Machine learning classification of SDSS transient survey images

L. du Buisson,,1,2★ N. Sivanandam,2 Bruce A. Bassett,1,2,3 and M. Smith4,5

1Department of Mathematics and Applied Mathematics, University of Cape Town, Cross Campus Rd, Rondebosch 7700, South Africa
2African Institute for Mathematical Sciences, 6–8 Melrose Rd, Muizenberg 7945, South Africa
3South African Astronomical Observatory, Observatory Rd, Observatory 7925, South Africa
4Department of Physics, University of the Western Cape, Cape Town 7535, South Africa
5School of Physics and Astronomy, University of Southampton, Southampton SO17 1BJ, UK

Accepted 2015 September 2. Received 2015 August 30; in original form 2014 July 26

ABSTRACT

We show that multiple machine learning algorithms can match human performance in classifying transient imaging data from the Sloan Digital Sky Survey (SDSS) supernova survey into real objects and artefacts. This is a first step in any transient science pipeline and is currently still done by humans, but future surveys such as the Large Synoptic Survey Telescope (LSST) will necessitate fully machine-enabled solutions. Using features trained from eigenimage analysis (principal component analysis, PCA) of single-epoch g, r and i difference images, we can reach a completeness (recall) of 96 per cent, while only incorrectly classifying at most 18 per cent of artefacts as real objects, corresponding to a precision (purity) of 84 per cent. In general, random forests performed best, followed by the k-nearest neighbour and the SkyNet artificial neural net algorithms, compared to other methods such as naive Bayes and kernel support vector machine. Our results show that PCA-based machine learning can match human success levels and can naturally be extended by including multiple epochs of data, transient colours and host galaxy information which should allow for significant further improvements, especially at low signal-to-noise.


1 INTRODUCTION

The quest to answer the deepest open questions about the cosmos has pushed astronomers and cosmologists to sample larger and larger volumes of the Universe. Current and next generation of surveys, such as Gaia,1 the Dark Energy Survey (DES),2 the Large Synoptic Survey Telescope (LSST)3 and the SKA,4 will usher in an era of exascale astronomy requiring new machine learning and statistical inference tools.

The LSST, for example, will image the night sky with such depth and frequency that upwards of a million Type Ia supernova (SN Ia) candidates detected over a decade of operations (LSST Science Collaboration 2009). This will swamp existing follow-up capabilities, pushing us into the era of photometric transient identification trained on small spectroscopic subsets (Kunz, Bassett & Hlozek 2007; Gong, Cooray & Chen 2010; Kessler et al. 2010; Newling et al. 2012; Hlozek et al. 2012; Campbell et al. 2013; Knights et al. 2013). Such techniques will always lead to a small set of misidentifications and the danger is that the resulting contamination, unless dealt with in a sophisticated way, will lead to biased results.

However, long before one reaches the final scientific analysis, the data deluge create challenges in the analysis pipeline. For example, difference images are created by subtracting a reference image from the most recent image of a given part of the sky. In the ideal case, this will leave a pure noise image unless a real transient such as an SN, asteroid or variable star exists in the image. In reality, there are inevitable artefacts that occur because of instrumental effects: diffraction spikes, CCD saturation and bleeding, registration errors and the like.

We thus need to disentangle the potential objects of interest from the artefacts. Historically, this sorting and classification into real objects and artefacts has been done by astronomers scanning the images as soon as possible after the images have been taken. In the case of the Sloan Digital Sky Survey (SDSS) SN survey, this typically led to hundreds or thousands of images being scanned each night, a tedious job. Recently, it has been shown that this hand scanning can be done effectively by crowdsourcing (Smith et al. 2011); the public correctly identified 93 per cent of the spectroscopically confirmed SNe.

However, using humans to do this classification makes it very difficult to quantify the biases that arise from the subtly different algorithms and internal decision trees in each human scanner’s brain.

★ E-mail: lisedubuisson@gmail.com
1 http://sci.esa.int/gaia/
2 http://www.darkenergysurvey.org/
3 http://www.lsst.org
In addition, the effective decision tree changes with time depending on the mood and tiredness of the hand scanner which obviously cannot be characterized systematically. This human bias was partially dealt with in the SDSS SN survey by injecting fake SNe into the pipeline yielding an average detection efficiency for each scanner, but it is clearly a fundamental limitation of human hand scanning which is even worse for crowd-sourced classifications. Apart from these biases, hand scanning will not be an option for LSST due to the millions of images that will need to be scanned each night.

Replacing humans with machine learning for this transient-artefact classification therefore represents an important frontier in achieving the goals of future transient surveys. Existing work includes the pioneering work of the SNfactory (Romano, Aragon & Ding 2006; Bailey et al. 2007) where features included the position, shape and full width at half-maximum of the main object, as well as distance to the nearest object in the reference image. Also of interest are the work done by the Palomar Transient Factory, where the focus falls on distinguishing between transients and variable stars (Bloom et al. 2012), and the discovery of variability in time-domain imaging surveys (Brink et al. 2013) where classifiers output probabilistic statements about the degree to which newly observed sources are astrophysically relevant sources of variable brightness. Recently, Goldstein et al. (2015) have presented a new random forest (RF) implementation for artefact/transient classification in the DES SN pipeline. Their feature set consisted of 38 features, most of which were based on analogues from Bloom et al. (2012) and Brink et al. (2013). Some of their new features were among the most important features for their classification – including flux and point spread function-based parameters that were obtained from their implementation of SExtractor (Bertin & Arnouts 1996).

In contrast, we use SDSS data and derive our features from principal component analysis (PCA) of the Sloan g, r and i difference images. We compare a number of different potential machine learning algorithms such as k-nearest neighbours (KNN), artificial neural network (ANN; SkyNet – Graff et al. 2014), naïve Bayes (NB) and support vector machine (SVM) and show that it is possible to achieve human levels of classification completeness/recall with limited degradation in purity.

In Section 2 we describe the SDSS data used; the testing and performance measures in Section 3 and the feature extraction and machine learning algorithms we employed are described in Sections 4 and 5. Our results are discussed in Section 6.

2 DATA

Our data are drawn from the second and third years of the SDSS-II SN survey (Frieman et al. 2008; Sako et al. 2014) which operated in drift scan mode for three months each year from 2005 to 2007, alternately observing each of the two strips making up the approximately 300 deg2 equatorial Stripe 82, weather permitting. In practice, the mean cadence on any patch of sky was about four nights.

The transient detection algorithm consists of image subtraction of the search image from a historical image of the same region of the sky in the Sloan colour bands g, r and i. The difference image is flagged as a candidate for human scanning if it passes certain threshold cuts. However, this process was imperfect and led to a large number of artefacts that constituted 70 per cent of the candidates or more.

A team of about 20 human hand scanners using both the search and difference images classified each candidate object into one of 10 distinguishable classes: artefacts, dipoles, saturated stars, moving objects, variables, transients, SN Other, SN Bronze, SN Silver and SN Gold (Frieman et al. 2008). A description of each of these classes, as given by Bassett, Richmond & Frieman (2005), can be found in Table 1, and Table A1 shows the visual appearance of each of these 10 classes. For our classification purposes, we regroup these original classes into three new visual classes, based on the observation that many of the classes have very similar visual appearances. These three classes are: real objects, artefacts and dipoles/saturated, illustrated in Fig. 1. Real objects have point-like residuals (convolved with the seeing from the atmosphere), artefacts have diffraction spike-like residuals while the dipoles/saturated class have residuals that are often close to point-like, but typically have negative flux in part of the difference image arising from registration errors or saturated CCD effects. Fig. 1 shows the three visual classes clearly: Fig. 1(a) serves as an example of what high-quality images, with signal-to-noise ratios (SNRs) above 40, look like, while Fig. 1(b) shows images with SNR below 20, a threshold that represents roughly 80 per cent of the objects in our data set. Table 1 shows which of the original classes correspond to each of the three visual classes. This visual classification was used for carrying out single-class PCA for feature extraction, as is discussed in Section 4. We are ultimately concerned with whether we can correctly predict whether an object is real or not-real (artefact or dipole/saturated), however, and this is therefore the main classification that will be recognized by our classifiers. To see which classes correspond to real objects and which correspond to not-real objects, see Table 1.

The 2006 and 2007 seasons of the SDSS-II SN survey introduced an automated software filter, autoscanner, to identify all objects detected in more than one epoch or bright objects detected for the first time and through statistical techniques, filter out non-SN objects. This algorithm significantly reduced the number of moving objects and diffraction spikes, and eliminated a significant fraction of objects with long-term variability by cross-matching with a veto catalogue. A detailed description of the autoscanner software is given in Sako et al. (2008).

As this algorithm was implemented after the 2005 season, our analysis considers only objects detected in the 2006 and 2007 seasons of the SDSS-II SN survey. The data set contains 27 480 objects, each consisting of three 51 × 51-pixel difference images (one for each of the g, r and i colour bands). Of this, 15 521 are real objects and 11 959 are not-real. For comparison, the 2005 season of SDSS-II detected 141 697 objects alone. The classifications used in this analysis are based on the classifications done by the SDSS hand scanners. About 2500 fake SNe were also inserted into the imaging pipeline in order to provide quality control and to characterize the selection function. Our data set is the full set including fakes to allow us to compare with the human performance on the fakes.

3 TESTING AND PERFORMANCE MEASURES

3.1 Testing

Our basic testing protocol was to withhold 25 per cent of the data (hereafter referred to as the ‘test set’) for method comparison after
our various learners have been trained, cross-validated (if necessary) and undergone preliminary testing using the other 75 per cent of the data. This 75/25 split was done prior to commencing building our various learning machines and the test set was sequestered until we were ready to perform our final testing.

Some of the classifiers have parameters that can be optimized using cross-validation. In general, 30 per cent of the remaining 75 per cent of the data was kept back to perform the optimization. This 30 per cent will be referred to as the ‘validation set’, while the remaining 70 per cent will be called the ‘training set’. We stress here that the sequestering of the true test data is essential for the unbiased comparison of our different classifiers. If one uses the same data to optimize an algorithm as to test there is a strong risk of ‘training to the test set’, a phenomenon that will almost certainly lead to poor real world results.

3.2 Performance measures

For both testing and for optimization, we have used four measures commonly applied to classification problems. These are the accuracy \((A)\), the precision \((P)\), the recall \((R)\) and the \(F_1\) score. They are defined in terms of the number of true/false positives/negatives \((t_p, t_n, f_p, f_n)\) as follows:

\[
A = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}
\]

\[
P = \frac{t_p}{t_p + f_p}
\]

\[
R = \frac{t_p}{t_p + f_n}
\]

\[
F_1 = \frac{2PR}{P + R}
\]

For our problem, we define the positive classes to be those corresponding to real objects.

The choice of which of these measures should be used to measure success is rather dependent on the problem at hand. Accuracy can be a misleading measure in situations where the numbers of positive and negative cases are vastly different, e.g. a classifier that always predicts 1 will have an excellent accuracy on a problem where 99 per cent of cases are positive. For our data, the classes are of similar sizes, so the accuracy is a reasonable initial measure. Somewhat better, however, is the \(F_1\) score, which is expressed in terms of the precision (the fraction of reported real objects that are so) and the recall (the fraction of true objects that are found by the classifier). The \(F_1\) score then is a measure that punishes false negatives and false positives equally, but weighted by their inverse fractional contribution to the full set to account for large class number hierarchies.

In real world applications, we are often more or less concerned with false positives than false negatives. For example, if the final number of items classified as objects is small enough to be easily verified by humans, the key will be to minimize \(f_n\) and to maximize the recall. Conversely, if we are met with a data stream too large to be rechecked, it may be more important to prevent contamination of data and so minimize \(f_p\) and the precision.

We quote all of these statistics for completeness and also include the full confusion matrix, which contains the total numbers of true/false positives/negatives. An archetype is shown in Table 2.

Most classifiers can output the probability of an object being in a certain class, and as a result we can study the trade-off between false positives and false negatives in a systematic way. The tool of choice for doing this is the receiver operating characteristic (ROC) curve. This is a plot showing the performance of a binary classifier as its discrimination threshold is varied, and is created by plotting the true positive rate (TPR), or recall, versus the false positive rate (FPR) at various threshold values. The FPR, also known as the fall-out, can be calculated as

\[
FPR = \frac{f_p}{f_p + t_n}.
\]
To bypass these problems, we used PCA on the full training data set (multi-class PCA, Section 4.1.2) as well as on the individual visual classes (single-class PCA, Section 4.1.3) and linear discriminant analysis (LDA, Section 4.2) to extract those features that most faithfully represent the data.

### 4.1 Principal component analysis

PCA is mainly used for the dimensionality reduction of data sets consisting of many correlated variables – data are orthogonally projected to a set of uncorrelated variables, referred to as principal components (PCs), ordered such that most of the variance present in the original data set is preserved in the first few PCs.

Defining a data set \( X = \{x_1, x_2, \ldots, x_n\} \), where \( x_n \) is a \( D \)-dimensional observation with \( n = 1, \ldots, N \), the aim of PCA is to project this data set onto a linear \( M \)-dimensional space, where \( M < D \), such that most of the original data set’s variance is preserved. To find these \( M \) PCs, we first find the covariance matrix \( S \) of the data set, defined as usual as

\[
S = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T,
\]

where \( \bar{x} \) represents the mean of the data set, and then diagonalize \( S \) to find the PCs (the eigenvectors) and eigenvalues of the covariance matrix. Selecting the \( M \) largest PCs, a variable \( x_n \) can then be expressed in \( M \)-dimensional space by projecting it onto these PCs as

\[
a_n = U^\top(x_n - \bar{x})
\]

with \( a_n \), the newly projected observation and \( U \) a \( D \times M \) dimensional matrix with columns corresponding to the \( M \) largest PCs.

In this paper, we typically use the validation data set to find the optimal value of \( M \) for each method. We further use the probabilistic PCA algorithm from the work of Pedregosa et al. (2011) which derives PCA from the perspective of density estimation and has a number of advantages for large data sets (Tipping & Bishop 1999).

#### 4.1.1 Creating algorithm inputs

To obtain the PCs (also referred to as ‘eigenimages’) of a set of \( N \) objects \( I_1, I_2, \ldots, I_N \), it is first necessary to represent each object \( I_i \) by a corresponding vector, \( \Gamma_i \). This is done by expressing each of the \( g \)-, \( i \)- and \( r \)-band \( N \times N \)-pixel images as vectors and then concatenating them. If the \( n \)th object’s \( g \)-band image is denoted as

\[
I_{n-g} = \begin{bmatrix}
g_{1,1} & g_{1,2} & \cdots & g_{1,N} 
g_{2,1} & g_{2,2} & \cdots & g_{2,N} 
\vdots & \vdots & \ddots & \vdots 
g_{N,1} & g_{N,2} & \cdots & g_{N,N}
\end{bmatrix},
\]

then its corresponding \( g \)-band vector is expressed as

\[
\Gamma_{n-g} = \begin{bmatrix}
g_{1,1} 
\vdots 
g_{1,N} 
g_{2,1} 
\vdots 
g_{2,N} 
\vdots 
g_{N,1} 
\vdots 
g_{N,N}
\end{bmatrix}.
\]
The $i$- and $r$-band vectors are also obtained in this way. All three vectors are then concatenated to form the object’s representative vector $\Gamma_a$ as

$$
\Gamma_a = \begin{bmatrix}
\Gamma_{a,i} \\
\Gamma_{a,r} \\
\Gamma_{a - j}
\end{bmatrix}
$$

and the data matrix $X$ is then expressed as

$$
X = \begin{bmatrix}
\Gamma_1 & \Gamma_2 & \cdots & \Gamma_M
\end{bmatrix}
$$

with $X^T$ being the input to the algorithm mentioned in Section 4.1.

### 4.1.2 Multi-class PCA

Multi-class PCA was done by carrying out PCA on the full training set (see Section 3.1) with all the classes mixed together. A small number of the resulting PCs were then kept for our feature extraction purposes. In order to derive the necessary features, all objects (from the test, validation and training sets) were first converted to vectors, as shown in Section 4.1.1, after which they were then expressed as a linear combination of the chosen PCs. The coefficients of these linear combinations were kept as the features of the objects [see equation (7) for the calculation of these features for one object].

In Fig. 2(a), one can see the first six PCs for the $r$-band images from the full training set. It can be seen that the last PCs in the figure are more noisy than the first ones. The optimal number of PCs used for feature extraction is dependent on the classifier at hand, and is therefore a parameter that was optimized for every classifier by using various feature sets for training and validation. These various feature sets were derived by making use of either 0, 5, 10, 25, 50, 100 or 200 PCs, respectively, so that the best set of features for each classifier could be determined. See Section 4.3 for more details regarding our feature sets.

### 4.1.3 Single-class PCA

Another approach to feature extraction is to apply PCA independently to each of the three visual classes of objects (see Table 1) in the training data, yielding a unique set of eigenimages for each class (see Figs 2(b)–(d) for $r$-band PCs of the different visual classes). Features for an object $I_a$ are then obtained by first reconstructing it using the 15 largest PCs of each class in turn, and then calculating the error per pixel (taken as the Euclidean distance) between each reconstructed image and the original image as

$$
e_{\text{error(class)}} = \sqrt{\frac{(I_a - \tilde{I}_{n(class)})^2}{m}},
$$

where $m$ is the number of pixels in an object (including all three of its passbands) and $I_{n(class)}$, with class $\in \{\text{real, art, sat/dip}\}$, is the reconstruction of the object $I_a$ using the PCs from the different visual classes in turn to yield three errors: $e_{\text{real}}$, $e_{\text{art}}$, and $e_{\text{sat/dip}}$. These three calculated errors are then used as features for the object.

### 4.2 LDA

One potential issue with PCA for classification problems is that the direction of maximum variance may not align with the boundary between classes, the so-called decision boundary. There are several ways to alleviate this problem. The simplest is to include more PCA components in the features – as $n_{\text{components}}$ increases so does the probability that the information required to separate the samples is contained.

Another possibility is to include components generated from LDA along with PCA features. LDA projects data in a direction that maximizes the variance between classes while simultaneously minimizing the variance within a class. LDA suffers from only being able to generate a number of components $\leq n_{\text{classes}} - 1$, so can be insufficient as a class separator if used alone. We implemented the LDA algorithm from the work done by Pedregosa et al. (2011) who, in turn, based their design on that of Hastie, Tibshirani & Friedman (2009). We did LDA on the full training set; seeing as we classify objects into one of two possible classes, an object therefore only had one LDA component for use as a feature.

### 4.3 Feature sets

After obtaining features from PCA (namely the PC weights from multi-class PCA and the reconstruction errors from single-class PCA) and LDA, our final step has been to normalize the data to give all the features a standard deviation of 0.5 and a mean of 0. This is done to improve the efficiency of the various routines that optimize the objective function for the different classifiers.

By varying the number of PC weights, omitting or including an LDA component, using either normalized or non-normalized features sets and deriving features using either uncropped images of $51 \times 51$ pixels or cropped images of $31 \times 31$ pixels, we created a total of 56 different feature sets. The various classifiers we have studied have been optimized by varying the feature sets as described above and carrying out intermediate testing. The final test results for a given classifier are reported using whichever feature set gave the best performance.

## 5 MACHINE LEARNING TECHNIQUES

### 5.1 Minimum error classification

Minimum error classification (MEC) is an extremely simple classifier, and is used here in order to form an idea of what the most basic tools can achieve.

MEC takes only the three error-related features obtained from carrying out a single-class PCA (as described in Section 4.1.3), and assigns an object to the class corresponding to the minimum reconstruction error [the calculation of which is described by equation (12)].

Intuitively, this simply reflects the logic that an image should, on average, belong to the class with the best PCA reconstruction (smallest error). We found that the best results were obtained when three visual classes were used for single-class PCA: real objects, artefacts and dipoles/saturated (as discussed in Section 2). A sample is then classified as not being an object if the minimum error corresponds with either the artefact class or the dipole/saturated class. It is classified as a real object otherwise.

Because there is no training process involved, MEC was tested directly on the final test data. The results, shown in Table 3, were
Machine classification of transient images

Figure 2. The first six PCA components in the $r$ band of (a) all the classes in the training data grouped together, (b) the real class, (c) the artefact class and (d) the dipole/saturated class.

obtained by using the central $31 \times 31$-pixel subimages and a non-normalized feature set of reconstruction errors.

5.2 Naïve Bayes

Bayesian reasoning dictates that all quantities of interest are controlled by their probability distributions, and that by reasoning about these probabilities while taking observed data into account, we can make optimal decisions. The importance of this reasoning to machine learning lies in its ability to provide a quantitative approach to weighing the evidence for alternative hypotheses (Mitchell 1997).

The NB classifier (Mitchell 1997) decides on the most probable class $c_m$ via the computation:

$$c_m = \arg\max_{c_j \in C} P(c_j | a_1, a_2, \ldots, a_n).$$

Here $C$ represents the finite set of class values (real and not-real objects) and $a_1, a_2, \ldots, a_n$ are the $n$ image feature values. We then use Bayes' theorem:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)},$$

obtained by using the central $31 \times 31$-pixel subimages and a non-normalized feature set of reconstruction errors.
Table 3. A summary of the performance results of the various classifiers, ordered from best-performing (top) to worst-performing (bottom). The best result for each performance metric is indicated in bold. The true labels are on the left side of the confusion matrices, and the predicted labels are at the top (see Table 2).

<table>
<thead>
<tr>
<th>Machine learning technique</th>
<th>AUC</th>
<th>Accuracy</th>
<th>Recall</th>
<th>Precision</th>
<th>F1-score</th>
<th>Confusion matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Object</td>
</tr>
<tr>
<td>Random forest (RF)</td>
<td>0.97</td>
<td>0.91</td>
<td>0.91</td>
<td>0.93</td>
<td>0.92</td>
<td>3541</td>
</tr>
<tr>
<td>Section 5.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Object</td>
</tr>
<tr>
<td>k-nearest neighbours (KNN)</td>
<td>0.94</td>
<td>0.89</td>
<td>0.90</td>
<td>0.91</td>
<td>0.90</td>
<td>3506</td>
</tr>
<tr>
<td>Section 5.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Object</td>
</tr>
<tr>
<td>SkyNet</td>
<td>0.94</td>
<td>0.88</td>
<td>0.89</td>
<td>0.90</td>
<td>0.89</td>
<td>3461</td>
</tr>
<tr>
<td>Section 5.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Object</td>
</tr>
<tr>
<td>Support vector machine (SVM)</td>
<td>0.93</td>
<td>0.86</td>
<td>0.90</td>
<td>0.85</td>
<td>0.87</td>
<td>3514</td>
</tr>
<tr>
<td>Section 5.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Object</td>
</tr>
<tr>
<td>Minimum error classification (MEC)</td>
<td>0.90</td>
<td>0.84</td>
<td><strong>0.92</strong></td>
<td>0.83</td>
<td>0.87</td>
<td>3559</td>
</tr>
<tr>
<td>Section 5.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Object</td>
</tr>
<tr>
<td>Naive Bayes (NB)</td>
<td>0.80</td>
<td>0.77</td>
<td>0.86</td>
<td>0.77</td>
<td>0.81</td>
<td>3333</td>
</tr>
<tr>
<td>Section 5.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Object</td>
</tr>
</tbody>
</table>

where \( h \) is a given hypothesis and \( D \) is the data, and assuming that 
\( P(a_1, a_2, \ldots, a_n) \) is independent of \( c_j \), we find

\[
c_m = \arg \max_{c_j \in C} P(a_1, a_2, \ldots, a_n | c_j) P(c_j).
\]

(15)

Estimating the \( P(c_j) \) priors is easily done by counting the frequency
with which each class \( c_j \) appears in the training data. To estimate the 
\( P(a_1, a_2, \ldots, a_n | c_j) \) terms in the same fashion, however, requires
a very large training set, and is therefore usually not feasible when
there are many features due to the exponentially large volume that
needs to be sampled.

NB classification rests on the simplifying assumption that, given
the class value \( c_j \), the feature values are conditionally independent,
leading to the final expression:

\[
c_m = \arg \max_{c_j \in C} P(c_j) \prod_i P(a_i | c_j).
\]

(16)

Now, instead of having to estimate various \( P(a_1, a_2, \ldots, a_n | c_j) \) terms
as before, we need to only estimate \( P(a_i | c_j) \), a much more feasible
process. Physically, this corresponds to projecting the full posterior
on to each of the features sequentially. Rather than assuming a
specific probability distribution for the estimation of \( P(a_i | c_j) \), we
binned the features and used the sample distribution of the training
data to estimate \( P(a_i | c_j) \) in each bin and for each class. Features
were binned such that the number of samples per bin averaged four.
It was found that varying the uniform bin sizes had little impact
on the performance of the NB algorithm – in terms of accuracy we
checked that four samples per bin was optimal in a search up to
50 samples per bin.

The major limiting issue for NB lies in its assumption of the
conditional independence of feature values – something that is rarely
true for real life problems and often leads to the degradation of the
performance of the classifier.

Applying this method to the various training and validation data
sets revealed that using the central 31 \( \times \) 31-pixel subimages with
a non-normalized feature set comprising of 50 PCA weights and
three reconstruction error values gives the best results. The results
obtained when applying NB to the corresponding test data can be
seen in Table 3.

5.3 k-nearest neighbours

KNN is an exceptionally simple classifier. It finds the \( k \)-nearest
training objects in feature space to a test point, averages their classes
(with a uniform or distance weight) and classifies the test item accordingly [see the book by Hastie et al. (2009)].

The classifier has the advantages of being simple, inherently non-
linear and free of intrinsic parameters. Performance can be poor in
high-dimensional spaces however, but this was not a problem in our
case. Initial training and validation were carried out with each of our
56 feature sets. Of the variations, the best result seemed to come
from using the central 31 \( \times \) 31-pixel subimages with 10 nearest
neighbours, 53 features (50 PCA components and 3 reconstruction
errors) and normalization of the data. In this case, the results for
the training and validation sets were similar: accuracy 93 per cent,
precision 92 per cent, recall 96 per cent and F1 score 94 per cent.
The results when this model was applied to the test data are given
in Table 3. The KNN algorithm here implemented was taken from
work done by Pedregosa et al. (2011).

5.4 Support vector machine

The SVM is a maximum margin classifier – it tries to find a decision
boundary such that not only are the classes separated, but also that
the separation is as large as possible. In the simplest implementation
of an SVM, this decision boundary is a hyperplane in the feature
space. The strength of the SVM lies, however, in the ease with
which the so-called kernel trick can be applied to the data that maps
the features into a higher dimensional space in which the classes
are well separated by a hyperplane. An illustrative example of the
kernel trick mapping a linear boundary into a curved boundary is
shown in Fig. 3, allowing more successful separation of real from
not-real objects. For detail on this classifier, see the book by Bishop

An SVM with a Gaussian (or radial basis function) kernel function
has two parameters that need to be optimized – the width of the
Gaussian (expressed in terms of \( \gamma = 1/2\sigma^2 \), where \( \sigma \) is the width)
and the soft margin parameter \( C \). The latter sets the trade-off be-
tween a smooth decision surface (small \( C \), less biased) and a better
fit (large \( C \), more accurate). These parameters are optimized using a
grid search on quasi-exponentially increasing parameters and \( n \)-fold
cross-validation – the training set is split randomly into \( n \) equal sets, the classifier is trained on \( n - 1 \) of these, the error on the remaining set is calculated, the process is repeated for the \( n \) choices of the test set, the average error is calculated and after repeating this procedure for every parameter value pair the \( C \) and \( \gamma \) values that minimize the cross-validation error are selected. This process is carried out for each of our 56 different feature sets. We chose to implement an SVM taken from the work done by Pedregosa et al. (2011).

After cross-validation (and trying a variety of different feature sets), the optimal results on the training and validation data came from \( C = 1000 \) and \( \gamma = 0.1 \), using the central 31 × 31-pixel subimages with a feature set consisting of 100 normalized components from PCA, one LDA component and three reconstruction errors. The results when this model was applied to the corresponding test data can be found in Table 3.

5.5 Artificial neural network

Loosely inspired by biological neural systems, ANNs form a well-known class of machine learning algorithms. These networks consist of interconnected nodes, each of which receives and processes information before passing the result onwards to other nodes along weighted connections. Generally speaking, ANNs can be of arbitrary structure, but for many machine learning problems only feed-forward ANNs are required – networks that are directed from an input to an output layer of nodes and that have none, one or many ‘hidden’ layers in between, as shown in Fig. 4. Using training data, the network ‘learns’ a mapping between the inputs and the outputs, and is then able to predict the outputs for new input test data (Graff et al. 2014).

We used SkyNet as our ANN as it is a well-known and robust neural network training tool developed by Graff et al. (2014). It can train large and deep feed-forward ANNs for use in a wide range of machine learning applications, e.g. regression, classification and dimensionality reduction, to name but a few. SkyNet has a few very advantageous features. It allows the training of an autoencoder, a feed-forward ANN where the inputs are mapped back to themselves – these networks provide an intuitive approach to non-linear dimensionality reduction. It also allows the training of recurrent neural networks (RNNs), a class of ANNs in which connections between nodes form a directed cycle, creating a network of nodes with feedback connections. Furthermore, SkyNet employs an optional pre-training method that obtains network weights and biases close to the global optimum of the network objective function, in- stead of starting the training process from a random initial state. SkyNet also makes use of convergence criterion that prevents it from overfitting to the training data.

5.5.1 Number of hidden layers and nodes

A universal approximation theorem (Hornik, Stinchcombe & White 1989) states that feed-forward ANNs with at least three layers (one input, one hidden and one output layer) can approximate any continuous function to some given accuracy. This is the case for as long as the activation function is piecewise continuous, locally bounded and not a polynomial – which is indeed the case for SkyNet.

Furthermore, empirical (Murtagh 1991) and theoretical (Geva & Sitte 1992) considerations suggest that the optimal structure for the approximation of a continuous function is through the use of one hidden layer with \( 2N + 1 \) nodes (\( N \) being the number of input nodes). It was decided that these guidelines would be followed when setting up the ANNs for our classification problem for as long as it remained computationally feasible.

5.5.2 Other network settings and classifier results

For our classification purposes, SkyNet was not configured to act as either an autoencoder or an RNN, and the pre-training method on offer was not used. Testing showed that enabling these options did not result in any significant changes in the classification performance for the problem at hand, and they were therefore deemed unnecessary. Furthermore, the inputs to the network were left unwhitened,\(^7\) and the average error-squared of the network outputs was used to determine convergence. For more details regarding these settings, see Graff et al. (2014).

We found that using the central 31 × 31-pixel subimages with a non-normalized feature set comprising 200 PCA weights and three reconstruction error values did the best for the training and validation phase. The results for the corresponding test set can be seen in Table 3.

For most of the smaller feature sets that were tested, a hidden layer with \( 2N + 1 \) nodes was used, as previously discussed. For the

\(^7\) Whitening here refers to the normalization of the inputs to the network – an option when specifying settings in SkyNet. For more detail on how this is done for SkyNet, see Graff et al. (2014).
larger feature sets (like the best performing feature set mentioned above), however, that would result in a computationally unfeasibly large number of hidden nodes. In the case of the winning feature set (above), a hidden layer with 100 nodes was used, and 140 iterations were needed for the algorithm to converge.

5.6 Random forest

RF, first introduced by Breiman (2001), are ensemble classifiers consisting of a collection of decision trees – they classify instances by combining the predictions of their trees together.

Each such tree is grown from a random bootstrapped sample (of equal size as the training set but selected with replacement) from the training set – this aggregation of bootstrapped samples, each of which is used to grow a separate tree, is referred to as bagging (Breiman 1996).

At each node of a tree, we need to decide how to split the data – this decision could be based on one or more feature values at a time. K-d trees (which were not used in our analysis), for instance, usually use only one feature value when deciding how to optimally split the data at each node. For RF, it is common practice to select \(k \leq K\) feature values when determining the optimal split for nodes in their decision trees, where \(K\) is the number of features in a data set – the choice of \(k\) is usually a parameter specified by the user. For the RF implementation we used in this paper, additional randomness was added by selecting the optimal splitting point from a random sample of \(k < K\) features at every node in a tree. The value of \(k\) was chosen as \(k = \sqrt{K}\), the suggested default value for the algorithm. As described by Carrasco Kind & Brunner (2013), \(k\) is related to the strength of a tree in the classifier (the stronger a tree, the lower its error rate) and the correlation between different trees (a forest with highly correlated trees will have a higher error rate).

To classify a new instance, the classifier combines the individual predictions of all the trees in the forest, either by having them vote for the class that is the most popular or by averaging their probabilistic predictions (as was the case for our implementation). Adding more trees does not improve test performance beyond a certain point, implying that RF are robust against overfitting.

We implemented the RF algorithm taken from the work of Pedregosa et al. (2011). After training and validation using our variety of different feature sets and optimizing for the number of decision trees (varied between 10 and 1000), we found that the best results were obtained for RF consisting of 600 trees and above using the central 31 \(\times\) 31-pixel subimages with a non-normalized feature set comprising 100 PCA components and three reconstruction errors. Applying a 600-tree RF model to the corresponding test set yielded the results given in Table 3.

6 RESULTS

6.1 Best performing classifier

For the SN application, we decided that accuracy and recall would serve as the most important performance metrics; see Section 3.2. Because the two main classes in our data (real objects and not-real objects) are of similar sizes, accuracy serves as a good general measure of performance, while recall was chosen because we are more concerned with missing true objects (false negatives) than we are with contaminating our set of predicted objects with false positives since these are easy for humans to weed out.

The ROC curves for each of the six classifiers based on their performance on the final test set are shown in Fig. 5, along with their AUC values. It is worth pointing out that by changing the threshold value of classifiers, the FPR and TPR can be adjusted accordingly to the survey at hand – for large surveys like the LSST, for example, a small FPR will be required in order to efficiently deal with the number of false positives in the classified data set.

Table 3 gives a summary of the results of the different classifiers, and lists them in order from best performing to worst performing based on accuracy, recall and AUC value. It is interesting to note that RF perform best in all metrics except recall, and further provide the best TPR at any value of the FPR (see Fig. 5). KNN, a simple nonparametric classifier, performed second best in all metrics except recall, closely followed by SkyNet and SVM. Interestingly, the simplest algorithm, MEC, had the best recall, though it lagged significantly in its TPR at small values of the FPR; Fig. 5.

How much can we trust the results in Table 3? One approach is to estimate the sample variance errors on the accuracies, recalls and precisions for each of the classifiers by using bootstrap resampling. The results produced through this procedure suggest sample variance errors of about \(\pm 0.01\) at 95 per cent confidence, and hence our rankings of classifier performances are reasonably robust. However, the differences between the best classifiers (RF, KNN, SkyNet and SVM) are small and it is almost certainly the case that better optimization of any of the algorithms could change the final ordering we obtained. However, from the point of view of replacing human image scanning, it is encouraging that multiple algorithms did very well.

6.2 Inter-classifier agreement

To form an idea of the extent of inter-classifier agreement in terms of images correctly and incorrectly classified, we calculate the Cohen kappa coefficient \(\kappa\) (Cohen 1960), a statistical measure of inter-classifier agreement argued to be more robust than a simple percentage agreement calculation, as it also takes into account chance agreement. Cohen’s kappa measures the agreement between two classifiers that each classifies \(N\) items into \(C\) mutually exclusive classes,
Table 4. Guidelines to the strength of agreement between two classifiers based on the value of the Cohen’s kappa coefficient $\kappa$ (Landis & Koch 1977).

<table>
<thead>
<tr>
<th>Kappa statistic</th>
<th>Strength of agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa \leq 0.00$</td>
<td>None</td>
</tr>
<tr>
<td>$0.00 &lt; \kappa \leq 0.20$</td>
<td>Slight</td>
</tr>
<tr>
<td>$0.20 &lt; \kappa \leq 0.40$</td>
<td>Fair</td>
</tr>
<tr>
<td>$0.40 &lt; \kappa \leq 0.60$</td>
<td>Moderate</td>
</tr>
<tr>
<td>$0.60 &lt; \kappa \leq 0.80$</td>
<td>Substantial</td>
</tr>
<tr>
<td>$0.80 &lt; \kappa &lt; 1.00$</td>
<td>Almost perfect</td>
</tr>
<tr>
<td>$\kappa = 1.00$</td>
<td>Perfect</td>
</tr>
</tbody>
</table>

and is calculated as

$$\kappa = \frac{P(a) - P(b)}{1 - P(b)},$$  \hspace{1cm} (17)

where $P(a)$ is the observed relative agreement and $P(b)$ is the hypothetical probability of chance agreement. If the classifiers agree perfectly, $\kappa = 1$, whereas if there is no agreement between classifiers except for that which would be expected by chance alone, then $\kappa = 0$. Statisticians differ slightly on the exact interpretation of different values of $\kappa$ [see the guidelines given by Landis & Koch (1977), Altman (1991) and Fleiss, Levin & Cho Paik (2003), respectively], but for a rough idea as to what these interpretations are, see Table 4, the guidelines given by Landis & Koch (1977).

Fig. 6 shows the $\kappa$-values calculated for all pairs of classifiers. It can be seen that the strongest agreement between any two classifiers is ‘moderate’ (from Table 4), and is found between SVM and NB, and MEC and NB. The fact that none of the top algorithms show strong agreement suggests that it might be fruitful to combine the predictions of different classifiers together with an ensemble classifier.

6.3 Classifier performance on different classes of images

The performance of the classifiers on the various different visual classes (real objects, artefacts and dipoles/saturated) in the best performing test data sets is shown in Table 5 (note that recall, here, has the same meaning as accuracy). For the calculation of the recall values, it should be noted that true positives ($t_p$) here correspond to the objects in a certain class classified correctly as either real or not-real, while false negatives ($f_n$) correspond to incorrect classifications.

For all classifiers the performance on real objects and artefacts is higher than that on the dipoles/saturated class, the latter therefore being the class that lowers the overall performance of the classifiers. The relatively poor performance on this class can be understood by noting that, first, the objects have features that can be quite similar to those of real objects [this can be seen when comparing the PCs in Fig. 2(d) to those in Fig. 2(b)], and will therefore confuse classifiers. On further visual inspection it can be noted that the images in this class (in general) show a greater diversity in features than real object images do, leading to there not being as strong a common feature to characterize almost all of the images with, as is the case with real object images. When an object in the dipole/saturated class is encountered that looks similar to a real object (this is often the case for noisy images), it can therefore easily be misclassified as strong features of the real object images may approximate the dipole/saturated image more closely than weaker/mixed features from the dipole/saturated class itself. Section 6.4 further discusses misclassification issues.

6.4 Examples of incorrectly classified objects

To give an idea of what causes certain images to get misclassified, we inspected images that were each incorrectly classified by all six classifiers. Fig. 7 shows examples of real objects that were classified as being not-real and Fig. 8 shows examples of not-real objects that were classified as being real.

In Fig. 7(a) it is observed that all three passbands have a faint point-like structure at the centre of their images; this is the reason for the object’s ‘real’ label, as these point-like residuals point towards there being a possible SN. The reason for its misclassification as a not-real object is probably due to the much brighter dipole-like residual in the top-right corner of all the images, something that would confuse the classifiers since we did not force them to focus only on the central few pixels. Figs 7(b) and (c) both show dipole-like structures in all three bands instead of the usual point-like residuals characteristic of real objects – it is no wonder then that all six classifiers misclassified these objects as being not-real. The original classification of these two objects as real might even be a mistake on the part of the hand scanners, which emphasizes how errors in the labelling of the training data can propagate through to the test set.

The same sort of problem is evident in Fig. 8(a) where human scanners classified an object as an artefact that the classifiers understandably mistook for a real object due to the point-like residuals observed in all bands. Fig. 8(b) is an artefact with masked areas in all three bands causing the object to lose most of its spike-like attributes, something very characteristic of the artefact subclass. The fact that all six classifiers classified this object as real is possibly mostly due to the point-like structure at the centre of the i-band.
image. Fig. 8(c) is a dipole, as can be seen by the presence of dipole-like residuals in the $i$ and $r$ bands. The fact that these images are quite noisy (especially the $g$-band image) possibly contributed to the fact that all classifiers predicted it to be a real object. Furthermore, the features of the real and dipole/saturated objects are quite similar in some cases, a possible further contribution to the misclassification of the object as real instead of not-real (this problem was discussed in Section 6.3).

### 6.5 Performance on spectroscopically confirmed SNe

The performance of the classifiers on the spectroscopically confirmed SNe in the best performing test data feature sets is shown in Table 6. Out of the 135 spectroscopically confirmed SNe found in our testing data, 110 were SNe Ia and 19 were other SNe (Types Ib, Ic and II).

It is clear that RF, KNN, SkyNet, SVM and MEC perform well, with the lowest recall value amongst them at 0.90 for SNe Ia, echoing earlier successes. NB continues to be the poorest performer of the group, with its highest recall value at 0.83 for spectroscopically confirmed SNe Ia.

### 6.6 Comparison with human scanners

As part of the SDSS SN survey fake SNe were inserted into the pipeline to test the efficiency of the hand scanners. The fake SN-tag recall, averaged over scanners, was $0.956 \pm 0.010$ (Kessler, private communication). In comparison, Table 7 shows the classification performance of the various classifiers on the same fake SNe found in the test set. It can be seen that all the algorithms other than NB perform comparably to, or even better than, the average human scanner on the fake SNe, with RF leading with a recall of 0.97. Furthermore, when looking at Fig. 5, it can be seen that by changing the discrimination threshold value and inducing only slight penalties (in the form of a larger FPR), a recall (TPR) value of 96 per cent can easily be achieved even when all the test data is used, outmatching the human classifiers.

### 7 CONCLUSIONS AND FUTURE WORK

In examining a broad spectrum of machine learning techniques for transient classification, we have not tried not only to find an optimal solution for the problem but also to demonstrate that it is relatively easy to match or exceed human efficiency with non-linear classifiers such as RF, ANNs, SVMs and KNN. The impressive success of even the linear minimum error classifier suggests that much of the heavy lifting here is done by the judicious use of...
(class-based) PCA as the basis for feature extraction. Overall, we found that RF performed best in all metrics other than recall on the full test set and also performed best on the fake SNe, in line with their superior performance in Brink et al. (2013). Their performance on the small spectroscopically confirmed subset of SN Ia, however, lagged behind KNN, MEC and the ANNs.

In addition to the obvious advantages that machine learning techniques offer in terms of handling large volumes of data, we also stress that automated classifiers have quantified and controllable errors and biases. The latter feature will be essential for building pipelines for data analysis which fully propagate all systematic errors in a reproducible way.

Having illustrated that PCA feature extraction plus a simple classifier provides a robust transient identification solution, the next step would be to build this approach into the data analysis for current and next-generation experiments. This will involve tailoring and optimizing the classification pipeline for further performance enhancements. One option would be to combine classifiers (through e.g. boosting or a weighted voting system); given the fact that the classifiers do not agree that strongly (see Fig. 6), this could be a fruitful avenue to explore. Another strategy would be to try and improve the efficiency for identifying dipole/saturated images (consistently the worst classified category) by augmenting the feature set.

Finally, we note that since we have only used difference images, we have not used any host galaxy information, nor have we used the relative colours of the bands nor multi-epoch data. All of these can be studied and will prove particularly useful for general-purpose surveys that look to classify objects beyond just two or three classes.

**Acknowledgements**

BB and MS thank our collaborators in the SDSS supernova survey team for insights gained during the duration of the survey. LdB thanks Johan A. du Preez and Ben Herbst for their insights during the initial phases of the project. The authors further thank the University College London (UCL) Physics and Astronomy Department for the use of their Splinter computer cluster. LdB is supported by a South African Square Kilometre Array (SKA) Project postgraduate scholarship; NS is supported by a Claude Leon Foundation fellowship; BB acknowledges funding from the South African National Research Foundation (NRF) and SKA; MS is supported by the South African SKA Project, the NRF and the UK Science & Technology Facilities Council (STFC).

Funding for the SDSS and SDSS-II has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, the US Department of Energy, the National Aeronautics and Space Administration, the Japanese Monbukagakusho, the Max Planck Society and the Higher Education Funding Council for England. The SDSS website is http://www.sdss.org/.

The SDSS is managed by the Astrophysical Research Consortium for the Participating Institutions. The Participating Institutions are the American Museum of Natural History, Astrophysical Institute Potsdam, University of Basel, Cambridge University, Case Western Reserve University, University of Chicago, Drexel University, Fermilab, the Institute for Advanced Study, the Japan Participation Group, Johns Hopkins University, the Joint Institute for Nuclear Astrophysics, the Kavli Institute for Particle Astrophysics and Cosmology, the Korean Scientist Group, the Chinese Academy of Sciences (LAMOST), Los Alamos National Laboratory, the Max-Planck-Institute for Astronomy (MPIA), the Max-Planck-Institute for Astrophysics (MPA), New Mexico State University, Ohio State University, University of Pittsburgh, University of Portsmouth, Princeton University, the United States Naval Observatory and the University of Washington.

**References**


Bloom J. S. et al., 2012, PASP, 124, 1175

Breiman L., 1996, Mach. Learn., 24, 123


Goldstein D. A. et al., 2015, AJ, 150, 82


Hornik K., Stinchcombe M. B., White H., 1989, Neural Netw., 2, 359

Kessler R. et al., 2010, PASP, 122, 1415


Murtagh F., 1991, Neurocomputing, 2, 183


**Appendix A**

This appendix assists in describing the classification system used by the SDSS hand scanners during visual inspection of images (see Section 2). While Table 1 describes each of the different classes, Table A1 shows how the objects in each of these classes look like. It shows 6 $r$-band difference images from each of the 10 classes and also illustrate the difference in image quality based on the SNR.
Table A1. The 10 classes, described in Table 1, used by the SDSS hand scanners during visual classification. For each class, six $r$-band difference images of objects are shown – three of which have an SNR above 40 and three having an SNR below 20 (representing roughly 80 per cent of the data set). Images were selected in such a way that they provide a faithful representation of their respective classes.

<table>
<thead>
<tr>
<th>Original Class</th>
<th>SNR &gt; 40</th>
<th>SNR &lt; 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artefact</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>Dipole</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>Saturated Star</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>Moving</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>Variable</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td>Transient</td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>SN Other</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
</tr>
<tr>
<td>SN Bronze</td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
</tr>
<tr>
<td>SN Silver</td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
</tr>
<tr>
<td>SN Gold</td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
</tbody>
</table>