

# Feature Selection in Highly Redundant Signal Data: A Case Study in Vehicle Telemetry Data and Driver Monitoring

Phillip Taylor<sup>1</sup>, Nathan Griffiths<sup>1</sup>, Abhir Bhalerao<sup>1</sup>,  
Thomas Popham<sup>2</sup>, Xu Zhou<sup>2</sup>, Alain Dunoyer<sup>2</sup>

<sup>1</sup> Department of Computer Science,  
The University of Warwick, UK  
Contact email: phil@dcs.warwick.ac.uk,  
<sup>2</sup> Jaguar Land Rover Research, UK

**Abstract.** Feature sets in many domains often contain many irrelevant and redundant features, both of which have a negative effect on the performance and complexity of agents that use the data [7]. Supervised feature selection aims to overcome this problem by selecting features that are highly related to the class labels, yet unrelated to each other. One proposed technique to select good features with few inter-dependencies is minimal Redundancy Maximal Relevance (mRMR) [10], but this can be impractical with large feature sets. In many situations, features are extracted from signal data such as vehicle telemetry, medical sensors, or financial time-series, and it is possible for feature redundancies to exist both between features extracted from the same signal (intra-signal), and between features extracted from different signals (inter-signal). We propose a two stage selection process to take advantage of these different types of redundancy, considering intra-signal and inter-signal redundancies separately. We illustrate the process on vehicle telemetry signal data collected in a driver distraction monitoring project. We evaluate it using several machine learning algorithms: Random Forest; Naïve Bayes; and C4.5 Decision Tree. Our results show that this two stage process significantly reduces the computation required because of inter-dependency calculations, while having little detrimental effect on the performance of the feature sets produced.

## 1 Introduction

Feature sets in a range of domains often contain numerous irrelevant and redundant features, both of which have a negative effect on the performance and complexity of agents that use the data [7]. Supervised feature selection aims to overcome this problem by selecting features that are highly correlated to the class labels, yet uncorrelated to each other. However, finding redundancy between features is computationally expensive for large feature sets.

In many cases the features themselves are extracted from multiple signal data such as vehicle telemetry [15], medical sensors [13], weather forecasting [9], or

financial time-series analysis [12]. When features are extracted from signal data, it is possible for feature redundancy to be either between features extracted from the same signal (intra-signal), or between features extracted from different signals (inter-signal). In this paper we propose to consider these types of redundancies separately, with the aim of both speeding up the feature selection process and minimizing redundancy in the feature set chosen. We illustrate this two stage process on vehicle telemetry data collected in a driver distraction monitoring project, although the concept generalizes to other domains where signal data is used as the basis for intelligent agents to build prediction models.

The remainder of this paper is structured as follows. In Section 2, we examine current approaches to feature selection and introduce driver monitoring and the issues associated with vehicle telemetry data. In Section 3 we propose a two stage feature selection process aimed at minimizing feature and signal level redundancies, which reduces the computational cost compared to existing methods. Finally, in Section 4, we describe our evaluation strategy and present results for the proposed method alongside results existing techniques.

## 2 Related work

### 2.1 Feature selection

In general, there are two approaches to performing feature selection: wrapper and filter [7]. The wrapper approach generates feature subsets and evaluates their performance using a classifier to measure their discriminating power. Filter approaches treat feature selection as a preprocessing step on the data without classification, thus being agnostic to the classification algorithm that may be used.

The wrapper approach requires a search through the feature space in order to find a feature subset with good performance [7]. The merit of a feature subset is determined by the performance of the learning algorithm using features from that set. Methods for searching through the space of feature combinations include complete, forward, and backward searches. A complete search generates all possible feature subsets in order to find the optimal one, but can be infeasible with more than a few features. Forward selection starts with no selected features and the feature which improves performance the most is then added to the selected features. This is repeated until some stopping criterion is met, such as when performance cannot be improved further, or the required number of features have been selected. Backwards selection begins by selecting all features and then removes those which degrade performance the least. This is repeated until a stopping criterion is met.

With large datasets however, the wrapper approach still requires significant computation in building several classification models, as features are added or removed from the set. Therefore, filter methods, which require considerably less computation, are often preferred. The filter methods can be split into ranking algorithms [6], which consider features independently, and those which consider inter-dependencies and redundancy within the feature sets [11, 1, 10].

This means that the process can be slow on datasets with large numbers of features and samples, as is the case in many domains with signal data. Kira and Rendell proposed the Relief algorithm [6], which was later extended by Kononenko to deal with noisy, incomplete and multi-class datasets [8]. The Relief algorithm repeatedly compares random samples from the dataset with samples that are most similar (one of the same label and one of a different label), to obtain a ranking for feature weights. Although less computationally expensive than wrapper approaches, this still requires searching through the dataset for Near-hit and Near-miss examples. This means that the process can be slow on datasets with large numbers of features and samples, as is the case in many domains with signal data.

Other ranking methods are based on information measures, such as Mutual Information (MI),

$$MI(f_1, f_2) = \sum_{v_1 \in \text{vals}(f_1)} \sum_{v_2 \in \text{vals}(f_2)} p(v_1, v_2) \log_2 \frac{p(v_1, v_2)}{p(v_1)p(v_2)}, \quad (1)$$

where  $f_1$  and  $f_2$  are discrete features,  $p(v_i)$  is the probability of seeing value  $v_i$  in feature  $f_i$ , and  $p(v_i, v_j)$  is the probability of seeing values  $v_i$  and  $v_j$  in the same sample.

Because MI is summed over all the values of a feature, it tends to score features with many values highly. Therefore, normalized variants of MI are often used to remove this bias. A common method of normalization is to divide by the entropy of the features,  $H(f)$ ,

$$H(f) = - \sum_{v \in \text{vals}(f)} p(v) \log_2 p(v). \quad (2)$$

Two variants of MI which normalize by entropy are the Gain Ratio (GR) and Symmetric Uncertainty (SU),

$$GR(f_1, f_2) = \frac{MI(f_1, f_2)}{H(f_1)}, \quad (3)$$

$$SU(f_1, f_2) = 2 \frac{MI(f_1, f_2)}{H(f_1) + H(f_2)}. \quad (4)$$

However, none of these ranking methods consider the issue of feature redundancy. Redundancy is known to cause problems for efficiency, complexity and performance of models or agents that use the data [1, 7]. Therefore, it is important to consider interdependencies between features and remove those which are redundant. Common approaches involve minimizing the inter-dependencies of the selected feature set, as with the concept of minimal redundancy maximum relevance (mRMR), introduced by Ding and Peng in [1, 10]. The relevancy,  $Rel(F, C)$ , of a feature set,  $F$ , is given by the mean MI of the member features and the class labels,  $C$ , namely,

$$Rel(F, C) = \frac{1}{|S|} \sum_{f_i \in S} MI(f_i, C). \quad (5)$$

The redundancy,  $Red(F)$ , of  $F$  is given by the mean of all inter-dependencies as measured by MI:

$$Red(F) = \frac{1}{|F|^2} \sum_{f_i, f_j \in F} MI(f_i, f_j). \quad (6)$$

The difference between relevance and redundancy can be used for mRMR,

$$\max_{S \subset F} Rel(S, C) - Red(S). \quad (7)$$

Other measures, such as ratios, can also be used [1]. In practice, Ding and Peng suggest performing forward selection for mRMR, iteratively selecting the features which satisfy

$$\max_{f \in F \setminus S} Rel(\{f\}, C) - Red(S \cup \{f\}). \quad (8)$$

Here, MI is used as a measure of both relevance and redundancy, and this may again bias towards features with many values. It is therefore possible to use normalized variants of MI, such as GR or SU instead. We will refer to these versions as MImRMR, GRmRMR and SUMRMR depending on whether MI, GR or SU are used as relevance measures respectively.

Other approaches exist, such as the correlation-based feature selector [4], but are outside the scope of this paper, and do not address the issue of redundancy.

## 2.2 Driver monitoring

Driving a vehicle is a safety critical task and demands a high level of attention from the driver. Despite this, modern vehicles have many devices with functions that are not directly related to driving. These devices, such as radio, mobile phones and even internet devices, divert cognitive and physical attention from the primary task of driving safely. In addition to these distractions, the driver may also be distracted for other reasons, such as dealing with an incident on the road or holding a conversation in the car. One possible solution to this distraction problem is for an intelligent agent to limit the functionality of in-car devices if the driver appears to be overloaded. This can take the form, for example, of withholding an incoming phone call or holding back a non-urgent piece of information about traffic or the vehicle.

It is possible for an autonomous agent to infer the level of driver distraction from observations of the vehicle and the driver. Based on these inferences, the agent can determine whether or not to present the driver with new information that might unnecessarily add to their workload. Traditionally, such agents have monitored physiological signals such as heart rate or skin conductance [2, 15]. However, such approaches are not practical for everyday use, as drivers cannot be expected to attach electrodes to themselves before driving. Other systems have used image processing for computing the driver's head position or eye parameters, but these are expensive, and unreliable in poor light conditions.

Therefore, our aim is to use non-intrusive, inexpensive and robust signals, which are already present in vehicles and accessible by the Controller Area Network (CAN) [3]. The CAN is a central bus to which all devices in the vehicle connect and communicate by a broadcast protocol. This allows sensors and actuators to be easily added to the vehicle, enabling agents to receive and process telemetric data from all modules of the car. This bus and protocol also enables the recording of these signals, allowing us to perform offline data mining.

Agents processing data from the CAN-bus on modern vehicles have access to over 1000 signals, such as vehicle and engine speeds, steering wheel angle, and gear position. From this large set of signals, many potential features can be extracted using sliding temporal windows. These include statistical measures such as the mean, minimum or maximum, as well as derivatives, integrals and spectral measures. In [15], Wollmer *et al.* extract a total of 55 statistical features over temporal windows of 3 seconds from 18 signals including steering wheel angle, throttle position and speed, and driver head position. This provides a total of 990 features for assessing online driver distraction. They used the correlation based feature selector as proposed in [4] with SU as a measure of correlation.

### 3 Proposed approach

As previously noted, redundancy in signal data can be considered as either intra-signal, between features extracted from within one signal, or inter-signal, between features extracted from different signals. For instance, in CAN-bus data there is unsurprisingly a large inter-signal redundancy between the features of *Engine Speed* and *Vehicle Speed* signals. This is confirmed by the correlation between the raw values of the signals, of 0.94 in our data. There may also be a high intra-signal feature redundancy, as with the minimum, mean and maximum features. This is particularly the case when the temporal window is small and the signal is slowly varying.

Therefore, we propose a two step procedure to remove these intra-signal and inter-signal redundancies, by considering them separately. In the first stage, feature selection is performed solely with extracted features from individual signals, aiming to remove intra-signal redundancies. In the second stage, selection is performed on these selected features as a whole, removing inter-signal redundancies. This then produces a final feature set with an expected minimal redundancy for an agent to use in learning a prediction model.

This two stage process has benefits with regards to computation. For instance, the forward selection method of mRMR requires a great deal of computation with large feature sets. Moreover, large feature sets, such as those extracted from CAN-bus data, often do not fit into memory in their entirety, meaning that features have to be loaded from disk in sections to be processed. This not only lengthens the feature selection process, but also impacts on the complexity of the implementation. With our two stage process however, smaller numbers of features are considered at a time, meaning that at each stage, these problems do not occur.

Secondary Task	Description
Select Radio Station	Selection of a specified radio station from presets
Mute Radio Volume	The radio is muted or turned off
Number Recall	Recite a 9 digit number provided before the drive
Satellite Navigation Programming	A specified destination is programmed into the in car Sat-Nav
Counting Backwards	Driver counts backwards from 200 in steps of 7 (i.e., 200, 193, 186...)
Adjust Cabin Temperature	Cabin temperature increased by 2°C

Table 1: Secondary tasks the driver was asked to perform. If there is a secondary task being performed, the data is labelled as *Distracted* for the duration, otherwise it is labelled as *Normal*. Tasks were performed in the same order for all experiments, with intervals of between 30 and 300 seconds between tasks.

In using this process, we expect there to be fewer redundancies in the final feature sets because redundancies are removed at both stages. However, returning fewer features in this first stage may reduce the relevance of the selected features to be used in learning. This will particularly be the case when many of the best performing features are from the same signal, but this is assumed to be an extreme and uncommon case.

## 4 Evaluation

### 4.1 CAN-bus data

CAN-bus data was collected during a study where participants drove under both normal and distracted conditions. To impose distraction on the driver, a series of tasks, as listed in Table 1, were performed at different intervals. For the duration of a task, the data is labelled as *Distracted*, otherwise it is labelled as *Normal*. In this study there are 8 participants, each driving for approximately 1.5 hours during which each of the 6 tasks are performed twice. Data was recorded from the 10 signals shown in Table 2 with a sample rate of 10Hz.

In addition to the tasks listed in Table 1, participants performed two driving manoeuvres, namely abrupt acceleration and a bay park. The data from these are, however, considered to be unrelated to distraction and therefore can be viewed as noise and were removed from the dataset. This removal was done after feature extraction to avoid temporal continuity issues.

The features listed in Table 2 are extracted temporally from each signal over sliding windows of sizes 5, 10, 25 and 50 samples (0.5, 1, 2.5, 5 seconds respectively), providing 120 features per signal. This gives a feature set of size  $10 \times 30 \times 4 = 1200$  in total. After feature extraction, the data is sub-sampled temporally by a factor of 10, providing a total of 6732 samples with the label *Distracted*, and 26284 samples with the label *Normal* over 8 datasets. This sub-sampling is done in order to speed up experiments and allow more features to

Signal	Features
Steering wheel angle	Convexity
Steering wheel angle speed	First, Second and Third derivatives
Pedal position	First 5 and Max 5 DFT coefficient magnitudes
Throttle position	Max 5 DFT coefficient frequencies
Absolute throttle position	Entropy
Brake pressure	Fluctuation
Vehicle speed	Gradient: positive, negative or zero
Engine speed	Integral and absolute integral
Yaw rate	Min, Max, Mean and Standard deviation
Gear selected	Number of zero values and Zero crossings

Table 2: The 10 Signals and 40 Features used in the data mining process. Signals are recorded from the CAN-bus at 10Hz. Features are extracted temporally from each signal over sliding windows of 5, 10, 25 and 50 samples (i.e. 0.5, 1, 2.5, 5 seconds). This gives 120 features per signal and a feature set of size 1200.

be selected per signal to go forward to the second stage, which would otherwise be limited due to computational limits.

## 4.2 Experimental set-up

We select features with the MImRMR, GRmRMR and SUMRMR feature selectors. During the first selection stage, 1, 2, 3, 4 and between 5 and 50 in intervals of 5 features are selected from each signal. The second stage outputs a ranking of features from which between 1 and 30 are selected for evaluation. In the cases where 1 and 2 features are selected from each signal, there will be a maximum of 10 and 20 features output from the two stage process respectively. In the evaluations where more than these numbers of features are required, we use all of the available features. The selection algorithm used in the first stage is always the same as the one used in the second stage.

The feature sets are then evaluated using the Naïve Bayes, C4.5 Decision Tree, and Random Forest learning algorithms available in WEKA [14]. In each experiment, after feature selection the data is sub-sampled in a stratified random way by a factor of 10. This gives 673 samples with the label *Distracted*, and 2628 samples with the label *Normal* for each experiment. This random sub-sampling reduces autocorrelation in the data, which would mean evaluations give overly optimistic performance and cause models to overfit. It also introduces some randomness into the evaluations, meaning that they can be run multiple times to gain a more accurate result. All evaluations are run 10 times, each giving an area under the receiver operator characteristic (ROC) curve (AUC).

In each evaluation, a cross folds approach is used, where training is performed on seven datasets and testing on the final one for each fold. These are averaged to give a performance metric for each feature selection procedure. A higher AUC value with smaller number of features indicates a good feature selector.

### 4.3 Results

First we present performance of our two stage process with the selection of varying numbers of features from each signal in the first stage. Second, we show computation times for our two stage selection process and compare these with selecting all of the features at once. Finally, we compare the performance of our two stage selection against selecting features without our two stage process.

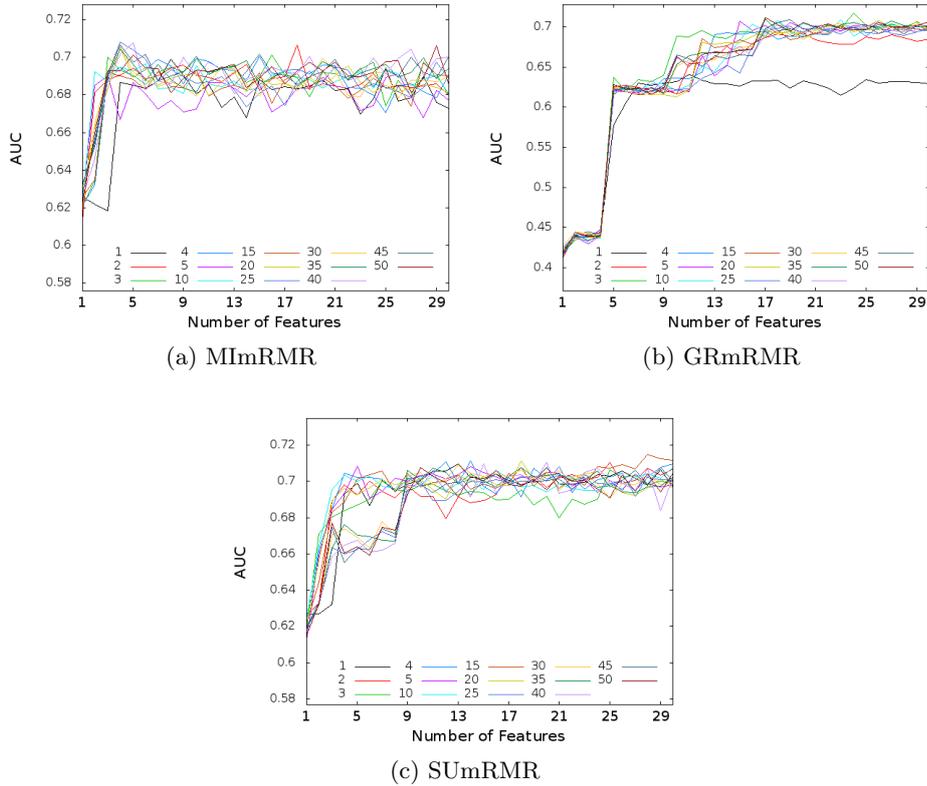


Fig. 1: AUC values for (a) MImRMR, (b) GRmRMR and (c) SUmRMR using a Naïve Bayes classifier for different numbers of features being selected after the two stage process. Each line represents a different number of features selected per signal in the first stage. In most cases, performance is unaffected by the number of features selected per signal. However, in some cases where small numbers of features are selected, performance can be worse.

Figure 1 shows AUC values for different feature set sizes, selected using the two stage selection process. Each line represents a different number of features

per signal selected in the first stage. From these results we can make three observations about the two stage selection process. First, learning with more features provides better performance. Second, with more than five features, there is little further performance gain. Finally, in most cases, performance is unaffected by the number of features selected per signal. However, in some cases where small numbers of features are used for learning, worse performance is observed when small numbers of features are selected per signal. Also, when using GRmRMR we can see that selecting 1 feature per signal is not sufficient for maximal performance. This is possibly because GRmRMR is not selecting the best feature from each signal first, and therefore produces a sub-optimal feature set.

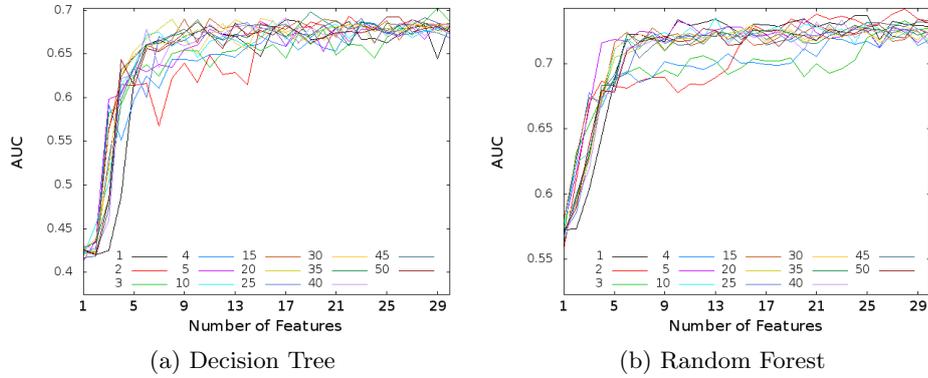


Fig. 2: AUC values for MImRMR using the (a) Decision Tree and (b) Random Forest classifiers for different numbers of features being selected after the two stage process. Each line represents a different number of features selected per signal in the first stage. We see the same results as with the Naïve Bayes learner, that there is little difference in performance in most cases.

The same results are also seen with other learning algorithms. In Figure 2, the AUC values are shown for the C4.5 Decision Tree and Random Forest classification algorithms built with features selected by MImRMR using our two stage process. One small difference here is that the Decision Tree performs much worse than Naïve Bayes and Random Forest with very few features, but still achieves comparable performance when 10 features are used for learning.

Second, we show that, while selecting features that have the same performance, computation times are substantially reduced. In Table 3, computation times for selecting 30 features using our two stage process are presented. Denoted in parentheses are the speed-ups when compared to selecting from the total 1200 features without using the two stage process. Selecting more than 15 features per signal does not provide any significant speed-up. Selecting 1 or 2 features per signal in the first stage provides speed-ups of over 30x, however, this

is not advised due to the performance results presented above. Selecting between 3 and 5 features provides equivalent performance to other methods investigated, and gives a speed-up, of over 10x for GRmRMR and SUmRMR. The smaller speed-ups for MImRMR are likely due to a simpler selection process, as multiple entropies do not have to be computed.

Features/Signal	MImRMR	GRmRMR	SUmRMR
1	2.6 (16.4x)	5.0 (73.2x)	5.1 (71.7x)
2	4.2 (9.9x)	11.3 (32.6x)	11.6 (31.6x)
3	6.1 (6.9x)	19.1 (19.3x)	19.4 (18.9x)
4	8.4 (5.0x)	25.6 (14.3x)	25.8 (14.2x)
5	9.9 (4.2x)	32.3 (11.4x)	33.0 (11.1x)
10	18.7 (2.2x)	67.6 (5.4x)	66.7 (5.5x)
15	26.5 (1.6x)	102.4 (3.6x)	102.3 (3.6x)
20	35.4 (1.2x)	144.2 (2.5x)	143.8 (2.5x)
25	43.3 (1.0x)	186.3 (2.0x)	186.0 (2.0x)
30	52.7 (0.8x)	236.1 (1.6x)	232.1 (1.6x)

Table 3: Computation times in seconds for selecting 30 features from 1200 using the two stage process with varying features per signal. Denoted in parentheses are the speed-ups, which are computed with respect to selecting 30 features from 1200 as a whole. We see that selecting more than 15 features per signal does not provide significant speed-up. However, as the performance results show, only a small number of features is required for equivalent performance. In these cases we see a much larger speed-up, of around 10x.

It is worth noting that these timings do not include reading the data from disk. Obviously, with larger feature sets, reading the data requires more time. However, in the two stage process IO times would continue to increase linearly with the number of features. This is because we consider each signal separately and only select a small number of features from each, meaning the features brought forward to the second stage are likely to fit in memory. Without this process, the increase in IO times may be more than linear because redundancies between all features are required. It is unlikely that all features will fit in memory, meaning that the data would have to be loaded in chunks, with each chunk being loaded multiple times. Therefore, the speed-ups we present here are likely to be optimistic for very large datasets.

Finally, we compare the performance of mRMR with and without our two stage feature selection process. The redundancies calculated using Equation 6 for selecting between 1 and 30 features with MImRMR are shown in Figure 3. When selecting only 1 or 2 features per signal, the redundancy increases to 0.57 and 0.16 respectively. This is likely because, as previously mentioned, several of the signals have a high correlation, meaning that features extracted from them will also be similar. This means that in the second stage of selection, there

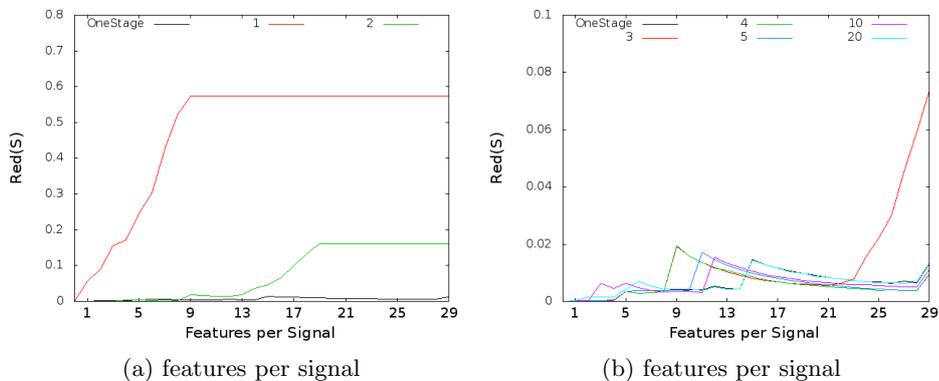


Fig. 3: Redundancies of feature sets produced when selected using MImRMR, with and without two stage selection. With 1 or 2 features per signal, the redundancy is high in the returned feature set. As more features per signal are used, the redundancies become indistinguishable from when selecting features in one stage (OneStage).

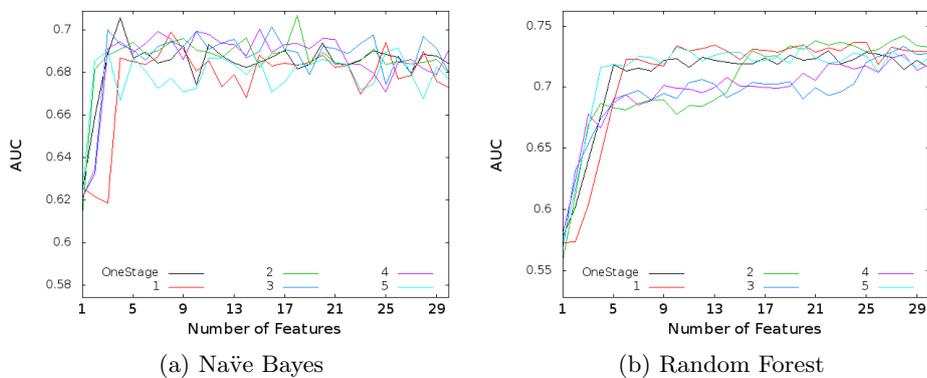


Fig. 4: AUC values for MImRMR using the (a) Naïve Bayes and (b) Random Forest classifiers for different numbers of features being selected. These results compare our two stage process (selecting 1, 2, 3 and 5 features per signal), with selecting features in one stage (OneStage). We again see very similar performance for the two methods.

are very few non-redundant features to select from, and a feature set with high redundancy is produced. As more features are selected per signal, the redundancy begins to resemble that of when two stage selection is not used, until 20 features are selected per signal, when the redundancies are indistinguishable. In Figure 4, the AUC values are again very similar, even when selecting a small number of features from each signal. For example, there is almost no difference in AUC between selecting with the two stage process with 5 features per signal, and selecting without it. Therefore, this equivalent performance and the speed-ups with this number of features, make it beneficial to use the two stage selection process.

## 5 Conclusions

In this paper we have investigated a two stage selection process to speed up feature selection with signal data. We evaluated this process on vehicle telemetry data for driver monitoring, and have shown that the process provides a computational speed-up of over 10x while producing the same results. It follows then, that it is worthwhile to consider features extracted from each signal before considering them as a whole. Furthermore, the first stage of our process can easily be parallelized, selecting features from each signal in a different thread, which would provide further speed-ups.

In future work, we intend to inspect the feature rankings after selection. This will provide insight into the features that are selected, rather than merely their performance and redundancy. Also, it will highlight any instabilities in the feature sets produced by the two stage process, which could harm performance [5].

In this paper, we have assumed that each signal has an equal probability of producing a good feature. This may not be the case, as some signals may have many high performing features whereas others may have none. Second, this assumption means that selecting 5 features from 1000 signals produces a total of 5000 features for selection in the second stage. Therefore, selection of *signals* may be necessary before any feature selection in order to gain a further speed-up.

Finally, although it is unlikely given the size of the data we have used, it is possible that the feature selection methods are overfitting to the data [7]. In future we will evaluate the feature selector algorithms on a hold-out set, not included during the feature selection process, to avoid this.

## References

1. C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, 3(2), 2003.
2. Y. Dong, Z. Hu, K. Uchimura, and N. Murayama. Driver inattention monitoring system for intelligent vehicles: A review. *IEEE Transactions on Intelligent Transportation Systems*, 12(2), 2011.

3. M. Farsi, K. Ratcliff, and M. Barbosa. An overview of controller area network. *Computing Control Engineering Journal*, 10(3), 1999.
4. M. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, Department of Computer Science, The University of Waikato, 1999.
5. K. Iswandy and A. Koenig. Towards effective unbiased automated feature selection. In *Proceedings of the Sixth International Conference on Hybrid Intelligent Systems*, 2006.
6. K. Kira and L. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the National Conference on Artificial Intelligence*, 1992.
7. R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2), 1997.
8. I Kononenko. Estimating attributes: analysis and extensions of relief. In *Proceedings of the European conference on Machine Learning*, 1994.
9. S Mathur, A Kumar, and M Chandra. A feature based neural network model for weather forecasting. *International Journal of Computational Intelligence*, 34, 2007.
10. H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8), 2005.
11. S. Saleh and Y. Sonbaty. A feature selection algorithm with redundancy reduction for text classification. In *22nd International Symposium on Computer and Information Sciences*, 2007.
12. R. Tsay. *Analysis of Financial Time Series (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2nd edition, 2005.
13. A. Wegener. Compression of medical sensor data [exploratory dsp]. *IEEE Signal Processing Magazine*, 27(4), 2010.
14. I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
15. M. Wollmer, C. Blaschke, T. Schindl, B. Schuller, B. Farber, S. Mayer, and B. Tröfflich. Online driver distraction detection using long short-term memory. *IEEE Transactions on Intelligent Transportation Systems*, 12(2), 2011.