Towards Automated Spike Sorting Curation Without Ground Truth Data

Robyn Greene

MInf Project (Part 2) Report
Master of Informatics
School of Informatics
University of Edinburgh

2022

Abstract

Gaining insight into the complex neural mechanisms which underpin cognitive processes is vital to our understanding of the brain in health and disease. Activity in the brain can be detected by extracellular neural recording devices. These probes record the weak electrical signal between neurons as information in the brain is transferred. Exciting developments in neural recording devices have lead to modern extracellular probes which can capture precise activity of thousands of neurons in the brain simultaneously. The data produced by these recordings must be de-mixed using *spike sorting* algorithms in order to establish the relationship between recorded signals and neurons in the brain. These algorithms, *spike sorters*, identify *units* which are putative neurons detected in the recording. The units identified by spike sorters form the basis for research into the workings of the brain.

As spike sorting algorithms cannot perfectly identify neurons from recordings, curation is necessary in order to eliminate false positives before analysing neural activity. For simple recording devices manual curation is sometimes used. However, this process is subject to operator bias and is unmanageable for more modern devices which produce very large quantities of data. Despite the increasing demand for more automated curation methods, few exist [24, 8]. Of the potential available methods, there is little guidance on appropriate implementation in practice [13]. While quality metrics pose one potential solution there are no formal validated guidelines for their use in curation. *Agreement* or *consensus* is an exciting emerging proxy to ground truth [5]. However, this method is limited in the number of true positives identified and lacks formal specifications for implementation in neuroscience research.

By drawing on the relative strengths of quality metrics and unit agreement, this project investigates the novel possibility of a fully automated curation method. By examining one extracellular dataset from a recent neuroscience study, this project identifies and addresses several challenges to automated curation [9]. Several practical neuroscience use cases are considered in the development of this method. This project demonstrates that it is possible to achieve interpretable, automated curation using quality metrics. In doing so, this project lays the foundation for progress in enhancing the reliability of neuroscience research which makes use of extracellular neural recording devices.

Acknowledgements

Foremost, I would like to express my immense gratitude to my supervisor, Dr. Matthias Hennig, for his incredible encouragement, patience and dedication throughout this project. His infectious enthusiasm for the subject has been an inspiring guiding force to this work, and has made this project a truly enjoyable experience.

I would also like to thank Dr. Antonio Vergari and Dr. Diego Oyarzún for their suggestions.

Finally, I would like to thank Klara Fuchs and my partner, Maya Khela, for their continual support throughout this year.

Table of Contents

Chapter 1

Introduction

Neurons communicate via short firing events that last for around 1ms, known as *spikes*. These weak electrical signals can be detected by placing a microelectrode in the space between firing neurons. As one electrode may detect signals from several neurons, a process of “spike sorting” is required to determine the relationship between signals detected and actual neurons firing. In modern recordings, multiple electrodes are used to record simultaneously. Algorithms which attempt to automate the process of spike sorting are referred to as “spike sorters” or “spike sorting algorithms”. Given recorded signals, these algorithms attempt to identify “units” which are purported neurons active during the recording. Many such algorithms now exist (eg: MountainSort4 [2], Ironclust [20], Kilosort2 [2], WaveClus [6], and Klusta [29]). However, recent studies have demonstrated that different spike sorters draw different conclusions about the neurons present [5, 19].

Since neuroscience research relies on the accuracy of this spike sorting step, it is vital that true positive units and false positive units can be separated reliably. This process of validating spike sorted results involves “curation” - removal of false positives. Occasionally, “ground truth” data is available for validation of some neurons. However, collection of this data is expensive and difficult to perform at scale. The curation step may be completed manually (“manual curation”) for simpler recording devices. However, as recording technologies improve, the quantity of data produced by single recordings grows, leading to a demand for more automated methods [13]. One approach to identify poor quality units is to use quality metrics [16]. Many such metrics exist, and there is little consensus regarding their use in curation [24]. A promising alternative to use of metrics is to consider unit agreement; this involves running multiple spike sorting algorithms and considering units which are found by multiple sorters to be reliable [5].

This project builds on the findings of Buccino et al., [5] as well as the work conducted in Part 1 of this project, in order to interrogate existing curation methods and provide novel suggestions for curation standards in the future.

1.1 Previous work carried out

Part 1 of this project focused on examining common quality metrics alongside agreement between sorters. Five common sorting algorithms were examined: MountainSort4 [8], Ironclust [20], Waveclus [6], Kilosort2 [2] and Klusta [29].

In Part 1, the following contributions were made:

- Using the SpikeInterface framework, an existing experimental pipeline was adapted to incorporate multiple sorting algorithms and a further pipeline was developed to calculate and store quality metrics and unit agreement values [3, 9].
- Results found by Buccino et al. regarding the general behaviour of spike sorters and the consensus method were corroborated [5].
- MountainSort4, Ironclust and Kilosort2 were identified as the strongest choices for spike sorting the dataset used.
- Quality metrics were investigated to establish relationships between matched and unmatched units, using a range of statistical tests.
- A baseline classifier model which predicted agreement of a unit based on ISI violations and L-ratio was produced.

1.2 Main achievements

Part 2 of this project uses the results found in Part 1 to build on the model used for curation of units.

The main contributions made this year were:

- Provide a qualitative analysis of common quality metrics to identify similarities and differences.
- Based on the analysis performed in the first part of the project, investigate the three best sorters with regards to matched units.
- Investigate the relationship between sorter used and quality metric distribution.
- Investigate the importance of different metrics to predicting unit quality.
- Investigate mechanisms for pre-processing quality metrics to improve classifier performance.
- Using the Gaussian Bayes model (developed in the previous part of the project) as a baseline, develop an automated unit curation model with improved performance.
- Consider the implication of different assumptions and uses of the model on its design, and suggest future work in this domain.

Chapter 2

Background

2.1 Extracellular neural recordings

Neurons mainly communicate with each other via “spikes” - all or nothing firing events which each last around 1ms [13]. By recording the resultant changes in extracellular potential, microelectrode devices are designed to detect these firing events when placed nearby to firing cells.

Extracellular neural recordings are used widely in neuroscience due to their uniquely high spatiotemporal resolution [19]. Additionally, this technique can be used to record from awake animals (“in vivo”) without interfering with neural function. Therefore this method can be used to gain insight into the workings of intact neural circuits in awake animals [14]. These advantages, when combined, provide a powerful basis for understanding the brain mechanisms underlying animal behaviour. Such recordings are used across many different levels of analysis; from identifying behaviours of a single neuron, to considering the interactions between populations of neurons.

This kind of recording can be performed using a single electrode. However, these devices record the activity of very few neurons and do not have the capacity to record from multiple locations in the brain. Subsequently, recordings of this kind are unable to support claims about the interaction between networks of neurons [11, 21]. Because neurons work as a system, accurate simultaneous recordings of many neurons are necessary in order to examine the neuron interactions which underpin cognitive processes [11, 35, 16, 13].

More recent recording devices have been developed with the aim of accurately recording larger populations of neurons simultaneously. In particular, the development of the tetrode (a four-electrode recording device) satiated the need to record from many neurons, detected from multiple locations at once. This device remains popular and has undergone significant validation, often by use of “paired recordings” (procedures in which intracellular and extracellular recordings are taken simultaneously) [11, 35, 15]. Overall, the accuracy and explanatory power of tetrode recordings motivates the need for methods which can reliably process the data produced.

Further developments in recording technology have since emerged and continue to develop at a greater pace than the associated analysis techniques can keep up with. Of particular interest is the high density micro-electrode array (HDMEA), which is capable of recording approximately 1000 sites at once (one such probe is the NeuroPixel probe) [20]. Because HDMEAs provide a substantial amount of data it is essential that automated methods are capable of processing this data reliably and at scale. Without trusted methods for extracting information from these recordings, it is impossible to use these devices to their full potential.

2.2 Spike sorting

In electrode recording devices, one recording site ("channel") may detect signals from multiple neurons. Additionally, signals from one neuron may be picked up by many channels. Therefore, "spike sorting" is required to de-mix signals in order to infer the activity of neurons which are firing [13].

One way to conceptualise the problem of spike sorting is the analogy of a cocktail party: given the conversations occurring at the party, the task is to isolate each of the speakers [14]. The analogy can be extended for tetrode data, with several microphones detecting the sound at a much larger party. Similarly, for HDMEAs, the analogy of many microphones situated around a football stadium of speakers is appropriate.

In computational terms, spike sorting amounts to a clustering problem. Given a set of signals from a set of electrodes, a spike sorting algorithm must group signals in order to infer the behaviour of neurons being recorded. Different spike sorting algorithms utilise different machine learning techniques in order to infer which signals are related to each other and to further infer which neuron is producing each set of signals ("spike train"). Spike sorting algorithms output a set of "units". This terminology is common in the literature (cf: [28, 16]) and is retained throughout this report to differentiate between the underlying neuron being recorded ("neuron") and the model of that neuron ("unit") produced by a spike sorter.

Many different spike sorting algorithms have been developed, yielding a wide variety of approaches to the task of identifying units from recorded signals. In general, the process involves first extracting spike waveforms from recordings, then clustering waveforms into units [16]. This process can be further broken down into the following steps:

- **Filtering**: High frequency signals are usually attributed to noise, whilst very low frequency signals are attributed to local field potentials (LFPs) [13]. To eliminate both, a bandpass filter is often applied, usually excluding signals outwith the range 300Hz-30000Hz [28]. Remaining signals are assumed to represent neuron firing activity.
- **Spike detection**: An amplitude threshold is set (often dynamically in relation to signal-to-noise ratio), and signals which exceed this threshold are considered to be spikes [28]. Choice of amplitude threshold is a trade off between the number of false positives produced and the number of false negatives (legitimate firing activity which is erroneously discarded at this stage) [16]. Spike sorters approach

this step differently, which is one source of variation in sorter outputs. The result of this step is a set of "spike trains" - temporal series which indicate the points in time at which spiking is thought to occur.

- **Feature extraction and clustering**: Spike trains are analysed to group similar spike trains together and then assign each cluster of signals to a single unit. Approaches to this step vary substantially between individual sorting algorithms. However, there are two broad methods which are in common use: density based and template matching approaches. In *density-based* approaches, dimensionality reduction is used alongside clustering techniques to first estimate the source of the signal and then cluster the signals together. By contrast, *template matching* approaches rely on the fact that neuron firing events have a "signature" - certain features (for example, amplitude and shape) specific to the neuron that produces them. Template spike shapes are either extracted or learned based on the spatio-temporal footprints of individual events [28, 14].

To put spike sorting into a neuroscience context, an experiment which makes use of extracellular recordings will produce data consisting of signals recorded at each site. A spike sorting algorithm can then be used to produce a set of units each of which is associated with some firing pattern. Further analysis can then be conducted, treating the units as real neurons to draw conclusions about the brain. Because sorting algorithms approach the task of spike sorting differently, use of a different algorithm can lead to a different set of units being used in analysis. Since the units produced by spike sorters are treated as neurons for the purposes of neuroscience research, the measure of success for spike sorting algorithms is whether each unit identified by a sorter actually corresponds to a neuron firing in the brain.

Based on the sorter comparison presented in the previous part of this project, three sorters were selected for analysis this year: MountainSort4 (density-based) [8], Kilosort2 (template matching) [2] and Ironclust (density-based) [20]. MountainSort4 was identified as being particularly useful in finding many true positives with a low ratio of false positives (11.27% of 71 identified units were matched). Similarly, Ironclust identified a large number of true positives relative to the total number of units identified (11.36% of 44 identified units). Kilosort2 identified many true positives, but had the highest ratio of false positives across all sorters tested in the first part of this project (2.63% of 266 identified units were matched). Kilosort2 was included in this analysis because it is commonly used, and because its high false positive rate motivates development of effective curation methods.

2.3 Validation of spike sorter results

Spike sorting as a process is imperfect, and sorting algorithms are known to generate errors about which neurons are present in a recording [13]. Given this, spike sorter outputs need to be curated by validating units identified and eliminating poor quality units. Specifically, units which do not reflect actual neuron activity should be removed from further analysis. Curating spike sorter results properly is vital. Studies rely on the data from extracellular recordings in order to make claims about the brain. Failure to remove poor quality units from such data could lead to erroneous conclusions being drawn [13].

Validation and curation can be considered with regards to two stages: Firstly, the validation of the spike sorting algorithm itself and secondly, the curation of results for specific experiments using extracellular neural recordings.

Spike sorters themselves can be validated using ground truth data, which takes the form of "paired recordings", and serves as the gold standard for spike sorter validation. Paired recordings involve taking both intracellular and extracellular recordings simultaneously. These recordings can then be compared to validate any firing behaviour detected [35]. Ground truth data is rarely available as it is expensive, difficult to collect successfully, and very limited in the number of neurons which can be confirmed [28]. Other methods for sorter validation include use of simulated or hybrid datasets [13]. Validation of spike sorting algorithms is not the focus of this project, however extensive discussion of this topic is available in the literature (see Harris et al. [11] and Wehr et al. [35]).

By contrast, curation of spike sorted results for a single experiment is a form of quality control. This curation involves eliminating false positives from spike sorted data, and should be performed prior to any neuroscience analysis. There are three common approaches to curation, namely: manual curation, metrics and the consensus method.

Manual curation relies on experts evaluating units to determine whether they reflect real neurons. Older spike sorting algorithms often assume manual curation will be performed. Such algorithms often intentionally allow for a higher false positive rate [2] By contrast, "fully automated" spike sorting algorithms attempt to eliminate the requirement for further manual curation [29, 13]. Fully automated spike sorters are strongly preferred for use with more modern recordings, where the quantity of data produced makes manual curation very difficult [13]. Additionally, manual curation poses a threat to reproducibility, since this kind of curation is subject to inherent operator bias [5, 36, 29, 13].

Quality metrics provide some attempt to formalise the properties that experts may look for when performing manual curation. Specifically, metrics are intended to give some quantification of the likelihood that a given unit corresponds to a real neuron in the brain. Many metrics have been developed and are often used as part of the curation process in neuroscience research. Nonetheless, there is little consensus on their use for curation, meaning there is no standard method for curation based on these metrics [1] [16, 8, 5, 13, 24].

Many neuroscience studies apply some quality metrics in order to exclude data, however the specific choice of metrics and thresholds for inclusion are rarely justified in these papers (eg: [9]) [24]. In fact, new metrics are often designed by those creating the spike sorting algorithms. For example NN-hit rate and NN-miss rate are metrics

[1] In Harris et al.'s review of quality control in neuron population recordings, an informal method is proposed for establishing an isolation quality threshold [13]. Their recommendation is essentially to consider the relationship between the quantity being measured and isolation quality.

which were presented by Chung et al. in the same paper as the MountainSort4 algorithm [8]. This observation raises the question of whether metrics can be used as part of an automated method for standardised curation pipelines. Demand for this work is alluded to by Chung et al., and others (for example Luo et al. in their 2018 review of neural analysis techniques [24]) [8]. This question forms the foundation for the work conducted for this thesis.

The consensus method is a relatively recent approach to curation. This method, outlined by Buccino et al. (2019), uses both ground truth data and manually curated results to show that units which were found by multiple sorters (*agreement* or *matched units*) were generally true positives, and that units which were found by only one sorter (*unmatched units*) were likely false positives [5]. This result is significant, as it demonstrates that unit agreement is a promising proxy to ground truth.

Theoretically, the consensus method alone provides a complete method for curation of spike sorted results. In practice however, many labs use only one sorting algorithm for their experiments [5, 19]. It is both computationally expensive and time consuming to run multiple spike sorting algorithms on a large number of recordings. A preferable alternative would be curation without the need to run multiple sorting algorithms on many recordings. Additionally, without clear standards on which sorting algorithms should be used for agreement-based curation, it is difficult for researchers to identify a specific method by which to curate results using this approach. Furthermore, the consensus method does not give an interpretable quantification of the confidence we may have in each unit's validity. That is, under the assumption that there are true positive units which are not detected by the agreement method, there is currently no way to identify which of these units are likely to be true positives. To put this in simpler terms, curation methods which use agreement as a proxy to ground truth should also provide some indication of confidence in units which are not detected by multiple sorters.

Streamlined curation This project investigates the possibility of a streamlined unit curation method which relies on quality metrics as an indicator of unit validity and unit agreement as a proxy to ground truth. Such a method should not rely on extensive use of multiple sorting algorithms and should be able to provide some indication of confidence in the validity of a unit. Importantly, this curation method should be usable within neuroscience research. To achieve this, the model should be interpretable so that researchers can clearly and reliably justify why a unit should be included or excluded from analysis. Further, the model should not require computational resources far exceeding those expected in current research (use of one sorting algorithm and calculation of a handful of metrics). The aim of this is to develop a reliable approach for automated curation of spike sorted results which can be used in neuroscience research.

2.4 Metrics

This section outlines the main qualitative differences between quality metrics used in curation. Individual metrics are then considered, highlighting the approach, expectation and intended use of each.

Different metrics are used to identify different types of problems with a spike sorted unit. These errors fall into three categories:

- **Type I (contamination):** Contamination occurs when spikes from multiple neurons have been erroneously assigned to one unit [17].
- **Type II (incompleteness):** Incompleteness occurs when spikes are erroneously excluded from a unit's activity. This can occur if a spike is poorly detected during recording, or if spiking behaviour from one neuron is mistakenly split into two units during clustering [17].
- **Drift:** Drift metrics identify changes in waveforms which occur when sorters fail to successfully track neurons in the case of electrode drift. Metrics relating to drift are not addressed here as drift is primarily a problem for dense recording devices such as HDMEAs.

The rest of this section details each of the quality metrics addressed in this analysis.

2.4.1 Firing rate

Calculation. Firing rate is simply the average number of spikes within the recording per second. Given N_s spikes in the unit across a full recording with a duration of T_r seconds, firing rate is:

$$\text{firing rate} = \frac{N_s}{T_r} \tag{2.1}$$

Expectation. Both very high and very low values of firing rate can indicate errors.

Use. Highly contaminated units (type I error) may have high firing rates as a result of inclusion of other neurons' spikes. Low firing rate units are likely to be incomplete (type II error), although this is not always the case (some neurons have highly selective firing patterns) [1].

2.4.2 Presence ratio

Calculation. The duration of the recording is split into bins [2]. Presence ratio is then the proportion of bins in which at least one spike occurred. Let B_s denote the number of bins in which spikes occurred, and B_n denote the number of bins in which no spikes occurred. Presence ratio is then:

$$\text{presence ratio} = \frac{B_s}{B_s + B_n} \tag{2.2}$$

[2]By default, 101 equally sized bins are used in SpikeInterface, which was here used to calculate metrics [3]

Expectation. Complete units are expected to have a high presence ratio (close to 1).

Use. A low presence ratio (close to 0) can be used to identify incomplete units (type II error). As with firing rate, a low presence ratio could also simply indicate a highly selective firing pattern.

2.4.3 Amplitude cutoff

Calculation. A histogram of spike amplitudes is created and deviations from the expected distribution are identified. This yields an estimate of the number of spikes missing from the unit (false negative rate) [16].

Expectation. Complete units are expected to have a low amplitude cutoff (close to 0).

Use. A high amplitude cutoff can be used as an indication of incompleteness (type II error).

2.4.4 Inter-spike-interval (ISI) violations

Calculation. Neurons have a refractory period after a spiking event during which they cannot fire again. Inter-spike-interval (ISI) violations refers to the rate of refractory period violations [16].

The threshold for ISI violations is the biological ISI threshold, ISI_t, minus the minimum ISI threshold, ISI_{min} enforced by the data recording system used. The array of inter-spike-intervals observed in the unit's spike train, ISI_s, is used to identify the count (#) of observed ISI's below this threshold. For a recording with a duration of T_r seconds, and a unit with N_s spikes, the rate of ISI violations is:

$$\text{ISI violations} = \frac{\#(ISI_s < ISI_t)T_r}{2N_s^2(ISI_t - ISI_{min})} \tag{2.3}$$

Expectation. Units with low contamination should have low ISI violations values (close to 0).

Use. Highly contaminated units (type I error) are indicated by high ISI violations values (close to 1).

2.4.5 Signal-to-noise ratio (SNR)

Calculation. The mean spike waveform for the unit is determined, and the maximum amplitude of this waveform, $A_{\mu s}$, is found. The noise is here σ_b, which is the standard deviation of the background noise on one channel. SNR is then:

$$\text{SNR} = \frac{A_{\mu s}}{\sigma_b} \tag{2.4}$$

Expectation. A high SNR unit has a signal which is greater in amplitude than the background noise and is likely to correspond to a neuron [17, 23].

Use. A low SNR value (close to 0) suggests that the unit is highly contaminated by noise (type I error).

2.4.6 Silhouette score

Calculation. Silhouette score was introduced by Rousseeuw, and gives the ratio between the cohesiveness of a cluster and its separation from other clusters [30]. Values for silhouette score range from -1 to 1.

Expectation. A good clustering with well separated and compact clusters will have a silhouette score close to 1.

Use. A low silhouette score (close to -1) indicates a poorly isolated cluster (both type I and type II error) [30].

2.4.7 Isolation distance

Calculation. The unit of interest is denoted as cluster C. The number of spikes within cluster C is denoted N_s and the number of spikes outwith the cluster is denoted N_n. The minimum of these two values is $N_{min} = \min(N_s, N_n)$. Each spike within C is represented by a vector of principal components (PCs). The mean vector for spikes in C is μ_C, and the covariance matrix of spikes in cluster C is Σ_C. For every spike i (represented by vector x_i) outwith cluster C, the Mahalanobis distance between μ_c and x_i is calculated as [17]:

$$D^2_{i,C} = (x_i - \mu_C)^T \Sigma_C^{-1} (x_i - \mu_C) \tag{2.5}$$

These distances are ordered from smallest to largest. The N_{min}'th entry in this list is the isolation distance.

Geometrically, the isolation distance for unit C is the radius of the circle which contains N_{min} spikes from unit C and N_{min} spikes from outwith unit C. Isolation distance can be interpreted as a measure of distance from the unit to the nearest cluster [12].

Expectation. Well separated clusters will have a high isolation distance.

Use. Jackson et al. noted that this measure is correlated with both type I and type II errors, but is particularly good at picking up type II errors [17].

2.4.8 L-ratio

Calculation. L-ratio uses 4 principal components (PCs) for each tetrode channel (the first being energy, the square root of the sum of squares of each sample in the waveform, followed by the first 3 PCs of the energy normalised waveform) [31]. This yields spikes which are each represented as a point in 16 dimensional space.

Define, for each cluster C, $D^2_{i,C}$, the squared Mahalanobis distance from the centre of cluster C for every spike i in the dataset (similarly to the calculation for isolation distance above). Assume that spikes in the cluster distribute normally in each dimension, so that D^2 for spikes in a cluster will distribute as χ^2 with 16 degrees of freedom. This yields $\mathrm{CDF}_{\chi^2_{\mathrm{df}}}$, the cumulative distribution function of the χ^2 distribution.

Define for each cluster C, the value $L(C)$, representing the amount of contamination of the cluster C:

$$L(C) = \sum_{i \notin \mathrm{C}} 1 - \mathrm{CDF}_{\chi^2_{\mathrm{df}}}(D^2_{i,C}) \tag{2.6}$$

L is then the sum of probabilities that each spike which is not a cluster member of C should be included in the cluster. Therefore the inverse of this cumulative distribution yields the probability of cluster membership for each spike i.

L is then normalised by the number of spikes N_s in C to allow larger clusters to tolerate more contamination [31]. This yields L-ratio, which can be expressed as:

$$\mathrm{L-ratio}(C) = \frac{L(C)}{N_s} \tag{2.7}$$

Expectation. A well separated unit should have a low L-ratio [31].

Use. Since this metric identifies unit separation, a high value indicates a highly contaminated unit (type I error) [31]. Jackson et al. suggests that this measure is also correlated with type II errors (although more strongly with type I errors) [17].

2.4.9 D-prime

Calculation. Introduced by Hill et al., D-prime uses linear discriminant analysis (LDA) to estimate the classification accuracy between two units [16, 5].

$P(v|C)$ probability distributions are assumed to be Gaussian (where v is a waveform and C is a class). LDA is then fit to the spikes in cluster C, as well as to the spikes which are not in cluster C (the set of spikes which are not in C is here denoted D). The mean and standard deviation of the LDA for cluster C is denoted $\mu_C^{(LDA)}$ and $\sigma_C^{(LDA)}$. Similarly, the mean and standard deviation of the LDA for D is denoted $\mu_D^{(LDA)}$ and $\sigma_D^{(LDA)}$. D-prime is then calculated as follows:

$$D_{\text{prime}}(C) = \frac{(\mu_C^{(LDA)} - \mu_D^{(LDA)})}{\sqrt{0.5((\sigma_C^{(LDA)})^2 + (\sigma_D^{(LDA)})^2)}} \tag{2.8}$$

Expectation. The magnitude of D-prime will be higher in well separated clusters.

Use. As a measure of cluster separation, D-prime identifies both type I and type II errors [16].

2.4.10 NN-hit rate and NN-miss rate

Calculation. Introduced by Chung et al., these non-parametric measures use nearest neighbours (NN) to estimate hit rate and miss rate [8].

The membership function, ρ is defined such that for any spike g_i in some cluster G, $\rho(g_i) = G$. Additionally, the nearest neighbour function $n_k(g_i)$ is defined such that the output of the function is the set of k spikes which are closest to g_i.

For a unit of interest with cluster C, a subset of spikes are randomly drawn to form the cluster A. A subset of spikes which are not in C are drawn to form the cluster B. The NN-hit rate for C is then:

$$NN_{\text{hit}}(C) = \frac{1}{k}\sum_{i=1}^{k} \frac{|\{x \in A : \rho(n_i(x)) = A\}|}{|A|} \tag{2.9}$$

Similarly, the NN-miss rate for C is:

$$NN_{\text{miss}}(C) = \frac{1}{k}\sum_{i=1}^{k} \frac{|\{x \in B : \rho(n_i(x)) = A\}|}{|B|} \tag{2.10}$$

Expectation. A unit with low contamination should have a high NN-hit rate. A more complete unit should have a low NN-miss rate.

Use. NN-hit rate is intended to identify unit contamination (type I errors). NN-miss rate is intended to identify unit incompleteness (type II errors).

2.5 Qualitative differences in metrics

Metrics which aim to identify the same type of error are expected to behave similarly. This would be reflected in the correlation between these metrics for the same dataset. The type of error that each quality metric aims to identify is summarised in Table 2.1.

Type I (contamination)	Mixed/ambiguous	Type II (incompleteness)
ISI violations	L-ratio	Presence ratio
SNR	Isolation distance	Amplitude cutoff
NN-hit rate	Silhouette score	NN-miss rate
	D-prime	
	Firing rate	

Table 2.1: Table summarising the error types addressed by different quality metrics.

In addition to categorising by error type, one may consider whether the metric determines cluster isolation. Silhouette score, isolation distance, L-ratio, D-prime, NN-hit rate and NN-miss rate are all concerned with determining cluster isolation and cohesiveness. Whilst these metrics differ in their exact approach to identifying unit isolation, they share an approach to conceptualising unit validity [8]. Another qualitative difference between metrics is whether they consider specific biological factors. ISI violations is unique in that it concerns the biological plausibility of the unit identified based on knowledge from neuroscience about the way neurons behave.

In summary, a wide variety of metrics have been developed over a long period of time, with the earliest metrics emerging around 1984 [23]. Whilst there is some discussion in the literature comparing these metrics (eg: [17]) there is a distinct lack of up to date, unifying accounts of the relationship between metrics and the validity of units [13].

2.6 Agreement

Unit agreement was expressed as a binary variable with "1" indicating that the unit was matched by at least one other sorter, and "0" indicating that the unit was not identified by any other sorters.

SpikeInterface was used to identify matches according to the following definition [5].

For two unit spike trains with n_1 and n_2 spikes respectively, the number of spikes which occur within 0.4ms of each other, n_m, is identified. The agreement score s between the spike trains is then:

$$s = \frac{n_m}{n_1 + n_2 - n_m} \tag{2.11}$$

Scores above 0.5 are considered "agreement units" or "matched units".

Whilst further validation work could refine the thresholds used in the above definition, this project follows the definition provided by Buccino et al., as this was validated as a good proxy to ground truth [5]. Therefore deviations from this definition may damage the reliability of the analysis presented here. Additionally, for the purpose of this project, the binary variable is sufficient to provide a proof of concept model for a curation method of the form described here. That is, this project aims to demonstrate that a curation method using metrics and agreement is possible, noting that refining such a model requires a substantial amount of progress in the study of the agreement method.

Chapter 3

Methods

3.1 Data

As per last year's report, the data used in this analysis is drawn from the experiments performed by Gerlei et al. for their 2020 study, "Grid cells are modulated by local head direction" [9]. In these experiments, tetrode devices were used to record from the medial entorhinal cortices (MEC) of mice as they roamed freely around a $1m^2$ enclosure. Gerlei et al.'s original analysis pipeline was adapted using SpikeInterface to allow for the use of multiple sorting algorithms as well as calculation of quality metrics and unit agreement [3, 5].

In this project, the adapted pipeline was used to extend the analysis from 4 to 26 recordings using three sorting algorithms (MountainSort4, Kilosort2 and Ironclust). The result was a pandas DataFrame ([25]) in which each row corresponds to a unit and the columns correspond to quality metric values, the recording and sorter used, and a binary "matched" variable.

3.2 Pre-processing metrics data for classification

Whilst several pre-processing approaches were explored throughout this project, three are considered within this report, each of which is consistently referred to as the: "Standard" dataset, "Dummies" dataset and "Log" dataset.

Standard A copy of the original full DataFrame was created and the metrics columns and matched column were retained. A standard scalar was applied to each metric to normalise the mean and standard deviation using scikit learn's `preprocessing.StandardScaler()` function [26]. Finally, the metrics columns were separated from the matched column.

Dummies A copy of the full DataFrame was created. The "sorter" column was converted into numeric dummy variables, yielding one binary variable for each sorter (with "1" or "0" indicating whether that sorter was used to identify the unit). Whilst there are three sorters, two dummy variables are sufficient to represent all three. The excluded sorter variable (in this case Kilosort2) is represented by a “0” in both of the remaining dummy columns. A standard scalar transform was again applied to each metric column. Finally, the metric and sorter dummy columns were separated from the matched column.

Log Some quality metrics have heavy-tailed distributions. These metrics were identified by visual inspection of histograms and reference to the analysis conducted in the previous part of this project. To produce more plausibly Gaussian distributions for these metrics, a natural logarithm was applied by using the NumPy `log()` function. Again, the full DataFrame was copied, retaining only metrics and matched columns. The following metrics were log-transformed: firing rate, presence ratio, ISI violations, SNR, isolation distance, L-ratio, NN-hit rate, NN-miss rate and the absolute values of D-prime. Note that the magnitude of D-prime is used to indicate unit quality, therefore use of the absolute value of this metric is appropriate here. A standard scalar transform was then applied, and the matched column was separated from the metrics columns.

3.3 Modelling problem

The aim of this project was to develop a useful curation method in the form of a predictive model. This model uses unit metrics to identify units which correspond to neurons in the brain (true positives). The potential features for the model are the metrics discussed in Section 2.4, with the possible inclusion of the sorter used to acquire the unit. The target variable for the model is whether a unit corresponds to a neuron in the brain, and agreement is used as a proxy to this information. Therefore, the modelling problem addressed by this report is a classification problem with a binary outcome.

To summarise, a successful model will be a binary classifier which uses unit quality metrics as features and returns an indication of whether the unit is likely to be a true positive unit.

Two modelling paradigms are considered in this project: Gaussian Bayes and logistic regression. The rest of this section details each of these paradigms with reference to their respective assumptions and limitations.

3.3.1 Gaussian Naïve Bayes (GNB)

In a Gaussian Naïve Bayes model, Gaussian distributions are fit to each class (matched or unmatched) in the training data. Since features are assumed to be approximately Gaussian, pre-processing of features must be carefully considered. To classify an unseen unit, the probability of the unit’s membership for each class is calculated and the class with higher probability is selected. GNB models were fit using the scikit-learn function `naive_bayes.GaussianNB()` [26].

GNB additionally assumes that, for each class, all features are independent [18]. The validity of this assumption should be considered, as improvements in model performance may be achieved through accounting for the relationships between features.

It is possible to extend this naïve model to account for covariance between features. However, this limits the number of features which can be included in the model because the non-naïve Gaussian Bayes model suffers the curse of dimensionality [18]. This motivates a change in paradigm to logistic regression which can better handle large feature spaces.

Another limitation of GNB is that modelling sorters separately would require 6 different distributions to be fit to the training data (matched and unmatched distributions for each of the three sorters). Given the very limited number of matched units associated with each sorter, distributions fitted to individual sorters' matched unit populations are likely to be strongly influenced by outliers [18].

3.3.2 Logistic Regression

Logistic regression uses a Logistic function to model the probability that a datapoint belongs to a each of two binary classes. In a fitted model, each predictor variable is associated with a coefficient. Binary logistic regression is an appealing modelling paradigm because individual classifications are easy to interpret with reference to the coefficients of the model [18]. Additionally, coefficients can be used to establish feature importance, which aids variable reduction. This is helpful in this modelling problem because an interpretable model will not use a large number of predictor variables. Logistic regression models were trained using the scikit-learn function `linear_model.LogisticRegression()` [26].

A general guideline for logistic regression is that there should be at least 10 cases with the least frequent outcome (here matched units) for every independent variable [27]. This limit should be taken into account when choosing features for a model.

Overfitting the model to the training data is a risk of logistic regression, especially in small datasets [18]. To reduce the likelihood of overfitting, regularisation was considered. This protects the model from overfitting by penalising large coefficient values. The degree to which coefficient magnitudes are penalised is controlled by the regularisation constant, $\lambda \geq 0$. With little or no regularisation ($\lambda \approx 0$) the model may overfit the training data and therefore perform poorly on unseen data. L1 (LASSO) regularisation was used in this case, as this method has the additional benefit of aiding variable elimination by setting coefficients to exactly 0 at sufficiently high regularisation constant values [10]. In scikit-learn's `LogisticRegression()` function, the parameter `c` is used to control the regularisation constant λ, where $c = 1/\lambda$ is the inverse of the regularisation constant. Larger values of λ, and correspondingly smaller values of c, yield stronger regularisation.

Logistic regression requires that there is little multicollinearity between independent variables [18]. As a general rule of thumb, a Pearson's correlation coefficient above 0.8 is considered problematic [33]. However, whilst high correlation between two variables indicates multicollinearity, a lack of correlation cannot be assumed to imply lack of multicollinearity. This is because correlation is a special case of collinearity. Collinearity can also occur in the case of a linear relationship between one variable and a combination of the others [4, 18]. The presence of multicollinearity among features causes logistic regression to become unstable when features are added or removed, and should be avoided. This instability causes inflated coefficients, so regularisation can help minimise the impact of colinearity by reducing the magnitude of coefficients [4]. Whilst this does not solve the assumption violation, regularisation stabilises the model and may therefore be sufficient for the purpose of modelling.

To properly diagnose multicollinearity among features, Variance Inflation Factor (VIF) was used [4, 18]. In the calculation of VIF, each feature is regressed against all other features. The VIF score for a feature i can be expressed as a function of the coefficient of multiple determination in linear regression, R_i^2, which is a goodness of fit measure [18, 4]:

$$VIF_i = \frac{1}{1-R_i^2} \tag{3.1}$$

VIF was calculated using the statsmodels function `stats.outliers_influence.variance_inflation_factor()` with default parameters. Values above 10 are usually taken to indicate multicollinearity [4, 18], however some sources use a stricter threshold of 5 [18].

Logistic regression does assume that observations are independent from each other [18]. This assumption is potentially violated in this modelling task; as different sorters are applied to the same recordings, they may identify similar units.

One limitation of the logistic regression paradigm is that it does not provide meaningful estimates of the confidence associated with classifications [18]. This is in conflict with a goal of the curation model. Namely, that the model should provide some indication of confidence that a unit is matched or unmatched.

3.4 Model evaluation

This section discusses the model evaluation metrics and procedures which were used to assess model performance in this project. A true positive (TP) result refers to a unit which is matched, and is classified as such. Similarly, a true negative (TP) is an unmatched unit which is classified as such. A false positive (FP) is a unit which is unmatched but is classified as matched, and a false negative (FN) is an unmatched unit that has been classified as matched.

Accuracy Whilst accuracy is a common performance metric for classification models, it is not applicable in this case owing to the imbalance between matched units and the majority class of unmatched units. This is because a model can achieve high accuracy by simply classifying all samples as unmatched (the majority class).

Precision The primary goal of the model is to remove incorrect (unmatched) units, thereby reducing the likelihood of drawing conclusions about the data based on inaccurate assertions about neuron activity. In modelling terms, this is reflected by the precision on the target set:

$$precision = \frac{TP}{TP + FP} \tag{3.2}$$

Recall Another implicit goal of curation is to retain as many matched units as possible. This can be expressed as recall on the target set:

$$recall = \frac{TP}{TP + FN} \tag{3.3}$$

To put these considerations into a neuroscience context, one can consider the nature of extracellular neural research. For a study which seeks to examine the behaviour of an individual neuron, inclusion of false positives may have detrimental implications on the reliability of the conclusions drawn about neuron behaviour. This is because each identified neuron is important to the conclusions drawn. Such a study must prioritise precision over recall. By comparison, a study which examines the interaction between large populations of neurons may prioritise recall over precision. In this case, a loss of units may severely hinder the study, while inclusion of some false positives is tolerated. These examples highlight the importance of good curation, and put into perspective the way that priorities may change based on the needs of the study in question [13]. Given this, an ideal model should grant the researchers control of the degree to which they allow false positives in their analysis.

F-measure An F-β measure can be useful in expressing the relative priority of recall and precision [18]. It is possible to tune β to express relative prioritisation of recall and precision ($\beta > 1$ will favour recall). However, as explained above, these priorities are highly specific to the use case of the model. Therefore for the purposes of this report, the F1 measure will be used as a general indicator of performance with reference to precision and recall values. F1 score is defined as:

$$F_1 = \frac{2 \times precision \times recall}{precision + recall} \tag{3.4}$$

Train, validation and test set As is standard in machine learning, a small portion of the data was held out for validation (tuning) and testing. Usually, a random subset of the data is held out for this purpose. However, the nature of the agreement data complicates this procedure. A matched unit will, by construction, be represented by multiple sorters (ergo multiple data points). These separate representations of the matched unit are likely to have very similar, if not identical, quality metric values, as they represent the behaviour of the same neuron. In a random datasplit, one representation may fall in the training set while another falls in the test set. In this way, the model will be tested on the training data, leading to inflated test performance. Since this essentially constitutes data leakage, a random train/validation/test split would cause a validity issue for the results presented here. Therefore, this report will hold out *recordings* rather than an arbitrary split of the *units*. This prevents the data leakage concern, since data from different recordings cannot represent the same unit.

Python's `random.choices()` function was used to select two test and three validation recordings. Once determined, data from these recordings was consistently held out from the training data, which consists of the remaining 21 recordings. The test set was not used at any point during training, and was only used for the final model evaluation in Section 4.6.

Cross-validation A simple validation set approach to tuning would involve tuning all parameters based on performance on the validation set. Given that this project involves substantial tuning of one model to one dataset, such an approach would potentially lead to overfitting the model to the validation set. This effect would be reflected in poor performance on test data. To avoid this, K-fold cross-validation is preferred over a simple validation set approach [18]. Usually, K-fold cross-validation involves randomly splitting the training data into K groups. For each iteration ("fold") of the cross-validation, one of these groups is held out and treated as a validation set from which performance can be reported. Given the issue described above, this project holds out recordings as opposed to random groups. Therefore, this project will essentially perform "hold one recording out" cross-validation (simply referred to as "cross-validation" in this report).

3.5 Addressing data imbalance

Since matched units are rare, there is a class imbalance in the dataset, with the target set of matched units being the minority class. This can cause problems for modelling performance, specifically low recall on the minority class.

There are two approaches to address this problem: adjusting the model itself (for example, by adjusting class priors to favour the minority class) and adjusting the data directly prior to modelling [7, 34]. Adjustment of the data usually involves either oversampling the minority class or undersampling the majority class. Extracellular datasets are generally quite small, therefore oversampling was used in order to make use of the data available without further limiting the sample sizes by undersampling.

In this project, the Synthetic Minority Over-sampling Techique (SMOTE) oversampling technique was used to oversample the minority class [7]. The Python imblearn package was used to perform SMOTE oversampling using the `over_sampling.SMOTE()` function [22]. An "oversampling ratio" is provided as a parameter to this function. This ratio is the amount of resampled data which represents the (previously minority) class. For example, an oversampling ratio of 0.4 indicates that after resampling the training data, 40% of rows in the training set were matched units.

Chapter 4

Results and analysis

4.1 Dataset overview

In total, 2566 units were identified across all three sorters and all 26 recordings. Each electrode is expected to detect the activity from up to 4 units, so 16 units per tetrode recording is a plausible maximum. Given that there are 26 recordings, no more than 416 units are expected across all recordings, for each sorter.

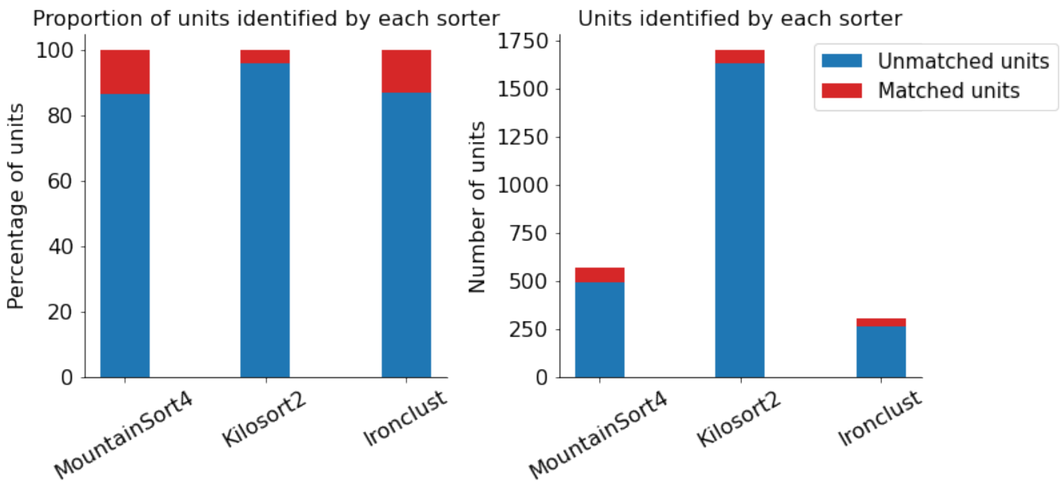

Figure 4.1: Bar charts showing the balance of matched (red) and unmatched (blue) units identified by each sorting algorithm. Left: Matched units as a percentage of overall unit output of each sorter. Right: Number of matched and unmatched units identified by each sorter.

Kilosort2 found 1698 units in total, of which 66 (3.89%) were matched. 1698 is 4.08 times the expected number of true units in the data. This result is in agreement with those found in the first part of this project, as well as results from the literature. Specifically, Kilosort2 reports implausibly high numbers of units, and therefore contains many false positive units [5]. Both MountainSort4 and Ironclust produce more reasonable numbers of units overall, which is in line with previous results. MountainSort4 found

565 units in total, of which 76 were matched (13.45%). Ironclust found 303 units in total, of which 39 units were matched (12.87%). Figure 4.1 (left) shows the proportion units identified by each sorter which were matched. Again, these results echo those presented in the previous year's report in that both MountainSort4 and Ironclust produce a high ratio of matched units relative to Kilosort2. As can be seen in Figure 4.1 (right), MountainSort4 and Ironclust both contribute similarly to the total collection of matched units in the data.

Across all recordings and sorters, 181 units were matched (7.05% of total units). The proportion of matched units overall was slightly lower in the first part of the project (average 4.27% over 4 recordings using 5 sorters). As mentioned in Section 2.2, sorters in this analysis were chosen for their high rates of matched units. Therefore it is unsurprising that a higher ratio of matched units was found in the three sorter dataset. Note that, despite this increase, these results are still consistent with previous literature demonstrating low agreement among sorters [5].

Summary These results highlight two issues for the modelling problem described in Section 3.3. Firstly, the results confirm that there is a class imbalance for the data used here. Secondly, the likelihood of a unit being in the target set is related to the sorter used. This suggests that the sorter used may be an important factor in determining the quality of a unit.

4.2 Quality metric distributions

Investigating the distributions of features used in modelling is important for two reasons. Firstly, some models make assumptions about the underlying distributions of features. For example, Gaussian Bayes assumes features are approximately Gaussian distributed. By visualising distributions, violations of these assumptions can be identified. Secondly, visualising distributions in a larger dataset may reveal mistakes or biases in the previous year's analysis, which were not visible at a smaller sample size.

Kernel Density Estimation (KDE) was used to attain estimated probability density functions for sample populations. This was done by use of pandas' `DataFrame.plot.kde()` function, using default parameters. Whilst histograms give a slightly more realistic picture of the dataset, KDE is here used to yield clear visualisations. For visualisation clarity, Figures 4.2 and 4.3 use the Log dataset. For completeness, visualisation of the raw data can also be found in Appendix A.

Since quality metrics will be used here to classify matched and unmatched units, some evidence of a relationship between distribution and class is a positive indication that a particular metric will be useful in identifying matched units. To investigate this, the Log dataset was split between matched and unmatched units and the KDE for each metric was visualised. Figure 4.2 shows the kernel density estimation distribution of each metric for matched and unmatched units, across all sorters.

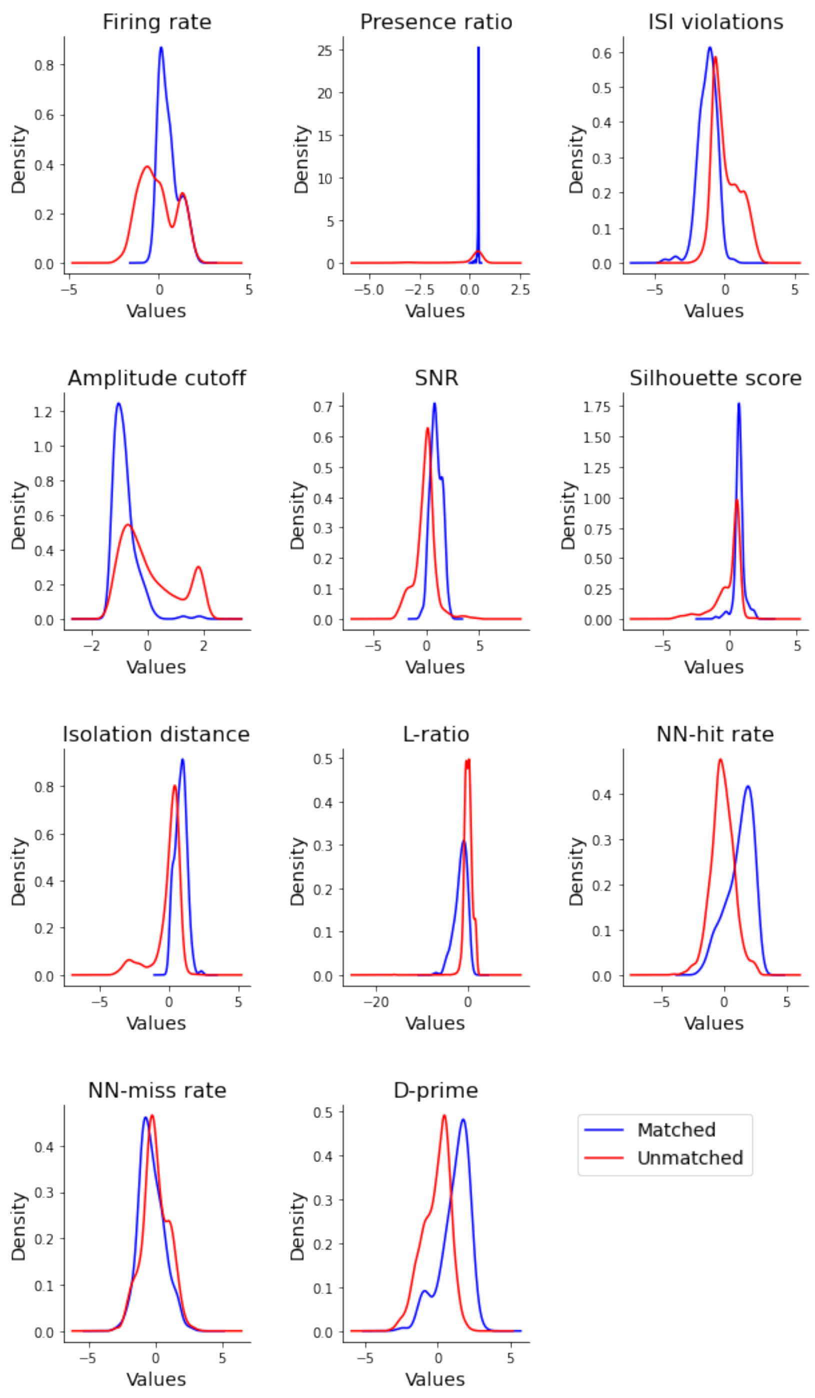

Figure 4.2: Kernel density estimation distribution plots for each metric in the Log dataset. Matched (blue) and unmatched (red) populations are represented separately.

For many metrics, there are clear differences in the distributions for the matched and unmatched populations. One example of this effect is amplitude cutoff, in which matched units tend to take on a more limited range of values, which are lower. Another example is presence ratio, in which matched units tend to take on the same value whist unmatched units take on a wider range of values, as can be seen by the flatter distribution here. An extensive discussion on this topic can be found in the previous part of this project. For the purposes of this part of the project, note that most metrics show some differences in the distribution of matched versus unmatched units. This is a promising indication that these metrics can be used as features in a classifier model.

Since sorters will ideally be modelled together in a classifier, there is an implicit modelling assumption that data for each metric is sampled from the same distribution. Therefore, units from different sorters should be similarly distributed for each metric. To investigate this, the full Log dataset was split by sorter to produce three datasets. Figure 4.3 shows the kernel density estimation distributions for each metric based on sorter.

Whilst no formal conclusions are drawn here, it appears that some quality metrics show similar distributions for different sorters (for example presence ratio, amplitude cutoff and isolation distance). By comparison, other metrics show distributions which are not similar across sorters. One example of this is firing rate, for which Kilosort2 units tend to take on lower values. Kilosort2 does have a higher ratio of unmatched units in comparison to MountainSort4 and Ironclust, so this difference in distribution may simply be reflective of lower quality units associated with this sorter.

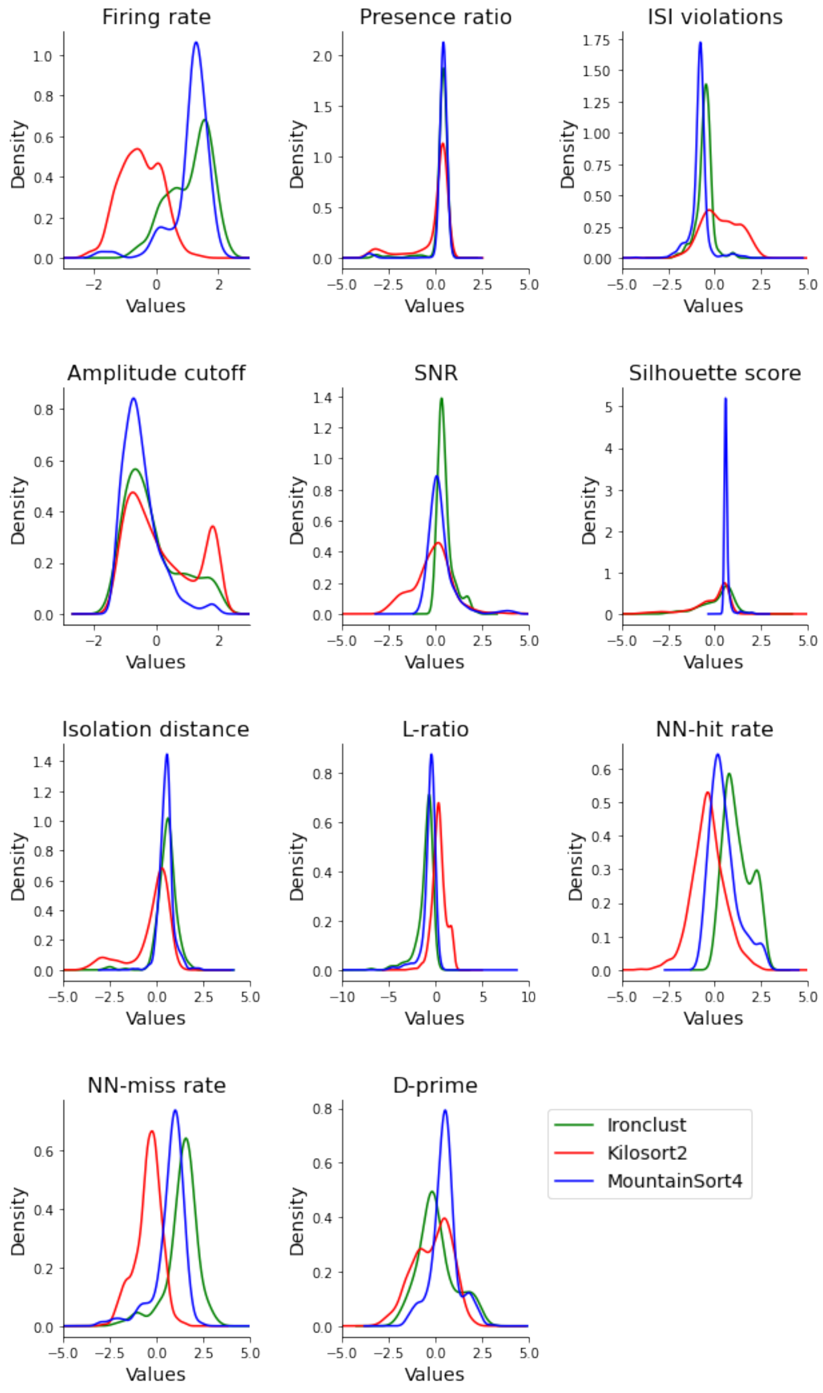

Figure 4.3: Kernel density estimation distribution plots for each metric in the Log dataset. Individual sorter populations are represented separately: Ironclust (green), Kilosort2 (red) and MountainSort4 (blue).

Summary Across the whole dataset, some metrics show highly similar distributions for matched versus unmatched units. These metrics may prove difficult to use in a classifier model because they do not separate classes well.

Quality metrics from different sorters are not necessarily similarly distributed. This makes it difficult to combine data from sorters when modelling. One potential solution to this is to separate metric information based on the sorter used. From a user perspective this is reasonable, since the sorter used is known to the user and can therefore be an input to the model. However, separating by sorter yields smaller datasets from which to train a classifier. Additionally, an ideal model should generalise to any choice of sorting algorithm. By accepting that a model requires experiment and sorter specific information in order to perform well, the assumption that this curation technique can be used in general for any collection of sorters is sacrificed. An alternative option is to fit a single model and include dummy variables indicating the sorter used (as in the Dummies dataset).

4.3 Feature correlation

As discussed in Section 2.5, metrics can be grouped by the type of error they aim to identify. This raises the question of whether these metrics with similar goals have strong relationships. That is, supposing that two different metrics both aim to describe unit isolation, one could expect that these metrics are highly correlated, with well isolated metrics scoring higher than poorly isolated metrics. If metrics are highly correlated, this presents an opportunity to reduce the feature space of the model. This is because using both metrics separately would essentially add redundant information which is already provided by another feature. Therefore, if two highly correlated metrics are present, one of these could be selected, discarding the other. A secondary reason for performing this correlation analysis is to check the direction of relationships between metrics. As described in Section 2.4, high quality units are indicated by high values of some quality metrics, and low values of others. That is, given metric A which takes on a lower value for good units, and metric B which takes on a higher value for good units, the correlation should have a negative sign.

To investigate correlation, Spearman's rank correlation coefficient was used to determine the covariation between pairs of metrics. Values for Spearman's correlation rank range from $\rho = -1$ (perfect monotonically decreasing relationship) to $\rho = 1$ (perfect monotonically increasing relationship). Spearman's rank coefficient is here preferable to alternatives which make stronger assumptions about the relationship between variables. For example, Pearson's correlation coefficient assumes that variables are linearly related [32]. Spearman's rank does assume that there is a monotonic relationship between quantities. Since D-prime is assessed based on magnitude (see Section 3.2), the Log dataset was used to ensure a monotonic relationship in this case. Figure 4.4 shows the correlation coefficients between metrics as a heatmap.

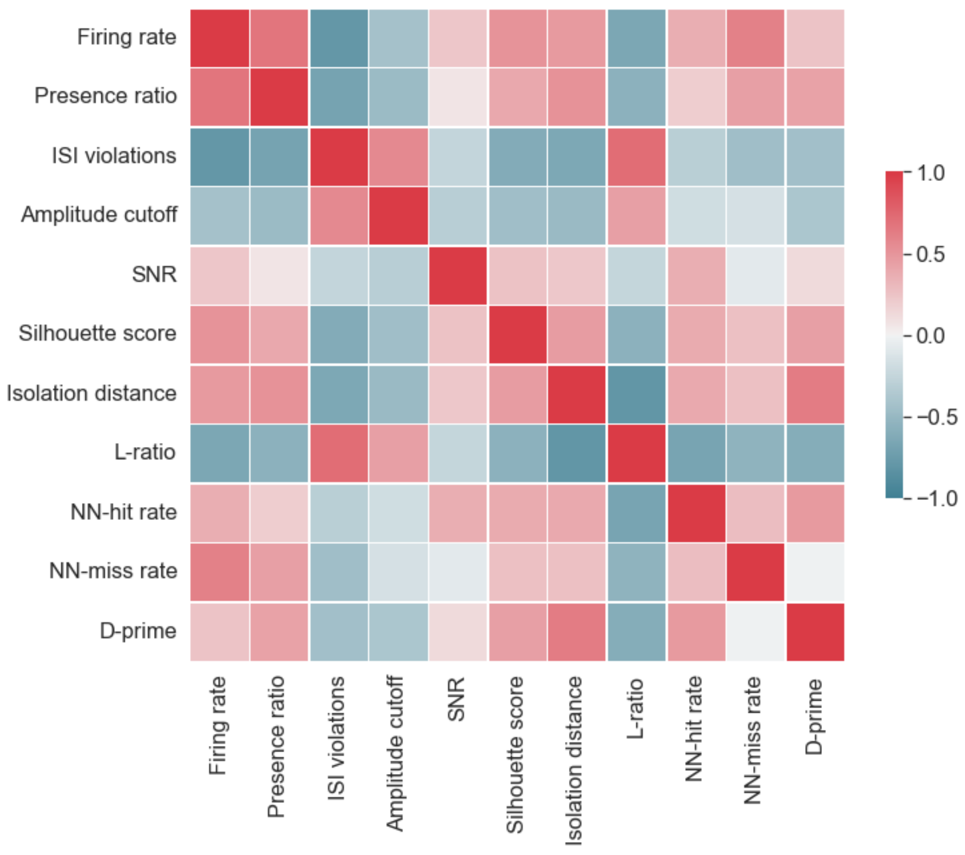

Figure 4.4: Heatmap showing the Spearman's correlation coefficient value between all quality metrics in the Log dataset.

The direction of the relationships are generally as expected. That is, for metrics where a low value indicates a good quality unit, a negative correlation is seen with metrics for which a positive value indicates a good unit. The exception to this is NN-miss rate, which should be lower in matched units, but shows positive correlation with some metrics which should be higher in matched units (eg: amplitude cutoff, presence ratio and isolation distance). This is surprising given that the overall relationship between matched and unmatched units for this metric (as outlined in Section 4.2) was as expected. One potential explanation for this is that NN-miss rate is related to other variables by functions which are not easily captured by Spearman's rank correlation coefficient [32].

Contamination metrics (type I error) As discussed in Section 2.5, the metrics which are strongly associated with identifying type I errors are: ISI violations, SNR, L-ratio and NN-hit rate. The relationship between these metrics is shown in Figure 4.5.

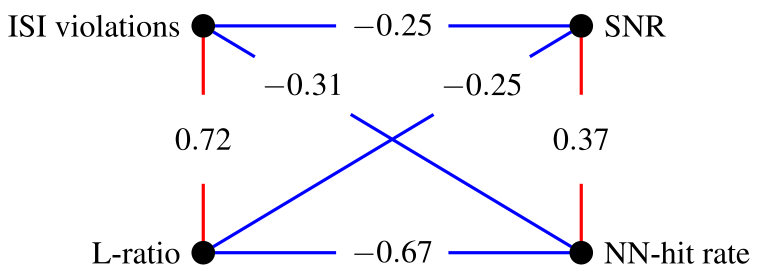

Figure 4.5: Connected graph showing the Spearman's correlation coefficient value between type I metrics. Positive correlations are indicated in red whilst negative correlations are indicated in blue.

Incompleteness (Type II) metrics As described in Section 2.5, metrics which are most strongly associated with type II errors are: presence ratio, amplitude cutoff, NN-miss rate and isolation distance. The relationship between these metrics is visualised in Figure 4.6.

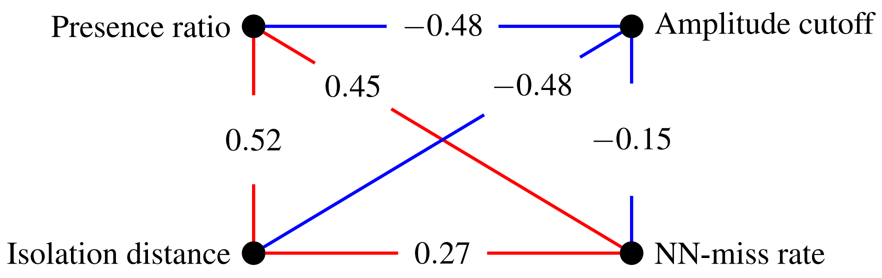

Figure 4.6: Connected graph showing the Spearman's correlation coefficient value between type II metrics. Positive correlations are indicated in red whilst negative correlations are indicated in blue.

Summary Whilst some evidence of correlation between metrics with the same error type is observed, these correlations are not very strong, certainly not strong enough to assert redundancy of some metrics. Some evidence of correlation between metrics is observed, however the correlations observed between conceptually related metrics are not substantially stronger than those of non-conceptually related metrics. This suggests that model features should not be reduced by simply considering the error type addressed by the metric.

4.4 Baseline model

The Gaussian naïve Bayes (GNB) model presented in Part 1 of this project was trained on the full dataset. Log-ISI violations and log-L-ratio were used as the features for this model by using the Log dataset.

Two `GaussianNB` models were then fit separately on the training data. The first model was exactly equivalent to the model presented in Part 1 of the project. This model assumed equal class priors to address class imbalance, $P(\text{matched} = 0) = 0.5$ and $P(\text{matched} = 1) = 0.5$.

Using this equal priors model, the F1 score on the validation set was 0.476. On the matched set, precision was only 0.316, but recall was very high at 0.962. These results are reflective of those seen in Part 1 (recall: 0.910, precision: 0.140), which used a smaller dataset. The improvements seen using the larger dataset are attributable to the improved balance of data in this years dataset as discussed in Section 4.1. As can be seen in Figure 4.7, the model which uses equal priors produced good rates of recall for both classes, and very few true positive units are classified as negatives. This model retains matched units well, but incorrectly classifies many unmatched units as matched.

The second model used priors according to class prevalence in the data. Approximately 7% of units were matched in the training set, and priors were set accordingly using the default parameters of scikit-learn's `GaussianNB()` function which automatically sets priors according to class prevalence. Unlike the equal priors model, this model does not make the assumption that a matched unit is equally as likely as an unmatched unit. In this way, the model is more accurate to the assumptions that should naturally be made about the data. On the validation set, the unequal priors model had a slightly higher F1 score of 0.585, with precision of 0.800 on the target set, and substantially lower recall of 0.462.

These results demonstrate that adjusting the priors to improve recall causes a decrease in precision, whilst adjusting priors to improve precision causes a decrease in recall. This trade off effect is visualised in Figure 4.7.

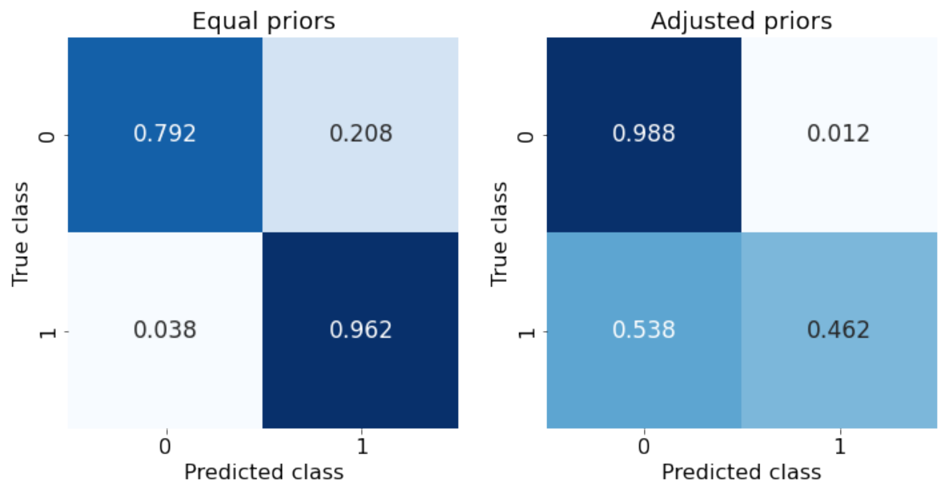

Figure 4.7: Confusion matrices showing performance of Gaussian Naïve Bayes models on the validation set. Values shown are adjusted to show the ratio of labels predicted correctly in each category. Left: Equal priors model. Right: Model using priors adjusted to class prevalence.

In the equal priors model, the model was adjusted as an attempt to address the effects of class imbalance. As discussed in Section 3.5, oversampling the training data is an alternative method to address the effects of imbalance. SMOTE oversampling was applied to the training data using an oversampling ratio of 0.5. This yielded 3536 training samples, of which 1768 (50%) were matched. In this case, priors in the GNB model were automatically tuned to 0.5 for each class, reflective of the prevalence of each class in the rebalanced training data. Testing on the validation set as before, results are very similar to that of the equal priors model above (F1: 0.472, precision: 0.313, recall: 0.962). In the case of this model, oversampling the minority set yields similar results to adjusting priors.

Summary GNB models produce reasonable results, and the trade off between recall and precision can be adjusted either through adjusting priors or oversampling the training data. However, prioritising either recall or precision severely impacts the performance of the other. An ideal model will retain good performance on both recall and precision, regardless of which is prioritised. The equal priors GNB model presented here serves as a baseline from which to compare the models developed throughout the rest of this chapter.

4.5 Logistic Regression

As discussed in Section 4.1, there are 181 matched units across the full dataset of 2566 units (7.01%). A single recording is therefore expected to have 98.69 units, of which 6.92 are matched. The smallest training set used to train the model here is a single cross-validation fold, consisting of 20 recordings (138 matched units expected). Given the assumption that there should be at least 10 observation per feature used (see Section 3.3.2), this limits the number of features to 13. This is a high limit, and even use of all metrics and sorter dummy variables only just reaches this limit of 13. For an interpretable model, a smaller limit is preferred, certainly less than 10, and ideally less than 5 or 6.

To address the assumption that there is no multicollinearity among features, the VIF score was calculated for the two scenarios: "metrics only" in which metrics are the predicting features, and "metrics and sorter" in which both metrics and sorter dummy variables are used as predicting features. None of the VIF scores exceed the threshold of 10, nor the stricter threshold of 5. Therefore, the multicollinearity assumption made by logistic regression is not violated in either case. A full table showing all VIF scores is available in Appendix B. This result further supports the conclusions drawn in Section 4.3, namely that assuming redundancy of features is ill founded. For this reason, the modelling approach here treats each quality metric as a feature, without attempting to combine features prior to modelling.

4.5.1 Pre-processing for logistic regression

This section considers the impact of data pre-processing on model outcome. The three datasets outlined in Section 3.2 were considered for the logistic regression model.

Logistic regression was used to acquire an estimate of feature importance under each of the three pre-processing conditions. To achieve this, cross-validation was performed. For each fold, a logistic regression model was trained, using default regularisation $c = 1$, to establish feature importance. Figure 4.8 shows the coefficient magnitude (averaged across folds) for each quality metric under differing pre-processing conditions. From this figure, it is clear that choice of pre-processing has an effect on the relative importance of features in the model. This result is important because variable elimination is applied in reverse order of coefficient magnitude, meaning that the reduced collection of variables used in a final model may differ depending on the pre-processing applied.

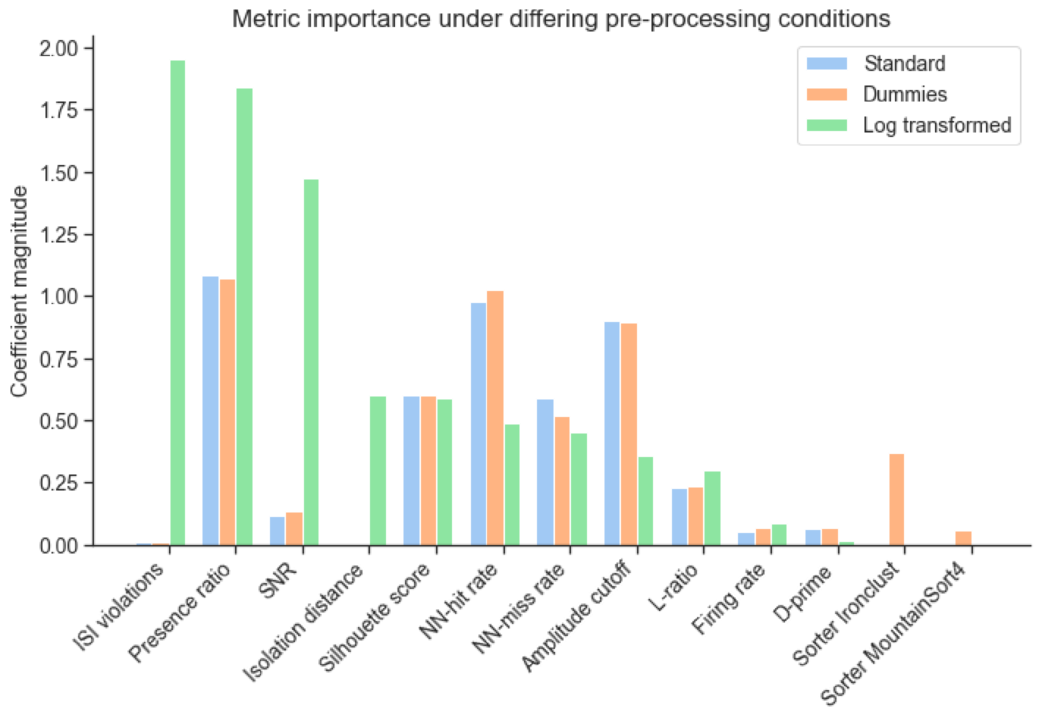

Figure 4.8: Bar chart showing the absolute values of quality metric coefficients produced by a logistic regression model under different pre-processing conditions. Each model uses the full feature set for each dataset, with regularisation $c = 1$. Coefficient values represent the average across cross-validation folds.

In comparison to the Standard dataset, inclusion of a dummy variable for sorter (Dummies dataset) has little impact on the importance of most metrics. Additionally, the sorter dummy variables are of fairly low importance, meaning they would be eliminated in a model with few variables. By comparison, use of the Log dataset yields substantial differences in metric coefficient size. Interestingly, ISI violations and isolation distance, which are both log-transformed in the Log dataset, are not ranked as important under the standard scalar pre-processing (Standard and Dummies datasets), but are very important in the Log dataset case. This highlights both the benefit and risk of re-parameterising variables. On one hand, transforming these variables has produced features which can better differentiate between matched and unmatched data, which is ideal for modelling. On the other hand, the model is clearly sensitive to pre-processing choices, therefore careful consideration is required.

Another observation is that the sign of the coefficients are usually as expected: metrics where high values are good have positive coefficients and vice versa. The exception to this is SNR, for which both Standard and Dummies datasets yield small negative correlation coefficients. The Log dataset does not have this problem, and SNR is associated with a positive coefficient. The same effect is also seen in D-prime; in the log transformed model, D-prime is reduced to 0 by L1 regularisation. In models which use the Log dataset, the signs of coefficients are consistently in line with expectations for each metric.

Based on the coefficients found above, an ordered list of metrics was created for each dataset. This list was used as an order for feature selection, with the first metric representing the most important feature. Feature selection was performed by training models on the training data (regularisation $c = 1$) using increasingly large numbers of variables, starting with the most important. For each dataset, the maximum F1 score was identified, as well as the number of variables used to attain this score.

Using the Standard dataset, the optimal F1 score of 0.756 was achieved using 4 variables: presence ratio, NN-hit rate, amplitude cutoff and NN-miss rate. Recall was 0.654 and precision was 0.895. Using the Dummies dataset, 9 variables (all but ISI violations and isolation distance) were required to achieve an optimal F1 score of 0.744 (recall: 0.615, precision: 0.941). Using the Log dataset, only 4 variables (ISI violations, presence ratio, SNR and isolation distance) were needed to achieve an optimal F1 score of 0.756 (recall: 0.654, precision: 0.895).

Performance for all pre-processing choices was similar, however far more variables were required to achieve this performance in the Dummies dataset. Ideally the final model should not use many variables, therefore the Dummies dataset was excluded from analysis at this point. It is surprising that the sorter variables are not required to achieve good modelling performance. This suggests that the differences in metric distributions between sorters (as in Section 4.2) do not detrimentally affect performance of the model.

In Part 1 of this project, ISI violations was identified as having a uniquely consistent relationship with unit agreement across all recordings and all sorters tested. This suggests that ISI violations is a useful predictor in the curation model. However, this metric is given a very small coefficient in the model which uses the Standard dataset. In the Log dataset models, ISI violations is given the largest coefficient. Given this, the Log dataset produces models which are more in line with expectations. Additionally, since the sign of coefficients is more consistently in line with expectations in the Log dataset, the rest of this analysis uses this pre-processing configuration.

4.5.2 Tuning regularisation

Having identified the Log dataset as appropriate for modelling, and identifying 4 variables as the optimal number in this model, the next step for tuning the model is to consider regularisation.

To identify an appropriate value for the inverse regularisation constant c, a hyperparameter search was conducted using cross-validation. The features used to train the model were the four features identified as optimal in the previous section: ISI violations, presence ratio, SNR and isolation distance. For each fold, logistic regression models were fit using c values ranging from 0.01 to 2 at intervals of 0.01. For each fold, the model which generated the maximum F1 score was identified to yield an optimal c value for that fold. Averaging over all folds, the mean optimal c value was 0.196 (min: 0.010, max: 0.710). Inspection of the individual fold results showed that the optimal F1 score was always achieved for values of c above the identified optimum. That is, the identified optimal c value represents a minimum within the tested range of 0.01 to 2. For c values below the suggested optimum value performance quickly dropped, suggesting underfitting. Given this, the highest observed c value represents a sensible minimum c value for the model. These results suggest that regularisation should be no stronger than $c = 0.71$

The same procedure was repeated with a higher range of c values (2 to 200 at integer intervals) to detect an upper threshold for c. Again looking at individual folds, the lowest c value for which performance dropped was 25. This suggests that overfitting may occur for c values above 25. These results suggest that some regularisation ($c <$ 25) is useful to prevent overfitting.

This report assumes that an interpretable model will not include more than 6 variables as predictors. To account for potential increases in the number of variables used, the same tuning procedure was performed using a 5 and 6 variable model. Similar thresholds for c were found. Given this, values for c between 1 and 25 are assumed to produce reasonable results for a model which uses 4-6 variables.

4.5.3 Variable selection

To establish the initial 4 variable model, variable elimination was performed against the validation set. Whilst this may be optimal for the full validation set, cross-validation may reveal variations in performance for a model with a given number of variables. Given this, cross-validation was used to assess the optimal number of variables to be used in the model. The same ordering of variables was used as in Section 4.5.1.

For each of the 21 cross-validation folds, variable elimination was performed in line with the procedure used on the validation set in Section 4.5.1. A regularisation parameter of $c = 1$ was used throughout, in line with results from Section 4.5.2. For each fold, the model which produced the highest F1 value was identified and the number of variables used was recorded.

Here, 2 is considered to be a reasonable lower limit for the number of features used in the curation model, with 6 as the upper limit as above. 10 of the folds produced optimal F1 values using 2-6 variables. 7 folds produced optimal F1 scores with just 1-2 variables. These folds generally had lower F1 scores (< 0.6) than those suggesting higher numbers of variables. This suggests that the held out recordings in these folds are difficult to classify using a logistic regression model based on any number of metrics. 4 folds suggested numbers of variables above 6. These folds were inspected visually, looking at the F1 score as variables are reduced.

In two of the folds, reducing the number of variables down from the suggested value to just 4 yielded minimal decreases in performance. Specifically, F1 score dropped from 0.667 to 0.636 in one case, and in the second case, F1 score was the same (0.500) to three significant figures after the reduction. Another fold suggested 9 variables, with an optimal F1 score of 0.857. When reduced to 4 variables, this score dropped marginally to 0.800. For this fold, the decrease in complexity by removing 5 variables is substantial in comparison to the reduction in performance, which is small. The largest performance drop was seen in a fold which suggested 9 variables, with F1 score of 0.609. Use of 4 variables decreases this score to 0.476, which was the only performance loss greater than 0.1. Even if 6 variables are used, this fold only achieves an F1 score of 0.552. This suggests that the units in the held out recording for this fold were difficult to classify, and required information from many different variables in order to be predicted correctly. Overall, these results show that in almost all cases, use of 4-6 variables produces either optimal or near-optimal performance based on F1 score.

To inspect model performance over all folds for a given number of variables, cross validation was performed. Mean and variance for F1 scores across folds are here reported using the notation (μ, σ^2). The best F1 score was seen when using 6 variables (μ = 0.508, σ^2 = 0.084). Mean recall in this case was 0.476 (σ^2 = 0.107) and mean precision was 0.638 (σ^2 = 0.144). Use of 5 variables does not cause a steep drop in performance; the mean F1 score for this model is 0.490 (σ^2 = 0.084), with mean recall of 0.458 (σ^2 = 0.104) and mean precision 0.629 (σ^2 = 0.137). Considering the results using only 4 variables, the mean F1 score is close to that of the 6 variable model (μ = 0.490, σ^2 = 0.084) with mean recall 0.462 (σ^2 = 0.104) and mean precision 0.629 (σ^2 = 0.134). Use of 4, 5, or 6 variables produce similar rates of recall and precision. Below 4 variables, recall declines quickly. These results suggest that a model of 4-6 variables produces reasonable results in comparison to a lower number of features. Use of 6 variables does produce some modest improvements in all performance metrics. The next section investigates whether this performance can be improved by addressing the class imbalance using SMOTE.

4.5.4 Oversampling to balance recall and agreement

Using a regularisation parameter of $c = 1$ and 6 variables (ISI violations, presence ratio, SNR, isolation distance, silhouette score, NN-hit rate), SMOTE was used to adjust the proportion of the training data which was matched. Scores on the validation set for each of these proportions ranging from 0.1 to 0.9 are shown in Figure 4.9.

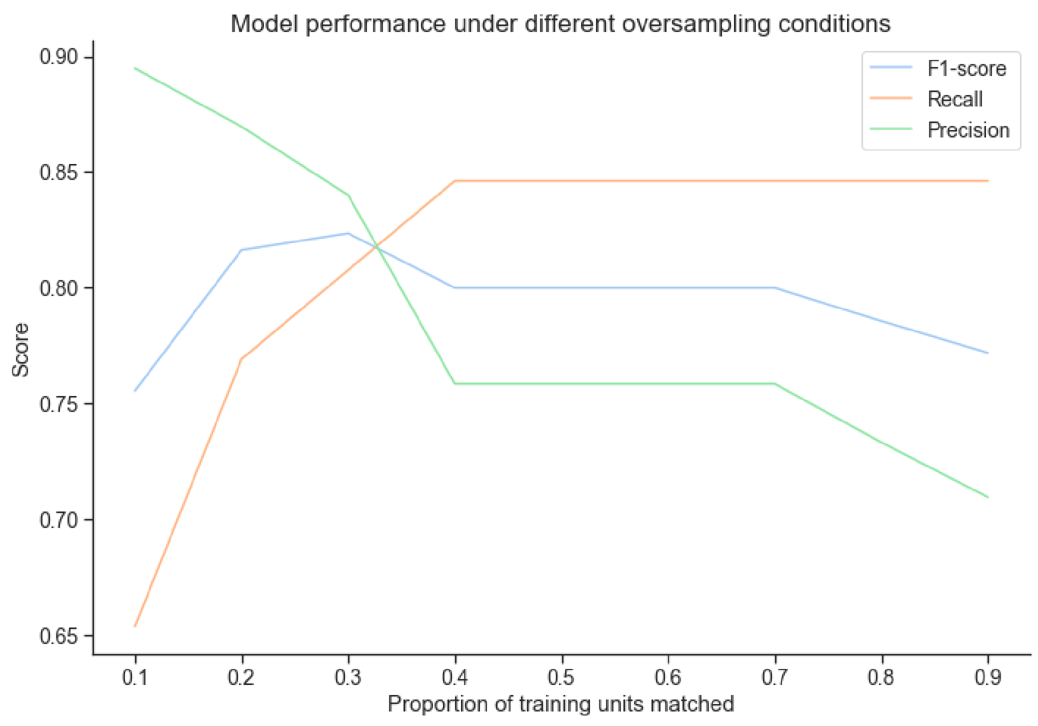

Figure 4.9: Graph showing the validation set performance of the oversampled logistic regression model ($c = 1$) across different oversampling ratios. The Log dataset was used. Features for the model are: ISI violations, presence ratio, SNR, isolation distance, silhouette score, NN-hit rate.

The optimal F1 score of 0.824 is reached at an oversampling ratio of 0.3 (recall: 0.808, precision: 0.840). This is a very exciting result, as this far exceeds the baseline performance mentioned in Section 4.4 (F1: 0.476, recall: 0.962, precision: 0.316). Of course, here a sacrifice has been made with regards to recall. However, as shown in Figure 4.9, recall can be improved by using a higher oversampling ratio of, for example, 0.4 to achieve 0.846 recall (F1: 0.80, precision: 0.759). This achieves close recall to that of the baseline, without the heavy sacrifice incurred on precision in the baseline model. On the other hand, if the user of the model wishes to prioritise precision and accept a decline in small recall, a lower oversampling ratio can be used (eg: 0.2) to achieve 0.870 precision (F1: 0.816, recall: 0.769).

4.5.5 Modelling sorters separately

As described in Section 4.2, different sorters' metrics are not necessarily similarly distributed, and cannot be assumed to be drawn from the same distribution. Ignoring this by modelling all sorters together may provide more data from which to train, but fails to capture information which is important to classification. This section investigates whether modelling sorters individually yields substantial performance gains over the pooled model.

Scikit learn's `LogisticRegression()` function was used to perform the regression fit using $c = 1$. Cross-validation was used to attain an average coefficient across folds for each sorter. The coefficient values for each metric and each sorter are shown in Figure 4.10. For most metrics, importance does not differ substantially between sorters. For a handful of metrics importance is very low for one sorter but moderately important to others. For example, both firing rate and isolation distance have very low coefficients in the Ironclust model, but coefficients are moderate for both MountainSort4 and Kilosort2.

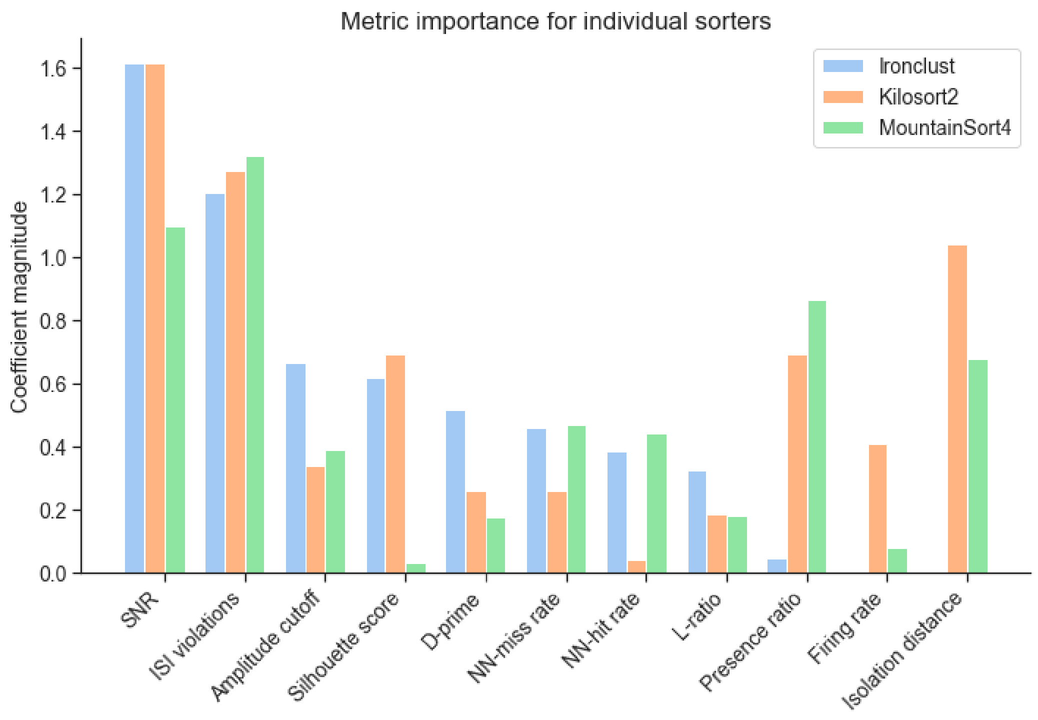

Figure 4.10: Bar chart showing the absolute values of quality metric coefficients produced by a logistic regression model using split-sorter Log datasets. Each model uses the full feature set for each dataset, with regularisation $c = 1$. Coefficient values represent the average across cross-validation folds.

Considering the metrics which are used in the pooled model: ISI violations and SNR are both highly important to all sorter models, presence ratio and isolation distance are both moderately important to all but Ironclust, silhouette score is moderately important to all but MountainSort4, and NN-hit rate is moderately important to all but Kilosort2. The pooled model features therefore reflect a well balanced set of features which are important to each sorter individually.

A similar procedure to that used on the full dataset was used to tune a model to the individual sorter data, again using cross-validation. In every case, $c = 1$ was found to be an acceptable regularisation constant.

Ironclust All but one fold produced optimal F1 scores using 5 or fewer variables. Oversampling did improve performance, with one peak using an oversampling ratio of 0.4 (F1: 0.553, recall: 0.619, precision: 0.552) and another at an oversampling ratio of 0.9 (F1: 0.604, recall: 0.710, precision: 0.558).

Kilosort2 All but one fold suggested a number of variables at or below 5, so again this value was chosen. The peak F1 score was seen at an oversampling ratio of 0.2 (F1: 0.331 recall: 0.419, precision: 0.366). Recall does improve with oversampling, with peak recall of 0.710 seen at an oversampling ratio of 0.9, but this causes substantial decreases in precision (0.279).

MountainSort4 All but two folds suggested 6 or fewer variables, therefore 6 were used. F1 score was fairly stable across different oversampling ratios, with a steady increase from 0.553 (oversampling ratio: 0.2) to 0.595 (oversampling ratio: 0.9). Recall improves with oversampling, with a maximum value of 0.726 using an oversampling ratio of 0.9 (F1: 0.595, precision: 0.556). MountainSort4 showed a less extreme decrease in precision at higher oversampling ratios, when compared with Kilosort2.

Summary As explained in Section 4.2, modelling sorters separately is not ideal. The results presented in this section demonstrate that modelling sorters separately does not even yield improvements in performance when comparing to the pooled model. One possible reason for this poor performance is that less data is used to train individual models. That is, the effect of a larger dataset in the pooled model is more influential over performance than the effect of differing metric distributions in each sorter. The fact that improved performance can be achieved by a pooled model is very encouraging for future work, as this demonstrates that treating sorters separately is not essential to achieve good model performance.

4.6 Evaluation

Given the intended use of the model, it is somewhat arbitrary to assert that a single oversampling ratio produces the "optimal" model. Therefore, results on the test set are given for three models, representing three potential modelling scenarios. A lower oversampling ratio (here 0.2) represents a slight priority of precision, in which few false positives are tolerated. By comparison, a higher oversampling ratio (here 0.4) represents a slight priority of recall, in which a higher number of false positives are tolerated. An oversampling ratio of 0.3 represents the balanced case, in which recall and precision are considered to be of equal importance.

Table 4.1 shows the test set performance of the equal priors Gaussian Bayes baseline model (Section 4.4) and the final logistic regression classifier (Section 4.5.4).

Model used	Oversampling ratio	F1 score	Recall	Precision
Baseline model		0.303	1.0	0.179
Final model	0.2	0.762	0.800	0.727
	0.3	0.727	0.800	0.667
	0.4	0.696	0.800	0.615

Table 4.1: Table showing F1 score, recall and precision for the test set using the baseline equal priors Gaussian Bayes model, and three settings for oversampling ratio (0.2, 0.3, 0.4) using the final logistic regression model presented in Section 4.5.4.

The baseline model has achieved perfect recall of the target set, but at the cost of precision, which is very low. In the Gaussian Bayes modelling paradigm, the gains in recall are achieved at a heavy cost to precision. By comparison, the rebalanced logistic regression model achieves good recall without this heavy cost to precision. This effect in particular is an achievement of the logistic regression model. Whilst there is still an unavoidable trade off between recall and precision, this effect is far less extreme than that seen in the Gaussian Bayes baseline model.

4.7 Discussion

Interpretation Whilst the specific oversampling ratio chosen for the model can be tuned based on the priorities of the user, this section addresses interpretation of the model trained with an oversampling ratio of 0.3, as an example. The coefficients associated with each of the 6 model features can be used to yield an interpretation of the factors which influence a particular unit's inclusion in further analysis. The coefficient with the largest magnitude was presence ratio, with a positive coefficient of 5.918. Following this, SNR had a coefficient of 2.438, ISI violations had a coefficient of -2.27 and isolation distance had a coefficient of 2.223. Finally, silhouette score had a coefficient of 1.645, and NN-hit rate had a coefficient of 0.352. The sign of each of these coefficients was in line with the expectations laid out in Section 2.4.

Features Reflecting on the intended use of each metric (see Section 2.5), the 6 variable model presented here consists of a balanced combination of different metric types. Type I errors are addressed by SNR, ISI violations and NN-hit rate. Type II errors are addressed by presence ratio (the largest coefficient). Isolation distance attempts to address both type I and type II errors, with a primary focus on type II errors. Silhouette score addresses both type I and type II errors. Additionally, there are three metrics in the model which address unit isolation (namely silhouette score, isolation distance and NN-hit rate) and three which identify other qualities.

Unlike the model presented in year 1, the features for the model presented here were selected based on the data alone, without directly attempting to select variables which capture a broad range of qualitative features. Given this, it is encouraging that the features selected for the final model represent a broad range of unit qualities. It is expected that an acceptable unit has several desirable qualities. For example, a good unit should be both uncontaminated and complete. If the model used only metrics which identified contamination, for example, incomplete units could erroneously be classed as positive. As noted in Section 4.5.5, the features used in the full model also represent a balanced set of the features which are most important to each individual sorter. This balanced combination of metrics in the tuned model is a pleasant reassurance that both the assumptions made when modelling, as well as the model itself, are coherent with our ideas on what a good curation model should capture.

Limits and extensions One limitation to the logistic regression modelling paradigm is that it is difficult to evaluate confidence in an individual classification. The final model can be used to gain some insight into the confidence of a unit, by checking the minimum oversampling ratio at which the unit is classed as positive. That is, if a unit is classified as matched, even at very low oversampling ratios, it seems likely that this unit is a true positive. However, this is not a formal gauge of confident, which would require some notion of class probabilities in the modelling paradigm used. To overcome this problem, a probabilistic model such as Bayesian logistic regression is suggested as future work.

A broader limitation to the approach taken here is that the model design is based on data from one experiment. Different datasets, especially those gathered using different recording devices, may require a completely different model in order to classify. This is suggested as future work in the domain of automated curation methods. Having said this, the goal of this project was to demonstrate that metrics can be used as predictors of agreement for the purposes of curation. Therefore the model presented here, and the accompanying analysis, form the foundation for future development to this novel curation approach.

Chapter 5

Conclusions

This report has examined the novel possibility of a fully automated curation method for spike sorter results. This analysis first corroborated results from Part 1 of this project regarding class imbalance and prevalence of agreement units. Use of MountainSort4, Kilosort2 and Ironclust produced a higher ratio of matched units across the full dataset, when compared with the five sorter analysis performed in Part 1 of this project. This suggests that choice of sorting algorithm should be a consideration in future curation attempts of this kind.

The relationship between different metrics was investigated. This revealed that metrics which share qualitative aims do not show high correlation, suggesting that different metrics should not be considered to provide equivalent information about unit quality. Investigation of the relative importance of different metrics supplemented this analysis, by demonstrating that a small subset of 4-6 metrics is sufficient to achieve performance close to that of a full 11 feature model. This suggests that a fully automated curation model does not need to sacrifice explainability in order to achieve good performance.

Methods for addressing class imbalance when modelling were examined. Adjusting class prior probabilities was found to be useful in raising recall in the Gaussian Bayes modelling paradigm, but at the cost of precision. Oversampling using SMOTE was found to be an effective method by which to raise recall without incurring heavy losses in precision. Additionally, oversampling presents an opportunity to express priorities of recall or precision.

The influence of sorting algorithm on unit agreement was addressed by examining the differences in metric distributions for different sorters, modelling sorters as dummy variables in a logistic regression model, and finally modelling sorters separately. Promising classification results were found without relying on sorter-specific features. This suggests that a general purpose curation model which performs well for many different sorting algorithms is possible.

Bibliography

[1] Allen Brain Atlas Guide to Quality Metrics. `https://allensdk.readthedocs.io/en/latest/_static/examples/nb/ecephys_quality_metrics.html`. Accessed: 2022-02-24.

[2] Kilosort2. `https://github.com/MouseLand/Kilosort2`. Accessed: 2021-09-24.

[3] SpikeInterface. `https://github.com/SpikeInterface`. Accessed: 2022-04-12.

[4] Aylin Alin. Multicollinearity. *Wiley interdisciplinary reviews. Computational statistics*, 2(3):370–374, 2010.

[5] Alessio P. Buccino, Cole L. Hurwitz, Samuel Garcia, Jeremy Magland, Joshua H. Siegle, Roger Hurwitz, and Matthias H. Hennig. Spikeinterface, a unified framework for spike sorting. 2020-11-30.

[6] Fernando J. Chaure, Hernan G. Rey, and Rodrigo Quian Quiroga. A novel and fully automatic spike-sorting implementation with variable number of features. *Journal of Neurophysiology*, 120(4):1859–1871, 2018.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *The Journal of artificial intelligence research*, 16:321–357, 2002.

[8] Jason E. Chung, Jeremy F. Magland, Alex H. Barnett, Vanessa M. Tolosa, Angela C. Tooker, Kye Y. Lee, Kedar G. Shah, Sarah H. Felix, Loren M. Frank, and Leslie F. Greengard. A fully automated approach to spike sorting. *Neuron*, 95(6):1381–1394.e6, 2017.

[9] Klara Gerlei, Jessica Passlack, Ian Hawes, Brianna Vandrey, Holly Stevens, Ioannis Papastathopoulos, and Matthew F. Nolan. Grid cells are modulated by local head direction. *Nature Communications*, 11(1), 2020.

[10] Ali S Hadi and Samprit Chatterjee. *Regression analysis by example*. Wiley series in probability and statistics. Wiley, Somerset, 5th ed edition, 2012.

[11] Kenneth D. Harris, Darrell A. Henze, Jozsef Csicsvari, Hajime Hirase, and György Buzsáki. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *Journal of Neurophysiology*, 84(1):401–414, 2000.

[12] Kenneth D Harris, Hajime Hirase, Xavier Leinekugel, Darrell A Henze, and György Buzsáki. Temporal interaction between single spikes and complex spike bursts in hippocampal pyramidal cells. *Neuron (Cambridge, Mass.)*, 32(1):141–149, 2001.

[13] Kenneth D Harris, Rodrigo Quian Quiroga, Jeremy Freeman, and Spencer L Smith. Improving data quality in neuronal population recordings. *Nature neuroscience*, 19(9):1165–1174, 2016.

[14] Matthias H. Hennig, Cole Hurwitz, and Martino Sorbaro. Scaling spike detection and sorting for next-generation electrophysiology. *Advances in neurobiology*, 22:171–184, 2019.

[15] Darrell A. Henze, Zsolt Borhegyi, Jozsef Csicsvari, Akira Mamiya, Kenneth D. Harris, and György Buzsáki. Intracellular features predicted by extracellular recordings in the hippocampus in vivo. *Journal of Neurophysiology*, 84(1):390–400, 2000.

[16] Daniel N. Hill, Samar B. Mehta, and David Kleinfeld. Quality metrics to accompany spike sorting of extracellular signals. *Journal of Neuroscience*, 31(24):8699–8705, 2011.

[17] Jadin Jackson, Neil Schmitzer-Torbert, K.D. Harris, and A.D. Redish. Quantitative assessment of extracellular multichannel recording quality using measures of cluster separation. *Soc Neurosci Abstr*, 518, 01 2005.

[18] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer texts in statistics. Springer, New York, NY, 2021.

[19] Magland Jeremy, James J. Jun, Elizabeth Lovero, Alexander J. Morley, Cole L. Hurwitz, Alessio P. Buccino, Samuel Garcia, and Alex H. Barnett. Spikeforest, reproducible web-facing ground-truth validation of automated neural spike sorters. *eLife*, 9, 2020.

[20] James J. Jun, Catalin Mitelut, Chongxi Lai, Sergey L. Gratiy, Costas A. Anastassiou, and Timothy D. Harris. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *bioRxiv*, 2017.

[21] Ryan C. Kelly, Matthew A. Smith, Jason M. Samonds, Adam Kohn, A. B. Bonds, J. Anthony Movshon, and Tai Sing Lee. Comparison of recordings from microelectrode arrays and single electrodes in the visual cortex. *Journal of Neuroscience*, 27(2):261–264, 2007.

[22] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[23] R. Lemon. Methods for neuronal recording in conscious animals. *IBRO Handbook Series*, 4:56–60, 1984.

[24] Liqun Luo, Edward M Callaway, and Karel Svoboda. Genetic dissection of neural circuits: A decade of progress. *Neuron (Cambridge, Mass.)*, 98(4):865–865, 2018.

[25] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[27] Peter Peduzzi, John Concato, Elizabeth Kemper, Theodore R Holford, and Alvan R Feinstein. A simulation study of the number of events per variable in logistic regression analysis. *Journal of clinical epidemiology*, 49(12):1373–1379, 1996.

[28] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. Past, present and future of spike sorting techniques. *Brain Research Bulletin*, 119:106 – 117, 2015.

[29] Cyrille Rossant, Shabnam N Kadir, Dan F M Goodman, John Schulman, Maximilian L D Hunter, Aman B Saleem, Andres Grosmark, Mariano Belluscio, George H Denfield, Alexander S Ecker, and et al. Spike sorting for large, dense electrode arrays. *Nature Neuroscience*, 19(4):634–641, 2016.

[30] Peter J Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20(C):53–65, 1987.

[31] Neil Schmitzer-Torbert and A. David Redish. Neuronal activity in the rodent dorsal striatum in sequential navigation: Separation of spatial and reward responses on the multiple t task. *Journal of Neurophysiology*, 91(5):2259–2272, 2004.

[32] Patrick Schober, Christa Boer, and Lothar A Schwarte. Correlation coefficients: Appropriate use and interpretation. *Anesthesia and analgesia*, 126(5):1763–1768, 2018.

[33] NAMR Senaviratna and T. Cooray. Diagnosing multicollinearity of logistic regression model. *Asian Journal of Probability and Statistics*, pages 1–9, 10 2019.

[34] Nguyen Thai-Nghe, Zeno Gantner, and Lars Schmidt-Thieme. Cost-sensitive learning methods for imbalanced data. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

[35] M. Wehr, J.S. Pezaris, and M. Sahani. Simultaneous paired intracellular and tetrode recordings for evaluating the performance of spike sorting algorithms. *Neurocomputing*, 26-27:1061–1068, 1999.

[36] F. Wood, M. J. Black, C. Vargas-Irwin, M. Fellows, and J. P. Donoghue. On the variability of manual spike sorting. *IEEE Transactions on Biomedical Engineering*, 51(6):912–918, 2004.

Appendix A

KDE for Raw Data

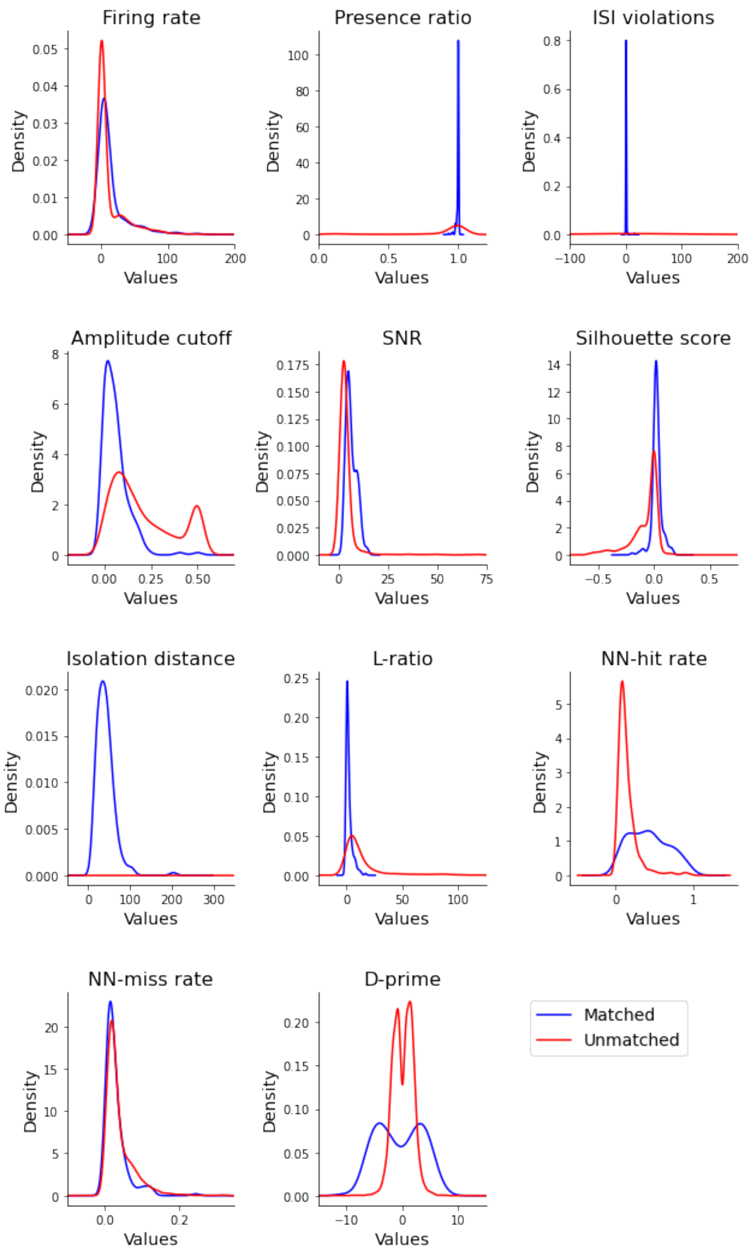

Figure A.1: Kernel density estimation distribution plots for each metric in the raw dataset. Matched (blue) and unmatched (red) populations are represented separately.

Appendix B

VIF scores

Feature	VIF	
	Metrics only case	Metrics and sorter case
Firing rate	1.78	2.30
Presence ratio	3.82	4.55
ISI violations	1.38	1.40
Amplitude cutoff	3.33	3.34
SNR	1.76	1.79
Silhouette score	1.77	1.84
Isolation distance	1.00	1.01
L-ratio	2.04	2.04
NN-hit rate	2.74	3.81
NN-miss rate	2.91	4.38
D-prime	1.01	1.01
MountainSort4		2.74
Ironclust		3.32

Table B.1: Table showing VIF scores for all metrics in the two possible feature sets. All values are below an acceptable threshold of 10 and a stricter threshold of 5.