# Evaluation of Interpretable Association Rule Mining Methods on time-series in the Maritime Domain

Manjunatha Veerappa✉, Mathias Anneken, and Nadia Burkart

Fraunhofer IOSB, Fraunhofer Str. 1, 76131 Karlsruhe, Germany
Fraunhofer Center for Machine Learning
{manjunatha.veerappa,mathias.anneken,nadia.burkart}@iosb.fraunhofer.de

**Abstract.** In decision critical domains, the results generated by black box models such as state of the art deep learning based classifiers raise questions regarding their explainability. In order to ensure the trust of operators in these systems, an explanation of the reasons behind the predictions is crucial. As rule-based approaches rely on simple if-then statements which can easily be understood by a human operator they are considered as an interpretable prediction model. Therefore, association rule mining methods are applied for explaining time-series classifier in the maritime domain. Three rule mining algorithms are evaluated on the classification of vessel types trained on a real world dataset. Each one is a surrogate model which mimics the behavior of the underlying neural network. In the experiments the *GiniReg* method performs the best, resulting in a less complex model which is easier to interpret. The *SBRL* method works well in terms of classification performance but due to an increase in complexity, it is more challenging to explain. Furthermore, during the evaluation the impact of hyper-parameters on the performance of the model along with the execution time of all three approaches is analyzed.

**Keywords:** association rule mining · interpretability · explainable artificial intelligence · time series classification · maritime domain.

## 1 Introduction

For human operators in surveillance tasks, like coastguards, decision support in critical situations is crucial. While the available amount of data is steadily increasing due to e. g. the availability of inexpensive sensors, an operator can easily be overwhelmed by it. Therefore, automatic systems for supporting the decision making are increasingly used. Depending on the underlying technology, one major drawback for the acceptance of such systems is the in-explainable nature of black box models. In order to make the results of such models more interpretable, different rule mining methods are evaluated in this work.

As a large share of the world's trade is conducted by sea, a smooth and efficient voyage of all participants in sea traffic is crucial. Here, a *multilayer*

*perceptron* (MLP) is used as classifier for the vessel types: By analysing the movement patterns of vessels, the trained model is able to distinguish common vessel types. This can be seen as a building block for more advanced decision support systems.

The structure of this paper is as follows: In section 2, a brief state of the art regarding explainable artificial intelligence (XAI) methods is given. Section 3 explains the foundation towards classification, data preprocessing along with the AIS dataset, and association rule mining (ARM). Section 4 provides an insight into the three evaluated rule mining methods. A list of quality measures, that are used for comparison of the implemented methods, is stated in section 5. Afterwards, in section 6 the results of the experiments are illustrated. The paper finishes in section 7 with a conclusion and a short outlook for future work.

## 2 Related Work

For decision support to be accepted in sensitive domains, trust in machine learning systems is an important factor. Therefore, it is crucial to understand why certain predictions are made by a model. For this purpose, Ribeiro et al. proposed an explanation technique called LIME (Local Interpretable Model-Agnostic Explanation) [16], that provides inside into the prediction of a single instance and is therefore suitable for a local explanation. This method tests what happens to the prediction by varying the data around the instance and trains a local interpretable model on this varied dataset. This allows us to create a picture that the model focuses on and uses to make its predictions.

Another approach for estimating the feature importance was introduced to correct Random Forest based importance measures [4] and is called "Permutation importance" [2]. In this method, feature importance is calculated by measuring how a score (i. e. accuracy, F1, etc.) decreases when a particular feature is permuted. But this method fails when correlated features are present in the dataset. To cope with this problem SHAP (Shapley Additive Explanations) [13] was introduced, which explains the output of any machine learning model. It is based on concepts borrowed from cooperative game theory.

Most of the XAI methods such as LIME, SHAP, and so on are typically highly focused on image data, text data, and tabular data. Unfortunately, the consideration of time-series is only limited for these methods. In comparison a XAI method evaluated on time-series is presented in [17]. This methods produces explanations in the form of heatmaps. The authors state, that this reliance on visual saliency masks can be quite challenging for the users to interpret.

Another work on time-series [11], where the attention mechanism concept is used to propose a deep-learning framework for interpretability. It identifies critical segments that are necessary for the classification performance. In other words, it highlights the important areas of the original data accountable for its corresponding prediction.

In healthcare, a prototype based time-series classification is proposed for interpretability [9], where prototypes are learned during the training of the clas-

sification model. These prototypes are then used to explain which features in the training data are responsible for the time-series classification.

"ShiftTree" is introduced in [10] as an interpretable model-based classification method for time-series. This method is an extension of decision tree methods, which labels different time-series by learning from the dataset. While labeling, instead of splitting the dataset using only a certain attribute, an EyeShifter operator and ConditionBuilder operator are used to move along the time axis of the time-series and compute the attributes dynamically which gives an interpretable description of the time-series.

## 3 Foundations

In this section, firstly, classification problems in general and the evaluated MLP model in particular are introduced. Next, information on the data acquisitions and the necessary preprocessing are given. Lastly, Association Rule Mining (ARM) is explained in brief along with its parameters.

### 3.1 Classification

Classification is a process of mapping the input data to discrete categories, whereas categories are often referred to as targets, labels, or classes. It can either be a binary classification or a multi-class classification problem. In this study, we are addressing a multi-class classification of maritime vessel types, i. e., to classify a vessel/ship-type based on its trajectories over time.

Given $\mathcal{X}$ as all possible inputs and $\mathcal{Y}$ as all possible outputs, multi-class classification can be defined as the mapping $C_{\text{multi}}$ of an input $x \in \mathcal{X}$ to a single class $\hat{y} \in \mathcal{Y}$ and is given by

$$C_{\text{multi}} \colon x \mapsto \hat{y} \,. \tag{1}$$

In this study, a neural network architecture based on an MLP [14] is used as classifier. An MLP is a feed-forward neural network. It consists of one input layer, at least one hidden layer, and one output layer. The layers are connected from the input to the output layer in one direction. The input will be directly forwarded to the next layer. For each neuron in the following layer, the inputs are weighted and together with a bias are summed up. This weighted sum is used in the activation function of each neuron. The result will be fed to the next layer. At the output layer, either a decision will be made in case of testing or calculations will be further used for the back-propagation algorithm [18] in case of training.

### 3.2 Data Preprocessing

The dataset was recorded using the Automatic Identification System (AIS). AIS is an automated tracking system designed for the exchange of information between ships, as well as on and off-shore facilities. It was introduced as a means

to prevent collisions of vessels by providing the position and other useful information to vessels in the vicinity. There are requirements for vessels to carry and use AIS, e. g. due to their capacity. Vessels with onboard transceivers will continuously broadcast information regarding the vessel together with identifiers like the Maritime Mobile Service Identity (MMSI). [15]

AIS information transmitted by a ship can be divided into three types: static information, dynamic information, and voyage information. Dynamic information consists of position, speed over ground, course over ground, heading, etc. which are sent (depending on the navigation status) at intervals of two to ten seconds While underway and three minutes otherwise, whereas static and voyage information such as ship type, vessel dimensions, draught as well as destination and ETA are sent every six minutes. [15]

The vessels are divided into a set of ship-types, since most of them are relatively rare, for the evaluation only the most frequent ones (with cargo and tanker due to their similar moving patterns being fused to a single type) are used: *Cargo-Tanker, Fishing, Passenger, Pleasure-Craft,* and *Tug.*

Before the data is usable for training the classifier, preprocssing is necessary. This includes the same steps as described in [3, 7]: First, the data is grouped by the attached MMSI. For each MMSI a segmentation step is performed which will result in trajectories. In this work, a trajectory $T$ is defined as a sequence of $n$ tuples, each containing the position $P_i$ of an object at time $t_i$:

$$T = \{(t_1, P_1), (t_2, P_2), \ldots, (t_n, P_n)\} \,, \tag{2}$$

where the points $P_1, P_2, \ldots P_n$ are the real positions on the surface of the earth as provided by the AIS. Afterwards, the trajectories are filtered and features directly derived from the stored data and additional geographic features are calculated. As a last step normalization is performed on the resulted sequences.

The raw data for every vessel is segmented using three methods: First, if two successive samples have a time difference greater than 2 hours, second if two successive samples have a greater spatial distance than $10^{-4}$ calculated in degrees, and third if the length of a trajectory exceeds the desired length. After that, if a sequence is shorter than 80 % of the sequence length, it is discarded, otherwise it is padded with zeros. As next step, all trajectories, which are stationary and which are traveling on rivers, are removed.

Nine features are extracted for each sample: Firstly, the speed over ground, the course over ground, the position in longitude and latitude (in a global and local manner) as well as the time difference between two samples can directly be used. Based on the positional feature additional features are derived: The distance to the coastline, and the distance to the closest harbour. The global position is a position normalization over the whole dataset, while the local position is a normalization over the trajectory itself. Before these steps are made, the trajectory will be moved to start in the origin in order to achieve translational invariance.

### 3.3 Association Rule Mining — ARM

ARM is one of the six tasks in data mining [8], which uncovers interesting relations between variables from a large dataset. A rule is basically an if-then condition/statement which has 2 items, an antecedent $p$ and a consequent $q$, and takes the form $p \to q$. It can be seen as a tuple $(A, \hat{y})$, where $A$ is a list of $k$ predicates, the antecedents of the rule, and a prediction $\hat{y}$ as its consequent. Typically, the rules can be expressed in the form of

$$\begin{aligned} &\textbf{IF } \text{feature}_n = \alpha \textbf{ AND } \text{feature}_m < \beta \textbf{ AND } \cdots \\ &\textbf{THEN } \text{class } \hat{y}. \end{aligned} \tag{3}$$

These rules are generated by looking for frequent patterns in data. Well known measures that are used to generate each rule are support and confidence. In general, the aim is to create association rules for which the support and confidence is greater than the user-defined thresholds.

The association rules are derived from itemsets, which consist of two or more items. Let $I = \{i_1, i_2, ..., i_m\}$ be a set of all items and $T = \{t_1, t_2, ..., t_n\}$ be a set of all transactions. The *support* $S$ can be defined as an indication of how frequently the itemset appears in all the transactions. It is the ratio of the number of transactions that contains $p$ and $q$ to the total number of transactions:

$$S(p \to q) = \frac{\text{Transactions containing both } p \text{ and } q}{\text{Total number of transactions}} = \frac{frq(p,q)}{|T|} \tag{4}$$

The *confidence* $C$ is defined as an indication of how often the generated association rule has been found to be true. It is defined as the ratio of the number of transactions that contain $p$ and $q$ to the number of transactions that contain $p$:

$$C(p \to q) = \frac{S(p \to q)}{S(p)} = \frac{frq(p,q)}{frq(p)} \tag{5}$$

Association rules are generated by using these two criteria. As for large datasets, the mining of all datasets is not necessarily feasible. Therefore, frequent pattern mining algorithms [1] are introduced to speed up the computation.

A pattern is said to be frequent if feature values are co-occurring in many transactions. A pattern can be a single feature or combination of features. In the dataset the frequency of a pattern is determined by its support [6]:

$$S(\mathcal{P}) = \{(x, \cdot) \in \mathcal{D} \mid \forall p \in \mathcal{P} \colon p(x) = \text{True}\}, \tag{6}$$

where $\mathcal{P}$ is a set of antecedents $p$. Algorithms for determining these patterns are called *Frequent Pattern Mining* [12]. In this study, the *Equivalence Class Transformation (ECLAT)* algorithm [21] is used to find frequent itemsets.

## 4 Methods

In this section, for each of the three evaluated ARM methods, a brief description is given. For the purpose of clarity, the notation found in [5] is adopted for all three methods.

### 4.1 Scalable Bayesian Rule Lists — SBRL

The first method is known as the *Scalable Bayesian Rule Lists (SBRL)*, which explains a model's prediction using a decision list generated by the Bayesian rule lists algorithm [20]. A decision list is a set of rules with simple if-then statements consisting of an antecedent and a consequent. Here, the antecedents are conditions on the input features and the consequent is the predicted outcome of interest.

The scheme of this method is as shown in fig. 1a. The typical objective of a model is to find a function $f\colon\ \mathcal{X} \to \mathcal{Y}$, given $\mathcal{X}$ as the space of all possible inputs and $\mathcal{Y}$ as the set of all possible outputs. The dataset $\mathcal{D}$ of size $N$, described by

$$\mathcal{D} = \{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i \in \mathcal{Y}\}_{i=1}^{N} \tag{7}$$

consisting of the feature vectors $x_i \in X$ and the target classes $y_i \in Y$, is passed onto the data preprocessing section before training a model. The preprocessing steps include transformation, segmentation, filtering and normalization as described in section 3.2. The output of the preprocessing step is continuous numeric data which is then used to train the main model. Here, the main model is a *Neural Network (NN)*, denoted by $M_\theta$, given by its parameters $\theta \in \Theta$. The aim of the main model is to minimize the loss function $L(\theta, \mathcal{D})$, which optimizes the algorithm by estimating the classification performance:

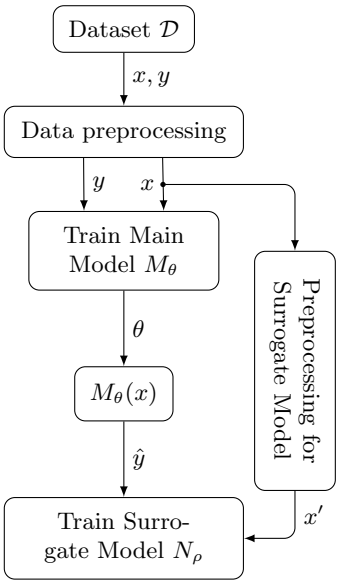
$$\theta^* = \arg\min_{\theta \in \Theta} L(\theta, \mathcal{D})\,. \tag{8}$$

Since the surrogate model in our case is an *SBRL* algorithm, which expects categorical features as an input, the output of the preprocessing data (numeric) must be converted to categorical data. To achieve this, numerical features are binned, i. e., the range of numerical values is divided into discrete intervals. For the experiment, the number of bins is varied between five and 30 bins. Now, each categorical feature shows which interval contains the original numeric value. So we obtain a new dataset $\mathcal{D}'$ of size $N$:

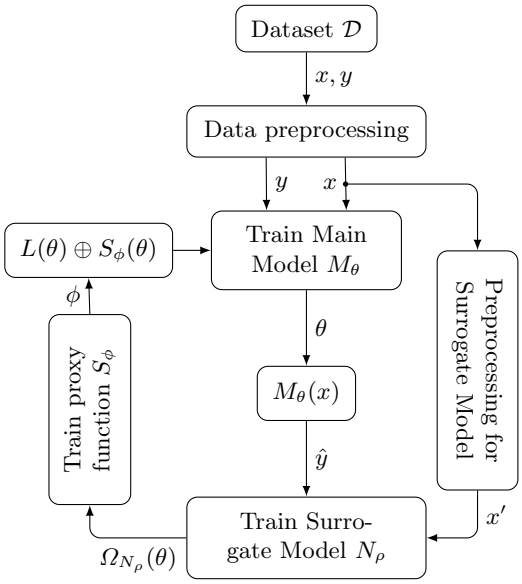
$$\mathcal{D}' = \{(x_i', M_\theta(x_i)) \mid x_i' \in X'\}_{i=1}^{N}\,. \tag{9}$$

The preprocessed input data $x_i'$ along with $M_\theta(x_i)$ is then fed to the surrogate model, denoted by $N_\rho$, $\rho$ being the surrogate rule list containing rules in the form of $(A, \hat{y})$. The surrogate model first mines the rules by extracting frequently occurring feature values/patterns from the dataset $\mathcal{D}'$ by using the *ECLAT* algorithm [21]. Next, the surrogate model learns a decision list from the pre-mined rules and generates high confidence rules from each frequent itemset by ensuring high posterior probability. In general, we could say that the surrogate model aims to optimize the posterior probability

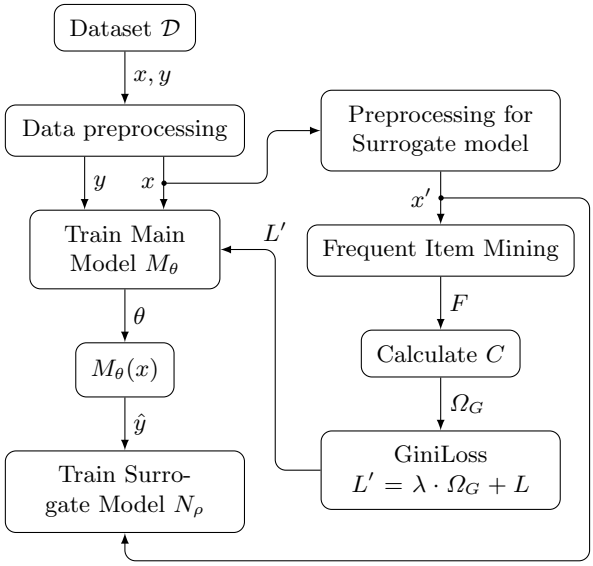
$$P(d \mid x, y, \mathcal{A}, \alpha, \lambda, \eta) \propto \underbrace{P(y \mid x, d, \alpha)}_{\text{likelihood}}\ \underbrace{P(d \mid \mathcal{A}, \lambda, \eta)}_{\text{prior}} \tag{10}$$




(a) The scheme of the *SBRL* method.




(b) The scheme of the *Rule Regularization* method [5].




(c) The scheme of the *Gini Regularization* method.

Fig. 1: The scheme of all three rule mining methods.

to obtain the best rule lists, where $d$ denotes the rules in the list, $\lambda$ is the desired size of the rule list, $\mathcal{A}$ is the set of pre-mined rules, $\eta$ is the desired number of terms in each rule and the preference over labels $\alpha$, usually set to 1 to avoid preference over target labels. Finally, with the help of a decision list, the surrogate model $N_\rho$ that mimics the output of the main model $M_\theta$, will be able to explain a model's prediction.

### 4.2 Rule-based Regularization Method

Unlike the first method, which is an unregularized method, *Rule-based Regularization (RuleReg)* [5] is a technique where a metric for complexity that acts as a degree of explainability is obtained from the surrogate model and fed back to the training of the main model as a regularization term. This method, similar to the *SBRL*, explains a model's prediction using a decision list consisting of a set of antecedents and consequent.

The scheme of the *RuleReg* technique is shown in fig. 1b. In contrast to the *SBRL* method, both the main model and the surrogate model are trained simultaneously. Thus, the dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, is passed through the preprocessing steps to the main model $M_\theta$. As usual, the main model aims at optimizing the objective function such as the loss function $L$. Since it is a regularization technique, the regularization term $\Omega$ is added to the actual loss $L$ resulting in the new loss function

$$L'(\theta, \mathcal{D}) = \lambda \cdot \Omega(\theta) + L(\theta, \mathcal{D}) \,, \tag{11}$$

where $\lambda$ is the regularization strength. Please note that the term $\Omega$ is derived from the surrogate model to add a penalty to the cost function.

During the training of the main model, *RuleReg* often builds a new surrogate model for every training batch and the metric $\Omega_{N_\rho}$ is calculated using the function

$$\Omega_{N_\rho}(\theta) = 1 + \sum_{(A,\hat{y}) \in \rho} |A| \,, \tag{12}$$

which sums up the number of antecedents for all rule lists. And this metric cannot be used directly in the eq. (11) as it is not differentiable. In order to overcome the problem of the regularization term not being differentiable, a proxy function $S_\phi$—here an MLP—as introduced by [19] is used to approximate $\Omega$. Now, the actual loss function is

$$L''(\theta, \mathcal{D}) = \lambda \cdot S_\phi(\theta) + L(\theta, \mathcal{D}). \tag{13}$$

The inputs to fit the proxy function are the vectors of the main model parameters collected during the training of the main model and its corresponding complexity obtained from the surrogate model $N_\rho$.

And certainly, the surrogate model is fitted with the dataset $\mathcal{D}'$ of size $N$, $\{(x_i', M_\theta(x_i))\}_{i=1}^N$, where the numerical features are converted to categorical because of the requirement of the rule-mining algorithm such as *SBRL*. Once both

the main model and the surrogate model have been trained successfully, a surrogate rule list is generated by the surrogate model which is human-simulatable. This is then used to explain a model's prediction.

### 4.3 Gini Regularization Method

Similar to the *RuleReg* method, *Gini Regularization* [6] is a regularization technique which fits a global surrogate model to a deep NN for interpretability. But instead of repeatedly building a new surrogate model for every training batch as in *RuleReg*, this method focuses on training only one surrogate model. In addition, a differentiable regularization term $\Omega$ allowing gradient-based optimization is used to penalize the inhomogeneous predictions.

The scheme of this method is as shown in fig. 1c. The dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ with $N$ being the size, is fed into the preprocessing section and then to train the main model (NN), denoted by $M_\theta$. Before training a main model, frequent item mining is performed on an input data $x'$ (categorical data) to obtain a set of frequent items $F$. For this purpose, the *ECLAT* algorithm [21] is used. Once the frequent itemsets are mined, a binary matrix called *Caught-Matrix* $C$ is calculated using

$$C_{i,j} = \begin{cases} 1, & x_i \in F_j \\ 0, & \text{otherwise} \end{cases} , \tag{14}$$

indicating whether a datapoint is a member of the set $F$, i. e., if a datapoint $x_i$ is in frequent itemset $F_j$, then the *Caught-Matrix* $C$ at $i, j$ is 1 or 0 otherwise. Now, a differentiable loss function $L'$ is constructed using *Caught-Matrix* and a classification loss function:

$$L'(\theta, D) = \lambda \cdot \Omega(\theta) + L(\theta, D) , \tag{15}$$

where $\Omega$ is the regularization term obtained despite constructing a rule-based model and $\lambda$ is the regularization strength.

The main model is then trained on a dataset $\mathcal{D}$ with regard to $L'$. To mimic the output of the main model, the surrogate model $N_\rho$ is trained on a dataset $\mathcal{D}' = \{(x'_i, M_\theta(x_i))\}_{i=1}^{N}$ where $x' \in X'$ is the preprocessed input data. This means the numerical features are binned in order to convert them into a categorical data. To put in short, the surrogate model such as the SBRL algorithm then mines the rules and generates a decision list containing set of rules, which is used to explain a model's prediction.

## 5 Evaluation Metrics

Evaluating any machine learning or deep learning algorithms is an essential part of the task. In this section, the list of quality measures that are used to compare the performance of the three methods are listed.

*$F_1$-score of the main model ($F_{1,main}$)* is the harmonic mean between precision and recall of the main model $M_\theta$, which seeks a balance between precision and recall. It is defined as

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{16}$$

Typically the *$F_1$-score* is more useful than accuracy, particularly if the class distribution is uneven. Here, the prediction function is an MLP algorithm.

*$F_1$-score of the surrogate model ($F_{1,surr}$)* is the harmonic mean between precision and recall of the surrogate model $N_\rho$ and defined as shown in eq. (16). Here, the prediction function is an SBRL algorithm, which is used to mimic the behavior of the main model $M_\theta$.

*Fidelity* is an accuracy score, which measures the predictions of the surrogate model to the predictions of the main model. This metric matters the most, as the surrogate model tries to mimic the behavior of the main model.

*Average Number of Rules (ANR)* is the number of rules which a decision list uses to classify a single sample. It should be noted that the notion of Average Path Length (APL) is not explicitly transferable to rule lists, but for comparison purposes, APL has been replaced with ANR.

## 6 Experimental Results

In this section, the derived datasets and the experimental setup used for training both the main model and the surrogate model for all three methods are explained. In addition, all three methods are evaluated. Each method is evaluated on every dataset. Finally, the results are compared with the other methods and are discussed.

### 6.1 Datasets

The experiments are carried out on four datasets. Each dataset has been extracted from the raw AIS data as described in section 3.2. All datasets use the same spatial and temporal thresholds. The only difference between the datasets is the sample count per sequence. The chosen lengths are 15, 30, 45 and 60 corresponding to 15 min, 30 min, 45 min and 60 mins of data respectively, with a sampling time of 1 min. In general, each dataset has 10000 sequences with the sequence length varying depending on the chosen timespan, and each sample in turn has nine features as specified in 3.2. The target is to classify a vessel/ship-type: *Cargo-Tanker, Fishing, Passenger, Pleasure-Craft,* and *Tug.*

### 6.2 Experimental setup

Through all our experiments the main model was a standard MLP. We used the same main model structure with two hidden layers for all datasets. The number of neurons present in the input layer depends on the number of features present in a

dataset, as noted in table 1, resulting in (number of samples·number of features) neurons. A *sparse categorical cross-entropy* is used as a classification loss function, $ReLU$ as an activation function in all hidden layers and *softmax* for the output. The main model uses the Adam optimization algorithm with a learning-rate of 0.001. The sequences of the form (number of samples, number of features) are collapsed into a single dimension (number of samples · number of features) to fit as input for the MLP. Numerical features were binned in order to train the surrogate model. The surrogate model is trained with default hyper-parameters of *SBRL* (as pysbrl on pip), except a *support* parameter which is varied and evaluated the models performance. The $max\_rule\_len$ parameter is chosen as 2, keeping in mind the processor and the RAM available to carry out our experiments.

### 6.3 Results

This study aims at evaluating the results of each method that are accurate and interpretable. Therefore, each method is implemented on every dataset and its corresponding results are shown in table 1. It can be seen that the *SBRL* method outperforms the other two methods in terms of metrics such as $F_{1,main}$, $F_{1,surr}$, fidelity. However, the length (number of rules) of the surrogate list is relatively high, making the model more complex to interpret. *GiniReg* on the other hand, though the $F_1$-score of both the main model and the surrogate model is a bit less compared to *SBRL*, relatively higher fidelity is achieved and the length of the resulting surrogate list is reduced, making the behavior of the model less complex to interpret. One such example of a surrogate model on 60 min dataset is shown in fig. 2. *RuleReg* performs the least since it generates or trains several surrogate models during the training of the main model as compared to the other two methods which train only one surrogate model on the whole. In case of *RuleReg*, It should be noted that training the models with less batch size decreases the performance of the surrogate model and with more batch size decreases the performance of the main model. Hence, choosing the correct batch size plays a major role.

We trained all three methods with variation in the *support* parameter of a surrogate model as shown in fig. 3. We can see that for increasing *support* value the performance of the surrogate model is going down. This means the surrogate model is highly unlikely to capture the pattern of the ship trajectories with an increase in *support* value. Hence, all the models are further trained with a *support* value of 0.01.

Since we bin the numerical features before training a surrogate model, we were curious how this would impact the overall model's performance. Therefore, we trained the models with different bins (no. of discrete intervals) as shown in fig. 4. It can be observed that the performance of all three models increases to some extent and starts decreasing gradually with an increase in the number of bins. This means the models are able to learn better if the features are categorized into more number of discrete intervals. But it should be noted that large discrete intervals make the model fail to learn the patterns in the trajectories of the ships.

Table 1: Performance of all three methods.

| Method | Duration | Layers | $F_{1,main}$ | $F_{1,surr}$ | Fidelity | No. of rules | APL/ANR | $\lambda$ |
|---|---|---|---|---|---|---|---|---|
| *SBRL* | | | **0.80** | **0.84** | 0.84 | 52 | 26.50 | 0.0 |
| *RuleReg* | 15 min | [128,128] | 0.66 | 0.67 | 0.82 | 46 | 22.71 | 1 |
| *GiniReg* | | | 0.74 | 0.76 | **0.86** | **41** | **22.11** | 0.05 |
| *SBRL* | | | **0.82** | **0.84** | 0.85 | 58 | 25.20 | 0.0 |
| *RuleReg* | 30 min | [256, 256] | 0.70 | 0.71 | 0.81 | 47 | 23.19 | 10 |
| *GiniReg* | | | 0.80 | 0.82 | **0.85** | **42** | **22.01** | 0.01 |
| *SBRL* | | | **0.83** | **0.85** | 0.85 | 59 | 29.50 | 0.0 |
| *RuleReg* | 45 min | [256, 256] | 0.71 | 0.74 | 0.85 | 42 | 21.52 | 1 |
| *GiniReg* | | | 0.80 | 0.81 | **0.85** | **37** | **19.99** | 0.01 |
| *SBRL* | | | **0.84** | **0.85** | 0.85 | 57 | 29.50 | 0.0 |
| *RuleReg* | 60 min | [512, 512] | 0.69 | 0.70 | 0.81 | 45 | 22.70 | 1 |
| *GiniReg* | | | 0.80 | 0.82 | **0.86** | **40** | **22.53** | 0.1 |

**IF** Course_161 in (0.123, 0.164] **AND** Course_503 in (-0.00102, 0.0409]
**THEN** Shiptype = Cargo-Tanker (0.95)†
**ELSE IF** (DisToharbor_186 in (0.959, 0.999]) **AND** (DisToCoast_221 in (0.369, 0.406])
**THEN** Shiptype = Passenger (0.83)†
**ELSE IF** (DisToCoast_392 in (0.963, 1.0]) **AND** (Local_Y_39 in (0.04, 0.08])
**THEN** Shiptype = Cargo-Tanker (0.88)†
**ELSE IF** (Speed_403 in (0.96, 1.0]) **AND** (DisToharbor_429 in (0.599, 0.639])
**THEN** Shiptype = Passenger (0.94)†
**ELSE IF** (DisToharbor_276 in (0.16, 0.2]) **AND** (Global_X_288 in (0.495, 0.516])
**THEN** Shiptype = Pleasure-Craft (0.65)†
**ELSE IF** (Global_Y_181 in (0.421, 0.436]) **AND** (DisToCoast_194 in (0.232, 0.27])
**THEN** Shiptype = Fishing (0.40)†
**ELSE IF** (Course_17 in (0.0818, 0.123]) **AND** (Course_296 in (0.0818, 0.123])
**THEN** Shiptype = Cargo-Tanker (0.91)†
**ELSE IF** (DisToCoast_374 in (0.961, 1.0]) **AND** (DisToharbor_465 in (0.919, 0.959])
**THEN** Shiptype = Tug (0.79)†
**ELSE IF** (DisToCoast_32 in (0.282, 0.32]) **AND** (Course_521 in (-0.00102, 0.0409])
**THEN** Shiptype = Pleasure-Craft (0.41)†

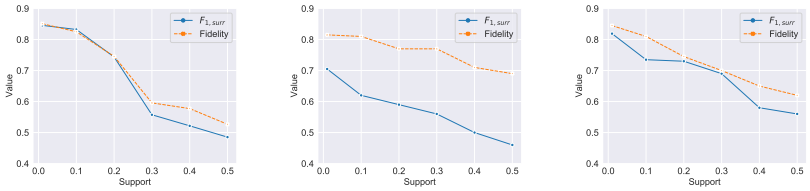
⋮

**ELSE** Shiptype = Cargo-Tanker (0.82)†

† predicted class with prediction probability in brackets

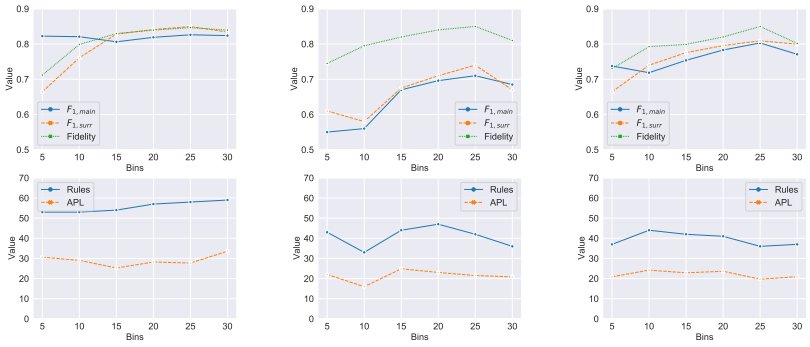
Fig. 2: Example Surrogate Model for 60 min AIS dataset.

The *GiniReg* and *RuleReg* methods are regularization-based techniques and hence we trained several models with different regularization strength, $\lambda$. We can see that the number of rules in the resulting surrogate model and APL/ANR is going down with an increase in $\lambda$. This indicates both these techniques make a model less complex to interpret. In case of *RuleReg*, if the strength is significantly increased to a higher value, the model would collapse into a single target output




(a) *SBRL* method (b) *RuleReg* method (c) *GiniReg* method

Fig. 3: The performance of the surrogate model ($F_1$, Fidelity) against different support parameter of the SBRL on 30 min AIS dataset.




(a) *SBRL* method (b) *RuleReg* method (c) *GiniReg* method

Fig. 4: The performance of both the main model and the surrogate model against different bins in the preprocessing for surrogate model on 45 min AIS dataset.

with just one default rule. However, this would not be the case in *GiniReg* technique.

In addition, we examined the runtime of all three methods on every dataset, which includes training of both the main model and the surrogate model. Table 2 represents the result in *secs* obtained in the experiments. In all the cases, *SBRL* method is faster and *RuleReg* is slower. As mentioned before, *RuleReg* generates several surrogate models compared to the other two methods which makes them relatively slow. We used an Intel(R) Core(TM) i7-5960X processor with 64GB RAM to conduct our experiments. An increase in memory should improve the runtime of all the methods.

## 7 Conclusion and Future Work

During this study, three ARM techniques are evaluated on time-series data in order to make the results interpretable. In our experiments, we find that the

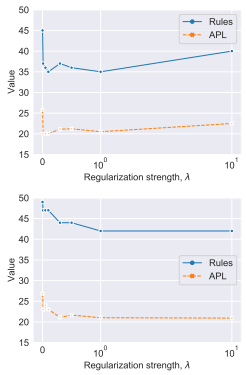



Fig. 5: Length of surrogate list and APL for increasing $\lambda$ on 45 min dataset. Top: *GiniReg*, bottom: *RuleReg*.

| Method | Dataset | Runtime in $s$ |
|---|---|---|
| *SBRL* | | 429 |
| *RuleReg* | 15 min | 3455 |
| *Ginireg* | | 466 |
| *SBRL* | | 1106 |
| *RuleReg* | 30 min | 11533 |
| *Ginireg* | | 1537 |
| *SBRL* | | 2142 |
| *RuleReg* | 45 min | 20500 |
| *Ginireg* | | 3119 |
| *SBRL* | | 3578 |
| *RuleReg* | 60 min | 31631 |
| *Ginireg* | | 4316 |

Table 2: Runtime of all the methods

*GiniReg* technique works best on every dataset having high fidelity to the main model and a lower number of rules, making a model less complex to interpret. However, *SBRL* works better in classifying a vessel but with more number of rules in the surrogate list. *RuleReg* on the other hand achieves higher fidelity but with low performance of the main model and the surrogate model. This paper also demonstrates the impact of *support*, *bins*, *regularization* parameters on the performance of the model. In addition, the runtime of all the methods was examined, where *SBRL* was faster and *RuleReg* was a lot slower.

Explainability or interpretability remains a very active area of research in deep learning. The major rule mining methods surveyed in this study each represent a step towards more fully understanding a deep learning model like MLP. Further work on applying these techniques to other architectures like CNN's or GRU's would be interesting.

**Acknowledgment.** Underlying projects to this article are funded by the WTD 81 of the German Federal Ministry of Defense. The authors are responsible for the content of this article. This work was developed in the Fraunhofer Cluster of Excellence "Cognitive Internet Technologies".

## References

1. Aggarwal, C.C., Bhuiyan, M.A., Al Hasan, M.: Frequent pattern mining algorithms: A survey. In: Frequent pattern mining, pp. 19–64. Springer (2014)
2. Altmann, A., Tolosi, L., Sander, O., Lengauer, T.: Permutation importance: A corrected feature importance measure. Bioinformatics (Oxford, England) **26**, 1340–7 (04 2010). https://doi.org/10.1093/bioinformatics/btq134

3. Anneken, M., Strenger, M., Robert, S., Beyerer, J.: Classification of Maritime Vessels using Convolutional Neural Networks. UR-AI 2020 (2020), accepted for publication
4. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
5. Burkart, N., Huber, M., Faller, P.: Forcing interpretability for deep neural networks through rule-based regularization. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). pp. 700–705 (2019)
6. Burkart, N., Huber, M., Faller, P.: Batch-wise regularization of deep neural networks for interpretability. In: 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI) (2020), accepted for publication
7. Burkart, N., Huber, M.F., Anneken, M.: Supported decision-making by explainable predictions of ship trajectories. In: 15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020). pp. 44–54. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-57802-2_5
8. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI magazine **17**(3), 37–37 (1996)
9. Gee, A., Garcia-Olano, D., Ghosh, J., Paydarfar, D.: Explaining deep classification of time-series data with learned prototypes. arXiv preprint arXiv:1904.08935 (2019)
10. Hidasi, B., Gáspár-Papanek, C.: Shifttree: An interpretable model-based approach for time series classification. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 48–64. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
11. Hsu, E.Y., Liu, C.L., Tseng, V.: Multivariate time series early classification with interpretability using deep learning and attention mechanism. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 541–553 (2019). https://doi.org/10.1007/978-3-030-16142-2-42
12. Letham, B., Rudin, C., McCormick, T.H., Madigan, D.: Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. The Annals of Applied Statistics **9**(3), 1350–1371 (2015). https://doi.org/10.1214/15-aoas848
13. Lundberg, S., Lee, S.I.: A unified approach to interpreting model predictions (2017)
14. Popescu, M.C., Balas, V.E., Perescu-Popescu, L., Mastorakis, N.: Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems **8**(7), 579–588 (2009)
15. Raymond, E.S.: AIVDM/AIVDO protocol decoding (2019), https://gpsd.gitlab.io/gpsd/AIVDM.html, accessed: 2020-09-28
16. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?": Explaining the predictions of any classifier (2016)
17. Schlegel, U., Arnout, H., El-Assady, M., Oelke, D., Keim, D.: Towards a rigorous evaluation of xai methods on time series. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW) (2019)
18. Skorpil, V., Stastny, J.: Neural networks and back propagation algorithm. Electron Bulg Sozopol pp. 20–22 (2006)
19. Wu, M., Hughes, M.C., Parbhoo, S., Zazzi, M., Roth, V., Doshi-Velez, F.: Beyond sparsity: Tree regularization of deep models for interpretability (2017)
20. Yang, H., Rudin, C., Seltzer, M.: Scalable bayesian rule lists. In: International Conference on Machine Learning. pp. 3921–3930. PMLR (2017)
21. Zaki, M.J.: Scalable algorithms for association mining. IEEE transactions on knowledge and data engineering **12**(3), 372–390 (2000)