MASTERS THESIS

Classifying SDSS Data using Active Learning

Author:
Nathan Steer

Supervisor:
Professor Anna Scaife

A thesis submitted to the University of Manchester
for the degree of Master of Science

in the

Department of Physics and Astronomy in the School of Natural Sciences
Faculty of Science and Engineering

2022

Contents

List of Figures

List of Tables

THE UNIVERSITY OF MANCHESTER

Abstract

Faculty of Science and Engineering
Department of Physics and Astronomy in the School of Natural Sciences

Master of Science

Classifying SDSS Data using Active Learning

by Nathan Steer

This thesis applies active learning to a dataset of spectroscopically labelled sources from the Sloan Digital Sky Survey (SDSS). The sources are selected from the photometric data in the SDSS and the Widefield Infrared Survey Explorer (WISE). Two machine learning techniques were used: a neural network and a random forest classifier. Four different active learning methods were investigated with these data: uncertainty sampling, best-vs-second-best, variance reduction, and learning active learning, plus a generic random method as a control. The uncertainty sampling was implemented using a form known as entropy measure, for which a binary case and a multi-class case were tested separately. These machine learning techniques were also applied to different configurations of Gaussian clouds to help understand their effect on different types of data. The learning active learning received particular focus as the most expandable method. To assist in the selection of active learning methods, the average accuracy scores and feature importances, as well as the class precision, recall, and F_1-scores were all compared. These tests resulted in the entropy sampling and the learning active learning being selected as most capable, requiring only $25,600$ datapoints in the training set, with the latter having the most room for improvement.

Declaration of Authorship

I, Nathan Steer, declare that this thesis titled, “Classifying SDSS Data using Active Learning” and the work presented in it are my own. I confirm that:

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright Statement

Acknowledgements

To begin with I would like to wholeheartedly thank my supervisor: Professor Anna Scaife, who has been of immense help throughout this course and through the writing of the thesis. Professor Scaife has provided support in the study of machine learning and its application to the SDSS dataset while also aiding in the in the implementation of the techniques used here.
I want to thank Thomas Dugdale, a fellow masters student who I could complain to and ask about the occasional question about python functionality, as well as Alexandra Bonta and Devina Mohan who I was able to clarify things with and use for support. I also want to thank my housemate Charlie Williams, for the copious amounts of tea he provided. Finally for my personal thank yous, I would like to thank the JBCA for being a wonderful community to work in.
Funding for the Sloan Digital Sky Survey IV has been provided by the Alfred P. Sloan Foundation, the U.S. Department of Energy Office of Science, and the Participating Institutions. SDSS-IV acknowledges support and resources from the Center for High Performance Computing at the University of Utah. The SDSS website is www.sdss.org. SDSS-IV is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS Collaboration including the Brazilian Participation Group, the Carnegie Institution for Science, Carnegie Mellon University, Center for Astrophysics | Harvard & Smithsonian, the Chilean Participation Group, the French Participation Group, Instituto de Astrofísica de Canarias, The Johns Hopkins University, Kavli Institute for the Physics and Mathematics of the Universe (IPMU) / University of Tokyo, the Korean Participation Group, Lawrence Berkeley National Laboratory, Leibniz Institut für Astrophysik Potsdam (AIP), Max-Planck-Institut für Astronomie (MPIA Heidelberg), Max-Planck-Institut für Astrophysik (MPA Garching), Max-Planck-Institut für Extraterrestrische Physik (MPE), National Astronomical Observatories of China, New Mexico State University, New York University, University of Notre Dame, Observatário Nacional / MCTI, The Ohio State University, Pennsylvania State University, Shanghai Astronomical Observatory, United Kingdom Participation Group, Universidad Nacional Autónoma de México, University of Arizona, University of Colorado Boulder, University of Oxford, University of Portsmouth, University of Utah, University of Virginia, University of Washington, University of Wisconsin, Vanderbilt University, and Yale University.
This publication makes use of data products from the Wide-field Infrared Survey Explorer, which is a joint project of the University of California, Los Angeles, and the Jet Propulsion Laboratory/California Institute of Technology, funded by the National Aeronautics and Space Administration.

Chapter 1

Introduction

Classifying data accurately and efficiently has always been a problem for large sets of data, especially for the reams of information found in astronomical surveys. Originally, once observations started to be made at an increased rate, human computers (mostly women) were employed en masse in some laboratories and observatories, a notable one being the Cavendish Physical Laboratory (Allan & Leedham, 2021). Human computers eventually became obsolete with the widespread adoption of programmable and mechanical computers with the ability to complete certain tasks faster and in greater volume. With the evolution of technology, data was gathered through wider and deeper surveys (Shore, 2009).

While the hardware advanced, software to match it was needed so was developed alongside; however, the sheer magnitude of data available today makes even the best computers inefficient at analysing the complete surveys accurately. This is a problem for datasets such as the Sloan Digital Sky Survey (SDSS), where more than several hundred million individual samples have been catalogued (Ahn et al., 2014). This is where different algorithms come into play, in particular those where not all data needs to be stringently analysed, only a small portion. An increasingly commonplace method in astronomy (among other subjects) is the adoption of machine learning in the analysis of a small portion of the data, then using the resultant model to evaluate the rest. Machine learning can be thought of as the development of computational models that get better through training or experience. This style of analysis has already been attempted with the Sloan Digital Sky Survey and the Wide-field Infrared Survey Explorer (WISE), however the methods are not perfect (Clarke et al., 2020).

Classifying data for the SDSS involves sorting each astronomical object into the target classes: *galaxies*, *quasars*, and *stars*. This is a far cry from the previous interpretations of the cosmos where only stars were really considered. In fact classification involving these three types of bodies has a rich history, opened up by Herschel's discovery of the Milky Way being made of stars (Herschel, 1785). His first interpretation can be seen in Figure 1.1. Wright's theory (Wright, 1750) also deserves a mention as a progenitor even if the spiral galaxy does not directly relate to it. It was not until the so called Great Debate (or the Shapley–Curtis Debate) where differing theories were put to the test on whether

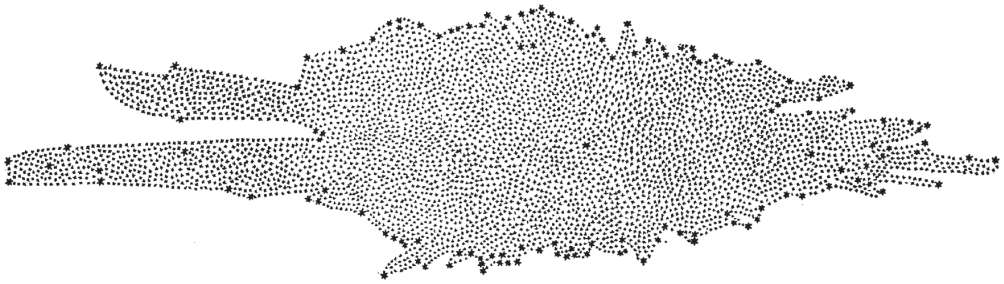


FIGURE 1.1: Herschel's first illustration of the stars of the Milky Way including 683 star-gauges (Herschel, 1785).

the Universe was just the Milky Way - or if it contained multiple collections of stars, so called *island Universes* at the time (Horvath, 2020). Edwin Hubble's discovery of *Cepheid Variable* stars outside the Milky Way (Hubble, 1925) proved that the Universe was much more expansive and holds multiple galaxies. This enabled the classification of both stars and galaxies. Quasars came about later as only through radio astronomy can they be properly observed, beforehand they were simply curious galactic nuclei (Shields, 1999). Before delving into the subtleties of quasars, it is best to understand roughly what stars and galaxies are as this can help uncover why quasars themselves are so unique.

1.1 Stars

Stars are one of the most common sights in the night sky and have fascinated astronomers and more for centuries. They are near spherical balls of plasma that have formed through the gravitational collapse of dust clouds (Woodward, 1978). These dust clouds exist in interstellar or intergalactic space as nebulae and gravitational instabilities can cause the slightly denser regions to begin coalesce (Clarke, 1992). Once the compressed dust has a high enough mass, the pressure exerted on the core from gravity allows nuclear fusion to begin. This nuclear fusion begins by turning hydrogen into helium and once the inner hydrogen is fused, helium to carbon. This process continues until reaching iron, at which point the energy required for fusion is too great (ESO, 2001). The more mass the star has, the faster this process occurs. The energy emitted from the fusion gradually travels outwards from the centre to the outer layers through photons where it may become trapped by the various solar constructs around the surface or, be emitted in the form of electromagnetic radiation. The tremendous amount of radiation being released is what gives the stars their glow.

Much like ogres and onions, stars are made of layers (Howe, 2009). A star with similar properties to the Sun can split their layers into two sets; the inner and outer layers. Called so due to the ability to observe them; the inner layers cannot be observed directly while the outer layers can. The inner layers hold most of the mass of the star and are believed to be what drives the immense magnetic fields emanating from these luminous objects. The inner layers consist of the: *core*, *radiative zone*, *tachocline*, and *convection zone*

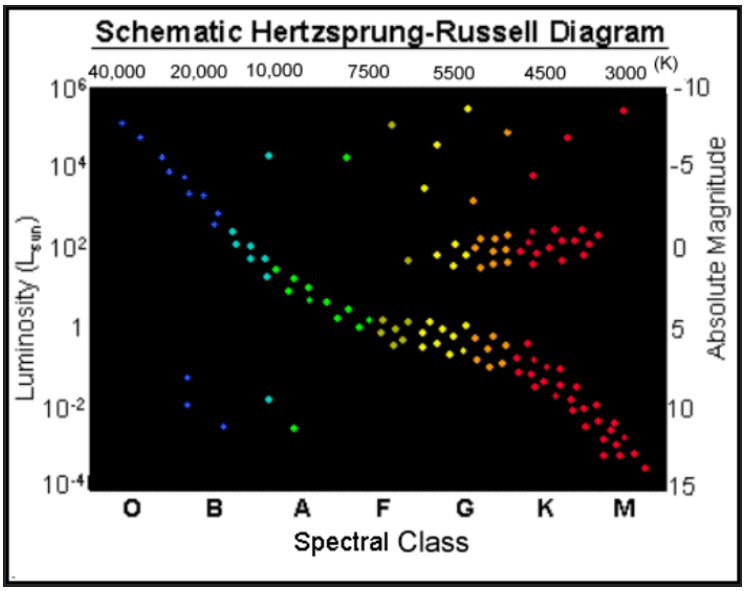


FIGURE 1.2: Hertzsprung-Russel diagram, showing the different star classifications as a function of temperature (O, B, A, F, G, K, M, L, T). The Luminosity classifications (I, II, III, IV, V, VI, VII) are not labelled (SDSS, 2004).

(Howe, 2009). The outer layers are observed to have a: *photosphere*, *chromosphere*, *transition region*, and finally the *corona* (Baker et al., 2021). The inner layers are not as well understood because there have been no direct observations; however they have been deduced through physical laws and models. Stars have a life cycle that ranges from their birth from the gaseous clouds to their eventual collapse once the fusion stops and they turn into dwarf or neutron stars (among other things) (Kepler et al., 2021; Meisel et al., 2018). These changes lead to different sub-classes related to their luminosity and temperature. The *luminosity* class type shows how bright a star is and is designated I through to VII and the function for the temperature of the star is shown through the spectral classes O-T - these can be seen on Figure 1.2).

1.2 Galaxies

After the realisation that the Universe held similar large-scale structures to the Milky Way, they were named galaxies and a new area of astronomical observation dawned. These massive collections of astronomical objects have been found to not only consist of stars but include dust, nebulae, rogue bodies, and theoretically dark matter (Rubin, 1983; Searle & Zinn, 1978). They can also orbit each other and some even have local dwarf galaxies surrounding them, a well-known example being our own Milky Way and the Magellanic Clouds (De Leo et al., 2020). Galactic structures are believed to be somewhat synonymous, with a galactic core full of younger, larger, metal rich stars and a surrounding disk or area full of older, metal poor stars (Steinmetz & Navarro, 2002; White & Rees, 1978), but the large variation in morphology makes them difficult to definitively say.

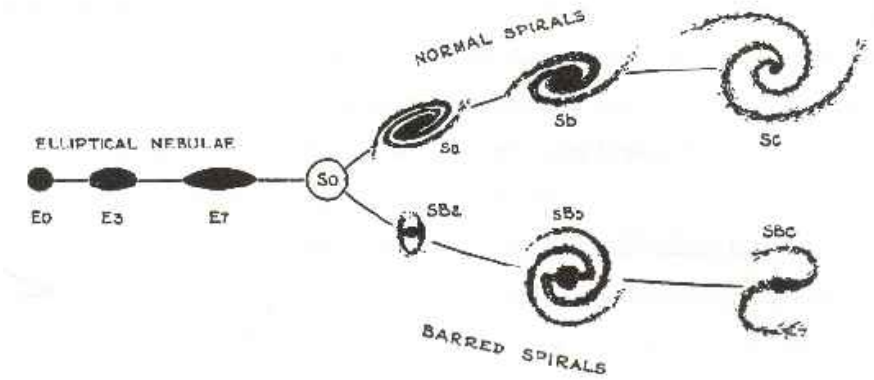


FIGURE 1.3: Hubble tuning fork (Buta, 2011).

Modelling them is also tough as the outer layers spin faster than conventional models allow, which is where the dark matter idea comes in (Rubin, 1983; Persic et al., 1996).

Different galactic structures allow them to be grouped into different morphological classifications. The original idea for this was from Edwin Hubble and was specified in his paper 'Extra-Galactic Nebulae' (Hubble, 1926). The classifications for galaxies were separated into two major groups: *ellipticals* and *spirals*. Elliptical galaxies are relatively smooth with no discernible patterns, spiral galaxies are split further into galaxies with two obvious arms extending from a central point with a noticeable 'bar' and spirals with multiple arms that look fairly clustered but without any noticeable 'bar'. These two major groupings were then split further for easier identification. There is a third intermediary group known as *lenticular* galaxies where a disk structure exists but there are no obvious spirals, and then a fourth group with no specific shapes related known as *irregulars*. Figure 1.3 shows these morphologies clearly (besides the irregular galaxies). Gerard de Vaucouleurs expanded on this system and insisted that *bars*, *rings*, and *spiral arms* were also important areas for classifying different galaxies (de Vaucouleurs, 1959). De Vaucouleurs also assigned numbers to Hubble classified galaxies (from -6 to 10), with negative numbers representing elliptical galaxies, 0 showing a lenticular galaxy, and finally positive values were associated with spirals. These classifications along with numbered values are generally accepted though there have been other attempts at applying alternate morphological classes (Sheehan, 2011).

1.3 Quasars

Quasars are deep space radio sources that release a stupendous amount of energy, originally named *quasi-stellar radio sources* before a paper shortened the naming. The original name was an accurate description of what they are; they were point-like objects (similar to stars) yet were receding at a velocity outstripping that of any other catalogued star - not to mention, they were the only star-like objects found in radio surveys (Smith & Hoffleit, 1963). There was much difficulty in identifying what they were even after successive

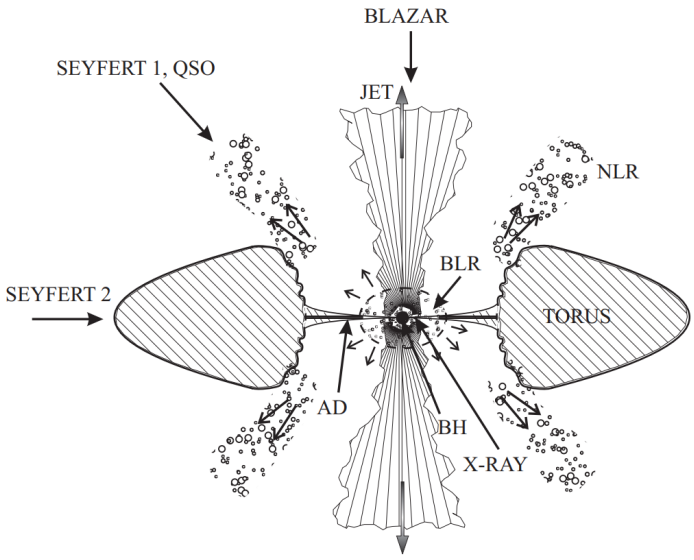


FIGURE 1.4: Diagram showing a unified model of an AGN, with related expulsions (Jovanović & Popović, 2009).

papers by Hazard et al. (1963); Schmidt (1963); Oke (1963); Greenstein (1963), where sensible models were proposed. Then a suggestion in Matthews & Sandage (1963) theorised a rough mass that would provide the right amount of energy for what was being seen, such a mass would have a Schwarzschild radius of 10^{-4} parsec, making it a supermassive black hole (Shields, 1999). This leads to the current understanding that quasars are an emission from a substantial active galactic nuclei (AGN), with the energy originating from the accretion disk falling into the black hole (Shakura & Sunyaev, 1973; Jovanović & Popović, 2009). A general unified model exists for these incomprehensible engines, a diagram of which can be seen in Figure 1.4, much of it is based on what can be observed though, and not all AGNs are quasars; only the largest.

1.4 Classification Methods

These three types of astronomical objects are distinct and need to be classified into separate groups. Just viewing pictures of distant objects compiled by telescopes is not an efficient way of going about it, though this can be helpful and is seen in projects such as Galaxy Zoo (Lintott et al., 2008). Instead, spectroscopic measurements are typically used. These measurements can be easily taken for close by objects, only there is no way to know if an object is close before measuring it, thus a more general approach is required; one which can be performed on every target. This broader method involves spectroscopic analysis performed on the redshift of objects (Hutchinson et al., 2016). First appearing in Davis et al. (1982), it has led to multiple surveys that purely focus on retrieving the redshift of targets which can then be used in classification (Jones et al., 2004; Newman et al., 2013). The spectral analysis of redshift is the attempt at deducing the physical parameters of a target from the measured values. The emission line(s) from the object are identified

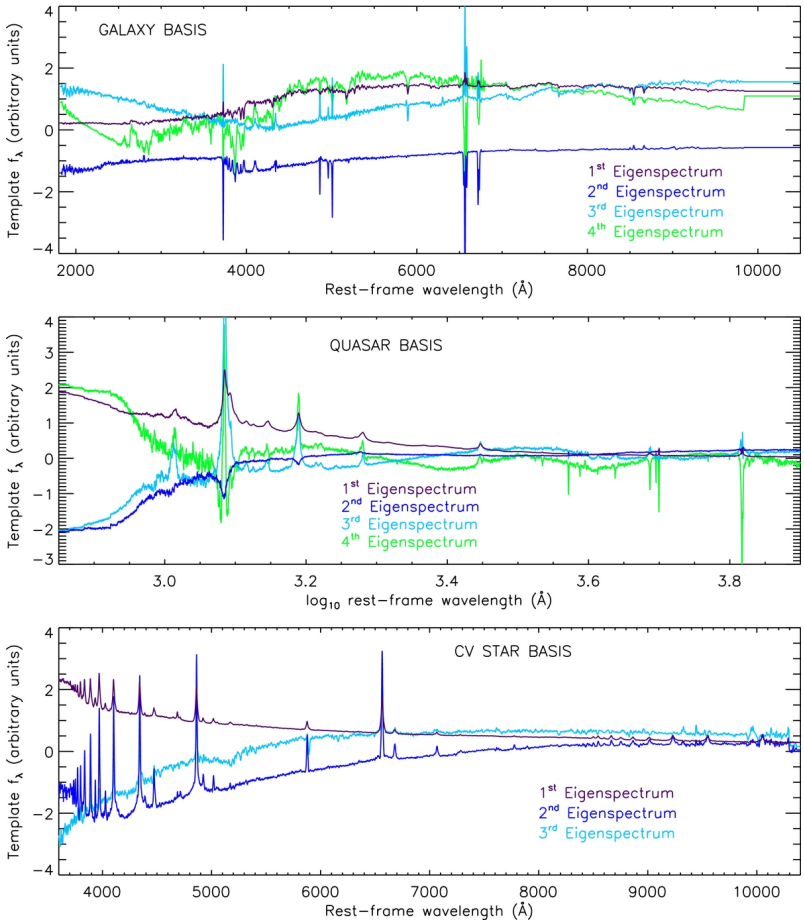


FIGURE 1.5: Basis BOSS redshift and classification template sets. The top graph is for galaxies, the middle for quasars, and the bottom for stars (Bolton et al., 2012).

by matching up the spectrum to a known atomic or ionic spectra, the difference between the observed-frame wavelength emission line(s) and the reference rest-frame emission line(s) is then used to calculate the difference in velocities, resulting in a redshift estimate (Vanden Berk et al., 2001; Hewett & Wild, 2010). The redshift retrieval is only the first step, they then need to be used in a classification model, an example is template fitting. The SDSS is a survey that currently uses such a technique Hutchinson et al. (2016); stars, galaxies, and quasars all have different redshift spectral templates with each major class having sub-classes of templates to distinguish the different types of each target. To assign a classification to an observed target; whatever template has the closest match to the target observation is the deduced label providing the error is not too large. How this error value is achieved can be seen in the next chapter, as well as a more in-depth explanation on template fitting.

The templates in the SDSS were originally based on earlier survey redshift measurements, with Figure 1.5 giving examples from an earlier SDSS release. The redshift templates for galaxies were made through sets of galaxies with redshifts between $0.05 < z < 0.8$, stellar continuum models were then applied as well as adding Hα, [N_{II}] and [S_{II}]

emission lines to the stacked spectra (Bolton et al., 2012), giving a full rest-frame wavelength view of $1900 - 9900\,\mathring{A}$. Quasar templates were also based off of earlier survey quasar redshifts, however were randomly selected as quasars are not understood to the same degree. The stellar templates were unusual as the SDSS spectrographs involved (explained further in the next chapter) were not built to target stars. Consequently the template creation was a hybrid affair, using multiple parameters from different surveys that would also give archetype sub-classes that matched up with the different stellar classes seen in Figure 1.2. These templates all provide decent measurements and are helpful in explaining what different approaches might be needed, but have since been upgraded with the updated software due to the SDSS advancing (Hutchinson et al., 2016).

Because the SDSS is such an extensive and meticulous astronomical survey, the data it has collected has been extensively used both alone and with other survey observations to advance our understanding of the Universe and objects within it. The analysis of the data has yielded estimations of the Milky Way's circular velocity (Xue et al., 2008), new measurements of the Hubble constant (Birrer et al., 2019; Cuceu et al., 2019; Said et al., 2020), insight into the history of star formation (Sánchez et al., 2019), the location of extrasolar planets (Manser et al., 2016), and much more. As new surveys are introduced, fresh techniques are performed which can benefit scientific areas other than astronomy. This is directly seen with the SDSS introducing *Panoptic Spectroscopy* (Kollmeier et al., 2017). The data is the foundation for this thesis and it will continue to be used in the future.

Chapter 2

The Sloan Digital Sky Survey dataset

Machine learning can also be used to classify SDSS data, but what data the machine learning model uses is incredibly important, thus this section is to explain in detail the data used in this work. The data used to train and test the models are a cleaned combination of selected SDSS and WISE data, the specific objects within which were chosen due to them having counterparts in both sets of data, essentially allowing more features for training and analysis (Clarke et al., 2020). They are both important sets of data within astronomy and have helped further astronomical research, some examples can be seen with: Norris et al. (2016) and Zhang et al. (2021).

2.1 Surveys

To perform a survey requires the correct instrumentation, as how else can any data be collected without the suitable mechanisms and procedures in place? The telescopes used for detecting and recording sources are invaluable in how astronomical objects are discovered, but these tools need to have precise usage and timing in order to map the correct area(s) of sky they are tasked with surveying. Before understanding the surveys and instruments for the SDSS however, it is necessary to understand the manner in which the data is collected and made available. The SDSS is split into five sections; SDSS I, II, III, IV, and V (York et al., 2000; Frieman et al., 2008; Dawson et al., 2013; Lundgren et al., 2015). Data is released almost yearly in cumulative *Data Releases*, with all data from the previous releases being included in the new set. SDSS I and II include the earliest data collections and releases up to DR7 comprise these collections, SDSS III was the evolution from this and is included in data releases 8 through 12, SDSS IV at its core is an updated version of III (Ahumada et al., 2020). The data releases for this one include 13 through to 16, and since it is still running it will also include DR17 once released. Once this phase of the Sky Survey has completed, it will move entirely onto SDSS V with fresh missions. In-fact, $SDSS\ V$ has already begun collecting data using three mappers: the *Milky Way Mapper* (MWM), the *Black Hole Mapper* (BHM), and the *Local Volume Mapper* (LVM) (Kollmeier et al., 2017).

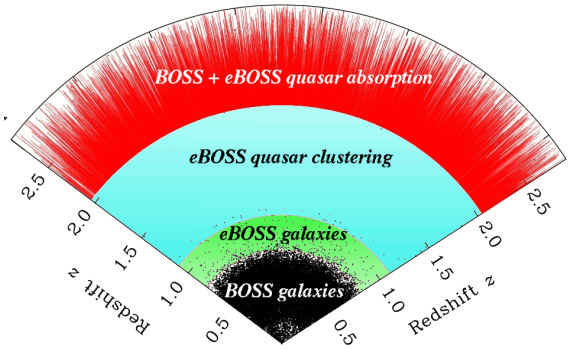


FIGURE 2.1: Simplified expectation of what eBOSS is attempting to observe (Marra et al., 2019).

2.1.1 SDSS

The Sloan Digital Sky Survey has observatories in two locations: *Apache Point Observatory* in New Mexico and *Las Campanas Observatory* in Chile, allowing the survey to cover both hemispheres. In these two locations there are three telescopes: the *Sloan Foundation 2.5m telescope* and the *NMSU 1m Telescope* in Apache Point, and the *Irénée du Pont Telescope* at Las Campanas. The first telescope was the Sloan Foundation telescope (Gunn et al., 1998) and this took images photometrically and spectroscopically as described in Gunn et al. (2006); the imaging part was retired with the advent of SDSS *III*. Currently all observations are taken spectroscopically for the three different missions in SDSS *IV* using two spectrograph designs. The three surveys are: the Apache Point Observatory Galactic Evolution Experiment (*APOGEE-2*), the extended Baryon Oscillation Spectroscopic Survey (*eBOSS*), and Mapping Nearby Galaxies at Apache Point Observatory (*MaNGA*). APOGEE-2 and eBOSS are extensions of previous surveys found in SDSS *III*: *APOGEE-1* and *BOSS*. Other operations that have since stopped include the *Legacy* survey, the *Supernova* survey, *SEGUE-1*, *SEGUE-2*, and *MARVELS* (Aihara et al., 2011). APOGEE-2 involves the search for stars within the Milky Way (Wilson et al., 2019), MaNGA was designed to observe the internal structure of nearby galaxies (Bundy et al., 2015), and eBOSS is the near five year observation of the Universe through itself and the subprograms TDSS and SPIDERS (Dawson et al., 2016). Figure 2.1 shows a rough idea on the redshift measurements taken by the eBOSS and BOSS spectrographs.

The data involved in this thesis originates from the BOSS spectrographs (Smee et al., 2013). These spectrographs are rebuilt and enhanced variants of the original SDSS spectrographs used in SDSS *I* and *II* and are used in the collection of data for eBOSS and MaNGA. APOGEE-2 uses its own dedicated spectrograph in the northern and southern hemispheres. The BOSS spectrograph has 1000 holes in each 7-square degree aluminium plate, each matched to a point in the sky containing an astronomical object or a blank area (for sky emission). Optical fibres are connected to these holes and a beamsplitter within is used to extract the blue and red wavebands for redshift measurements. Each optical cable has a 2″ diameter. The data is extracted from these cables with cameras,

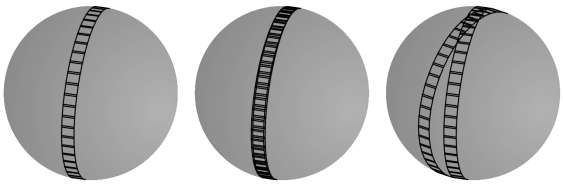


FIGURE 2.2: Representation showing WISE orbiting once (left), twice (central), and two separated by twenty days (right) (Wright et al., 2010).

with the blue cameras working at wavelengths of $3600 - 6350\,\mathring{A}$ and the red cameras at $5650 - 10000\,\mathring{A}$ (Dawson et al., 2016).

There are 1800 plates enabling a survey area of $9000\,\mathrm{deg}^2$; 200 fibres on average are used for TDSS and SPIDERS, so 800 are left for eBOSS. Furthermore 300 of the plates are dedicated to the Emission Line Galaxy (ELG) program, meaning a total of 1500 plates and $7500\,\mathrm{deg}^2$ are used (Raichoor et al., 2017). The program was designed with an expected limitation from the telescopes being located on the ground resulting in 50% weather efficiency. Thus roughly 5400 hours were dedicated to the eBOSS program over the four years (Dawson et al., 2016).

2.1.2 WISE

The WISE instrumentation consists of a single telescope orbiting the Earth in a low Earth orbit satellite, see e.g. Figure 2.2. It orbits across the North and South Ecliptic poles ($\pm 90^\circ$ longitude) at a mean height of 540 km above the ground and takes a measurement every eleven seconds in order to build a well defined survey. The mission is designated as a *MIDEX* or a medium class explorer mission. The telescope itself is a 0.4 m telescope with 4000000 pixels and a field of view of $47'$ recording in four infrared bands (Wright et al., 2010). The original mission for WISE only existed for ten months as afterwards the coolant ran dry. Figure 2.2 shows different frames of images for select orbit types to represent how the images are taken in orbit. The WISE mission has also been extended in NEOWISE (Myhrvold, 2018).

2.2 SDSS Data

The data from the Sloan Digital Sky Survey used in this work is selected from Data Release 15 (Aguado et al., 2019) and contains both labelled and unlabelled sources. The labelled data are spectroscopically observed sources, while the unlabelled data was photometrically observed and have no spectroscopic association. As mentioned, SDSS data releases are cumulative, thus this data release will contain all data from previous releases. The new data from this fresh release is specifically from the BOSS and SDSS spectrographs.

The SDSS measurements use five bands that were observed photometrically: u ($\lambda = 0.355\,\mu$m), g ($\lambda = 0.477\,\mu$m), r ($\lambda = 0.623\,\mu$m), i ($\lambda = 0.762\,\mu$m) and z ($\lambda = 0.913\,\mu$m).

They are all honed for certain types of sources (Richards et al., 2002) and have been deduced using the spectroscopic method mentioned in Hewett & Wild (2010). A sixth data point has been added to this set that distinguishes whether a source is more likely to be point-like or a resolved source. This was done using the following expression,

$$resolved_r = |psf_r - cmod_r| \tag{2.1}$$

and involves the $cmodel_r$ and the psf_r. The first of these in Equation 2.1; $cmodel_r$, is a magnitude derived from a composite flux model involving de Vaucouleurs fits and is now also considered to be an accurate representation of not only a galaxy's flux, but also a universal magnitude for all astronomical objects (Abazajian et al., 2004). The second variable, psf_r, is the instrumental point spread function in the red band - a choice used to represent points on a mapper. The comparison of these two is partly used in the determination of whether the source is a star or a galaxy and at the same time, gives another feature to use. The labelled spectroscopic data makes up 3,238,003 sources, out of a total 1231051050 sources from the photometric catalogue (this value includes repeat and contaminated observations). The photometric magnitudes exist in the feature data, with the spectroscopic data being used only to provide the target labels.

2.3 WISE Data

The WISE data used in this work is taken from the *WISE ALL-Sky Data Release*. The photometric data involves maps of the sky in four different wavelengths; W1 ($\lambda = 3.4\,\mu$m), W2 ($\lambda = 4.6\,\mu$m), W3 ($\lambda = 12\,\mu$m), and W4 ($\lambda = 22\,\mu$m). This data is from the full cryogenic mission phase, a single survey conducted by the telescope as it orbited the Earth. This data was taken from a source catalogue containing over 563 million sources with positional and photometric information for each of them (Cutri et al., 2012).

2.4 Combined Data

The datasets were matched between the SDSS and WISE to give 3,238,003 unique sources with spectroscopic labels (Clarke et al., 2020), all containing ten different features; $resolved_r$, bands u, g, r, i, z, and $W1$, $W2$, $W3$, $W4$. Each source has a corresponding label that represents one of: a *galaxy* (class 0), a *quasar* (class 1), or a *star* (class 2). These spectroscopic sources were used in the training, validation, and testing of the models within this thesis, the additional photometric data from Clarke's 2019 paper were not used. Overall the WISE set is matched to the SDSS set, meaning the SDSS is the higher priority of the two. This data is also all cleaned so data that is too unclear is removed.

Some sources from the SDSS dataset were removed before combining. These included: 21,086 sources that have more than one matching WISE source within five arcseconds, 130 sources where the cmodel fit failed, 116,026 sources where a *zwarning* flag

had come up that indicates a high signal-to-noise ratio, and 1,304 sources where the extraction of WISE magnitudes failed. Meaning overall 138546 sources were cleaned (90969 galaxies, 38890 quasars, 8687 stars), leaving 3,099,457 sources for training and testing the model (2,209,333 galaxies, 377,888 quasars, 512,236 stars). 1,789,194 of which were observed with the BOSS spectrograph and 1,310,263 with the SDSS spectrograph (Clarke et al., 2020). Thus the resultant class ratio resulted in roughly 70 : 10 : 20 for galaxies, quasars, and stars, respectively. Furthermore, these values are not using the extinction corrected values, but the uncorrected values, as Clarke found there was no noticeable increase in classification performance when using extinction corrected data.

Finally, the combined set of 3,099,457 objects, each with ten features was split into seen and unseen data randomly, using 309,946 sources as the seen data and (10% of the dataset) and 2,789,511 as the unseen data (90% of the dataset). This was deemed a sufficient number of data points for the active learning to take place on, as in Clarke et al. (2020), a value of 10% of the spectroscopic data was found to give the best performance compared to the amount of data. More data did not increase the performance in any significant way. This seen set of data was split in an 80 : 20 partition to give the training and validation sets. Overall leaving 247,957 objects for training and 61,989 for validation.

2.5 Current Classification for the SDSS

There are ongoing attempts to improve the classification for the SDSS releases, as the currently used *redmonster* model is only 90% accurate (Hutchinson et al., 2016). Like previous approaches in the SDSS observations, redmonster uses a template fitting model where spectra that fit the closest to the established template outlines of the classes are labelled as that template class. The template fitting model for redmonster utilises χ^2 fitting to estimate the confidence in assigning a template, which is based on minimising the following equation:

$$\chi^2 = \sum_{i=1}^{N} \frac{\left(d_i - \sum_{k=1}^{n} a_k x_{k,i}\right)^2}{\sigma_i^2}, \tag{2.2}$$

where, assuming N datapoints and a set of n basis vectors, x_k is the basis vector for a single physical template with $n-1$ polynomial terms, a_k is the coefficient for the basis vector, and σ_i^2 is the statistical variance of the ith pixel d (Hutchinson et al., 2016). Repeating this process across all trial redshifts gives a surface $\chi^2(\mathbf{P}, z)$, with $\mathbf{P}$ being the vector along the parameter-space of the template class. Spectra are considered to fit a class if the templates match and the error is low enough, this error is considered too great if the ($\triangle\chi^2_{threshold} = 0.005$). While Equation 2.2 works on a general basis, it was further customised and improved for different objects later on in the paper. The $\triangle\chi^2_{threshold}$ was also finely tuned to allow more objects to be classified, yet this comes at a cost of increased failure rates. The algorithm's foundations can be seen in Tonry & Davis (1979). This was also a marked improvement over the previous pipeline (Hutchinson et al., 2016), with the redmonster success rate being above 90% (for luminous red galaxies), an improvement of roughly 24% over previous models. The previous model for identification

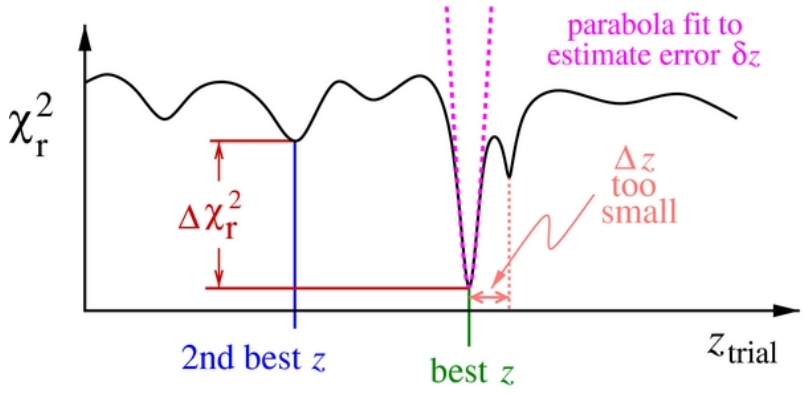


FIGURE 2.3: Illustration of a redshift algorithm from Bolton et al. (2012), using the best fit linear combination of basis spectra at each trial redshift value to determine the reduced χ^2 curve (black). The best redshift is where the global minimum (green) is and any minima with fewer than $1000 kms^{-1}$ between are considered the same (pink). The second best redshift is the second lowest point that is still considered separated (blue). Confidence is calculated using the difference in $\triangle\chi^2$ between the best and second best redshifts (red) and the error estimate is found with the curvature of the parabolic fit to the χ^2 curve at the minimum (magenta).

within the SDSS data releases was the *spectro1d* pipeline, in place since the early data release (Stoughton et al., 2002). It was also a template fitting model approached with the same general χ^2 minimum technique, though less refined and having less effective templates. Figure 2.3 shows how values from the χ^2 minimum technique were found in the earlier data releases.

Because of the original mission intention, WISE data on their own do not have a classification pipeline like the SDSS does (Wright et al., 2010). Instead WISE data is taken and classified in combination with other data (Kurcz et al., 2016).

Chapter 3

Machine Learning

Machine learning as an academic field began in the 1950s with Alan Turing's paper: 'Computing Machinery and Intelligence' (Turing, 1950), it was later deemed a topic of Artificial Intelligence in Dartmouth (McCarthy et al., 2006). Artificial Intelligence itself had been a concept without theory for years in fiction, notably associated with Isaac Asimov's short story *runaround* in 1942 (Haenlein & Kaplan, 2019). Additionally, discoveries beforehand, such as the mathematical model for a neuron in McCulloch & Pitts (1943), meant the foundations were laid for the topic to explode once the subject was acknowledged by the wider scientific community. Machine learning is now considered a research area within artificial intelligence and pertains to the idea of a computer program that has been trained to perform a specific (or non-specific) task.

There are different machine learning models for different objectives, these models differ in how they are trained and how they apply their algorithms, therefore can be applied to various situations. Some machine learning models are far more specialised for certain outcomes and others are more general, so the correct architecture should be used when able. The different architectures may also be able to take advantage of different hardware, allowing access to potentially faster calculations due to the way objects are stored or computed. Again promoting the idea that the correct choice of models is of major concern. For classification; *neural networks* and *random forest classifiers* are common models.

Machine learning models are trained with data that is similar to the real data the algorithm will be applied to, then tested with data for which the user knows the outcome, but that the trained model has not seen before. The training data can be split further into what is commonly known as training and validation data; validation data is also sometimes called test data but this can lead to confusion among readers (whether it is the code or a paper being read). So for practicality purposes within this write-up: the unseen data will be referred to as the *test* data and the *training* and *validation* data will be the aforementioned seen data. Training data in this case is the data the model looks at and augments certain values within to match. Validation data is used to understand whether the hyper-parameters of the machine learning model are optimal.

Within machine learning there are three leading paradigms; *supervised learning*, *unsupervised learning*, and *reinforcement learning*. It is easy to think of supervised and unsupervised learning as a spectrum where different approaches to these two types of learning have branched off. The naming refers to what the data contain: if there are pre-designated labels the model can map the inputs to then the model is 'supervised'. Two very common instances of this are *'classification'* problems and *'regression'* problems. Unsupervised learning involves a model that finds relationships and patterns between the inputs which otherwise might not be known, it is referred to as because there is no specific outcome the model is training towards. A common example of unsupervised learning is *'clustering'*, where similar inputs are grouped together (Soni Madhulatha, 2012a). The third type; *reinforcement learning* is completely different.

Reinforcement learning involves using feedback from the environment it is in, constantly learning from what is around it. The system is rewarded for specific actions and the goal is to maximise that reward, discovering the optimal path to a goal. This branch of machine learning is incredibly common in game design where it can be used to help or hinder someone at the right time, or even outmanoeuvre them. In fact Google's AlphaGo is an example of this kind of AI and has been developed to beat even the world's top *Go* player (Granter et al., 2017). Reinforcement learning also has applications in radio astronomy as seen in Yatawatta & Avruch (2021) and can be used for classification problems, however models using this type of learning will not use class specific labels and typically take longer than other archetypes due to the way they are designed.

Methods that are intermediate to those described above are denoted as *hybrid* learning methods, (e.g. *self-supervised*, *semi-supervised*, *multi-instanced*) and statistical inference methods (e.g. *inductive*, *deductive*, *transductive*). Learning techniques can also be applied that do not alter how the model works, but rather how the available data is selected, notably seen in *active learning*. The inputs for a model drastically change how a model works as it can have totally different aspects which is what leads to the effectiveness of techniques such as active learning. Inputs for machine learning models range from images (Barchi et al., 2020) to sounds (Bermant et al., 2019) or include only simple values as seen in this thesis.

3.1 Understanding a Model

Building and understand a machine learning model fully is a process that takes time and requires a significant amount of knowledge about the internal processes involved. There needs to be at least a basic grasp on a multitude of aspects; what task needs to be performed, what parameters are needed, how the model performs the task, and how to interpret the outcome at which the model arrives. The model can be designed and built from the ground up or through similar instances where another machine learning algorithm completes an objective akin to the current one. To begin, the function of the machine learning model must be made clear and as this paper is on classification, this functionality will be explained.

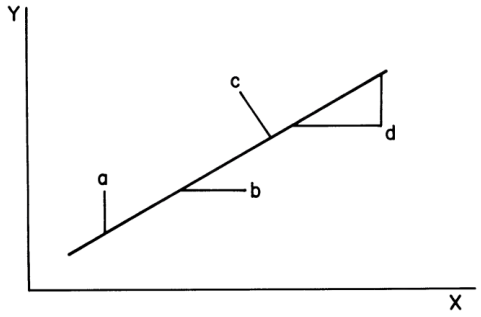


FIGURE 3.1: Sketch of different minimization methods for linear regression. a - $OLS(Y|X)$, where the distance is measured vertically; b - $OLS(X|Y)$, where the distance is taken horizontally; (c) OR, where the distance is measured vertically to the line; and d - RMA, where the distances are measured both perpendicularly and horizontally Isobe et al. (1990).

3.1.1 Regression

An overview of machine learning in relation to classification would not be complete without mentioning *regression*, sometimes referred to as regression analysis. This topic involves gauging the strength of relationship or relationships between a dependent variable and one or more independent variables, and is mostly used for prediction and projection through these relationships. *Linear regression analysis*, a form of regression analysis, is used a considerable amount in astronomy with multiple applications established over the years (Feigelson & Babu, 1992). Figure 3.1 shows four of the five methods entrenched in astronomical usage: *ordinary least squares* (both horizontal and vertical), *orthogonal regression*, and the *reduced major axis*. This image does not show the *OLS bisector* regression method (Isobe et al., 1990).

3.1.2 Classification

In the context of machine learning, classification is a regression problem with defined integer outcomes, otherwise known as *logistic regression*, it can be supervised or unsupervised. The scope of classification is far too broad to not think of it as a separate area. For supervised classification to work, the outcomes must be identified so a model knows what to predict. These outputs are placed into machine learning models as discrete integers (compared to the continuous variables found in regression problems), where the machine learning model must then match the inputs to the correct outputs. The trained model can then be used on other data and classifies this data based on the features seen in the training data.

3.2 Machine Learning Models

Different machine learning methods use different calculations to complete their tasks. Even models that are attempting the same task can have a wide range of algorithms

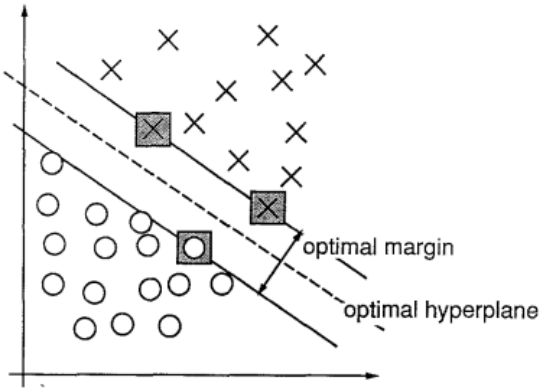


FIGURE 3.2: An example of a separable problem in a 2-dimensional space, the support vectors (grey squares) define the margin of largest separation between the two classes (Dimitriadis & Liparas, 2018).

which can end with the same result (although this is not always the case). The two classification approaches focused on in this thesis are *Neural Networks* and *Random Forest Classifiers* because both are established methods for classifying data and have been used to classify spectral SDSS data sets previously (Clarke et al., 2020; Bazarghan & Gupta, 2008). The underlying data was totally different in these cases. However before explaining the usage and choices behind the models used in this paper, other algorithms in classification deserve to be briefly mentioned.

Linear methods involve the premise that a linear amalgamation of the inputs describes the output accurately or as an approximation (Louppe, 2014). This is used in the linear regression method mentioned earlier and can be shown as:

$$\psi(x) = b + \sum_{j=1}^{p} x_j w_j \tag{3.1}$$

for a general method. In the case of a binary classification problem,

$$\psi(x) = \begin{cases} c_1 \;\; if \;\; + \sum_{j=1}^{p} x_j w_j > 0 \\ c_2 \;\; otherwise \end{cases} \tag{3.2}$$

is used. In both cases, ψ is the output of the model. To use the method in a multi-class setting would involve using multiple linear methods, making it relatively inefficient in comparison to others.

Support vector machines (SVMs) rely on linear methods, but use multiple linear models at once. With these linear models, a separation known as the margin is calculated by focusing on a model that is furthest away from all training data points. This gives the lowest generalization error (Cortes & Vapnik, 1995). The practice is much easier to visualise when looking at a 2-dimensional hyperspace, as shown in Figure 3.2. A basic SVM relying on a combination of support vectors can be given as:

$$\psi(z) = sign\left(\sum_{support\ vectors} \omega_i z_i \cdot z + b_0\right) \tag{3.3}$$

Where in Equation 3.3, the decision function $\psi(z)$ is defined by weights ω_i, dot-product between support vectors z_i and vector z, and bias b_0. Support vector networks can receive optimisations on top of this base function and there are two schools of thought to this; *primal* and *dual* (Abe, 2009), however the base function is all that needs covering here.

Nearest neighbour methods are a unique case as they do not always require training. When used, a nearest neighbour model will try to find training samples closest to the sample it is trying to predict or classify, then will deduce the label (Cunningham & Delany, 2020). An example that can be seen often is the *k-nearest neighbour algorithm* which can apply to both regression using

$$\psi(x) = \frac{1}{k} \sum_{(x_i,y_i)\in\mathcal{N}N(x,\mathcal{L},k)} y_j, \tag{3.4}$$

and classification using

$$\psi(x) = \arg\max_{c\in\mathcal{Y}} \sum_{(x_i,y_i)\in\mathcal{N}N(x,\mathcal{L},k)} 1(y_j = c). \tag{3.5}$$

Where $\mathcal{N}N(x, \mathcal{L}, k)$ is the nearest neighbour k to x and $\mathcal{L}$. The maximum argument within the classification equation denotes the majority class and is thus, the predicted class (Soni Madhulatha, 2012b).

Both linear methods and support vector machines can also be used in non-linear fashions, but this is beyond the scope of this thesis. All types mentioned have their uses, however for classification of SDSS data the focus is on the use of random forests and neural networks.

3.2.1 Random Forests

A random forest model is a method that relies on *decision trees*, a form of predictive modeling that grows from the input into the output, commonly described as 'top down' (Quinlan, 1986). A simple example of which can be seen in Figure 3.3. Like the previous models, they can be used for both classification and regression using the same method, only instead of having a defined output in the form of a class, as with the random forest classifier, a random forest regressor outputs continuous variables. Not all decision tree models use the same method, for example some trees are used as a boosting method (Freund & Schapire, 1997), therefore the actual decision tree technique being used should always be specified.

Random forest models contain separate decision trees where each tree is trained independently on a random set of training data. When classifying, the independent trees then collectively vote on the class by the number of leafs, as the trees will have arrived

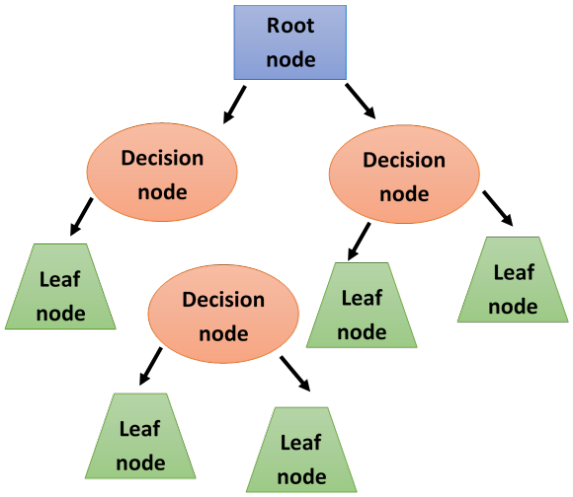


FIGURE 3.3: A simple general decision tree structure.

at the outcome from the input separately. Because different trees take random sets of different data and features, they are less likely to over-fit the data, also their variance and bias are minimised (Clarke et al., 2020). This is thanks to *bootstrap aggregation* (bagging) (Grimshaw et al., 1995), where an aggregate is taken from all trees in the model. Within the model, it is this is referred to as the *out-of-bag* score or the *oob score*. For regression problems, the average is the mean of the outcomes and for classification problems the average taken is the mode of classes (Breiman, 1996). Random forest classifiers are also useful for categorising over large volumes of data and are good with both binary and multi-class problems. With each node, a classifier or regressor will calculate how much information gain the input will give, and this is calculated using an impurity measure. These differ for classifiers or regressors. To measure the probability of a node within a random forest

$$p_{nk} = \frac{1}{N_n} \sum_{y \in Q_n} I(y = k) \tag{3.6}$$

Equation 3.6 is used. Where the probability p for node n is calculated based on k outcomes and I is the information gain. For classification, different impurity measures can be used, such as *gini impurity*:

$$H(Q_n) = \sum_k p_{nk}(1 - p_{nk}), \tag{3.7}$$

entropy:

$$H(Q_n) = -\sum_k p_{nk} log(p_{nk}), \tag{3.8}$$

and *misclassification error*:

$$H(Q_n) = 1 - \text{argmax}(p_{nk}) \tag{3.9}$$

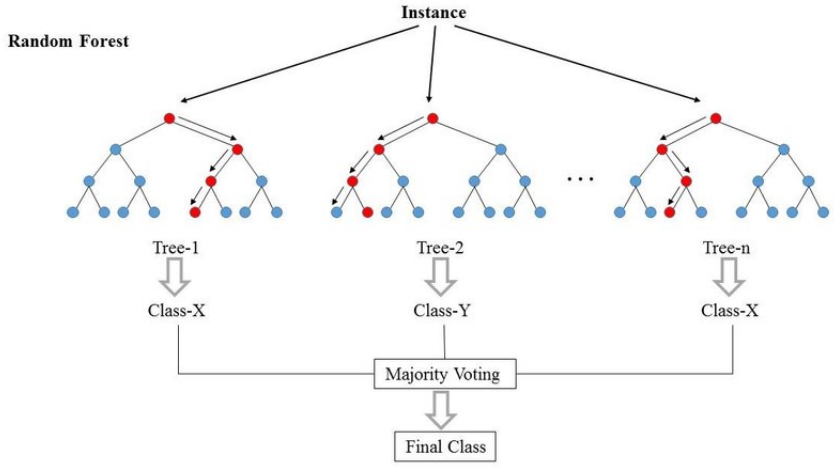


FIGURE 3.4: Simplified view of a Random Forest Classifier (Dimitriadis & Liparas, 2018).

These can all be found in Louppe (2014). Similarly, regressor impurity equations include *mean squared error*:

$$H(Q_n) = \frac{1}{N_n} \sum_{y \in Q_n} (y - \bar{y}_n)^2, \tag{3.10}$$

the *half Poisson deviance*:

$$H(Q_n) = \frac{1}{N_n} \sum_{y \in Q_n} (ylog\frac{y}{\bar{y}_n} + \bar{y}_n), \tag{3.11}$$

and the *mean absolute error*:

$$H(Q_n) = \frac{1}{N_n} \sum_{y \in Q_n} |y - median(y)_n|, \tag{3.12}$$

Although for the latter, the average is actually the median instead of the mean (Breiman, 2001; Hastie et al., 2009; Botchkarev, 2018).

Trees can also be 'pruned' to reduce the number of trees generated by selecting only a subset of trees from the original ensemble. This reduces the complexity of the overall model while typically increasing the diversity of classifiers within the random forest in relation to its size and increasing the performance. This also tends to reduce over-fitting, so this method is consider to be a form of hyper-parameter tuning. More can be found in Yang et al. (2012), as no time in this thesis was devoted to pruning due to the model already being specified.

3.2.2 Neural Networks

An artificial neural network is an attempt at reproducing a living creature's neurons in a computerised setting, officially begun as a subject of interest in 1943 (McCulloch &

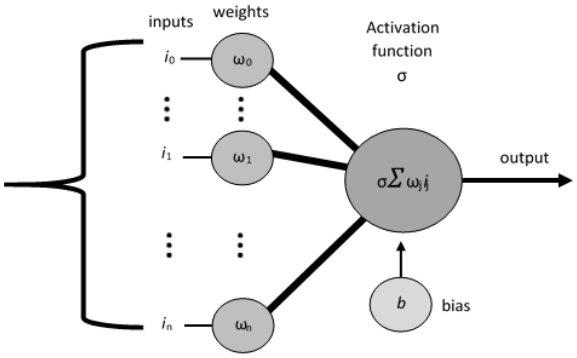


FIGURE 3.5: Simplified artificial neuron.

Pitts, 1943). It is a combination of individual *neurons* grouped together in *layers*, where they process information by activating neurons in each ordered layer. The neurons themselves are mathematical functions made up of different variables that receive inputs and produce an output. When data is processed in a single direction, the network is referred to as a *feedforward* neural network. These *multi-layered perceptrons* are activated by each neuron giving an output that the next layer's neurons use as an input, passing through all layers before finally reaching the output designated by the user. The first and last layers are called *input layers* and *output layers* respectively, the ones in between are called *hidden layers*, of which there can be any number and this is considered a hyper-parameter of the model (Nwankpa et al., 2018). When referring to feedforward networks, there is typically no *backpropagation* - where layers refer back to the previous ones in an attempt to adjust the weights in favourable directions (Rumelhart et al., 1986). The input layer is where the features that require classification are placed; the output layer is where the classes (or regression values) are produced; the hidden layers are where the neurons operate on the data and perform calculations.

Individual neurons are functions that result in a number, each one containing a specific weighting and bias required for the network to operate, the basic formulae for a neuron is:

$$a_0^{(1)} = \sigma\left(\omega_{0,0}a_0^{(0)} + \omega_{0,1}a_1^{(0)} + \cdots + \omega_{0,n}a_n^{(0)} + b_0\right), \tag{3.13}$$

which can be generalised as

$$a_0^{(1)} = \sigma\left(\sum_n \omega_{k,n}a_n^{(0)} + b_0\right). \tag{3.14}$$

A rough model of an individual neuron can be seen in Figure 3.5. Equation 3.13 and Equation 3.14 are called propagation functions. Where the activation of the current node $a^{(1)}$ is built from these values; σ being the *activation function*, ω represents the weights of the previous layer's neurons, a is the activation of the neurons in the previous layer, and b represents the bias (Nwankpa et al., 2018). The current node being calculated is shown with the superscript $a^{(1)}$ and the previous nodes are recognised by the superscript $a^{(0)}$ where they are also numbered with the subscript from 0 to n showing there are n in the

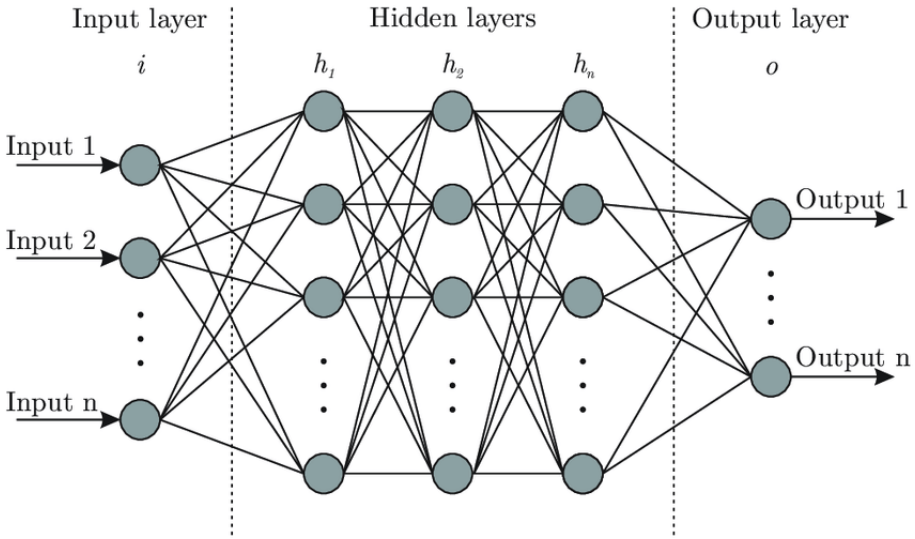


FIGURE 3.6: Simplified view of an artificial neural network (Martínez-Álvarez et al., 2015).

previous layer. The ω subscript holds similar connotations, the first subscript showing which neuron in the layer being calculated it relates to (k neurons) and the second showing which in the previous layer (n neurons). These are more easily represented in matrix form:

$$a_0^{(1)} = \sigma\left(\begin{bmatrix} \omega_{0,0} & \dots & \omega_{0,n} \\ \vdots & \ddots & \vdots \\ \omega_{k,0} & \dots & \omega_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ \vdots \\ b_k \end{bmatrix}\right). \tag{3.15}$$

Or can be summed up using more compact notation with:

$$a_0^{(1)} = \sigma\left(\mathbf{W}a^{(0)} + b\right). \tag{3.16}$$

When summarised in such a fashion, it is easy to think of each row being a specific neuron, while the columns are all the same terms in that layer, thus representing the full layer, this idea is represented graphically in Figure 3.6. Due to the number of neurons, it would be near impossible to calculate the weighting and bias of each of the previous neurons for every layer (even with few hidden layers it may be thousands of calculations), thus the neural network is trained so it learns what the weights and biases should be for each neuron. This is where training the neural network comes in. Neural networks are trained iteratively, the number of iterations are termed the *epochs* and again, is a hyperparameter. Generally the more epochs there are, the better the model is; however, after a certain point the increase in performance for each additional epoch becomes negligible. When done correctly the weighting and bias will be such that the desired outcomes appear a significant amount (how successful it is being determined by the user).

Finally, the activation function is a function that introduces non-linearity to the model (Varshney & Singh, 2021) and is not to be confused with the activation of the neuron. The activation function can be one of many types which are either linear or non-linear. There are many that apply to different situations however, so this paper will focus on three

commonly recognised methods; the *sigmoid* activation function,

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3.17}$$

the *softmax* activation function,

$$\sigma(x) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{3.18}$$

and the *rectified linear unit* activation function

$$\sigma(x) = max(0,x) = \begin{cases} x_i \;\; if \; x_i < 0 x_i \geq 0 \\ 0, \;\; if \; x_i < 0 \end{cases} \tag{3.19}$$

(or the ReLU - Krizhevsky et al. (2012)). A sigmoid function (3.17 is used in feed-forward neural networks, with much success in relation to binary classification or logistic regression problems (Nwankpa et al., 2018). The softmax function (equation 3.18) is used to compute probability distributions (between 0 and 1 and totalling 1 for each vector), and as such perform better with multi-class models. The softmax function is also utilized in most deep neural networks after the output layer, after computing the probabilities on the last output vector (Nwankpa et al., 2018). Finally the ReLU activation function - which is by far the most common method for neural networks (Ramachandran et al., 2017) as it gives a better performance than other activation functions and also preserves the properties of linear models with gradient-descent methods. The ReLU function has been expanded on many times in an attempt to improve it - with various degrees of success (Nwankpa et al., 2018), however this paper just uses the basic ReLU activation function.

Loss

Like random forests, a neural network must calculate how much error there is in order to make the correct predictions, this is done through a loss function and is not to be confused with the error rate. Other machine learning models use variants of this performance metric too (see the impurity measures for random forests), however they are typically specific for the machine learning model itself with loss being used as a general term occasionally (James, 2003). Because of the way neural networks are trained, their loss function (sometimes referred to as a *cost function*) is incredibly important to the performance (Zhao et al., 2015).

Loss functions differ depending on the task needed to be performed by the neural network. There are some general functions that many losses are based off; L_1 loss, L_2 loss, *hinge* loss, and *log* loss. There are other functions that are not based on these equations but they are not as common. L_1 and L_2 (as well as derivatives) are normally found in regression problems, they can potentially be used other areas (such as classification problems) to act as supplementary functions that perform specific tasks only they can do (Janocha & Czarnecki, 2017). Both L_1 and L_2 functions are also alternative names for

mean absolute error and mean squared error respectively, indicating that the L_1 function uses the mean absolute error between inputs:

$$\mathcal{H}_{\text{L1}}(p,q) = ||p-q||, \tag{3.20}$$

and the L_2 function uses the mean square error between inputs:

$$\mathcal{H}_{\text{L2}}(p,q) = ||p-q||^2. \tag{3.21}$$

The inputs for each element in this case are the true label p and outcome of the previous layer q. All are in relation to loss $\mathcal{H}$.

Hinge loss (or margin loss) is specifically for classifiers, however it is seen more frequently in relation to support vector machines because it looks for the maximum distance (Rosasco et al., 2004). It can be used in both binary and multi-class since it can take a one-vs-one or one-vs-all approach and can perform very well in both depending on the extension used (Janocha & Czarnecki, 2017).

$$\mathcal{H}(p,q) = \sum_j max(0, \frac{1}{2} - \hat{p}^{(j)} q^{(j)}) \tag{3.22}$$

Cross-entropy loss is the final one getting focus due to its actual usage in the chosen network. It is sometimes referred to as log loss, though in most cases this term is used when referring to binary classification systems and cross-entropy is used when referring to multi-class systems (Janocha & Czarnecki, 2017). In general terms it can be mathematically expressed for log and cross entropy loss with equations 3.23 and 3.24 (Janocha & Czarnecki, 2017).

$$\mathcal{H}(p,q) = -plog(q) - (1-p)log(1-q) \tag{3.23}$$

$$\mathcal{H}(p,q) = \sum_j \mathbf{p}^{(j)} log(\mathbf{q}^{(j)}) \tag{3.24}$$

These loss functions can all be calculated and averaged over to produce the average loss, which can then be used to view how effective the network is at performing the task at hand.

Dropout

Dropout is a technique exclusive to artificial neural networks and was developed relatively recently as seen in Hinton et al. (2012), given how old the concept of neural networks is (McCulloch & Pitts, 1943). It involves the gradual decrease of the number of nodes in a layer as the model iterates over time - thereby 'dropping' them. This supposedly increases the effectiveness of the leftover neurons, the reasoning behind this is that neurons are no longer training on super specific features and have to look at more general, useful features as other neurons are taken away (Hinton et al., 2012). Dropout also conveniently serves a secondary purpose of applying multiple different neural network architectures at once as different weights will be applied. Over the past eight years

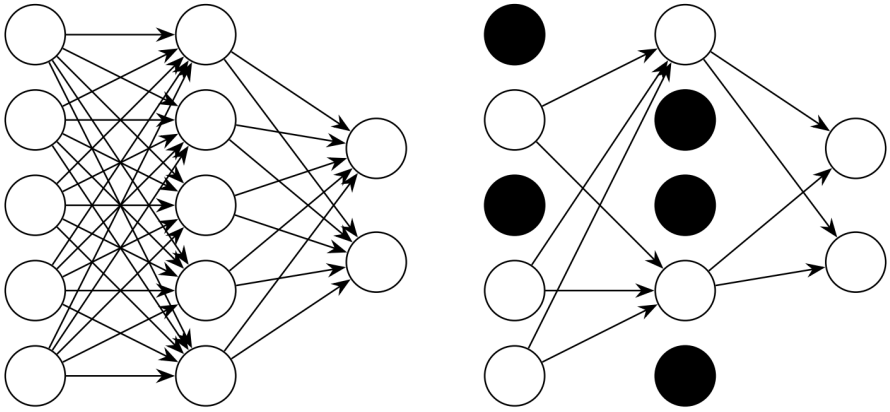


FIGURE 3.7: A comparison of a simple neural network with (right) and without (left) dropout (Labach et al., 2019).

dropout has evolved to include multiple dropout methods, each with a slightly different mathematical principle but most still involve dropping neurons during training. Standard dropout has also become standard practice in almost all deep neural networks due to its effectiveness in improving test accuracy and reducing overfitting (Labach et al., 2019).

The basic concept involves assigning a probability p for the neuron to be removed in training, then once it is trained fully, the entire network is used with the nodes being multiplied by the probability of them being omitted. Input and hidden layer probability are normally assigned different values due to the inputs being the actual features. In Hinton's original paper it was found that the probability of dropout for a hidden layer can be 0.5 while the input layer should be closer to 0 (Hinton et al., 2012). A simple version of this can be seen with figure 3.7. The output layer dropout should be omitted because then classification will not occur. The basic dropout equation for a layer can be seen with equation 3.25.

$$y = \sigma(\mathbf{W}x) \cdot \mathbf{m}, \quad m_i = Bernoulli(1-p) \tag{3.25}$$

where y is the layer output, σ is the activation function, $\mathbf{W}$ is the weight, x is the layer input, and $\mathbf{m}$ is the layer dropout mask. After training, the layer output becomes equation 3.26 as each element in the dropout is given by $(1-p)$. Other methods can alter the standard dropout as well, however standard dropout is written with these two equations 3.25 and 3.26.

$$y = (1-p)\sigma(\mathbf{W}x) \cdot \mathbf{m} \tag{3.26}$$

Much like hyper-parameters and loss functions must be optimised for the tasks at hand, the correct procedure of decreasing neurons must be found too. Sadly, dropout techniques do increase training times - by roughly two to three times according to Srivastava & et. al (2014). Some of the first advances in this sub-topic of dropout were *Dropconnect* and *fast-dropout* each with specific aims on improving aspects of the dropout method.

Dropconnect involves changing the weighting rather than getting rid of specific nodes making it a generalization of dropout (Labach et al., 2019). Fast-dropout aims to improve training times by approximating the dropout across the entire layer instead of node by node (Labach et al., 2019). These examples just show how effective the method was as they quickly developed, and the amount of methods do not stop there. Thus it is generally recommended to use even basic dropout because of the increase in performance metrics for the model.

3.3 Performance Metrics

Performance metrics are values used to determine how a machine learning model performs on unknown data. They are calculated from the statistics called during the training process, then these can be compared to statistics pulled from how the model performs on the validation data. Evaluating these will help understand how the model performs in specific areas and as a whole depending on what metrics are called. They can also be pulled from the test data as well to actually analyse how the algorithm performs on unseen data (thus supporting how it will appear in real life), however this should not be then used to alter the model unless performing a method that specifies it (e.g - reinforcement learning seen in Michie (1963)). There are many performance metrics which differ on what aspects of the model to look at and are called in various manners depending on the architecture. Some are used for different families of machine learning, others only working with binary models, and even ones that are specific to a single algorithm (Botchkarev, 2018).

For classification networks; seven common metrics are used (these are typically all built into the framework used when coding a model): *accuracy*, *error rate*, *precision*, *recall*, *specificity*, *sensitivity*, and *f1-score*. Classification accuracy is the total number of correct predictions over the total number of predictions (equation 3.28), it can also be multiplied by 100 to give it as a percentage. Error rate is the amount of incorrect predictions over the total number of predictions (3.30) and like accuracy, can be multiplied by 100 to give a percentage. The other metrics need more of an explanation as they are calculated with specific values taken from the classifier. These four specific values are: *True Positives*, *True Negatives*, *False Positives*, and *False Negatives*. The positive and negative terms refer to the model prediction and the true and false terms the correct observation. True positives and true negatives are essentially the correct results, the former shows the features that have been labelled correctly and the latter gives the features that have been rejected correctly. False positives and false negatives are at the other end as the incorrect results, with the former being the features that have been labelled incorrectly and the latter showing results that have been missed incorrectly. These incorrect results are based on type errors; type I errors and type II errors. The first being the rejection of a correct null hypothesis, the second being a failure to reject a false null hypothesis (Rothman, 2010).

The terms together can form a confusion matrix like table 3.1, which was based on Visa et al. (2011). Precision, specificity, recall, and the f1-score can now all be calculated

TABLE 3.1: A basic confusion matrix showing the different outcomes based on whether the predicted values are correct or incorrect with their classification.

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

with these terms - as well as the accuracy and error rate if needed (equation 3.28 and 3.30).

$$accuracy = \frac{correct\ predictions}{size\ of\ data} \tag{3.27}$$

$$= \frac{TP + TN}{TP + TN + FP + FN} \tag{3.28}$$

$$error\ rate = \frac{incorrect\ predictions}{size\ of\ data} \tag{3.29}$$

$$= \frac{FP + FN}{TP + TN + FP + FN} \tag{3.30}$$

Precision looks at how many correct predictions (TP) there are compared to the total number of predictions ($TP + FP$) made within the class, examining which class the model is better at classifying:

$$precision = \frac{TP}{TP + FP}. \tag{3.31}$$

Recall is how many correct predictions (TP) there are over the total number of correct labels there can potentially be ($TP + TN$):

$$recall = \frac{TP}{TP + TN}. \tag{3.32}$$

Specificity checks the fraction of data that is correctly labelled as not belonging to the target class ($TN + FP$):

$$specificity = \frac{TN}{TN + FP}. \tag{3.33}$$

Sensitivity measures the fraction of correct predictions that is accurately identified ($TP + FN$):

$$sensitivity = \frac{TP}{TP + FN}. \tag{3.34}$$

Lastly, the F1-score is used to combine both precision and recall into one metric using the mean of both. There are other ways to combine them yet the F1 score is a commonly accepted method (Hossin & M.N, 2015):

$$F_1 - score = \frac{2TP}{2TP + FP + FN}. \tag{3.35}$$

Equations 3.28 to 3.35 show how the performance metrics are calculated. These do not

only apply to binary cases, but to multi-class tasks as well. This can by done by using a method mentioned earlier for the hybridisation of binary and multi-class systems; the one-vs-all approach. Simply take each specific class and view the true positives as correct for that class, the true negatives as correct if they are labelled another class, false positives as any data that has been incorrectly labelled as the class being analysed, and false negatives as any of this class that have been incorrectly assigned another label. For both binary and multi-class cases, these summations per class can be averaged into a use able performance metric for the overall model or they can be looked at individually. To average them, take the mean of each class.

Just having these metrics does not mean the model is understood however, so for clarity some generalisations have been used to explain them further. A high accuracy would mean the model is correct in most cases, a low accuracy would of course mean the opposite. Error rate is essentially the reverse, a high error rate means the model is not labelling correctly (and a low error rate must thus mean the reverse). These do not show the whole picture though as the average accuracy and error rate could be skewed by imbalanced classes, where one class is being consistently labelled correctly or incorrectly at a higher rate than another. Hence the others must be evaluated and are done so on a per-class basis. A low precision in a class would mean the model is not correctly identifying many inputs, a low recall shows an input belonging to a class is being misinterpreted as another class, low specificity shows that many values will be incorrectly labelled into the class being assessed, and low sensitivity means the model is not placing enough of the correct features into the class. A high amount of any of these features will obviously result in the converse meaning. The F1-score is slightly different as it is related to both the precision and recall; a low F1-score however will mean both poor precision and recall. Having access to all these metrics is helpful in identifying problems with the dataset or model. Other machine learning models will not use the same metrics and need others to show their performance, these could be regression metrics, ranking metrics, or others (Botchkarev, 2018).

3.4 Considerations when training models

3.4.1 Binary vs Multi-class

There are two types of classification; *binary classification* and *multi-class classification*. The first one is very simple being that there are two target groups and if an input is not labelled as the first one, logic dictates it must be the other and therefore those inputs can be associated with the second one. This is mostly for models that compare two different objects. Can the model identify the difference between a dog or a cat for instance? Multi-class models instead may look at any number of outputs (giving a much wider reach for what a classifier is needed to do). Using the same example as before, it may try to classify whether an animal is a dog, a cat, or a platypus (really any number of classes - or

animals in this case). Furthermore there is almost a hybridised form of binary and multi-class classification, where a one-vs-all approach can be taken (compared to a one-vs-one case with binary and an all-vs-all with multi-class). This is more to do with the logic behind what needs to be classified, again using the set example; the model could decide on whether an object is a cat - only instead of looking at all different types of animals the model could identify features that are specific to a cat and then features that decidedly invalidate an object as a cat.

3.4.2 Class Imbalance

Another criteria to think about is if the classes available for the training data have the same or a similar distribution; estimating the outputs to be balanced or imbalanced. It has always been a problem for classification models as models will be training to identify the features of one class much more than another - leading to a potentially high accuracy for the largest available label, a model could also misidentify the certain features as only belonging to one class as it lacks enough evidence to point it to another. Thus overfitting can occur where a model specifically fits too closely to the training set, meaning the model cannot generalize when looking at unseen data. Underfitting also may appear in the case of the latter, and the model will not be able to even accurately predict data it has seen before (the training set). To be referred to as a balanced dataset, the available target classes do not necessarily need to have exactly the same amount in the training set - but must have a similar amount. As described in Krawczyk (2016), a ratio of $1:4-1:100$ is considered a slight imbalance while upwards of a ratio $1:100$ is seen as a severe imbalance. Then in the same paper are ratios of $1:1000$ (or more) as real world examples of extremely severe imbalanced data sets. Class imbalance can certainly appear in astronomy depending on what the original survey was intended for, however there are not many examples of such severe imbalance. There are certain methods of mitigating class imbalance: algorithmic, data-preprocessing, and feature selection (Longadge & Dongre, 2013), one of which is used in this paper: active learning. Furthermore a model can be tuned to avoid or reduce the problems caused by class imbalance through its hyper-parameters.

3.4.3 Hyper-Parameters

Hyper-parameters are any variables associated with a machine learning model that are not optimised through training, but may be defined by the user. The number of inputs, outputs, and even how the model calculates classification are all parameters that may be changed to increase performance and are set before the model is trained. This task is known as *hyper-parameter optimisation* and should increase the effectiveness of the model by changing the aforementioned variables or others. In most cases, some hyper-parameter tuning is required to optimise the model for the task at hand (Wu et al., 2019).

There are multiple approaches to hyper-parameter optimisation and these are split into two groups: *manual searches* and *automatic searches*. Manual searches typically require

knowledge of how the system works and an idea of general ideal conditions for different architectures - an example is Masters & Luschi (2018). For automatic searches the traditional method is a *grid search*; looping through every possible combination (within reason) of parameters individually until the best ones are found (Wu et al., 2019). Of course this is an incredibly time consuming and inefficient process so other methods have been put forward that either make the grid-search more efficient or use a different way of finding the best parameters altogether (Wu et al., 2019). Other automatic searches include: *Bayesian Optimization*, *Particle Swarm Optimisation*, *genetic algorithms*, and more. Because manual searching is used in this paper due to prior knowledge and exploration, these optimisation styles will not be explored. For both manual and automatic searches, it has been found that certain hyper-parameters affect the model to a much greater extent, meaning they are the ones to focus on.

This can occur when a model is underfitting on a label that does not have enough information - simply remove that output or mix it with another class. The opposite is also true, perhaps a model is also overfitting based on the features, an option is to lower the amount of inputs available for the model to train on. Of course these two solutions are not optimal as they are tampering with the data, so instead the other hyper-parameters are looked at, though these may differ from model to model. To know when to change these parameters, *performance metrics* of validation data is looked at and compared to the training metrics.

Chapter 4

Active Learning

Active learning is a growing technique for machine learning, it involves selecting data-points for training by querying a user or an algorithm to find the most appropriate data to train on next - supposedly finding better features to train on thereby not needing as much data. Essentially making the model training more efficient. Active learning is thus a form of feature selection mentioned in the previous chapter.

The usage of active learning stems from two sets of logic; data may be difficult to obtain - increasing the importance of each point in the dataset and what is done with it, or the feature set in training is simply too large and needs to be cut down without the model losing performance. This type of learning has the potential to speed up training some machine learning models due to training with less data - this may not always be the case however as a model needs to be trained each time for the active learning to take effect (unless using a saved model). Active learning is typically very greedy when it comes to computation cost, but the benefits can outweigh this expense.

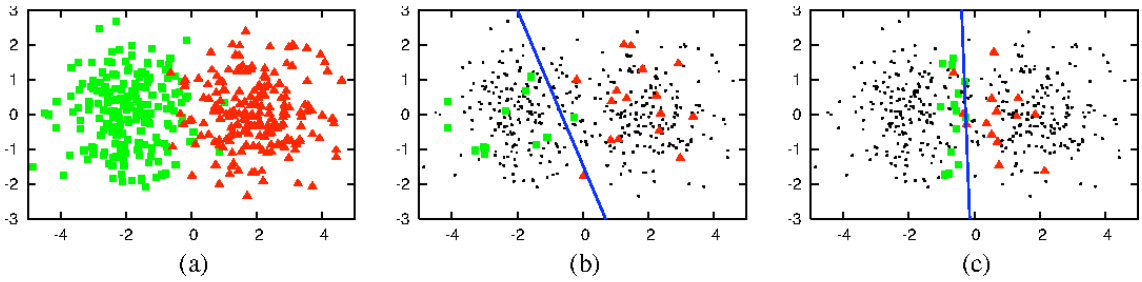


FIGURE 4.1: An example of pool-based active learning. *(a)* shows a data set of 400 instances, evenly sampled from two class Gaussians represented as points in 2-dimensional space. *(b)* shows a model trained with 30 labeled instances randomly selected from the original domain. The line is the decision boundary of the classifier results in a 70% accuracy. (c) shows a logistic regression model trained with 30 actively queried instances using uncertainty sampling with a resulting 90% (Settles, 2009).

4.1 Active Learning Methods

Various active learning strategies have appeared over the past years due to the major development machine learning has seen as a field. These have their uses in different forms of classification and regression (Joshi et al., 2009; Konyushkova et al., 2017). Though this thesis is on the identification of astronomical bodies given a dataset of values, thus focuses on classification. Like machine learning models, there is no one technique that is inherently better than another for every task, so each must be analysed and compared depending on the context.

There are three major paths of active learning; *Membership Query Synthesis*, *Stream-Based Selective Sampling*, and *Pool-Based Sampling*. Membership query synthesis occurs when the model asks for specific features as part of the training, relying on so called oracles (they may be human or artificial) to define the next inputs (Chen et al., 2016). Stream-based selective sampling analyses each unlabelled data point individually and looks at how promising the point is by viewing the parameters (Song, 2016), filtering out useless unlabelled data before training with them (thereby saving computational power) and increasing the efficiency of what data the model is trained with (Lewis & Gale, 1994). Pool-based sampling is the third avenue that can be taken and is especially useful with large datasets (Lewis & Gale, 1994). The unlabelled data is fitted to the model and given a score on how confident it is, the querying method chooses the unlabelled data point(s) that match the confidence score (Lewis & Gale, 1994; Wu, 2018).

Algorithm 1 Active Learning

1. Active learning begins with a small labelled training dataset $\mathcal{L}_\tau$ and a large pool of unlabelled data $\mathcal{U}_\tau$, with $\tau = 0$.
2. A classifier f_τ is trained using $\mathcal{L}_\tau$.
3. A querying method is used on $\mathcal{U}_\tau$, choosing the next feature(s) x^* to be annotated.
4. The new feature(s) x^* is given a label(s) y^* by an oracle, $\mathcal{L}_\tau$ and $\mathcal{U}_\tau$ are updated.
5. τ is incremented and steps $2-5$ are looped until the predefined conclusion.

The methods described in this chapter are all pool-based sampling methods for consistency and to keep the algorithms somewhat alike . Thus all follow the same process: the model is given a pool of labelled, $\mathcal{L}_\tau$, and unlabelled, U_τ, data all from the training set (different to the seen and unseen data). The features, x, and labels, y, are available in the labelled data for the model to be trained on so the model can be trained initially. This labelled dataset can be as small or as large as needed, indeed the seed size can be considered to be a hyper-parameter. So the model is trained first on the labelled data, the unlabelled data is fitted by the classifier and the query algorithm will calculate which are the ideal features to train on next and move these chosen features (x^*) with the associated labels (y^*) across. The classifier is then optimised again using the incremented labelled set and the loop continues until the model has been sufficiently trained or a limit established beforehand has been reached (Konyushkova et al., 2017), a summary of general active learning can be seen with algorithm 1. The difference in how pool-based sampling

can make a difference is displayed in figure 4.1 - exhibiting randomly selected data points alongside uncertainty sampling.

4.1.1 Uncertainty Sampling

Uncertainty sampling is the selection of data points that the model is least confident in classifying, and thus the most epistemically uncertain. It is a relatively simple active learning query with regard to both computational power and the algorithms used to define it, and yet it is still remarkably effective. There can be different ways of finding how uncertain a model is, but a popular method is selecting samples that maximise the entropy, $\boldsymbol{H}$, of the probability distribution over all classes. Because this method is very frequently wielded ,the term *entropy measure* has been used to describe this query (Joshi et al., 2009). The following equation is used to describe it mathematically,

$$x^* = \underset{x_i \in \mathcal{U}_{\ll}}{\arg\max}\, \mathcal{H}\left[p_\tau(y_i = y|x_i)\right]. \tag{4.1}$$

Equation 4.1 is choosing the data point(s), x_i, with the maximum entropy, $\mathcal{H}$, from the available pool of unseen features, U_τ. The entropy itself is the log loss of the least certain probability, p_τ, in which the correctly labelled points, $(y_i = y)$, match the features, x_i. The equation can also be simplified when classifying binary targets: instead of calculating the entropy fully, the probability of one label can be taken and then the minimum argument is taken from this value minus 0.5 such that

$$x^* = \underset{x_i \in \mathcal{U}_{\ll}}{\arg\min}\, \left[p0_\tau(y_i = y|x_i - 0.5)\right]. \tag{4.2}$$

Uncertainty sampling as a topic has expanded over the years, in effect becoming a sub-genre of active learning, leading to an exploration into the problems associated with the sampling method. Yang et al. (2015) is an exploration into uncertainty sampling with diversity maximisation, where there is an added variable that looks at how similar a feature is to a previous one thus measuring it's 'informativeness'. A major problem that comes up is class imbalance, which uncertainty sampling supposedly deals with. But, when the classes are severely imbalanced, the choices made by such a simple algorithm still might not be the correct ones to train on, instead there could be other variables that influence the selection process that should not be ignored.

4.1.2 Best vs Second Best

The best-vs-second-best approach can be considered as similar to uncertainty sampling due to approximating the entropy in classification uncertainty; however, this method adds an additional factor into the algorithm in order to increase performance. The method first takes into account the predicted label of the targets, and uses the difference in probability between the most likely class and the second most likely class. The data point with the smallest difference is chosen to be labelled, essentially giving data points the

model is least confident in when distinguishing between the two most likely classes, or as Joshi's paper has described it: "a more direct way of estimating confusion about class membership from a classification standpoint" (Joshi et al., 2009). Mathematically,

$$x^* = \arg\min_{x_i \in \mathcal{U}_\tau} p_\tau\left(\hat{y_1}|x_i\right) - p_\tau\left(\hat{y_2}|x_i\right). \tag{4.3}$$

Where the minimum difference is taken from the probability of the most likely class ($p_\tau\left(\hat{y_1}|x_i\right)$ and the probability of the second most likely class $p_\tau\left(\hat{y_2}|x_i\right)$. This type of active learning should be an improvement over uncertainty sampling - especially when more classes are involved, as the query then has more information to draw on in terms of which classes are less important (Joshi et al., 2009). It is sometimes referred to as smallest margin sampling (Scheffer et al., 2001).

4.1.3 Variance Reduction

Variance reduction does exactly what its name suggests: the algorithm chooses unlabelled data that reduce the variance of the model. Focusing on the probabilities of the unlabelled dataset, the variance is calculated and the datapoint(s) with the maximum variance are chosen. Variance is one of the components of generalization error, therefore minimizing it will indirectly minimize the generalization error also. The maximum variance is found through one minus the maximum probability, p_τ, of each unlabelled point, as one is the total sum of all probabilities. The largest remaining value(s) will thus be indexed for labelling. It can be described with:

$$x^* = \arg\max_{x_i \in \mathcal{U}_\tau} \left(1 - p_\tau(y_i = y|x_i)\right). \tag{4.4}$$

4.2 Learning Active Learning

Learning Active Learning is an especially complex and greedy method of active learning. It is a form of pool-based querying like the aforementioned active learning methods, only the querying is more elaborate. A problem found with most active learning processes is that they do not take into account many or all elements related to the task. Learning Active Learning attempts to resolve this issue by at least taking into account multiple factors in an attempt to minimise bias. The Learning Active Learning method used here is adapted from one of the methods mentioned in Konyushkova et al. (2017). The *Independent LAL* approach has been tuned for the purpose at hand. Before the active learning can be used however, a foundation needs to be laid. Learning active learning views choosing the correct unlabelled data to add to the training set as a regression problem, where the best unlabelled data to train on is a summary of different elements that fit the regression model closest. It can be thought of as a regression function predicting "the potential error reduction of annotating a specific sample in a given classifier state" (Konyushkova et al., 2017).

4.2.1 Monte Carlo Data

Before being able to use the LAL regressor, the actual regression model must be trained. To do this a representative dataset that is similar to the data in the classification problem is built using data from that very classification problem. Firstly a number of data points from within the original data, $\mathcal{D}$, are selected as a labeled set, $\mathcal{L}_\tau$, and an unlabelled set $\mathcal{U}_\tau$. τ serves as an indicator for the initial set. A classifier, f, is trained on $\mathcal{L}_\tau$ and then used as a function, f_τ, to predict the output of features, x', from a test set, $\mathcal{D}'$. From this the loss l_τ is estimated and the classifier's parameters, $\phi_n = (\phi_n^1, \ldots, \phi_n^k)$ of dimension K, are saved. New data points, x, from $\mathcal{U}_\tau$ are picked randomly and contain R parameters, $\psi_x = (\psi_x^1, ..., \psi_x^R)$ that do not contain the original features x. A new labelled set is made, $\mathcal{L}_x = \mathcal{L}_\tau \cup x$, and f is retrained as f_x. The new classifier allows measurement of the test-set loss, l_x. The difference between the two losses can now be calculated as $\delta_x = l_\tau - l_x$ and recorded. The learning state is described as a vector, $\mathcal{E}_\tau^x = (\phi_n^1, ..., \phi_n^k / \psi_x^1, ..., \psi_x^R) \in \mathbb{R}^{K+R}$, where the elements depend on the datapoint, x, and the state of the classifier, f_τ. This second part that evaluates the test loss is then repeated to build the Monte Carlo data.

The parameters for x should be specific to the classifier type depending on the base issue the classifier is solving and what machine learning architecture is being used.

Algorithm 2 Monte Carlo Data

1. A seen and unseen data set - $\mathcal{D}$ and $\mathcal{D}'$ are made.
2. $\mathcal{L}_\tau$ and $\mathcal{U}_\tau$ are initialised.
3. A classifier f_τ is trained.
4. A test set loss l_τ is estimated.
5. State parameters $\phi \leftarrow (\phi_\tau^1, \ldots, \phi_\tau^K)$ are calculated.
6.

for $t = 1$ to T: **do**

6i. Random point $x \in \mathcal{U}_\tau$ is chosen.
6ii. A new labelled dataset is formed $\mathcal{L}_x \leftarrow \mathcal{L}_\tau$
6iii. Datapoint parameters $\psi \leftarrow (\psi_x^1, \ldots, \psi_x^R)$ are calculated.
6iv. A new classifier f_x is trained.
6v. A new test set loss l_x is estimated.
6vi. A loss reduction $\delta_x \leftarrow l_\tau - l_x$ is computed
6vii. $\mathcal{E}_t \leftarrow (\phi_\tau^1, ..., \phi_\tau^k / \psi_x^1, ..., \psi_x^R)$, $\delta_t \leftarrow \delta_x$

end for

7. $\Pi \leftarrow \mathcal{E}_t$, $\triangle \leftarrow \delta_t : 1 \geq t \geq T$
8. A matrix of learning states $\Pi \in \mathbb{R}^{T \times K+R}$ with a vector of reductions in error $\triangle \in \mathbb{R}^T$ is returned.

4.2.2 Building the Learning Active Learning Regressor

Once the representative data has been created, the learner can be trained on this data. This can be a simple process in just building a regression machine learning model and fitting it to certain features chosen by the user (ϕ and ψ). An in-depth version of this independent active learning process is seen in algorithm 3. The regression model in theory

Algorithm 3 Monte Carlo Data

1. A seen and unseen data set - $\mathcal{D}$ and $\mathcal{D}'$ are made.
2. $\mathcal{L}_\tau$ and $\mathcal{U}_\tau$ are initialised.
3.

for τ **in** $(\tau_{min}, \ldots, \tau_{max})$ **do**
    3i.
    for $q = 1 \rightarrow Q$ **do**
        3ii. $\Pi_{rq}, \triangle_{r,q} \leftarrow$ Monte Carlo Data
    end for
end for

4. $\Pi, \triangle \leftarrow \Pi_{rq}, \triangle_{r,q}$
5. A regressor $g : \mathcal{E} \rightarrow \delta$ is trained on data $\Pi, \triangle$
6. The LAL regressor is $\mathcal{A}(g)$ is constructed with equation 4.5
7. The LAL regressor g is returned.

is able to to detect how the real feature sets will increase performance through selecting the datapoint with the highest predicted error reduction at that iteration, τ. This can be done by simply taking the maximum predicted value(s) from the regressor, g, much like the queries in the above subsection, otherwise written as

$$x^* = \underset{x_i \in \mathcal{U}_\tau}{\arg\max}\, g\left(\phi_\tau, \psi_x\right)) \tag{4.5}$$

(Konyushkova et al., 2017).

4.2.3 Making a more general Learning Active Learning Regressor

While the general algorithm seen in Algorithm 3 was used to make the regressor. The individual features used need to be adapted for extended usage. The regressor in Konyushkova et al. (2017) was initially designed around a binary classification system (though it did work on other data), meaning some of the regressor's features should only be usable in a similar situation. One obvious example of this is prediction variance compared to the probability of a Class 0 result. Instead this had to be changed into the prediction variance related to the unlabelled testing. The features investigated of course differed for the random forest classifier and the neural network, and they also had to be general enough to both work on the Gaussian clouds and the SDSS dataset, thus not focus on any one particular class. They could still count as unoptimized though.

Random Forest

This list denotes the order and features for the learning active learning when used within a random forest. Items 3, 4, 5, 6, and 7 are all averaged values taken from the different trees within the model. Items 1 and 2 are specifically for each prediction. This regression model is suitable for work on the Gaussian clouds and the SDSS dataset.

1. most likely class
2. prediction variance compared to most likely class
3. size of the training set

4. oob score
5. variance of feature importances
6. variance of forest
7. average depth of trees

Neural Network

This list is the same as above, showing the order and features for the learning active learning but for a neural network instead. Items 3, 5, 6, and 7 are averaged values - differing from the random forest model as the loss can be shown for each prediction. Again, the regression model can be used for the Gaussian clouds or the SDSS dataset.

1. most likely class
2. prediction variance compared to most likely class
3. size of the training set
4. cross entropy loss
5. variance of loss
6. average variance of output weights
7. mean of summed output weights

4.3 Active Learning on Gaussian Clouds

To demonstrate and compare the active learning methods described above and how effective they are in different situations a similar experiment to Konyushkova's Gaussian cloud showcase was performed. A major difference is the amount of active points chosen and the size of the training and testing sets, to more accurately reflect what will be happening with with the full SDSS datasets.

Four different scenarios involving Gaussian clouds were created; *balanced binary Gaussian clouds*, *balanced trinary Gaussian clouds*, *unbalanced binary Gaussian clouds*, and *unbalanced trinary Gaussian clouds*, see Figure 4.2. These types of Gaussian clouds were chosen because they represented many possible outcomes for the active learning methods and as they get more complex they gradually mimic a somewhat similar classification structure to the SDSS data, with the trinary systems representing a multi-class system. The unbalanced clouds have a ratio of $1 : 7 : 2$ to simulate the disparity within the SDSS dataset. Furthermore they have different covariance structures to show how different active learning procedures perform under other circumstances. However that is not to say the clouds are perfect substitutes for the cleaned SDSS data, as they only contain two features per point. What it does enable though, is a faster showcase as there are fewer features to train on.

As well as using these methods, a control query was used. This control query returned random values and if these values matched the index of an unlabelled data point, it would be moved to the pool of labelled data. The results in theory should show whether active learning is actually useful or if the model just needs more data to be improved. This is referred to as *random sampling*, and should match the proportions of the datasets.

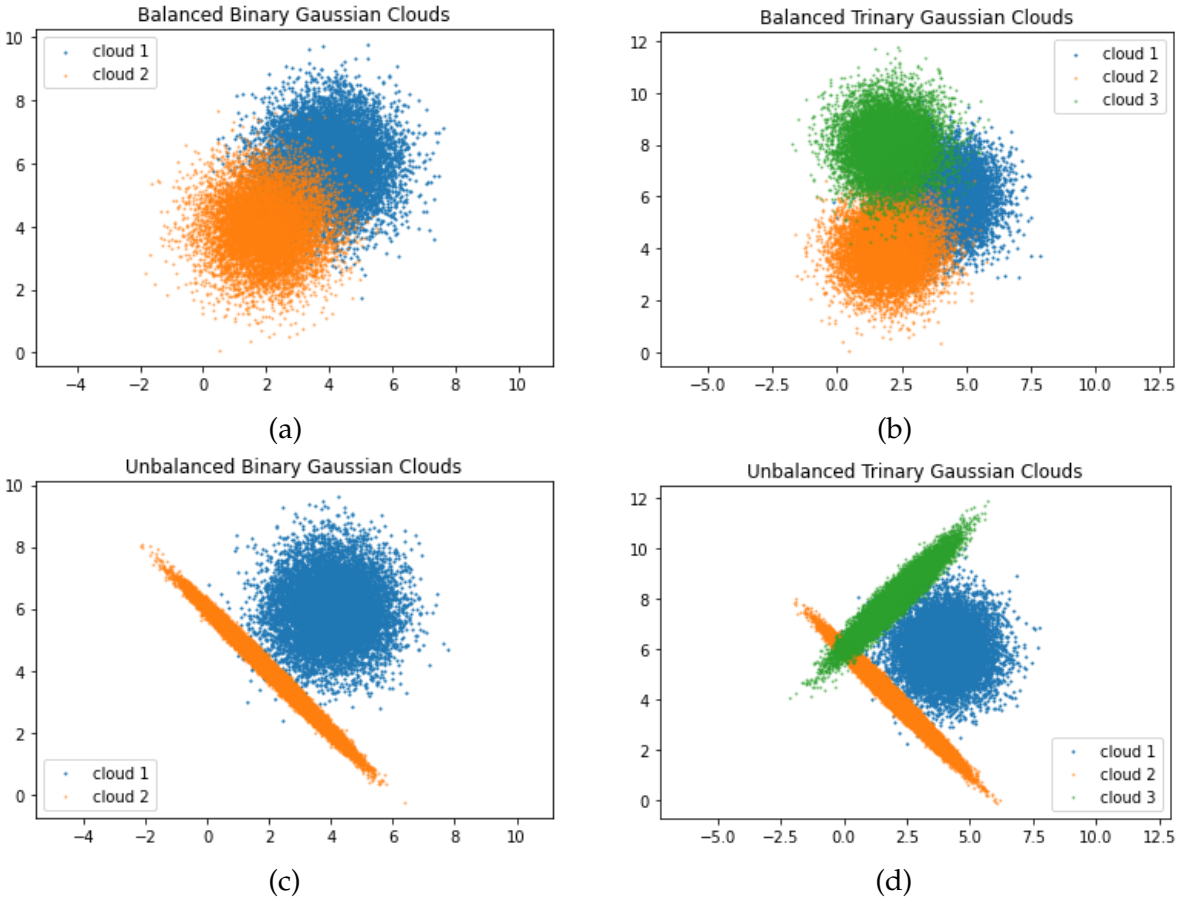


FIGURE 4.2: Plots of the features involved in the Gaussian clouds. Labelled: (a) for a balanced binary dataset, (b) for an imbalanced binary dataset, (c) for a balanced trinary dataset, and (d) for an imbalanced trinary dataset.

TABLE 4.1: Accuracy scores for each Gaussian cloud dataset - trained on the random forest.

Features	Average Accuracy
balanced binary	0.912
unbalanced binary	0.999
balanced trinary	0.880
unbalanced trinary	0.992

TABLE 4.2: Class specific performance metrics for each Gaussian cloud dataset - trained on the random forest. The precision, recall, and F_1-score are shown for all classes.

Gaussian	Precision			Recall			F_1-Score		
Clouds	1	2	3	1	2	3	1	2	3
balanced binary	0.908	0.917	–	0.917	0.908	–	0.912	0.912	–
unbalanced binary	0.999	0.999	–	0.991	1.000	–	0.995	0.999	–
balanced trinary	0.835	0.904	0.900	0.834	0.894	0.910	0.835	0.899	0.905
unbalanced trinary	0.994	0.994	0.984	0.975	0.998	0.980	0.985	0.996	0.982

4.3.1 Performance of the Classifiers

Before demonstrating the active learning, the basic models are shown off. These did not have much hyper-parameter tuning, instead basic versions of the models used for the SDSS training were used. They also served as a toy box for suitable active learning methods and whether they could be incorporated later on. Both models were tested and averaged five times. For distinction for the rest of this dissertation, uncertainty sampling and entropy sampling will be referred to as two distinct methods, where uncertainty sampling uses the binary case found in equation 4.2 and entropy sampling uses the more general method found in equation 4.1.

Random Forest

The random forest model used for Gaussian clouds had 100 estimators in, again with no definite depth. This is likely overkill for two or three Gaussian clouds where each datapoint only has two features, but the random forest classifier needed to work on the initial data set as well, there too-small a model might not be enough. Just like the SDSS random forest, the oob score was set to true for use in the active learning. The base model accuracy at the end of training can be seen with figure 4.1 and the per-class performance metrics with figure 4.2.

Neural Network

The neural network for the Gaussian clouds is much simpler; there is no dropout (the feature set is too small for the technique to be useful), there are only 2 inputs in the input layer and a single hidden layer consisting of 100 neurons. A batch size of 64 and 40 epochs were also found to be enough for the active learning to take place. A small manual search was used to find these parameters, but it was not in-depth as the model does not need

TABLE 4.3: Accuracy scores for each Gaussian cloud dataset - trained on the neural network.

Features	Average Accuracy
balanced binary	0.919
unbalanced binary	0.999
balanced trinary	0.892
unbalanced trinary	0.993

TABLE 4.4: Class specific performance metrics for each Gaussian cloud dataset - trained on the neural network. The precision, recall, and F_1-score are shown for all classes.

Gaussian	Precision			Recall			F_1-Score		
Clouds	1	2	3	1	2	3	1	2	3
balanced binary	0.909	0.931	–	0.932	0.907	–	0.920	0.919	–
unbalanced binary	1.000	0.999	–	0.990	1.000	–	0.995	0.999	–
balanced trinary	0.862	0.905	0.909	0.840	0.919	0.917	0.851	0.911	0.913
unbalanced trinary	0.998	0.994	0.987	0.973	0.999	0.996	0.986	0.996	0.984

much for it to process the feature set. Some parameters are similar to the $10k$ SDSS dataset so as to test whether the active learning will be effective using such a large amount of choice data with each iteration. The average accuracy at the end of training can be seen with figure 4.3 and the per-class performance metrics with figure 4.4. The results are nearly alike with the random forest classifier, verifying the model performance.

4.3.2 Performance of the Active Learning Queries

Now the base models have been established, how the active learning queries perform can be shown. This test bed is needed to determine whether active learning works on such a scale and whether it is worth it for certain datasets. The feature set has an initial seed of 64 values, and active learning is applied 49 times, each one adding 64 samples to the dataset - giving a total usage of 3200, half as many values in the final feature set as the full set used in the standard performance. The legends on the graphs involving active learning use abbreviated names: random sampling is *ran*, uncertainty sampling is *unc*, entropy measure is *ent*, the best-vs-second-best approach is *bsb*, variance reduction is *var*, and learning active learning is *lal*.

Random Forest

Table 4.5 displays the final test accuracy found for each active learning method on the different clouds. When comparing to see if the active learning is useful, using the random sampling is a way of checking if the methods worked, or simply if the amount of data used contributed more. Because the accuracy for the random sampling in all cases is almost as high as the base model's metric on all clouds, it is safe to say that part of why the model is so accurate is because of the amount of data involved regardless of the

TABLE 4.5: Accuracy scores from all active learning methods for each Gaussian cloud dataset - trained with a random forest.

Features	average test accuracy for sampling methods					
	ran	unc	ent	bsb	var	lal
balanced binary	0.908	0.908	0.909	0.909	0.909	0.899
unbalanced binary	0.998	0.999	0.999	0.999	0.999	0.999
balanced trinary	0.880	0.884	0.885	0.884	0.884	0.881
unbalanced trinary	0.990	0.989	0.992	0.992	0.992	0.992

sampling method used. This does not devalue the active learning, as the methods do still result in slightly higher accuracy scores.

Because the amount of data is clearly affecting how the end model performs regardless of active learning in the random forest, it is best to look at figure 4.3. These graphs show how the model is performing as values are added. The training graphs down the left hand side are relatively consistent, with little deviation from most active learning methods for all clouds - besides random sampling and learning active learning. These methods are noticeably lower in quality (slower rise and a larger deviation) when referring to binary classifications, despite what the test accuracies have shown. Uncertainty sampling is also obviously falling behind when looking into the unbalanced multi-class cloud, which is to be expected as the algorithm should not be suitable for multi-class data. The validation graphs show a different story for the binary cases. In fact active learning for both balanced and unbalanced binary clouds is incredibly unstable. With the accuracy actually dropping after roughly 800 samples for the balanced case and 1500 samples for the imbalanced case - the latter is more trending toward a single accuracy which is surprising. The trinary clouds are much more stable, and again there is a poor performance from the uncertainty sampling method in relation to the unbalanced clouds. Another aspect to look it is what points were chosen by the different active learning methods in figure A.1, figure A.2, figure A.3, and figure A.4.

Neural Network

Active learning on Gaussian clouds using a neural network generally follows the same patterns as the random forest. What can be seen when looking at table 4.6, is that compared to the other types of established clouds - the final accuracies for the sampling methods on the balanced multi-class clouds are improved on all but the variance sampling. Again though, because the random sampling is so similar it would be amiss to assume this is down to the active learning.

Thus the accuracy graphs in figure 4.4 and the loss graphs in figure 4.5 should be investigated. One thing to note in all cases is how the training and validation accuracy using random sampling for all clouds is so good. This is after a dip in form in the training accuracy for every method after a certain amount of samples - meaning the neural network is overall too confident in its results and each new selection is not actually the optimum selection. At least this would be the case if the validation accuracy agreed with

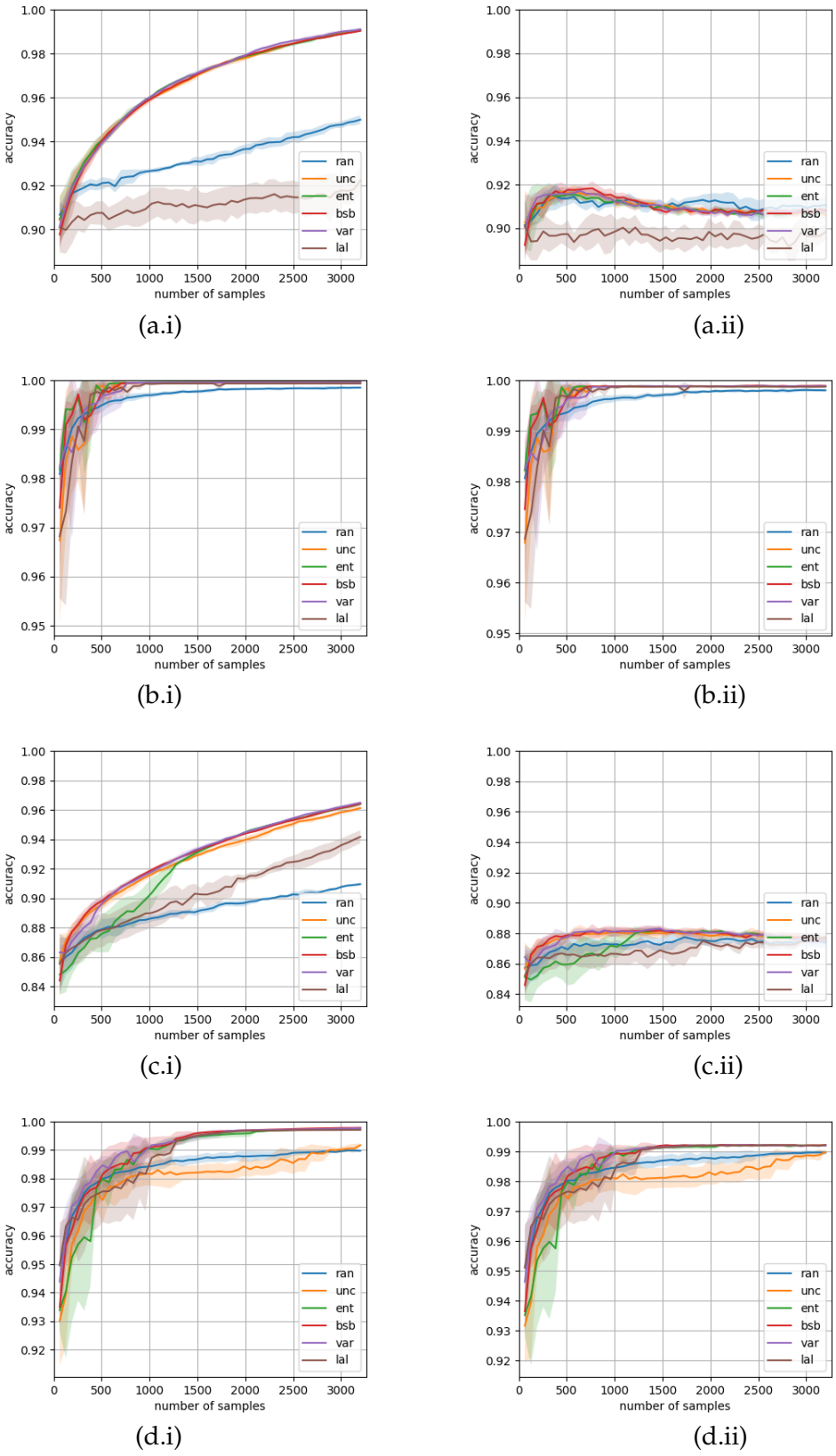


FIGURE 4.3: Accuracy curves for the different active learning methods trained using a random forest. The left graphs (i) represent training curves and the right graphs (ii) represent the validation curves. Labelled (a) for a balanced binary dataset, (b) for an imbalanced binary dataset, (c) for a balanced trinary dataset, and (d) for an imbalanced trinary dataset.

TABLE 4.6: Accuracy scores from all active learning methods for each Gaussian cloud dataset - trained with a neural network.

Features	average test accuracy for sampling methods					
	ran	unc	ent	bsb	var	lal
balanced binary	0.918	0.920	0.919	0.918	0.918	0.666
unbalanced binary	0.997	0.998	0.998	0.998	0.998	0.999
balanced trinary	0.889	0.885	0.890	0.889	0.891	0.889
unbalanced trinary	0.985	0.985	0.989	0.989	0.990	0.990

that statement. Instead the validation accuracy picks up when the training accuracy is lower showing that even if the model is not confident in the training features, it is still performing well on other data.

This can be seen to a greater extent in the loss, where the models are consistently getting more confident as samples are added. The only issue here is the random sampling shows the fastest drop rate for loss. Evidently the increase in performance for the neural networks can said to be down to the increase in data more than anything else, and this is further backed up with figure A.13, figure A.14, figure A.15, and figure A.16, where the metrics flatten out after a only a portion of the total samples. Additionally, in figure A.9, figure A.10, figure A.11, and figure A.12, the points chosen to be learned on seem much closer to the actual boundaries than the points chosen by the random forest, meaning the neural network could be more effective at identifying active points.

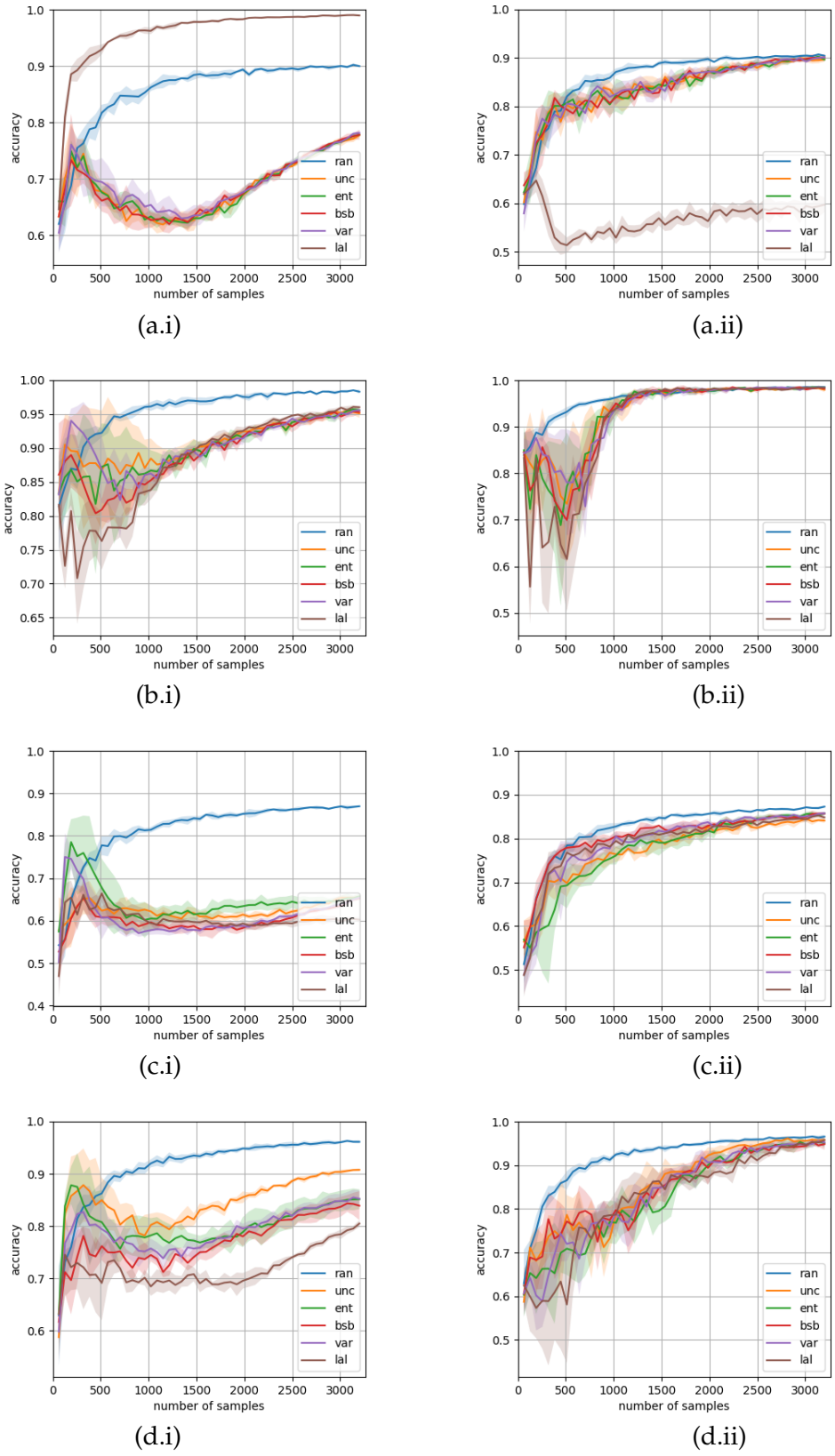


FIGURE 4.4: Accuracy curves for the different active learning methods trained using a neural network. The left graphs (i) represent training curves and the right graphs (ii) represent the validation curves. Labelled (a) for a balanced binary dataset, (b) for an imbalanced binary dataset, (c) for a balanced trinary dataset, and (d) for an imbalanced trinary dataset.

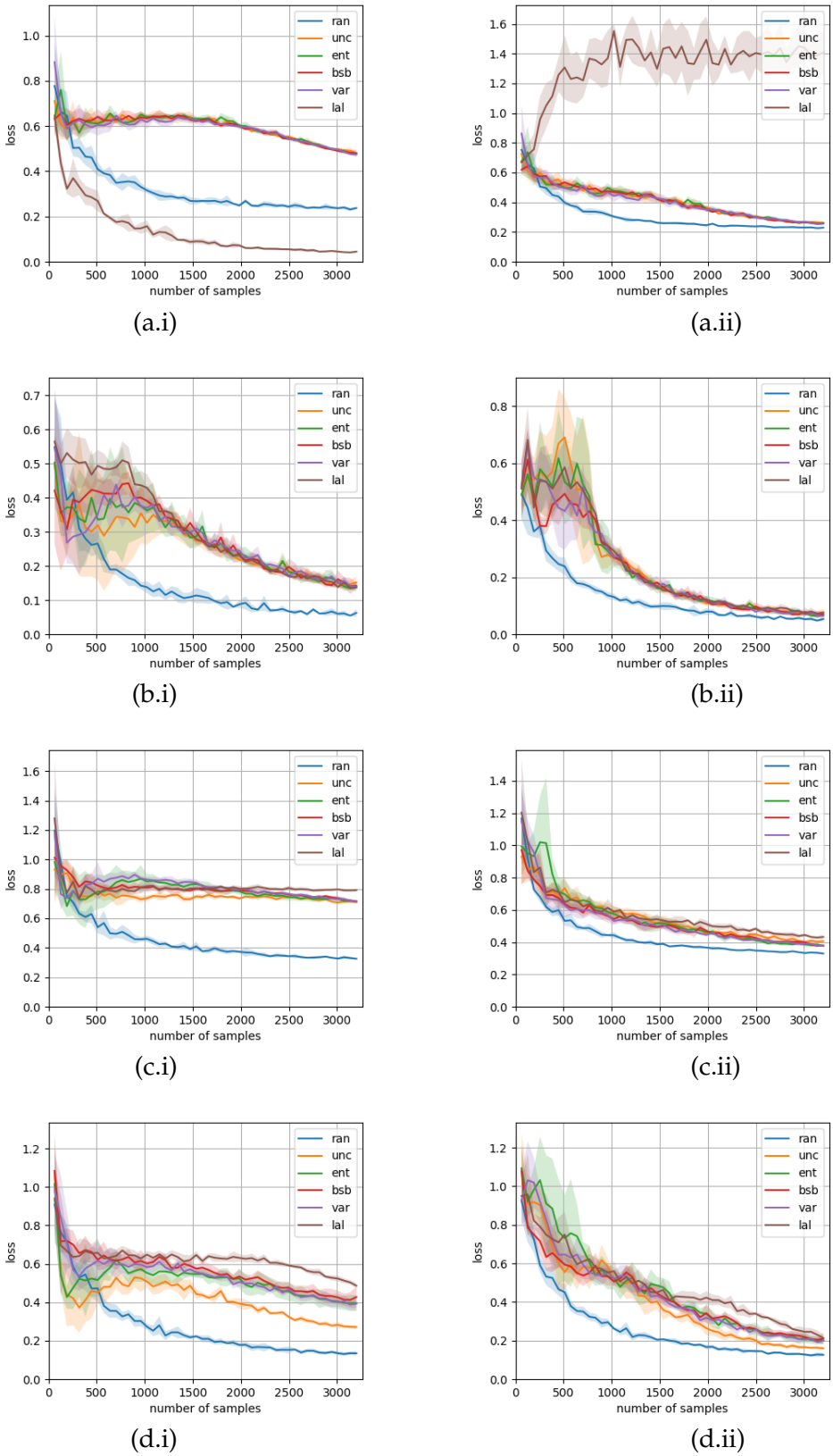


FIGURE 4.5: Loss curves for the different active learning methods trained using a neural network. The left graphs (i) represent training curves and the right graphs (ii) represent the validation curves. Labelled (a) for a balanced binary dataset, (b) for an imbalanced binary dataset, (c) for a balanced trinary dataset, and (d) for an imbalanced trinary dataset.

Chapter 5

Active Learning on the SDSS Data

Now that active learning has been thoroughly explained, it can be applied to the chosen SDSS data. Before choosing the ideal active learning acquisition function(s), each one must be vetted against the basic models used in order to check whether there is an improvement. The most complete model(s) will be used further to explore a dataset with a much larger pool - equivalent to Clarke's $300,000$ total training set where the active learning will be employed in a heavier manner.

5.1 Exploring the Models

To understand how the chosen models would work under the circumstances and to gauge rough hyper-parameters, a small dataset of 10000 datapoints was used for training and validation. Although much smaller than Clarke's paper where they command roughly 300000 datapoints (Clarke et al., 2020), the 10000 values used were still of use as a testbed for the hyper-parameter tuning and as an indicator of performance of the multi-layer perceptron and random forest classifier. For future reference, these values will be referred to as the $10K$ dataset. This toy dataset had a class ratio of $\sim 7:1:2$ for galaxies, quasars, and stars respectively. It was split into 8000 seen and 2000 unseen

TABLE 5.1: A small description of each feature the models use to train on within this section (in order), the wavelengths are all photometric measurements and $resolved_r$ is the calculated value from Equation 2.1.

Feature	Description
$resolved_r$	resolved number determining a resolved or a point-like object
psf_u	SDSS optical band u - $\lambda = 0.355\mu$m
psf_g	SDSS optical band g - $\lambda = 0.477\mu$m
psf_r	SDSS optical band r - $\lambda = 0.623\mu$m
psf_i	SDSS optical band i - $\lambda = 0.762\mu$m
psf_z	SDSS optical band z - $\lambda = 0.913\mu$m
W_1	WISE infrared band 1 - $\lambda = 3.4\mu$m
W_2	WISE infrared band 2 - $\lambda = 4.6\mu$m
W_3	WISE infrared band 3 - $\lambda = 12\mu$m
W_4	WISE infrared band 4 - $\lambda = 22\mu$m

TABLE 5.2: Accuracy scores for variations on the 10k SDSS feature set - trained on the random forest.

Features	Average Accuracy
SDSS + WISE + $resolved_r$	0.976
SDSS + WISE	0.964
SDSS	0.929
SDSS + $resolved_r$	0.967
WISE	0.851

TABLE 5.3: Class specific performance metrics for variations on the 10k SDSS feature set - trained on the random forest. The precision, recall, and F_1-score are shown for all classes.

Features	Precision			Recall			F_1-Score		
	Galaxy	Quasar	Star	Galaxy	Quasar	Star	Galaxy	Quasar	Star
SDSS + WISE + $resolved_r$	0.979	0.940	0.985	0.993	0.918	0.931	0.986	0.929	0.957
SDSS + WISE	0.970	0.900	0.982	0.985	0.859	0.938	0.977	0.879	0.959
SDSS	0.941	0.857	0.918	0.977	0.818	0.772	0.958	0.837	0.839
SDSS + $resolved_r$	0.979	0.906	0.946	0.991	0.877	0.907	0.985	0.891	0.926
WISE	0.872	0.830	0.630	0.958	0.845	0.299	0.913	0.837	0.407

datapoints, with the seen points being split further into 6400 training points and 1600 validation points. Making the final training/validation/testing split 0.64/0.16/0.2. Because this split was random, it was assumed the ratios of the full 10k set would carry over. Table 5.1 is a reminder of each of the features used within the current chapter's modelling.

5.1.1 Random Forest Classifier

The random forest model used is based on Clarke's model, with 200 estimators being used and no maximum depth - meaning the trees can expand until the leaves are pure. Unlike Clarke's random forest however, the oob score has been set to true to enable its later use in the active learning. The models were trained and tested five times before averaging the results. This gives an average test accuracy for each feature set as shown in Table 5.2, with per-class precision, recall, and F_1-score shown in Table 5.3.

These values presented are not quite as good as Clarke's when on his optimised model and training set, however the full feature set metrics are similar enough to show that the 10k dataset is sufficient to verify the models used and that a smaller training size than what was originally anticipated can be employed. When looking into the less complete feature sets, there is a noticeable drop off in quality, however that does not affect the active learning procedures because they only use the full length of features.

5.1.2 Neural Network

The neural network will have more of an explanation, as Clarke did not test a neural network model, thus the hyper-parameters were tested out with the 10k dataset. The

TABLE 5.4: Accuracy scores for variations on the 10k SDSS feature set - trained on the neural network.

Features	Average Accuracy
SDSS + WISE + $resolved_r$	0.947
SDSS + WISE	0.918
SDSS	0.765
SDSS + $resolved_r$	0.811
WISE	0.788

TABLE 5.5: Class specific performance metrics for variations on the 10k SDSS feature set - trained on the neural network. The precision, recall, and F_1-score are shown for all classes.

Features	Precision			Recall			F_1-Score		
	Galaxy	Quasar	Star	Galaxy	Quasar	Star	Galaxy	Quasar	Star
SDSS + WISE + $resolved_r$	0.948	0.946	0.942	0.985	0.826	0.843	0.966	0.882	0.888
SDSS + WISE	0.908	0.954	0.968	0.991	0.724	0.688	0.948	0.822	0.801
SDSS	0.763	0.881	0.911	0.997	0.190	0.172	0.864	0.312	0.338
SDSS + $resolved_r$	0.810	0.912	0.552	0.988	0.586	0.745	0.890	0.714	0.131
WISE	0.781	0.916	0.100	0.995	0.420	0	0.875	0.575	0

hyper-parameter search was accomplished with a manual search for different major intervals in the parameters. For this dataset, it was found that 3 hidden layers with 800 neurons were optimal for the network. More hidden layers and neurons did improve results but not to any significant degree and were thus discarded. A batch size of 64 across 300 epochs was also set. The batch size being small enough to view any variations but large enough to cover the entire training set quickly. This amount epochs could be considered overkill, yet the value was chosen so that each feature set was given ample time for training and so that the optimum number of epochs for the full feature set could be seen. This optimum point is where the curve flattens enough while not being stable for too long as this can be considered wasted time for active learning. Finally, dropout was implemented due to the enhancement of the model, even when considering the increase in training time. The average accuracy for each feature set the neural network was trained in is shown in Table 5.4, with per-class precision, recall, and F_1-score shown in Table 5.5.

On an initial look, the performance of the neural network in relation to the random forest is discouraging. Yet the full feature set still exhibits a good performance so similar outcomes could be expected for the active learning. The per-class metrics are similar, with precision, recall, and F_1-score being mostly lower than the metrics extracted from the random forest and Clarke's data. Unlike the random forests though, the neural network seems to have better scores for quasars rather than stars when using the full feature set, making it a viable option for identifying quasars if the model is improved or perhaps if the correct data is selected. When looking at the WISE feature set, the model performed incredibly poorly despite an overall accuracy of 0.788, showing just how much certain results can be skewed. This could be because of the class imbalance, where the features

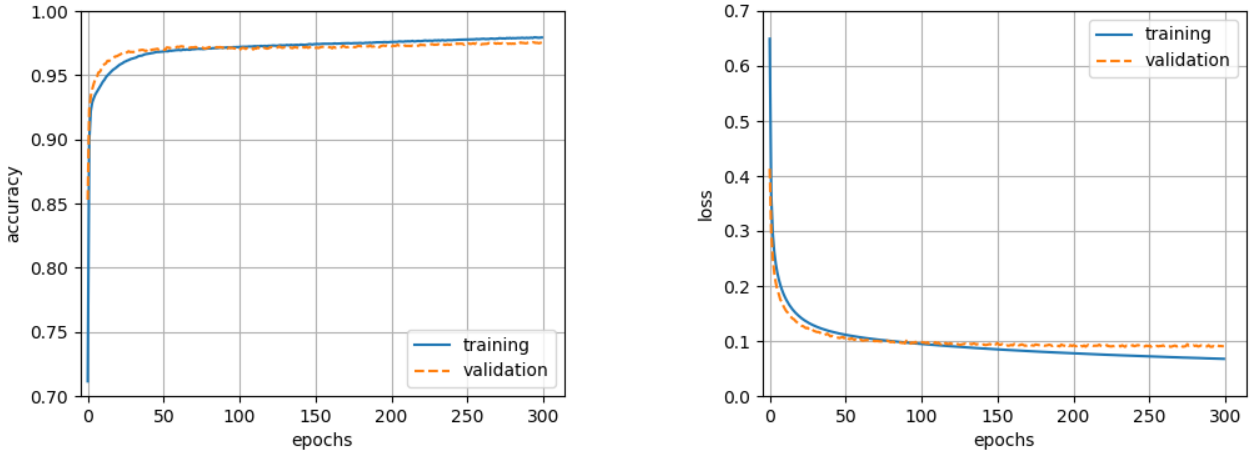


FIGURE 5.1: Accuracy and loss curves for the basic neural network using the 10*k* dataset. The left graph represents the accuracy curves and the right graph represents the loss curves.

from WISE alone on such a small dataset are not enough for the model to understand what a star is. However the active learning is done on the full feature set so this should not occur later on.

The test metrics alone do not indicate where the optimum epoch is, they are just how the model performs after 300 epochs. Instead a more in-depth view of how the model performs at each epoch is exhibited in Figure 5.1, allowing analysis on when the model is trained enough so that active learning could theoretically occur. Within Figure 5.1, the accuracy and loss curves follow predictable patterns where they become more asymptotic as the data is iterated over. There is some over-fitting, represented in the validation loss curve going above the training loss curve, however the difference is insignificant until after epoch 250. This can be seen with the accuracy curve but to a lesser extent. Ideally the hyper-parameters would be fine-tuned for even slighter margins, however it takes significantly more time to train a neural network over 300 (or more) epochs compared to a random forest, thus the model can be considered complete. Because of the timescale, slightly different parameters need to be chosen for the active learning to be employed effectively.

To choose the optimum epoch for active learning, the value at which the curves flatten out is looked at rather than the point of intersection between the training and validation curves. This point of intersection is 80 epochs, this is where the model is ideally trained with no over or underfitting. However it is not hard to notice the accuracy curve is flat for half (40) of those epochs. The loss curve on the other hand only fully flattens out at epoch 100, but the curve is relatively flat in epoch 50. Thus a mix of epochs has been chosen for testing. The 10*k* dataset will use 50 epochs before applying the active learning, 40 will be used for some balanced datasets. To reiterate; the epoch values chosen are not for optimal neural network, but rather the values at which the network is considered 'good enough' to evaluate the unlabelled pool of data.

TABLE 5.6: Accuracy scores from all active learning methods for the 10*k* SDSS dataset - trained with both machine learning algorithms.

algorithm used	average test accuracy for sampling methods					
	ran	unc	ent	bsb	var	lal
Random Forest	0.974	0.977	0.977	0.977	0.977	0.972
Neural Network	0.932	0.933	0.940	0.933	0.941	0.941

5.2 Active Learning on different variations of SDSS data

To identify which active learning processes are ideal for use with the full dataset, the 10*k* dataset was used. Furthermore, a balanced binary and a balanced multi-class dataset were used to assess whether the active learning is still performing well enough in comparison to the Gaussian clouds and checks whether they are more or less applicable in balanced cases with larger feature sets. If balanced sets do produce models with greater effectiveness, they shall be recommended for surveys with less class imbalance than the SDSS.

Each balanced set has 20,000 values for each class, totalling 40,000 for the binary case and 60,000 values for the multi-class case. This larger volume also gives some insight into how active learning applies on larger scales than the Gaussian clouds considered in Chapter 4, as well as demonstrating how a larger pool may affect results. The binary classes used were the class 0 and class 1, or galaxies and quasars respectively. Not only was this easier to code due to the already assigned labels, but it investigates how the active learning rates the features between quasars and galaxies.

To pull the feature importances, **Scikit-learn**'s built in features were used for the random forest and **Captum** was used for the neural network. When viewing the feature importance for the first one, the features are ranked and given scores that when summed, total 1. However Captum instead looks into how important the features are individually rather than how they sum up to a specific value. Any positive values are considered important, and negative values are considered unimportant.

5.2.1 10K Multi-Class Dataset

The models trained on this 10*k* dataset thus use the established parameters as for the base models Active learning is applied 49 times from a seed of 64 in a similar manner to the Gaussian clouds. This gives a total dataset of 3200 chosen points out of a total training set of 6400 once the active learning has completed. Like the Gaussian clouds, a random selection will take place as a control to confirm whether the active learning is the reason behind the improvement or whether it is simply the increased number of samples in the training set. Table 5.6 shows the test accuracy scores for the different active learning methods and will be explored further below. In general, the random forest performs better with the active learning while none of the active learning methods on the neural network equal the accuracy found with the full dataset. A two dimensional reduction on

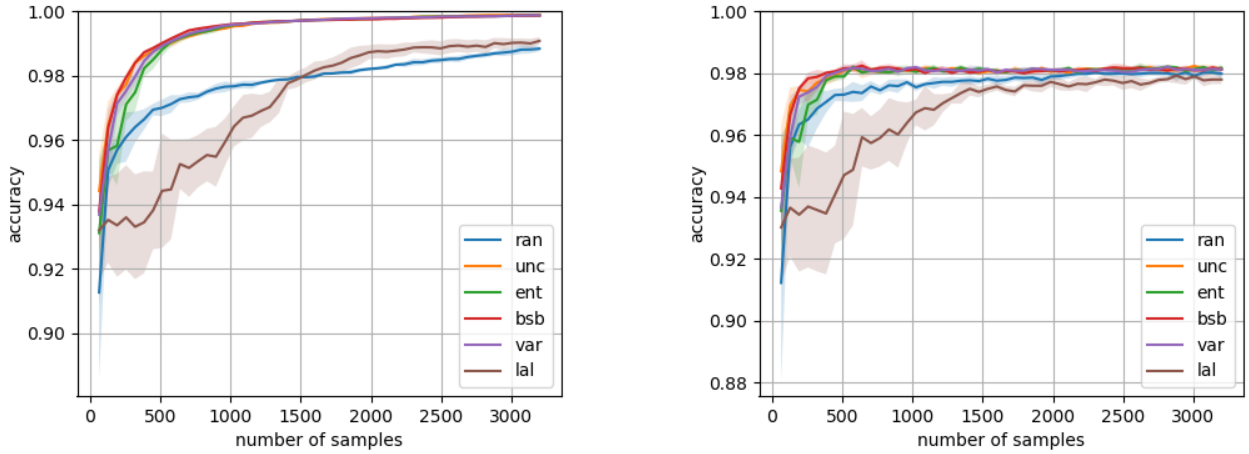


FIGURE 5.2: Accuracy curves for the different active learning methods on the 10k dataset trained using a random forest. The left graph represents the training curves and the right graph represents the validation curves.

the full feature set was also performed and the active points chosen are shown, this can be seen for the random forest in Figure B.1 and the neural network in Figure B.2. These were intended for all clouds yet from these images it is difficult to distinguish between the cloud clusters and understand the ratio of points chosen, thus they were no longer used after the 10k dataset.

Random Forest

As mentioned, the active learning has generally improved the accuracy scores. The random sampling and learning active learning are the exceptions. Because the random sampling is adding datapoints randomly the lower accuracy is expected, meaning only the learning active learning has failed to improve the model. This is backed up further and validation graphs seen in Figure 5.2. Both graphs show the active learning is working due to the major improvements made during the initial few sets of data added, with a slight decrease in the validation after 640 labelled values - possibly in part to the training accuracy being too great. Apart from the learning active learning, all active learning methods perform significantly better than the random sampling, demonstrating further that it is not only because the amount of data is increasing, but that the right points are being selected. All active learning methods do seem to tend towards the same final performance, with the difference in the final validation accuracy scores being less than 0.01. This occurs in the training accuracy too, where the top four sampling methods converge. In fact because the sampling methods are so effective early on, it could skew their selection of the next data points to be labelled, which does not apply to the learning active learning. The learning active learning also does not seem to have fully developed, its effectiveness could be improved by simply increasing the number of sources used in each iteration or the number of iterations.

TABLE 5.7: Class specific performance metrics for the 10k SDSS dataset - trained on the random forest. The precision, recall, and F_1-score are shown for all classes.

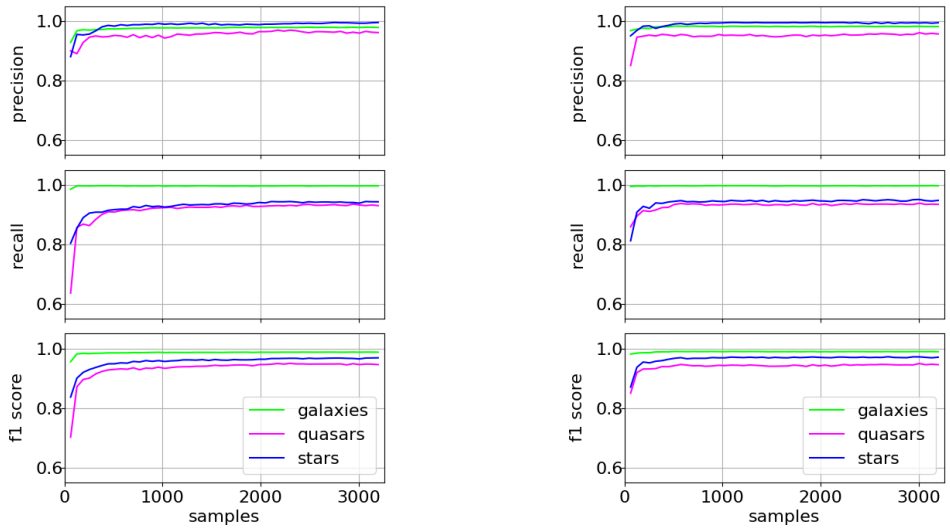
Active	Precision			Recall			F_1-Score		
Learning	galaxy	quasar	star	galaxy	quasar	star	galaxy	quasar	star
ran	0.978	0.931	0.988	0.992	0.915	0.923	0.985	0.923	0.955
unc	0.979	0.938	0.991	0.994	0.923	0.930	0.987	0.930	0.959
ent	0.979	0.939	0.990	0.994	0.923	0.930	0.986	0.931	0.959
bsb	0.979	0.938	0.991	0.994	0.923	0.930	0.986	0.930	0.960
var	0.979	0.939	0.990	0.994	0.922	0.931	0.986	0.930	0.960
lal	0.978	0.920	0.982	0.994	0.899	0.917	0.986	0.909	0.948

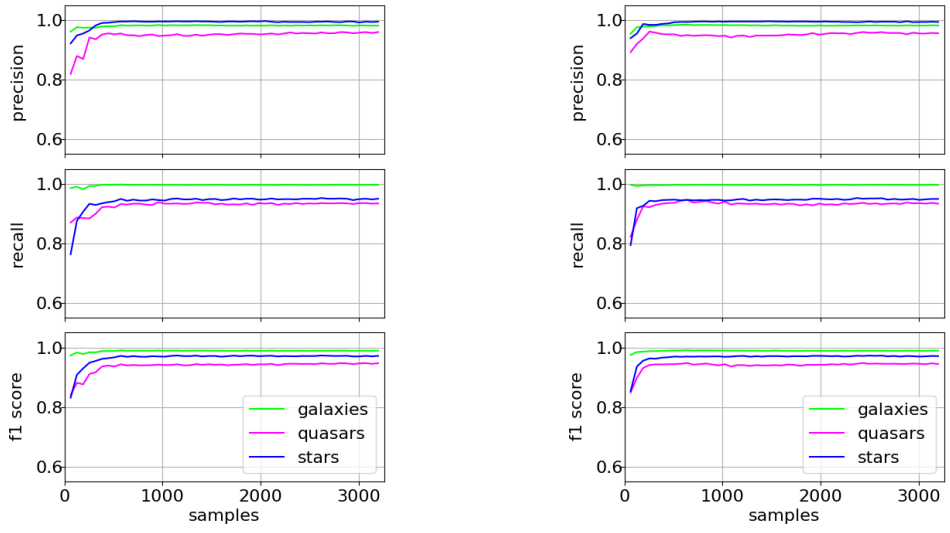
The precision, recall, and f_1-score for each active learning method are similar in Table 5.7 (besides random sampling and learning active learning). The entropy sampling gives the best of these metrics for quasars, which are the more difficult to classify than the other two. This is on top of using less computing power than the best-vs-second-best sampling and the variance sampling, so it could be used further.

Reviewing the graphs in Figure 5.3, the metrics given for all classes rise quickly and level off in the first few iterations of learning. All besides the learning active learning which take more samples for the metrics to become level for stars and quasars. Only these two are mentioned as the galaxy class is almost immediately maxed out for every method, likely because of the class discrepancy and even with a small labelled pool of data to train from, the imbalance is so great that it is obvious to the random forest which input is a galaxy. Figure 5.4 shows the importance of each feature. They are not ranked, but are in the order of input. In all learning methods, the most important feature is $resolved_r$ and the second most is psf_z, the least important feature is $W4$. Furthermore the variation in these features is tiny even when looking at the random sampling, showing the feature importance does not change much despite different sets of inputs. The learning active learning features show more variation in their importance and do not quite fit the general trend. Although the most important and least important features are the same as the rest of the active learning methods, the WISE features are not valued at all. These features are also ranked differently to the feature importances found in Clarke et al. (2020), where the WISE features were not ranked above any of the SDSS features. The most important and least important ($resolved_r$ and $W4$) are still the same however, meaning the rankings have been shuffled slightly with active learning. Possibly indicating that these features have not been trained on much, making the learning active learning results comparable to the SDSS + $resolved_r$ feature set rather than the full feature set. This could also be why the metrics are worse than the other methods, as both the SDSS and WISE sets are needed to accurately classify the data.

Neural Network

The neural network accuracy graphs seen in Figure 5.5 appear dense at a glance, however when compared to the random forests, the accuracy curves are further apart compared to





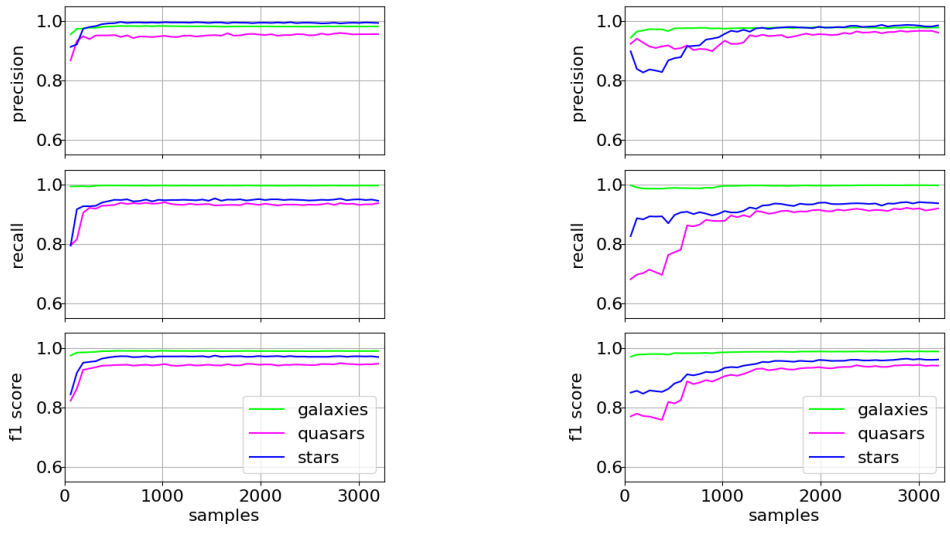


FIGURE 5.3: Precision, recall, and f_1-score for the $10k$ SDSS dataset trained on a random forest. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

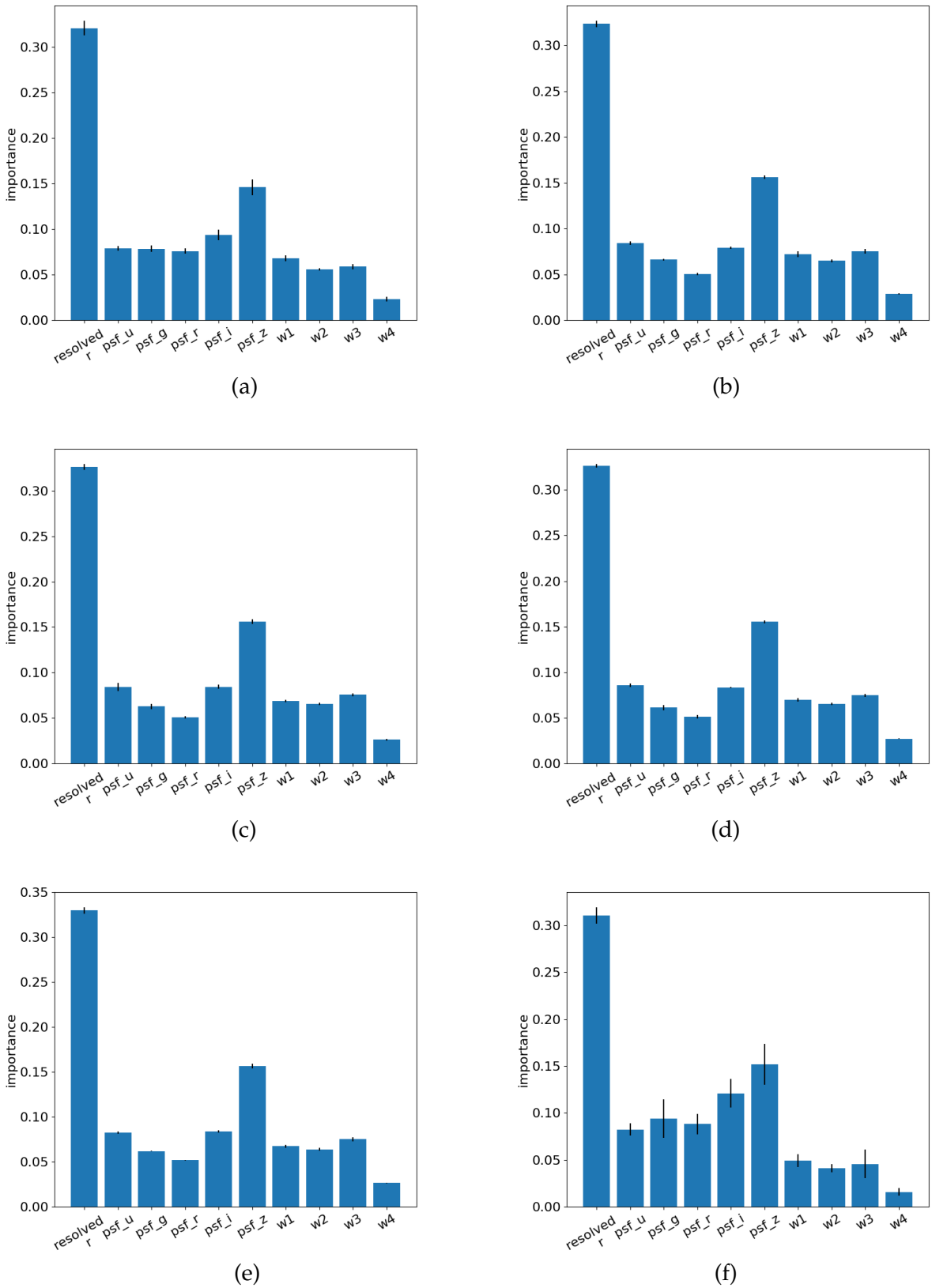


FIGURE 5.4: Attribute importance for the 10k SDSS dataset trained on a random forest. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

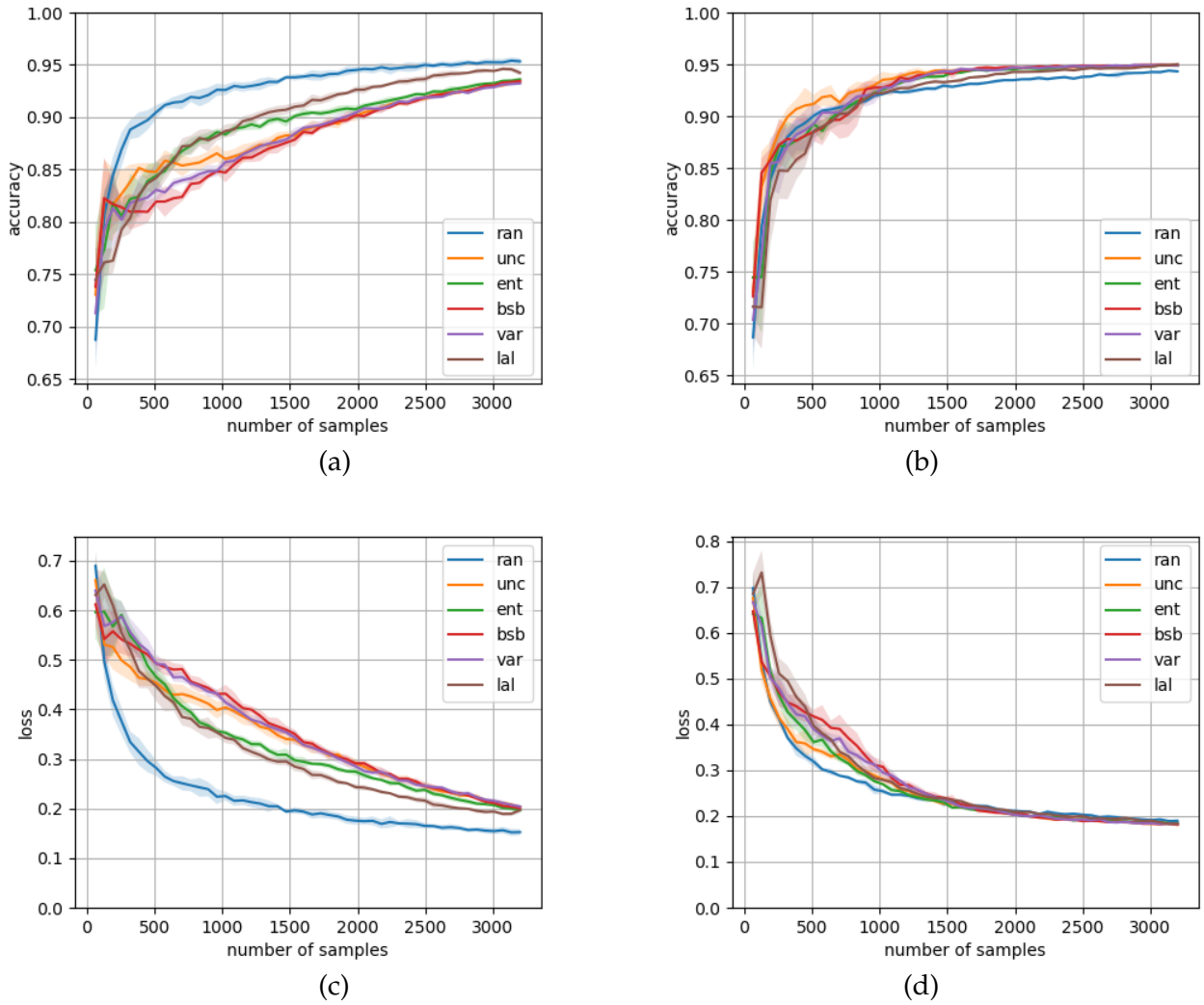


FIGURE 5.5: Accuracy and loss curves for the different active learning methods on the 10k dataset trained using a neural network. The left graphs (a) and (c) represent the training curves for accuracy and loss respectively, the right graphs (b) and (d) represent the validation curves for accuracy and loss respectively.

TABLE 5.8: Class specific performance metrics for the 10k SDSS dataset - trained on the neural network. The precision, recall, and F_1-score are shown for all classes.

Active	Precision			Recall			F_1-Score		
Learning	galaxy	quasar	star	galaxy	quasar	star	galaxy	quasar	star
ran	0.929	0.936	0.952	0.986	0.813	0.745	0.960	0.870	0.835
unc	0.926	0.962	0.962	0.991	0.789	0.748	0.957	0.866	0.840
ent	0.936	0.942	0.961	0.988	0.814	0.787	0.961	0.873	0.865
bsb	0.928	0.957	0.956	0.989	0.783	0.760	0.957	0.860	0.846
var	0.938	0.946	0.958	0.988	0.808	0.800	0.962	0.871	0.870
lal	0.940	0.937	0.952	0.986	0.835	0.791	0.962	0.883	0.862

random forests. This disparity is because of the larger scale that has to be included in the graph. With both the accuracy and loss, the validation curves are much closer together than the training curves and no active learning methods really improve over the random sampling, meaning that the active learning does not look like it is working. The final test accuracies contrast this however, as they show improvements for all active learning methods compared to the random sampling. So despite the poor training improvements as the number of samples increases, the end result is still better. With regards to the learning active learning too, the graphs and final accuracy scores indicate that this method is improving or keeping up with the other active learning methods - in stark contrast to the random forest learning active learning which failed in comparison. Perhaps the learning active learning features are actually better for the neural network.

The precision, recall, and f_1-score for the active learning methods in Table 5.8 have a wider variation compared to the test metrics from the random forest in Table 5.7. Strangely, the uncertainty sampling results in the greatest precision for quasars despite the method being designed for binary cases. It could be a fluke though despite the averaging as the recall (and therefore the F_1-score) are not as good. The metrics for the learning active learning are also a bit more on par with the other active learning methods, further backing up the point that the features the learning active learning examines are more viable features.

Figure 5.6 gives a large difference in performance for the different classes, especially when compared to the same graphs extracted from the random forest. Both quasars and stars take various amounts of iterations before leveling out thus a neural network can probably be assumed to have much better performance for galaxies over anything else, likely due to the massive class disparity. This can be confirmed upon viewing the balanced sets of data. In Figure 5.7, the feature importances are displayed. The most important feature always looks to be $W1$, while the least important feature is mostly $W2$ though does switch to psf_g in the uncertainty sampling. The other feature importances also follow less of a trend in comparison to the random forest importances, with the favoured feature from it; $resolved_r$, actually varying a lot. In fact Captum seems to be unsure of how to place the $resolved_r$, whether it is important or unimportant, potentially signalling its usefulness. Considering the learning active learning is much better under these circumstances, it is odd how $resolved_r$ is placed so highly.

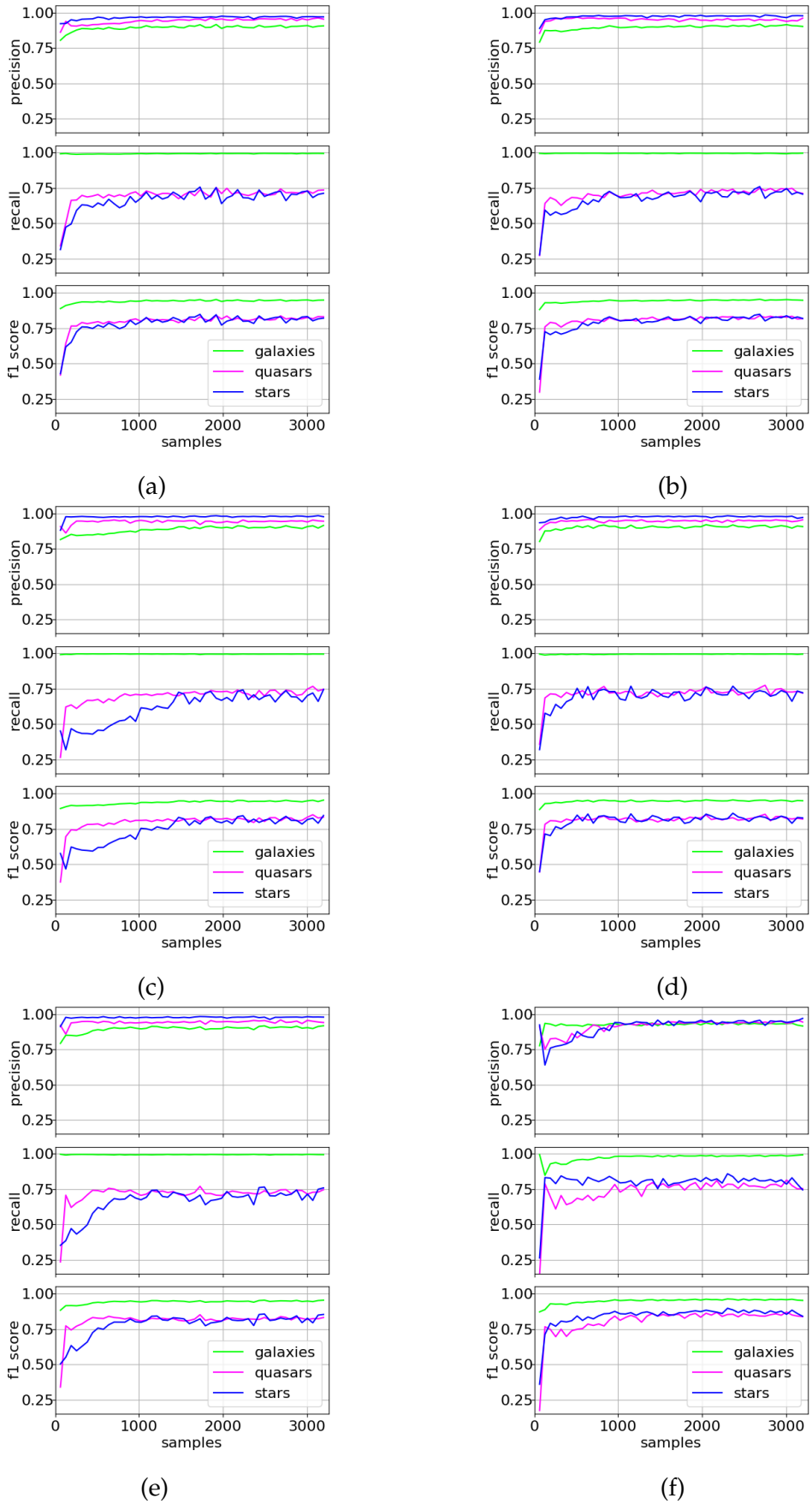


FIGURE 5.6: Precision, recall, and f_1-score for the $10k$ SDSS dataset trained on a neural network. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

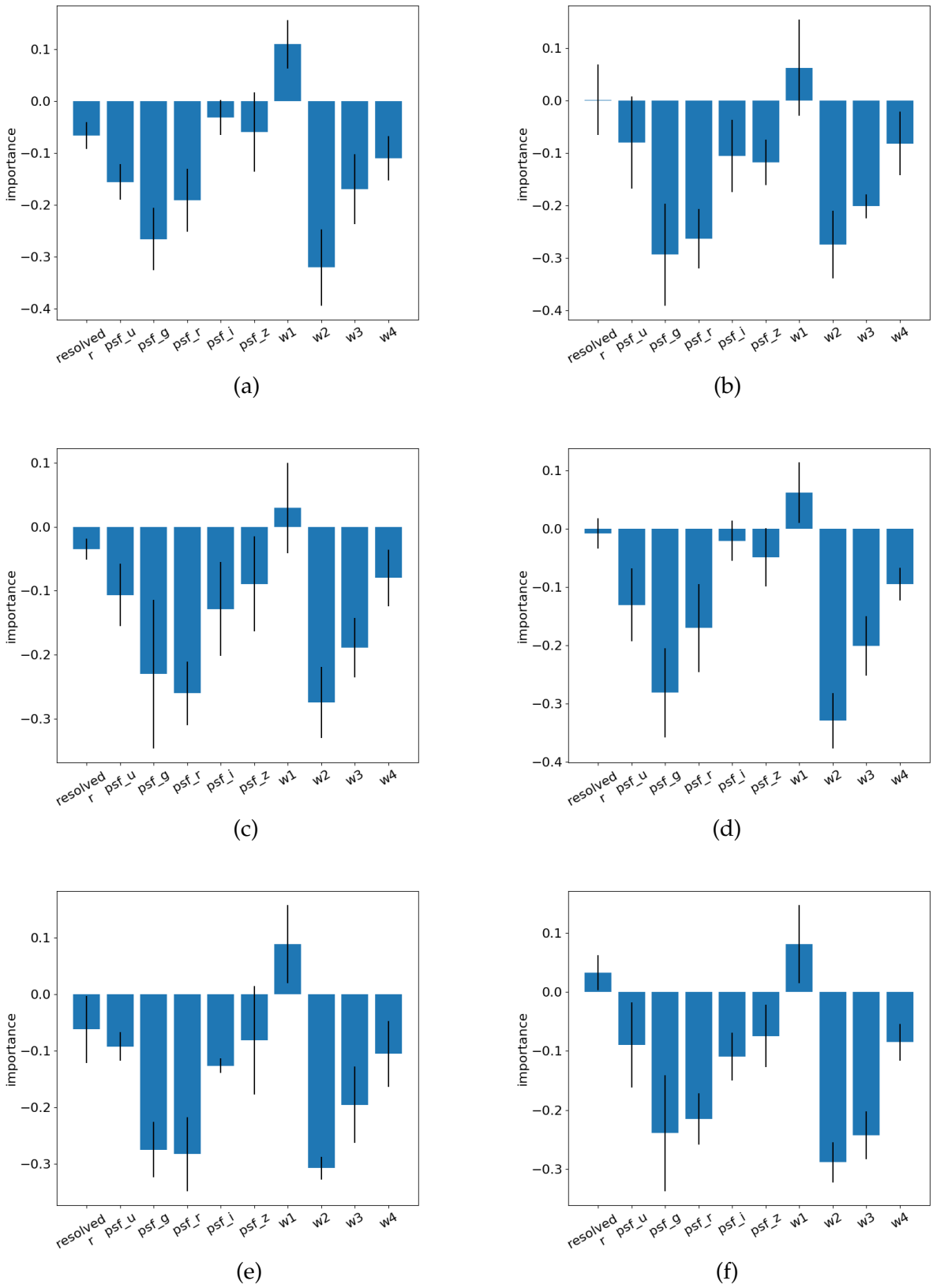


FIGURE 5.7: Attribute importance for the 10k SDSS dataset trained on a neural network. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

TABLE 5.9: Accuracy scores from all active learning methods for the balanced datasets - trained with both machine learning algorithms.

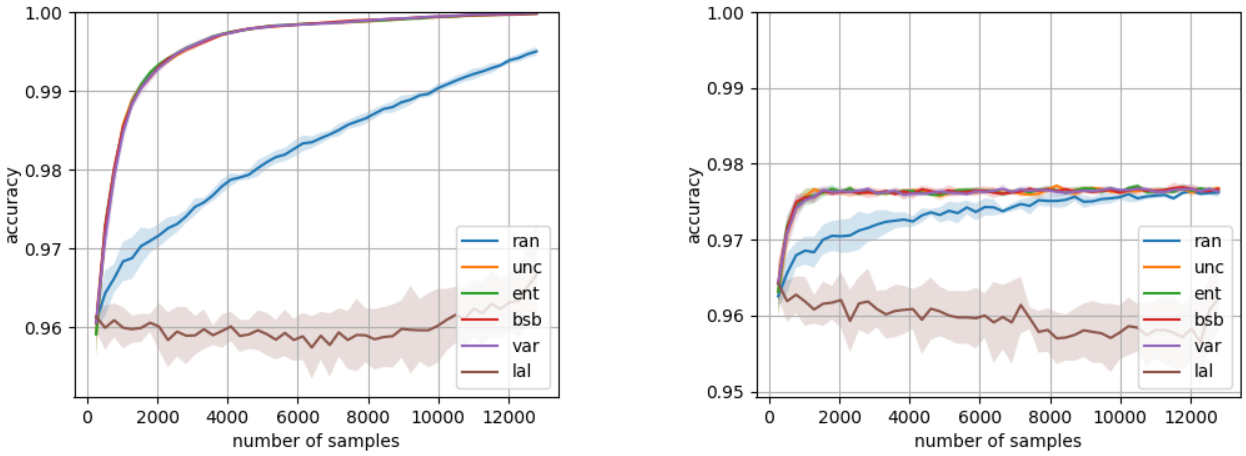
algorithm used	average test accuracy for sampling methods					
	ran	unc	ent	bsb	var	lal
Random Forest	0.973	0.974	0.974	0.974	0.974	0.959
Neural Network	0.953	0.955	0.940	0.947	0.951	0.941
Random Forest	0.964	0.966	0.967	0.967	0.966	0.967
Neural Network	0.925	0.924	0.914	0.921	0.928	0.768

5.2.2 Balanced Datasets

The balanced datasets of SDSS data were used in conjunction with the $10k$ dataset to evaluate how the different models perform under different circumstances and to validate whether any class discrepancy is the actual reason for galaxies to receive such good metrics, or whether the models are just better at classifying galaxies. Some of the hyper-parameters have been changed however as these were useful datasets to view how active learning performs on something larger than a dataset of ten thousand. Thus the active learning sample size was increased to 256 and the neural network had a modified epoch value and batch size. To begin, the accuracy scores in Table 5.9 again show the random forest as consistent while the neural network is relatively inconsistent. For the neural network trained on the binary dataset, the random sampling outperformed all active learning methods besides the uncertainty sampling which is to be expected considering it is entirely designed for a binary case. The multi-class case shows the same outcome, where the random forest scores are consistent while the neural network has varied scores. This time though, the neural network has an improvement in the variance sampling.

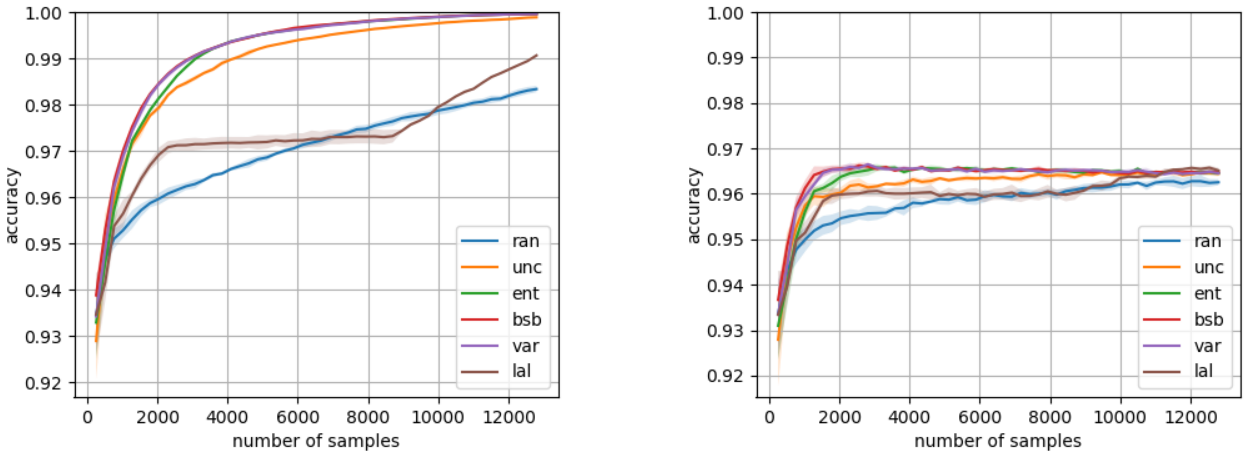
Random Forest

The balanced cases shown in Figure 5.8 and Figure 5.9 show incredible training accuracies - perhaps to the detriment of the active learning. Because these values are so good as values are added, the active learning has been affected as can be seen with the validation accuracy, which stagnates for all methods. The active learning for the binary case clearly does the job quickly, but after 2000 values they no longer improve, so there is clearly a limit that the active learning reaches. This cannot be seen for the random sampling however thus there is no way to fully verify if the active learning methods stop choosing the correct points due to the training accuracy being so high, or if there is just a hard limit that the random forest hits in this binary situation. The learning active learning does not improve, instead decreasing in validation accuracy - thus the regressor should not be applied when trained with the current feature set. The multi-class training and validation accuracies are similar in terms of hitting a hard limit, but this time the learning active learning does a much better job, with the validation accuracy being comparable to the other active learning methods. The learning active learning also overtakes the random



(a) (b)

FIGURE 5.8: Accuracy curves for the different active learning methods on a balanced binary set trained using a random forest. The left graph represents the training curves and the right graph represents the validation curves.



(a) (b)

FIGURE 5.9: Accuracy curves for the different active learning methods on a balanced multi-class set trained using a random forest. The left graph represents the training curves and the right graph represents the validation curves.

TABLE 5.10: Class specific performance metrics for the balanced binary SDSS dataset - trained on the random forest. The precision, recall, and F_1-score are shown for all classes.

Active Learning	Precision			Recall			F_1-Score		
	galaxy	quasar	star	galaxy	quasar	star	galaxy	quasar	star
ran	0.959	0.988		0.988	0.958		0.974	0.973	
unc	0.960	0.988		0.989	0.958		0.974	0.973	
ent	0.960	0.988		0.989	0.958		0.974	0.973	
bsb	0.960	0.988		0.989	0.958		0.974	0.973	
var	0.960	0.988		0.989	0.958		0.974	0.973	
lal	0.947	0.973		0.974	0.944		0.960	0.959	

TABLE 5.11: Class specific performance metrics for the balanced multi-class SDSS dataset - trained on the random forest. The precision, recall, and F_1-score are shown for all classes.

Active Learning	Precision			Recall			F_1-Score		
	galaxy	quasar	star	galaxy	quasar	star	galaxy	quasar	star
ran	0.941	0.967	0.987	0.982	0.955	0.956	0.961	0.961	0.971
unc	0.944	0.967	0.990	0.983	0.957	0.959	0.963	0.962	0.974
ent	0.945	0.967	0.989	0.982	0.957	0.960	0.963	0.962	0.974
bsb	0.944	0.967	0.990	0.983	0.958	0.960	0.963	0.962	0.974
var	0.945	0.966	0.989	0.982	0.957	0.960	0.963	0.962	0.974
lal	0.945	0.968	0.987	0.983	0.956	0.961	0.964	0.962	0.974

sampling accuracy score after 10000 values, so perhaps it needs more iterations before levelling out.

Table 5.10 shows how confident the random forest is in identifying quasars for every active learning method, even more-so than galaxies which is surprising. So perhaps the class disparity seen in the $10k$ dataset does actually change the active learning points quite a bit. Both classes receive near perfect metrics from the outset as seen in Figure 5.10, clearly this is because of the $resolved_r$ feature that identifies whether the value is a point or a larger object, as Figure 5.12 proves. The $resolved_r$ feature makes up for almost half or more than half of the feature importance for every single method, meaning this was widely used in classifying and shows that when other point like objects are not involved, the machine learning process can easily identify galaxies and quasars.

The same can almost be said for the multi-class dataset. Now that the class imbalance has disappeared, both quasars and stars have 'caught up' and in fact, the models are least sure about the galaxies (regardless of active learning) as seen in Table 5.11. In fact, when comparing the metrics as the labelled pool is increased using Figure 5.11, it is easy to see that stars and quasars have improved their maximum limits and that galaxies no longer occupy the model's most confident class, showing that even with active learning the class disparity is affecting the random forest to some degree. Figure 5.13 shows a similar pattern to the binary case for feature importance, in that $resolved_r$ is still dominant. It is not as dominant however, as the average importance of the feature is around half (0.25) of what was seen in Figure 5.12 (0.5) and about five sixths of what was seen in Figure 5.4. The other features are therefore rated higher than normal, showing how important this

TABLE 5.12: Class specific performance metrics for the balanced binary SDSS dataset - trained on the neural network. The precision, recall, and F_1-score are shown for all classes.

Active	Precision			Recall			F_1-Score		
Learning	galaxy	quasar	star	galaxy	quasar	star	galaxy	quasar	star
ran	0.923	0.987		0.987	0.919		0.954	0.952	
unc	0.924	0.989		0.990	0.921		0.956	0.954	
ent	0.907	0.978		0.980	0.900		0.942	0.937	
bsb	0.916	0.983		0.984	0.911		0.948	0.945	
var	0.978	0.989		0.989	0.913		0.952	0.949	
lal	0.990	0.978		0.979	0.905		0.943	0.940	

feature is in exploring the difference between galaxies and quasars or stars.

Neural Network

Figure 5.14 has the accuracy curves demonstrate contrasting results to the random forest for a binary case, where the learning active learning is actually a great improvement over the other sampling methods for the training accuracy. Yet as it is more in line with the random sampling, it could simply be because the learning active learning is not actually effective at active learning the points, and they are instead more random compared to the other methods. The validation accuracy curve does not help the case either, as none seem to be outright better than another and all have flattened out. The loss curves show the same result and do not really help in understanding how the active learning performs, besides when looking at when the models stagnate. In fact although the validation accuracy is no longer increasing much after 6000 values, the loss is still decreasing and has not quite become level indicating a greater length of time or values could lead to increased performance. Figure 5.15 proves the conjecture that as the training accuracy increases, the active learning methods become less effective. This can be seen most with the learning active learning, where it is so confident in the training results initially that when 'testing' on the validation features - the model classifies over half the labels incorrectly (as can be seen in (b) at 8000 samples). The learning active learning does seem to improve after, however this improvement varies so much that it can be dismissed. Unlike the random forest, there does not seem to be a hard limit, or if there is - the samples used are not enough to reach it. Finally, the loss curve shows that random sampling here is outperforming the other methods making them obsolete and really a waste of computational power. The neural network therefore does not perform that well with the current active learning on a balanced multi-class dataset.

The active learning does give a more varied set of test metrics for a binary case as seen in Table 5.12, the uncertainty and variance sampling just edging out over the others. The former is expected, yet variance sampling again shows how surprisingly effective it can be when used in conjunction with a neural network. The performance metrics for both classes are much less static in Figure 5.16 so there are actually performance drops when some samples are added. Like the random forest, quasars are easier to identify

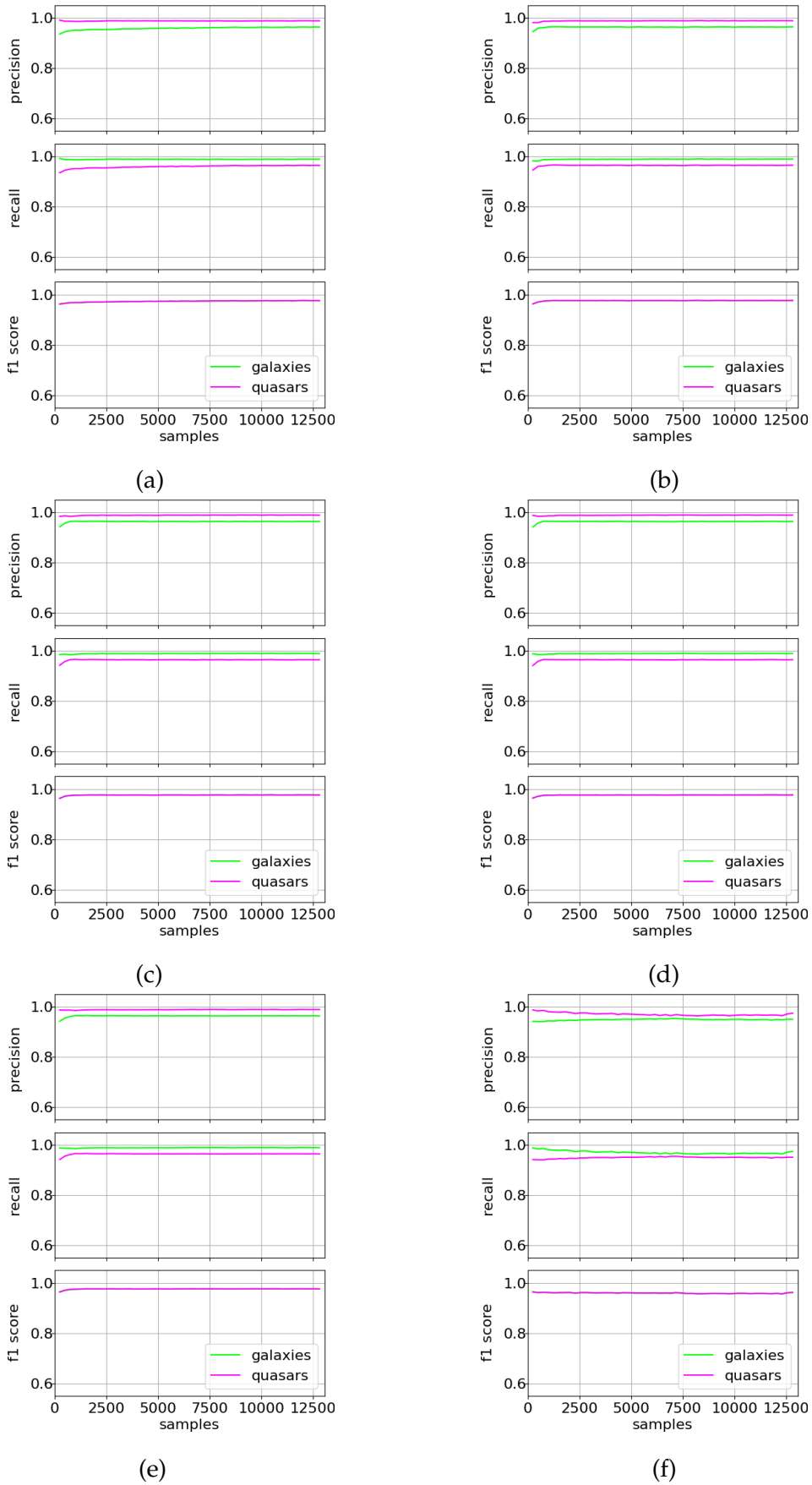
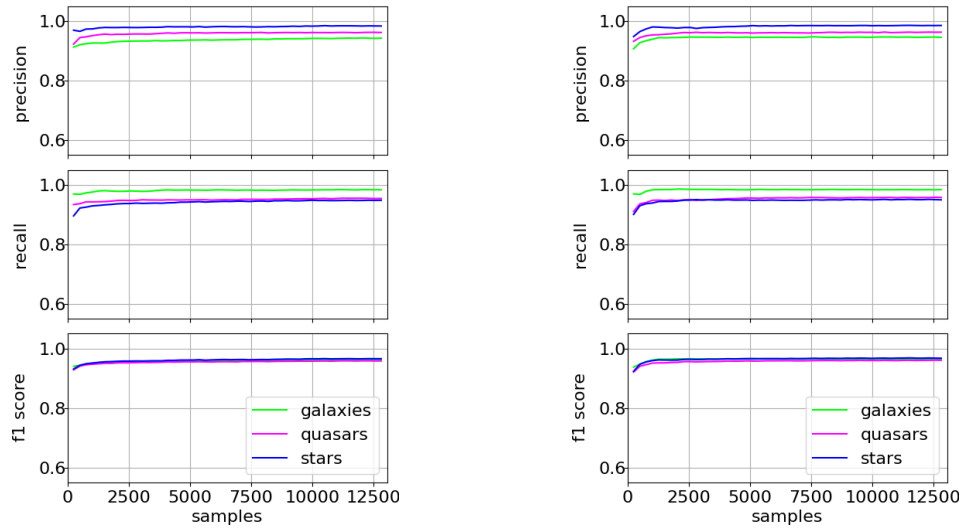
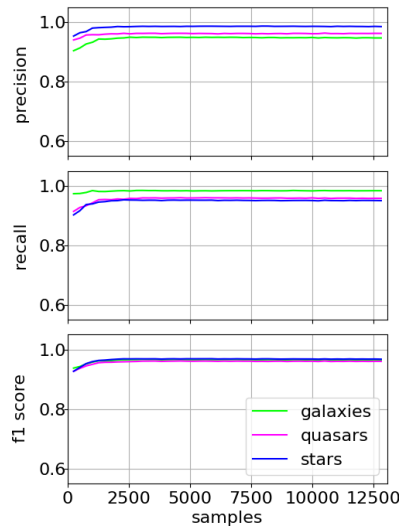


FIGURE 5.10: Precision, recall, and f_1-score for a balanced binary set trained on a random forest. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

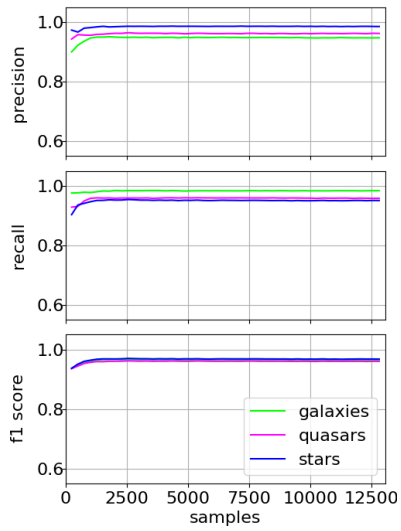


(a)

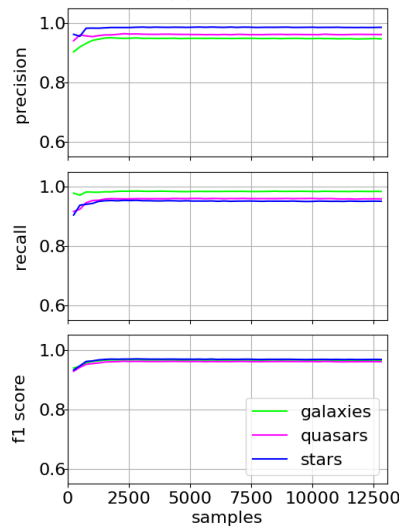
(b)



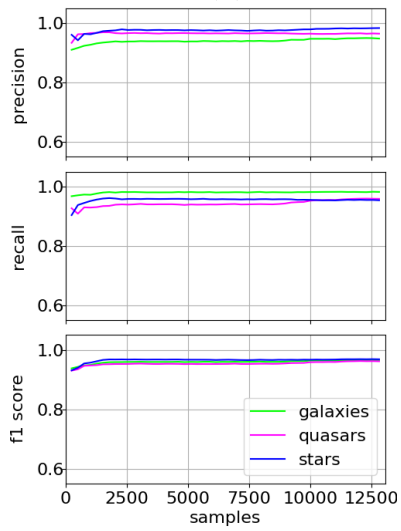
(c)



(d)



(e)



(f)

FIGURE 5.11: Precision, recall, and f_1-score for a balanced multi-class set trained on a random forest. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

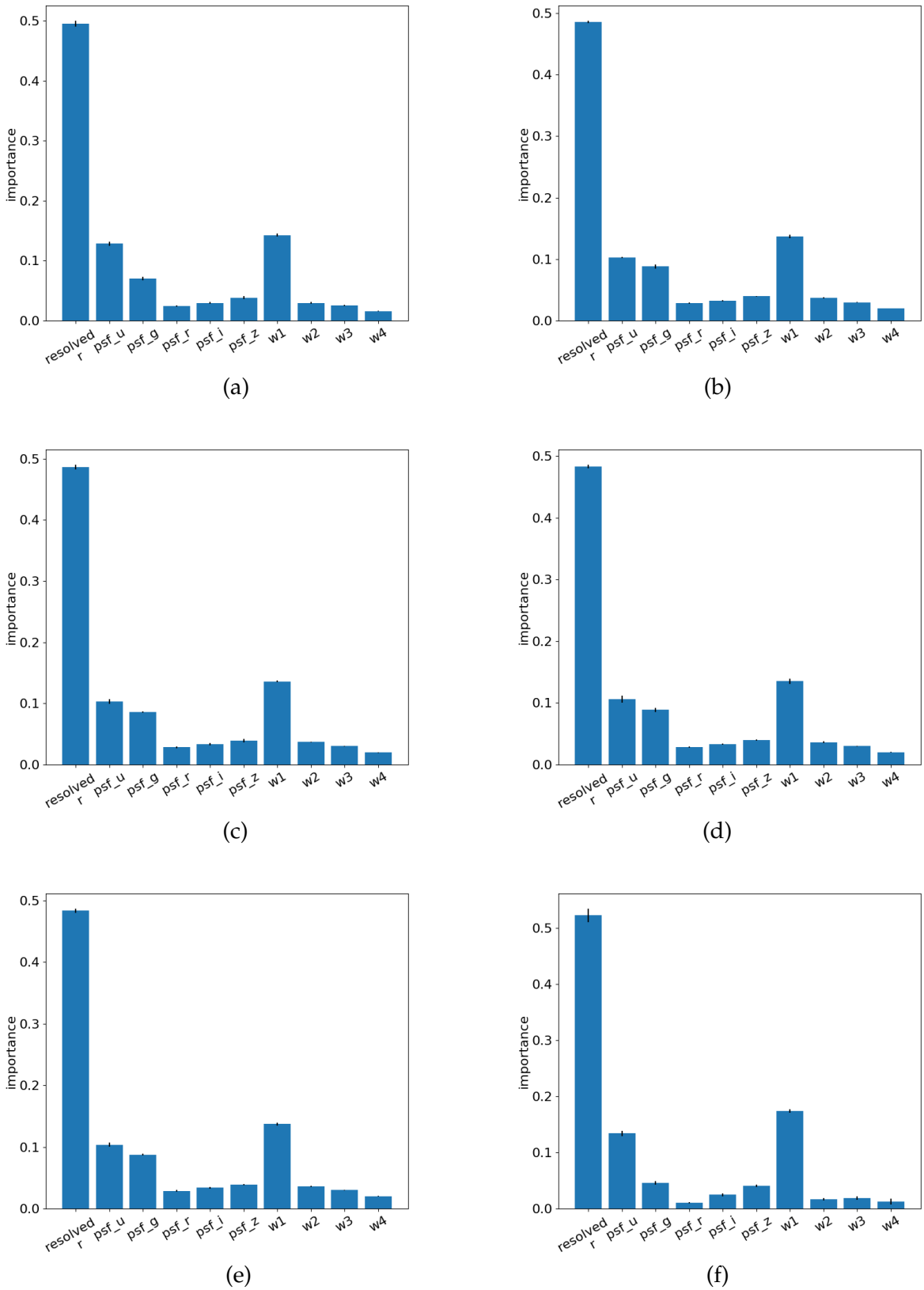


FIGURE 5.12: Attribute importance for a balanced binary set trained on a random forest. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

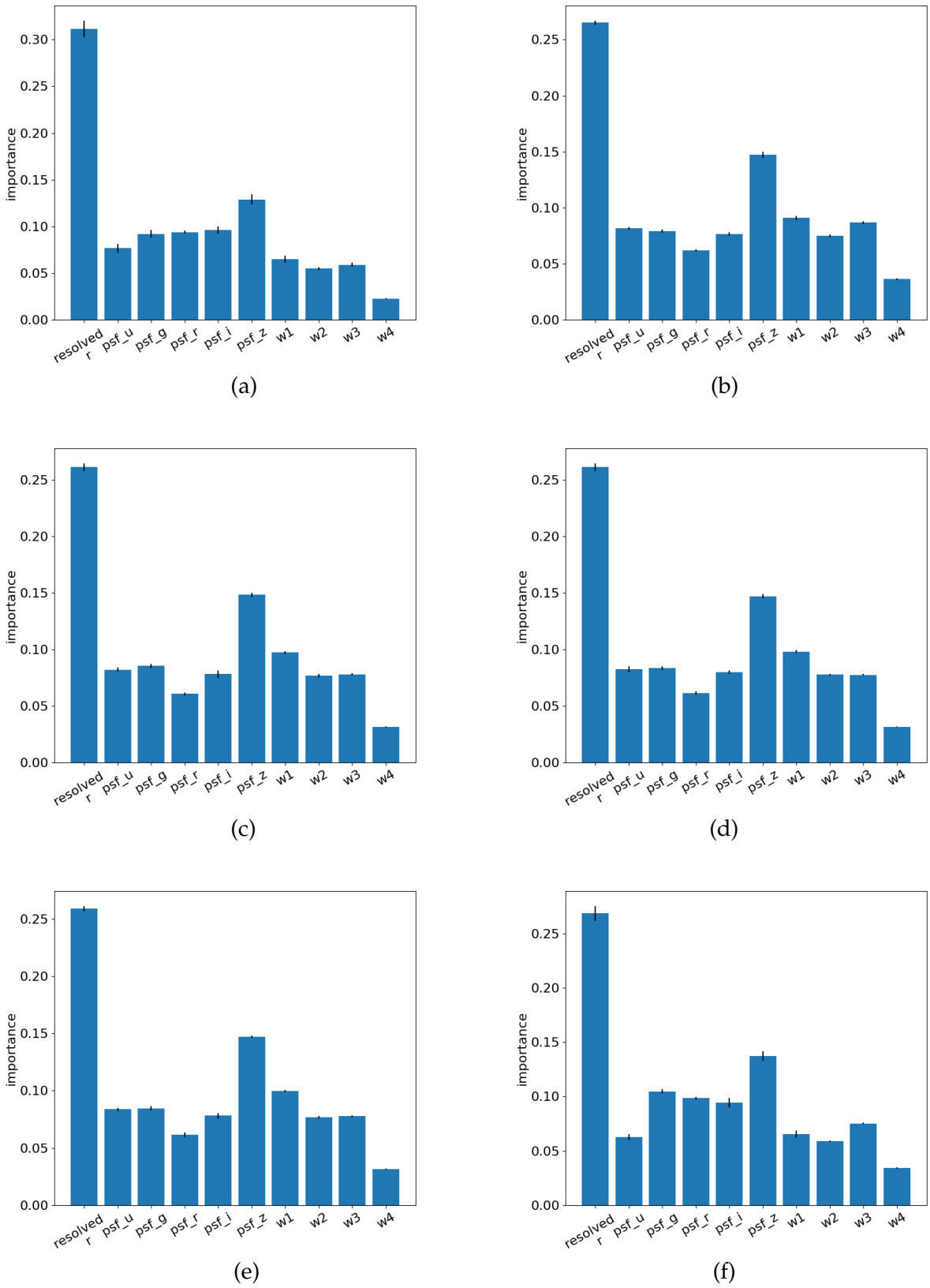


FIGURE 5.13: Attribute importance for a balanced multi-class set trained on a random forest. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

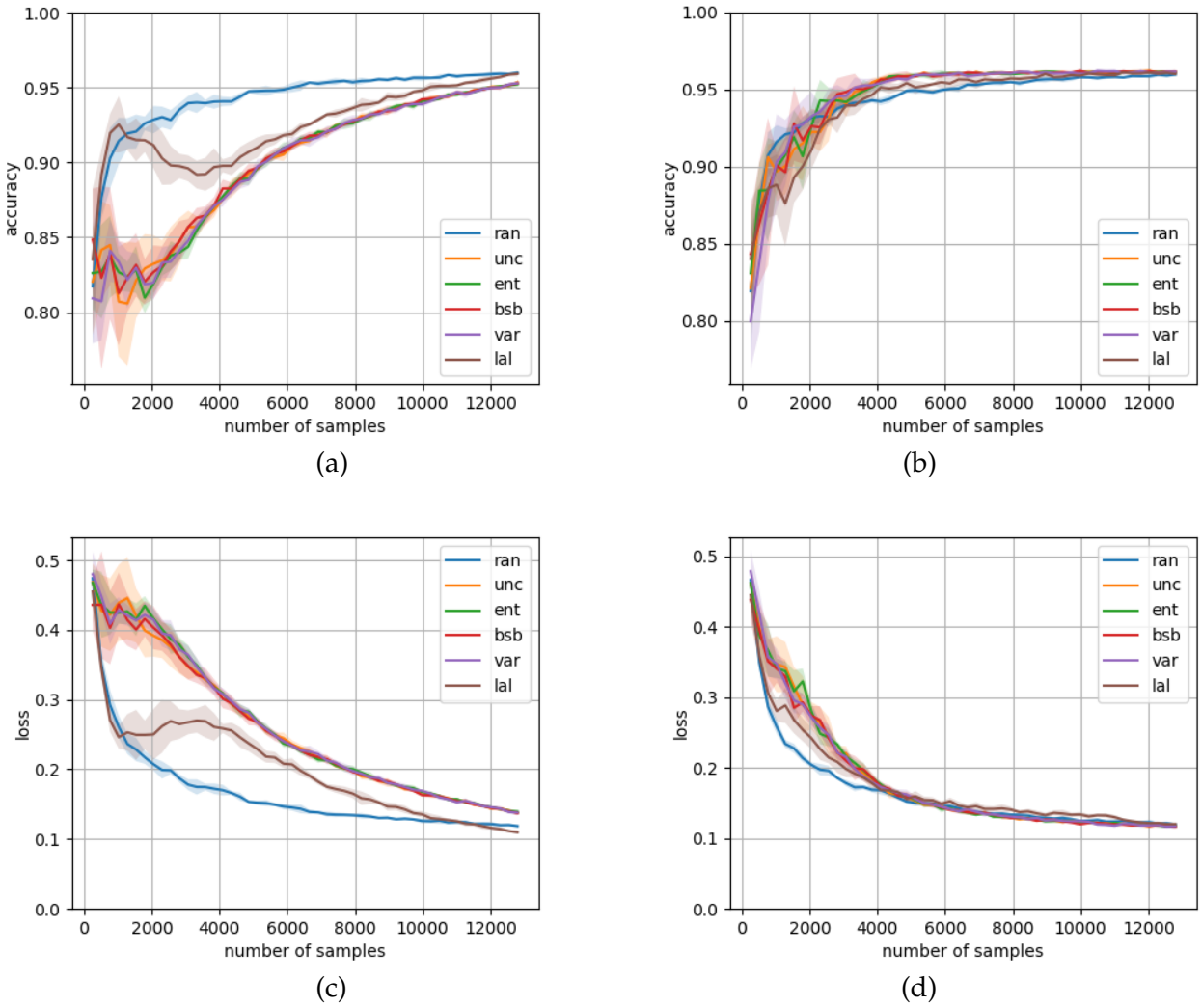


FIGURE 5.14: Accuracy and loss curves for the different active learning methods on a balanced binary set trained using a neural network. The left graphs (a) and (c) represent the training curves for accuracy and loss respectively, the right graphs (b) and (d) represent the validation curves for accuracy and loss respectively.

TABLE 5.13: Class specific performance metrics for the balanced multi-class SDSS dataset - trained on the neural network. The precision, recall, and F_1-score are shown for all classes.

Active Learning	Precision			Recall			F_1-Score		
	galaxy	quasar	star	galaxy	quasar	star	galaxy	quasar	star
ran	0.872	0.954	0.959	0.950	0.894	0.931	0.909	0.923	0.944
unc	0.877	0.968	0.940	0.943	0.909	0.920	0.908	0.938	0.929
ent	0.860	0.956	0.937	0.928	0.908	0.907	0.892	0.931	0.920
bsb	0.873	0.954	0.944	0.933	0.927	0.904	0.902	0.940	0.923
var	0.884	0.957	0.950	0.940	0.902	0.943	0.911	0.929	0.946
lal	0.842	0.976	0.636	0.573	0.751	0.982	0.679	0.847	0.771

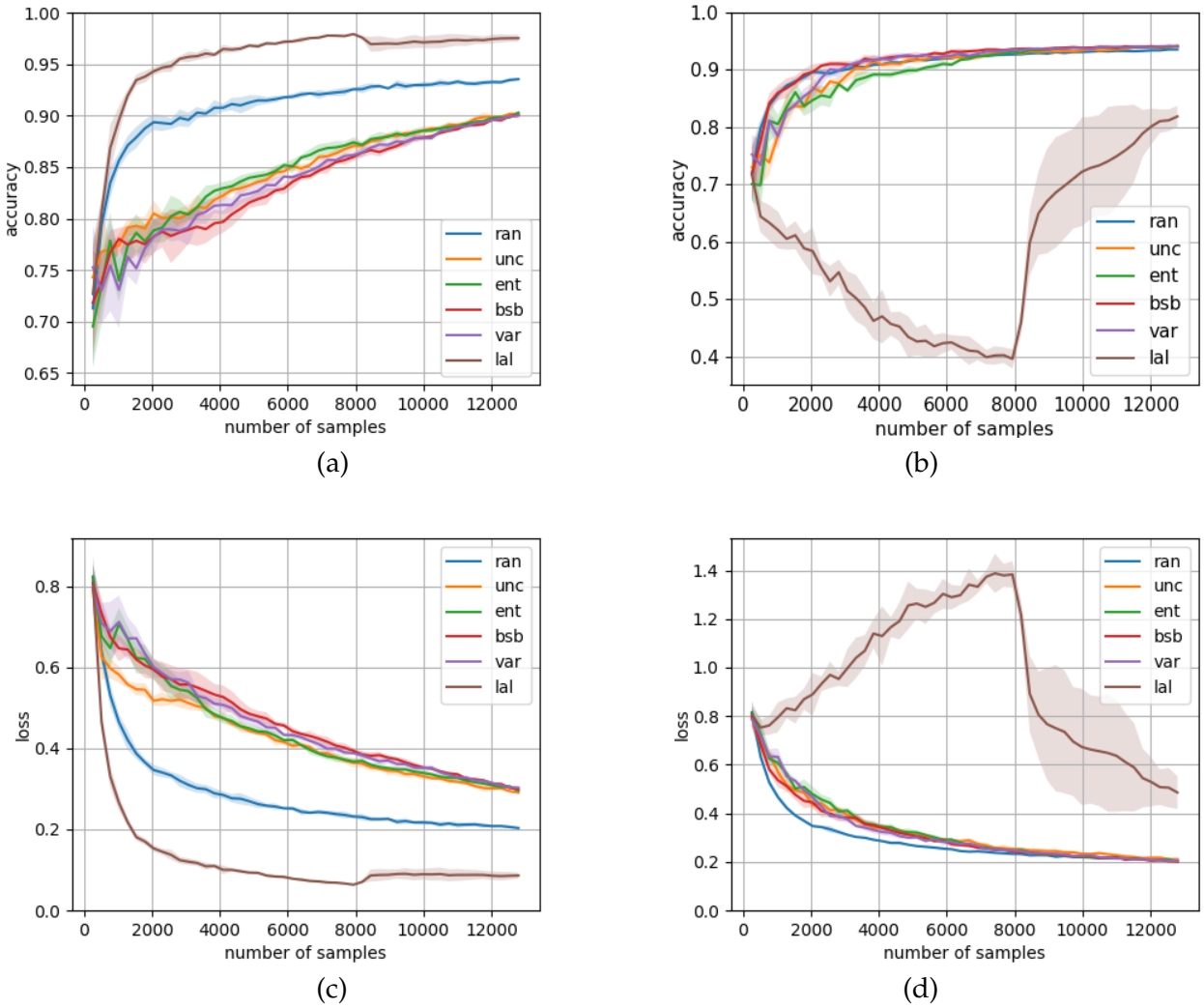


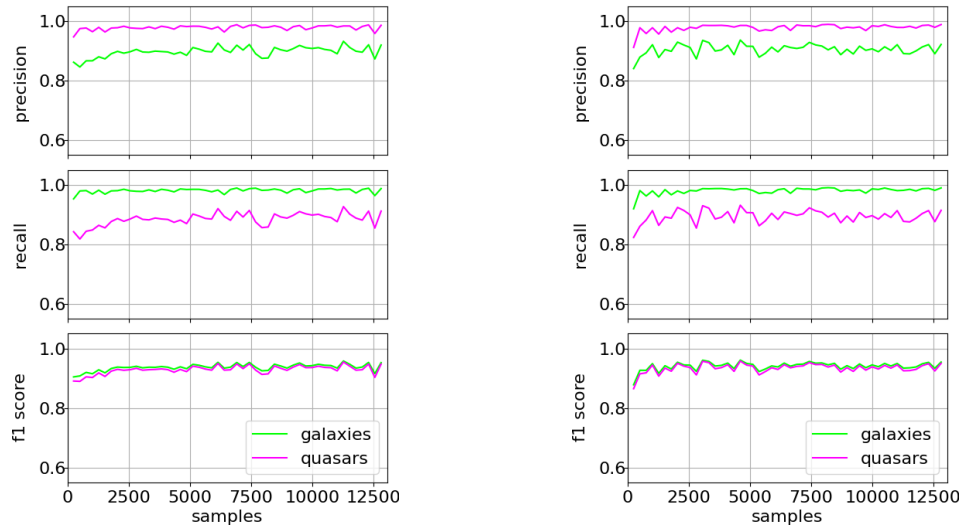
FIGURE 5.15: Accuracy and loss curves for the different active learning methods on a balanced multi-class set trained using a neural network. The left graphs (a) and (c) represent the training curves for accuracy and loss respectively, the right graphs (b) and (d) represent the validation curves for accuracy and loss respectively.

and again this is somewhat down to $resolved_r$ - among the other features reflected in Figure 5.18. Although $W1$ looks to be the most important at a glance, the change in feature importances compared to the $10k$ feature importances in Figure 5.7 indicates that the neural network is no longer relying on a single feature and that it has improved its the view of these other features, especially $resolved_r$ which is now consistently viewed as important rather than the model being unsure of the importance or not. The difference in feature importance could also mean that psf_i and psf_z specify quasars or galaxies, something the random forest may not have picked up on.

The multi-class metrics shown in Figure 5.13 also give quite a bit of variation, with entropy measure and learning active learning standing out. The performance metrics for the learning active learning follows the accuracy and loss curves, so this is not concerning, however the entropy sampling notably has poor galaxy precision and poor star recall. All methods besides the learning active learning do show improved recall and overall F_1-score for stars and galaxies compared to the $10k$ metrics seen in Figure 5.6, meaning that it is actually the class disparity that causes the poor metrics in some active learning methods. The ones to have benefited the most from balancing the datasets are: uncertainty and best-vs-second-best methods, as said before though - the former is not optimised for a multi-class dataset so really the best-vs-second-best method actually loses out the most due to class imbalance. The balanced multi-class feature importances appearing in Figure 5.19 have also changed drastically when comparing against the $10k$ and binary cases. A big thing to note is that psf_u and psf_g are now ranked higher here than the other two cases, indicating that these two features are useful for identifying stars. In the $10k$ dataset, the feature importances could be smothered simply because of the class disparity, but now stars are equal to galaxies the features are more rated highly. The $resolved_r$ feature varies in importance, but now almost equals $W1$ (which is still the most important feature in every method), so clearly when comparing galaxies to stars and quasars, $resolved_r$ is significant.

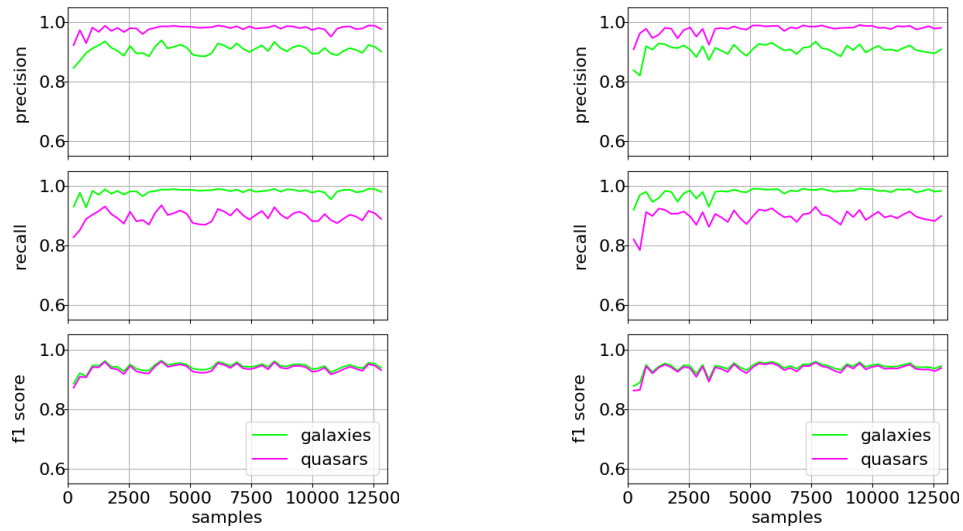
5.3 Chosen Active Learning Methods

Now that the different active learning methods have been assessed for both a neural network and a random forest, the best procedure(s) can be chosen. The first issue to evaluate is the which machine learning algorithm to use.. The ideal active learning method is one that combines speed with efficiency in choosing points. In general, the neural networks took much longer to train in all cases, this is purely because of how a neural network is trained in that the data is passed through multiple times and thus with each iteration of the active learning, the model is trained over the number of epochs making each set take too long. Ideally the neural network can be used to take advantage of different hardware such as a GPU, however as the active learning was designed for more general use, it could not utilize such hardware. That does not mean the neural network method was entirely in vain, as the feature importance evaluation helped in understanding what a machine learning model might be looking for when identifying galaxies, quasars, or stars.



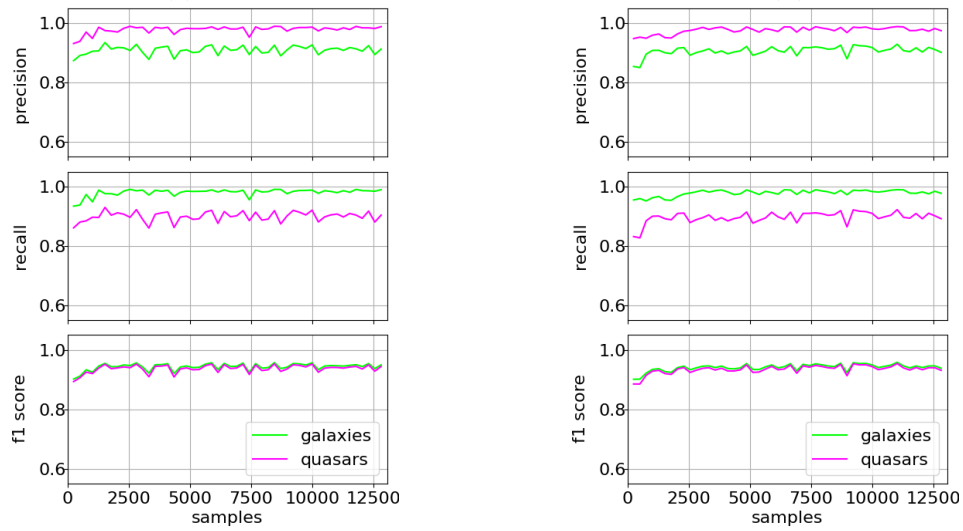
(a)

(b)



(c)

(d)



(e)

(f)

FIGURE 5.16: Precision, recall, and f_1-score for a balanced binary set trained on a neural network. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

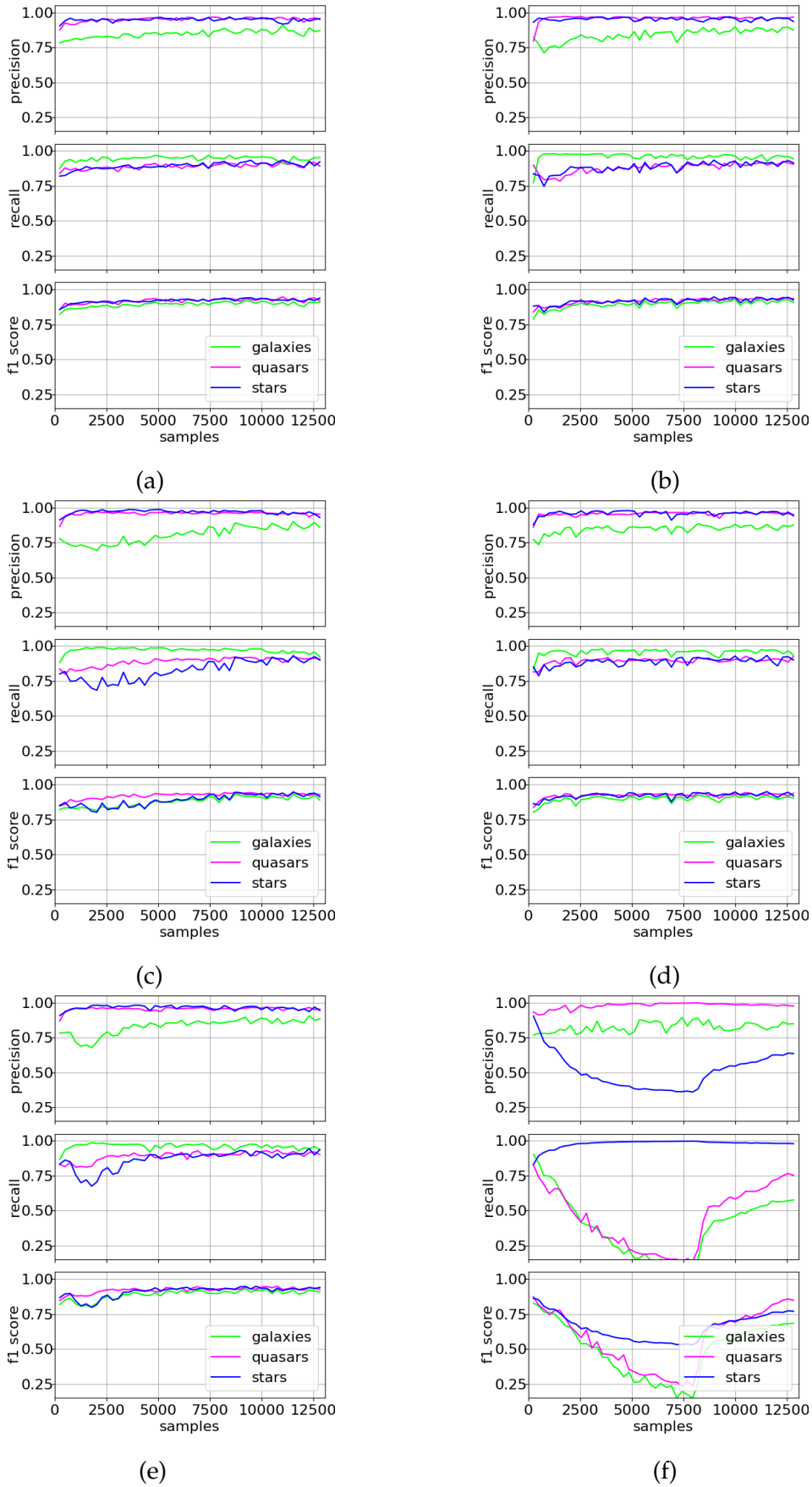


FIGURE 5.17: Precision, recall, and f_1-score for a balanced binary set trained on a trained on a neural network. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

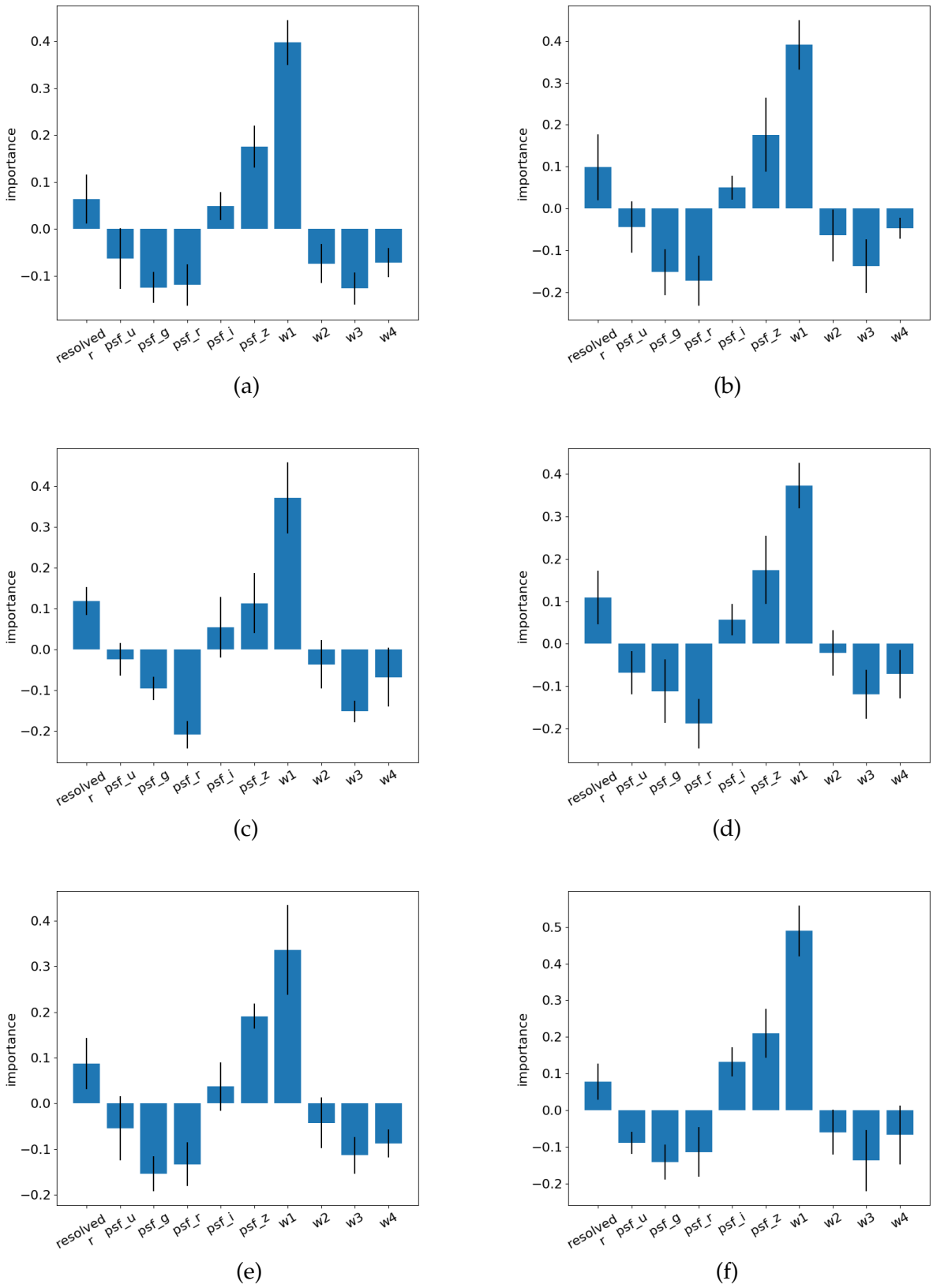


FIGURE 5.18: Attribute importance for a balanced binary set trained on a neural network. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

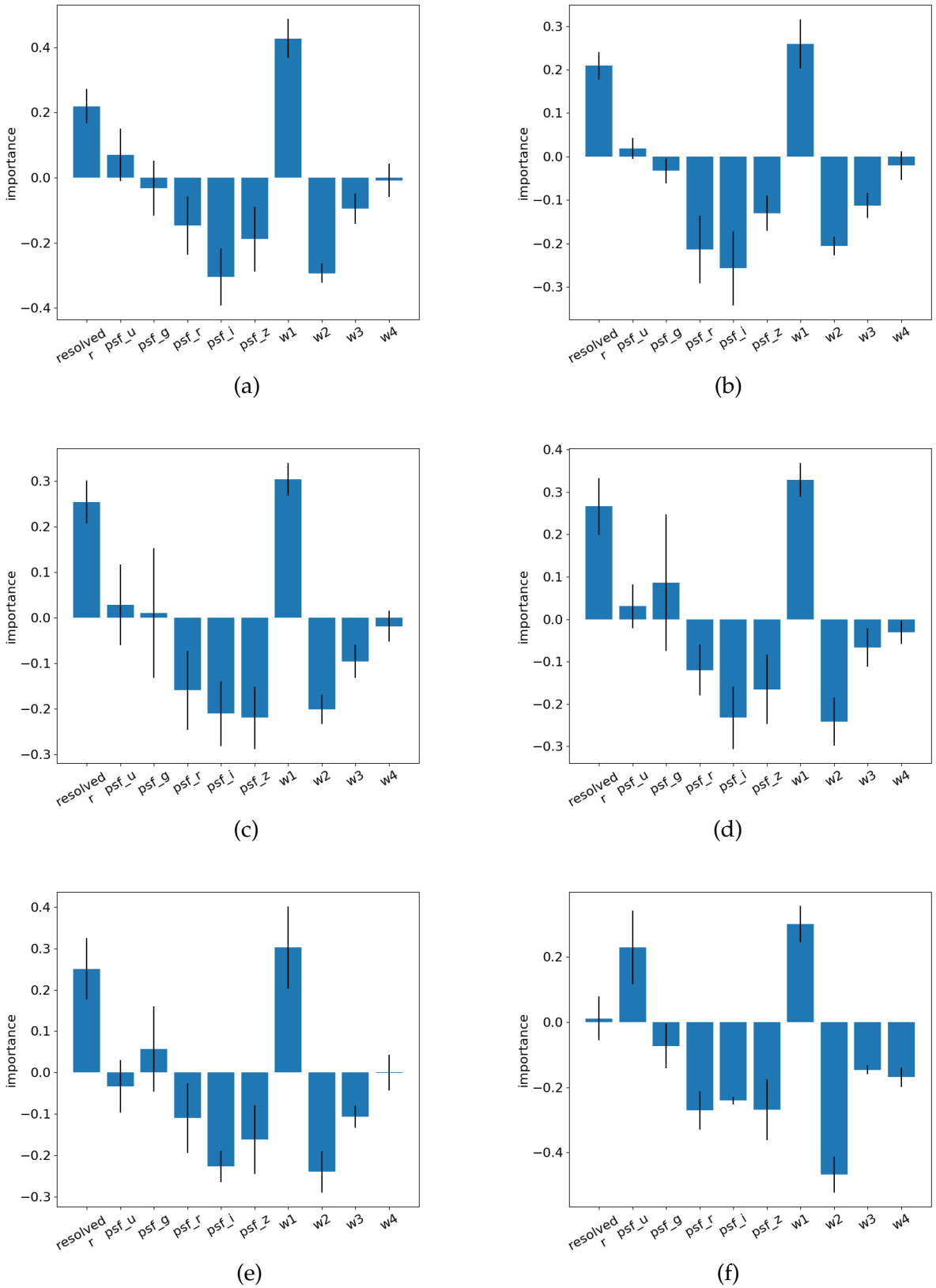


FIGURE 5.19: Attribute importance for a balanced multi-class set trained on a neural network. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

Furthermore it is useful to know that the features chosen for a neural network learning active learning regressor are more useful than the features chosen for the random forest, as the learning active learning was generally better when it came to the multi-layer perceptron. In general, the variance sampling has been optimal for the neural network, such conjecture is not only based on the variance sampling metrics either - but also the fact that variance as a feature was used in the learning active learning regressor (which was more effective given the circumstance). To summarise, the random forest classifier was chosen mostly because of the time. The accuracy of the random forest was also generally better and more stable, being much less reliant on the number of samples added.

The next item on the checklist is which active learning methods are most effective. Just looking at the random forest classifier in this case is naive and does not take into account why certain methods can be so effective. Before attempting to judge the best algorithm, some can be ruled out. Random sampling is not an active learning method and it has already been established that actually using active learning helps in improving the model. Uncertainty sampling is the next method to be ruled out; although it does produce similar stats to the other proven approaches, it is not exactly designed around a multi-class system and further investigation would be needed to fully understand why it does so well, although it is likely because the method still utilises probability in the equations. This leaves entropy measure, best-vs-second-best, and variance reduction. Variance reduction is a more computationally expensive method yet achieves results slightly below the first two just mentioned, it is also much more effective when applied to the neural network rather than the random forest. So if using a neural network it would be recommended, while here - it is not. This leaves entropy measure and best-vs-second-best fit. Between these two, entropy outperforms the other when looking at quasars as the precision is slightly higher, while best-vs-second-best outperforms in precision for stars. However both precision scores in Table 5.7 for stars are above or equal to 0.99, making them both much more precise when it comes to stars anyway, framing the quasar precision to be of greater significance. Furthermore, entropy measure in Figure 5.2 has shown a slightly slower accuracy increase as the labelled pool has grown which has already been established to be a small detriment to the latter stages of active learning. Both methods do have a hard limit in all cases though, as can be perceived in all accuracy curves. Because the end results are so similar, it should thus depend on what class would be preferred in the test results. Thus the entropy measure method has been chosen, and the best-vs-second-best dropped. They are nearly interchangeable, but the minute improvement in precision for quasars - which have less precision overall than stars - edges this out. It might be easier to visualise with values; entropy measure has a precision for stars at 99.998% of the best-vs-second-best method, while the best-vs-second-best has a precision for quasars at 99.894% of the entropy measure method. Both are great, however using two would be inefficient.

The learning active learning has been ignored in the previous paragraph. This is because it is also being used in the full dataset. The balanced multi-class dataset has shown in Figure 5.9 that perhaps there were not enough samples for the learning active

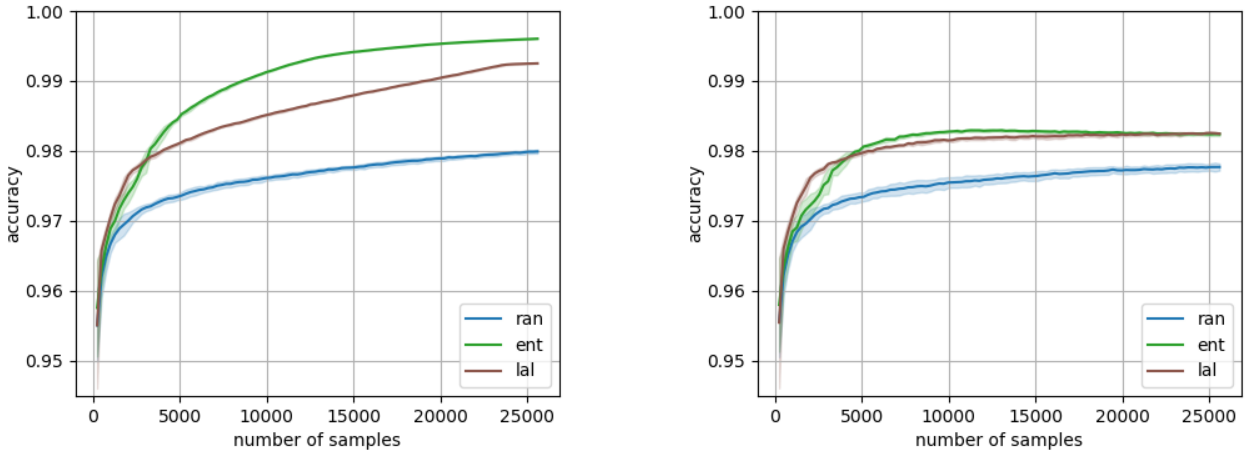


FIGURE 5.20: Accuracy curves for the choice active learning methods on the full dataset with a large pool. The left graph represents the training curves and the right graph represents the validation curves.

TABLE 5.14: Accuracy scores from the chosen learning methods on the full dataset - trained with a the random forest.

Active Learning	Average Accuracy
random sampling	0.9780
entropy sampling	0.9822
learning active learning	0.9824

learning to be utilised. Also the accuracy curves in Figure 5.2 give the impression that the learning active learning has not stagnated. After 2000 samples, the learning method does not increase by much, however it is still rising. This is then backed up by Figure 5.9. Thus the active learning methods chosen to represent active learning on the full dataset are: entropy measure and learning active learning. Random sampling is also applied at the same time to verify that these algorithms are performing.

5.3.1 The Full Dataset

To expand on the current best methods, the active learning has been increased. Like the balanced datasets, the initial set contains 256 points, then 256 points are moved from the unlabelled pool to the labelled pool each time and the number of active learning iterations has been increased to 99, making the final dataset 25,600, still less than Clarke's recommendation of 300,000 values. Models were trained ten times and the results averaged over. The final accuracy scores shown in Table 5.14 show great promise. Both active learning methods are above the random sampling so the points chosen to train on are clearly better.Because the other methods were so similar in end result to entropy sampling - it is not too far-fetched to presume the other methods would achieve the same

TABLE 5.15: Class specific performance metrics for the full SDSS dataset - trained on the random forest. The precision, recall, and F_1-score are shown for all classes.

Active	Precision			Recall			F_1-Score		
Learning	galaxy	quasar	star	galaxy	quasar	star	galaxy	quasar	star
ran	0.980	0.950	0.988	0.994	0.927	0.945	0.987	0.938	0.967
ent	0.984	0.958	0.991	0.995	0.940	0.959	0.990	0.949	0.975
lal	0.987	0.952	0.987	0.993	0.946	0.963	0.990	0.949	0.975

result. One surprising takeaway from Table 5.14 is that the learning active learning is above entropy measure, proving the speculation that it needed more samples to be effective. This is partially reflected in the accuracy curves in Figure 5.20 where the training curve for entropy sampling is higher, yet the validation curves are equal. Perhaps less data is needed to 'complete' the model too, as although the training curves increase as samples are added, the validation curves for the learning active learning have flattened at 20,000 values added, showing the model is not actually improving with more datasets, thus the hard limit from earlier has made an appearance. This hard limit might be able to be altered though, as the features for the learning active learning were clearly better for the neural network, so maybe altered features within the regressor could heighten the limit.

Although the overall accuracy for the learning active learning is above entropy measure, the metrics are more even as seen in Table 5.15. Learning active learning outperforms entropy measure for galaxy identification but not for quasars or stars, indicating it has a penchant for non-point like objects, but fails (comparatively) to always accurately identify point like objects. The method is correct in understanding the different classes when considering the ratio of the three classes as shown with the recall. Entropy measure almost has an opposite effect, where it is not as successful in identifying galaxies, but can easily identify quasars and stars. Table 5.15 also confirms entropy sampling as the right choice for the active learning, as the star precision has now increased to the same level as best-vs-second-best while the quasar precision has increased massively. The F_1-scores are the same across the board though, so the different class metrics cancel each other out. The F_1-scores are also close to Clarke's scores of: 0.990, 0.953, and 0.977 respectively - despite using much less data.

Figure 5.21 give the metrics as samples are added. This time, entropy measure seems to be much slower to result in equivalent scores. Unlike the $10k$ dataset; the learning active learning is relatively efficient in how fast the metrics level out, showing that giving an increased pool selection might be the best choice for the method. The feature importance graphs in Figure 5.22 have actually changed quite a bit. They all still display the same trends, and the random sampling has maintained similar values - however the features show more equal importance for the two active learning methods indicating they both chose feature sets that do not depend on $resolved_r$ to the same degree.

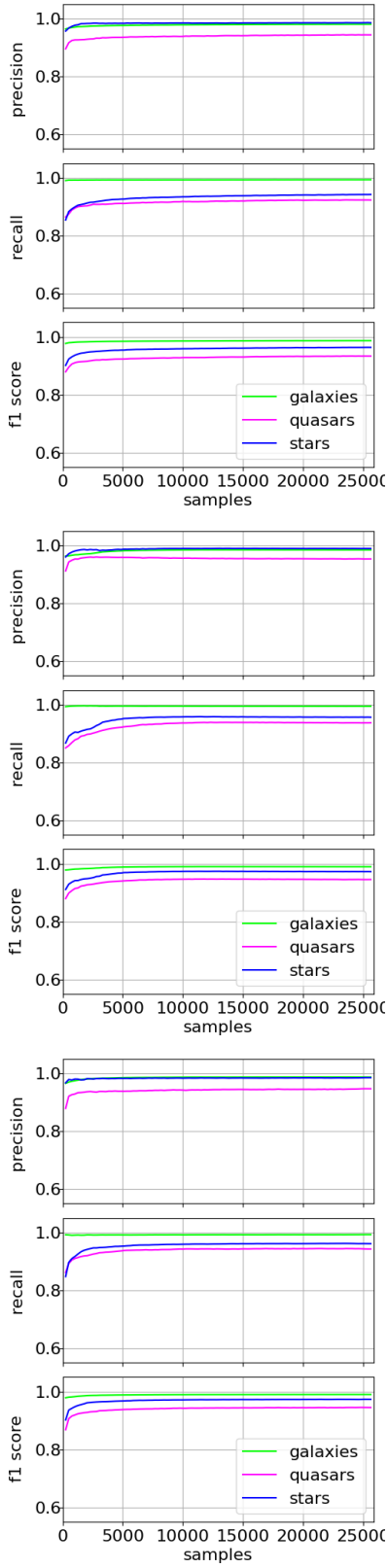


FIGURE 5.21: Precision, recall, and f_1-score for the full unlabelled pool of data selected for training - trained on a random forest. The top graph shows random sampling, the middle shows entropy measure, and (f) for learning active learning.

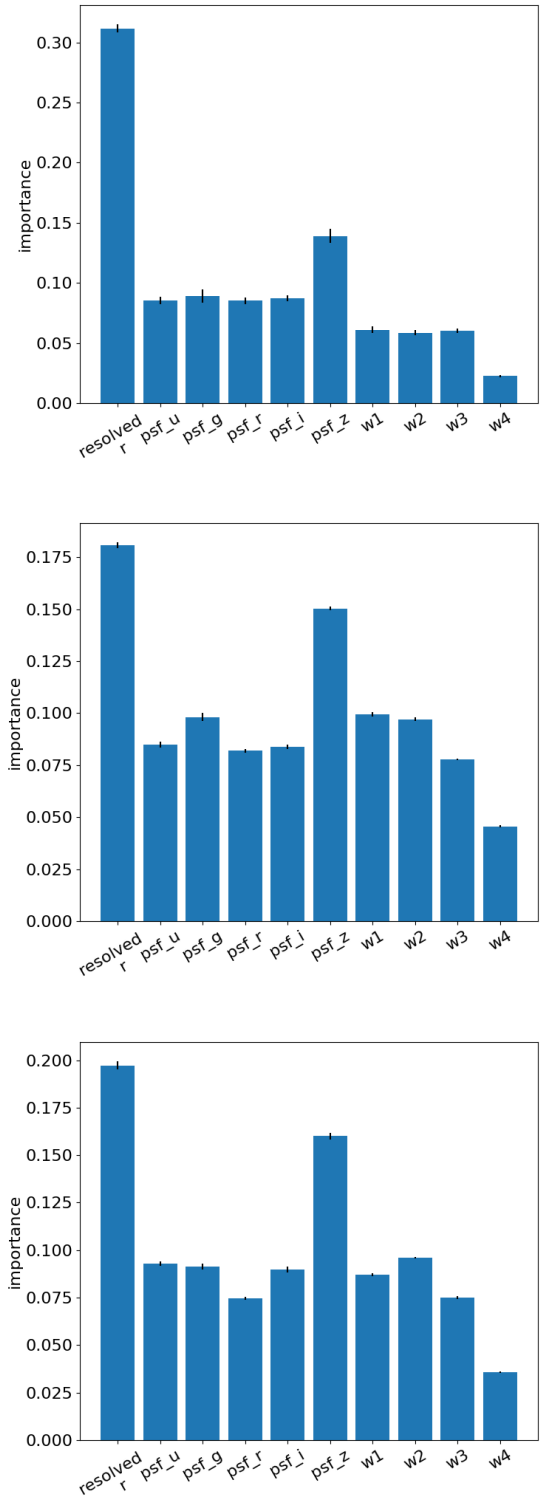


FIGURE 5.22: Attribute importance for the full unlabelled pool of data selected for training - trained on a random forest. The top graph shows random sampling, the middle shows entropy measure, and (f) for learning active learning.

Chapter 6

Conclusion

As evidenced in Chapter 5, active learning is a tool that has proved to be useful in decreasing the amount of data needed to train a model. Even on the 10k dataset the active learning achieved good results for a random forest, and decent results with the neural network. The final results are also comparable to Clarke's despite using roughly one twelfth of the data. In general, when the active learning had high training accuracy scores initially; the validation scores would not increase as much as when the training accuracy scores were lower. Active learning therefore relies on the model being somewhat unconfident in the probability calculations.

The imbalanced and balanced cases for both algorithms demonstrated the class imbalance is not a problem that is easily solved even when using active learning. Though it is strange that the active learning does perform better overall on unbalanced sets. This could be explained by the fact that when training on a balanced dataset, there might not be a class that the model considers to be much easier or harder to label, thus the active learning is unsure of which points to move from the unlabelled pool, regardless of the active learning algorithm used. What the balanced cases do show is that the models potentially prefer identifying point like objects as the quasar precision goes above the galaxy precision in these cases. Though it could also mean that the addition of stars makes determining between the two point like classes hinders the progress. A proposed solution would be to first analyse point like objects and train a model to identify these, and then use the model to classify the point like objects instead.

The neural network was notably worse than the random forest in all tests involving the SDSS data. Within the scope of the neural network, the learning active learning vastly improved and ended in a similar league to the other active learning methods. This was not seen in the random forest where the learning active learning was worse in every one of the balanced cases and also the 10k dataset. As mentioned, the variance sampling also outperformed in the neural network, and because the learning active learning used variance as a feature - variance must be an important factor in active learning when applying a neural network. In fact the neural network is able to be improved through a multitude of different factors even just by changing the hyper-parameters. Of course this would increase training times to an unreasonable level. But, if the active learning was able to be

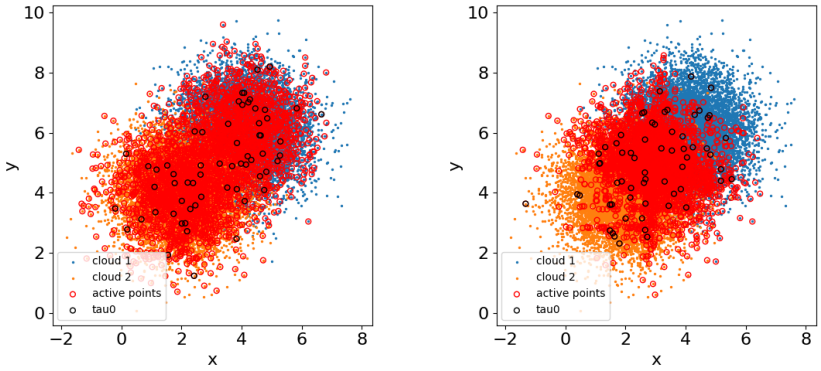
employed internally within the training of the neural network, it could vastly improve performance while still keeping the training times low. This could be done at the end of each epoch or a set number of epochs. It would then allow more customisation in the model when used in the future.

The main takeaway is how effective the random forest was when combined with active learning. The $10k$ results, of which the final labelled pool was 3200, performed admirably for all active learning methods. The final dataset used, of which the active pool totalled $25,600$, had comparable results to Clarke's usage of the random forest with $300,000$ datapoints - proving the effectiveness of the chosen active learning methods. The pool could potentially be reduced as well, because the validation accuracy levels out around $20,000$ points for both chosen active learning methods, and stops rising sharply after $10,000$ points. A hard limit seems to have been reached with the entropy measure and learning active learning, however the learning active learning has the capability of increasing this limit through the regressor features. The learning active learning regressor was a more general one that had been designed for both the Gaussian clouds and the SDSS dataset, it also allowed for binary classification. This binary classification was poor across the board however, and not as comparable to the learning active learning results the method is based off of, showing how an unoptimised regressor might not perform so well. This is even more evident with the neural network regressor features which are presented as more adequate features. All of the code is in the Github, found here: https://github.com/as595/SDSS-AL-MSc/releases/.

To conclude; with a bit of tweaking, the learning active learning method could be the algorithm needed to firmly establish the use of a random forest classifier as the classification method for the SDSS or for other surveys with similar outputs. It would likely require less than $25,000$ correct values for training and could achieve an average accuracy of above 98.24%. The F_1-scores for galaxies, quasars, and stars can likewise increase above 0.990, 0.949, and 0.975 respectively. The entropy measure method could also give these scores, however it has not indicated much room for growth like the learning active learning has.

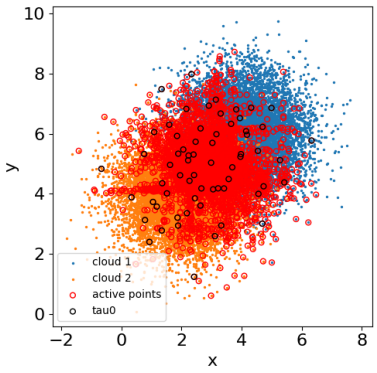
Appendix A

Appendix A

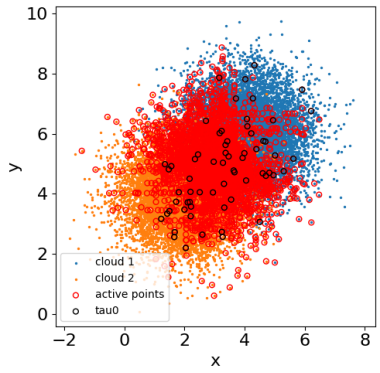


(a)

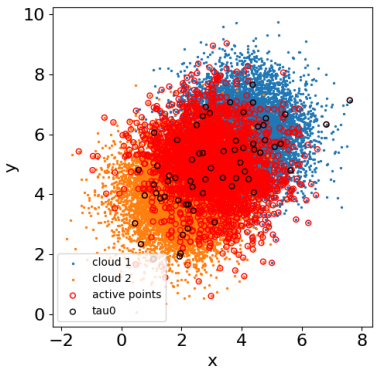
(b)



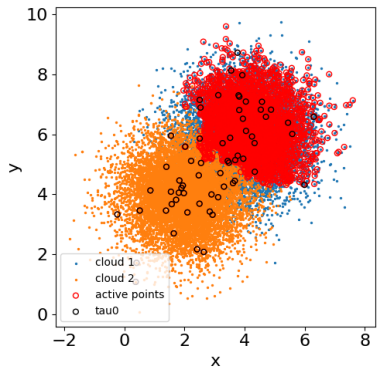
(c)



(d)



(e)



(f)

FIGURE A.1: The labelled points selected for active learning for balanced binary Gaussian clouds trained on a random forest. Designated: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

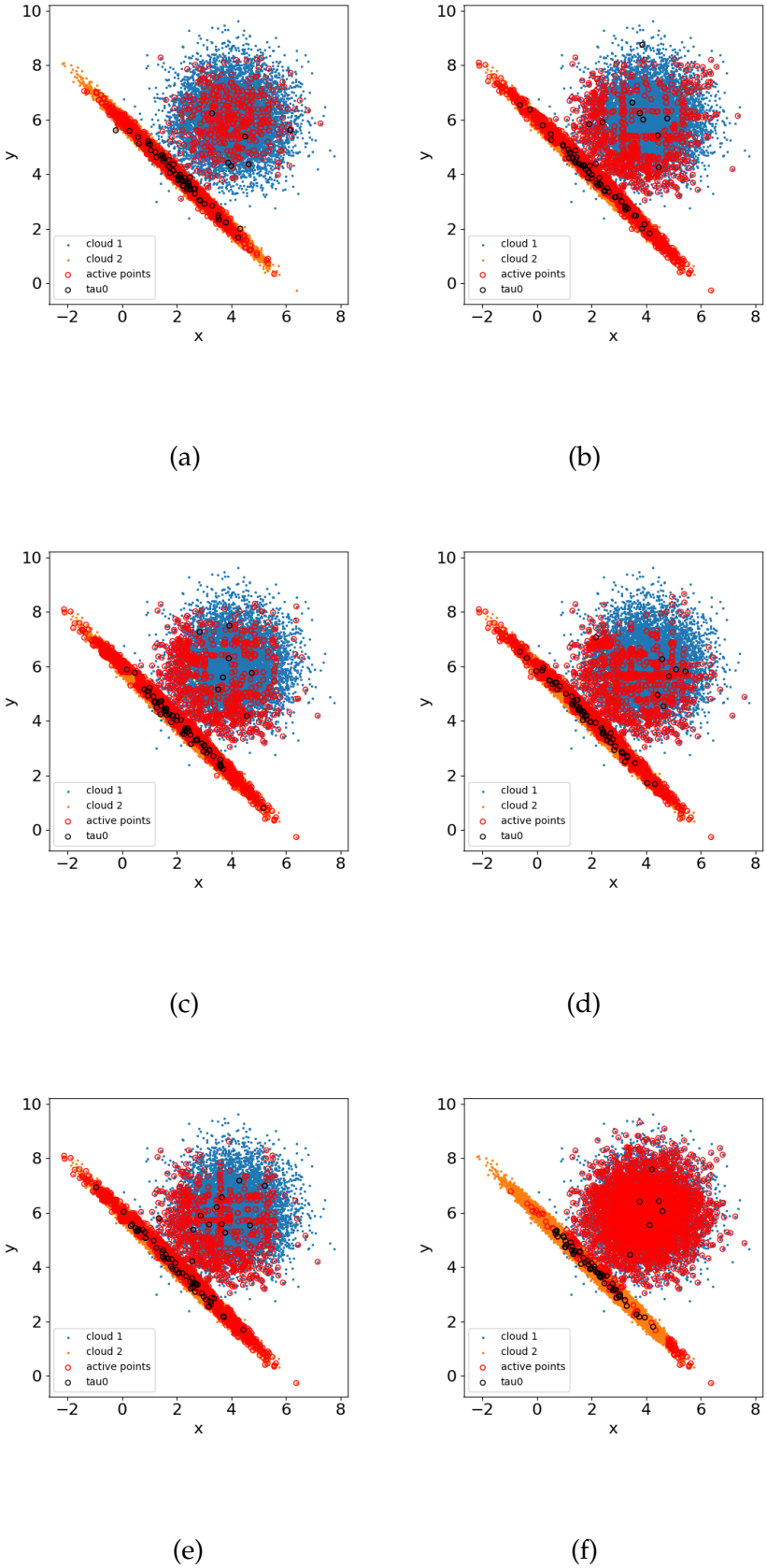
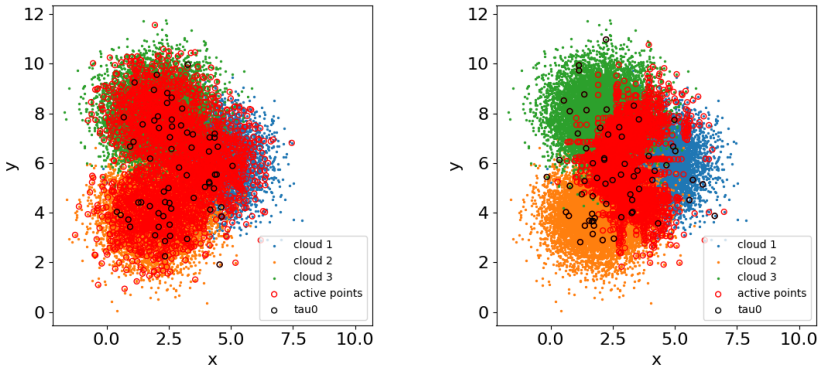
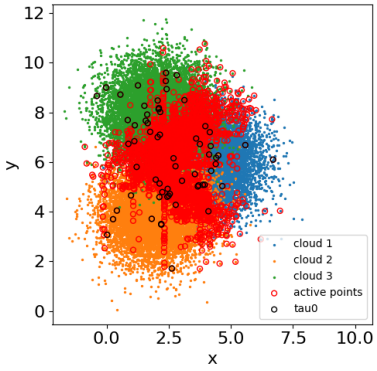


FIGURE A.2: The labelled points selected for active learning for imbalanced binary Gaussian clouds trained on a random forest. Designated: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

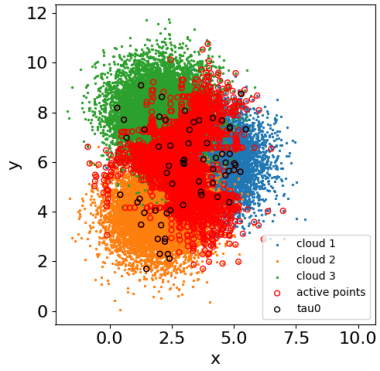


(a)

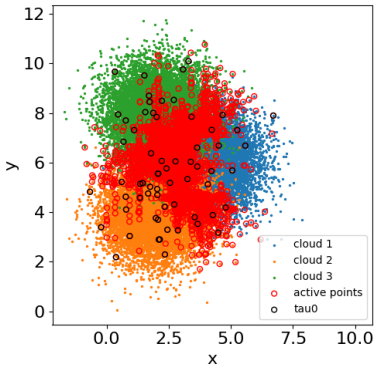
(b)



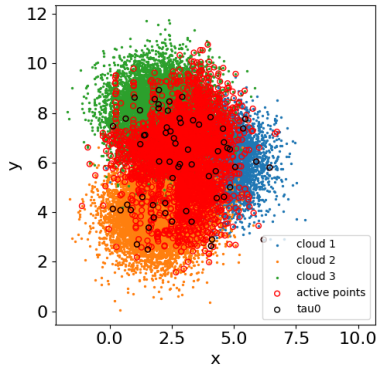
(c)



(d)

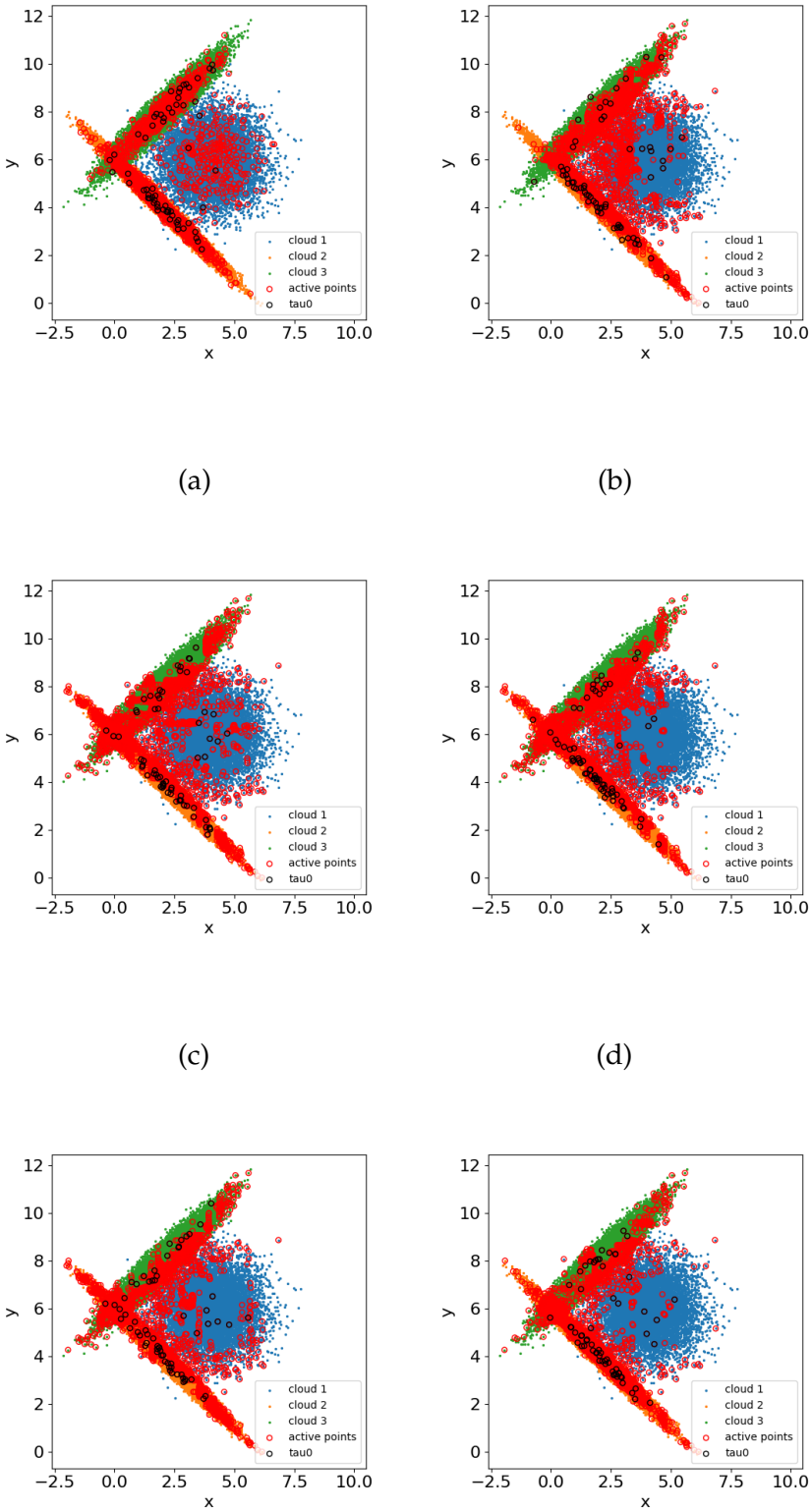


(e)



(f)

FIGURE A.3: The labelled points selected for active learning for balanced trinary Gaussian clouds trained on a random forest. Designated: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.



(a) (b)

(c) (d)

(e) (f)

FIGURE A.4: The labelled points selected for active learning for imbalanced trinary Gaussian clouds trained on a random forest. Designated: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

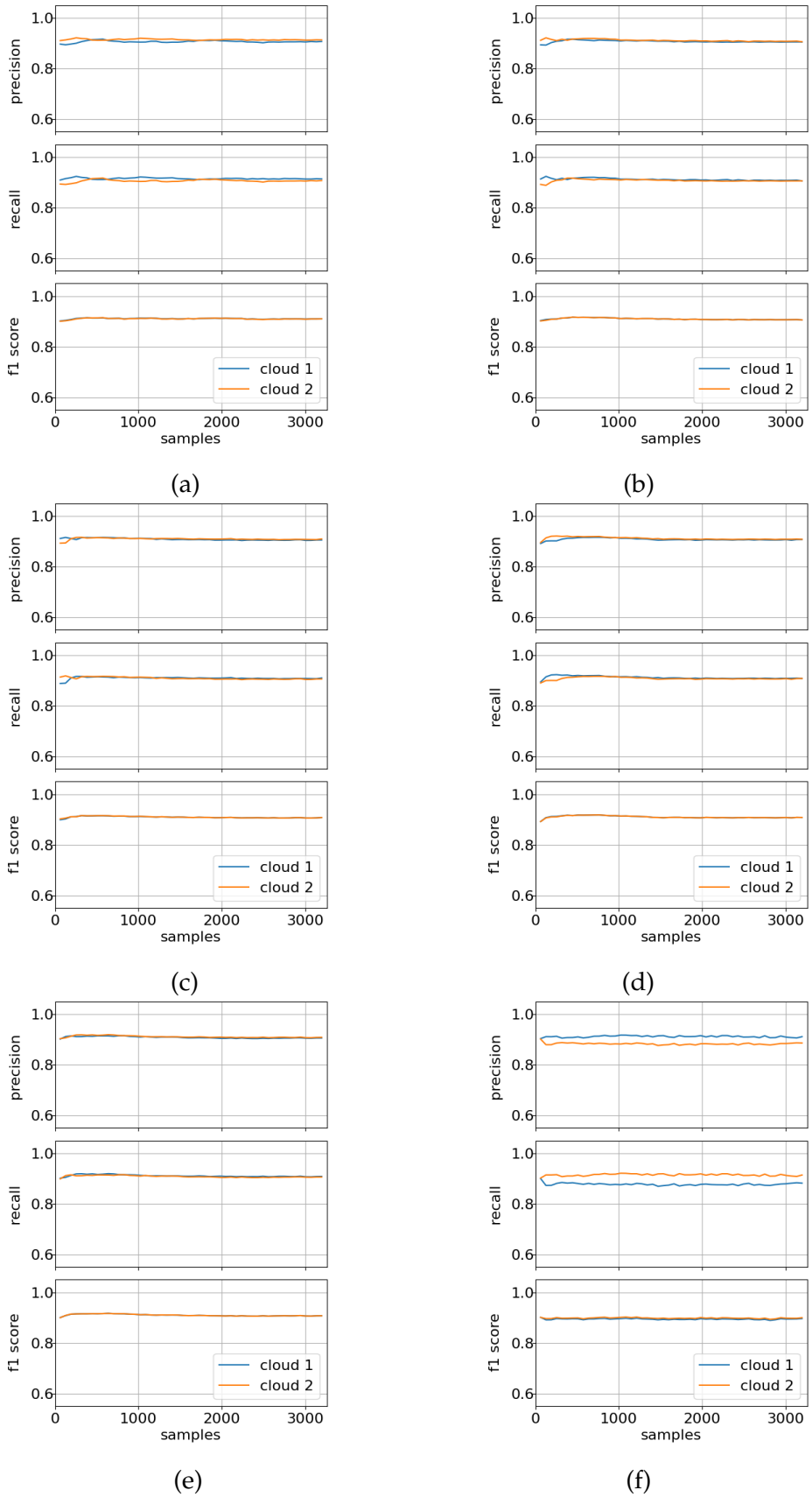


FIGURE A.5: Precision, recall, and f_1-score for balanced binary Gaussian clouds trained on a random forest. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

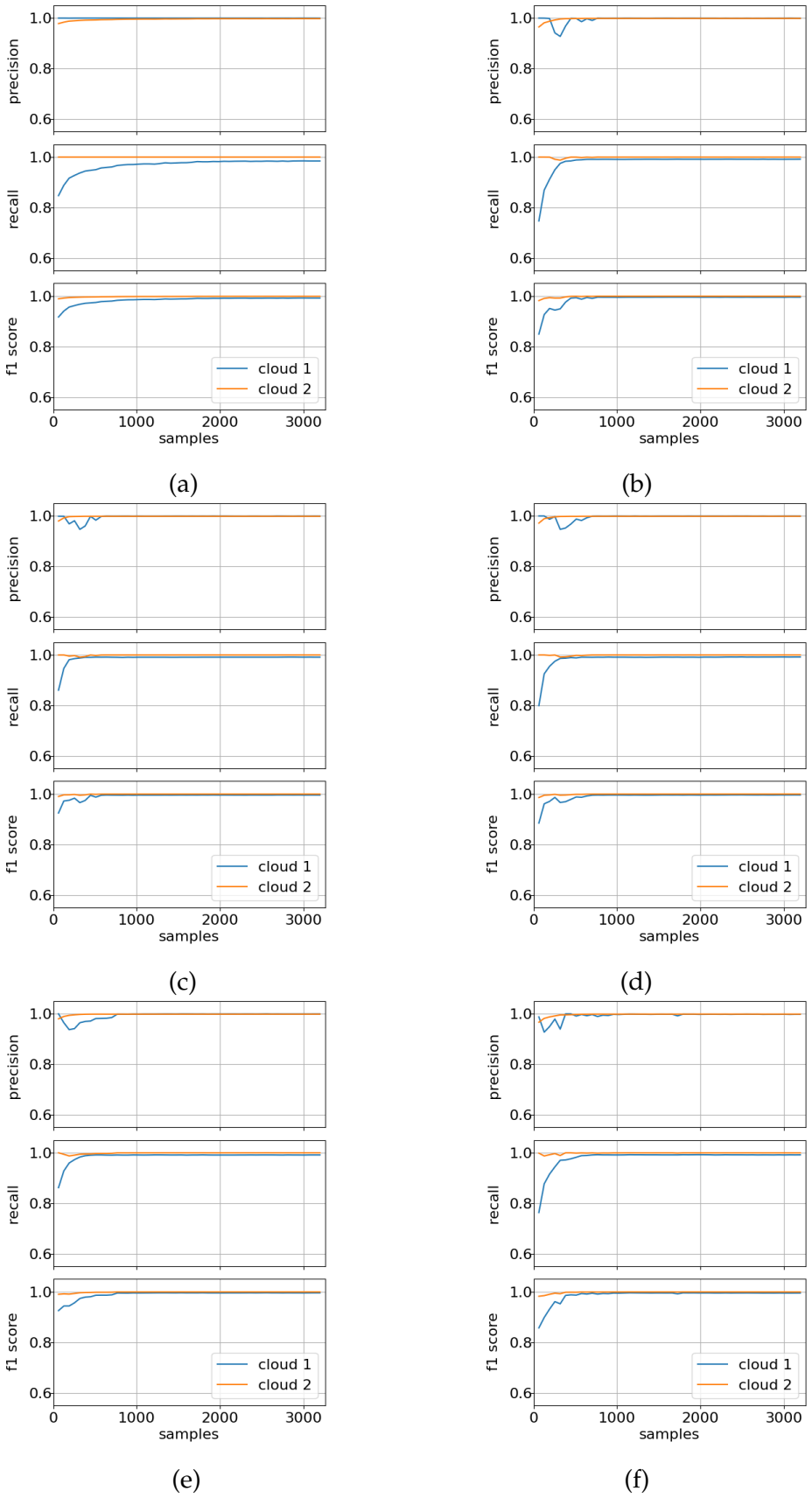


FIGURE A.6: Precision, recall, and f_1-score for imbalanced binary Gaussian clouds trained on a random forest. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

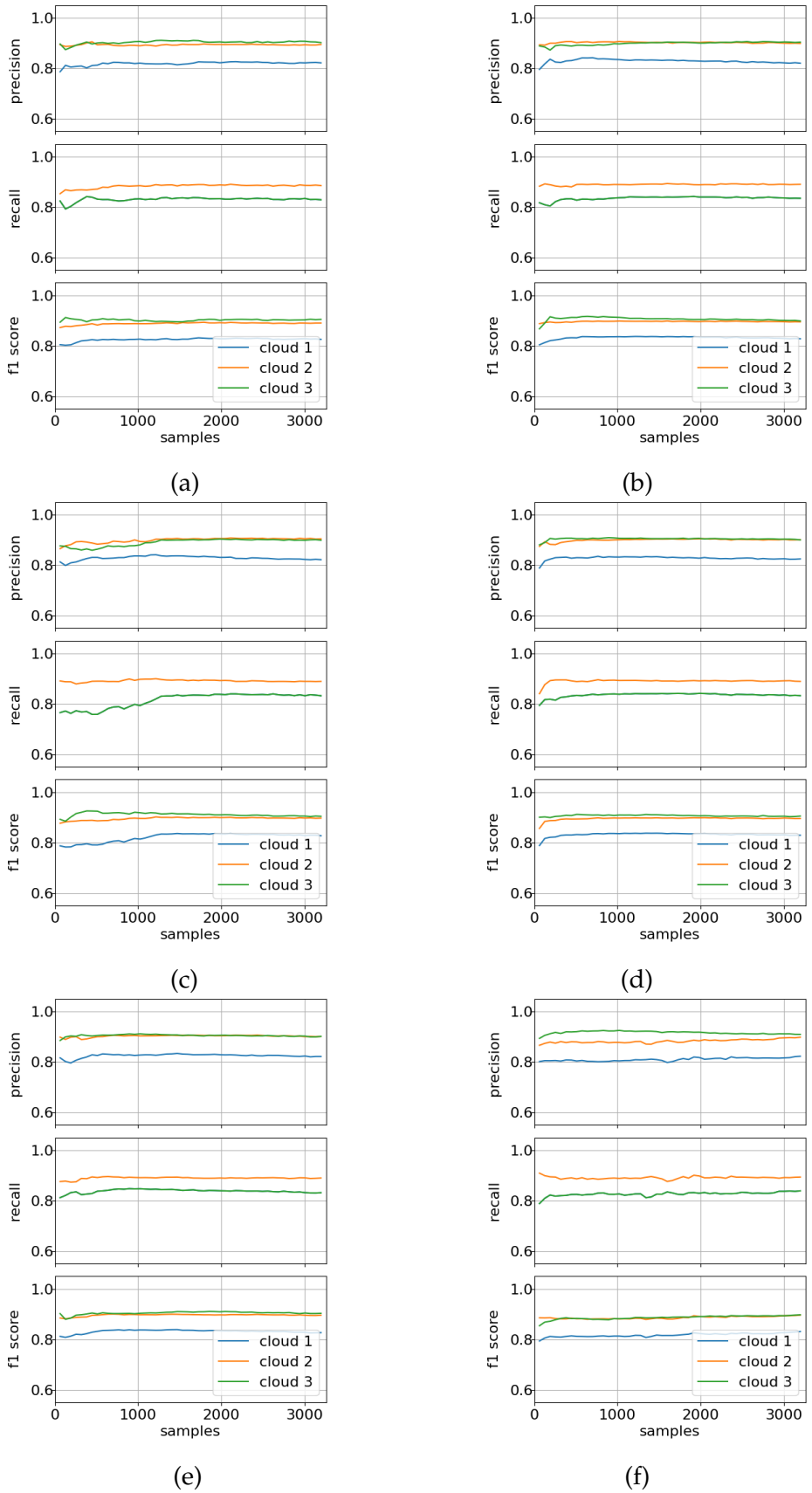


FIGURE A.7: Precision, recall, and f_1-score for balanced trinary Gaussian clouds trained on a random forest. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

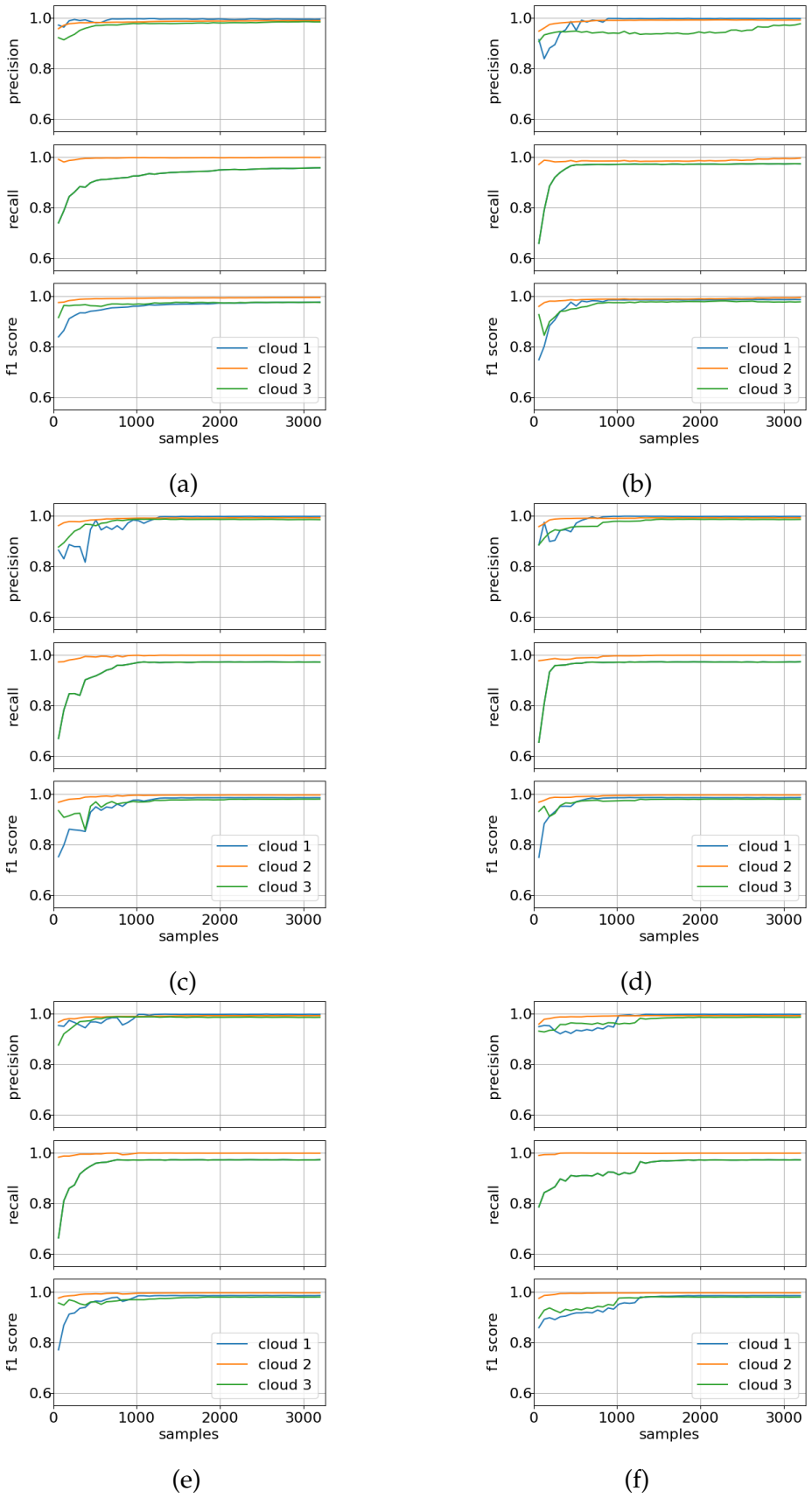
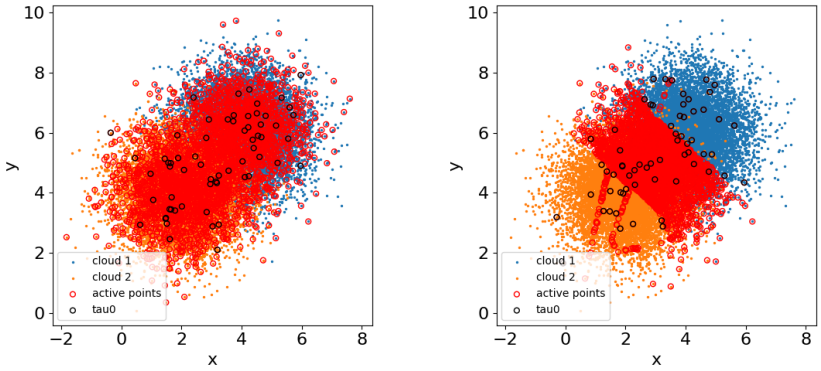
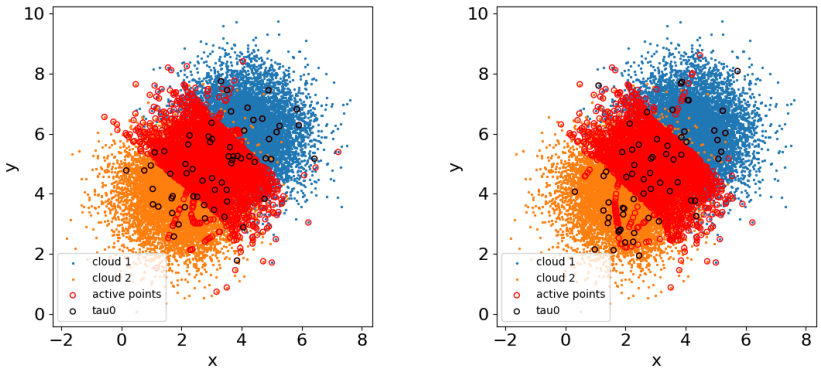


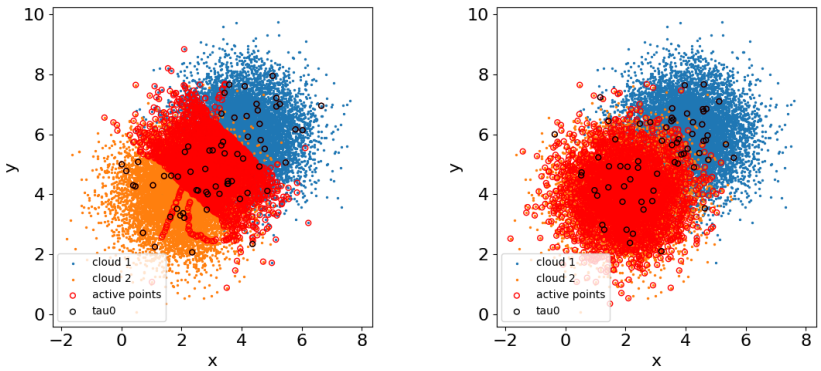
FIGURE A.8: Precision, recall, and f_1-score for imbalanced trinary Gaussian clouds trained on a random forest. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.



(a) (b)



(c) (d)



(e) (f)

FIGURE A.9: The labelled points selected for active learning for balanced binary Gaussian clouds trained on a neural network. Designated: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

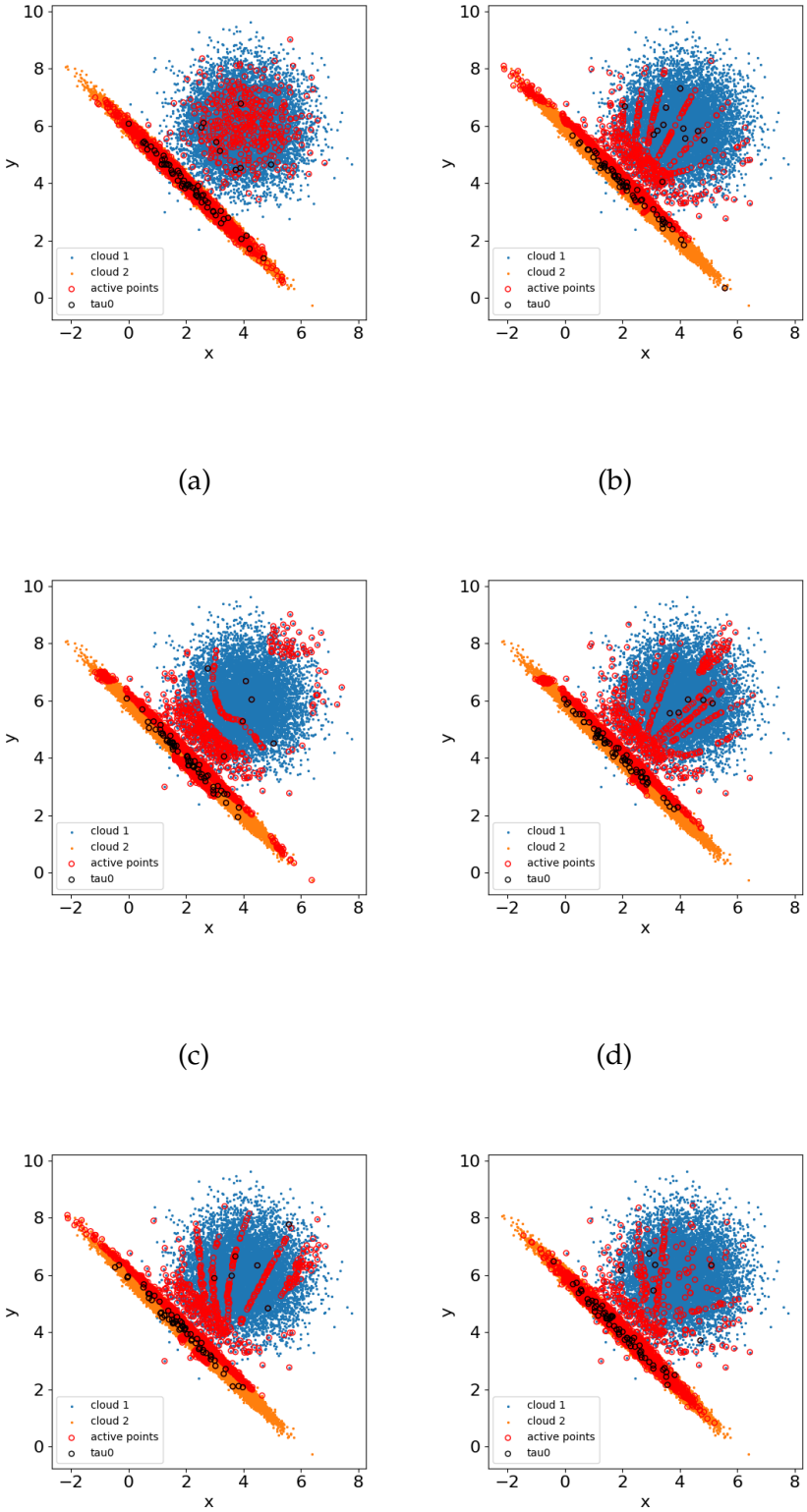


FIGURE A.10: The labelled points selected for active learning for imbalanced binary Gaussian clouds trained on a neural network. Designated: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

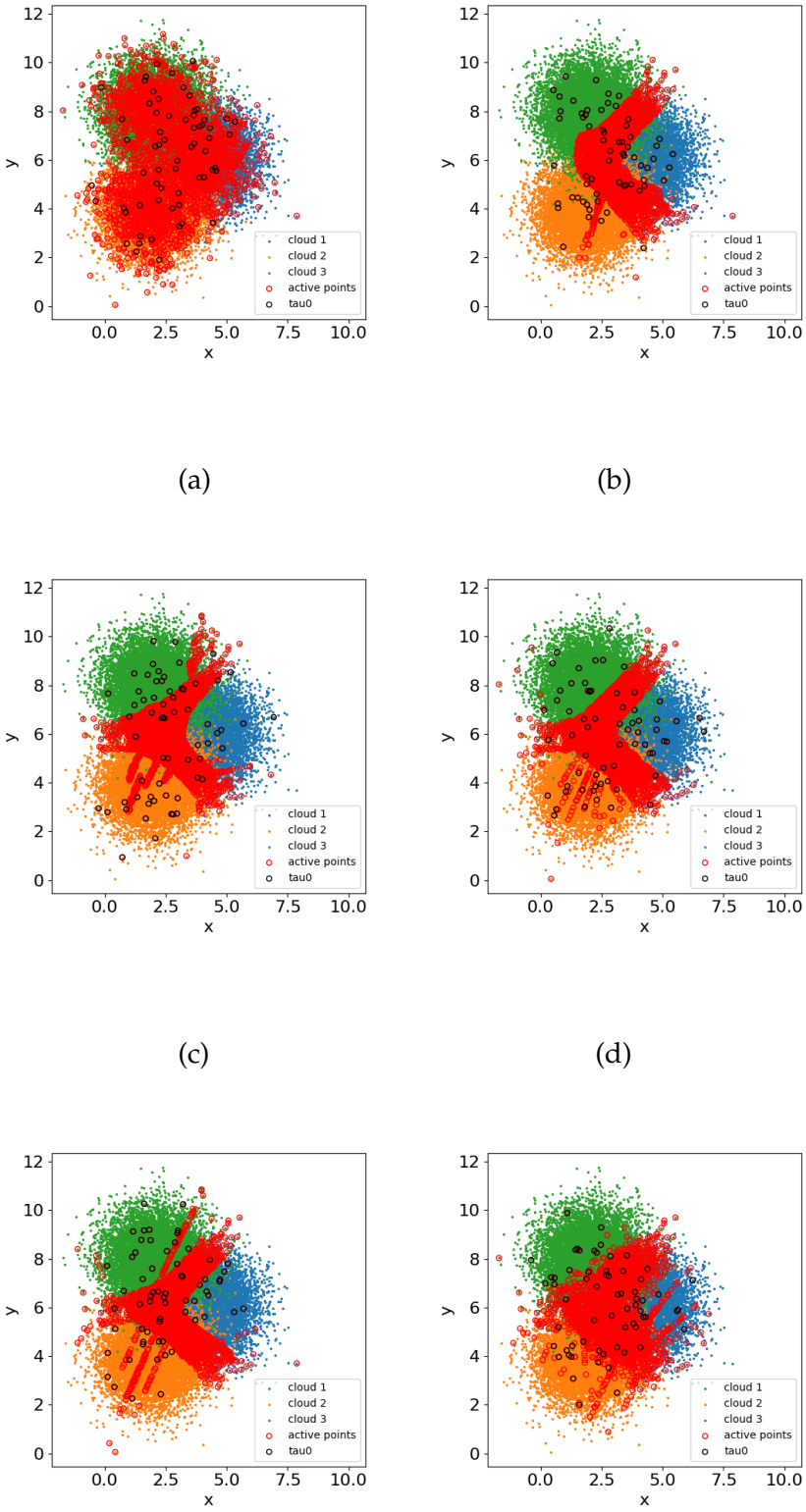
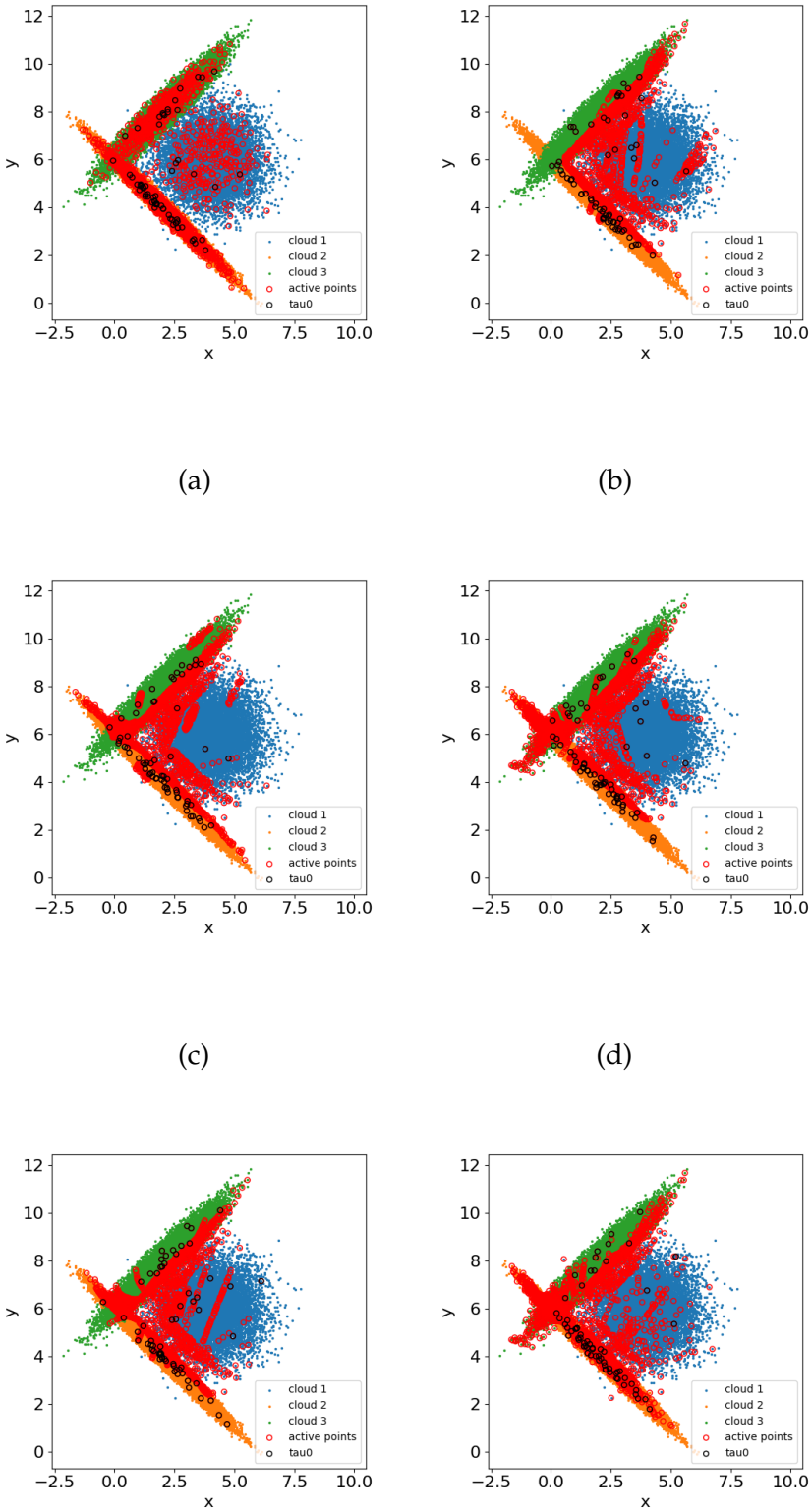


FIGURE A.11: The labelled points selected for active learning for balanced trinary Gaussian clouds trained on a neural network. Designated: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.



(a) (b)

(c) (d)

(e) (f)

FIGURE A.12: The labelled points selected for active learning for imbalanced trinary Gaussian clouds trained on a neural network. Designated: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

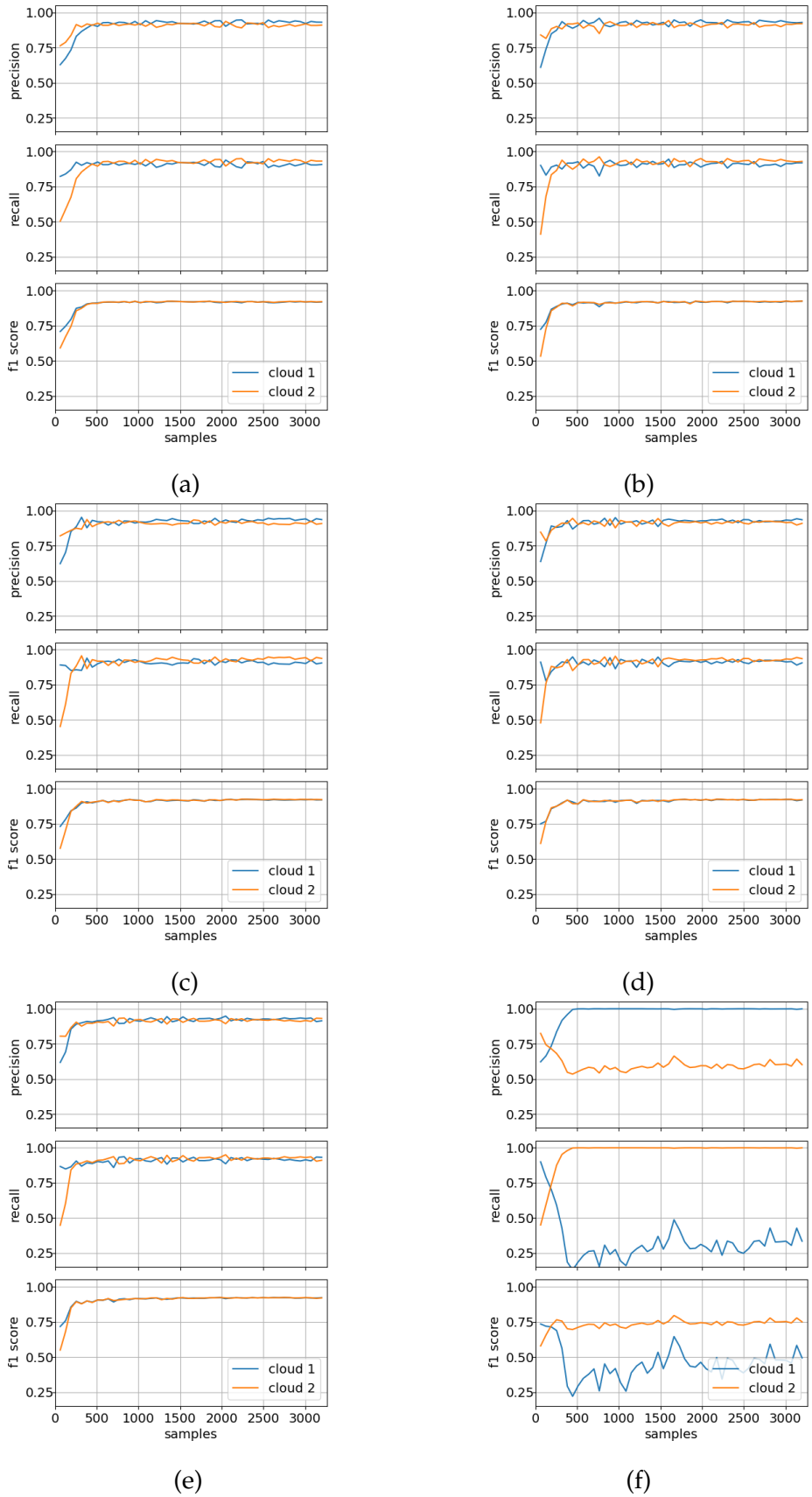


FIGURE A.13: Precision, recall, and f_1-score for balanced binary Gaussian clouds trained on a neural network. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

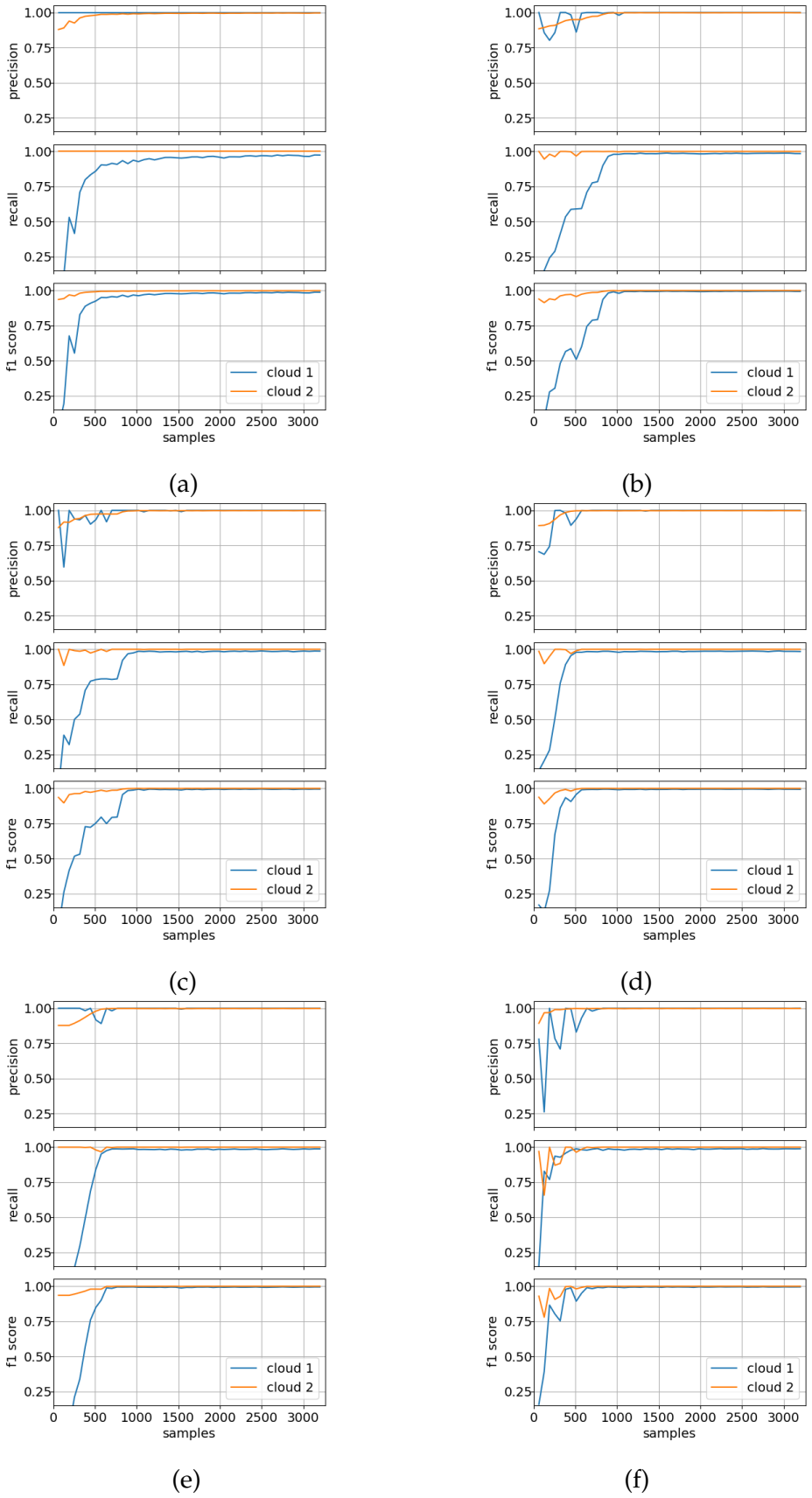


FIGURE A.14: Precision, recall, and f_1-score for imbalanced binary Gaussian clouds trained on a neural network. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

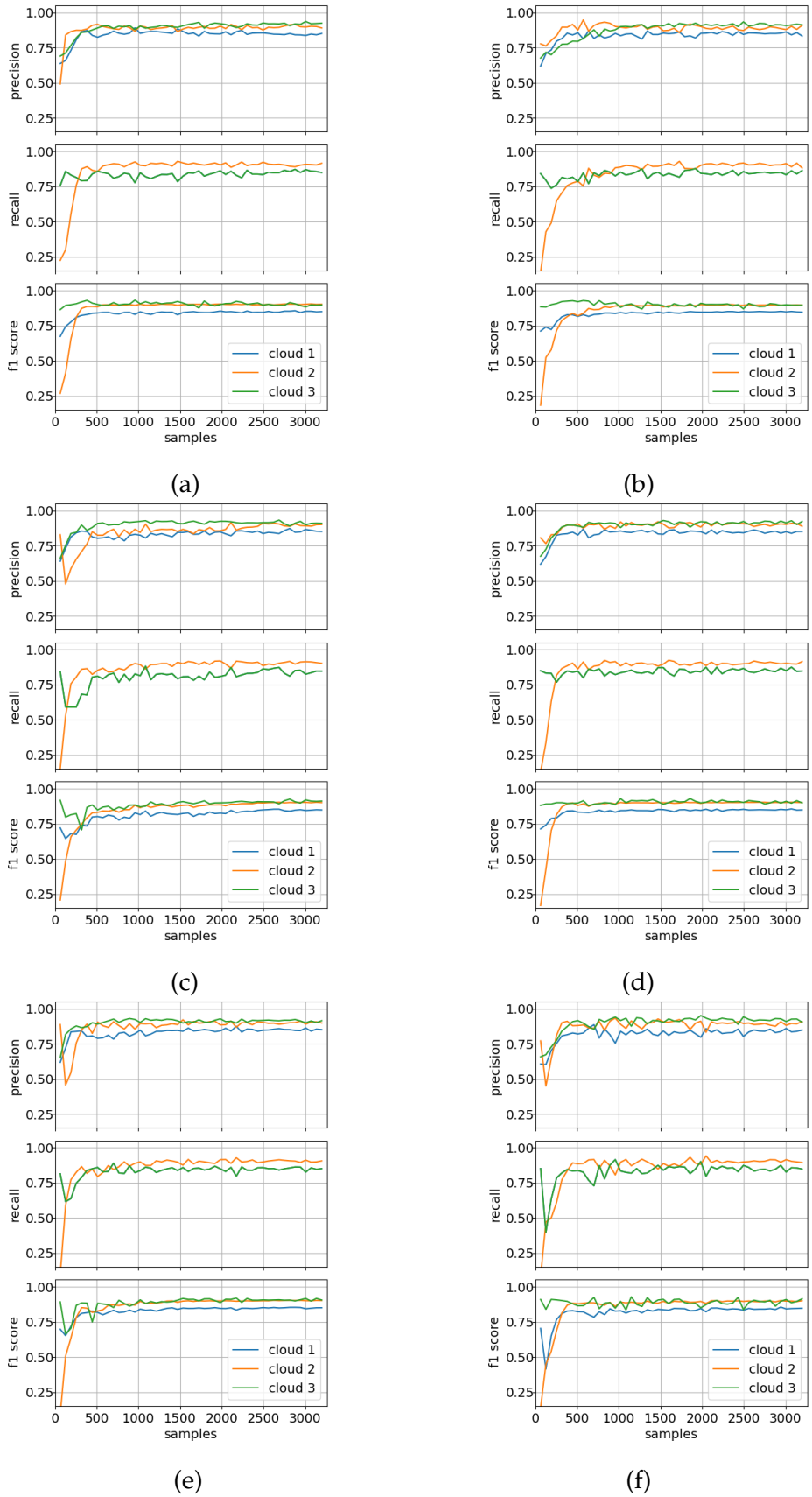


FIGURE A.15: Precision, recall, and f_1-score for balanced trinary Gaussian clouds trained on a neural network. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

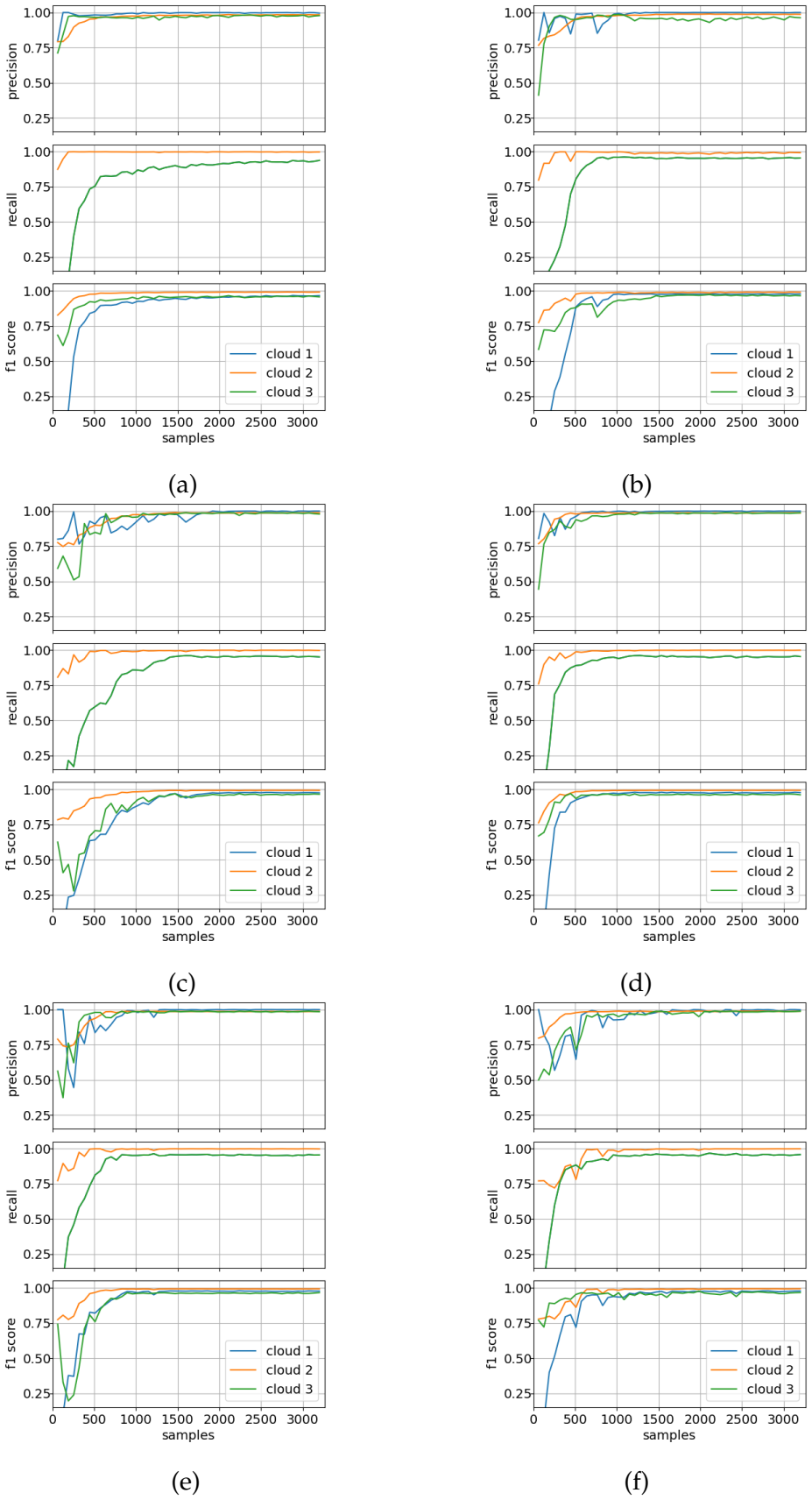


FIGURE A.16: Precision, recall, and f_1-score for imbalanced trinary Gaussian clouds trained on a neural network. Labelled: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

Appendix B

Appendix B

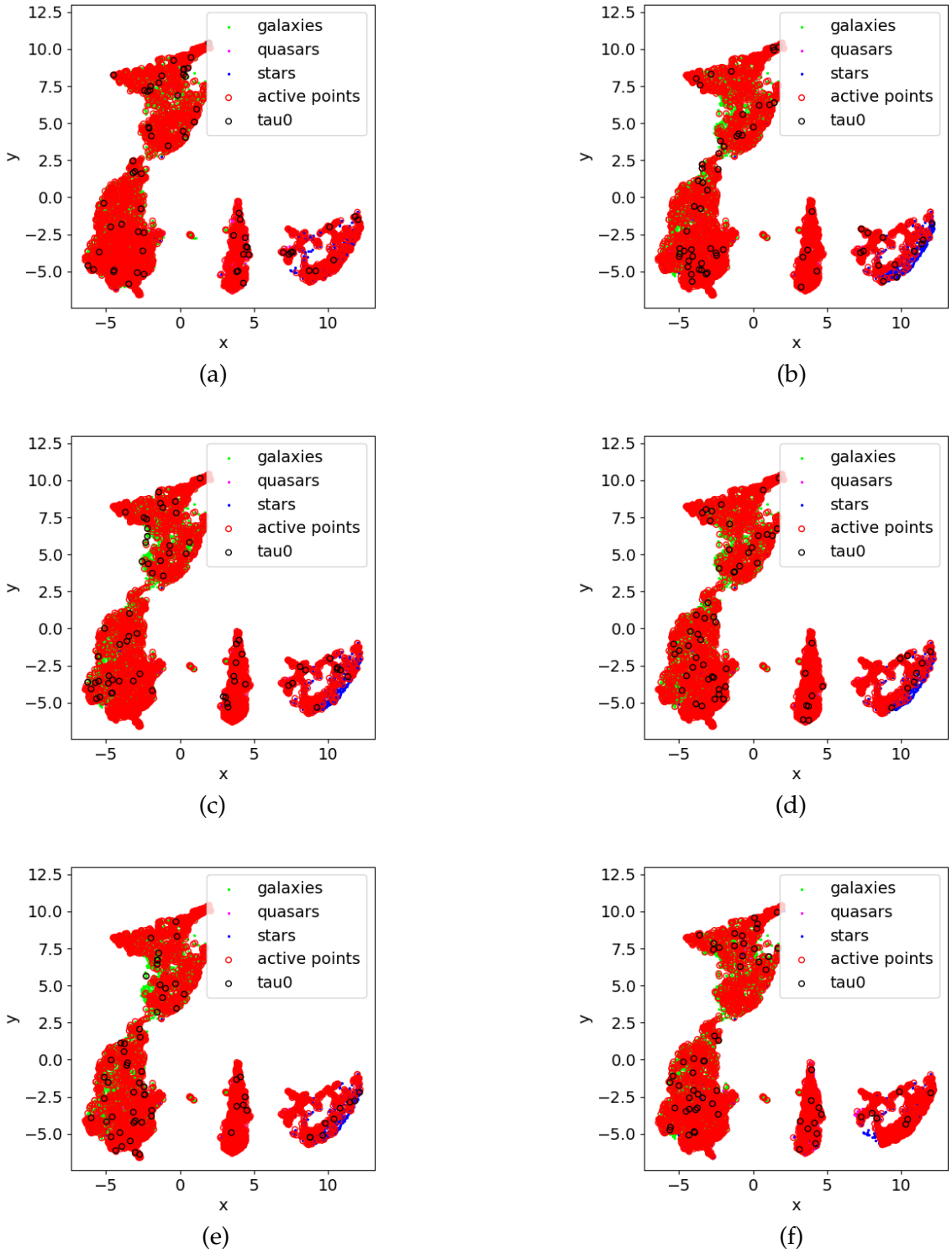
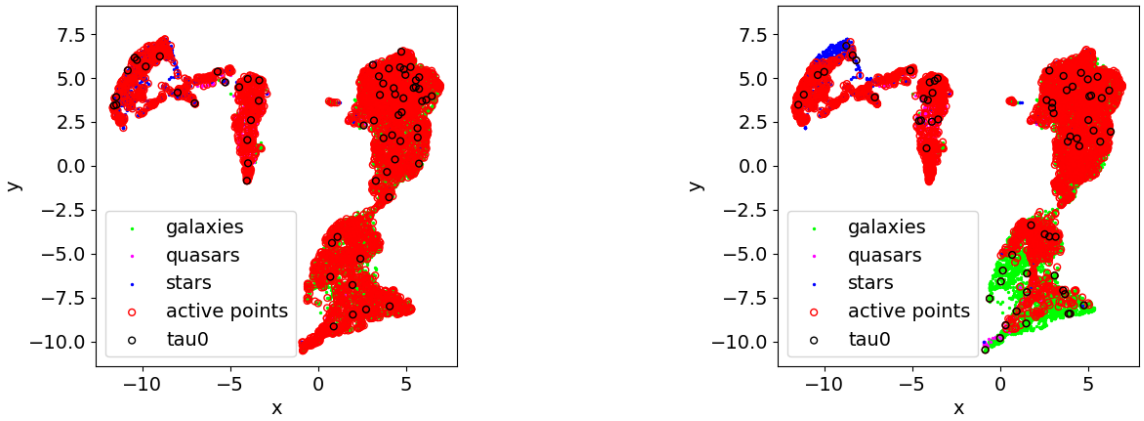
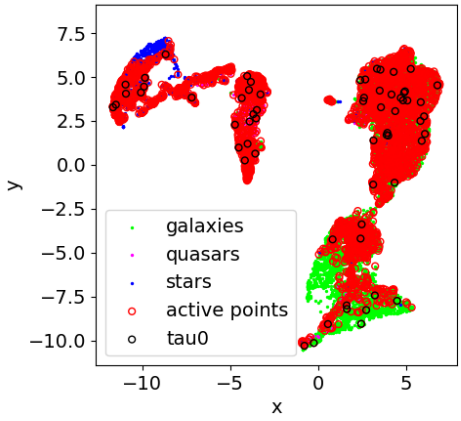


FIGURE B.1: The labelled points selected for active learning for the $10k$ dataset trained on a random forest. Designated: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

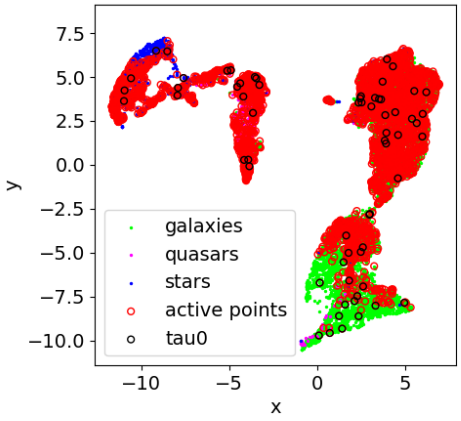


(a)

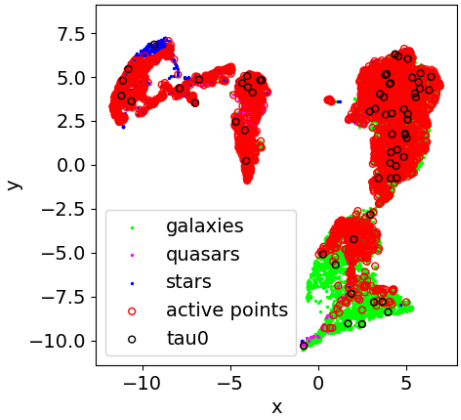
(b)



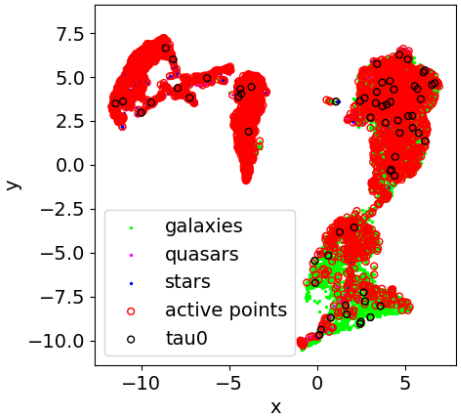
(c)



(d)



(e)



(f)

FIGURE B.2: The labelled points selected for active learning for the 10k dataset trained on a neural network. Designated: (a) for random sampling, (b) for uncertainty sampling, (c) for entropy measure, (d) for best-vs-second-best fit, (e) for variance reduction, and (f) for learning active learning.

Bibliography

Abazajian K., et al., 2004, Astronomical Journal, 128, 502

Abe S., 2009. pp 854–863, doi:10.1007/978-3-642-04274-4_88

Aguado D. S., et al., 2019, Astrophysical Journal, Supplement, 240, 23

Ahn C. P., et al., 2014, Astrophysical Journal, Supplement, 211, 17

Ahumada R., et al., 2020, Astrophysical Journal, Supplement, 249, 3

Aihara H., et al., 2011, Astrophysical Journal, Supplement, 193, 29

Allan V., Leedham C., 2021, arXiv e-prints, p. arXiv:2106.00365

Baker D., et al., 2021, Astrophysical Journal, 907, 16

Barchi P. H., et al., 2020, Astronomy and Computing, 30, 100334

Bazarghan M., Gupta R., 2008, Astrophysics and Space Science, 315, 201

Bermant P. C., Bronstein M. M., Wood R. J., Gero S., Gruber D. F., 2019, Scientific Reports, 9, 12588

Birrer S., et al., 2019, Monthly Notices of the RAS, 484, 4726

Bolton A. S., et al., 2012, Astronomical Journal, 144, 144

Botchkarev A., 2018, arXiv e-prints, p. arXiv:1809.03006

Breiman L., 1996, Machine Learning, 24, 123

Breiman L., 2001, Machine Learning, 45, 5

Bundy K., et al., 2015, Astrophysical Journal, 798, 7

Buta R. J., 2011, arXiv e-prints, p. arXiv:1102.0550

Chen L., Hassani H., Karbasi A., 2016, arXiv e-prints, p. arXiv:1603.03515

Clarke C. J., 1992, in Dusty Discs. p. 159

Clarke A. O., Scaife A. M. M., Greenhalgh R., Griguta V., 2020, Astronomy and Astrophysics, 639, A84

Cortes C., Vapnik V., 1995, Machine Learning, 20, 273

Cuceu A., Farr J., Lemos P., Font-Ribera A., 2019, Journal of Cosmology and Astroparticle Physics, 2019, 044

Cunningham P., Delany S. J., 2020, arXiv e-prints, p. arXiv:2004.04523

Cutri R. M., et al., 2012, Explanatory Supplement to the WISE All-Sky Data Release Products, Explanatory Supplement to the WISE All-Sky Data Release Products

Davis M., Huchra J., Latham D. W., Tonry J., 1982, Astrophysical Journal, 253, 423

Dawson K. S., et al., 2013, Astronomical Journal, 145, 10

Dawson K. S., et al., 2016, Astronomical Journal, 151, 44

De Leo M., Carrera R., Noël N. E. D., Read J. I., Erkal D., Gallart C., 2020, Monthly Notices of the RAS, 495, 98

Dimitriadis S., Liparas D., 2018, Neural Regeneration Research, 13, 962

ESO 2001, Heavy Metal Stars, European Southern Observatory Press Release

Feigelson E. D., Babu G. J., 1992, Astrophysical Journal, 397, 55

Freund Y., Schapire R. E., 1997, Journal of Computer and System Sciences, 55, 119

Frieman J. A., et al., 2008, Astronomical Journal, 135, 338

Granter S., Beck A., Papke D., 2017, Archives of Pathology & Laboratory Medicine, 141, 619

Greenstein J. L., 1963, Nature, 197, 1041

Grimshaw S. D., Efron B., Tibshirani R. J., 1995, Technometrics, 37, 341

Gunn J. E., et al., 1998, Astronomical Journal, 116, 3040

Gunn J. E., et al., 2006, Astronomical Journal, 131, 2332

Haenlein M., Kaplan A., 2019, California Management Review, 61, 000812561986492

Hastie T., Tibshirani R., Friedman J. H., 2009, The elements of statistical learning, 2 edn. Springer

Hazard C., Mackey M. B., Shimmins A. J., 1963, Nature, 197, 1037

Herschel W., 1785, Philosophical Transactions of the Royal Society of London Series I, 75, 213

Hewett P. C., Wild V., 2010, Monthly Notices of the RAS, 405, 2302

Hinton G. E., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R. R., 2012, arXiv e-prints, p. arXiv:1207.0580

Horvath I., 2020, arXiv e-prints, p. arXiv:2001.00159

Hossin M., M.N S., 2015, International Journal of Data Mining & Knowledge Management Process, 5, 01

Howe R., 2009, Living Reviews in Solar Physics, 6, 1

Hubble E. P., 1925, The Observatory, 48, 139

Hubble E., 1926, Contributions from the Mount Wilson Observatory / Carnegie Institution of Washington, 324, 1

Hutchinson T. A., et al., 2016, Astronomical Journal, 152, 205

Isobe T., Feigelson E. D., Akritas M. G., Babu G. J., 1990, Astrophysical Journal, 364, 104

James G., 2003, Machine Learning, 51, 115

Janocha K., Czarnecki W. M., 2017, arXiv e-prints, p. arXiv:1702.05659

Jones D. H., et al., 2004, Monthly Notices of the RAS, 355, 747

Joshi A., Porikli F., Papanikolopoulos N., 2009. pp 2372–2379, doi:10.1109/CVPR.2009.5206627

Jovanović P., Popović L. Č., 2009, arXiv e-prints, p. arXiv:0903.0978

Kepler S. O., Koester D., Pelisoli I., Romero A. D., Ourique G., 2021, Monthly Notices of the RAS, 507, 4646

Kollmeier J. A., et al., 2017, arXiv e-prints, p. arXiv:1711.03234

Konyushkova K., Sznitman R., Fua P., 2017, arXiv e-prints, p. arXiv:1703.03365

Krawczyk B., 2016, Progress in Artificial Intelligence, 5

Krizhevsky A., Sutskever I., Hinton G., 2012, Neural Information Processing Systems, 25

Kurcz A., Bilicki M., Solarz A., Krupa M., Pollo A., Małek K., 2016, Astronomy and Astrophysics, 592, A25

Labach A., Salehinejad H., Valaee S., 2019, arXiv e-prints, p. arXiv:1904.13310

Lewis D. D., Gale W. A., 1994, arXiv e-prints, pp cmp–lg/9407020

Lintott C. J., et al., 2008, Monthly Notices of the RAS, 389, 1179

Longadge R., Dongre S., 2013, arXiv e-prints, p. arXiv:1305.1707

Louppe G., 2014, arXiv e-prints, p. arXiv:1407.7502

Lundgren B., et al., 2015, Publications of the ASP, 127, 776

Manser C. J., et al., 2016, Monthly Notices of the RAS, 455, 4467

Marra V., Rosenfeld R., Sturani R., 2019, Universe, 5, 137

Martínez-Álvarez F., Troncoso A., Cortés G., Riquelme J., 2015, Energies, 8, 13162

Masters D., Luschi C., 2018, arXiv e-prints, p. arXiv:1804.07612

Matthews T. A., Sandage A. R., 1963, Astrophysical Journal, 138, 30

McCarthy J., Minsky M. L., Rochester N., Shannon C. E., 2006, AI Magazine, 27, 12

McCulloch W. S., Pitts W., 1943, The Bulletin of Mathematical Biophysics, 5, 115

Meisel Z., Deibel A., Keek L., Shternin P., Elfritz J., 2018, Journal of Physics G Nuclear Physics, 45, 093001

Michie D., 1963, The Computer Journal, 6, 232

Myhrvold N., 2018, Icarus, 314, 64

Newman J. A., et al., 2013, Astrophysical Journal, Supplement, 208, 5

Norris M. A., et al., 2016, Astrophysical Journal, 832, 198

Nwankpa C., Ijomah W., Gachagan A., Marshall S., 2018, arXiv e-prints, p. arXiv:1811.03378

Oke J. B., 1963, Nature, 197, 1040

Persic M., Salucci P., Stel F., 1996, Monthly Notices of the RAS, 283, 1102

Quinlan J. R., 1986, Machine Learning, 1, 81–106

Raichoor A., et al., 2017, Monthly Notices of the RAS, 471, 3955

Ramachandran P., Zoph B., Le Q. V., 2017, CoRR, abs/1710.05941

Richards G. T., et al., 2002, Astronomical Journal, 123, 2945

Rosasco L., Vito E. D., Caponnetto A., Piana M., Verri A., 2004, Neural Computation, 16, 1063

Rothman K. J., 2010, European Journal of Epidemiology, 25, 223

Rubin V. C., 1983, Science, 220, 1339

Rumelhart D. E., Hinton G. E., Williams R. J., 1986, Nature, 323, 533

SDSS 2004, A schematic plot of the H-R diagram. http://skyserver.sdss.org/dr1/en/astro/stars/stars.asp

Said K., Colless M., Magoulas C., Lucey J. R., Hudson M. J., 2020, Monthly Notices of the RAS, 497, 1275

Sánchez S. F., et al., 2019, Monthly Notices of the RAS, 482, 1557

Scheffer T., Decomain C., Wrobel S., 2001, in IDA.

Schmidt M., 1963, Nature, 197, 1040

Searle L., Zinn R., 1978, Astrophysical Journal, 225, 357

Settles B., 2009.

Shakura N. I., Sunyaev R. A., 1973, Astronomy and Astrophysics, 500, 33

Sheehan W., 2011, arXiv e-prints, p. arXiv:1112.0243

Shields G. A., 1999, Publications of the ASP, 111, 661

Shore S. N., 2009, Astronomy and Astrophysics, 500, 491

Smee S. A., et al., 2013, Astronomical Journal, 146, 32

Smith H. J., Hoffleit D., 1963, Nature, 198, 650

Song L., 2016, arXiv e-prints, p. arXiv:1607.03182

Soni Madhulatha T., 2012a, arXiv e-prints, p. arXiv:1205.1117

Soni Madhulatha T., 2012b, arXiv e-prints, p. arXiv:1205.1117

Srivastava N., et. al 2014, JMLR, 15

Steinmetz M., Navarro J. F., 2002, New Astronomy, 7, 155

Stoughton C., et al., 2002, Astronomical Journal, 123, 485

Tonry J., Davis M., 1979, Astronomical Journal, 84, 1511

Turing A. M., 1950, Mind, LIX, 433

Vanden Berk D. E., et al., 2001, Astronomical Journal, 122, 549

Varshney M., Singh P., 2021, Signal, Image and Video Processing, 15, 1

Visa S., Ramsay B., Ralescu A., Knaap E., 2011. pp 120–127

White S. D. M., Rees M. J., 1978, Monthly Notices of the RAS, 183, 341

Wilson J. C., et al., 2019, Publications of the ASP, 131, 055001

Woodward P. R., 1978, Annual Review of Astron and Astrophys, 16, 555

Wright T., 1750, An original theory or new hypothesis of the universe : founded upon general phaenomena of the visible creation; and particularly the Via the laws of nature, and solving by mathematical principles : the Lactea ...compris'd in nine familiar letters from the author to his friendand : illustrated with upward of thirty graven and mezzotinto plates ..., doi:10.3931/e-rara-28672.

Wright E. L., et al., 2010, Astronomical Journal, 140, 1868

Wu D., 2018, arXiv e-prints, p. arXiv:1805.04735

Wu J., Chen X.-Y., Zhang H., Xiong L.-D., Lei H., Deng S.-H., 2019, Journal of Electronic Science and Technology, 17, 26

Xue X. X., et al., 2008, Astrophysical Journal, 684, 1143

Yang F., hang Lu W., kai Luo L., Li T., 2012, Neurocomputing, 94, 54

Yang Y., Ma Z., Nie F., Chang X., Hauptmann A., 2015, International Journal of Computer Vision, 113

Yatawatta S., Avruch I. M., 2021, Monthly Notices of the RAS, 505, 2141

York D. G., et al., 2000, Astronomical Journal, 120, 1579

Zhang Y., Yang X., Guo H., 2021, Monthly Notices of the RAS, 507, 5320

Zhao H., Gallo O., Frosio I., Kautz J., 2015, arXiv e-prints, p. arXiv:1511.08861

de Vaucouleurs G., 1959, Handbuch der Physik, 53, 275