The Astronomical Journal, 141:189 (12pp), 2011 June doi:10.1088/0004-6256/141/6/189

DECISION TREE CLASSIFIERS FOR STAR/GALAXY SEPARATION

E. C. Vasconcellos[1], R. R. de Carvalho[2], R. R. Gal[3], F. L. LaBarbera[4],
H. V. Capelato[2], H. Frago Campos Velho[5], M. Trevisan[6], and R. S. R. Ruiz[1]
[1] CAP, National Institute of Space Research, Av. dos Astronautas 1758, São José dos Campos 12227-010, Brazil
[2] DAS, National Institute of Space Research, Av. dos Astronautas 1758, São José dos Campos 12227-010, Brazil
[3] Institute for Astronomy, University of Hawaii, 2680 Woodlawn Dr., Honolulu, HI 96822, USA
[4] INAF-Osservatorio Astronomico di Capodimonte, via Moiariello 16, Napoli 80131, Italy
[5] LAC, National Institute of Space Research, Av. dos Astronautas 1758, São José dos Campos 12227-010, Brazil
[6] IAG, University of São Paulo, Rua do Matão 1226, São Paulo 05508-090, Brazil
Received 2010 October 27; accepted 2011 February 11; published 2011 May 9

ABSTRACT

We study the star/galaxy classification efficiency of 13 different decision tree algorithms applied to photometric objects in the Sloan Digital Sky Survey Data Release Seven (SDSS-DR7). Each algorithm is defined by a set of parameters which, when varied, produce different final classification trees. We extensively explore the parameter space of each algorithm, using the set of 884,126 SDSS objects with spectroscopic data as the training set. The efficiency of star–galaxy separation is measured using the completeness function. We find that the Functional Tree algorithm (FT) yields the best results as measured by the mean completeness in two magnitude intervals: $14 \leqslant r \leqslant 21$ (85.2%) and $r \geqslant 19$ (82.1%). We compare the performance of the tree generated with the optimal FT configuration to the classifications provided by the SDSS parametric classifier, 2DPHOT, and Ball et al. We find that our FT classifier is comparable to or better in completeness over the full magnitude range $15 \leqslant r \leqslant 21$, with much lower contamination than all but the Ball et al. classifier. At the faintest magnitudes ($r > 19$), our classifier is the only one that maintains high completeness (>80%) while simultaneously achieving low contamination (∼2.5%). We also examine the SDSS parametric classifier (psfMag − modelMag) to see if the dividing line between stars and galaxies can be adjusted to improve the classifier. We find that currently stars in close pairs are often misclassified as galaxies, and suggest a new cut to improve the classifier. Finally, we apply our FT classifier to separate stars from galaxies in the full set of 69,545,326 SDSS photometric objects in the magnitude range $14 \leqslant r \leqslant 21$.

Key words: catalogs – methods: data analysis – surveys – virtual observatory tools

Online-only material: color figures, machine-readable table

1. INTRODUCTION

Astronomical data acquisition has experienced a revolution both in quality and complexity during the last three decades. The main driver has been the deployment of, and enormous growth in, modern digital CCD detectors which replaced venerable photographic plates in the 1980s. Digital images provided by CCDs, coupled with rapid developments in computation and data storage, made it possible and even routine to produce terabytes of astrophysical data in a year. Moreover, several large-scale surveys are being planned for the next 10 years, which will generate a vast quantity of deep and wide photometric images. These surveys will provide data at rates and volumes much greater than any previous projects. Therefore, it is necessary to not only develop new methods for processing and analyzing such huge data volumes, but also to ensure that the techniques applied to extract information from the data are optimal.

A basic step in the extraction of sensible astronomical data from photometric images is separating intrinsically pointlike sources (stars) from extended ones (galaxies). Distinguishing between these two classes becomes increasingly difficult as sources become fainter because of the lack of spatial resolution and signal to noise. Our goal in this work is to test a variety of decision tree (DT) classifiers, and ultimately perform reliable star/galaxy separation for objects from the Seventh Data Release of the Sloan Digital Sky Survey (SDSS-DR7; Abazajian et al. 2009) based on photometric data. We use the SDSS because it also contains an enormous number of objects with spectroscopic data (which give true object classes), and because of the quality, consistency, and accuracy of its photometric data. In the 1970s and 1980s, when digitized images became widespread in astronomy, many authors undertook projects to create automated methods to separate stars from galaxies. The first efforts relied on purely parametric methods, such as the pioneering works of MacGillivray et al. (1976), Heydon-Dumbleton et al. (1989), and Maddox et al. (1990). MacGillivray et al. (1976) used a plot of transmission versus log (area),[7] fitting a discriminant function to separate stars and galaxies. Their star/galaxy separation had a completeness (i.e., the fraction of all galaxies classified as such; also known as the True Positive Rate) of 95% and a contamination (fraction of non-galaxy objects classified as galaxies; also known as the False Positive Rate) of 5%–10%. Heydon-Dumbleton et al. (1989) performed star/galaxy separation on 200 photographic plates digitized by COSMOS. Rather than use object attributes measured directly from the plate scans, they generated classification parameters formed from various combinations and functions of the attributes, plotting these as a function of magnitude in bidimensional parametric diagrams. In these classification spaces, they then used an automated procedure to derive separation functions. They reached 98%±2% completeness with 8%±2% contamination. Maddox et al. (1990) used a set of 10 parameters measured by Automatic Plate Measuring in 600 digitized photographic plates from the UK Schmidt Telescope. They reached 90% completeness with 10% contamination at magnitudes $B_j < 20.5$.

[7] Decimal logarithm of occupied area of the object in the image. The area is measured as the number of squares with the side equals to 8 μm.

All these completenesses and contaminations must be treated with caution, as they are based on comparison to expected number counts, and the plate overlaps, instead of a spectroscopic "truth" sample. Sebok (1979) suggested a classifier based on Bayesian pattern recognition theory. A numerical function was constructed from a statistical model defining the probability of a given object being a star or a galaxy. The main advantage of this method is that there are no free parameters and its calibration depends solely on obtaining a set of images of definite stars. The value of the function establishes the classification.

As the volume of digital data expanded, along with the available computing power, many authors began to apply machine learning methods like DTs (see below, Section 3) and neural networks to address star/galaxy separation. Unlike parametric methods, machine learning methods do not suffer from the subjective choice of discriminant functions and are more efficient at separating stars from galaxies at fainter magnitudes (Weir et al. 1995). These methods can incorporate a large number of photometric measurements, allowing the creation of a classifier more accurate than those based on parametric methods. Weir et al. (1995) applied two different DT algorithms, the GID*3 (Fayyad 1994) and the O-Btree (Fayyad & Irani 1992) as star/galaxy separators for images from the Digitized Second Palomar Observatory Sky Survey (DPOSS), obtaining 90% completeness and 10% contamination. Odewahn et al. (1999) applied a neural network to DPOSS images and established a catalog spanning 1000 deg^2. Odewahn et al. (2004) used a DT and a neural network to separate objects in DPOSS and found that both methods have the same accuracy, but the DT consumes less time in the learning process. Suchkov et al. (2005) were the first to apply a DT to separate objects from the SDSS. The authors applied the oblique DT classifier ClassX, based on OC1, to the SDSS-DR2 (Abazajian et al. 2004). They classified objects into stars, red stars (type M or later), active galactic nuclei (AGNs), and galaxies, giving a percentage table of correct classifications that allows one to estimate their completeness and contamination. Ball et al. (2006) applied an axis-parallel DT. These authors used 477,068 objects from SDSS-DR3 (Abazajian et al. 2005) to build the DT—the largest training set ever used. They obtained a completeness of 93.8% for galaxies and 95.4% for stars.

In this paper, we employ a DT machine learning algorithm to separate objects from SDSS-DR7 into stars and galaxies. We evaluate 13 different DT algorithms provided by the Waikato Environment for Knowledge Analysis (WEKA) data mining tool. We use a training data set containing only objects with measured spectra. The algorithm with the best performance on the training data is used to separate objects in the much larger data set of objects having only photometric data. This is the first work published testing such a large variety of algorithms and using all of the data in the final SDSS data release (see also Ruiz et al. 2009).

Improving star/galaxy separation at the faintest depths of imaging surveys is not merely an academic exercise. By significantly improving the completeness in faint galaxy samples, and reducing the contamination by misclassified stars, many astrophysically important questions can be better addressed. Mapping the signature of baryon acoustic oscillations requires large galaxy samples—and the more complete at higher redshift, the better. The measurement of galaxy–galaxy correlation functions is of course improved, both by increasing the number of galaxies used, and by reducing the washing out of the signal due to the smooth distribution of erroneously classified stars. Weak lensing surveys, which need the largest and purest sample of background (lensed) galaxies and excellent photometric redshifts benefit on both fronts. Similarly, searches for galaxy clusters using galaxy overdensities increase their efficiency when there are fewer contaminant stars and more constituent galaxies. Searches for rare objects, both stellar and extended, also win with reduced contamination, as do any programs that target objects for follow-up spectroscopy based on the source type.

For future imaging surveys, optimized classifiers will require a new breed of training set. Because they cover large sky areas, programs like the Dark Energy Survey, the Large Synoptic Survey Telescope, and Pan-STaRRS can utilize all available spectroscopy to create training samples, even to quite faint magnitudes. Because most spectroscopy has targeted galaxies, the inclusion of definite stars must be accomplished in another way. *Hubble Space Telescope* (*HST*) images have superb resolution and can be used to determine the morphological class (star or galaxy) of almost all objects observed by *HST*. Although covering only a tiny fraction of the sky, the depth of even single-orbit *HST* images and the area overlap with these large surveys will provide star/galaxy (and perhaps even galaxy morphology) training sets that are more than sufficient to implement within an algorithm like the one we describe.

The structure of this paper is as follows. In Section 2, we describe the SDSS data used to evaluate the WEKA algorithms. In Section 3, we give a brief description of the DT method and discuss the technique used to choose the best WEKA DT building algorithm. In Section 4, we discuss the evaluation process and the results for each algorithm tested. In Section 5, we compare our best star/galaxy separation method to the SDSS parametric method (York et al. 2000), the 2DPHOT parametric method (La Barbera et al. 2008), and the axis-parallel DT used by Ball et al. (2006). We also examine whether the SDSS parametric classifier can be improved by modifying the dividing line between stars and galaxies in the classifier's parameter space. We summarize our results in Section 6.

2. THE DATA

We used simple Structured Query Language (SQL) queries to select data from the SDSS Legacy survey.[8] Objects were selected having *r*-magnitudes in the range 14$^{\mathrm{m}}$–21$^{\mathrm{m}}$. We obtained two different data samples: the *spectroscopic*, or *training*, sample and the *application* sample. The spectroscopic sample (see Section 4.1) contains only those objects with both photometric and spectroscopic measurements, while objects in the application sample have only photometric measurements. The spectroscopic sample was obtained through the following query:

```
SELECT
  p.objID, p.ra, p.dec, s.specObjID,
  p.psfMag_r, p.modelMag_r, p.petroMag_r,
  p.fiberMag_r, p.petroRad_r, p.petroR50_r,
  p.petroR90_r, p.lnLStar_r,p.lnLExp_r,
  p.lnLDeV_r, p.mE1_r, p.mE2_r, p.mRrCc_r,
  p.type_r,p.type, s.specClass
FROM PhotoObj AS p
  JOIN SpecObj AS s ON s.bestobjid = p.objid
WHERE
  p.modelMag_r BETWEEN 14.0 AND 21.0
```

8 These queries were written for the SDSS Sky Server database which is the DR7 Catalog Archive Server (CAS); see http://cas.sdss.org/astrodr7/en/. Photometric data were obtained through the `photoObj` view of the database and spectroscopic data through the `specObj` view.

This query returned slightly over one million objects assigned to six different classes according to their SDSS spectral class.[9] However, only objects with a spectral class of star or galaxy were used, leaving us with 884,378 objects for the spectroscopic sample. The majority of excluded objects are spectroscopically QSOs (9.1% of the query results), many of which have one or more saturated pixels in photometry. We also removed 51 stars and 147 galaxies with non-physical values (e.g., −9999) for some of their photometric attributes. Finally, we excluded 54 objects found to be repeated SDSS spectroscopic targets, leaving a final training sample of 884,126 objects, consisting of 84,043 stars and 800,083 galaxies. These all have reliable SDSS star or galaxy spectral classifications and meaningful photometric attributes.

The application sample was built similarly to the spectroscopic sample with the following query:

```
SELECT
 objID, ra, dec, psfMag_r, modelMag_r,
 petroMag_r, fiberMag_r, petroRad_r,
 petroR50_r, petroR90_r, lnLStar_r,
 lnLExp_r, lnLDeV_r, mE1_r, mE2_r,
 mRrCc_r, type_r, type
FROM PhotoObj
WHERE
 modelMag_r BETWEEN 14.0 AND 21.0
```

This retrieved photometric data for nearly 70 million objects from the Legacy survey. We use the Legacy survey rather than SEGUE because we are interested in classifying distant objects at the faint magnitude limit of the SDSS catalog.

3. THE DECISION TREE METHOD

Machine learning methods are algorithms that allow a computer to distinguish between classes of objects in massive data sets by first "learning" from a fraction of the data set for which the classes are known and well defined—the *training* set. Machine learning methods are essential when searching for potentially useful information in large, high-dimensional data sets.

A DT is a well-defined machine learning method consisting of nodes which are simple tests on individual or combined data attributes. Each possible outcome of a test corresponds to an outgoing branch of the node, which leads to another node representing another test and so on. The process continues until a final node, called a leaf, is reached. Figure 1 shows a graphical representation of a simple DT constructed with 50,000 randomly chosen SDSS objects having spectroscopic data. At its topmost node (the root node), the tree may branch left or right depending on whether the value of the data attribute petroR90 is less than or greater than 2.359318. Either of these branches may lead to a child node which may test the same attribute, a different one, or a combination of attributes. The path from the root node to a leaf corresponds to a single classification rule.

Building up a DT is a supervised learning process, i.e., the DT is built node by node based on a data set where the classes are already known. This data set is formed from training examples, each consisting of a combination of attribute values that leads to a class. The process starts with all training examples in the root node of the tree. An attribute is chosen for testing in the node. Then, for each possible result of the test a branch is created and the data set is split into subsets of training examples that have the attribute values specified by the branch. A child node is created for each branch and the process is repeated, splitting each subset into new subsets. Child nodes are created recursively until all training examples have the same class or all the training examples at the node have the same values for all the attributes. Each leaf node gives either a classification, a set of classifications, or a probability distribution over all possible classifications. The main difference between the different algorithms for constructing a DT is the method(s) employed to select which attribute or combination of attributes will be tested in a node.

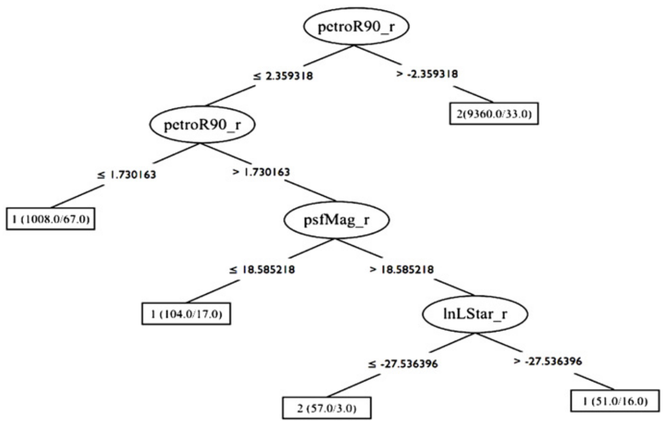

Figure 1. Simple decision tree built with the J48 algorithm (Witten & Frank 2000). This tree was trained with 50,000 objects from the spectroscopic sample, as described in Section 2, and has a minimum number of objects per leaf equal to 50.

3.1. WEKA and Tree Construction

WEKA[10] is a Java Software package for data mining tasks developed by the University of Waikato, New Zealand. It consists of a collection of machine learning algorithms that can either be applied directly or called from an another Java code. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

In this work, we use the WEKA DT tools, which include 13 different and independent algorithms for constructing DTs.[11] In the following, we give a brief description of each algorithm.

J48 is the WEKA implementation of the C4.5 algorithm (Quinlam 1993). Given a data set, it generates a DT by recursive partitioning of the data. The tree is grown using a depth-first strategy, i.e., the algorithm calculates the information gain for all possible tests that can split the data set and selects a test that gives the greatest value. This process is repeated for each new node until a leaf node has been reached.

J48graft generates a grafted DT from a J48 tree. The grafting technique (Geoffrey & Geelong 1999) adds nodes to an existing DT with the purpose of reducing prediction errors. This algorithm identifies regions of the multidimensional space of attributes not occupied by training examples, or occupied only by misclassified training examples, and considers alternative branches for the leaf containing the region in question. In other words, a new test will be performed in the leaf, generating new branches that will lead to new classifications.

BFTree (Best-First DT; Haijian 2007) has a construction process similar to C4.5. The main difference is that C4.5 uses a fixed order to build up a node (normally left to right), while BFTree uses the best-first order. This method first builds the

[9] For more information about spectral class please refer to Section 4.1.

[10] http://www.cs.waikato.ac.nz/ml/weka/

[11] There are other DT algorithms in WEKA that are not capable of working with numerical attributes.

nodes that will lead to the longest possible paths (a path is the way from the root node to a leaf).

FTs (functional trees; Gama 2004) combines a standard univariate DT, such as C4.5, with linear functions of the attributes by means of linear regressions. While a univariate DT uses simple value tests on single attributes in a node, FT can use linear combinations of different attributes in a node or in a leaf.

LMTs (Logistic Model Trees; Landwehr et al. 2006) builds trees with linear functions in leafs as does the FT algorithm. The main difference is that instead of using linear regression, LMT uses logistic regression.

Simple Cart is the WEKA implementation of the CART algorithm (Breiman et al. 1984). It is similar to C4.5 in the process of tree construction, but while C4.5 uses information gain to select the best test to be performed on a node, CART uses the Gini index.

REPTree is a fast DT learner that builds a decision/regression tree using information gain/variance as the criterion to select the attribute to be tested in a node.

Random tree models have been extensively developed in recent years. The WEKA Random Tree algorithm builds a tree considering K randomly chosen attributes at each node.

Random Forest (Breiman 2001) generates an ensemble of trees, each built from random samples of the training set. The final classification is obtained by majority vote.

NBTree (Naive Bayesian Tree learner algorithm; Kohavi 1996) generates a hybrid of a Naive-Bayesian classifier and a DT classifier. The algorithm builds up a tree in which the nodes contain univariate tests, as in a regular DT, but the leaves contains Naive-Bayesian classifiers. In the final tree, an instance is classified using a local Naive Bayes on the leaf in which it fell. NBTree frequently achieves higher accuracy than either a Naive-Bayesian classifier or a DT learner.

ADTree (Alternating DT; Freund et al. 1999) is a boosted DT. An ADTree consists of prediction nodes and splitter nodes. The splitter nodes are defined by an algorithm test, as, for instance, in C4.5, whereas a prediction node is defined by a single value $x \in R^2$. In a standard tree like C4.5, a set of attributes will follow a path from the root to a leaf according to the attribute values of the set, with the leaf representing the classification of the set. In an ADTree, the process is similar but there are no leaves. The classification is obtained by the sign of the sum of all prediction nodes existing in the path. Different from standard trees, a path in an ADTree begins at a prediction node and ends in a prediction node.

LADTree (Holmes et al. 2001) produces an ADTree capable of dealing with data sets containing more than two classes. The original formulation of the ADTree restricted it to binary classification problems; the LADTree algorithm extends the ADTree algorithm to the multi-class case by splitting the problem into several two-class problems.

Decision Stump is a simple binary DT classifier consisting of a single node (based on one attribute) and two leaves. All attributes used by the other trees are tested and the one giving the best classifications (petroR50 in our case) is chosen to use in the single node.

3.2. *Accuracy and Performance: the Cross-Validation Method*

The accuracy of any method for star/galaxy separation depends on the apparent magnitude of the objects and is often measured using the completeness function CP(m) (fraction of all galaxies classified as such) and the contamination function CT(m) (fraction of all stars classified as galaxies) in a magnitude interval δm. These are defined as

$$\mathrm{CP}(m) = 100 * \frac{N_{\mathrm{gal-gal}}(m)\delta m}{N^{\mathrm{tot}}_{\mathrm{galaxy}}(m)\delta m} \qquad (1)$$

and

$$\mathrm{CT}(m) = 100 * \frac{N_{\mathrm{star-gal}}(m)\delta m}{N^{\mathrm{tot}}_{\mathrm{star}}(m)\delta m}, \qquad (2)$$

where $N_{\mathrm{gal-gal}}(m)\delta m$ is the number of galaxy images correctly identified as galaxies within the magnitude interval $(m - \delta m/2, m + \delta m/2)$; $N_{\mathrm{star-gal}}(m)\delta m$ is the number of star images falsely identified as galaxies; $N^{\mathrm{tot}}_{\mathrm{galaxy}}(m)\delta m$ is the total number of galaxies and $N^{\mathrm{tot}}_{\mathrm{star}}(m)\delta m$ is the total number of stars.

It is also useful to define the mean values of these functions within a given magnitude interval. Thus for the mean completeness we have $\langle \mathrm{Compl}\rangle_{\Delta \mathrm{m}} = (1/\Delta m)\sum \mathrm{CP}(m_i)\delta m_i$, with $\Delta m = \sum \delta m_i$. A similar definition holds for the mean contamination. Note that $\langle \mathrm{Compl}\rangle_{\Delta \mathrm{m}} \cdot \Delta m$ gives the area under the completeness function in the interval Δm. Unless otherwise stated, we calculate the completeness and contamination functions using a constant bin width $\delta m = 0\overset{\mathrm{m}}{.}5$.

Our first goal is to find the best performing DT algorithm among those described in Section 3.1, in terms of accuracy, especially at faint magnitudes. However, for large data sets, the processing time is also a concern in this evaluation.

There are various approaches to determining the performance of a DT. The most common approach is to split the training set into two subsets, usually in a 4:1 ratio, and construct a tree with the larger subset and apply it to the smaller. A more sophisticated method is called Cross Validation (CV; Witten & Frank 2000). The CV method, which is used here, consists of splitting the training set into 20 subsamples, each with the same distribution of classes as the full training set. While the number of subsamples, 20, is arbitrary, each subsample must provide a large training set for the CV method. For each subsample, a DT is built and applied to the other 19 subsamples. The resulting completeness and contamination functions are then collected and the median and dispersion over all subsets is found. This gives the CV estimate of the robustness in terms of a completeness function.

4. STAR/GALAXY SEPARATION FOR SDSS

The spectroscopic information provided by SDSS provides the true classification—star or galaxy—of an object. Despite the size of the SDSS spectroscopic sample (~1 million objects), it represents only a tiny fraction of all objects in the SDSS-DR7 photometry (230 million). How can we classify the SDSS objects for which there is no spectroscopic data? The SDSS pipeline already produces a classification using a parametric method based on the difference between the magnitudes `psfMag` and `modelMag` (see Section 4.1). However, it is known (see Figure 6) that this classification is not very accurate at magnitudes fainter than 19.0.

We will take advantage of the large spectroscopic sample from SDSS, for which we know the correct classes for all objects, to train a DT to classify SDSS objects based only on their photometric attributes. We expect that by using such a vast training set, the resulting DT will be capable of maintaining good accuracy even at faint magnitude.

Table 1
SDSS-DR7 Attributes used for Star/Galaxy Separation

Attribute	CAS Variable
PSF magnitude	`psfMag`
Fiber magnitude	`fiberMag`
Petrosian magnitude	`petroMag`
Model magnitude	`modelMag`
Petrosian radius	`petroRad`
Radius carrying 50% of Petrosian flux	`petroR50`
Radius carrying 90% of Petrosian flux	`petroR90`
Likelihood PSF	`lnLStar`
Likelihood exponential	`lnLExp`
Likelihood deVaucouleurs	`lnLDeV`
Adaptive moments	`mRrCc`, `mE1` e `mE2`
Spectroscopic classification	`specClass`

4.1. Attributes

We selected 13 SDSS photometric attributes and a single spectroscopic attribute (`specClass`), as shown in Table 1.

This set of photometric attributes is the same for both the spectroscopic (training) and the application samples. While one could ask what set of attributes produces the most accurate star/galaxy separation, the enormous variety of attributes measured by SDSS for each photometric object places examination of that question beyond the scope of this work. We instead focus on those attributes that are known or expected to strongly correlate with the object classification. These attributes are as follows:

1. The point-spread function (PSF) magnitude (`psfMag`), described in detail in Stoughton et al. (2002), obtained by fitting a PSF Gaussian model to the brightness distribution of the object. We expect the PSF magnitude to be a good flux measure for stars, but it tends to overestimate the flux of extended objects due to their irregular shapes.
2. The fiber magnitude (`fiberMag`) is the flux contained within the $3''$ diameter aperture of a spectroscopic fiber.
3. The Petrosian magnitude (`petroMag`) is a flux measure proposed by Petrosian (1976). He defined a function $\eta(r)$ representing the ratio between the mean surface brightness within a specific radius and the surface brightness at this radius. For a given value of η, one can define a Petrosian radius (`petroRad`); the flux measured within this radius is the Petrosian magnitude. Note that the SDSS pipeline adopts a modified form of the Petrosian (1976) system, as detailed in Yasuda et al. (2001).
4. The SDSS pipeline fits two different galaxy models to the two-dimensional image of an object, in each band: a de Vaucouleurs profile and an exponential profile. The model magnitude (`modelMag`) is taken from the better fitting of these two models.[12]
5. The attributes `petroR50` and `petroR90` are the radii containing 50% and 90% of the Petrosian flux for each band. These two attributes are not corrected for seeing and this may cause the surface brightness of objects of size comparable to the PSF to be underestimated. Nevertheless, the amplitude of these effects is not yet well characterized, and machine learning algorithms may still find relationships distinguishing stars from galaxies.
6. The model likelihoods `lnLStar`, `lnLExp`, and `lnLDeV` are the probabilities that an object would have at least the measured value of chi-squared if it were well represented by one of the SDSS surface brightness models: PSF, de Vaucouleurs, or exponential, respectively.
7. The adaptive moments `mRrCc`,`mE1`, and `mE2` are second moments of the intensity, measured using a radial weight function adapted to the shape and size of an object. A more detailed description can be found in Bernstein & Jarvis (2002). Adaptive moments can be a good measure of the ellipticity.
8. The spectroscopic attribute `specClass` stores the object spectral classification, which is either `unknown`, `star`, `galaxy`, `qso`, `hiz_qso`, `sky`, `star_late`, or `gal_em`. Only objects classified as stars or galaxies are kept in the training set. All other spectroscopic classes constitute a small fraction of the objects, and are not uniquely tied to a specific morphological class.

4.2. Objective Selection of the Optimal Tree Algorithm

As discussed in Section 3.1, the WEKA data mining package provides 13 different algorithms to generate a DT from a training set containing only numerical attributes. Each algorithm employs different computational procedures using different sets of internal parameters[13] resulting in construction of distinct final trees. For each algorithm, we test various sets of internal parameters (always with the same input object attributes). In Table 2, we list all of the internal parameters and the ranges and step sizes with which they were tested. The range of each parameter and the step size within that range were chosen based on the sensitivity of the completeness value to the step size. For the continuous parameters, we tested a variety of step sizes and chose those which resulted in a change of $\sim$5% in the completeness function. When variations were less than 5% we used the default value set by WEKA. We note that these ranges and step sizes are specific to the data and classifications used here and should not be considered appropriate for all cases. For algorithms with many parameters, there are potentially many combinations of parameters to test. Thus, we list in the first column, in parentheses, the number of tests run for each method. We use the CV procedure (Section 3.2) to compute the completeness function for each algorithm and all sets of its internal parameters, to find the best set of parameters maximizing the completeness. Then, we compare the performance among the various algorithms using the optimal internal parameters for each algorithm. In this section, we discuss these tests and compare their results to find the optimal algorithm (and its best internal parameters) that will ultimately be used to provide star–galaxy separation for the entire DR7.

We first exhaustively explored the internal parameter space of each algorithm to determine which parameters significantly change the resulting completeness functions, and discarded the irrelevant ones.[14] We tested the sensitivity of the completeness function to the variation of each parameter taken individually ("single tests") or in combination with others ("combined tests"). For each set of completeness functions generated by the variation of a given parameter ν, we computed the dispersion σ_{m_i} at the middle points $m_i = 14\overset{\text{m}}{.}25 + i * 0\overset{\text{m}}{.}5$ and then averaged over all the intervals to obtain $\sigma_\nu = (1/N_{\text{intervals}}) \sum \sigma_{m_i})$. A parameter was considered irrelevant whenever $\sigma_\nu \leqslant 5\%$. This procedure typically allowed us to discard one parameter per algorithm.

[12] For more details, see http://www.sdss.org/dr7/algorithms photometry.html

[13] We do not provide a full description of all the parameters involved in each algorithm; for more detailed descriptions please refer to http://www.cs.waikato.ac.nz/ml/weka/.

[14] Note that the Decision Stump and NBTree have no free parameters.

Table 2
Exploration of the Parametric Space for Each WEKA Algorithm Used Here

Method	Parameter	Values	$N_{\rm values}$	Best Value
J48graft (1444)	confidenceFactor	0.1, 0.15, 0.2, . . . , 0.5	9	0.45
	minNumObj	2, 3, . . . , 10, 20, . . . , 100, 120, . . . , 200	19	8
	relabel	True, false	2	False
	subtreeRaising	True, false	2	True
	unpruned	True, false	2	False
	useLaplace	True, false	1	False
Simple Cart (190)	minNumObj	2, 10, 40, 100, 150, . . . , 400	10	2
	numFoldsPruning	2, 3, 4, . . . , 10	9	5
	sizePer	1	1	1
	useOneSE	True, false	2	False
	usePrune	True, false	2	False
J48 (2898)	confidenceFactor	0.1, 0.15, 0.2, . . . , 0.5	9	0.45
	minNumObj	2, 3, . . . , 10, 20, . . . , 100, 120, . . . , 200	23	7
	numFolds	3, 4, . . . , 10, 20, . . . , 100, 120, . . . , 200	22	3
	reducedErrorPruning	True, false	2	False
	subtreeRaising	True, false	2	False
	unpruned	True, false	2	False
	useLaplace	True, false	2	True
REPTree (288)	maxDepth	−1, 0, 1, 2, . . . , 10	12	−1
	minNum	0, 1, 2, 3	4	0
	minVarProp	0.001	1	0.001
	noPruning	True, false	2	True
	numFolds	10, 20, . . . , 50	5	50
Random tree (35)	KValue	1,5,10,15,20	5	15
	maxDepth	0, 5, 10, . . . , 30	7	0
	minNum	1	1	1
Random Forest (60)	maxDepth	0, 10, 20, 30	4	20
	numTrees	5, 10, 20, 30, 40	5	40
	numFeatures	0, 10, 20	3	10
BFTree (1008)	minNumObj	2, 10, 20, . . . , 40, 60, 80, 100, 125, . . . , 200	12	2
	numFoldsPruning	5, 10, 20, 30, 40	5	5
	pruningStrategy	postPruned, prePruned, unpruned	3	PostPruned
	sizePer	1	1	1
	useErrorRate	True, false	2	True
	useGini	True, false	2	False
	useOneSE	True, false	2	False
ADTree (80)	numOfBoostingIterations	5,10,50,100,150	5	150
	randomSeed	0,50,100,150	4	150
	searchPath	All, the heaviest, the best z-pure, a random	4	All Paths
LMT (576)	errorOnProbabilits	True, false	2	False
	fastRegression	True, false	2	True
	minNumInstances	0, 15, 30	3	30
	numBoostingIterations	−1, 5, 10	3	−1
	splitOnResiduals	True, false	2	False
	useAIC	True, false	2	False
	weightTrimBeta	0,0.1, 0.2, 0.3	4	0.2
LADTree (4)	numOfBoostingIterations	10, 50, 100, 200	4	200
FT (864)	errorOnProbabilits	True, false	2	False
	minNumInstances	0,50,100	3	0
	modelType	FT, FTLeaves, FTInner	3	FT
	numBoostingIterations	10,20, . . . ,60	6	60
	useAIC	true, false	2	False
	weightTrimBeta	0,0.1,0.2,0.3	4	0

Notes. The columns are: (1) the name of the algorithm; (2) the names of their internal parameters; (3) the range of values studied; (4) the number of values in the interval; (5) and the best parameter value based on the highest completeness found.

In the second step, we searched for the optimal value of each remaining parameter. To do this, we first computed the range for each parameter wherein $\sigma_\nu \leqslant 5\%$, as before. Then, within these limits we tested each algorithm to find its optimal parameter set. The results of these tests are shown in Figure 2, which displays the range of completeness functions computed using the CV procedure when varying the internal parameters of each algorithm. At bright magnitudes, $r \lesssim 19$, the algorithms behave very similarly, and their efficiency is stable under variation of their internal parameters.

We then analyze the relative performance of the 13 algorithms by comparing their completeness and contamination functions as well as their processing times when run with their optimal sets of internal parameters. We define the quantities $\langle \text{Compl} \rangle_{\text{bright}}$,

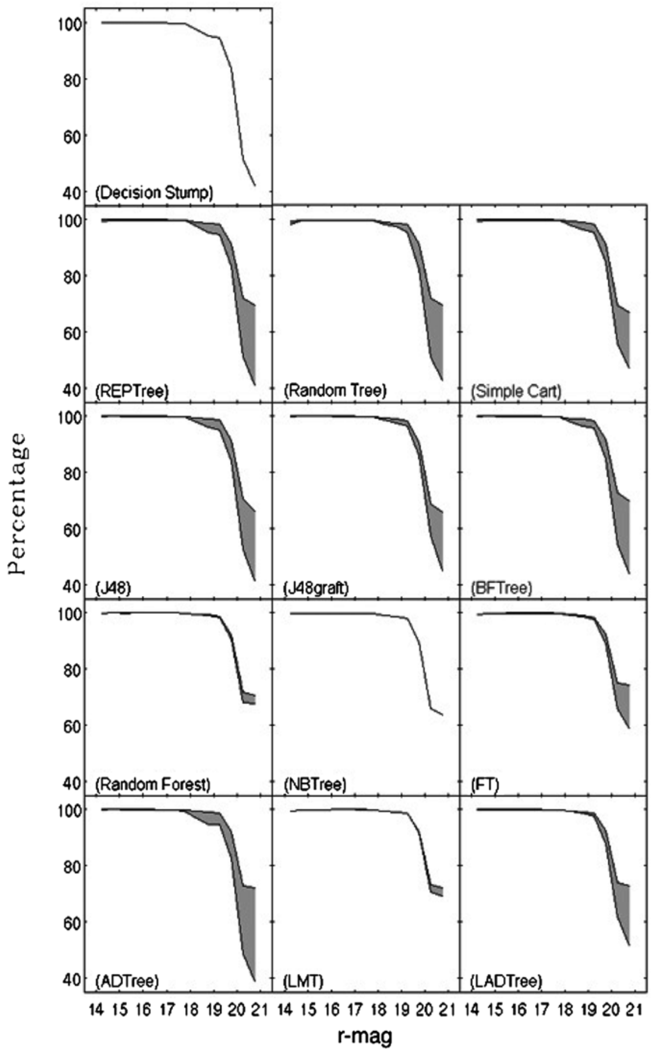

Figure 2. Results of the parameter space exploration for each of the 13 WEKA DT algorithms. The hatched areas show the loci of the completeness functions for galaxies obtained from the CV procedure.

$\langle\text{Compl}\rangle_{\text{faint}}$, and $\text{Compl}_{20.75}$ which are the mean completenesses (see Section 3.2) in the magnitude intervals $14 \leqslant r < 19$, $19 \leqslant r \leqslant 21$, and $20.5 \leqslant r \leqslant 21$, respectively. The results of this comparison are given in Table 3. Column 1 gives the algorithm name as in Section 3.1. Column 2 gives the total number of significant internal parameters for that algorithm. Column 3 gives the processing time for each algorithm (running on a 64 bit PC AMD Phenom X3 8650 triple-core processor-2.3 GHz), and Columns 4, 5, and 6 give the quantities $\langle\text{Compl}\rangle_{\text{bright}}$, $\langle\text{Compl}\rangle_{\text{faint}}$, and $\text{Compl}_{20.75}$ along with their standard deviations. The rows in Table 3 have been ordered according to the values of $\langle\text{Compl}\rangle_{\text{faint}}$.

It is clear from Table 3 (and Figure 2) that all of the algorithms have comparable efficiency at brighter magnitudes ($r < 19$). However, at $r \geqslant 19$ their performance varies significantly. The Decision Stump, with no internal parameters, unsurprisingly performs worst, although it is the fastest algorithm. The NBTree, which also has no internal parameters, is considerably better albeit more computationally expensive. The fast algorithms, including Simple Cart, J48 and J48graft, REPTree and random tree, have mean faint-end completeness of $\langle\text{Compl}\rangle_{\text{faint}} \sim 81\%$–$83\%$ but with $\text{Compl}_{20.75} < 70\%$. The remaining algorithms provide better faint-end completeness at the cost of increasing processing times. Note that all of the algorithms are almost equally robust, as measured from the dispersions of their mean completeness. We conclude that the FT algorithm is optimal because of its greater accuracy at faint magnitudes while still requiring only modest processing time. In fact, as shown in Table 3, the FT algorithm is not only the most accurate among the 13 we tested, but also very robust (ranked second in dispersion of mean completeness).

It is not surprising that the FT produces the best result. Previous works like Breiman et al. (1984), Brodley & Utgoff (1995), Murthy et al. (1994), and Gama & Douglas (1997) explored a set of algorithms called multivariate trees. In these algorithms, the decision nodes can contain tests based on a combination of attributes. This kind of node test makes it possible to build split surfaces oblique to the axes of the attribute space, instead of the axis-parallel splits of univariate DT algorithms like C4.5 (Quinlam 1993). Quinlam (1992) and Witten & Frank (2000) studied the use of a different sort of multivariate trees, called model trees, applied to problems with a number of classes in R domain (in this case there is no predefined set of known classes), called regression problems. Model trees, different from standard multivariate trees, use combinations of attributes in the leaf nodes instead of decision nodes. The FT algorithm combines these two approaches in one framework to construct a DT with linear combinations in both the decision nodes and the leaves. The FT uses linear regression to generate the linear combinations of attributes, building multivariate decision nodes while growing the tree and leaf nodes when pruning it. The results of Gama (2004) suggest that multivariate models using linear functions both at decision nodes and leaves have some advantage over standard axis-parallel, multivariate, and model trees, especially in large data sets. In contrast, Suchkov et al. (2005) use a multivariate tree algorithm called OC1 (Murthy et al. 1994) to classify objects from the second data release of SDSS (DR2). The OC1 algorithm is a standard multivariate tree and uses linear functions only in the decision nodes.

To examine what causes the different success rates among algorithms, we compare the best (FT) and worst (NBTree) performing DTs. The NBTree was considered the worst among those algorithms for which the final number of attributes selected by the algorithm is the same. All methods used all 13 attributes except the Decision Stump (01) and ADTree (12). We examined the attribute values for objects where the classifications from the two methods agreed and where they disagreed. We find that there are some attributes where the objects for which the two classifiers agree and for which they disagree are clearly separated. In the FT versus NBTree comparison, the separation is greatest in Petrosian attributes (especially PetroR90 and petroRad). The exact reasons for this are unclear, but because each algorithm uses the same attributes, it must be an artifact of the tree construction. This further reinforces the need to test classifiers with large training samples to find the best algorithm.

4.3. Constructing the Final Decision Tree

Having found that the FT algorithm provides the best star/galaxy separation performance in CV tests, we must choose a training set to construct the final DT to classify objects from the SDSS-DR7 photometric catalog. As described in Section 2, the CAS database provides 884,126 objects with star or galaxy spectral classification, which comprises our full training sample. However, using this entire data set within the WEKA implementation of the FT algorithm requires large amounts of computer

Table 3
Main Results of the Comparative Study of WEKA Algorithms

Algorithm	Number of Parameters	Processing Time (hr)	$\langle\mathrm{Compl}\rangle_{\mathrm{bright}}$ (%)	$\langle\mathrm{Compl}\rangle_{\mathrm{faint}}$ (%)	$\mathrm{Compl}_{20.75}$ (%)
Decision Stump	0	0.03	99.20(±0.17)	68.06(±1.20)	42.29(±9.64)
NBTree	0	1.12	99.64(±0.16)	79.19(±1.39)	63.55(±14.90)
J48graft	6	0.09	99.74(±0.12)	80.93(±1.16)	65.84(±10.39)
Simple Cart	5	0.05	99.63(±0.16)	81.56(±1.13)	67.06(±9.51)
J48	7	0.08	99.73(±0.12)	81.70(±0.96)	66.30(±7.69)
REPTree	5	0.09	99.50(±0.18)	82.76(±1.09)	69.32(±8.80)
Random tree	3	0.06	99.50(±0.18)	82.76(±1.09)	69.32(±8.80)
Random Forest	3	1.13	99.77(±0.12)	83.15(±1.14)	70.48(±9.91)
BFTree	7	0.24	99.69(±0.15)	83.18(±1.10)	69.85(±9.55)
ADTree	3	1.42	99.73(±0.12)	83.80(±1.12)	71.88(±9.81)
LMT	7	5.50	99.66(±0.15)	83.91(±1.14)	72.18(±9.39)
LADTree	1	7.90	99.70(±0.14)	84.39(±1.10)	72.74(±9.41)
FT	6	2.50	99.64(±0.15)	84.98(±1.08)	74.04(±8.45)

Notes. The columns are the name of the algorithm tested, the number of parameters of the algorithm that can change the resultant tree, the mean processing time averaged over all parameter sets tested, the mean completeness averaged over the magnitude interval [14, 19], the mean completeness averaged over the magnitude interval [19, 21], and the completeness in the faintest magnitude bin ($20.5 \leqslant r \leqslant 21.0$).

memory, decreasing the overall performance. To see if the final tree depends strongly on the training set size and the exact values of each object's attributes, we performed a test using subsets of the training data while perturbing the object attributes. For each photometric attribute discussed in Section 4.1, we generate a perturbed value

$$X = X_{\mathrm{obs}} + \sigma u, \qquad (3)$$

where X_{obs} is the observed attribute value and u is a random Gaussian deviate, with the dispersion σ computed from the first and fourth quartiles of the attribute's uncertainty distribution. We then subdivided the training data with these perturbed attributes into four samples, each with 221,029 objects, and built four different DTs. We required each subset to have the same class distribution as the full training set. We then used these four DTs to classify the original training set. The results are shown in Figure 3, which shows that the completeness function is unchanged in the magnitude range $14 \leqslant r < 20$. At the faintest magnitudes, $20 \leqslant r \leqslant 21$, the completeness function varies slightly, by $\lesssim 5\%$. These results suggest that we can safely reduce the training set size by a factor of a few with no significant loss of accuracy. With this reduced training data set, the DT construction is speeded up considerably. This test simultaneously confirms that the final DT is insensitive to measurement errors on the individual object attributes.

We further tested the dependence of DT classification success on training set size. In future optical surveys, we expect that spectroscopic data for the faintest objects will be at a premium, available for only a small fraction of the photometric objects. At $r > 19$, the fainter part of our training sample, we constructed DTs using different sized subsamples of the available training data, ranging from 10% to 100% of the full training set. We find that the completeness remains essentially unchanged as long as at least 20% of the faint-end training sample is used (about 7100 objects). This result suggests that future deep surveys may be able to perform accurate star/galaxy separation even with a modest truth sample for training, as long as there is sufficient resolution in the images.

To select data to train the final DT, we consider bright and faint objects separately. At $14 \leqslant r < 19$, we selected 1/4 of the objects from the spectroscopic sample, maintaining the same magnitude distribution as in the full sample, yielding 205,348 objects. For magnitudes $19 \leqslant r \leqslant 21$, we kept all objects from the training set. These faint objects are poorly sampled spectroscopically, so we attempted to provide all possible information to the FT algorithm to maintain the completeness function at faint magnitudes.

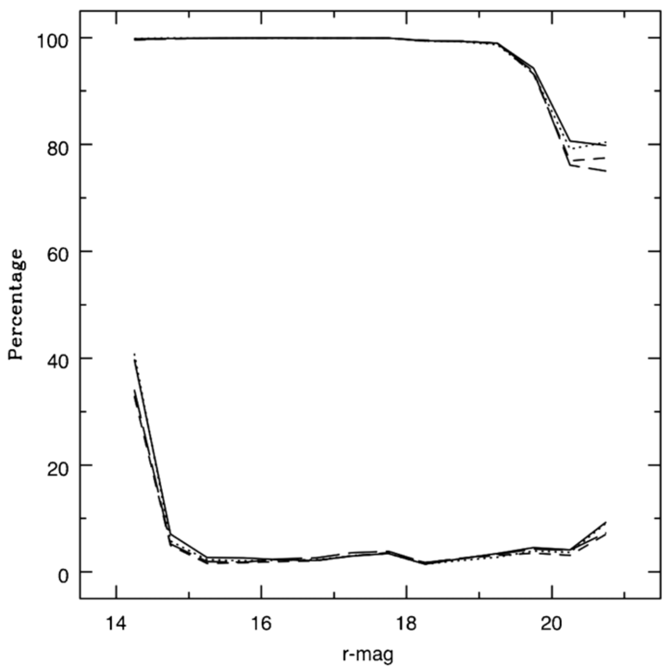

Figure 3. Completeness (upper curves) and contamination (lower curves) functions for all four perturbed data sets used to train a DT with the FT algorithm. Each data set is represented by a different line type.

The final training set contains 240,712 objects with 13 attributes each, extracted from the original spectroscopic sample. The resultant DT was then applied to classify objects with r-magnitudes between 14 and 21 from the entire SDSS-DR7 Legacy survey catalog. A portion of the resulting catalog is shown in Table 4; the full catalog is available in the online version of the journal. Column 1 gives the unique SDSS ObjID and Column 2 contains the SDSS `ModelMag` magnitude in r band. Columns 3 and 4 give $type_r$ and `type`, the SDSS classifications

Table 4
Star/Galaxy Classification Provided by SDSS and by Our FT Decision Tree

SDSS ObjID	ModelMag$_r$	Type$_r$	Type	FT Class
588848900971299281	20.230947	3	3	2
588848900971299284	20.988880	3	3	2
588848900971299293	20.560146	3	3	2
588848900971299297	19.934738	3	3	2
588848900971299302	20.039648	3	3	2
588848900971299310	20.714321	3	3	2
588848900971299313	20.742567	3	3	2
588848900971299314	20.342773	3	3	2
588848900971299315	20.425304	3	3	2
588848900971299331	20.582634	3	3	2

Notes. Column 1 lists the unique SDSS ObjID, Column 2 contains the SDSS modelMag magnitude in r-band. Columns 3 and 4 give $type_r$ and $type$, the SDSS classifications using r-band alone and using all five bands, respectively. Column 5 provides our FT Decision Tree classification, where 1 is a star and 2 is a galaxy.

(This table is available in its entirety in a machine-readable form in the online journal. A portion is shown here for guidance regarding its form and content.)

using the r band alone and using all five bands, respectively. The SDSS classifications are 0 = unknown, 3 = Galaxy, 6 = Star, and 8 = Sky. Column 5 provides our FT DT classification, where 1 is a star and 2 is a galaxy. We note that we classified *all* objects with $14 < \texttt{ModelMag}_r < 21$, regardless of their SDSS classification. Detections which the SDSS has classified as $\texttt{type} = 0$ or 8 should be viewed with caution as there is a high likelihood that they are not truly astrophysical objects.

In the next section, we present a comparative study of our catalog and other catalogs providing star/galaxy classification for the SDSS.

5. COMPARISON WITH OTHER SDSS CLASSIFICATIONS

We compare the results of our DT classification of objects in the SDSS photometric catalog with those from the parametric classifier used in the SDSS pipeline, the 2DPHOT software for image processing (La Barbera et al. 2008) and the DT classification from Ball et al. (2006). These comparisons utilize only objects with spectroscopic classifications so that the true classes are known. All three methods give classifications other than just stars or galaxies. However, as we are interested only in stars and galaxies, all samples described in this section are exclusively composed of objects which were classified by the respective methods as star or galaxy.

5.1. FT Algorithm Versus 2DPHOT Method

2DPHOT is a general purpose software package for automated source detection and analysis in deep wide-field images. It provides both integrated and surface photometry of galaxies in an image and performs star–galaxy separation by defining a stellar locus in its parametric space (La Barbera et al. 2008). The comparison was done for 10,391 objects from the spectroscopic sample which have been reprocessed by 2DPHOT, with the results presented in Figure 4. We see that both classifiers have the same completeness trends with magnitude, but our FT classifier generates almost no contamination, while contamination in 2DPHOT reaches ∼40%.

5.2. FT Algorithm Versus Axis-Parallel DT

Ball et al. (2006) were the first to apply the DT methodology to SDSS data. They use an axis-parallel DT to assign a probability that an object belongs to one of three classes: stars, galaxies, or nsng (neither star nor galaxy). The completeness and contamination functions for both Ball's axis-parallel DT and for our FT, calculated using a sample of 561,070 objects from Ball's catalog, are shown in Figure 5. These results show that our FT tree performs similarly to the axis-parallel tree, but the FT generates lower contamination than the axis-parallel tree.

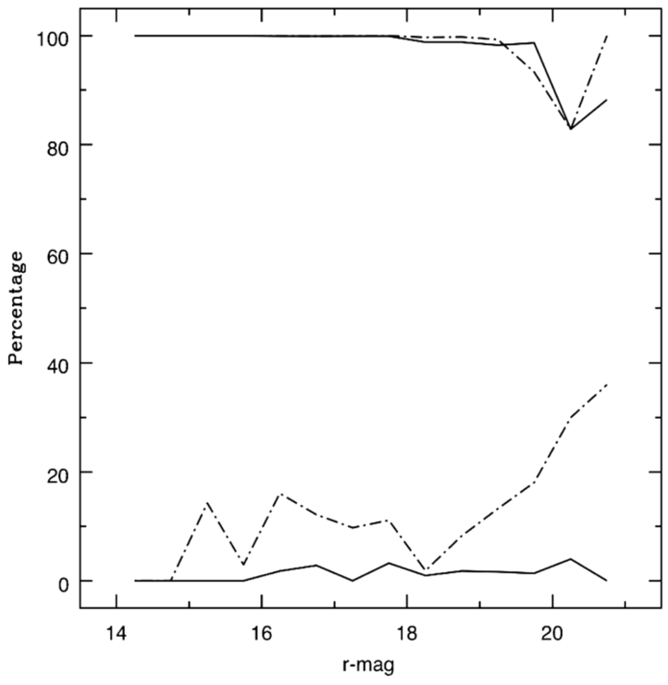

Figure 4. Completeness (upper curves) and contamination (lower curves) for the sample of 10,391 SDSS objects reprocessed with 2DPHOT and classified as either star or galaxy. The full lines show the completeness and contamination functions from our DT classification whereas the dash-dotted lines are for the 2DPHOT classification.

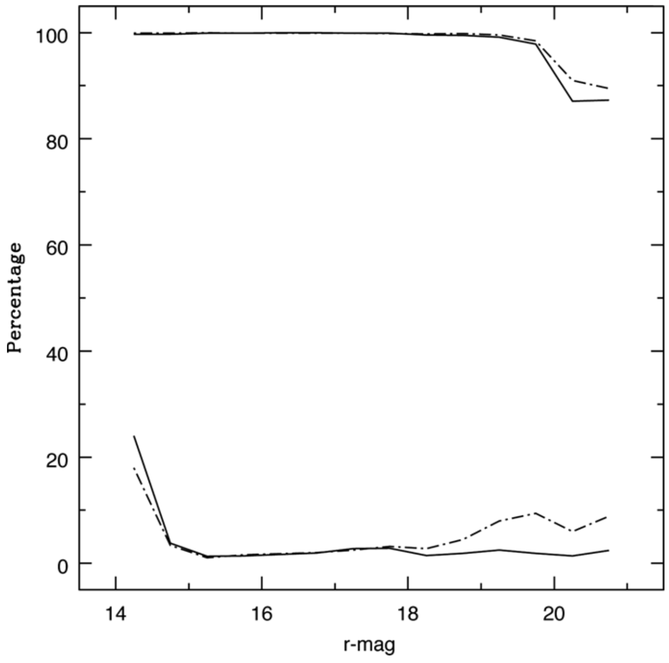

Figure 5. Completeness (upper curves) and contamination (lower curves) for the 561,070 objects with SDSS spectroscopic data processed by Ball et al. (2006). The full lines give the completeness and contamination functions of our DT classification whereas the dash-dotted lines give the same for Ball's classification.

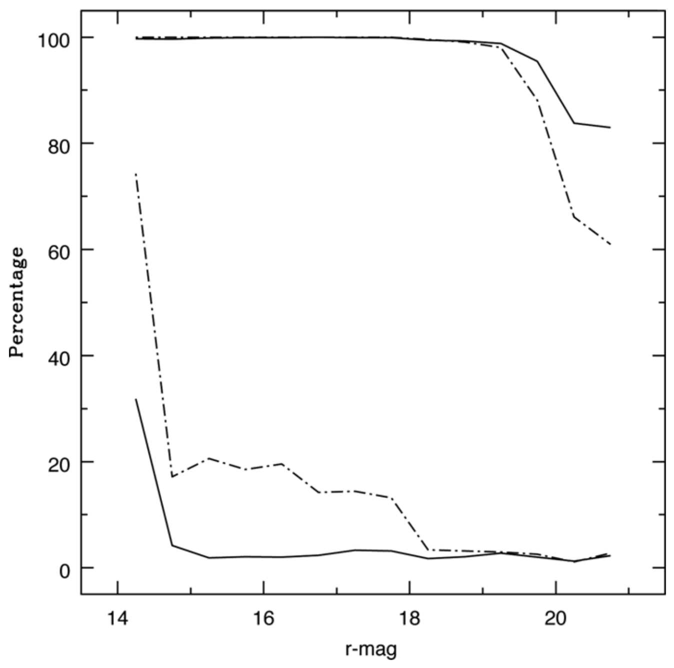

Figure 6. Completeness (upper curves) and contamination (lower curves) functions for the SDSS parametric method (dash-dotted lines) and for our DT method (solid lines). In both cases, we have used 25% of the whole spectroscopic sample to train the classifier and then applied it to the remaining 75% of the data not used for training.

At the faint end ($r \geqslant 19$), our contamination remains constant around 3% while Ball's catalog has a contamination rate of ∼9%.

5.3. FT Algorithm Versus SDSS Pipeline Parametric Method

The SDSS pipeline classifies an object into three classes: unknown, galaxy, or star using the difference between the PSF and model magnitudes (cf. Section 4.1). If the condition psfMag − modelMag > 0.145 is satisfied, the object is classified as galaxy; otherwise, it is classified as a star.

We first analyzed the behavior of our final DT in comparison with that of the SDSS pipeline for the full spectroscopic sample. We note that this will contain the same objects used to initially train the DT, as discussed in Section 4.3. However, because only $\sim 25\%$ of the entire sample is used for training, we neglect its influence on the results. The completeness and contamination curves are shown in Figure 6 for a subsample of the entire spectroscopic sample where 8406 objects with at least one saturated pixel have been removed. The results show that the completeness of our DT classification stays above 80% for magnitudes $r \geqslant 19$ with negligible stellar contamination. In contrast, the completeness for the SDSS pipeline drops to 60% in the same range, and is highly contaminated at the brighter magnitudes. These results show that application of our DT provides a gain in completeness of ∼20% at faint magnitudes when compared with the SDSS pipeline. Our FT tree also yields much lower contamination than the SDSS parametric method; while the FT contamination stays around 6%, the SDSS parametric contamination is about 17% for magnitudes brighter than 18.

Finally, we compared our DT classifications and the SDSS parametric classifications for all objects in the application sample (see Section 2). Since there is no a priori true classification in this case (unlike the training sample), we compare these two methods by assuming that the DT is correct, based on its better

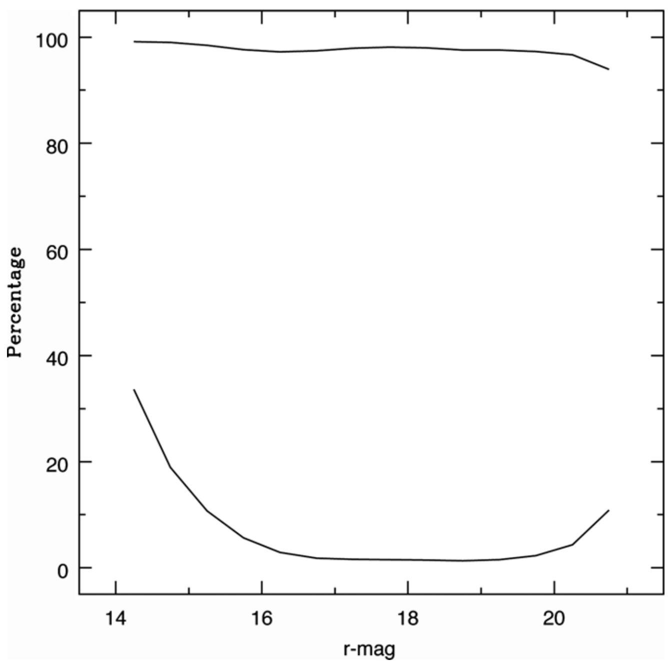

Figure 7. Completeness and contamination for the SDSS parametric method when our DT classification is taken as the truth.

performance (described above). In Figure 7, we show the completeness and contamination functions for the SDSS parametric classification assuming that our DT classification is correct. Only objects that are classified as either star or galaxy by both the SDSS parametric method and the FT DT are considered here, with other classifications in SDSS being an irrelevant fraction (0.05% of the total sample). We see from this figure that there is significant disagreement between our DT classification and the SDSS parametric method when classifying stars, implying a greater stellar contamination in the SDSS classification. The completeness shows, at faint magnitudes $20.5 \leqslant r \leqslant 21$, that the two classifiers disagree on ∼6% of the whole application sample.

5.4. A Simple Test of the SDSS Pipeline Parametric Method

The SDSS parametric method relies on a single parameter test: if the condition psfMag − modelMag > 0.145 is satisfied, the object is classified as a galaxy; otherwise, it is classified as a star. The cutoff between the two classes was chosen based on simulated images of stars and galaxies. However, the choice of dividing line between stars and galaxies in this parameter space can be improved using the extensive spectroscopic training sample. To test this, we retrieved psfMag − modelMag for the training sample, and used the Decision Stump to generate a single node DT. In creating such a tree, the Decision Stump will seek the value of psfMag − modelMag which maximizes completeness while minimizing contamination. Surprisingly, we find that the optimal requirement for an object to be classified as a galaxy is psfMag − modelMag > 0.376, significantly different from the value used by SDSS.

To see why there is such a large difference, we have examined images of bright objects misclassified by the SDSS parametric method, which are responsible for the high contamination rate shown in Figure 6. We find that many of these objects are in close pairs. An example of nine such objects are shown in Figure 8, where each panel includes an object misclassified by the SDSS parametric method. We clearly see that many

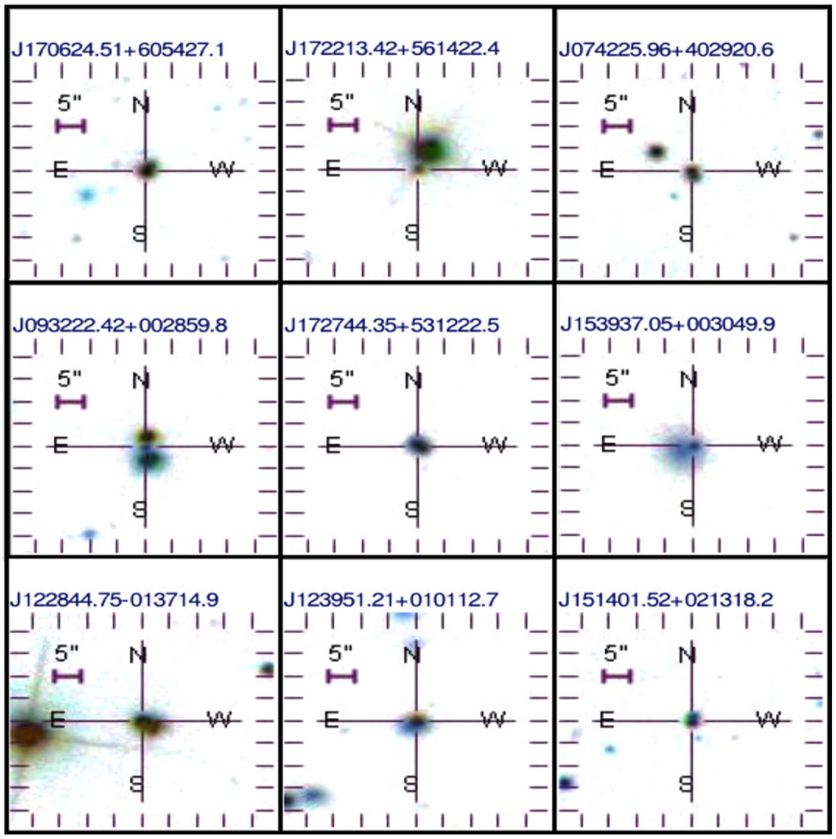

Figure 8. Postage stamps from SDSS-DR7 for nine objects which are misclassified by the SDSS parametric method. Almost all are blends or have brighter nearby companions which affect the photometry.

(A color version of this figure is available in the online journal.)

are either heavily blended with a companion or close enough to another object for the value of psfMag − modelMag used by the SDSS parametric classifier to be influenced by the neighboring object. This is clearly visible in Figure 9, which shows psfMag − modelMag as a function of magnitude for the training sample. Spectroscopically classified stars are shown in red, while galaxies are shown in green. A second ridgeline of stars is clearly seen with psfMag − modelMag $\sim$ 0.3, all of which are misclassified with the standard SDSS parametric cut. It is important to emphasize that this second ridgeline is due to these stars being pairs or multiples in a majority of cases. This issue was noted already in the SDSS Early Data Release (Stoughton et al. 2002), where they noted that un-deblended star pairs and galaxies with bright nuclei are improperly classified. This further validates our optimal DT classifier, which, by using a larger number of attributes and not combining them, is better able to create a rule set for correctly classifying even those objects with nearby companions.

We note that it is possible that there exist attributes in the SDSS database that we do not employ but which could improve classification accuracy when used in a DT. It may also be the case that PSF-deconvolved attributes, such as those measured by 2DPHOT, can improve performance. Other "indirect" attributes, such as colors or combinations of multiple attributes, can also be considered, as done by Ball et al. (2006) and the SDSS parametric method. However, testing all possible sets of object attributes is outside the scope of this work. Other avenues for further testing include generating a Committee Machine, which looks at the outputs of multiple classification algorithms and chooses a final meta-class for each object based on majority

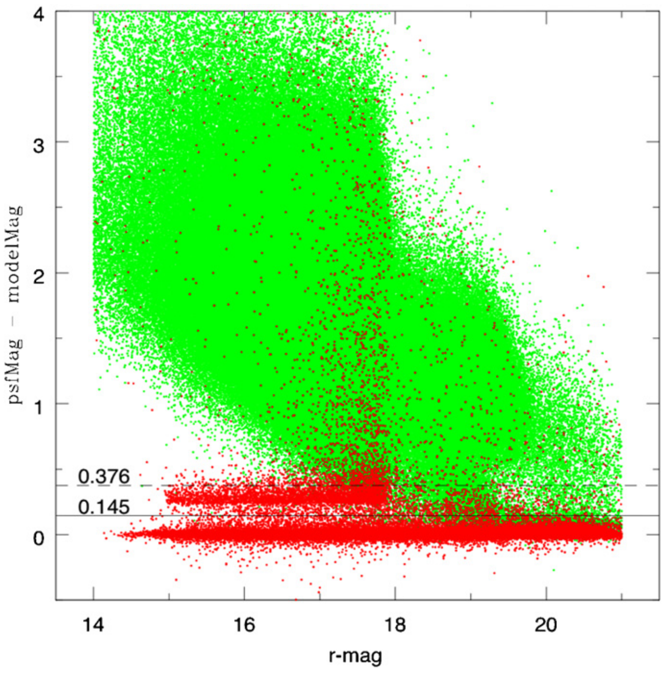

Figure 9. psfMag − modelMag parameter used in the standard SDSS classifier is plotted as a function of magnitude for the spectroscopic training sample. Stars are shown as red dots, while galaxies are in green. The dividing lines used by the SDSS classifier (psfMag − modelMag = 0.145) and derived by the Decision Stump (psfMag − modelMag = 0.376) are also shown. The SDSS classifier incorrectly assigns galaxy classifications to many relatively bright stars, most of which have nearby neighbors.

(A color version of this figure is available in the online journal.)

vote or some other criterion. Nevertheless, our DT significantly outperforms all other published classifiers applied to SDSS photometry.

6. SUMMARY

We analyzed the star/galaxy separation performance of 13 different DT algorithms from the publicly available data mining tool WEKA when applied to SDSS photometric data. This is the first examination of a public data mining tool applied to such a large catalog. We show the completeness and contamination functions for all of the algorithms in an extensive study of each algorithm's parameter space. These functions were obtained using CV tests and demonstrate the capability of each algorithm to classify objects from training data. Thus, our study may be used as a guide for astronomers who desire to apply such data mining algorithms in star–galaxy separation tasks and other similar data mining problems. The main results of our work are as follows:

1. Thirteen different algorithms from WEKA were tested and Figure 2 shows the locus of the resultant completeness functions.
2. All algorithms achieved the same accuracy in the magnitude range $14 \leqslant r < 19$, but with large differences in the required processing time (Table 3 and Figure 2).
3. The completeness functions in the faint magnitude interval ($r \geqslant 19$) show that the ADTree, LMT, and FT are the most robust and have similar performance (see Table 3). However, FT requires approximately half the time of the others to build a DT.
4. The WEKA FT algorithm is therefore chosen as the optimal DT for classifying SDSS-DR7 objects based on photometric attributes.
5. We show, using this FT, that reducing the size of the training data by a factor of $\sim$5 does not change significantly the completeness and contamination functions (see Figure 3).
6. We use the FT WEKA algorithm to construct a DT trained with photometric attributes and spectral classifications for 240,712 objects from the SDSS-DR7, and apply this DT to separate stars from galaxies in the Legacy survey sample of objects in the magnitude range $14 \leqslant r \leqslant 21$ from SDSS-DR7 (see URL in Section 4.3).
7. Finally, we compare our results with the SDSS parametric method, 2DPHOT and Ball's axis-parallel DT. Our catalog has much lower contamination than all three methods (Figures 4–6), and a higher completeness than the SDSS parametric method for faint objects ($r \geqslant 19$).

We thank Robert Hanisch for helpful comments and suggestions and Nick Ball for providing data for comparison with his star/galaxy separation based on SDSS-DR3. Funding for the SDSS and SDSS-II has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, the U.S. 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 Web site 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, University of Cambridge, 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

Abazajian, K., et al. 2004, AJ, 128, 502
Abazajian, K., et al. 2005, AJ, 129, 1755
Abazajian, K., et al. 2009, ApJS, 182, 543
Ball, N. M., Brunner, R. J., Myers, A. D., & Tcheng, D. 2006, ApJ, 650, 497
Bernstein, G. M., & Jarvis, M. 2002, AJ, 123, 583
Breiman, L. 2001, Mach. Learn., 45, 5
Breiman, L., Friedman, J., Olshen, R., & Stone, C. 1984, Classifications and Regression Trees (Belmont, CA: Wadsworth)
Brodley, C. E., & Utgoff, P. E. 1995, Mach. Learn., 19, 5
Fayyad, U. M. 1994, in Proc. 12th National Conf. on Artificial Intelligence, (Menlo Park, CA: AAAI), 601
Fayyad, U. M., & Irani, K. B. 1992, in Proc. 10th National Conf. on Artificial Intelligence (Menlo Park, CA: AAAI), 104
Freund, Y., Mason, L., Bratko, I., & Dzeroski, S. (ed.) 1999, Proc. 16th International Conf. on Machine Learning (San Fancisco, CA: Morgan Kaufmann), 124
Gama, J. 2004, Mach. Learn., 55, 219
Gama, J., & Douglas, H. F. (ed.) 1997, Proc. 14th International Conf. on Machine Learning (San Fancisco, CA: Morgan Kaufmann), 134
Geoffrey, I. W., & Geelong, V. 1999, in Proc. 16th International Joint Conf. on Artificial Intelligence, Vol. 2 (San Fancisco, CA: Morgan Kaufmann), 702
Haijian, S. 2007, Master's thesis, Univ. Waikato
Heydon-Dumbleton, N. H., Collins, C. A., & MacGillivray, H. T. 1989, MNRAS, 238, 379
Holmes, G., Pfahringer, B., Kirkby, R., Frank, E., & Hall, M. 2001, ECML, 161
Kohavi, R. 1996, in 2nd International Conf. on Knowledge Discovery and Data Mining, 202
La Barbera, F., de Carvalho, R. R., Kohl-Moreira, J. L., Gal, R. R., Soares-Santos, M., Capaccioli, M., Santos, R., & Sant'anna, N. 2008, PASP, 702, 120
Landwehr, N., Hall, M., & Frank, E. 2005, Mach. Learn., 95, 161
MacGillivray, H. T., Martin, R., Pratt, N. M., Reddish, V. C., Seddon, H., Alexander, L. W. G., Walker, G. S., & Williams, P. R. 1976, MNRAS, 176, 265
Maddox, S. J., Efstathiou, G., Sutherland, W. J., & Loveday, J. 1990, MNRAS, 243, 692
Murthy, S., Kasif, S., & Salzberg, S. 1994, JAIR, 2, 1
Odewahn, S. C., Djorgovski, S. G., Brunner, R., Gal, R. R., & de Carvalho, R. R. 1999, BAAS, 31, 1235
Odewahn, S. C., et al. 2004, AJ, 128, 3092
Petrosian, V. 1976, ApJ, 209, L1
Quinlam, J. R. 1986, Mach. Learn., 1, 81
Quinlam, J. R. 1992, in 5th Australian Joint Conf. on Artificial Intelligence
Quinlam, J. R. 1993, C4.5: Programs for Machine Learning (San Fancisco, CA: Morgan Kaufmann)
Ruiz, R. S. R., Campos Velho, H. F., Santos, R. D. C., & Trevisan, M. 2009, TEMA, 10, 75
Sebok, W. L. 1979, AJ, 84, 1526
Stoughton, C., et al. 2002, AJ, 123, 485
Suchkov, A. A., Hanisch, R. J., & Margon, B. 2005, AJ, 130, 2401
Weir, N., Fayyad, U. M., & Djorgovski, S. 1995, ApJ, 109, 2401
Witten, I. H., & Frank, E. 2000, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations (San Fancisco, CA: Morgan Kaufmann)
Yasuda, N., et al. 2001, AJ, 122, 1104
York, D. G., et al. 2000, AJ, 120, 1579