



PHOTOMETRIC SUPERNOVA CLASSIFICATION WITH MACHINE LEARNING

MICHELLE LOCHNER¹, JASON D. McEWEN², HIRANYA V. PEIRIS¹, OFER LAHAV¹, AND MAX K. WINTER¹¹Department of Physics and Astronomy, University College London, Gower Street, London WC1E 6BT, UK; dr.michelle.lochner@gmail.com²Mullard Space Science Laboratory, University College London, Surrey RH5 6NT, UK

Received 2016 March 15; revised 2016 July 6; accepted 2016 July 6; published 2016 August 23

ABSTRACT

Automated photometric supernova classification has become an active area of research in recent years in light of current and upcoming imaging surveys such as the Dark Energy Survey (DES) and the Large Synoptic Survey Telescope, given that spectroscopic confirmation of type for all supernovae discovered will be impossible. Here, we develop a multi-faceted classification pipeline, combining existing and new approaches. Our pipeline consists of two stages: extracting descriptive features from the light curves and classification using a machine learning algorithm. Our feature extraction methods vary from model-dependent techniques, namely SALT2 fits, to more independent techniques that fit parametric models to curves, to a completely model-independent wavelet approach. We cover a range of representative machine learning algorithms, including naive Bayes, k -nearest neighbors, support vector machines, artificial neural networks, and boosted decision trees (BDTs). We test the pipeline on simulated multi-band DES light curves from the Supernova Photometric Classification Challenge. Using the commonly used area under the curve (AUC) of the Receiver Operating Characteristic as a metric, we find that the SALT2 fits and the wavelet approach, with the BDTs algorithm, each achieve an AUC of 0.98, where 1 represents perfect classification. We find that a representative training set is essential for good classification, whatever the feature set or algorithm, with implications for spectroscopic follow-up. Importantly, we find that by using either the SALT2 or the wavelet feature sets with a BDT algorithm, accurate classification is possible purely from light curve data, without the need for any redshift information.

Key words: cosmology: observations – methods: data analysis – supernovae: general

1. INTRODUCTION

Astronomy is entering an era of deep, wide-field surveys and massive data sets, which requires the adoption of new, automated techniques for data reduction and analysis. In the past, supernova data sets were small enough to allow spectroscopic follow-up for the majority of objects, confirming the type of each. Only type Ia's are currently used for cosmology, and the type is of course required for astrophysical modeling and studies. With the onset of surveys such as the Dark Energy Survey (DES; Dark Energy Survey Collaboration 2005; Dark Energy Survey Collaboration et al. 2016) and the upcoming Large Synoptic Survey Telescope (LSST; LSST Science Collaboration 2009), only a small fraction of the data set can be spectroscopically followed-up. The current commonly used data set, the Joint Light curve Analysis (Betoule et al. 2014), contains only 740 supernovae, while DES is expected to detect thousands (Bernstein et al. 2012) and LSST hundreds of thousands (LSST Science Collaboration 2009) of supernovae. Thus alternative approaches to supernova science must be developed to leverage these largely photometric data sets.

Supernova cosmology is possible without strictly knowing the supernova type using, for example, Bayesian methods (Kunz et al. 2007; Hlozek et al. 2012; Newling et al. 2012; Knights et al. 2013; Rubin et al. 2015). However, these techniques benefit from having a reasonable probability for the type of each object in the data set, so some form of probabilistic classification is useful. Additionally, studies of core-collapse supernovae and other transients rely on good photometric classification. Furthermore, the observing strategy for LSST has not yet been finalized and the effect of observing strategy on supernova classification has not yet been established. Here we outline a multi-faceted pipeline for photometric supernova

classification. In future work, we will apply it to LSST simulations to understand the effect of observing strategy on classification.

Current photometric supernova classification techniques focus on empirically based template fitting (Sako et al. 2008, 2014). However, in the past few years there have been several innovative techniques proposed to address this problem (see Kessler et al. 2010a and the references therein and Ishida & de Souza 2013).

Here, we apply machine learning to this problem, as a well-established method of automated classification used in many disciplines. As astronomical data sets become larger and more difficult to process, machine learning has become increasingly popular (Ball & Brunner 2010; Bloom & Richards 2012). Machine learning techniques have been proposed as a solution to an earlier step in the supernova pipeline, that of classifying transients from images (du Buisson et al. 2015; Wright et al. 2015). Machine learning is also already being employed at some level for photometric supernova classification in the Sloan Digital Sky Survey (SDSS; Frieman et al. 2008; Sako et al. 2008), using the parameters from template-fitting as features (Sako et al. 2014).

We investigate the effect of including host galaxy photometric redshift information in automated classification. Reliable redshifts are important for current classification techniques in order to reliably fit the templates. However, for future surveys such as LSST, well-constrained, unbiased redshifts may be difficult to obtain and could negatively affect classification. A classification technique that is independent of redshift information could therefore be invaluable.

Additionally, we investigate the effect of providing the machine learning algorithms with a non-representative training set. In general, one would expect the spectroscopically

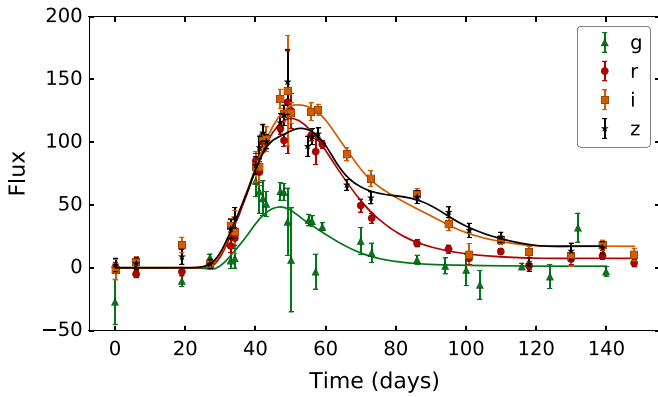


Figure 1. An example simulated DES type Ia supernova light curve (object ID 009571), at redshift 0.42, from the Supernova Photometric Classification Challenge (Kessler et al. 2010a, 2010b). The photometric light curve is measured in the four DES *griz* filter bands, showing the rise in brightness after the explosion of the star and subsequent decay. The points and error bars are the data points, while the curves are from the best fitting SALT2 model (see Section 3).

confirmed training set to be biased in terms of brightness (and hence redshift) because it is far easier to obtain spectra for brighter objects. However, any classification technique is likely to underperform with a biased training set. In our analysis, we explore both representative and non-representative training sets.

In this paper, we investigate four different methods of extracting useful features from light curves, as well as five representative machine learning algorithms. In Section 2 we summarize the simulated data set used. Section 3 explains our feature extraction methods, varying from highly model-dependent approaches to a completely model-independent method. Section 4 introduces the machine learning algorithms used in this work. We present our results in Section 6 and conclude in Section 7.

2. SUPERNOVA SIMULATIONS

To test and compare methods, we use existing supernova simulations from the Supernova Photometric Classification Challenge (SPCC; Kessler et al. 2010a, 2010b), which are simulated DES light curves built from an existing library of non-Ia templates and using both SALT2 (Guy et al. 2007) and MLCS2k2 (Jha et al. 2007) type Ia models. Simulated photometric host galaxy redshifts are available for each object and we investigate the effect of withholding this information from the machine learning algorithms, in order to see if redshift is an essential ingredient for classification. We have not tested whether or not the results would improve if spectroscopic redshifts were available, since for DES and LSST this is an unlikely scenario.

In the original challenge, a training set of 1103 spectroscopically confirmed objects was provided while a test set of 20216 objects was retained to compare the techniques under study. It is important to note that this training set is non-representative in brightness and hence also in redshift. This was done to emulate the way spectroscopic follow-up is currently performed, which prioritizes bright objects. We return to this point in Section 6. Figure 1 shows an example type Ia light curve from the challenge data.

3. EXTRACTING USEFUL LIGHT CURVE FEATURES

Feature extraction is a crucial step in the machine learning process. Raw data almost inevitably require dimensionality reduction or some form of summarizing of key features in the data, and very few data sets can be successfully classified in a raw format. Good feature extraction techniques will produce a feature set of much lower dimensionality than the original data with features that adequately describe the original data and are well-separated between classes. Examples of commonly used features include principal component analysis (PCA; Pearson 1901; Hotelling 1933) coefficients or derived quantities such as amplitude, period, phase etc. For a general overview of feature extraction, including methods of evaluating class-separability, see Li et al. (2016).

We test a variety of feature sets and evaluate their performance based on the effectiveness of subsequent classification for a variety of machine learning algorithms. Alternative feature selection approaches that are agnostic of machine learning algorithms could be considered, although, since these approaches evaluate the feature sets in isolation, they could lead to biases after a machine learning algorithm is applied. A detailed investigation of alternative approaches to choosing and evaluating features is beyond the scope of this article.

An alternative approach to feature extraction and subsequent machine learning classification is deep learning (Schmidhuber 2014; LeCun et al. 2015), which attempts to automatically learn summary features. Despite this advantage, deep learning has high computing requirements and needs a much larger training set than traditional algorithms. We found traditional feature extraction methods to be more than adequate for this problem. We thus consider the added flexibility and complexity of the deep learning approach, though worth exploring further, unnecessary for this application at present.

We explore three different, broad approaches to feature extraction, ranging from a highly model-dependent method based on supernova templates to a completely model-independent wavelet based approach, with a parametric approach somewhere in between. These bracket a wide range of approaches that one could consider using to extract features from light curves. We run a pipeline whereby we extract the four feature sets from the same data, and then run five different machine learning algorithms (described in Section 4) on each feature set to compare performance.

Before discussing the feature extraction methods we have studied, we introduce a useful visualization technique for understanding feature sets.

3.1. Visualizing Feature Sets with *t*-distributed Stochastic Neighbor Embedding (*t*-SNE)

Although feature extraction generally reduces dimensionality from raw data, the number of features often remains large, making it difficult to determine whether or not a particular feature set separates the classes well. *t*-SNE (Van der Maaten & Hinton 2008) is a sophisticated technique designed to produce a low-dimensional representation of a high-dimensional space, while clustering similar features together. *t*-SNE works by first computing, for each pair of points, the probability that the two points are similar based on their Euclidean distance. The aim then is to find the corresponding low-dimensional values for these points that preserves the same probability. Stochastic neighbor embedding (SNE) determines these values by

Table 1
Priors on the SALT2 Model Parameters

Parameter Name	Prior
z	$\mathcal{U}(0.01, 1.5)$
t_0	$\mathcal{U}(-100, 100)$
x_0	$\mathcal{U}(-10^{-3}, 10^{-3})$
x_1	$\mathcal{U}(-3, 3)$
c	$\mathcal{U}(-0.5, 0.5)$

minimizing the sum of Kullback–Leibler divergences (a simple measure of divergence between probability distributions) over all data points, using a gradient descent algorithm with a stochastic element to avoid local minima. t-SNE differs from SNE in the choice of exact probability distribution defining the similarity between points (using a Student-t distribution instead of a Gaussian) and in the exact choice of cost function to minimize (see Van der Maaten & Hinton 2008 for details). If the classes are well-separated on the t-SNE plot, one can generally expect accurate classification. However, the converse is not strictly true and a feature set with poor class-separation may still be well-classified with a sophisticated machine learning algorithm. Each of the feature extraction methods discussed below has an accompanying t-SNE plot for visualization purposes.

3.2. Template Fitting Approach to Classification

Historically, classification of photometric supernovae (generally used to identify candidates for spectroscopic follow-up) has focused on the use of supernova templates constructed from existing data sets of supernovae (Sako et al. 2008). Light curve features such as stretch and color parameters have been used to classify photometric supernovae from SDSS (Sako et al. 2011). We choose the SALT2 (Spectral Adaptive Light curve Template 2) model (Guy et al. 2007) of type Ia supernovae, as it is the most commonly used.

In the SALT2 model, the flux in a given band of a type Ia light curve is given by

$$F(t, \lambda) = x_0 \times [M_0(t, \lambda) + x_1 M_1(t, \lambda) + \dots] \times \exp[\text{cCL}(\lambda)], \quad (1)$$

where t is the phase (time since maximum brightness in the B -band), λ is the rest frame wavelength, $M_0(t, \lambda)$ is average spectral sequence (using past supernova data as described in Guy et al. (2007)), and $M_k(t, \lambda)$ for $k > 0$ are higher order components to capture variability in the supernovae. In practice, only $M_0(t, \lambda)$ and $M_1(t, \lambda)$ are used. $\text{CL}(\lambda)$ is the average color correction law. For each object, the redshift z is also used as either a given or fitted parameter. Thus this method has five features: z , t_0 (the time of peak brightness in the B -band), x_0 , x_1 , and c . We use the implementation of SALT2 in `sncosmo`³ (Barbary 2014) and obtain the best fitting parameters using `MultiNest`⁴ (Feroz & Hobson 2008; Feroz et al. 2009, 2013) with the `pyMultiNest`⁵ (Buchner et al. 2014) software. Table 1 lists the priors used on all parameters.

³ <http://sncosmo.readthedocs.org/en/v1.2.x/>

⁴ <https://ccpforge.cse.rl.ac.uk/gf/project/multinest/>

⁵ <https://johannesbuchner.github.io/PyMultiNest/>

Figures 2(a) and (b) show the t-SNE plots for the SALT2 features both without and with redshift information. The t-SNE plots show a lot of structure, partly due to the relative low-dimensionality of the feature space, which corresponds to a highly constrained 2D plot. The classes are fairly well-separated, especially when redshift is included, but it is clear that it is difficult to distinguish between type Ia and Ib supernovae.

3.3. General Parametric Approach to Classification

The template-fitting approach relies on a significant amount of prior knowledge about supernovae. An alternative is to use a more general parametric description and use the fitted parameters as features in the machine learning pipeline. We use two different parametric models that have been proposed in the literature as good parameterizations of a supernova light curve and were both used in solutions to the SPCC. Model 1 is the parameterization used in Newling et al. (2011), while Model 2 is proposed in Karpenka et al. (2013).

We use these models because they are both proposed solutions to precisely the problem of supernova classification. While similar, the models have some differences, especially in the treatment of the tail of the light curves, the ability to describe double peaks and the small difference in the number of parameters, which makes it interesting to compare them.

We also considered the linear piecewise model proposed in Sanders et al. (2015) but found that the relatively large number of parameters (11 per filter) made it less robust when fitting to data, and consequently classification with this feature set did not typically perform well.

3.3.1. Model 1

The model used in Newling et al. (2011) describes the flux of a supernova (in any band) by

$$F(t) = A \left(\frac{t - \phi}{\sigma} \right)^k \exp \left(- \frac{t - \phi}{\sigma} \right) k^{-k} e^k + \Psi(t), \quad (2)$$

where

$$\Psi(t) = \begin{cases} 0 & -\infty < t < \infty \\ \text{cubic spline} & \phi < t < \tau, \\ \psi & \tau < t < \infty \end{cases} \quad (3)$$

and $A + \psi$ is the peak flux, ϕ is the starting time of the explosion, k governs the relative rise and decay times of the light curve, and σ is the width or “stretch” of the light curve. The time of the peak flux, τ , is given by $\tau = k\sigma + \phi$. The function $\Psi(t)$ is a “tail” function that ensures that the flux tends to a finite value as $t \rightarrow \infty$ and the cubic spline is determined to have zero derivative at $t = \phi$ and $t = \tau$.

Each filter band is fitted individually with these 5 parameters (A , ϕ , σ , k , ψ), giving a total of 20 features for each object (or 21 if redshift is included). The parameters are all fitted with `MultiNest`, as in Section 3.2, and the best fit is taken to be the maximum posterior value for each parameter. The priors on the parameters (which are the same for all four filter bands) are given in Table 2.

Figures 2(c) and (d) show the t-SNE plots for the Model 1 feature set both without and with redshift information, respectively. In both cases, the features do not appear to be easily separable.

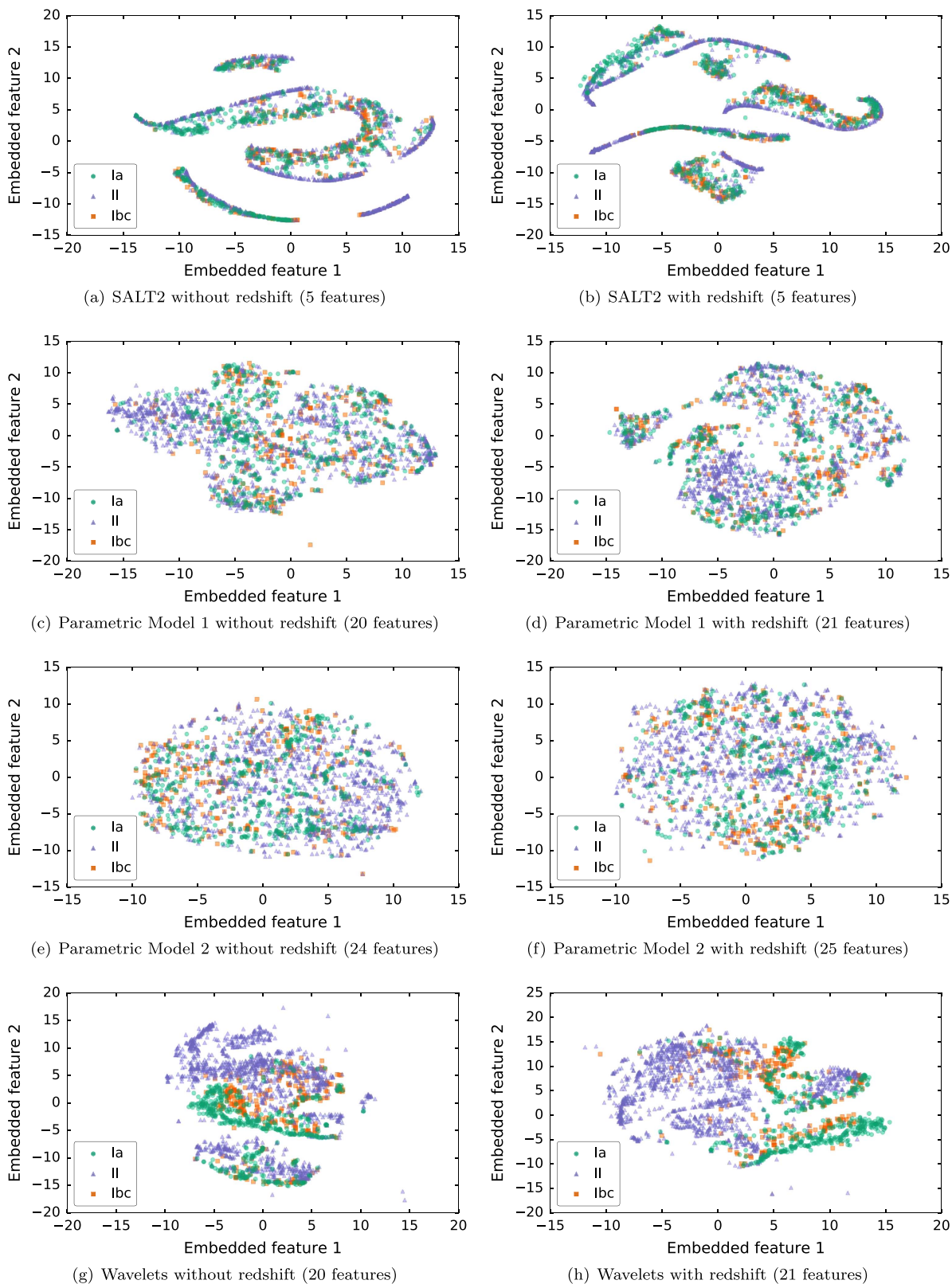


Figure 2. t-SNE (t-distributed stochastic neighbor embedding) visualizations for feature sets with and without redshift information. This is a two-dimensional visualization of the high-dimensional feature space. Each of the three classes of supernovae is represented by a different color; if the classes are well-separated in the embedding space, we would expect these features to provide good classification. Note that all units are arbitrary and the original dimensionality is indicated in brackets. The reader is reminded that these plots are for visualization purposes only and the full classification pipeline is required to draw conclusions about the success of any feature set for classification.

Table 2
Priors on the Model 1 Parameters

Parameter Name	Prior
$\log(A)$	$\mathcal{U}(0, 10)$
ϕ	$\mathcal{U}(-60, 100)$
$\log(\sigma)$	$\mathcal{U}(-3, 4)$
$\log(k)$	$\mathcal{U}(-4, 4)$
$\log(\psi)$	$\mathcal{U}(-6, 10)$

3.3.2. Model 2

The proposed parameterization from Karpenka et al. (2013) is similar to Model 1 but allows for the possibility of double peaks in light curves. The flux is given by

$$F(t) = A[1 + B(t - t_1)^2] \frac{\exp[-(t - t_0)/T_{\text{fall}}]}{1 + \exp[-(t - t_0)/T_{\text{rise}}]}, \quad (4)$$

where t is assumed to be the earliest measurement in the r -band light curve. We again fit all parameters using `MultiNest` and take the maximum posterior parameters. There are 6 parameters per filter band, thus forming a feature set of 24 features (or 25 including redshift) for each object. We use the same priors as in Karpenka et al. (2013), given in Table 3.

The t-SNE plots for the Model 2 features can be seen in Figures 2(e) and (f) without and with redshift, respectively. As in Model 1, the features do not appear to be well-separated.

3.4. Wavelet Decomposition Approach to Classification

While the model-dependent approaches to feature extraction considered so far are useful for incorporating prior information about SN light curves, they are sensitive to mis-specification of light curve models. It is quite possible that important light curve characteristics cannot be captured efficiently by simple models, such as Models 1 and 2 described above. A model-independent, non-parameteric approach to feature extraction is therefore required as a complementary approach that is not dependent on prior information.

A wavelet decomposition is a good choice for feature extraction since certain wavelet transforms can be constructed that are approximately invariant under translation and stretch. One could then expect the wavelet coefficients of two similar supernovae to be similar, even if the light curves have different explosion times or redshifts.

Recently, Varughese et al. (2015) showed that a wavelet-based approach to photometric supernova classification can be highly effective. Here we consider a substantially different method in the actual wavelet decomposition of the light curves and subsequent classification, but also find wavelets to be an excellent feature extraction method.

The procedure we follow to extract features using wavelets is to first interpolate the light curves onto the same grid of points (using Gaussian processes), then to perform a redundant wavelet decomposition, and lastly to use PCA to reduce the dimensionality of the feature set. We note that there are many alternative dimensionality reduction techniques, such as independent component analysis (Comon 1994), autoencoders (Ballard 1987; Kramer 1991) and Sammon mapping (Sammon 1969), which may improve our classification results, but conclude that for this work PCA is more than sufficient.

Table 3
Priors on the Model 2 Parameters

Parameter Name	Prior
$\log(A)$	$\mathcal{U}(\log(10^{-5}), \log(1000))$
$\log(B)$	$\mathcal{U}(\log(10^{-5}), \log(100))$
t_0	$\mathcal{U}(0, 100)$
t_1	$\mathcal{U}(0, 100)$
T_{rise}	$\mathcal{U}(0, 100)$
T_{fall}	$\mathcal{U}(0, 100)$

3.4.1. Interpolation Using a Gaussian Process Regression

In order to perform a wavelet decomposition, all light curves are interpolated onto the same uniform grid of points. We use Gaussian process regression since it has the advantage over (for example) spline interpolation in that it allows the straightforward inclusion of uncertainty information and produces a less biased estimate of interpolated values.

A Gaussian process is the generalization of a Gaussian distribution, forming a distribution for which each point of the input variable is associated with a normally distributed random variable (MacKay 2003). A Gaussian process is fully specified by a mean function of the input variable (time, in the case of a light curve) and a covariance function, which specifies the covariance between values of the function at pairs of points in time. Gaussian process regression then proceeds by determining the mean function and hyperparameters of the covariance matrix. We use the package `GaPP`⁶ used in Seikel et al. (2012) to perform the Gaussian process regression of each of the light curves.

3.4.2. Wavelet Decomposition

Wavelet transforms are an invaluable tool in signal analysis due to the ability of wavelets to localize signal content in scale and time simultaneously, whereas real space or Fourier representations provide signal localization, respectively, in time or frequency only (see Valens 1999 for a gentle introduction to wavelets). For the continuous wavelet transform (CWT), the set of wavelet atoms forming the transform dictionary is scale- and translation-invariant, i.e., a scaled or translated atom also belongs to the dictionary, hence the CWT achieves scale- and translation-invariance (Mallat 2009). However, the discrete wavelet transform (DWT), which permits fast transforms and exact synthesis from a finite, discrete set of wavelet coefficients, suffers from a lack of translation-invariance due to the critical sub-sampling performed. Approximate translation-invariance can be achieved by eliminating sub-sampling from the DWT, leading to the redundant à trous wavelet transform (Holschneider et al. 1989; Mallat 2009), which is also called the stationary wavelet transform.

We adopt the à trous wavelet transform, which also achieves dyadic scale-invariance, and use the symlet family of wavelets, which are a more symmetric version of the widely used Daubechies family of wavelets (Daubechies 1988). We use the implementation provided in the `PyWavelets`⁷ software package. Our results were found not to be highly dependent on the family of wavelet chosen. Alternative wavelet

⁶ <http://www.acgc.uct.ac.za/~seikel/GAPP/main.html>

⁷ <http://www.pybytes.com/pywavelets/contents.html>

constructions that achieve better translation- and scale-invariance could be considered (e.g., Kingsbury 2001; Lo et al. 2009; Xiong et al. 2000) but a detailed optimization of the wavelet transform used was not performed, highlighting the robustness of our approach.

For this work, we used 100 points on the Gaussian process curve and a two-level wavelet transform, returning 400 (highly redundant) coefficients per filter, or 1600 coefficients per object.

3.4.3. Dimensionality Reduction with PCA

The output of the wavelet decomposition is highly redundant (which preserves translation invariance) and thus extremely high-dimensional. The final step in the process is to run a PCA to reduce dimensionality. PCA is a linear transform that transforms a set of correlated variables into linearly uncorrelated variables using singular value decomposition of the matrix of variables. From this, a set of eigenvectors (components) and eigenvalues is obtained. The components with large eigenvalues describe the majority of the variability of the data set and dimensionality reduction is achieved by discarding components with negligible eigenvalues. After PCA, we reduce the dimensionality of the feature set from 1600 to 20 dimensions while retaining 98% of the information in the data set (as measured by determining the fraction of the sum of eigenvalues of components retained to that of all components).

We show the t-SNE plots for the wavelet features without and with redshift in Figures 2(g) and (h). Here, the features seem encouragingly well-separated between classes, even between type Ia and Ibc in some parts of feature space. We would thus expect the wavelet features to perform well in classification.

4. MACHINE LEARNING FOR CLASSIFICATION

Machine learning is a powerful tool for modern astronomy (Ball & Brunner 2010; Bloom & Richards 2012), becoming increasingly popular in the era of massive data sets. It allows the automation of tasks previously performed by humans and also, as in this case, allows the classification of objects that are seemingly inseparable to the human eye. Supervised machine learning algorithms learn a mapping function from training data to allow them to classify new test objects. A large variety of machine learning algorithms exist, and it is beyond the scope of this paper to consider each one. Instead, we consider five popular machine learning algorithms that are fairly representative of the main approaches used. All these algorithms and several others are comprehensively compared in Caruana & Niculescu-Mizil (2006), who find (as we do) that boosted decision trees (BDTs) and random forests often outperform other classifiers for many problems. For all algorithms, we use the Python package `scikit-learn`⁸ (Pedregosa et al. 2011). The selected algorithms used are able to compute not just the classification of a particular object, but also a classification probability. This probability can be used to apply a cut to the set of classified objects to ensure a minimum level of purity in a given class.

4.1. Description of Machine Learning Algorithms

The algorithms used in this work are: naive Bayes (a basic probabilistic classifier), k -nearest neighbors (KNN) (a classifier based on clustering of features), a multi-layer perceptron (MLP; the simplest form of artificial neural network; ANN), a support vector machine (SVM; which works by finding a separating hyperplane between classes) and BDTs (an ensemble method that combines a multitude of decision trees, a weaker classifier). What follows is a brief overview of the five algorithms considered, in each case referring the reader to references for more details.

4.1.1. Naive Bayes (NB)

Given a set of features for an object, x_1, x_2, \dots, x_n , the Naive Bayes (NB) algorithm (e.g., Zhang 2004) states that the probability of that object belonging to class y is

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_i P(x_i|y), \quad (5)$$

assuming independence between features. $P(y)$ can be estimated by determining the frequency of y in the training set. In the implementation of NB that we use, the likelihood, $P(x_i|y)$, is assumed to be Gaussian, the parameters of which are determined from the training data by maximizing the likelihood. The predicted class is simply determined to be the class that maximizes Equation (5).

The advantage of NB is that it is fast and scales very well to high dimensions. The disadvantage is that it assumes independence between features and that the features are Gaussian-distributed. One or both of these assumptions is frequently broken. NB has recently been used in astronomy to classify asteroids to determine targets for spectroscopic follow-up (Oszkiewicz et al. 2014).

4.1.2. k -nearest Neighbors

KNN (see e.g., Altman 1992) is a simple algorithm that classifies an object by performing a majority vote of the classes of the k nearest neighbors. We use a Euclidean distance measure to determine the nearest neighbors, applying a weight to each neighbor that is inversely proportional to the distance.

The probability of an object belonging to class y is found by summing the weights of all neighbors also belonging to class y and dividing by the sum of the weights of all neighbors. The advantage of KNN is its relative simplicity and its use as a clustering, unsupervised learning algorithm. The disadvantages are that it is computationally intensive for large data sets, given that all pairwise distances have to be calculated; furthermore, the results are highly dependent on the exact training set used, making the results fairly inconsistent, especially for small training sets. An example of a recent application of KNN in astronomy can be found in Kügler et al. (2015), where it is applied to spectroscopic redshift estimation.

4.1.3. Artificial Neural Networks

ANNs (e.g. Jeffrey & Rosner 1986) are a family of machine learning algorithms inspired by biological systems of interconnected neurons. The aim of an ANN is to learn a nonlinear function to map input features to output classes. For this work we use the commonly used MLP implemented in `scikit-learn`.

⁸ <http://scikit-learn.org/>

Each neuron transforms the inputs of the previous layer by constructing a weighted sum of the features, followed by some nonlinear activation function, in this case a hyperbolic tan function. While in theory different topologies could be used, the MLP is constructed as a series of layers of neurons. In each layer, each neuron is connected to all the neurons of the previous layer, but not any others.

The standard topology is: $n \rightarrow h_1 \rightarrow h_2 \dots h_m \rightarrow c$, where n is the number of input features, h_i is the number of neurons in layer i , and c is the final class. As is often seen in the literature, we find it does not improve performance to add more than one hidden layer. The `scikit-learn` implementation of a MLP uses backpropagation (Werbos 1974) to determine the weights.

The probability of an object belonging to a particular class is easily estimated from a MLP by normalizing the final activation function values so that they sum to one, thus treating them as probabilities. Neural networks have the advantage of being capable of learning highly nonlinear mappings, often allowing highly accurate classifications. The disadvantages of neural networks are their sensitivity to the tuning of hyperparameters, such as the number of hidden layers and the number of nodes in each layer, and their tendency to overfit the data. There are many examples of the use of ANNs in astronomy, including galaxy classification (Lahav et al. 1995) and their recent application to photometric redshift estimation (Sadeh et al. 2015).

4.1.4. Support Vector Machines

SVMs (e.g., Cortes & Vapnik 1995) are a type of classifier that works by finding the hyperplane in feature space that best separates classes (Ball & Brunner 2010). This hyperplane is defined by the vectors to the data points nearest the hyperplane, called the support vectors. Linear SVMs can be extended to the nonlinear case by using the kernel trick (Aizerman 1964) to transform the features to a high-dimensional space so that nonlinear relationships become linear. For this work, we use a radial basis function as a kernel, which is a Gaussian function of the Euclidean distance between features of different objects.

Probabilities from an SVM can be obtained using Platt scaling (Platt et al. 1999, p. 61), which performs a logistic regression of the SVM's decision function scores. Platt scaling is computationally intensive, requiring an extra cross-validation step to determine the parameters of the transformation function. SVMs perform extremely well in general, even in high dimensions, and are highly versatile, as different kernel functions can be considered to improve feature separation in any particular problem. The disadvantage of SVMs lies largely in their computational complexity when computing probabilities (if required). A very recent application of SVMs in astronomy is the classification of sources (Kurcz et al. 2016) for the *WISE* survey (Wright et al. 2010).

4.1.5. Boosted Decision Trees

Random forests (Breiman 2001) and BDTs (Friedman 2002) are ensemble methods built up of a multitude of decision trees (e.g., Morgan & Sonquist 1963; Breiman et al. 1984).

Decision trees construct a model to map input features to output classes using a series of decision rules (Ball & Brunner 2010). In our case, the decisions are based on whether or not a given feature is in a particular range. The tree is trained by recursively selecting which feature and boundary gives the

highest information gain in terms of classification. The problem with decision trees is that the trees created are often complex and do not generalize well beyond the training set, leading to over-fitting. They are also sensitive to small changes in the data, leading to large variance in predicted classes.

These problems can be overcome by combining decision trees in an ensemble method. Ensemble learning creates a multitude of decision trees on different subsets of data and averages the resulting classifications. There are two main approaches of combining classifiers: boosting and bagging. Random forests are created by bagging, which selects subsets of data with replacement (randomly in the case of random forests) on which to perform the classification. Boosting repeatedly fits the same data set but with each iteration, it increases the weights of incorrectly classified objects, meaning subsequent classifiers focus on difficult cases. We found BDTs (using the AdaBoost algorithm (Freund & Schapire 1997)) gave the same performance as bagging and were marginally faster in this case.

Probabilities are straightforward for estimating from decision trees: the probability of belonging to a given class is simply proportional to the fraction of trained objects in that class on the corresponding leaf of the tree. With an ensemble method, the probabilities from each of the decision trees in the ensemble is averaged to produce a final probability. BDTs are in general excellent classifiers (Dietterich 2000). Averaging over a large number of classifications makes them robust and unbiased estimators. The disadvantage is that they can be computationally intensive, although their computation time is still comparable to SVMs and ANNs. BDTs have been successfully applied to the problem of star-galaxy separation in images in Sevilla-Noarbe & Etayo-Sotos (2015).

4.2. Interpreting Machine Learning Results

There are a variety of metrics for measuring the performance of a machine learning algorithm. Every machine learning algorithm we consider returns a probability for each classification. It is common to produce a set of predicted classifications by simply selecting, for each object, the class associated with the highest probability. However, this is not the full picture, as the probability can be used as a threshold at which an object is considered to belong to a particular class. For instance, a very pure sample of type Ia supernovae can be obtained by demanding a probability of, say, 95% before considering the object a type Ia. This purity will always come at the cost of completeness, however. Many metrics common in machine learning literature (such as purity or completeness) will depend on this subjective threshold and can be optimized for a specific class or goal (for example, obtaining a pure sample of type Ia supernovae for cosmology). Here we use a more general metric, receiver operating characteristic (ROC) curves, to directly compare algorithms.

4.3. ROC Curves

A confusion matrix is useful for understanding most commonly used machine learning metrics. For a binary classification of two classes, positive and negative, the confusion matrix is shown in Table 4. An ideal classifier would produce only true positives without any contamination from false positives or losses from false negatives. In reality,

Table 4
Confusion Matrix for a Classification Problem with Two Classes: Positive (P) and Negative (N)

		True Class	
		P	N
Predicted class	P	True positive (TP)	False positive (FP)
	N	False negative (FN)	True negative (TN)

any classification problem has a trade-off between purity and completeness.

An excellent method of comparing machine learning algorithms and visualizing this trade-off is to use ROC curves (Green & Swets 1966; Hanley & McNeil 1982; Spackman 1989) (see Fawcett (2004) for an introductory tutorial). These allow one to get an overview of the algorithms without having to select an arbitrary probability threshold at which to consider an object as belonging to a particular class. ROC curves can only compare one class (assumed to be the desired class) against all others, but can easily be constructed for each class in turn. In this paper, all ROC curves are plotted assuming the binary classification of type Ia versus non-Ia.

ROC curves are constructed by comparing the true positive rate (TPR) against the false positive rate (FPR), as the probability threshold is varied. The TPR, also called recall, sensitivity, or completeness, is the ratio between the number of correctly classified positive objects and the total number of positive objects in the data set,

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (6)$$

The FPR, also referred to as the false alarm rate or 1-specificity, is the ratio between the number of objects incorrectly classified as positive and the total number of negative objects in the data set,

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \quad (7)$$

It is straightforward to see that a good classification algorithm will maximize the TPR (thus resulting in a more complete data set) while simultaneously minimizing the FPR (thus reducing contamination). The area-under-the-curve (AUC) of a ROC curve provides a straightforward way to directly compare classifiers. An AUC of 1 represents perfect classification, visually represented by a curve that reaches the top left corner. By contrast, a diagonal line would correspond to an AUC of 0.5, meaning the classifier does no better than random. In the machine learning literature, an AUC higher than 0.9 typically indicates an excellent classifier.

In Section 6 we use ROC curves to examine the differences between machine learning algorithms and our three different approaches to feature extraction.

4.4. Purity and Completeness

Commonly used metrics for classification include purity and completeness. These are only defined for classified objects, meaning some choice of probability threshold must be made. Usually, an object’s class is selected as that with the highest probability. A more pure subset can be obtained by applying a high threshold probability. For any subset chosen, it is useful to determine the purity and completeness to evaluate it.

We can use the confusion matrix in Table 4 to define purity (also known as precision) as

$$\text{purity} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (8)$$

and the completeness, which is equivalent to the TPR, as

$$\text{completeness} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (9)$$

4.5. Algorithm Parameter Choices

In supervised learning, the data are split into a training set, where each object has a known class label, and a test set, with unknown objects on which the classifier is run. Choice of training set is crucial, as a training set that is not representative of the entire data set can dramatically decrease a classifier’s performance (see Section 6). Additionally, almost all machine learning algorithms have hyperparameters that need to be optimized. In this work, we use five-fold cross-validation (Kohavi et al. 1995) to determine the values of the hyperparameters (see Table 5 in the Appendix for some example parameter values).

In general, k -fold cross-validation divides the training set into k equally sized “folds.” The classifier is trained on $k - 1$ folds, leaving the remaining fold as a validation set on which the chosen scoring metric (here we use AUC) is evaluated. The score is averaged over all k validation sets to produce a final score for the full training set. We use a grid search algorithm with cross-validation to find the hyperparameters for each algorithm that maximize AUC.

5. PHOTOMETRIC SUPERNOVA CLASSIFICATION PIPELINE

For each of the four feature sets (Section 3) and five machine learning algorithms (Section 4) considered, we perform the following procedure.

We split the data into a training and a test set, considering both a representative and a non-representative training set separately. The non-representative training set is the same as the one used in the SPCC. For a representative training set, we randomly select a set of objects the same size as the one in the challenge (1103 objects). We were able to repeat the representative case by randomly drawing several new training sets and running the full pipeline to see by how much results vary. We did not attempt to create a new non-representative training set with the same properties as the original.

In the SPCC, photometric host galaxy redshift information was provided, so we also investigate the importance of redshift information by running the full pipeline with all four feature extraction methods both with and without redshift. This is important because accurate photometric redshifts are difficult to obtain and the sensitivity of current and proposed classification techniques to the presence of redshift information has not yet been established. In the case of parametric models 1 and 2 and the wavelet decomposition, the redshift is simply added as an additional feature. For the SALT2 case, we fix the redshift in the model to the photometric redshift instead of allowing it to vary, thus better constraining the other parameters for type Ia’s.

Before running any machine learning algorithm, we rescale the features with a standard scaling. We scale the features by removing the mean (centering on zero) and scaling to unit

variance (dividing by the standard deviation). After rescaling, we run each of the machine learning algorithms outlined in Section 4 using cross-validation to determine the optimal hyperparameters. We then compare the results using ROC curves.

6. RESULTS

Figure 3 shows the ROC curves for a representative training set, both without and with redshift information included. From the ROC curves, it is clear that not all machine learning algorithms perform equally and indeed, in every case the BDTs outperform the other algorithms. These ROC curves also illustrate that if the fundamental assumptions of an algorithm are broken, it performs poorly. Naive Bayes performs reasonably well in most cases, but is barely better than random in the case of the wavelets, which are highly non-Gaussian in their distribution. We can also see that the SALT2 features and the wavelet features outperform the parametric models, implying that it is best to use either a highly model-dependent approach, making use of prior knowledge, or use a highly model-independent approach for supernova classification.

Figure 4 shows the Venn diagram for object classification for each of the four feature extraction methods, using BDTs. Objects were classified according to the class with highest probability. From this, one can see that type II supernovae are straightforward to classify, with all feature extraction methods achieving excellent accuracy. Type Ia's are somewhat more difficult, although the SALT2, Model 1, and wavelets methods agree well. There is little agreement between classifiers when it comes to the type Ibc's. This is unsurprising, as Ibc light curves are often similar to Ia's and there are relatively few Ibc's in the data set. The SALT2 method does significantly better than the others at identifying Ibc's.

As stated in Section 4.2, the probability threshold at which you select an object's class is arbitrary, meaning that the given class for each supernova will change depending on the threshold used. It is well known that commonly used metrics such as accuracy, purity, and completeness are dependent on the threshold used and do not give a complete picture (hence why we advocate the use of the AUC as a comparison metric). However, as an example of the values of these metrics for the SPCC data set, we can vary the threshold probability and insist on a 90% pure type Ia data set. Then the corresponding completeness is 85%, 41%, 7.8%, 83% for SALT 2, Model 1, Model 2, and wavelets respectively. However, if a more complete data set is more important, we find that demanding a 90% completeness results in a purity of 87%, 77%, 75%, 87% respectively.

6.1. The Effect of Redshift

The effect of redshift is summarized succinctly in Figure 5, comparing all feature extraction methods and machine learning algorithms. Figure 5 shows that providing redshift information plays an important role for SALT2 features, mildly improves the results of the parametric features, and is fairly unimportant to the wavelet features. The notable exception to this is when considering the best machine learning algorithm, BDTs. With this algorithm, including redshift information is not essential, as the algorithm is capable of differentiating between classes without redshift information.

This raises the interesting point that with an algorithm with poor performance, such as KNN, redshift information is crucial to classify well with the SALT2 model. However, a better machine learning algorithm eliminates the need for redshift information (under the assumption of a representative training set). This is important since obtaining reliable redshifts for every object in future surveys will be challenging. We have shown that it will be possible to obtain a relatively pure subsample of supernovae without redshift information, on which to focus follow-up observations.

It is also interesting to note that the wavelet method, which uses no prior knowledge of supernovae, is able to perform as well as the current state-of-the-art classification technique, based on supernova templates. This is promising, since the simulated data set used here was constructed using largely the same known templates, so it is expected that the SALT2 model should perform well. However, the wavelet approach requires no previous knowledge and thus should perform well with new data sets. This also suggests that wavelet approaches to classification are likely to be useful for broader transient classification, as also noted in Varughese et al. (2015).

6.2. The Effect of Representativeness

We also investigate the effect of using a representative or non-representative training set. It is well known that most machine learning algorithms underperform when the training set is in any way not representative of the distribution of features. The training set from the SPCC was known to be biased in the way spectroscopic follow-up data sets usually are, in that only the brightest objects tend to be followed up due to telescope time constraints and the difficulty in obtaining good spectra. It is not surprising then that, as shown in Figure 6, all feature extraction methods and all algorithms perform significantly worse with the non-representative training set. It should be noted that redshift information was not provided here, which only mitigates the effect of non-representativeness in the case of the SALT2 model. This is because the SALT2 model is already based on a large training set of supernovae and thus, if redshift information is given, will produce similar sets of features.

The clear conclusion is that the more representative the training set, the better any classification method will perform. As the training set was only around 5% of the size of the full data set, we would argue that if spectroscopic follow-up time is limited, it is rather better to obtain fewer, fainter objects of all types, than to get a larger sample of brighter objects (as noted in Richards et al. 2012).

However, the effect of non-representativeness, also called the "domain adaptation problem" or "sample selection bias," can be mitigated using modern techniques in transductive transfer learning (Pan & Yang 2010). These include reweighting techniques to reweight the data in the test set to more closely match that of the training set and techniques to learn new sets of features that minimize the difference between the two data sets. Transfer learning has been successfully used in astronomy in Sasdelli et al. (2015). An in-depth study of transfer learning is beyond the scope of this paper but will likely prove useful for dealing with non-representativeness in the training sets of future supernova surveys.

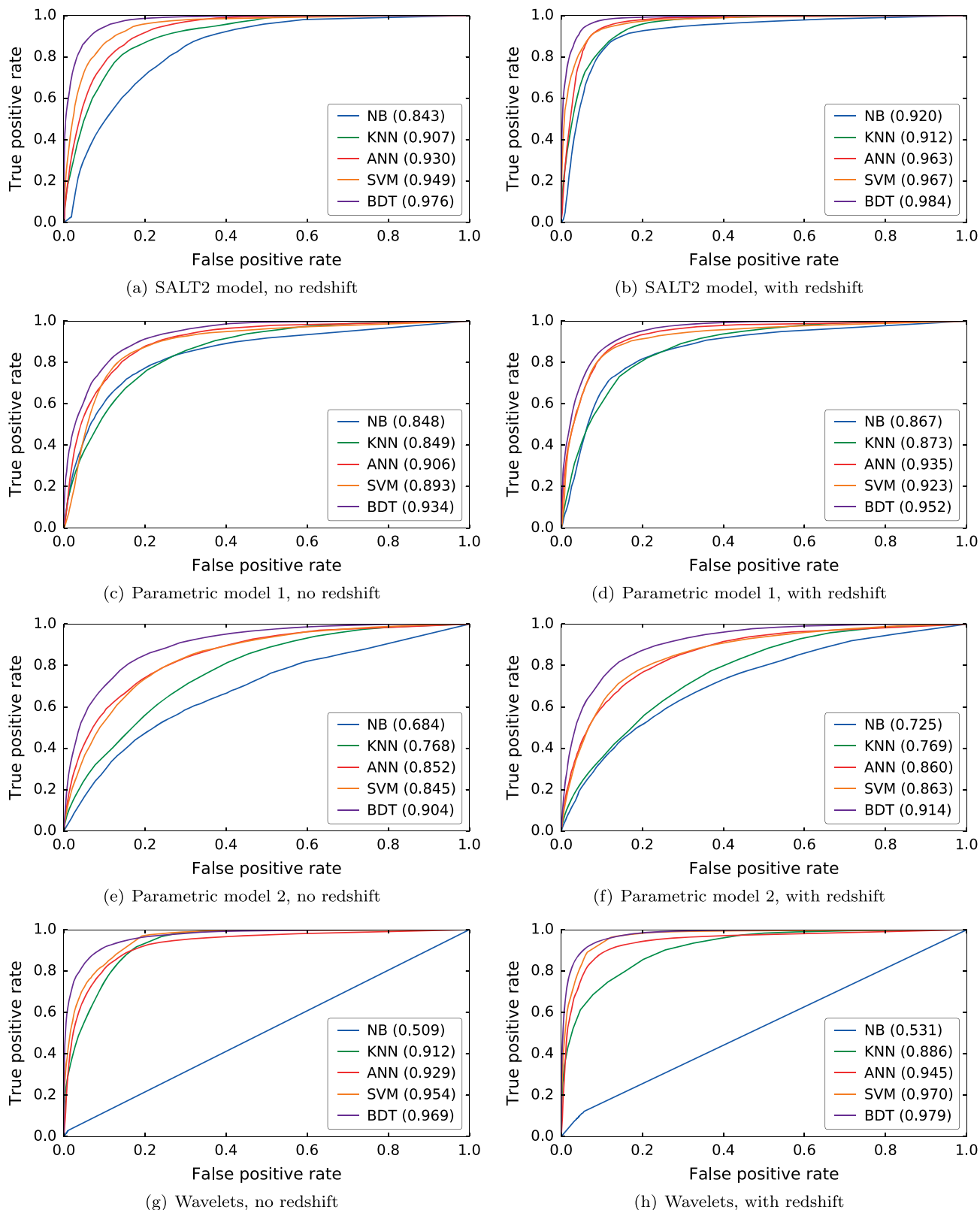


Figure 3. Receiver operating characteristic (ROC) curves for the various feature extraction methods, using a representative training set without (first column) and with (second column) redshift data included. Each curve represents a different machine learning algorithm (as outlined in Section 4) with the area-under-curve (AUC) score in brackets. An AUC of 1 indicates perfect classification and corresponds to a curve close to the top left corner. It is clear that boosted decision trees (BDT), support vector machines (SVMs), and artificial neural networks (ANNs) are the best-performing algorithms, superior to k -nearest neighbors (KNN) and naive Bayes (NB). The SALT2 and wavelet feature sets provide the best classification.

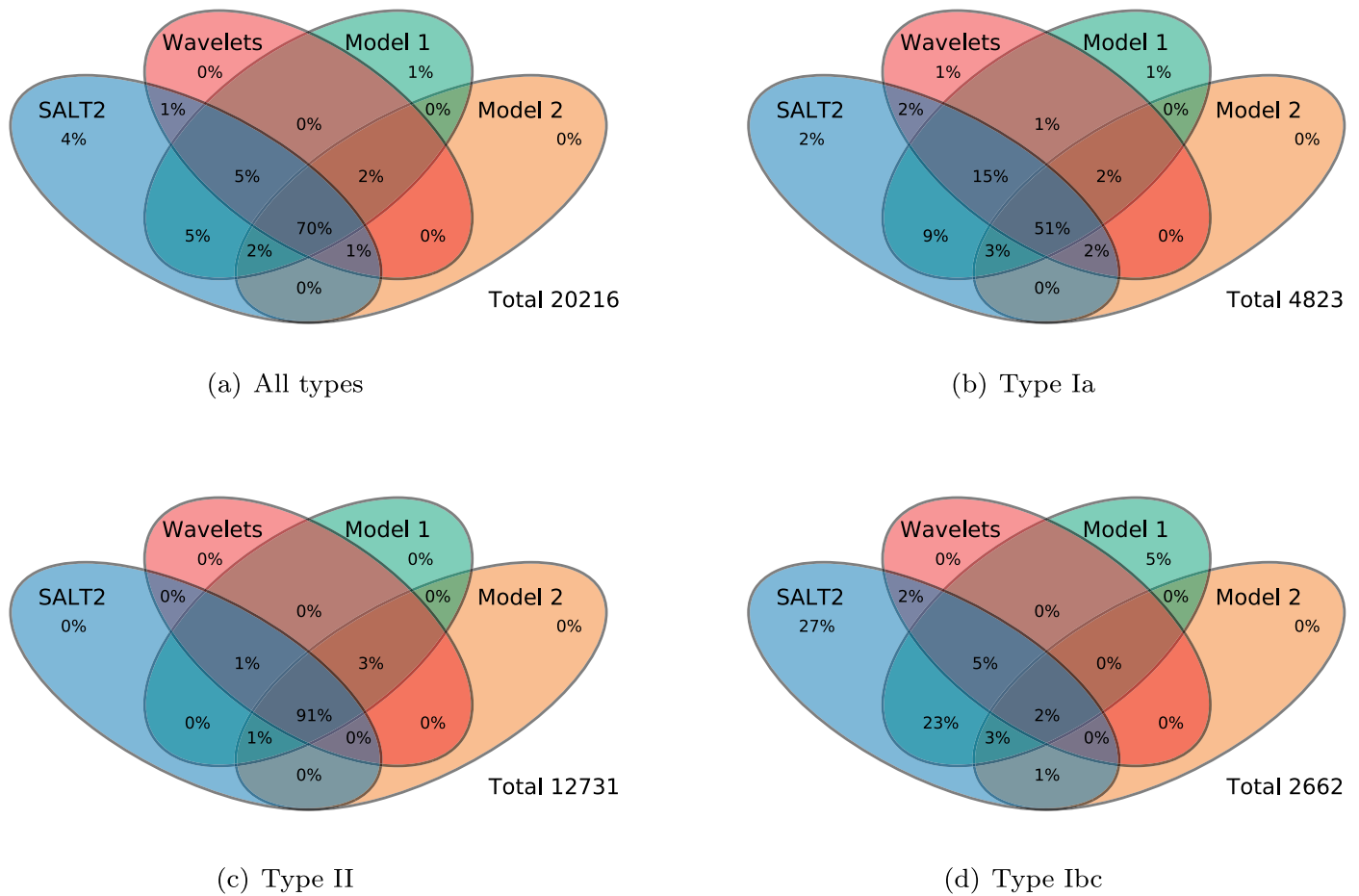


Figure 4. Venn diagrams showing the number of correctly classified objects for each feature extraction method using boosted decision trees, with each area indicating the percentage of objects correctly classified. This is repeated for all objects and for each type of supernova individually. It is clear that, for example, type II supernovae are easy to classify, with all feature extraction methods agreeing on the types and achieving excellent accuracy, while type Ibc’s are much more difficult to classify.

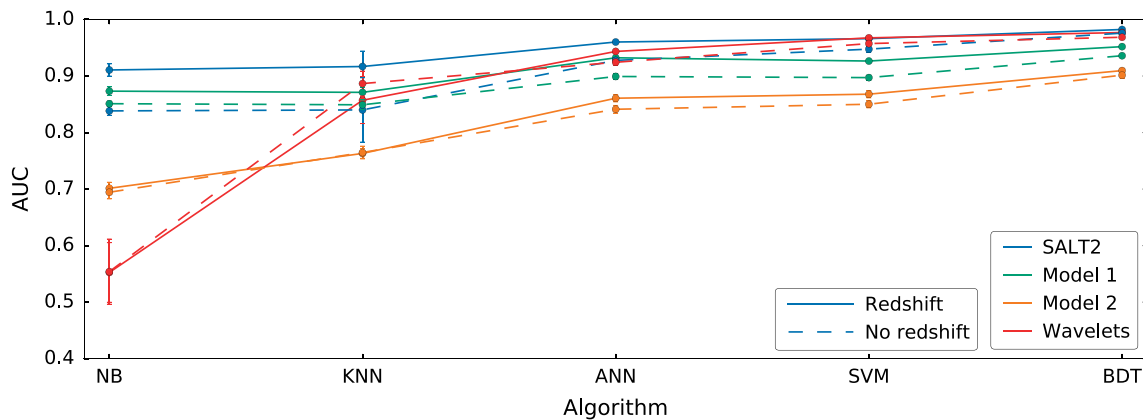


Figure 5. Area-under-curve (AUC) as a function of feature set and machine learning algorithm. Higher AUC scores correspond to better classification. Solid lines indicate that redshift was included as a feature, while dashed lines represent feature sets without redshift information. The five machine learning algorithms considered are on the x -axis, while each color corresponds to a different feature extraction method. Error bars are from the standard deviation of the AUC from ten runs with randomized training sets. When using BDT, the SALT2, and wavelet features are able to classify equally well with or without redshift. By comparison, the SALT2 features are highly sensitive to redshift when using, for example, the NB or KNN algorithm.

6.3. Relative Feature Importance

It is possible to investigate the relative importance of each feature in a feature set using BDTs using the “gini importance” (also known as “mean decrease impurity”), described in Breiman et al. (1984). This is illustrated for each feature set

in Figure 7, both with and without redshift. While there are many other methods to determine the relevance of each feature (see Li et al. 2016 for a summary), we only use this simple measure of feature relevance to gain insight into the features and use the full classification pipeline to evaluate the

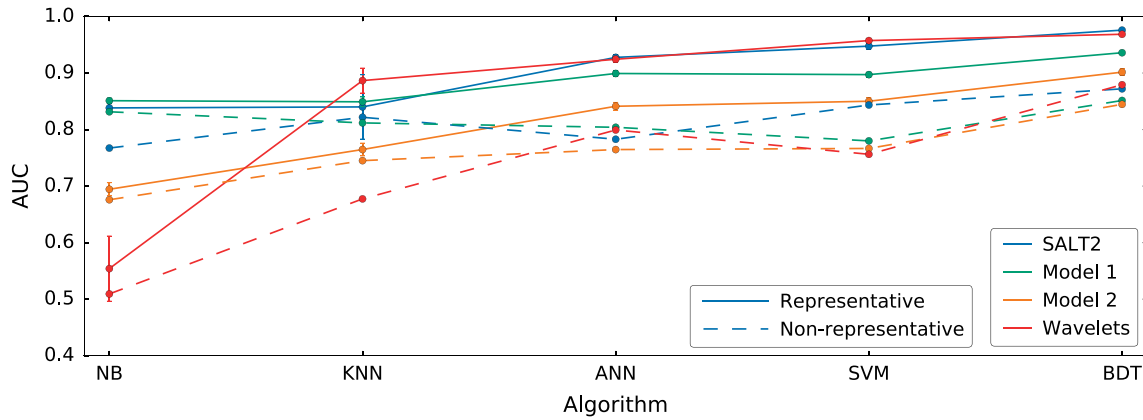


Figure 6. Area-under-curve (AUC) as a function of feature set and machine learning algorithm. Higher AUC scores correspond to better classification. Solid lines indicate that a representative training set was used, while dashed lines indicate the use of the non-representative training set of Kessler et al. (2010a, 2010b). The five machine learning algorithms considered are on the x -axis, while each color corresponds to a different feature extraction method. Error bars for the representative training set case are the standard deviation from 10 runs with randomized training sets. It is clear that any feature extraction method or machine learning algorithm used is highly sensitive to non-representativeness in the training set.

effectiveness of our chosen features, rather than performing initial feature selection based on relevance.

For the SALT2 model, the most important features are the shape and color parameters. Note that counterintuitively, the importance of redshift goes *down* when external redshift information is added. By adding redshift information, the Ia's in the data set are better fit by the SALT2 model as expected, resulting in accurate estimates of the stretch and color parameters. On the other hand, with the redshift fixed the best fits for the non-Ia's result in unphysical stretch and color parameters. Because the distributions of these important features now differ much more between the non-Ia's and Ia's, they increase in importance when classifying. Because all the importances must add up to one, this necessarily dictates a decrease in the importance of other features, including the redshift.

There are a few features that are more important for the parametric models, but none are notable. Interestingly, it is the third and fourth principal components for the wavelet features that are most important, while all others have fairly equivalent importance.

7. CONCLUSIONS

Classification of photometric supernovae is an important problem in light of current and upcoming surveys. We have compared several different approaches to this problem, some based on current techniques and some new. We have shown that in terms of feature extraction, both SALT2 model fits and wavelet decomposition outperform parametric fits. Out of the five representative machine learning algorithms we used, BDTs performed the best, followed by support vector machines and artificial neural networks. The SALT2 fits and the wavelet approach both achieved an AUC of 0.98, representing excellent classification. The SALT2 and wavelet methods can both produce a data set that is about 90% pure and 84% complete, although the purity can be increased at the cost of completeness.

Importantly, we found that with a powerful ensemble algorithm like BDTs and a representative training set, redshift information does not improve classification performance, meaning a data set can be well classified without any redshift

information at all. This is extremely useful for current and upcoming surveys where reliable host galaxy photometric redshifts may be difficult to obtain.

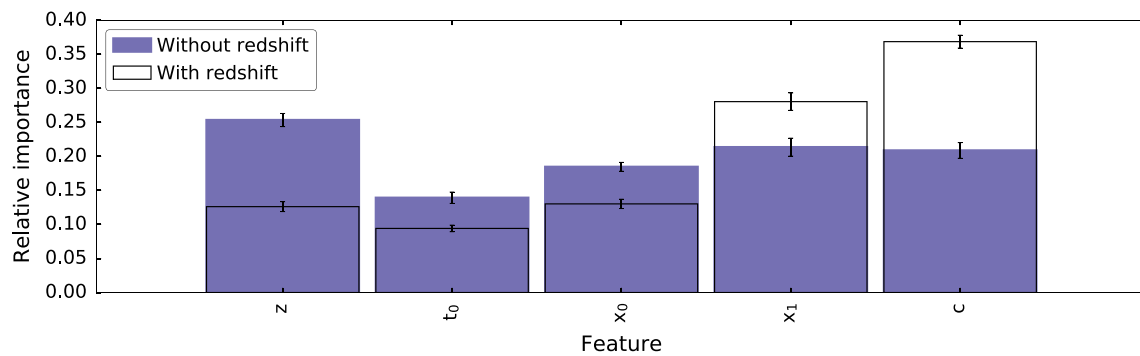
Although redshift information is needed for cosmology, this means a well-classified subset of supernovae will be available without any redshift information, with which follow-up studies can be done if necessary. This also implies uncertainties in classification will be completely uncorrelated with uncertainties in redshift estimates. It also means there will be a large sample of robustly classified supernovae available for core collapse research.

On the other hand, with all algorithms and feature extraction methods considered, a representative training set is crucial for good performance. The training set used was very small (around 5% of the data set), suggesting that a spectroscopic follow-up program should rather focus on observing fainter objects of all types rather than obtaining a larger training set that is not representative.

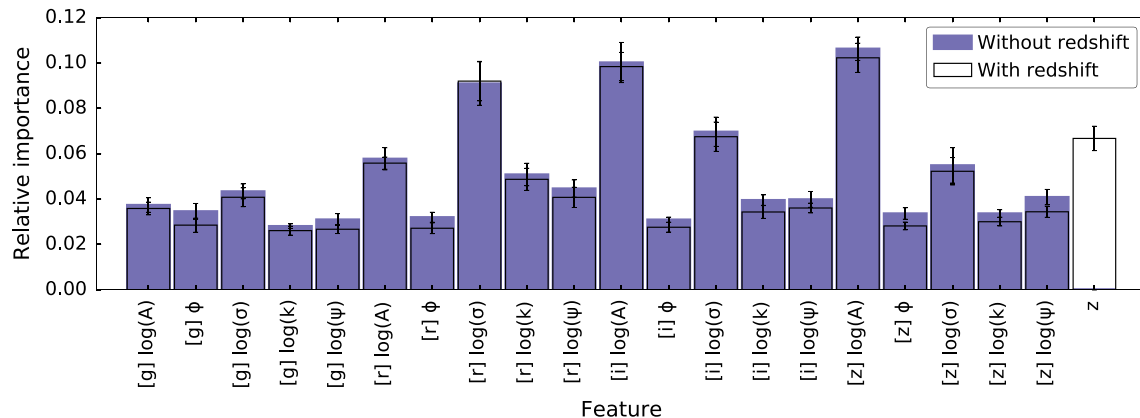
There are many further improvements one can consider in both the feature extraction methods and the machine learning algorithms. We have provided a framework here for directly and easily comparing these and any future approaches to the photometric supernova classification problem.

In future work we will apply this pipeline to SDSS data and compare against current photometric classifications. Additionally, the pipeline developed here will be crucial for investigating how changes in observing strategy affect classification for LSST.

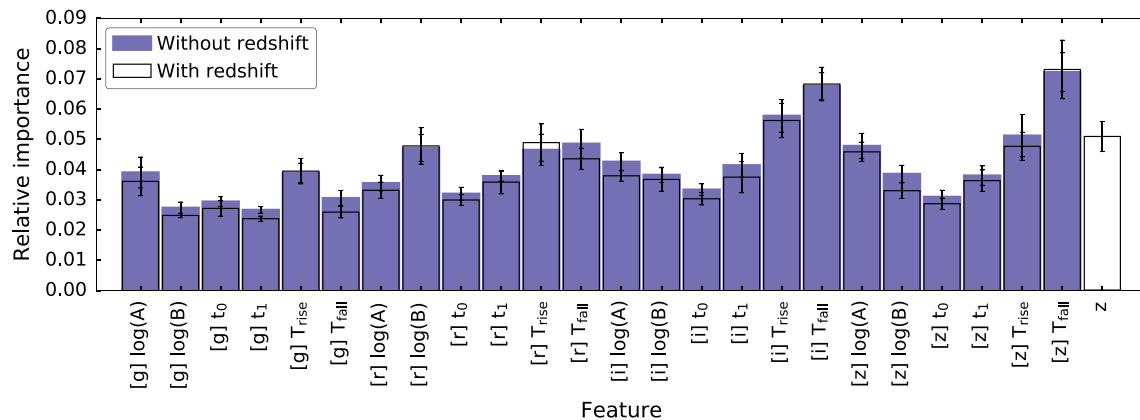
We thank Michael Betancourt, Rahul Biswas, Saurabh Jha, Bob Nichol, Stephen Smartt, and Mark Sullivan for useful discussions. We acknowledge support from STFC for UK participation in LSST through grant ST/N00258X/1 and travel support provided by STFC for UK participation in LSST through grants ST/L00660X/1 and ST/M00015X/1. H.V.P. and M.L. were supported by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 306478-CosmicDawn. O.L. and M.L. acknowledge a European Research Council Advanced Grant FP7/291329 (TESTDE) and support from STFC.



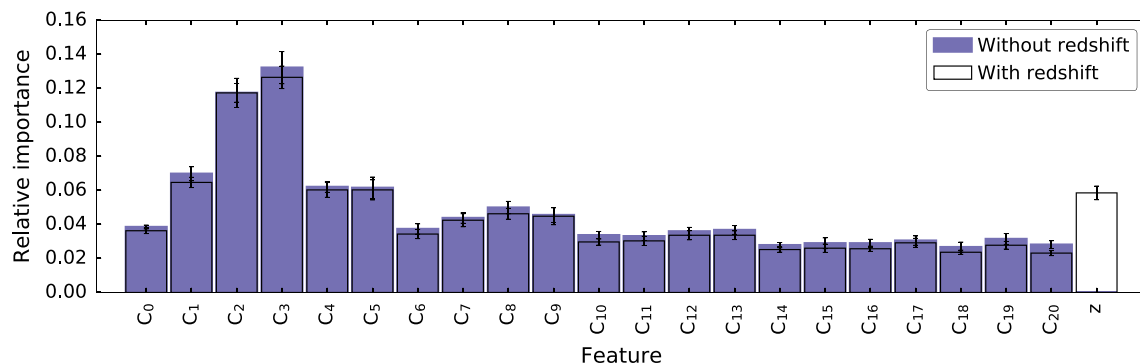
(a) SALT2



(b) Parametric model 1



(c) Parametric model 2



(d) Wavelets

Figure 7. Relative importance of individual features with and without redshift information, as obtained from the boosted decision trees algorithm. The SALT2 model is able to estimate the redshift, which is thus still used as a feature even when no redshift information is provided (and a broad, flat prior is used for the fitting). For parametric models 1 and 2, the filter is given in square brackets before the parameter. For the wavelets, the 20 principal components used are labeled C₁ to C₂₀.

Table 5

Hyperparameters for Each Machine Learning Algorithm and Each Feature Set for the Case Where Redshift is Included and a Representative Data set is Used

	SALT2	Model 1	Model 2	Wavelets
NB
KNN	n_neighbors = 141	n_neighbors = 101	n_neighbors = 46	n_neighbors = 121
SVM	C = 0.56, $\gamma = 1.78$	C = 1780, $\gamma = 0.0032$	C = 31.6, $\gamma = 0.0032$	C = 0.56, $\gamma = 1.78$
ANN	neurons = 110	neurons = 110	neurons = 115	neurons = 80
BDT	n_estimators = 40	n_estimators = 75	n_estimators = 65	n_estimators = 70

APPENDIX HYPERPARAMETER SELECTION

Table 5 shows the values of the hyperparameters for each machine learning algorithm for an example case. These hyperparameters are selected using cross-validation (as described in Section 4) to maximize the AUC. These values vary slightly depending on the training set used.

REFERENCES

- Aizerman, M. 1964, *Automation and Remote Control*, 25, 821
- Altman, N. S. 1992, *The American Statistician*, 46, 175
- Ball, N. M., & Brunner, R. J. 2010, *IJMPD*, 19, 1049
- Ballard, D. H. 1987, in Proc. of the 6th International Conference on Artificial Intelligence, 279
- Barbary, K. 2014, snocosmo, v0.4.2, Zenodo, doi:10.5281/zenodo.11938
- Bernstein, J. P., Kessler, R., Kuhlmann, S., et al. 2012, *ApJ*, 753, 152
- Betoule, M., Kessler, R., Guy, J., et al. 2014, *A&A*, 568, A22
- Bloom, J. S., & Richards, J. W. 2012, *Data Mining and Machine Learning in Time-Domain Discovery and Classification* (Boca Raton, FL: CRC Press)
- Breiman, L. 2001, *Machine Learning*, 45, 5
- Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. 1984, *Classification and Regression Trees* (Boca Raton, FL: CRC Press)
- Buchner, J., Georgakakis, A., Nandra, K., et al. 2014, *A&A*, 564, A125
- Caruana, R., & Niculescu-Mizil, A. 2006, in Proc. 23rd Int. Conf. on Machine Learning, ICML '06 (New York: ACM), 161
- Comon, P. 1994, *SigPr*, 36, 287
- Cortes, C., & Vapnik, V. 1995, *Machine Learning*, 20, 273
- Dark Energy Survey Collaboration 2005, arXiv:astro-ph/0510346
- Dark Energy Survey Collaboration, Abbott, T., Abdalla, F. B., et al. 2016, *MNRAS*, 460, 1270
- Daubechies, I. 1988, *Communications on Pure and Applied Mathematics*, 41, 909
- Dietterich, T. G. 2000, *Multiple Classifier Systems* (Berlin: Springer)
- du Buisson, L., Sivanandam, N., Bassett, B. A., & Smith, M. 2015, *MNRAS*, 454, 2026
- Fawcett, T. 2004, *Machine Learning*, 31, 1
- Feroz, F., & Hobson, M. P. 2008, *MNRAS*, 384, 449
- Feroz, F., Hobson, M. P., & Bridges, M. 2009, *MNRAS*, 398, 1601
- Feroz, F., Hobson, M. P., Cameron, E., & Pettitt, A. N. 2013, arXiv:1306.2144
- Freund, Y., & Schapire, R. E. 1997, *Journal of Computer and System Sciences*, 55, 119
- Friedman, J. H. 2002, *Computational Statistics & Data Analysis*, 38, 367
- Frieman, J. A., Bassett, B., Becker, A., et al. 2008, *AJ*, 135, 338
- Green, D., & Swets, J. 1966, *Society*, 1, 521
- Guy, J., Astier, P., Baumont, S., et al. 2007, *A&A*, 466, 11
- Hanley, J. A., & McNeil, B. J. 1982, *Radiology*, 143, 29
- Hlozek, R., Kunz, M., Bassett, B., et al. 2012, *ApJ*, 752, 79
- Holschneider, M., Kronland-Martinet, R., Morlet, J., & Tchamitchian, P. 1989, *Wavelets* (Berlin: Springer)
- Hotelling, H. 1933, *Journal of Educational Psychology*, 24, 417
- Ishida, E. E. O., & de Souza, R. S. 2013, *MNRAS*, 430, 509
- Jeffrey, W., & Rosner, R. 1986, *ApJ*, 310, 473
- Jha, S., Riess, A. G., & Kirshner, R. P. 2007, *ApJ*, 659, 122
- Karpenka, N. V., Feroz, F., & Hobson, M. P. 2013, *MNRAS*, 429, 1278
- Kessler, R., Bassett, B., Belov, P., et al. 2010a, *PASP*, 122, 1415
- Kessler, R., Conley, A., Jha, S., & Kuhlmann, S. 2010b, arXiv:1001.5210
- Kingsbury, N. 2001, *Applied and Computational Harmonic Analysis*, 10, 234
- Knights, M., Bassett, B. A., Varughese, M., et al. 2013, *JCAP*, 1, 039
- Kohavi, R., et al. 1995, in Proc. of the International Joint Conference on Artificial Intelligence, Vol. 14, 1137
- Kramer, M. A. 1991, *AICHe*, 37, 233
- Küglér, S. D., Polsterer, K., & Hoecker, M. 2015, *A&A*, 576, A132
- Kunz, M., Bassett, B., & Hlozek, R. 2007, *PhRvD*, 75, 103508
- Kurcz, A., Bilicki, M., Solarz, A., et al. 2016, arXiv:1604.04229
- Lahav, O., Naim, A., Buta, R. J., et al. 1995, *Sci*, 267, 859
- LeCun, Y., Bengio, Y., & Hinton, G. 2015, *Natur*, 521, 436
- Li, J., Cheng, K., Wang, S., et al. 2016, arXiv:1601.07996
- Lo, E. H. S., Pickering, M. R., Frater, M. R., & Arnold, J. F. 2009, in Proc. ACM Int. Conf. on Image and Video Retrieval, CIVR '09 (New York: ACM), 5:1
- LSST Science Collaboration 2009, arXiv:0912.0201
- MacKay, D. J. 2003, *Information Theory, Inference and Learning Algorithms* (Cambridge: Cambridge Univ. Press)
- Mallat, S. G. 2009, *A Wavelet Tour of Signal Processing* (3rd ed.; Burlington: Academic)
- Morgan, J. N., & Sonquist, J. A. 1963, *Journal of the American Statistical Association*, 58, 415
- Newling, J., Varughese, M., Bassett, B., et al. 2011, *MNRAS*, 414, 1987
- Newling, J., Bassett, B., Hlozek, R., et al. 2012, *MNRAS*, 421, 913
- Oszkiewicz, D. A., Kwiatkowski, T., Tomov, T., et al. 2014, *A&A*, 572, A29
- Pan, S. J., & Yang, Q. 2010, *IEEE Transactions on Knowledge and Data Engineering*, 22, 1345
- Pearson, K. 1901, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2, 559
- Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, *Journal of Machine Learning Research*, 12, 2825
- Platt, J., et al. 1999, in *Advances in Large Margin Classifiers*, Vol. 10 (Cambridge, MA: MIT Press), 61
- Richards, J. W., Homrighausen, D., Freeman, P. E., Schafer, C. M., & Poznanski, D. 2012, *MNRAS*, 419, 1121
- Rubin, D., Aldering, G., Barbary, K., et al. 2015, *ApJ*, 813, 137
- Sadeh, I., Abdalla, F. B., & Lahav, O. 2015, arXiv:1507.00490
- Sako, M., Bassett, B., Becker, A., et al. 2008, *AJ*, 135, 348
- Sako, M., Bassett, B., Becker, A. C., et al. 2014, arXiv:1401.3317
- Sako, M., Bassett, B., Connolly, B., et al. 2011, *ApJ*, 738, 162
- Sammon, J. W. 1969, *ITCmp*, 18, 401
- Sanders, N. E., Soderberg, A. M., Gezari, S., et al. 2015, *ApJ*, 799, 208
- Sasdeli, M., Ishida, E. E. O., Vilalta, R., et al. 2015, arXiv:1512.06810
- Schmidhuber, J. 2014, arXiv:1404.7828
- Seikel, M., Clarkson, C., & Smith, M. 2012, *JCAP*, 6, 036
- Sevilla-Noarbe, I., & Etayo-Sotos, P. 2015, *Astro & Computing*, 11, 64
- Spackman, K. A. 1989, in Proc. Sixth Int. Workshop on Machine Learning (Burlington, MA: Morgan Kaufmann Publishers Inc.), 160
- Valens, C. (ed.) 1999
- Van der Maaten, L., & Hinton, G. 2008, *Journal of Machine Learning Research*, 9, 85
- Varughese, M. M., von Sachs, R., Stephanou, M., & Bassett, B. A. 2015, *MNRAS*, 453, 2848
- Werbos, P. 1974
- Wright, D. E., Smartt, S. J., Smith, K. W., et al. 2015, *MNRAS*, 449, 451
- Wright, E. L., Eisenhardt, P. R., Mainzer, A. K., et al. 2010, *AJ*, 140, 1868
- Xiong, H., Zhang, T., & Moon, Y. S. 2000, *ITIP*, 9, 2100
- Zhang, H. 2004, *AA*, 1, 3