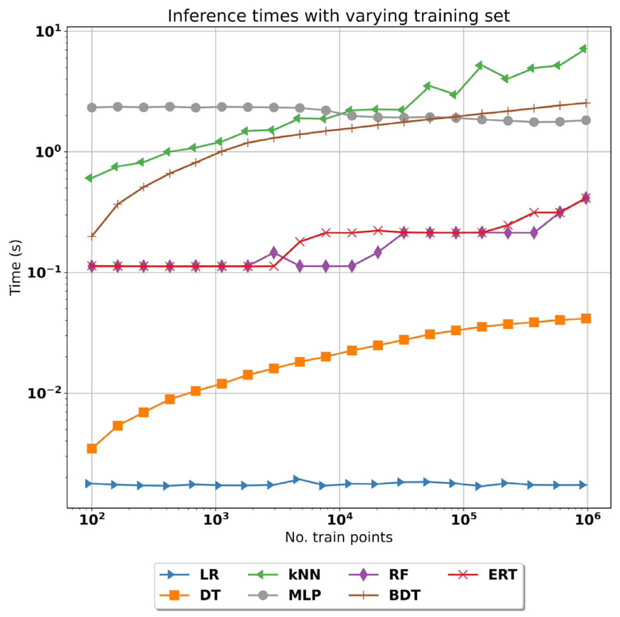

Figure 2. Graph of the inference time plotted against the number of galaxies used in the training set to show how each algorithm scaled with different sizes of training data sets (and a constant test set of 327 870 galaxies). We saw that all algorithms other than LR and MLP exhibit a training bloat, whereby the inference time increased with the number of galaxies included in the training set; however, the algorithms inference times generally increased by only a factor of 10 despite the training data set increasing by a factor of 10^4.

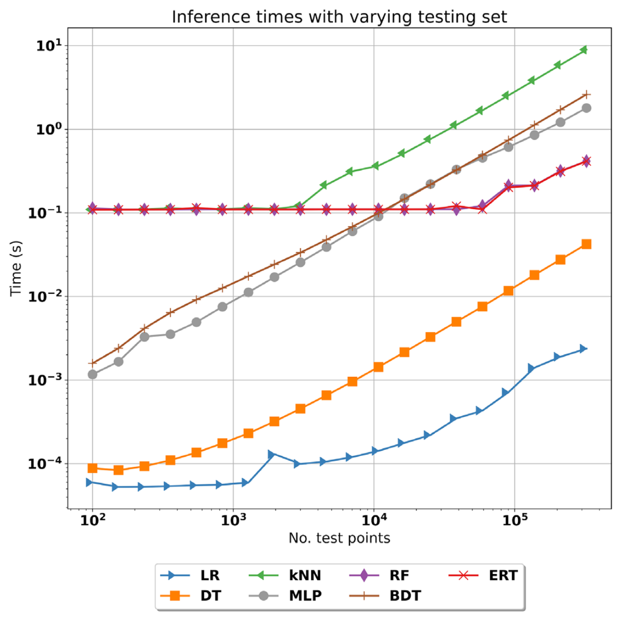

Figure 3. Graphs of the inference time plotted against the number of galaxies used in the testing set to show how each algorithm will scale with different sizes of testing data sets (and a constant training set of 983 608 galaxies). In inference, we saw all algorithms scaling very similarly with the main difference being the RF and the ERT where, during the period between 10^2 and 10^5 galaxies used in the test set, the inference time did not increase despite the number of galaxies to provide an estimate for increasing by a factor of 10^3. This meant that both algorithms ended up being faster to provide redshift estimations for larger test sets.

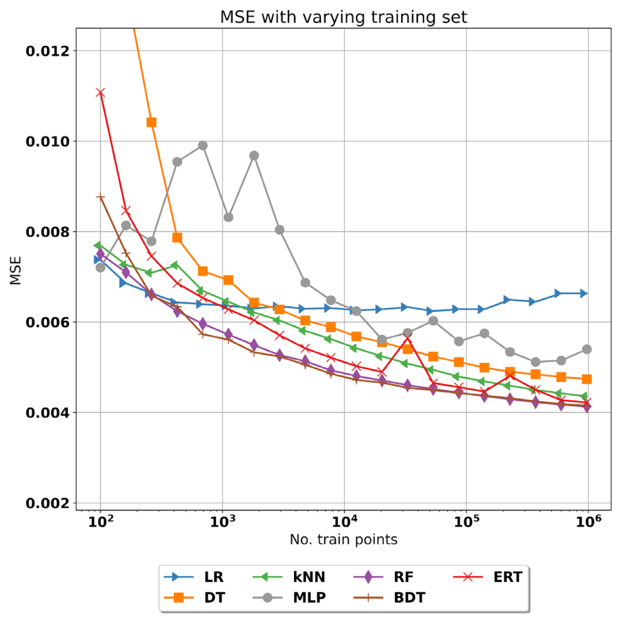

Figure 4. Graphs of the mean-squared error (MSE) plotted against the number of galaxies used in the training set to show how each algorithm's performance will scale with different sizes of training data sets (and a constant test set of 327 870 galaxies). As expected, in general, we saw all algorithms (other than LR) achieving lower MSE as the number of galaxies included in the training set was increased. However, we saw this increased error performance quickly plateau, and past 10^4 galaxies in the training set, there was very little reduction in error.

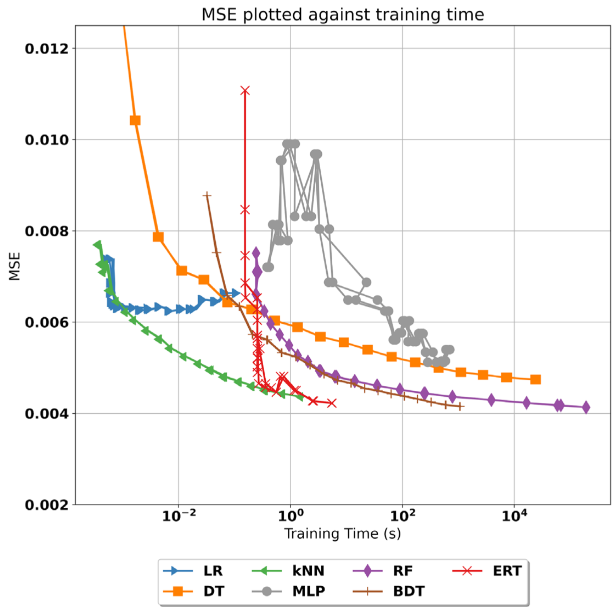

Figure 5. Graphs of the mean-squared error (MSE) plotted against the training times for each algorithm tested. For the ideal algorithm, one would see a curve down to the bottom left corner, thereby achieving the best possible result for the error in the shortest amount of time. In general, the algorithms do improve their errors as the training time (and number of galaxies included in the training set) increased; however, as shown, the improvement and the time taken vary greatly for each algorithm. The RF achieved the lowest error but with the longest training time, whereas the ERT can be seen to reach a very similar error much faster.

identically performing to the RF with a slightly improved precision and outlier fraction, and with such close performances of all the other algorithms, it was impossible to sufficiently determine which algorithm would be the most useful. To be able to further differentiate between them and determine which would be the best algorithm to use, it was therefore necessary to use the benchmarking results.

The results of the benchmarking performed for each algorithm are plotted in Fig. 1 (that shows the speed of training with varying sizes of training data sets), Figs 2 and 3 (that show the inference speeds with varying sizes of either training or testing data sets), Fig. 4 (that shows how the MSE varies as the number of galaxies in the training set increases), and Fig. 5 (that shows how the MSE varies with the time taken during training). As shown by these figures, the fastest algorithm overall was LR, which remained the fastest both in training and inference with increasing sizes of training and testing data sets. This was perhaps not surprising as out of the algorithms tested, it was the most simple model and as such required less computational resources both to train the model and to make its predictions. However, as LR also had by far the worst errors out of the algorithms tested (with errors around 30 per cent higher than those of the better performing algorithms), it seemed unlikely that it would ever be implemented for the problem of photometric redshift estimation.

Out of the other algorithms, the DT and the MLP were the poorer performing in terms of error. The DT was the second fastest behind LR in terms of inference, using its simple decision rules to quickly obtain the redshift estimations; however, as it also resulted in only estimating certain redshift bands, the final estimates were not as useful as other algorithms. Furthermore, the DT was the worst scaling algorithm for training and became the second slowest algorithm to train on a million galaxies. The MLP was also one of the slowest algorithms tested, starting as the slowest to train with small training sets and also being one of the slowest in inference. Although it is the simplest example of deep learning, it suffered from being one of the more complex algorithms tested and would perform better on even larger data sets with far more features, where it would have more chance to catch up to the other algorithms in both speed and error performance.

The remaining kNN, BDT, RF, and ERT algorithms all performed well in terms of error and were the hardest to differentiate between; however, using the benchmarking results, it was possible to see how differently they scaled. kNN was the simplest of the four better performing algorithms and, using the nearest neighbours to produce its estimates, resulted in the second fastest training times, only being beaten by LR. Although kNN was very fast to train, it was the slowest in inference and exhibited a bad ‘training bloat’ whereby the inference time increased as the number of galaxies in the training set was increased. While most other algorithms also displayed some level of this training bloat, it was worst for kNN due to the nature of its nearest neighbour search, which became more and more computationally expensive as more training points were added, and as such, it would not be as useful an algorithm for giving estimates for incredibly large data sets.

Out of the three ensemble tree-based methods, the RF scaled the worst in terms of training, becoming the slowest algorithm to train on the one million galaxies. Whereas, the ERT scaled surprisingly well and became the third fastest algorithm in training and similar to kNN. In training, the BDT was quite fast, scaling much better than the RF but worse than the ERT; however, when it came to inference, the BDT scaled worse than both the ERT and the RF and was the second slowest algorithm for large data sets. The RF and the ERT scaled almost identically in inference, which made sense being such similar algorithms, both only being beaten by the much simpler LR and DT.

As a result, it seemed like there was no clear best performing algorithm, but rather each algorithm could be useful in different situations. While the RF had the best error metrics, its terrible scaling with increasing training data meant that it would only be the best algorithm for problems where it could be trained once and it would be inefficient to use for problems that required the algorithm to be regularly retrained on large amounts of data. In that case, the BDT that had similar errors but was faster to train could be a more useful alternative, and similarly if both the training and inference times were required to be lower, the ERT would be a good compromise.

5 CONCLUSIONS

Producing reliable photometric redshift estimations will continue to be an incredibly important area of cosmology, and with future surveys producing more data than ever before, it will be vital to ensure that the methods chosen to produce the redshifts can be run efficiently.

Here, we showed how benchmarking can be used to provide a more complete view of how various ML algorithms’ performances scale with differing sizes of training and testing data sets. By combining the benchmarking results with the regression metrics, we were able to demonstrate how it is possible to distinguish between algorithms that appear to perform almost identically and suggest which could be better to implement in different scenarios. Furthermore, by suggesting a novel time-considered optimization process that takes

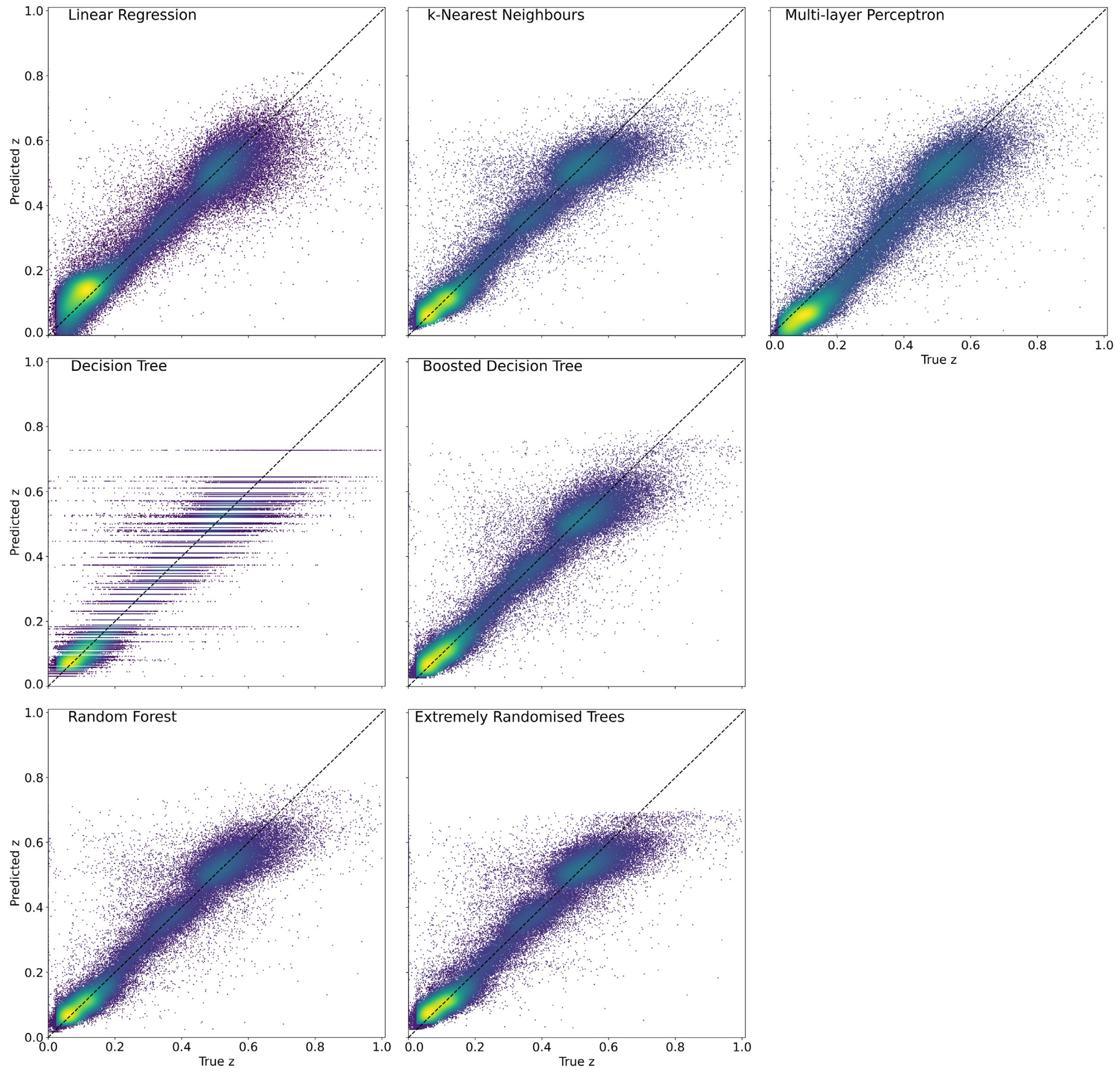

Figure 6. Graphs of photometric redshift estimates against the true spectroscopic redshift where the lighter shaded contours display the more densely populated regions. From top left to bottom right – Linear Regression (LR), *k*-Nearest Neighbours (kNN), Multilayer Perceptron (MLP), Decision Tree (DT), Boosted Decision Tree (BDT), Random Forest (RF), and Extremely Randomized Trees (ERT).

into account the benchmarking results during model selection, it was possible to provide additional insight into how ML algorithms can be fine-tuned to provide more appropriate models.

From our tests, we determined that while the kNN, BDT, RF, and ERT methods all seemed to perform very similarly, obtaining a good result for the MSE <0.0046, it was the RF that achieved the best metrics and was also one of the faster algorithms in inference. However, depending on which area of the pipeline an experiment requires to be faster, the RF method could also be inefficient as it scaled worse than all other algorithms in training. Hence, for problems that require regular retraining of models on large data sets, one of the other algorithms such as the BDT or the ERT could allow for a greater improvement. As large sky surveys producing enormous data sets will require the most efficient methods possible, it could also be necessary to investigate the use of deep learning neural networks that could benefit the most when using even larger amounts of data with more features.

Further work could be done to include a wider range of ML algorithms, including more deep learning networks, and to test them on larger simulated data sets to confirm their scaling. By making use of the time-considered optimization, it would also be possible to further examine the trade-offs between minimizing errors and the training/inference times in each individual algorithm. We could also run the benchmarks on a variety of computer architectures, making use of GPUs that have the potential to speed up the algorithms that are most parallizable, as well as allowing us to examine the

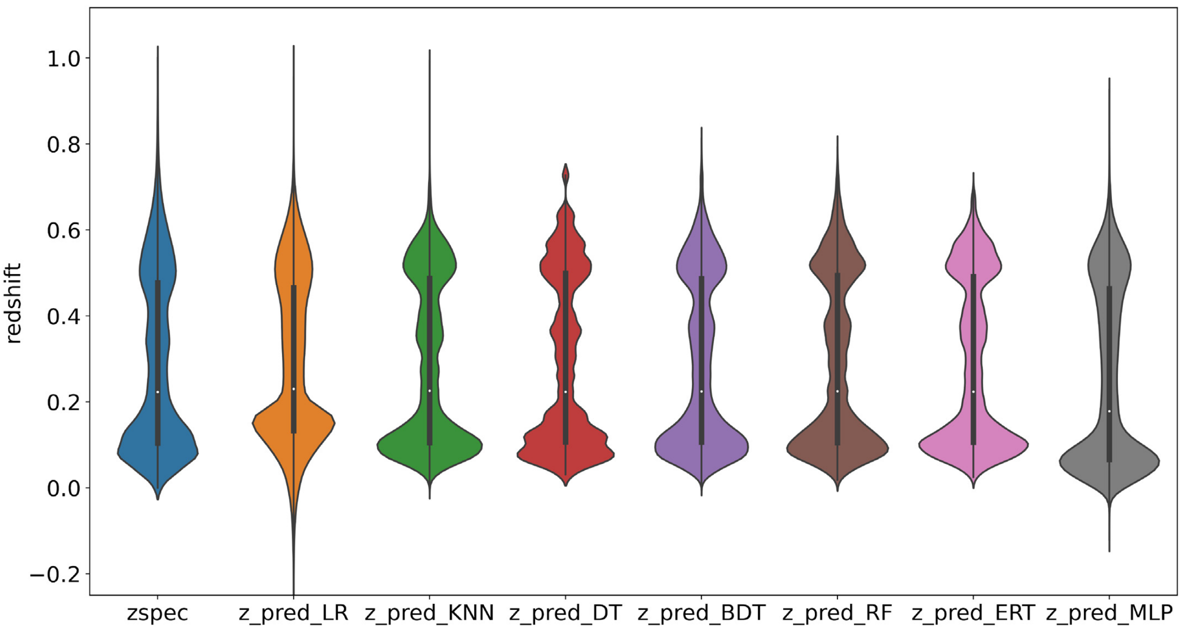

Figure 7. Violin plots showing the kernel density estimation of the underlying distributions of photometric redshift estimates of each algorithm along with the true spectroscopic redshift. From left to right – True spectroscopic redshift (zspec), Linear Regression (LR), *k*-Nearest Neighbours (kNN), Decision Tree (DT), Boosted Decision Tree (BDT), Random Forest (RF), Extremely Randomized Trees (ERT), and Multilayer Perceptron (MLP).

environmental impact of running such computationally expensive tasks.

ACKNOWLEDGEMENTS

BH was supported by the Science and Technology Facilities Council (STFC) UCL Centre for Doctoral Training in Data Intensive Science (grant no. ST/P006736/1). OL thanks All Souls College, Oxford, for a visiting fellowship. The authors also acknowledge the support from following grants: OL's European Research Council Advanced Grant (TESTDE FP7/291329), STFC Consolidated Grants (ST/M001334/1 and ST/R000476/1), JT's UKRI Strategic Priorities Fund (EP/T001569/1), particularly the AI for Science theme in that grant and the Alan Turing Institute, Benchmarking for AI for Science at Exascale (BASE), and EPSRC ExCALIBUR Phase I Grant (EP/V001310/1).

Funding for SDSS-III has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, and the U.S. Department of Energy Office of Science. The SDSS-III website is http://www.sdss3.org/.

SDSS-III is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS-III Collaboration including the University of Arizona, the Brazilian Participation Group, Brookhaven National Laboratory, Carnegie Mellon University, University of Florida, the French Participation Group, the German Participation Group, Harvard University, the Instituto de Astrofisica de Canarias, the Michigan State/Notre Dame/JINA Participation Group, Johns Hopkins University, Lawrence Berkeley National Laboratory, Max Planck Institute for Astrophysics, Max Planck Institute for Extraterrestrial Physics, New Mexico State University, New York University, Ohio State University, Pennsylvania State University, University of Portsmouth, Princeton University, the Spanish Participation Group, University of Tokyo, University of Utah, Vanderbilt University, University of Virginia, University of Washington, and ⟨0:funding-source 3:href="http://dx.doi.org/10.13039/100005326"⟩Yale University⟨/0:funding-source⟩.

DATA AVAILABILITY

The data used in this paper came entirely from the Sloan Digital Sky Survey data release 12 (SDSS-DR12) and are openly available from: https://www.sdss.org/dr12/.

REFERENCES

Abdalla F. B., Banerji M., Lahav O., Rashkov V., 2011, MNRAS, 417, 1891
Alam S. et al., 2015, ApJS, 219, 12
Altman N. S., 1992, Am. Stat., 46, 175
Amendola L. et al., 2018, Living Rev. Relativ., 21, 2
Benitez N., 2000, ApJ, 536, 571
Bolzonella M., Miralles J.-M., Pelló R., 2000, A&A, 363, 476
Breiman L., 1996, Mach. Learn., 24, 123
Breiman L., 2001, Mach. Learn., 45, 5
Breiman L., Spector P., 1992, International statistical review/revue internationale de Statistique. p. 291
Breiman L., Friedman J., Olshen R., Stone C. J., 1984, in Classification and Regression Trees Routledge Boca Raton, FL, USA
Collister A. A., Lahav O., 2004, PASP, 116, 345
D'Isanto A., Polsterer K. L., 2018, A&A, 609, A111
DES Collaboration, 2005, Int. J. Mod. Phys. A, 20, 3121
DES Collaboration, 2016, MNRAS, 460, 1270
Dewdney P. E., Hall P. J., Schilizzi R. T., Lazio T. J. L., 2009, Proc. IEEE, 97, 1482
Dongarra J. J., Moler C. B., Bunch J. R., Stewart G. W., 1979, LINPACK users' guide. SIAM. Philadelphia, USA
Dongarra J. J., Luszczek P., Petitet A., 2003, Concurrency Comput. Pract. Exp., 15, 803

Eisenstein D. J. et al., 2011, AJ, 142, 72
Euclid Collaboration 2020, A&A, 644, A31
Flaugher B., Bebek C., 2014, in Ground-based and Airborne Instrumentation for Astronomy V, The Dark Energy Spectroscopic Instrument (DESI). SPIE. Bellingham, WA, USA, p. 91470S
Friedman J. H., 2002, Comput. Stat. Data Anal., 38, 367
Geurts P., Ernst D., Wehenkel L., 2006, Mach. Learn., 63, 3
Gunn J. E. et al., 2006, AJ, 131, 2332
Hastie T., Tibshirani R., Friedman J., 2009, in The elements of statistical learning. Springer, New York, NY, USA. p. 43
Hoyle B., 2016, Astron. Comput., 16, 34
Ilbert O. et al., 2006, A&A, 457, 841
Ivezić Ž. et al., 2019, ApJ, 873
Kirk D., 2007, in Proceedings of the 6th International Symposium on Memory Management. ISMM '07. Association for Computing Machinery, New York, NY, USA, p. 103
LeCun Y. A., Bottou L., Orr G. B., Müller K.-R., 2012, Efficient BackProp. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 9
Martini P. et al., 2018, in Ground-based and Airborne Instrumentation for Astronomy VII. SPIE, Bellingham, WA, USA, p. 410
Pasquet J., Bertin E., Treyer M., Arnouts S., Fouchez D., 2019, A&A, 621, A26
Pedregosa F. et al., 2011, J. Mach. Learn. Res., 12, 2825
Petrosian V., 1976, ApJ, 209, L1
Schmidt S. J. et al., 2020, MNRAS, 499, 1587
Sejnowski T. J., 2018, The deep learning revolution. Mit Press Cambridge, MA, USA
Soo J. Y. H. et al., 2017, MNRAS, 475, 3613
Springel V., 2005, MNRAS, 364, 1105
Tyson J., Wittman D., Hennawi J., Spergelb D., 2003, Nucl. Phys. B Proc. Suppl., 124, 21
Werbos P. J., 1988, Neural Netw., 1, 339
York D. G. et al., 2000, AJ, 120, 1579

This paper has been typeset from a TEX/LATEX file prepared by the author.