Data mining and compression: where to apply it and what are the effects?

Phillip Taylor
phillip.taylor@warwick.ac.uk
University of Warwick
Coventry, UK

Nathan Griffiths
nathan.griffiths@warwick.ac.uk
The University of Warwick
Coventry, UK

Zhou Xu
xzhou1@jaguarlandrover.com
Jaguar & Land Rover Research
Coventry, UK

Alex Mouzakitis
amouzak1@jaguarlandrover.com
Jaguar & Land Rover Research
Coventry, UK

ABSTRACT

In data mining it is important for any transforms made to training data to be replicated on evaluation or deployment data. If they is not, the model may perform poorly or be unable to accept the input. Lossy data compression has other considerations, however, for example it may not be known whether or not lossy compression will be applied to deployment data, or if a variable compression ratio is to be used. Furthermore, lossy data compression typically reduces noise, which may not affect or even improve model performances, and performing feature selection on lossy data may find better features than selecting from the original data. In this paper, we investigate the effects of selecting features, learning, and making predictions from data that has been compressed using lossy transforms. Using vehicle telemetry data, we determine where in the data mining methodology lossy compression is detrimental or beneficial, and how it should be compressed. We also propose a specialised feature selection approach that considers predictive performance alongside compressibility, measured by compressing them either individually or in a single concatenated stream.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Theory of computation** → **Data compression**; • **Computing methodologies** → **Feature selection**; *Supervised learning by regression.*

KEYWORDS

Data mining, Data compression, Data compression

ACM Reference Format:
Phillip Taylor, Nathan Griffiths, Zhou Xu, and Alex Mouzakitis. 2018. Data mining and compression: where to apply it and what are the effects?. In *Proceedings of The 8th SIGKDD International Workshop on Urban Computing (UrbCom '19).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

UrbCom '19, August 05, 2019, Anchorage, AK

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Data mining is a process that turns data into patterns that describe a part of its structure [2, 9, 23]. For example, a city may wish to estimate the likelihood of traffic congestion or assess air pollution, using data collected from sensors on a road network. The data mining methodology [12] defines a series of activities where data is analysed and models are learned. Broadly, it comprises a problem specification (where a concrete task is defined), data collection, data engineering (data preprocessing such as re-sampling, discretisation, and feature selection), algorithm engineering (fitting models and optimising hyper-parameters), and evaluation (assessing the model performance). Finally, refinement ensures each stage is revisited so that design decisions are reconsidered as a continuous process both before and after deploying the learned model.

It is generally accepted that any transform to training data should also be applied to testing data, or data input after the model is deployed [23]. For example, if the training data is normalised or discretised before learning a model, the same normalisation or discretisation scheme should be applied to any data input into the model. If it is not, the model may perform poorly or be unable to accept the input. In lossy data compression, however, there are other considerations. Firstly, it may be unknown whether data in deployment will be compressed using lossy techniques, in particular if variable compression is to be used. Secondly, information loss due to a lossy transform is often limited to signal noise, which may not affect or even improve model performances, even if models are trained on the original data. Finally, performing feature selection on lossy data may find features that are more robust to lossy compression than selecting features from the original data.

In this paper, we investigate the effects of selecting features, learning, and making predictions from data that has been compressed using lossy transforms. Specifically, we

- determine whether or not compression is improved by compressing signals separately or together,
- evaluate the trade-off in performance and compression when using lossy versus lossless compression, and
- discuss considerations to make when learned models are intended to take lossy compressed data as input.

As well as selecting features from lossy compressed data, the amount of compression achieved is also considered alongside their predictive performances in the selection process, using the Minimal

Redundancy Maximal Relevance and Compression (MRMR+C) filter [21]. In doing so, features with slightly worse performances but significantly improved compression can be selected.

The remainder of this paper is structured as follows. Section 2 discusses the related work on data compression and feature selection. Section 3 outlines the data mining methodology with consideration of lossy compression, and Section 4 presents a feature selection approach that considers four different measures of compressibility. Section 5 then evaluates the feature selection approaches and compares the performances of considering compression at the different stages of the methodology. Finally, Section 6 concludes this paper.

2 RELATED WORK

Data compression aims to remove redundancy, thus making the data representation smaller, in such a way that it can be restored [18]. Characteristics of urban and vehicle telemetry data, including temporal consistency, noise, and signal redundancy, all support good compression. While signal redundancy is typically removed when performing feature selection, noise and temporal consistency must be considered using other kinds of data compression. There are two broad categories of data compression, namely lossless and lossy.

Lossless compression aims to compress the data in such a way that the uncompressed version is indistinguishable from the original [18]. Typically, lossless compression inspects the frequencies of symbols, and looks for repeating symbols or sequences of symbols in the data stream. Perhaps the most simple method of compression is runlength encoding, in which symbols are encoded along with their number of consecutive repetitions. For example, the string 'AAAABBA' can be encoded as 'A4B2A1'. Using this simple method can increase the length of the string, however, as in the case of 'ABAB', which would be encoded as 'A1B1A1B1'.

Two other notable compression algorithms are LZ77 dictionary encoding [24] and Huffman coding [11]. LZ77 uses a sliding window and searches for repeating sequences, which are encoded as the length and location of its first occurrence in the window. Huffman coding produces a variable length prefix-code defining the path to the encoded symbol in a Huffman tree. Symbols that occur with higher frequencies are located closer to the root node in the tree, and thus have shorter Huffman codes. Taken together, LZ77 and Huffman encoding make up the DEFLATE compression algorithm [5], which is the basis of the ZIP file format.

Lossy compression relaxes the guarantee that the decompressed stream is the same as the original, and aims only to minimise information loss. In particular, lossy compression aims to keep information where it is important and lose information where its loss will not be noticed. In JPEG image compression, for example, high frequency information that is difficult for humans to perceive is removed. For vehicle telemetry data, similar components of the signals can be removed if they are not useful to further analysis. Some signals such as vehicle speed, for example, contain noise that may even be detrimental to analyses.

JPEG compression is based on the discrete cosine transformation, which transforms the signal into the the frequency domain in the the form of cosine functions. The high frequency cosine functions, or those with small amplitudes, can be ignored and removed using quantisation without noticeably affecting the quality of the image while significantly improving compression. Another lossy compression method is to use the Discrete Wavelet Transform (DWT), which operates by extracting two signals that are each half the length of the original [1]. The first represents an approximation of the original signal, and is referred to as the low frequency component. The second is the high frequency component, and is a representation of the detail in the original signal. The low and high frequency components are produced by a convolution of the original signal with wavelet kernels, followed by a down-sampling by a factor of two. Typically the high frequency component contains many small values and, as with the discrete cosine transform, can be considered the noise component of the original signal and removed using quantisation. By quantising the high frequency component, many of these small values become zero and can be encoded very efficiently using lossless compression methods such as DEFLATE. This process can be performed recursively to the low frequency component, further increasing the potential for lossless compression of the coefficients. Quantisation introduces errors into the signal when it is reconstructed, but the error is minimised because only the high frequency component (detail) is typically affected.

One form of data compression often applied in data mining is feature selection, which reduces the data that must be processed by machine learning algorithms and models. This has several advantages, including reducing model complexity and improving performance, but selected features often have high variances and entropies that typically coincide with good predictive performance [8]. This bias toward high entropy features limits the data compression that can be achieved with those that are selected. It may be the case, particularly with the high redundancies found with vehicle telemetry and urban data, that some features with lower entropies and better compression may provide comparable predictive performances [21].

Feature selection aims to find a subset of all possible features to reduce their number, while still sufficiently describing the data with respect to a particular task [4]. In unsupervised learning a task may be to group data samples in an efficient way, or in a supervised setting it may be to predict a given target variable. Feature selection for unsupervised learning typically aims to find the features that best capture differences between samples [7, 16]. This typically relies on transforming the feature space, assessing clustering properties, or other discriminatory properties measured using heuristics [3].

In supervised learning, which is the focus of this paper, the three main approaches are embedded, wrapper, and filter methods [8]. Embedded methods perform feature selection as part of the learning algorithm [15]. For example, in decision tree induction, the choice of variable on which to split nodes can be seen as feature selection. The nodes used, and their associated features, are then the selected features. Trees in random forests may also be removed if their estimated performance is poor, which may mean poor performing features are never used in the learned model.

In wrapper methods, feature subsets are assessed by estimating the performances of models that use them as inputs [13]. This is done by comparing the performances of models learned using the same process and training samples, but with different sets of input features. Wrapper methods are typically computationally expensive, as they require a machine learning algorithm to build a model for each feature subset evaluation. Filter methods, on the

other hand, generally assess the performances of feature subsets using heuristics that are typically less computationally expensive. Some commonly used heuristics include similarity measures such as Pearson's Correlation Coefficient (PCC) or Mutual Information (MI), which can be used to estimate both relevance and redundancy of a feature set [23].

In general, selecting up to l features from a set of features, X, can be represented as an optimisation problem [14],

$$\underset{S \subseteq X, |S| \leq l, l \leq |X|}{\operatorname{argmax}} \quad P(S, y), \tag{1}$$

where $P(S, y)$ estimates the performance of a subset of these features, S, with respect to predicting the target variable, y. Ideally, all possible subsets would be evaluated to find the feature set that provides the highest performance, but the number of possible feature subsets is $\binom{|X|}{l}$ and this exhaustive search is infeasible. In practice, therefore, a more efficient combinatorial search algorithm is applied.

Possibly the most common search strategy is the forward greedy search [14], which selects the feature, $s_k \in X \setminus S$, that satisfies a condition,

$$\underset{s_k \in X \setminus S}{\operatorname{argmax}} \; P(S_k \cup \{s_k\}, y) - P(S, y), \tag{2}$$

in each of $k = 1, 2, \ldots, l$ iterations. The search begins with no selected features, $S_0 = \emptyset$, to which the feature with the highest individual performance is selected. It is typical that during this first iteration $P(\emptyset, y) = 0$. In the subsequent $l - 1$ iterations, $P(S_k, y)$ is the performance of the already selected features, and the feature which increases the performance most is selected. The search stops when a stopping criteria is met, such as when a given number of features are selected or if the performance score decreases after selecting a new feature.

3 DATA MINING METHODOLOGY

In this section we introduce the consideration of lossy compression at different stages of the data mining methodology [12], namely before feature selection, training a predictive model (learning), and making predictions with the learned model (evaluation or deployment). A simplified data mining methodology presented in Figure 1, describing the feature selection, model learning, and evaluation stages. These stages are a subset of the wider data mining methodology, which includes data preprocessing, feature extraction, refinement, and deployment.

In the feature selection stage, training data is input into a feature selection algorithm, which outputs the feature identifiers of those feature that are selected. The selected feature identifiers are then used to get the feature subset from the training data and produce the training features in the model learning stage, where a machine learning algorithm is applied to fit a predictive model to the training features. Finally, the selected features in the testing data are then input into learned model in the evaluation stage, which produces some performance metrics via the model evaluation.

In each stage of Figure 1 there is potential for lossy data compression to be considered, highlighted in red. In each case there is a boolean parameter, namely SC for feature selection, LC for model learning, and EC for evaluation. If this parameter is true, the features are compressed and decompressed using lossy data compression before they are used, leading to some information loss that affects the following processes. In the feature selection stage the lossy data compression is applied prior to feature selection, in the model learning stage it is applied prior to learning the model, and in evaluation it is applied before the features are input into the learned model.

In applying data compression, a lossy transform is applied to each feature followed by reconstructing them using the inverse transform. Lossy transforms are typically used to represent the signal in such a way that the lossless compression requires fewer bytes to represent it, usually because it consists of several consecutive zeros that can be run-length encoded. Although the lossy transform changes the signal, the information loss is usually limited to the detail or noise of the signal. Using DWT, for example, coefficients in the detail components are typically small, and those below a threshold are set to zero. The approximation coefficients typically have larger values and are not set to zero. Combining the approximation and detail coefficients that are not changed, an accurate reconstruction of the original signal can be generated.

In this paper, we measure signal reconstruction accuracy using an adaptation of Symmetric Mean Absolute Percentage Error (SMAPE),

$$RA(a, \hat{a}) = 1 - \frac{1}{n} \sum_{i=0}^{n-1} \frac{|a_i - \hat{a_i}|}{|a_i| + |\hat{a_i}| + 1}, \tag{3}$$

where 1 is added to the denominator to avoid division by zero and it is subtracted from 1 to produce an accuracy rather than error measure. $RA(a, \hat{a}) = 1$ when the original signal, a, and reconstructed signal, $\hat{a}$, are identical and $RA(a, \hat{a}) < 1$ when they differ. In the results of this paper, we ensure the same accuracy is maintained across the different features, and search for the threshold that achieves the minimum accuracy that is greater than a specified reconstruction accuracy.

4 FEATURE SELECTION

MRMR+C [21] is based on the Minimal Redundancy Maximal Relevance (MRMR) feature selection filter, which assesses relevancy and redundancy to produce a performance measure for a feature set [6, 17]. MRMR has several different instantiations, defined by how relevance and redundancy is assessed, as well as how they are combined [10]. In general, the performance score in MRMR increases with higher relevance and decreases with higher redundancies. One such instantiation defines the performance of a feature set as the difference between relevance and redundancy,

$$P_{\text{MRMR}}(S, y) = Rel(S, y) - Red(S), \tag{4}$$

where $Rel(S, y)$ is the relevancy of feature set, S, to target, y, and $Red(S)$ is its redundancy.

To assess the relevance of a feature set, the individual feature relevancies, $\rho(x_i, y)$, can be aggregated,

$$Rel(S, y) = \frac{1}{|S|} \sum_{x_i \in S} \rho(x_i, y). \tag{5}$$

We do not specify the similarity measure, $\rho(\cdot)$, which may be instantiated using any correlation measure [19, 20], such as PCC or MI [23]. For the simulations in this paper we use PCC due to its computational efficiency.

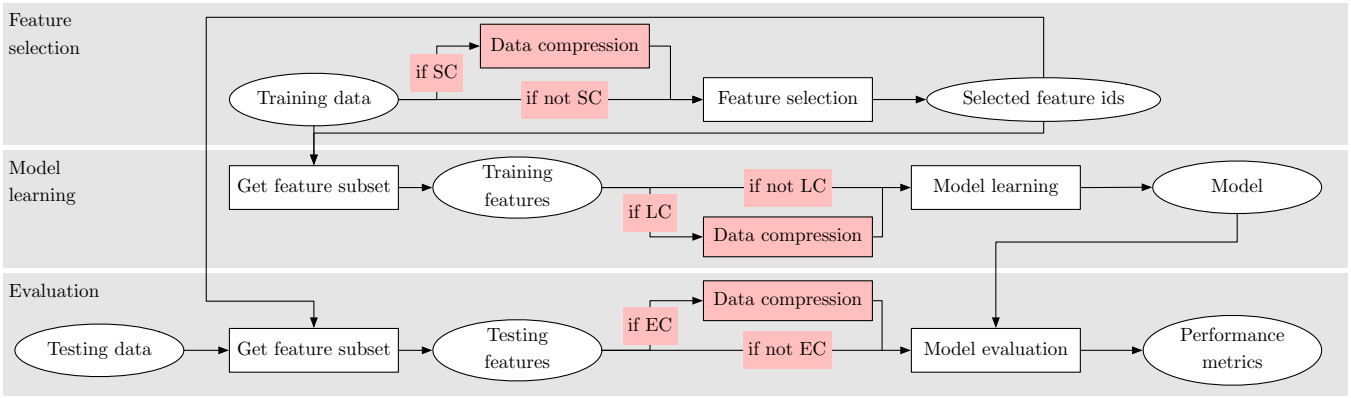


Figure 1: Data mining methodology with compression.

The redundancy of a feature set can be assessed as the mean of all pairwise feature similarities,

$$Red(S) = \frac{1}{|S-1|^2} \sum_{\substack{x_i, x_j \in S^2, \\ x_i \neq x_j}} \rho(x_i, x_j). \tag{6}$$

To maximise the compression of selected features, we use a compressibility factor, $Com(S)$, which increases with the compressibility of the features. In particular we apply data compression to the features and measure the space savings achieved,

$$Com(S, S') = 1 - \frac{B(S')}{B(S)}, \tag{7}$$

where $B(S)$ is the number of bytes in the original data S, and $B(S')$ is the number of bytes in the compressed data, S'. This measure is larger for variables that are more easily compressed than those that do not compress well, and has a range of $[0, 1]$ whenever good compression is achieved.

We propose two different forms of compression, one where the features are compressed individually (depicted for m features in Figure 2(a)) and another where they are compressed together (depicted for m features in Figure 2(b)). To compress individual features, lossless compression, $LLC(\cdot)$, is applied to each one separately and the compressed outputs are combined (using concatenation) in a single stream, $\{LLC(x) : \forall x \in S\}$. In this paper we employ the DEFLATE lossless compression algorithm. When compressing the features together, they are concatenated into a single data stream to which lossless compression, $LLC(S)$, is applied to compress the data.

It is also possible to apply a lossy transform, $LT(\cdot)$, to each feature, x, prior to lossless compression. In many cases this lossy transform will improve compression, but in others it may increase the number of bytes required to represent the data. Therefore, we use an option function,

$$LTO(x) = \begin{cases} LT(x) & \text{if } B(LLC(LT(x)) < B(LLC(x)) \\ x & \text{otherwise,} \end{cases} \tag{8}$$

so the lossy transform is only used when it improves compression and the best compression is achieved. If compression is not improved, lossless compression is applied directly to the feature values rather than its lossy transform. Clearly, if the lossy transform is not applied to a signal, its reconstruction accuracy will always be 1.

In this paper we employ the DWT lossy transform, using 'HAAR' wavelets with three levels. A DWT threshold is computed for each feature in the training data such that it maintains a given reconstruction accuracy, RA, (as defined in Equation 3 and discussed in Section 3). RA is therefore a parameter of the data mining methodology used to find a DWT threshold. The DWT threshold is then used in data compression of training data in feature selection stage (when SC=1) and the model learning stage (when LC=1), and applied to features the testing data in the evaluation stage (when EC=1).

The performance metric in MRMR+C is then defined as,

$$P_{\text{MRMR+CI}} = P_{\text{MRMR}}(S, y) + \omega_{com} \times Com(S, CI(S)), \tag{9}$$

where

$$CI(S) = \{LLC(LTO(x)) : \forall x \in S\} \tag{10}$$

if compressing them individually (Figure 2(a)), and

$$P_{\text{MRMR+CT}} = P_{\text{MRMR}}(S, y) + \omega_{com} \times Com(S, CT(S)), \tag{11}$$

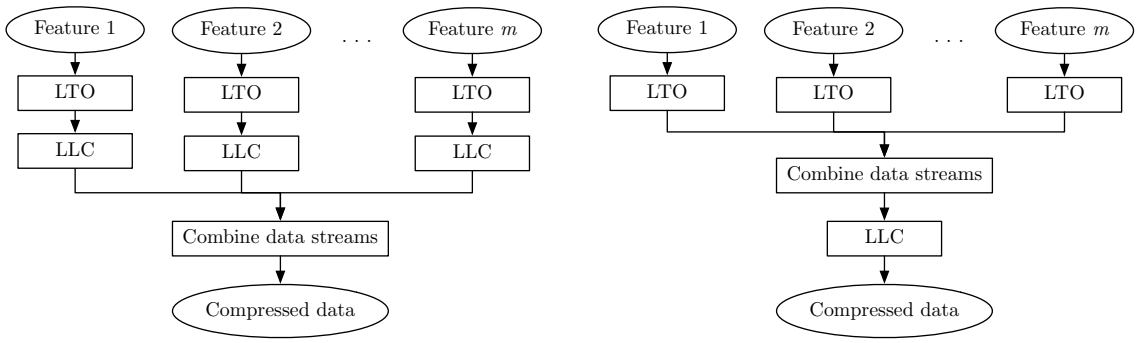
where

$$CT(S) = LLC(\{LTO(x) : \forall x \in S\}) \tag{12}$$

if compressing them in one stream (Figure 2(b)).

5 RESULTS

In this section we demonstrate the methodology and feature selection described in Sections 3 and 4 using two datasets. First, we use the Location Extraction Dataset (LED) [22], which consists of over 1900 signals collected in nine structured journeys, each repeated eight times (for a total of 72 journeys), while performing various pick-up and drop-off scenarios in an urban environment. The mean length of each journey was 19.7 minutes, the standard deviation of journey lengths was 8.2 minutes, and the range was 29.1 minutes. Second, the Pattern of Life dataset (POL) is used, which was collected using seven different drivers who had access to the vehicle for personal use, and as such the journeys were unstructured. In this paper we use ten randomly selected journeys of at least twenty minutes for each driver, providing a mean journey length of 50 minutes. The longest journey length was 130 minutes and the standard deviation was 24.1 minutes.



(a) Individual feature compression **(b) Combined feature compression**

Figure 2: Flow diagram of (a) individual and (b) combined feature compression.

All signals in both the LED and POL were sampled at a constant frequency of 10hz, and interpolated using the last observed value. Before applying the data mining methodology in Figure 1, we also applied feature extraction. The signals were split temporally into equally sized blocks of 10 samples (1 second), from which the mean (if continuous) or modal (if categorical) values were extracted. This block size was chosen as it provided the most interesting trade-off for predictive performance and feature compression.

We use a regression task to demonstrate the methodology, namely estimating the instantaneous fuel consumption. All signals containing the string 'fuel' and 'torq' were removed from the data, because these signals can be used to achieve high performances in simple models using one feature. Signals with names containing the strings 'time' and 'minutes' were also removed prior to any feature selection or model learning, as they were found to be detrimental to the results due to each sample having a unique value.

Features were selected using MRMR+C, as defined by Equation 9 (when compressing features individually) and Equation 11 (when compressing them as one stream). In either case, forward greedy search (defined by Equation 2) was used to select fifteen features from the datasets. Support vector machines were then learned and evaluated using features selected at each of the fifteen stages of the forward greedy search, and the performance of the best model was then recorded. All performance results presented are therefore the best performances when selecting up to fifteen features.

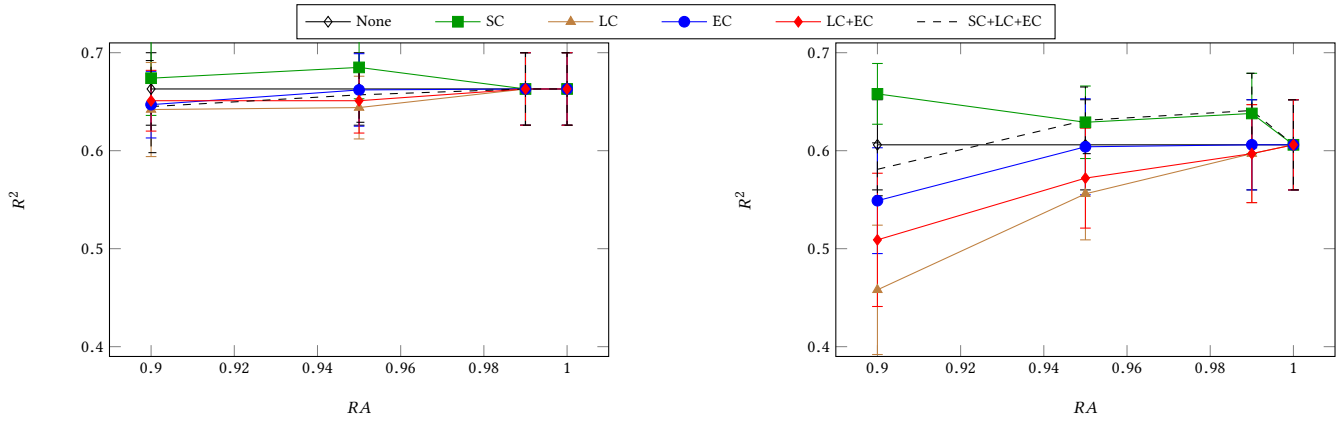
For both datasets we use a form of k-folds cross validation, where data collected from one scenario (for LED) or driver (for POL) was taken as the training data, and remaining data from the other scenarios or drivers as the testing data. For LED, therefore, the results are presented as a mean over nine folds, each with training data taken from the eight journeys of one scenario and the remaining 63 journeys as the testing data. For POL, the ten journeys from one driver were used as the training data in each of the seven folds, and the remaining 60 journeys of the other drivers as the testing data. The results presented are then the mean over seven iterations of the methodology (where the journeys from each driver is used as training data). The error bars show the standard errors.

Figure 3 shows the R^2 scores for the (a) LED and (b) POL, when features were compressed using a lossy transform at different points in the data mining methodology (in Figure 1) and with different values of RA. All results in this figure were produced by selecting features with $\omega_{com} = 0$ (i.e., MRMR was used in feature selection). Clearly, performing no lossy compression at any stage during the methodology (SC=LC=EC=0) meant that there was no change in performance for different values of RA. All compression strategies also had the same performances when the compression applied to features was lossless ($RA = 1$). For the POL, performing lossy compression at all stages (SC=LC=EC=1) had the best performances, in particular when $0.9 < RA < 1$. With the LED, however, the R^2 performances of SC=LC=EC=1 reduced as RA decreased, indicating that there may be less noise in this dataset as a result of the structured scenarios. In both datasets, performances were also improved slightly by applying lossy compression to features at only the feature selection stage (SC=1).

The respective MAE performances, which mirror the R^2 scores, are shown in Figure 4. The MAEs achieved for the LED were significantly lower than those for POL, even though the R^2 performances of POL were slightly higher. This is likely due to the greater variance caused by the unstructured driving by various drivers in POL, as opposed to the scenarios performed by a single driver in LED.

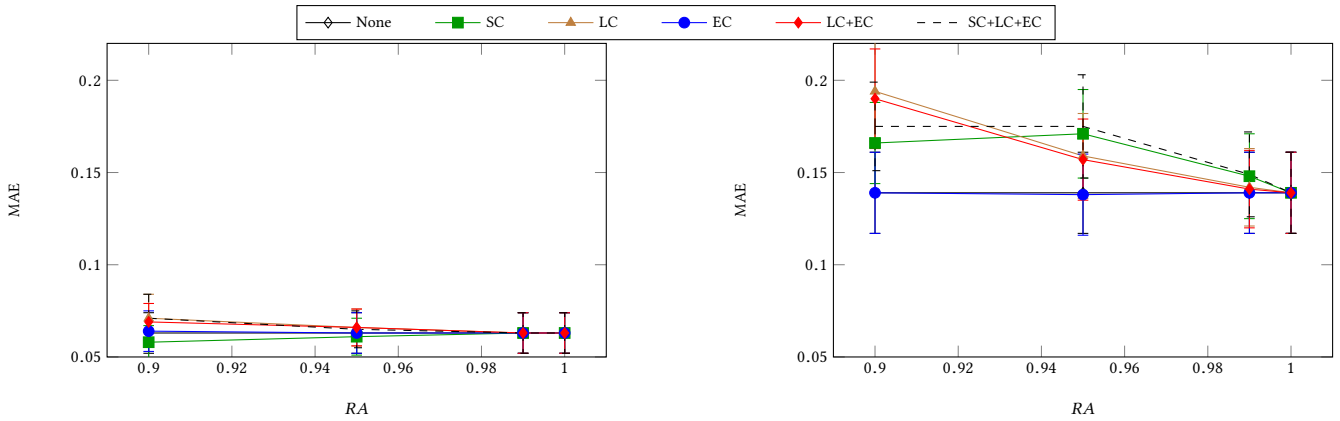
Performing lossy compression at either the model learning or evaluation stages (i.e. SC=0, and LC=1 or EC=1) significantly reduced R^2 performances and increased MAE as RA decreased. The worst performances in both the LED and POL were achieved by applying lossy compression to only the training data, after feature selection (i.e. LC=1 and SC=EC=0). Poor performances were also achieved when lossy compression was applied to only the testing data (i.e. EC=1 and SC=LC=0), or to the training and testing data but not during selection (i.e. LC=EC=1 and SC=0). This is likely because features that are sensitive to lossy compression are being selected when their original lossless versions are used to assess their predictive performances. When these features are compressed using a lossy transform after selection, the information loss caused them to be poor predictors of the target variable. It is clear, therefore, that for good performances, information loss due to compression must be considered when assessing the predictive performances of features during feature selection.

The R^2 performances for the (a) LED and (b) POL for the different instantiations of MRMR+C (Equations 9 and 11) and different values of ω_{com}, are shown in Figure 5. The respective MAE performances



(a) LED **(b) POL**

Figure 3: R^2 performances when compressing at different stages of the data mining methodology for (a) LED and (b) POL. The legend key shows which boolean values are true in the methodology (e.g. LC+EC means lossy compression was performed at the model learning and evaluation stages).



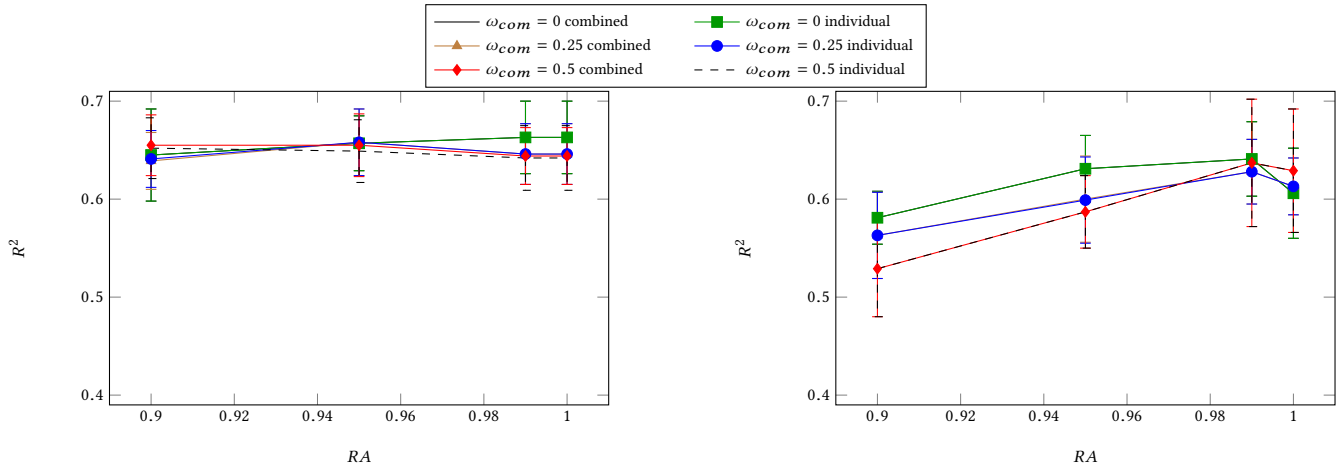
(a) LED **(b) POL**

Figure 4: MAE performances when compressing at different stages of the data mining methodology for (a) LED and (b) POL. The legend key shows which boolean values are true in the methodology (e.g. LC+EC means lossy compression was performed at the model learning and evaluation stages).

are shown in Figure 6, which again mirror the R^2 scores. In all cases, compression was applied at all stages in the data mining methodology (i.e. SC=LC=EC=1), with different reconstruction accuracies, *RA*. In general, the performances were unaffected by compressing features independently or combined to assess their compressibility. For the same value of ω_{com}, the relevant performances were almost identical. Performances were, however, affected by changing the value of ω_{com} in both datasets. In particular, larger values of ω_{com} led to worse predictive performances in most cases, reflecting the results of Taylor et al. [21].

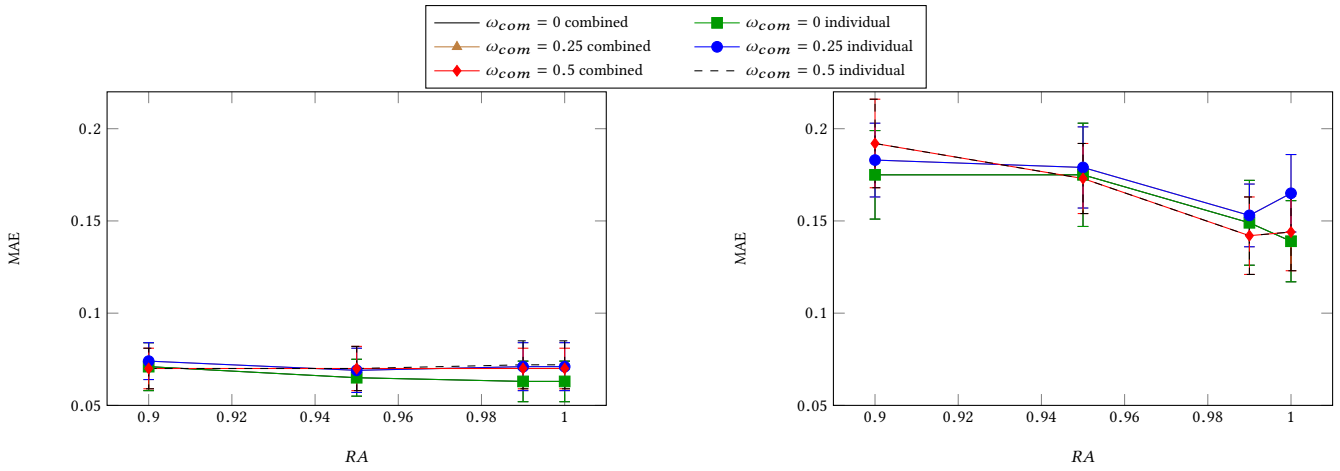
Figure 7 shows the number of bytes required to represent the first ten selected from the (a) LED and (b) POL, when SC=1. The number of bytes are shown for features when compressed individually (as in Equation 10) or together (as in Equation 12) and for different values of ω_{com}. There were more bytes required to represent the features in POL than those in LED, due to the larger number of samples in the testing data. The number of bytes required decreased significantly with lower values of *RA*, meaning that applying a lossy transform does indeed improve compression of vehicle telemetry data. Data compression also improved when the compressibility factor was used during feature selection, demonstrated by the number of required bytes decreasing as ω_{com} increased.

There was a small reduction in the number of bytes required when compressing the features individually over compressing them together. This is possibly because each signal is sufficiently different (due to minimising redundancy in MRMR+C) and each having a distinct alphabet or different symbol frequencies. Handling each feature separately with individual dictionaries and Huffman trees (as in Figure 2(b)), may therefore improve compression when applying DEFLATE. When combining the signals into one stream before



(a) LED (R^2) **(b) POL (R^2)**

Figure 5: R^2 performances for (a) LED and (b) POL with different compression factors, when SC=LC=EC=1.



(a) LED **(b) POL**

Figure 6: MAE performances for (a) LED and (b) POL with different compression factors, when SC=LC=EC=1.

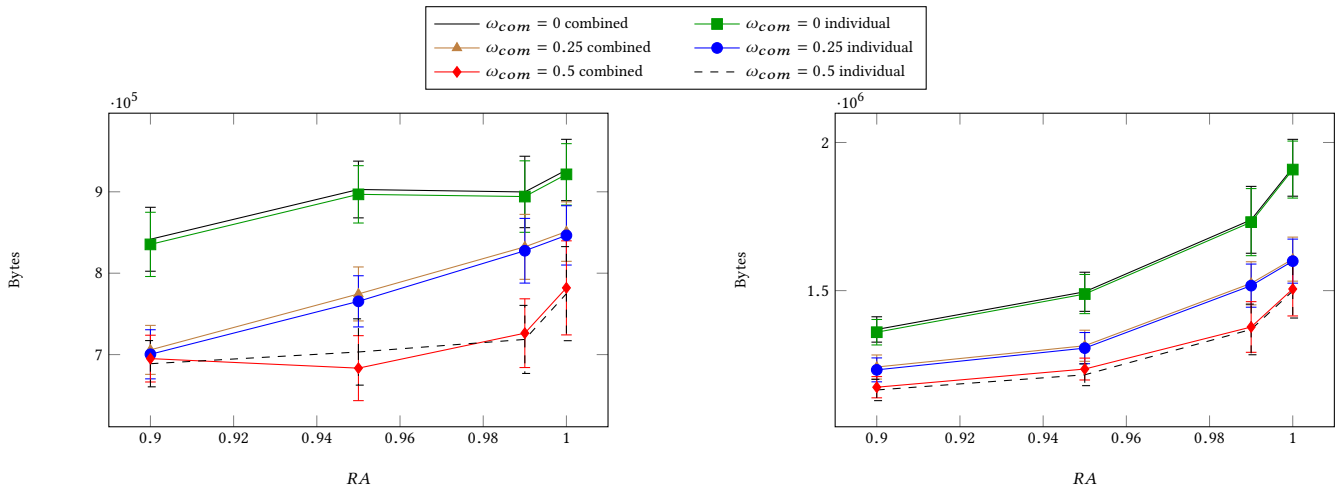
applying DEFLATE (as in Figure 2(a)), two neighbouring signals may share a dictionary to the detriment of compression.

The reconstruction accuracies measured on the testing data using Equation 3, are shown for the (a) LED and (b) POL in Figure 8. In this figure, RA is the parameter used in the training stages, which determines the level of lossy compression applied to each feature, and RA_{eval} is the reconstruction accuracy measured on the features in the testing data. Compressing the features individually or together cannot affect the accuracy as it is a lossless process, so results are presented only for different values of ω_{com}. RA_{eval} improved with higher values of ω_{com} for both datasets, suggesting that consideration of compressibility may lead to the selection of features that are more robust to lossy transforms.

In both datasets the reconstruction accuracies on the testing data, RA_{eval}, decreased as with lower values of the parameter, RA. The reconstruction accuracies measured on testing data, R_{eval}, were higher than the training parameter RA parameter. This was because of the choice in Equations 8, where lossless compression was used if it achieves better compression than lossy compression. Lossless compression always has an accuracy of 1, and so this increased the reconstruction accuracies when measured on data. If lossy compression was always used, the accuracies measured on testing data would be lower than the accuracy used in training data in general.

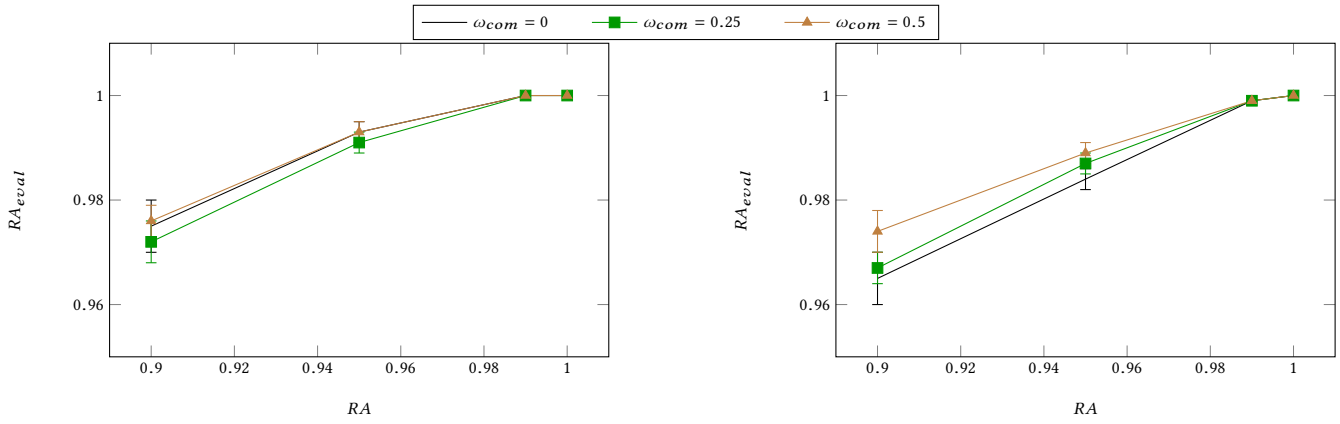
6 CONCLUSION

In this paper we have investigated the effects of lossy data compression on the data mining process. We have identified three stages in the data mining process where lossy data compression should be



(a) LED **(b) POL**

Figure 7: Number of bytes in features selected from (a) LED and (b) POL when compressed, when SC=1.



(a) LED **(b) POL**

Figure 8: Testing feature reconstruction accuracies, RA_{eval} of features selected from (a) LED and (b) POL when SC=1 and for different training reconstruction accuracies, RA.

considered, and evaluated the effects of doing so. We then demonstrated the methodology in selecting features, learning a model, and evaluating it, using two vehicle telemetry datasets, namely the LED and the POL. We found that considering it during the feature selection stage improved the predictive performance of models, but considering it at all stages gave the best performances overall.

We also investigated the benefits of compressing features individually or concatenated in a single stream, and investigated the application of lossless and lossy compression to features. Based on these compression methods, two different compression factors were investigated for use during feature selection, to consider compressibility and predictive performance together, in MRMR+C. We found a small improvement in compression when compressing features individually, rather than together. In MRMR+C, higher values of ω_{com} significantly improved the compression of selected features, but their predictive performances in models were worse. Predictive performance of features compressed using a lossy transform also worsened with lower reconstruction accuracies, but compression improved.

As future work we intend to investigate the compression of signals using more complex and specialised methods. For example, choosing from a variety of lossy transforms for each feature in order to enable the best compression. We also intend to investigate feature relationships and their redundancy structure to improve the compression further.

ACKNOWLEDGEMENTS

This work was supported by Jaguar Land Rover and the UK-EPSRC grant EP/N012380/1 as part of the jointly funded Towards Autonomy: Smart and Connected Control (TASCC) Programme.

REFERENCES

[1] Paul Addison. 2017. *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance.* CRC press.

[2] Charu Aggarwal. 2013. *Managing and mining sensor data.* Springer US, Boston, MA.

[3] Salem Alelyani, Jiliang Tang, and Huan Liu. 2013. Feature selection for clustering: A review. In *Data Clustering*. Chapman and Hall/CRC, 29–60.

[4] Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. *Computers & Electrical Engineering* 40, 1 (2014), 16–28.

[5] Peter Deutsch. 1996. *DEFLATE Compressed Data Format Specification version 1.3. RFC 1951.* Technical Report. Internet Engineering Task Force.

[6] Chris Ding and Hanchuan Peng. 2005. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology* 3, 02 (2005), 185–205.

[7] Jennifer Dy and Carla Brodley. 2004. Feature selection for unsupervised learning. *Journal of machine learning research* 5, Aug (2004), 845–889.

[8] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research* 3, Mar (2003), 1157–1182.

[9] David Hand. 2006. Data Mining. *Encyclopedia of Environmetrics* 2 (2006).

[10] Gunawan Herman, Bang Zhang, Yang Wang, Getian Ye, and Fang Chen. 2013. Mutual information-based method for selecting informative feature sets. *Pattern Recognition* 46, 12 (2013), 3315 – 3327.

[11] David Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE* 40, 9 (1952), 1098–1101.

[12] George John. 1997. *Enhancements to the data mining process.* Ph.D. Dissertation. stanford university Ph. D. thesis.

[13] Ron Kohavi and George John. 1997. Wrappers for feature subset selection. *Artificial intelligence* 97, 1-2 (1997), 273–324.

[14] Vipin Kumar and Sonajharia Minz. 2014. Feature Selection: A Literature Review. *Smart Computing Review* 4, 3 (2014), 211–229.

[15] Thomas Lal, Olivier Chapelle, Jason Weston, and André Elisseeff. 2006. Embedded methods. In *Feature extraction*. Springer, 137–165.

[16] Pabitra Mitra, CA Murthy, and Sankar Pal. 2002. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence* 24, 3 (2002), 301–312.

[17] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. 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), 1226–1238.

[18] David Salomon and Giovanni Motta. 2010. *Handbook of data compression.* Springer Science & Business Media.

[19] Phillip Taylor. 2015. *Data mining of vehicle telemetry data.* Ph.D. Dissertation. University of Warwick, Coventry, UK.

[20] Phillip Taylor, Nathan Griffiths, and Abhir Bhalerao. 2015. Redundant feature selection using permutation methods. In *Automatic machine learning workshop*. 1–8.

[21] Phillip Taylor, Nathan Griffiths, and Alex Mouzakitis. 2018. Selection of Compressible Signals from Telemetry Data. In *Mining Urban Data Workshop*.

[22] James Van Hinsbergh, Nathan Griffiths, Phillip Taylor, Alasdair Thomason, Zhou Xu, and Alex Mouzakitis. 2018. Vehicle Point of Interest Detection Using In-Car Data. In *Proceedings of the 2Nd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery (GeoAI'18)*. ACM, New York, NY, USA, 1–4.

[23] Ian Witten, Eibe Frank, Mark Hall, and Christopher Pal. 2016. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann.

[24] Jacob Ziv and Abraham Lempel. 1978. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory* 24, 5 (September 1978), 530–536.