RIPPER-k and FURIA for Interpreting the Output of a Multiclass Neural Network

Master's thesis
Communication and Information Sciences
Specialisation track Cognitive Science & Artificial Intelligence
Tilburg University - School of Humanities and Digital Sciences

Anton Stepanov
a.stepanov@tilburguniversity.edu
ANR: 134879
SNR: 2030716

Supervisor: Dr Grzegorz Chrupała

Second reader: Drew Hendrickson

January 10, 2020

Preface

This thesis is written to fulfil the master's program of Cognitive Science & Artificial Intelligence which is a specialisation track within the master's program of Communication and Information Sciences.

I want to thank my supervisor Dr Grzegorz Chrupała for his critical, specific and valuable feedback. His questions challenged my findings and the way of thinking and stimulated to analyse the thesis topic from different perspectives.

I also want to thank Madhumita Sushil – the co-author of the paper that gave rise to this research project. Madhumita helped me to understand rule induction algorithms better and provided critical insights which allowed to install FURIA.

Last but not least, I want to say a big thank you to my wife - Ekaterina Stepanova for patience and willingness to support me in all my endeavours.

Abstract

This research project shows that rule induction algorithms, such as RIPPER-k and FURIA, generate sets of decision rules which can be understood by non-technical users. However, it also unveils that they are slow and cannot replicate the classification performance of a feedforward neural network.

This research project attempts to combine the classification performance of the feedforward neural network with understandable decision rules of rule induction algorithms. It applies RIPPER-k and FURIA to generate sets of rules from the output of the neural network. In addition, the research compares rule induction algorithms on their ability to interpret predictions of the neural network using the F score. The results on the 20 Newsgroups scikit-learn dataset did not confirm the hypothesis that FURIA provides a better interpretation of the neural network predictions than RIPPER-k.

The project also compares how understandable rulesets generated by RIPPER-k and FURIA are. Overall, FURIA ruleset is more understandable because it provides rules for all classes. RIPPER-k rules are more concise than FURIA rules, but RIPPER-k only provides the default rule (else clause) for the majority class.

The code available at https://github.com/stepanov-a-o/interpret_with_rules_furia
Keywords: Interpretable Machine Learning, Rule Induction, FURIA, RIPPER-k

Table of Contents

RIPPER-k and FURIA for Interpreting the Output of a Multiclass Neural Network

ANTON STEPANOV

1. Introduction

1.1 Context

Machine Learning (ML) models empower and stand behind the recent success of Artificial Intelligence (AI). However, a drawback of most ML models is that they are black-boxes which provide final results with no explanation. Moreover, those results are not interpretable by people without specialised knowledge (Marcus et al., 2018). Lack of interpretability is the main limiting factor for the real-world application of AI, especially in the areas where human lives depend on the model's decision, such as medicine and autonomous driving (Samek, Wiegand, & Müller, 2017).

This research project aims to study the interpretability of ML models. More specifically, it will focus on the ability of rule induction algorithms to interpret the classification output from a neural network. Rule induction algorithms are known as Data Mining and ML techniques for inducing decision rules from data (Góra & Wojna, 2002). Rule induction algorithms are applied to generate rules from the output of a feedforward neural network. The research will use a feedforward neural network (hereinafter the neural network) as this is one of the most widely used ML models for classification tasks (Kowsari et al., 2019; Sidath, 2018). It will also use two rule induction algorithms to generate decision rules, namely: Repeated Incremental Pruning to Produce Error Reduction (RIPPER-k) and Fuzzy Unordered Rule Induction Algorithm (FURIA). The motivation for using specific rule induction algorithms as well as their detailed review and specifications of the neural network will be provided in the literature review and methodology sections.

The most influential work which gave rise to this project is the paper 'Rule induction for global explanation of trained models' by Sushil, Šuster and Daelemans (2018). In this paper, the authors used RIPPER-k to induce sets of if-then-else rules capturing the relations between features and classes to explain the predictions of the neural network globally. The speciality of this work is that authors extracted 1,000 top features from the neural network, and then generated sets of rules that approximate the interaction between these features. In this way, Sushil et al. (2018) encoded some information about the network performance within the features. The authors used 4 classes from the 20 Newsgroups scikit-learn dataset to train the network and generate rulesets explaining its predictions. The final rulesets achieved an outstanding result, and they can explain the predictions of the neural network to a macro-average F score (also known as F_1 score) of 0.80.

1.2 Research Questions

This research project is structured around three main goals. Firstly, it will compare the classification accuracy of RIPPER-k and FURIA on pre-processed subsets of the 20 Newsgroups dataset with the neural network. The subsets will consist of 4 and 8 groups. The literature suggests that the neural network has greater predictive power and therefore, should score a higher classification accuracy than rule induction algorithms (Hagras, 2018). Also, FURIA is expected to be more accurate than RIPPER-k, especially with 8 classes. The reason for this is that FURIA provides independent rules as opposed to hierarchical rules of RIPPER-k. Based on these findings and the mentioned goal, the following research question (RQ) has been formulated:

RQ1: *To what extent do RIPPER-k and FURIA rule induction algorithms replicate the performance of a feedforward neural network in a classification task with 4 and 8 classes?*

The answer to the RQ1 is expected to confirm that the neural network is more accurate than rule induction algorithms. Probably, it will also show that rule induction algorithms provide rulesets which can be understood by a non-technical user. High classification performance of the neural network and interpretable rules of RIPPER-k and FURIA will serve as a foundation for the second goal of this project. The goal is to evaluate whether FURIA is better than RIPPER-k in interpreting the predictions from the network trained on a classification task with 4 and 8 groups from the 20 Newsgroups dataset. The research project uses the F score to measure the interpretability. The hypothesis is that more classes will harm the interpretability of both rule induction algorithms with a more significant impact on RIPPER-k. The reason is that FURIA does not have a default rule for the majority class, and it has more flexible decision boundaries than RIPPER-k, which should be useful for problems with many classes. The sub-goal is to compare rules generated by rule induction algorithms to see whether they can be understood. The goal and sub-goal are translated into the following research question (RQ) and sub-question (SQ):

RQ2: *To what extent does FURIA outperform RIPPER-k in interpreting the predictions from a feedforward neural network trained with 4 and 8 classes?*

> **SQ2:** *How understandable are the rules generated by RIPPER-k and FURIA? (subjective comparison)*

The final goal is to study the impact of the number of input features on RIPPER-k's and FURIA's ability to interpret the output of the neural network. The research project will experiment with 1,000, 500 and 100 features. It will also evaluate the influence of the number of folds for error pruning on the interpretability score. The research question and sub-question are:

RQ3: *What is the impact of the number of input features on RIPPER-k's and FURIA's ability to interpret predictions from a feedforward neural network?*

> **SQ3:** *What is the impact of the pruning hyperparameter on the interpretability score?*

1.3 Contribution

This research project has both scientific and societal contribution. The scientific contribution involves the application of FURIA to interpret the output of the neural network. Besides, this research compares the interpretability (F score) of RIPPER-k and FURIA on the output of the neural network. Only one research has been found that used RIPPER-k to interpret predictions of neural networks, and there are no studies that used FURIA (or any other rule induction algorithms) to do so. There are also no studies found that compare the interpretability (F score) of rule induction algorithms on the output of the neural network. The societal contribution is the improvement of interpretability of ML models, especially by users with no technical background.

1.4 Applicability

This research has several applications. For example, it provides scientists with clear, step-by-step instructions on how to set up FURIA. As it will be covered later (section 3.2.1), the setup process is not straight forward, and only few people in the world have experience in working with FURIA. The instructions will help future scientists to move ahead with their experiments, avoiding many initial complexities.

1.5 Outline

The rest of the research project is split into the following sections. Section 2 provides insights into the background of interpretable ML related to this research. Section 3 describes the dataset, explains the workflow for all research questions and outlines the evaluation metrics. Section 4 presents the results, while section 5 gives more in-depth insights into the results together with the research limitations. Section 6 concludes the research project and gives directions for future research.

2. Background: Interpretable Machine Learning

2.1 Key Terminology

Interpretability and Explainability

Interpretability and explainability are important and closely related concepts for achieving Explainable Artificial Intelligence (XAI). Although these concepts are used interchangeably in the literature, there is a substantial difference between them. Interpretability refers to models which can provide cues about where their 'focus' lies, e.g. by visually highlighting features with more weight. In turn, explainability refers to models that can generate a systematic explanation, outlining reasons for their behaviour (Gilpin et al., 2019). Existing ML models are not capable yet of providing these systematic explanations. Therefore, this research project focuses on interpretability as a crucial step towards explainability.

Local and Global Interpretability

There are two types of interpretability - local and global. Local interpretability refers to the interpretation of a single decision, while global interpretability refers to the interpretation of the entire model (Sushil et al., 2018). A global interpretation may include aspects such as the setup information (models descriptions, hyperparameters), summary statistics of input data, performance metrics, how models were tested and trained, another model-specific information (Edwards & Veale, 2017). This research project focuses on global interpretability, further explained in section 2.2.

2.2 Reasons for Global Interpretability

The research chose global interpretability for reasons outlined below.

Transparency. Understanding models' rationale is critical, for example, in the healthcare industry where the level of precision is extremely high, and a single mistake may have severe consequences. A transparent decision-making process allows people to question and verify models' findings.

Model debugging. Transparency allows to identify models' weaknesses, and this knowledge helps to debug and improve them.

Learn from a model. AI provides endless opportunities for people to learn. ML models do not get bored or tired and therefore can analyse and learn from a vast amount of data, identify patterns and irregularities which cannot be detected by humans (Samek et al., 2017).

Trust. An ability to communicate ideas clearly and understandably is a prerequisite for establishing a trustworthy relationship between people (Samek et al., 2017). The same concept applies to AI, where only those models which present their findings logically and concisely, have a chance to secure users' trust and confidence.

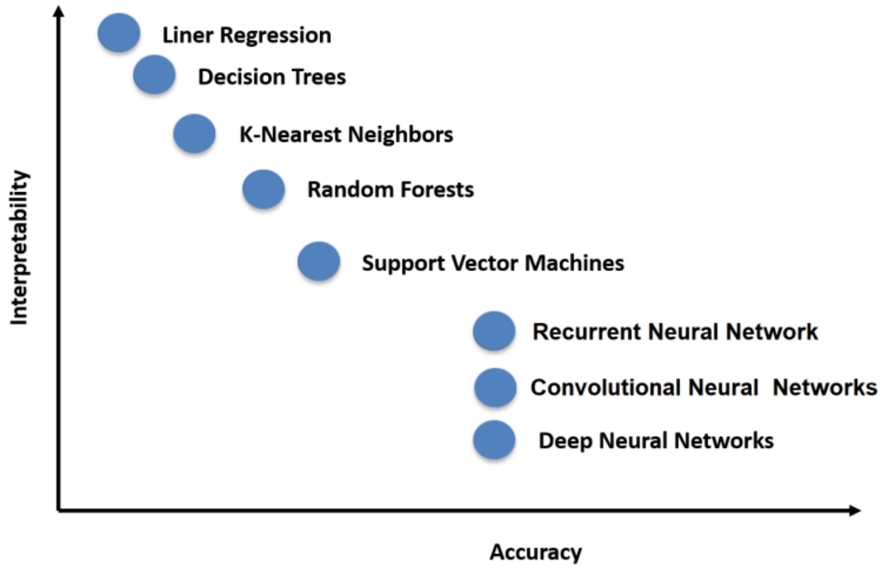
Ethics & Legislation. AI models need to be interpretable to comply with legislation requirements. According to the EU General Data Protection Regulation (GDPR), any individual reserves a right to receive an explanation for a decision made by a system about them (Samek et al., 2017). For example, if a system rejects a client loan application, he has a right to learn about factors leading to that decision (Preece, 2018). There are also ethical aspects of interpretability. Models which are trained on biased data may exhibit undesirable discrimination based, for example, on gender, race or religion.

2.3 An Overview of Interpretable Models in AI

There are three main approaches to interpretable AI (Hagras, 2018). The first approach focuses on the use of interpretable models, such as Linear Regression, Decision Trees or Random Forests. These models are generally considered as interpretable; however, their output can only be analysed by experts with specific knowledge. Moreover, interpretable models have lower performance than more recent, complex classifiers (as shown in Figure 1).

The second approach is 'Deep Explanation'. It involves modification of deep neural networks to learn features with the highest impact on the model output (Hagras, 2018). Deep Explanation models have a superior performance, but they only provide a local explanation which can be interpreted by industry experts.

FIGURE 1: AI MODELS ACCURACY VERSUS INTERPRETABILITY



'Figure 23. The model interpretability comparison between traditional and deep learning techniques.' by Kowsari et al. (2019), used under CC BY 4.0 / Changed the original colour scheme

The third approach is based on induction of decision rules from any black-box model, and it aims to combine the performance of the best techniques with the global interpretability (Sushil et al., 2018). The approach is model-agnostic, and it embraces an if-then-else logic to generate rules. This logic is intuitive, and it provides a natural representation of problems, which is easy to comprehend by non-technical users (Hagras, 2018).

From the above discussion, it becomes clear that the third approach with the induction of decision rules can facilitate models' interpretability without compromising the performance. The next section provides a detailed overview of relevant rule induction algorithms.

2.4 Literature Review

2.4.1 RIPPER-k & FURIA

RIPPER-k was introduced in 1995 as a successor of the Incremental Reduced Error Pruning (IREP) algorithm (Cohen, 1995). RIPPER-k keeps the working principles of the IREP. The main differences between the algorithms are the rule optimisation process (described in the Methodology section 3.2.2), and RIPPER-k's ability to handle multiclass problems. The next paragraph outlines the working principle of RIPPER-k.

RIPPER-k generates a set of if-then-else rules. To do so, it sorts all the data by class labels in ascending order, according to the class frequencies. Then, the algorithm induces if-then rules for each class in a one at a time manner starting from the smallest class. RIPPER-k creates a rule, and it removes all the instances covered by that rule from the training data. The process repeats until rules cover all the instances from the target class or until the last rule becomes 'too complicated'. The latter property is fulfilled if the minimum description length (MDL) of a rule is at least d bits larger than the shortest description length discovered so far. The value for d suggested by Cohen is 64. The algorithm continues to create rules for next classes until it reaches the last and hence the most frequent class. For the last class, RIPPER-k uses the default else rule (J. Hühn & Hüllermeier, 2009).

The disadvantage of this approach is that RIPPER-k cannot start learning rules for a new class until the rules for the current class are learned. The hierarchical order in which classes are examined has a significant impact on the final ruleset. The order of the rules in the ruleset is critical because the first rule that matches an instance will define the class of that instance, and all other rules will be ignored. This disadvantage is particularly harmful to multiclass problems since there can be other rules in the ruleset, which can be a better match for the instance. However, they are never examined (Asadi & Shahrabi, 2016).

FURIA builds on top of RIPPER-k algorithm; therefore, only critical modifications and differences will be highlighted. The first modification is about the structure of a ruleset and the use of a default rule. FURIA does not arrange classes according to class frequencies, and the order in which classes are examined is irrelevant. It learns independent rules for all classes, and there is no default rule for the majority class (J. C. Hühn, 2009). An unordered rule set without a default rule has two potential problems. Rules from different classes can equally well cover a new instance, or the instance maybe not covered by any rule. Although these problems are unlikely to occur, FURIA has mechanisms to deal with them. For example, to avoid a conflict between rules from different classes covering the same instance, FURIA calculates confidence factors (CF) for each rule. In the case of a close tie between two rules, a decision is made in favour of a rule with the highest CF (J. Hühn & Hüllermeier, 2009).

To handle uncovered instances, FURIA uses a rule stretching technique. The idea behind it is to generalise existing rules until they cover all instances. To generalise a rule, FURIA removes rule antecedents (precondition) starting from the last added precondition. The reason for that is the order represents antecedents' importance for a rule where preconditions which were learned earlier are the most important (J. Hühn & Hüllermeier, 2009).

The second modification is about rules decision boundaries. Opposite to RIPPER-k, which generates rigid decision boundaries for each rule, FURIA creates more general boundaries with smooth transitions between classes. For example, a rule $\langle A \leq 10 | + \rangle$ means that the class is positive if attribute A is less or equal to 10. The rule is valid for $A \leq 10$, and it is invalid for $A > 10$. A fuzzy rule $\langle A \in (-\infty, -\infty, 10, 15 | + \rangle$ means that the rule is valid for $A \leq 10$, invalid for $A > 15$, and partially valid in between (J. Hühn & Hüllermeier, 2009). A partially valid rule refers to the idea that for many tasks, it is unreasonable to say that an instance either has or it does not have specific class properties. The instance may exhibit only some characteristics of a specific class and still belong to that class (Stoklasa, Talášek, & Musilová, 2014).

The final significant modification is about pruning. Rule induction with RIPPER-k can be divided into the growing and optimisation phases (described in section 3.2.2). Both of these phases have a pruning step. FURIA omits pruning during rule growing since it harms its performance. FURIA only keeps pruning in the optimisation phase (J. Hühn & Hüllermeier, 2009).

2.4.2 Other Relevant Studies

Besides the works stated above, there are other influential works in the rule induction field. For example, Bayesian Rule Lists (BRL) by Letham, Rudin, McCormick, & Madigan (2015) generate decision lists instead of decision sets of RIPPER-k and FURIA. The decision lists are harder for humans to comprehend than decision sets. In the decision lists, the order of rules is important, and every rule depends on the rule above it not being correct (Lakkaraju, Bach, & Leskovec, 2016). In addition, the chaining if-then-else statements of BRL split the feature space into increasingly small parts, making every next rule less interpretable than the one before (Robeer, 2018).

Bayesian Rule Set (BRS) is a follow-up work by Wang, Rudin, Liu, Klampfl and Macneille (2017) that uses and extend BRL. BRS produces a classification decision set, and it connects features with AND/OR conjunctions. The method utilises Bayesian probabilities, and it allocates rules in decreasing order according to their likelihood with the goal to maximise classification accuracy (Robeer, 2018).

Lakkaraju et al. (2016) and Malioutov, Varshney, Emad and Dash (2017) proposed Interpretable Decision Sets (IDS), and 1Rule induction algorithms. These algorithms learn a single rule per class. IDS constructs rules connected with AND conjunction, and 1Rule creates Boolean statements balancing between interpretability and accuracy. Both algorithms produce decision sets which remain interpretable even for multiclass classification. However, with complex problems, the length of rules can limit interpretability.

The discussion above highlights the advantages and limitations of different rule induction algorithms. Most of the rule learners produce a decision list which consists of rules learned from smallest to the second largest class (in terms of relative frequency), and a default rule is added for the majority class (J. Hühn & Hüllermeier, 2009). The limitation of this approach is the bias towards the majority class. FURIA overcomes this limitation by learning unordered sets of rules. Also, it has more flexible decision boundaries than other rule learners. These factors make FURIA an exciting alternative to other rule induction algorithms, especially for multiclass classification problems.

This research project focuses not only on rule induction algorithms but also on whether they can interpret the output from trained models. Deep Learning Important FeaTures (DeepLIFT) is the inverse prediction method. It uses backpropagation to decompose the output prediction of a neural network to a reference input. DeepLIFT explains the difference in output using the difference between the reference and actual input (Shrikumar, Greenside, & Kundaje, 2017). Layer-Wise Relevance Propagation (LRP) propagates signals backwards through a neural network, and it multiplies them with every convolutional layer activation. The output of LRP is a heatmap highlighting the most important pixels (features) for classification (Robeer, 2018; Samek et al., 2017). Sushil et al. (2018) trained the network to feed its output to RIPPER-k and generate rules explaining the network prediction outcomes. The next section will provide more details about this setup.

3. Experimental Setup

This section describes the dataset, software requirements, and it explains the workflow to answer RQs.

3.1 Data

The research project used the 20 Newsgroups dataset from scikit-learn. The dataset consists of about 20,000 documents almost evenly partitioned in 20 different Newsgroups (Pedregosa FABIANPEDREGOSA et al., 2011). Every group corresponds to a different topic. Some of the groups are closely related (e.g. 'Baseball' and 'Hockey'), while others are highly distanced from each other (e.g. 'the Windows X Toolkit' and 'Atheism'). There are about 535 training, 60 development and 395 testing instances for every newsgroup. The development instances were used to optimise the network (Sushil et al., 2018). Table 1 below gives an overview of all newsgroups arranged by subject area ('Home Page for 20 Newsgroups Data Set,' n.d.).

TABLE 1: 20 NEWSGROUPS BY SUBJECT

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

('Home Page for 20 Newsgroups Data Set,' n.d.)

The research project used two subsets with either 4 or 8 newsgroups from different subjects. The first subset incorporated 'Guns', 'Christian', 'Autos' and 'IBM PC Hardware' groups. Pairs of related classes were used for the second subset. There were 'Mideast' and 'Guns', 'Christian' and 'Atheism', 'Motorcycles' and 'Autos', 'Macintosh Hardware' and 'IBM PC Hardware.'

The motivation for using newgroups from different subject areas was to check whether rule induction algorithms generalise well across different classes and whether a higher number of closely related classes affect their performance (F score). The reason for using subsets instead of the entire dataset was to reduce complexity and the running time of experiments, since, even with 8 classes, the rule induction step can take hours on a general PC.

The data pre-processing included the same steps as in Sushil et al. (2018) work. Headers, footers, quotes and English stop words were removed. The rest of the workflow is described in the next section.

3.2 Methodology

3.2.1 Libraries, Software and Hardware

All experiments were conducted in Python 3.6.8 (Guido Rossum, 1995) from the Anaconda distribution version 1.8.7 (Anaconda Software Distribution, 2016). The research utilised the following libraries Numpy 1.17.2 (Van Der Walt, Colbert, & Varoquaux, 2011), Scipy 1.3.1 (Virtanen et al., 2019), Scikit-learn 0.19.1 (Pedregosa FABIANPEDREGOSA et al., 2011), Pytorch 0.4.0 (Paszke et al., 2017), Javabridge 1.0.18 ('javabridge: running and interacting with the JVM from Python - python-javabridge 1.0.12 documentation,' n.d.), WEKA 3.7 (Witten, Frank, Hall, & Pal, 2016), Python weka wrapper3 0.1.9 ('Introduction - python-weka-wrapper3 0.1.7 documentation,' n.d.).

Please mind version numbers since for example, newer versions of Pytorch may not function correctly with other libraries. Also note that Xcode with Command Line Tools (for Mac users) or Visual Studio Code (for Windows users) is required to use Javabridge library (Kelly & Nozzi, 2013). Moreover, the research project used FURIA 1.0.2, which works under the Weka library, but it is not part of it (J. Hühn & Hüllermeier, 2009). For the detailed set-up instructions, please refer to the project GitHub repository https://github.com/stepanov-a-o/interpret_with_rules_furia/blob/master/setup_instructions.md.

A thorough search of the literature and online forums yielded that there is no option to install FURIA using a command-line interface, which creates a barrier for running it on cloud servers, such as Google Colaboratory. Therefore, the parameters of the machine used to run all the experiments are given: Processor 2,8 GHz Dual-Core Intel Core i7, Memory 16GB 1600 MHz, GPU support is not available.

3.2.2 Workflow for Research Question 1

To compare classification accuracy, the project conducted two experiments with RIPPER-k, FURIA and the neural network. The first experiment was with 4 classes, and the second experiment was with 8 classes. Both experiments used the original data with minor preprocessing (as described in Data section). Rule induction algorithms and the network ran independently on the same subsets of the dataset, solving the same multiclassification problem. Their corresponding accuracy scores were recorded and compared.

The project used RIPPER-k implementation (‘JRip’) from the WEKA framework, FURIA implementation from the University of Waikato repository, and the neural network from the paper by Sushil et al. (2018). The rule induction principle of RIPPER-k and FURIA, their hyperparameters and the structure of the neural network are outlined below.

Rule Induction

The rule induction with RIPPER-k has a growing and optimisation phases. The growing phase starts by splitting the training data into growing and pruning sets. RIPPER-k uses the growing set to learn preconditions to create rules, and it uses the pruning set to simplify rules (Vasile, Silvescu, Kang, & Honavar, 2006). RIPPER-k learns rules in a one-vs-rest manner. It marks instances from one class as positive and instances from all other classes as negative. RIPPER-k starts learning with an empty rule, and it adds preconditions until the rule covers only positive instances, i.e. instances from the target class. Each new precondition (i) is selected using the information gain (IG) formula (defined below), where p_i and n_i represent the number of positive and negative instances covered after adding a new precondition, and p and n represent the number of positive and negative instances covered before adding the precondition (J. Hühn & Hüllermeier, 2009).

$$IG_i = p_i \times \left(\log_2\left(\frac{p_i}{p_i + n_i}\right) - \log_2\left(\frac{p}{p + n}\right)\right)$$

RIPPER-k tends to overfit the training data during rule growing. To prevent overfitting, it cuts rules to maximise their performance on the pruning set. The rule cutting is done by assessing all preconditions in the order they were learned. The goal is to find an optimal cutting point. The assessment criterion for the optimal point is v_i – the rule-value metric. It is defined as

$$v_i = \frac{p - n}{p + n}.$$

Preconditions allowing to maximise v_i are preserved, and all others are cut (J. Hühn & Hüllermeier, 2009)

The outcome of the growing phase is a ruleset. This ruleset serves as input for the optimisation phase where each rule is re-evaluated. For every rule (r), two new rules *r'* and *r''* are created. *r'* is a replacement (empty) rule. It is learned and pruned to minimise the error of the modified ruleset. *r''* is a revision rule. It is created in the same way as *r'*, but *r''* starts with *r'* instead of an empty rule. Afterwards, the Minimum Description Length (MDL) is used to select between *r'* and *r''* for the final ruleset (J. Hühn & Hüllermeier, 2009). The rule induction with FURIA is similar to the rule induction with RIPPER-k. The main differences were outlined in section 2.4.

Hyperparameters of RIPPER-k and FURIA

RIPPER-k and FURIA share most of the hyperparameters. This paragraph summarises them and outlines their default values. '-F' is the number of folds for error pruning, and it is equal to 3. '-N' is the minimum weights of instances, and it sets to 2.0. '-O' is the number of runs of optimisation. The default value for it is 2. '-D' is whether to run on the debug mode (default: false). '-S' is the seed, and it is equal to 1. '-E' is whether to check the error rate (>= 0.5) in stopping criteria (default: check) (Witten et al., 2016).

There is one difference between the hyperparameters of rule induction algorithms. RIPPER-k has a hyperparameter '-P', which refers to whether to use pruning (default: use). FURIA has a hyperparameter '-s', which refers to whether to use rule stretching (default: use) (Witten et al., 2016).

For the RQ1, the hyperparameters of rule induction algorithms were set to default values, except the number of runs of optimisation ('-O'). '-O' was set to 1 since the second and subsequent round of optimisation did not help to improve the performance. Cross-validation with 5 folds was used to assess the generalisation performance of rule induction algorithms.

The Neural Network

The neural network represented a structure with 2 hidden layers 100 units each, ReLU activation function, and a softmax output layer. The network used Adam optimiser, and it was trained for 50 epochs to minimise the cross-entropy loss (Sushil et al., 2018). The neural network was trained for 4 and 8 classes text classification task. Prior to training, the dataset was featured to corresponding TF-IDF values to get vocabulary (dictionary) and TF-IDF vectors (scipy sparse matrix).

3.2.3 Workflow for Research Question 2

To compare RIPPER-k and FURIA interpretability scores, the project used the same hyperparameter settings for rule induction algorithms and the neural network as described in the previous section. The neural network was trained on a classification task with 4 and 8 classes, and its output predictions were used as the input for RIPPER-k and FURIA algorithms. The algorithms' goal was to generate rules that capture the relationship between the most important input features and the class labels predicted by the trained network (Sushil et al., 2018). The interpretability (F score) and precision scores of RIPPER-k and FURIA were recorded and compared.

The working process precisely followed the steps from the original work by Sushil et al. (2018) with one modification. The authors of the original work approached a multiclass problem as a set of binary tasks. For a problem with 4 classes, the original work merged three categories and generated rules for one class at a time. This research project did not merge classes, and it induced rules concurrently for all of them.

The first step was to estimate the contribution of the input features towards the predicted output of the trained neural network. After that, the change in the predicted output by modifying the input features was computed, and the gradient was used to get features saliency scores. A negative saliency score means that there is a negative correlation, zero means that there is no correlation and a positive sign refers to a positive correlation.

The next step was to multiply saliency scores by the original input to get new input data reweighted according to the importance in the trained network. After that, it is required to simplify the new input data to their sign, i.e. -1, 0 or 1. The simplified input data makes rules more interpretable since it is easier to read and understand discrete instead of continuous feature values (Sushil et al., 2018).

The following step applied sensitivity analysis. It is an unsupervised technique that uses the trained network weights and the original input data, but not the labels, to reduce the feature space. The technique takes gradient values to find the most important features. The idea is to calculate gradients of all input nodes and use root mean square to obtain overall importance scores of all the features in the trained network. Features with the highest importance scores were marked as top features. One thousand top features were used to answer RQ2.

The final step was to select a subset of vocabulary corresponding to the top features. This step allows using keywords associated with features and makes rules interpretable (Sushil et al., 2018). The output from the above steps was used to train rule induction algorithms. After the training, RIPPER-k and FURIA were fitted with the test predictions of the neural network to induce rules and get interpretability scores.

3.2.4 Workflow for Research Question 3

The project followed the same steps as described in the previous section to study the impact of the number of input features on the interpretability score of RIPPER-k and FURIA. However, instead of 1,000 top features, rule induction algorithms used only 500 or 100 features to induce rules. The interpretability scores (F scores) gained with 500, and 100 features were compared with each other and with the results obtained with the experiments for RQ2.

The same workflow as for RQ2 was used to study the impact of the pruning hyperparameter. The default value of the pruning hyperparameter is equal to 3. The project tested values 4 and 6. All other parameters remained the same as in the workflow for RQ2.

3.3 Evaluation Methods

Accuracy was the evaluation metric for RQ1, and F score was the evaluation metric for RQ2 and RQ3. The precision score served as a supplement evaluation metric for RQ2. The accuracy tells us how well a rule induction algorithm or the network generalises on unseen data points. The F score refers to the extent to which a rule induction algorithm can interpret the output from the neural network. The precision score illustrates the reliability of rules in a ruleset (Sushil et al., 2018).

4. Results

This section has four sub-sections. Sub-section 4.1 presents results for RQ1, 4.2 for RQ2 and SQ2, sub-section 4.3 for RQ3 and sub-section 4.4 for SQ3. All experiments were repeated multiple times to ensure results consistency.

4.1 Classification Performance

The RIPPER-k, FURIA and neural network (NN) performance on classification tasks with 4 and 8 classes is summarised in Table 2 below. The neural network achieved the highest accuracy scores with a value of 90.26% for a problem with 4 classes and 77.84% for a task with 8 classes. FURIA obtained the second-highest accuracy scores with values of 74.57% and 62.24%, and RIPPER-k achieved the lowest accuracy scores with values of 68.89% and 59.65% respectively.

TABLE 2: NN, RIPPER-K AND FURIA CLASSIFICATION ACCURACIES

Number of Classes	RIPPER-k	FURIA	NN
4 classes	68.89%	74.57%	**90.26%**
8 classes	59.65%	62.24%	**77.84%**

*All rounding is to two decimal places.

To show that RIPPER-k and FURIA rules are understandable, Figure 2 and 3 illustrate classification rules produced by rule induction algorithms for the class 'Autos'. Those rules are subsets of the rulesets generated in the classification task with 4 classes. For complete rulesets, please refer to Appendix A.

In a ruleset, each row represents a rule. On the left-hand side of the arrow (=>) is the rule condition(-s), and on the right-hand side is the class. The discrete value next to a rule precondition refers to a correlation between a feature value and the probability of the class. The value 1 means a positive correlation, -1 refers to a negative correlation, and 0 represents an absence of a feature (Sushil et al., 2018). At the end of each rule, RIPPER-k has two figures in brackets; those figures represent the number of correctly/incorrectly covered instances by a rule. Rules with a high proportion of correctly covered instances are trustworthy. FURIA shows a confidence factor (CF) at the end of each rule. CF is the rule accuracy.

FIGURE 2: RIPPER-K RULES FOR AUTOS CLASS

```
(car = 1) => text_class = rec.autos (199.0/6.0)
(cars = 1) => text_class = rec.autos (45.0/0.0)
(god = 0) and (ford = 1) => text_class = rec.autos (16.0/0.0)
(god = 0) and (church = 0) and (engine = 1) => text_class = rec.autos (13.0/0.0)
(god = 0) and (people = 0) and (oil = 1) => text_class = rec.autos (11.0/1.0)
(god = 0) and (people = 0) and (air = 1) => text_class = rec.autos (9.0/0.0)
(god = 0) and (people = 0) and (toyota = 1) => text_class = rec.autos (9.0/0.0)

Number of rules: 7
```

*The rules induced on the original data.
*Numbers in brackets represent correctly/incorrectly covered instances by a rule.

FIGURE 3: FURIA RULES FOR AUTOS CLASS

```
(car = 1) and (gun = 0) and (think = 0) and (problem = 0) and (use = 0) =>
text_class=rec.autos (CF = 0.99)
(cars = 1) and (gun = 0) => text_class=rec.autos (CF = 0.97)
(car = 1) and (gun = 0) and (computer = 0) and (question = 0) and (486 = 0) =>
text_class=rec.autos (CF = 0.98)
(god = 0) and (ford = 1) => text_class=rec.autos (CF = 0.96)
(god = 0) and (know = 0) and (does = 0) and (engine = 1) and (32 = 0) =>
text_class=rec.autos (CF = 0.96)
(god = 0) and (does = 0) and (oil = 1) and (18 = 0) =>
text_class=rec.autos (CF = 0.94)
(god = 0) and (does = 0) and (toyota = 1) => text_class=rec.autos (CF = 0.95)

Number of rules: 7
```

*The rules induced on the original data.
*CF refers to a certainty factor of a rule on the scale between 0 and 1.

4.2 Interpretability Comparison

Table 3 shows RIPPER-k and FURIA scores to which they can interpret predictions of the neural network with 4 and 8 classes. RIPPER-k is slightly better than FURIA in both cases. The RIPPER-k F score with 4 classes is 0.84 (+0.03 to FURIA score), and it is 0.65 with 8 classes (+0.04 to FURIA score).

TABLE 3: RIPPER-K AND FURIA INTERPRETABILITY SCORES

	RIPPER-k		**FURIA**	
Number of Classes	**P**	**F**	**P**	**F**
4 classes	0.83	**0.84**	0.77	0.81
8 classes	0.68	**0.65**	0.83	0.61

*All results are with the 1,000 top features. All rounding is to two decimal places.
*Abbreviations: Precision (P) and F score (F).

Table 3 also reports precision (P) scores. RIPPER-k has a higher precision score, and therefore more reliable rules than FURIA in the case with 4 classes (+0.06 to FURIA score). However, FURIA rules are more reliable in the case with 8 classes (+0.15 to RIPPER-k score).

The next two pages show rules for ‘Guns’ and ‘Christian’ classes. The rules were induced on the testing data transformed according to the neural network weights. The rules for ‘Guns’ class were induced on the output of the neural network with 4 classes (Figure 4 & 5), and for ‘Christian’ class on the output with 8 classes (Figure 6 & 7). For complete rulesets, please refer to Appendix B.

FIGURE 4: RIPPER-K RULES FOR GUNS CLASS

```
(fbi = 1) => text_class=talk.politics.guns (84.0/0.0)
(gun = 1) => text_class=talk.politics.guns (45.0/2.0)
(government = 1) and (want = 0) => text_class=talk.politics.guns (28.0/2.0)
(weapons = 1) => text_class=talk.politics.guns (16.0/1.0)
(batf = 1) => text_class=talk.politics.guns (9.0/0.0)
(house = 1) and (book = 0) and (god = 0) and (pc = 0)
=> text_class=talk.politics.guns (11.0/2.0)

(guns = 1) => text_class=talk.politics.guns (8.0/3.0)
(doubt = -1) => text_class=talk.politics.guns (7.0/3.0)
(come = -1) => text_class=talk.politics.guns (7.0/3.0)
(right = -1) and (meaning = 1) => text_class=talk.politics.guns (3.0/0.0)
(people = 1) and (suicide = 1) => text_class=talk.politics.guns (3.0/0.0)

Number of rules: 11
```

*The rules induced on the testing data transformed according to the NN weights.
*Numbers in brackets represent correctly/incorrectly covered instances by a rule.

FIGURE 5: FURIA RULES FOR GUNS CLASS

```
(fbi = 1) => text_class=talk.politics.guns (CF = 0.98)
(gun = 1) => text_class=talk.politics.guns (CF = 0.98)
(government = 1) and (car = 0) => text_class=talk.politics.guns (CF = 0.97)
(weapons = 1) and (car = 0) => text_class=talk.politics.guns (CF = 0.96)
(house = 1) and (god = 0) and (church = 0)
=> text_class=talk.politics.guns (CF = 0.95)

(texas = 1) => text_class=talk.politics.guns (CF = 0.96)
(com = 1) and (jmd = 1) => text_class=talk.politics.guns (CF = 0.89)
(batf = 1) => text_class=talk.politics.guns (CF = 0.97)
(koresh = 1) => text_class=talk.politics.guns (CF = 0.97)
(compound = 1) => text_class=talk.politics.guns (CF = 0.96)
(com = 1) and (dave = 1) => text_class=talk.politics.guns (CF = 0.8)
(shot = 1) and (god = 0) => text_class=talk.politics.guns (CF = 0.95)
(guns = 1) => text_class=talk.politics.guns (CF = 0.96)
(like = 0) and (investigation = 1) => text_class=talk.politics.guns (CF = 0.89)
(law = 1) and (god = 0) and (long = 0) and (car = 0)
=> text_class=talk.politics.guns (CF = 0.95)

(waco = 1) => text_class=talk.politics.guns (CF = 0.95)
(country = 1) and (god = 0) => text_class=talk.politics.guns (CF = 0.9)
(assault = 1) => text_class=talk.politics.guns (CF = 0.94)
(article = -1) and (sin = 0) and (thanks = 0)
=> text_class=talk.politics.guns (CF = 0.89)
(know = 0) and (control = 1) and (drive = 0)
=> text_class=talk.politics.guns (CF = 0.95)
(arms = 1) => text_class=talk.politics.guns (CF = 0.91)

Number of rules: 21
```

*The rules induced on the testing data transformed according to the NN weights.
*CF refers to a certainty factor of a rule on the scale between 0 and 1.

FIGURE 6: RIPPER-K RULES FOR CHRISTIAN CLASS

```
(god = 1) => text_class=soc.religion.christian (183.0/11.0)
(church = 1) => text_class=soc.religion.christian (55.0/0.0)
(christian = 1) => text_class=soc.religion.christian (34.0/3.0)
(christians = 1) => text_class=soc.religion.christian (17.0/1.0)
(bible = 1) => text_class=soc.religion.christian (17.0/3.0)
(faith = 1) => text_class=soc.religion.christian (12.0/3.0)
(jesus = 1) => text_class=soc.religion.christian (11.0/1.0)
(heaven = 1) => text_class=soc.religion.christian (6.0/1.0)
(catholic = 1) => text_class=soc.religion.christian (5.0/0.0)
(word = 1) and (does = 1) => text_class=soc.religion.christian (3.0/0.0)
(marriage = 1) => text_class=soc.religion.christian (4.0/0.0)
 (revelation = 1) => text_class=soc.religion.christian (3.0/0.0)
(feel = 1) and (world = 1) => text_class=soc.religion.christian (3.0/0.0)
(understanding = 1) => text_class=soc.religion.christian (2.0/0.0)

Number of rules: 14
```

*The rules induced on the testing data transformed according to the NN weights.

FIGURE 7: FURIA RULES FOR CHRISTIAN CLASS

```
(god = 1) and (jesus = 1) => text_class=soc.religion.christian (CF = 0.97)
(god = 1) and (support = 0) and (christians = 1) =>
text_class=soc.religion.christian (CF = 0.94)
(christian = 1) and (church = 1) => text_class=soc.religion.christian (CF = 0.94)
(church = 1) and (interesting = 0) => text_class=soc.religion.christian (CF = 0.95)
(god = 1) and (life = 1) => text_class=soc.religion.christian (CF = 0.94)
(god = 1) and (religious = 0) and (christian = 1) =>
text_class=soc.religion.christian (CF = 0.96)
(jesus = 1) => text_class=soc.religion.christian (CF = 0.96)
(christian = 1) and (use = 0) => text_class=soc.religion.christian (CF = 0.9)
(god = 1) and (religious = 0) and (interesting = 0) =>
text_class=soc.religion.christian (CF = 0.9)
(lord = 1) and (world = 0) => text_class=soc.religion.christian (CF = 0.96)
(bible = 1) => text_class=soc.religion.christian (CF = 0.89)
(sin = 1) => text_class=soc.religion.christian (CF = 0.97)
(christ = 1) and (drive = 0) => text_class=soc.religion.christian (CF = 0.98)
(faith = 1) and (people = 0) => text_class=soc.religion.christian (CF = 0.91)
(homosexual = 1) => text_class=soc.religion.christian (CF = 0.95)
(doctrine = 1) => text_class=soc.religion.christian (CF = 0.9)
(christianity = 1) and (people = 0) => text_class=soc.religion.christian (CF = 0.96)
(scriptural = 1) => text_class=soc.religion.christian (CF = 0.84)
(translation = 1) => text_class=soc.religion.christian (CF = 0.88)
(clh = 1) => text_class=soc.religion.christian (CF = 0.69)

Number of rules: 20
```

*The rules induced on the testing data transformed according to the NN weights.

From the rulesets above and in the Appendix B, it can be observed that FURIA uses more preconditions for each rule than RIPPER-k. For example, majority of RIPPER-k rules (in Figures 4 & 6) have one precondition, e.g. (god = 1) => text_class=soc.religion.christian. Most of the FURIA rules (in Figures 5 & 7) have two preconditions, e.g. (god = 1) and (jesus = 1) => text_class=soc.religion.christian.

4.3 Interpretability and the Number of Input Features

Table 4 shows the relation between the number of input features and the interpretability score (F score) of RIPPER-k and FURIA trained on the output of the neural network with 4 and 8 classes.

TABLE 4: INTERPRETABILITY SCORE AND THE NUMBER OF INPUT FEATURES

	4 Classes		8 Classes	
Number of Input Features	**RIPPER-k**	**FURIA**	**RIPPER-k**	**FURIA**
100	0.64	**0.71**	**0.62**	0.55
500	0.78	**0.82**	0.58	**0.63**
1,000	**0.84**	0.81	**0.65**	0.61

*All rounding is to two decimal places.

Overall, a higher number of input features allows RIPPER-k to obtain a better interpretability score. The exception is the case with 8 classes and 500 input features, where the F score was lower than with 100 features. RIPPER-k achieved its absolute maximum interpretability scores with 1,000 input features.

FURIA obtained its best F scores with 500 features. To review associated rulesets, please refer to Appendix C. The appendix does not include RIPPER-k rulesets or other FURIA rulesets with a different number of input features due to their lower interpretability scores.

4.4 Interpretability and the Pruning Hyperparameter

Table 5 outlines the relationship between the pruning hyperparameter and the interpretability score. The pruning factor '-F' = 4 did not help RIPPER-k to achieve a higher interpretability score with 4 classes. However, it allowed getting a better score with 8 classes (+0.04) comparing to the value obtained with '-F' = 2 (default setting). With '-F' = 6, RIPPER-k's interpretability score was lower for both 4 and 8 classes than the values obtained with the default setting.

TABLE 5: INTERPRETABILITY SCORE AND PRUNING HYPERPARAMETER

	RIPPER-k			FURIA		
Number of Classes	**'-F' = 2**	**'-F' = 4**	**'-F' = 6**	**'-F' = 2**	**'-F' = 4**	**'-F' = 6**
4 classes	**0.84**	0.80	0.79	0.81	0.83	0.62
8 classes	0.65	0.69	0.60	0.61	**0.72**	0.51

*All results are with the 1,000 top features. All rounding is to two decimal places.
*'-F' = 2 is the default setting.

The pruning factor '-F' = 4 helped FURIA to obtain better interpretability scores with 4 and 8 classes. With 4 classes FURIA improved its score by 0.02, and with 8 classes its F score increased by 0.11. A subsequent increase of the pruning factor to '-F' = 6 did not help to improve FURIA's interpretability score.

5. Discussion

5.1 Research Questions and Findings

RQ1: *To what extent do RIPPER-k and FURIA rule induction algorithms replicate the performance of a feedforward neural network in a classification task with 4 and 8 classes?*

RIPPER-k and FURIA accuracy rates are well below the neural network results. For example, in the classification task with 4 classes, the neural network beat FURIA by 15.69% and RIPPER-k by 21.37%. Besides, the accuracy rate of the neural network can be further improved by changing the structure of the network and with careful hyperparameters tuning. It is not possible to adjust the structure of a rule induction algorithm, and hyperparameter tuning alone will not allow bridging the gap with the neural network.

With that said, rule induction algorithms generate rules which can be understood by lay users. For example, to describe the class 'Autos', the algorithms used words such as car (-s), engine and Toyota.

RQ2: *To what extent does FURIA outperform RIPPER-k in interpreting the predictions from a feedforward neural network trained with 4 and 8 classes?*

Opposite to the initial hypothesis and theoretical background, FURIA did not outperform RIPPER-k in interpreting the predictions from the neural network. RIPPER-k's interpretability score was higher than FURIA's F score by 0.03 with 4 classes and 0.04 with 8 classes. This research project did not find a clear explanation for this unexpected performance of FURIA. Therefore, it will be interesting to repeat the experiments with different datasets.

SQ2: *How understandable are the rules generated by RIPPER-k and FURIA? (subjective comparison)*

Most of the rules generated by RIPPER-k and FURIA are understandable. However, the term understandability is subjective. One person may think that rules are understandable, while another person can think that they do not make sense (Bibal & Frénay, 2016). Moreover, the judgement about rules understandability depends on the person ability to study and analyse individual rules (Croce, Rossini, & Basili, 2019). This task can be hard, especially when there are no clear boundaries between classes (Croce et al., 2019).

FURIA rules can be claimed as more understandable because RIPPER-k outputs rules according to class frequencies and the larger class has only an 'else' clause with it (Sushil et al., 2018). Hence, RIPPER-k does not generate understandable rules for the majority class. On the contrary, FURIA rules tend to be longer and therefore, can be considered as less understandable.

RQ3: *What is the impact of the number of input features on RIPPER-k's and FURIA's ability to interpret predictions from a feedforward neural network?*

In most of the cases, a higher number of input features has a positive impact on RIPPER-k's ability to interpret predictions of the neural network. On the contrary, more features are not always beneficial for FURIA. As it was shown in Table 4, FURIA achieved higher F scores with 500 than with 1,000 input features. At the same time, it is important to note that these observations hold for the specific dataset and may not apply to other problems. Also, the research project did not experiment with more than 1,000 input features.

SQ3: *What is the impact of the pruning hyperparameter on the interpretability score?*

As it was shown in Table 5, pruning can have both a positive and negative impact on the interpretability score. Within the scope of this research project, the pruning factor '-F' = 4 helped RIPPER-k and FURIA to improve their interpretability scores with 8 classes. The pruning factor '-F' = 6 had an apparent adverse effect on the interpretability score.

5.2 Running Time

This research project showed that rule induction algorithms are slow. For example, it took 53 minutes for RIPPER-k and 2 hours 51 minutes for FURIA to execute the classification task with 8 classes from RQ1. For comparison, the neural network executed the same task under 17 minutes.

Moreover, the hyperparameter tuning can have a dramatic impact on the running time of rule induction algorithms. For example, running time of RIPPER-k with 8 classes, 1,000 top input features and the pruning factor '-F' = 4 was 10 hours 54 minutes. FURIA running time under the same conditions was 15 hours and 45 minutes. Hence, the running time of rule induction algorithms can be unacceptable for many real-world applications.

5.3 Limitations

This research project has limitations. First of all, all experiments were conducted on one dataset, which means that the research results can be specific to the data. Also, the project explored a limited number of settings while answering RQ3. It analysed the impact of the number of input features and pruning hyperparameter on the interpretability score. At the same time, it is still interesting to evaluate the impact of other parameters, such as the minimum weights of instances.

The research methodology has its limitations. For example, RIPPER-k and FURIA are data mining algorithms. It means that they can only work with textual and numerical data, and it is not possible to use them, for example, with images. Besides, both rule induction algorithms are global explanation techniques, which means that it is not possible to use them to interpret a single instance/decision.

FURIA has its specific limitations. It is hard to set up. FURIA is not part of the WEKA library by default, and the algorithm requires technical knowledge and access to a computer file system to install it. This factor prevents from running FURIA on cloud servers such as Google Colaboratory. Cloud servers with GPU power would allow running experiments on more massive sets of data, in a shorter time, and test different settings simultaneously. Another limiting factor for using FURIA is the lack of information online. If there is a problem, it is almost impossible to find an answer on the internet, and there is a relatively small community of experts who have the necessary knowledge and can help.

6. Future Work and Conclusion

6.1 Future Work

For future research, it will be interesting to repeat the experiments from RQ2 on other datasets. It will allow drawing a stronger conclusion about RIPPER-k and FURIA capabilities, and possibly identity cases when one algorithm interprets the neural network better than the other.

It is also desirable to setup FURIA on a more powerful computer or server and run experiments with other combinations of hyperparameters and explore their impact on the interpretability score. The research results showed that pruning can help FURIA to achieve higher interpretability. Therefore, it is interesting to study other hyperparameters and their impact on the interpretability score.

6.2 Conclusion

Rule induction algorithms such as RIPPER-k and FURIA cannot replicate the classification performance of a feedforward neural network. Besides, their running time can be unacceptably long for many tasks. The advantage of rule induction algorithms over the feedforward neural network is that they output sets of decision rules which can be understood by non-technical users.

This research project used the classification output from the feedforward neural network as an input for RIPPER-k and FURIA. The goal was to generate sets of rules interpreting predictions of the neural network. The initial hypothesis was that FURIA would be better than RIPPER-k in interpreting the output of the network (have higher F score). The reason for this hypothesis was that opposite to RIPPER-k, FURIA does not have a default rule, and it generates an unordered rule set. The results on the 20 Newsgroups scikit-learn dataset did not confirm the hypothesis. Possibly, FURIA's performance is subject to the specific dataset and therefore testing on other datasets is required.

The project compared how understandable rulesets generated by RIPPER-k and FURIA are. Overall, FURIA ruleset is more understandable because it provides rules for all classes. RIPPER-k rules are more concise than FURIA rules, but RIPPER-k only provides the default rule (else clause) for the majority class.

The project also compared the interpretability score of RIPPER-k and FURIA with the different number of input features from the neural network and tested several values of pruning hyperparameter. The result is that more input features do not necessarily provide the highest interpretability score. Moreover, additional pruning can help rule induction algorithms to interpret the output of the neural network better.

Bibliography

Anaconda Software Distribution. (2016). *Frequently asked questions — Anaconda documentation*. Retrieved from https://anaconda.com/

Asadi, S., & Shahrabi, J. (2016). ACORI: A novel ACO algorithm for rule induction. *Knowledge-Based Systems*, *97*, 175–187. https://doi.org/10.1016/j.knosys.2016.01.005

Bibal, A., & Frénay, B. (2016). *Interpretability of Machine Learning Models and Representations: an Introduction Interpretability of Nonlinear Dimensionality Reduction Mappings View project Machine Learning and Formal Verification View project Interpretability of Machine Learning Models and Representations: an Introduction*. Retrieved from http://www.i6doc.com/en/.

Cohen, W. (1995). *Fast Effective Rule Induction*.

Croce, D., Rossini, D., & Basili, R. (2019). *Auditing Deep Learning processes through Kernel-based Explanatory Models*.

Edwards, L., & Veale, M. (2017). Slave to the Algorithm: Why a Right to an Explanation Is Probably Not the Remedy You Are Looking for. *Duke Law & Technology Review*, *16*. Retrieved from https://heinonline.org/HOL/Page?handle=hein.journals/dltr16&id=18&div=3&collection=journals

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2019). Explaining explanations: An overview of interpretability of machine learning. *Proceedings - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA 2018*, 80–89. https://doi.org/10.1109/DSAA.2018.00018

Góra, G., & Wojna, A. (2002). *Local Attribute Value Grouping for Lazy Rule Induction Infobright Approximate Query View project Rseslib 3-rough set and machine learning open source in Java View project Local Attribute Value Grouping for Lazy Rule Induction*. https://doi.org/10.1007/3-540-45813-1_53

Guido Rossum. (1995). Welcome to Python.org. Retrieved November 30, 2019, from https://www.python.org/

Hagras, H. (2018). Toward Human-Understandable, Explainable AI. *Computer*, *51*(9), 28–36. https://doi.org/10.1109/MC.2018.3620965

Home Page for 20 Newsgroups Data Set. (1994). Retrieved from http://qwone.com/~jason/20Newsgroups/

Hühn, J. C. (2009). WAIKATO ENVIRONMENT FOR KNOWLEDGE ANALYSIS. Retrieved from http://weka.sourceforge.net/packageMetaData/fuzzyUnorderedRuleInduction/index.html

Hühn, J., & Hüllermeier, E. (2009). FURIA: an algorithm for unordered fuzzy rule induction. *Data Min Knowl Disc*, *19*, 293–319. https://doi.org/10.1007/s10618-009-0131-8

Introduction — python-weka-wrapper3 0.1.7 documentation. (n.d.). Retrieved November 8, 2019, from https://fracpete.github.io/python-weka-wrapper3/index.html

javabridge: running and interacting with the JVM from Python — python-javabridge 1.0.12 documentation. (n.d.). Retrieved November 30, 2019, from https://pythonhosted.org/javabridge/

Kelly, M., & Nozzi, J. (2013). *Mastering Xcode: Develop and Design*. Retrieved from https://books.google.nl/books?hl=en&lr=&id=tXwQAAAAQBAJ&oi=fnd&pg=PR6&dq=Xcode&ots=bdyU_egVc0&sig=wAz7JSHraO0xUVcEkKpep14FvGA&redir_esc=y#v=onepage&q=Xcode&f=false

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). *Text Classification Algorithms: A Survey*. https://doi.org/10.3390/info10040150

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 1675–1684. https://doi.org/10.1145/2939672.2939874

Letham, B., Rudin, C., McCormick, T. H., & Madigan, D. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, *9*(3), 1350–1371. https://doi.org/10.1214/15-AOAS848

Malioutov, D. M., Varshney, K. R., Emad, A., & Dash, S. (2017). *Learning Interpretable Classification Rules with Boolean Compressed Sensing*. https://doi.org/10.1007/978-3-319-54024-5_5

Marcus, G., thank Christina, I., Chollet, F., Davis, E., Lipton, Z., Pacifico, S., … Vouloumanos, A. (2018). *Deep Learning: A Critical Appraisal*. Retrieved from http://www.nytimes.com/2012/11/24/science/scientists-see-advances-in-deep-learning-a-part-of-artificial-

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., Facebook, Z. D., … Lerer, A. (2017). *Automatic differentiation in PyTorch*.

Pedregosa FABIANPEDREGOSA, F., Michel, V., Grisel OLIVIERGRISEL, O., Blondel, M., Prettenhofer, P., Weiss, R., … Duchesnay EDOUARDDUCHESNAY, Fré. (2011). Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot. In *Journal of Machine Learning Research* (Vol. 12). Retrieved from http://scikit-learn.sourceforge.net.

Poerner, N., Roth, B., Schütze, H., & Schütze, S. (n.d.). *Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement*.

Preece, A. (2018). Asking 'Why' in AI: Explainability of intelligent systems - perspectives and challenges. *Intelligent Systems in Accounting, Finance and Management*, *25*(2), 63–72. https://doi.org/10.1002/isaf.1422

Robeer, M. J. (2018). *Contrastive explanation for machine learning (Master's thesis)*. Retrieved from https://dspace.library.uu.nl/handle/1874/368081

Samek, W., Wiegand, T., & Müller, K.-R. (2017). *EXPLAINABLE ARTIFICIAL INTELLIGENCE: UNDERSTANDING, VISUALIZING AND INTERPRETING DEEP LEARNING MODELS*.

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). *Learning Important Features Through Propagating Activation Differences*. Retrieved from http://goo.gl/qKb7pL,

Sidath, A. (2018). Machine Learning Classifiers - Towards Data Science. Retrieved December 2, 2019, from https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

Stoklasa, J., Talášek, T., & Musilová, J. (2014). *FUZZY APPROACH-A NEW CHAPTER IN THE METHODOLOGY OF PSYCHOLOGY? 1*. https://doi.org/10.2478/s13374-014-0219-8

Sushil, M., Šuster, S., & Daelemans, W. (2018). *Rule induction for global explanation of trained models*. 82–97. https://doi.org/10.18653/v1/w18-5411

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, *13*(2), 22–30. https://doi.org/10.1109/MCSE.2011.37

Vasile, F., Silvescu, A., Kang, D.-K., & Honavar, V. (2006). *TRIPPER: Rule learning using taxonomies*.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … van Mulbregt, P. (2019). *SciPy 1.0-Fundamental Algorithms for Scientific Computing in Python*.

Wang, T., Rudin, C., Liu, Y., Klampfl, E., & Macneille, P. (2017). A Bayesian Framework for Learning Rule Sets for Interpretable Classification. In *Journal of Machine Learning Research* (Vol. 18). Retrieved from http://jmlr.org/papers/v18/16-003.html.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. Retrieved from https://www.elsevier.com

Appendix A

RIPPER-k Ruleset with 4 Classes

(gun = 1) => text_class = talk.politics.guns (147.0/7.0)
(government = 1) and (state = 0) and (far = 0) => text_class=talk.politics.guns (48.0/5.0)
(fbi = 1) => text_class = talk.politics.guns (36.0/2.0)
(weapons = 1) => text_class = talk.politics.guns (30.0/2.0)
(guns = 1) => text_class = talk.politics.guns (26.0/1.0)
(god = 0) and (car = 0) and (federal = 1) => text_class = talk.politics.guns (13.0/0.0)
(god = 0) and (car = 0) and (firearms = 1) => text_class = talk.politics.guns (10.0/0.0)
(does = 0) and (davidians = 1) => text_class = talk.politics.guns (7.0/1.0)
(god = 0) and (weapon = 1) => text_class = talk.politics.guns (8.0/0.0)
(does = 0) and (atf = 1) => text_class = talk.politics.guns (7.0/0.0)
(does = 0) and (handheld = 1) => text_class = talk.politics.guns (6.0/0.0)

(card = 1) => text_class = comp.sys.ibm.pc.hardware (123.0/6.0)
(drive = 1) and (car = 0) and (drives = 1) => text_class = comp.sys.ibm.pc.hardware (34.0/0.0)
(pc = 1) => text_class = comp.sys.ibm.pc.hardware (64.0/2.0)
(people = 0) and (thanks = 1) and (car = 0) and (drive = 1) => text_class = comp.sys.ibm.pc.hardware (22.0/2.0) (dos = 1) => text_class = comp.sys.ibm.pc.hardware (31.0/0.0)
(bus = 1) => text_class = comp.sys.ibm.pc.hardware (27.0/2.0)
(people = 0) and (car = 0) and (windows = 1) => text_class = comp.sys.ibm.pc.hardware (20.0/1.0)
(486 = 1) => text_class = comp.sys.ibm.pc.hardware (17.0/0.0)
(thanks = 1) and (car = 0) and (email = 1) => text_class = comp.sys.ibm.pc.hardware (14.0/3.0)
(people = 0) and (port = 1) => text_class = comp.sys.ibm.pc.hardware (14.0/0.0)
(god = 0) and (monitor = 1) => text_class = comp.sys.ibm.pc.hardware (17.0/2.0)
(motherboard = 1) => text_class = comp.sys.ibm.pc.hardware (12.0/0.0)
(god = 0) and (car = 0) and (board = 1) => text_class = comp.sys.ibm.pc.hardware (13.0/3.0)

(car = 1) => text_class = rec.autos (199.0/6.0)
(cars = 1) => text_class = rec.autos (45.0/0.0)
(god = 0) and (ford = 1) => text_class = rec.autos (16.0/0.0)
(god = 0) and (church = 0) and (engine = 1) => text_class = rec.autos (13.0/0.0)
(god = 0) and (people = 0) and (oil = 1) => text_class = rec.autos (11.0/1.0)
(god = 0) and (people = 0) and (air = 1) => text_class = rec.autos (9.0/0.0)
(god = 0) and (people = 0) and (toyota = 1) => text_class = rec.autos (9.0/0.0)

=> text_class = soc.religion.christian (1048.0/529.0)

Number of Rules: 32

RIPPER-k Ruleset with 8 Classes

(atheism = 1) => text_class = alt.atheism (54.0/8.0)
(bobby = 1) => text_class = alt.atheism (23.0/2.0)
(morality = 1) => text_class = alt.atheism (28.0/5.0)
(argument = 1)and (arguments = 1) and (way = 0) => text_class = alt.atheism (9.0/1.0)
(islamic = 1) and (israel = 0) => text_class = alt.atheism (33.0/9.0)
(bobbe = 1) => text_class = alt.atheism (11.0/0.0)
(atheists = 1) => text_class = alt.atheism (28.0/10.0)
(deletion = 1) => text_class = alt.atheism (15.0/5.0)
(thanks = 0) and (humans = 1) and (people = 0) => text_class = alt.atheism (13.0/3.0)
(kent = 1) => text_class=alt.atheism (12.0/5.0)

(gun = 1) => text_class = talk.politics.guns (171.0/26.0)
(fbi = 1) => text_class = talk.politics.guns (47.0/6.0)
(weapons = 1) and (armenian = 0) and (israel = 0) and (bike = 0) =>
text_class=talk.politics.guns (39.0/5.0)
(firearms = 1) => text_class = talk.politics.guns (20.0/2.0)
(constitution = 1) => text_class = talk.politics.guns (24.0/8.0)
(guns = 1) => text_class = talk.politics.guns (28.0/9.0)
(atf = 1) => text_class = talk.politics.guns (11.0/0.0)
(batf = 1) => text_class = talk.politics.guns (8.0/1.0)
(does = 0) and (handheld = 1) => text_class = talk.politics.guns (6.0/0.0)
(davidians = 1) => text_class = talk.politics.guns (9.0/2.0)
(weapon = 1) and (israel = 0) => text_class = talk.politics.guns (8.0/1.0)
(waco = 1) and (god = 0) => text_class = talk.politics.guns (13.0/3.0)
(does = 0) and (hunting = 1) => text_class = talk.politics.guns (6.0/1.0)
(think = 0) and (bd = 1) => text_class = talk.politics.guns (7.0/1.0)

(israel = 1) and (christ = 0) => text_class = talk.politics.mideast (135.0/6.0)
(turkish = 1) and (christians = 0) => text_class = talk.politics.mideast (58.0/0.0)
(arab = 1) => text_class = talk.politics.mideast (24.0/0.0)
(armenians = 1) => text_class = talk.politics.mideast (30.0/0.0)
(israeli = 1) => text_class = talk.politics.mideast (22.0/2.0)
(jews = 1) and (god = 0) and (jesus = 0) => text_class = talk.politics.mideast (32.0/11.0)
(turkey = 1) => text_class = talk.politics.mideast (9.0/0.0)
(lebanon = 1) => text_class = talk.politics.mideast (6.0/1.0)
(serdar = 1) => text_class = talk.politics.mideast (7.0/0.0)
(palestinian = 1) and (don = 0) => text_class = talk.politics.mideast (5.0/0.0)

(mac = 1) and (5mb = 0) and (clinton = 0) and (controller = 0) =>
text_class=comp.sys.mac.hardware (139.0/4.0)
(apple = 1) => text_class = comp.sys.mac.hardware (92.0/11.0)
(centris = 1) => text_class = comp.sys.mac.hardware (21.0/0.0)
(simms = 1) and (486 = 0) => text_class = comp.sys.mac.hardware (27.0/7.0)
(iisi = 1) => text_class = comp.sys.mac.hardware (10.0/0.0)
(scsi = 1) and (ide = 0) and (cd = 0) and (controller = 0) and (card = 0) => text_class =
comp.sys.mac.hardware (26.0/6.0)
(people = 0) and (040 = 1) => text_class=comp.sys.mac.hardware (9.0/0.0)
(lciii = 1) => text_class=comp.sys.mac.hardware (11.0/0.0)
(powerbook = 1) => text_class=comp.sys.mac.hardware (9.0/0.0)
(quadra = 1) and (16 = 0) => text_class=comp.sys.mac.hardware (9.0/0.0)
(people = 0) and (duo = 1) => text_class=comp.sys.mac.hardware (4.0/0.0)
(nubus = 1) => text_class=comp.sys.mac.hardware (7.0/0.0)
(internal = 1) and (use = 0) and (drive = 1) => text_class=comp.sys.mac.hardware
(8.0/1.0)
(people = 0) and (macs = 1) => text_class=comp.sys.mac.hardware (4.0/0.0)

```
(card = 1) => text_class=comp.sys.ibm.pc.hardware (126.0/15.0)
(pc = 1) => text_class=comp.sys.ibm.pc.hardware (66.0/7.0)
(dos = 1) => text_class=comp.sys.ibm.pc.hardware (40.0/1.0)
(thanks = 1) and (drive = 1) and (car = 0) => text_class=comp.sys.ibm.pc.hardware
(29.0/5.0)
(486 = 1) => text_class=comp.sys.ibm.pc.hardware (29.0/2.0)
(bios = 1) => text_class=comp.sys.ibm.pc.hardware (22.0/0.0)
(thanks = 1) and (connect = 1) => text_class=comp.sys.ibm.pc.hardware (6.0/0.0)
(ide = 1) => text_class=comp.sys.ibm.pc.hardware (16.0/0.0)
(monitor = 1) and (video = 0) => text_class=comp.sys.ibm.pc.hardware (21.0/5.0)
(using = 1) and (windows = 1) => text_class=comp.sys.ibm.pc.hardware (4.0/0.0)
(bus = 1) and (bike = 0) and (cars = 0) => text_class=comp.sys.ibm.pc.hardware
(18.0/5.0)
(port = 1) => text_class=comp.sys.ibm.pc.hardware (11.0/4.0)
(computer = 1) and (mail = 1) => text_class=comp.sys.ibm.pc.hardware (6.0/1.0)
(drives = 1) and (problem = 1) => text_class=comp.sys.ibm.pc.hardware (5.0/0.0)
(vlb = 1) => text_class=comp.sys.ibm.pc.hardware (6.0/0.0)
(gateway = 1) => text_class=comp.sys.ibm.pc.hardware (6.0/0.0)

(car = 1) and (bike = 0) => text_class=rec.autos (216.0/24.0)
(cars = 1) and (bike = 0) => text_class=rec.autos (58.0/8.0)
(god = 0) and (ford = 1) => text_class=rec.autos (12.0/0.0)
(god = 0) and (bike = 0) and (toyota = 1) => text_class=rec.autos (10.0/0.0)
(god = 0) and (bike = 0) and (dealer = 1) => text_class=rec.autos (16.0/5.0)
(god = 0) and (bike = 0) and (oil = 1) => text_class=rec.autos (16.0/5.0)
(god = 0) and (vw = 1) => text_class=rec.autos (8.0/1.0)
(god = 0) and (wagon = 1) => text_class=rec.autos (6.0/0.0)

(bike = 1) => text_class=rec.motorcycles (163.0/1.0)
(dod = 1) => text_class=rec.motorcycles (36.0/1.0)
(motorcycle = 1) => text_class=rec.motorcycles (23.0/0.0)
(god = 0) and (helmet = 1) => text_class=rec.motorcycles (15.0/0.0)
(people = 0) and (ride = 1) => text_class=rec.motorcycles (17.0/1.0)
(god = 0) and (bikes = 1) => text_class=rec.motorcycles (13.0/0.0)
(god = 0) and (motorcycles = 1) => text_class=rec.motorcycles (9.0/1.0)
(god = 0) and (riding = 1) => text_class=rec.motorcycles (11.0/2.0)
(god = 0) and (harley = 1) => text_class=rec.motorcycles (8.0/0.0)
(god = 0) and (bmw = 1) => text_class=rec.motorcycles (16.0/5.0)
(god = 0) and (dog = 1) => text_class=rec.motorcycles (10.0/1.0)

=> text_class=soc.religion.christian (1693.0/1216.0)
```

Number of Rules: 84

FURIA Ruleset with 4 Classes

(card = 1) and (know = 0) and (atf = 0) and (happened = 0) =>
text_class=comp.sys.ibm.pc.hardware (CF = 0.98)
(pc = 1) and (asks = 0) and (woman = 0) =>
text_class=comp.sys.ibm.pc.hardware (CF = 0.98)
(thanks = 1) and (drive = 1) and (read = 0) and (auto = 0) =>
text_class=comp.sys.ibm.pc.hardware (CF = 0.97)
(people = 0) and (dos = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.97)
(people = 0) and (bus = 1) and (cars = 0) and (radar = 0) =>
text_class=comp.sys.ibm.pc.hardware (CF = 0.98)
(people = 0) and (thanks = 1) and (car = 0) and (help = 1) and (church = 0) and (children
= 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.97)
(people = 0) and (monitor = 1) and (die = 0) => text_class=comp.sys.ibm.pc.hardware
(CF = 0.97)
(drives = 1) and (drive = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.97)
(486 = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.97)
(people = 0) and (port = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.97)
(people = 0) and (car = 0) and (windows = 1) and (crime = 0) =>
text_class=comp.sys.ibm.pc.hardware (CF = 0.97)
(motherboard = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.97)
(don = 0) and (disk = 1) and (abs = 0) and (american = 0) =>
text_class=comp.sys.ibm.pc.hardware (CF = 0.97)

(car = 1) and (gun = 0) and (think = 0) and (problem = 0) and (use = 0)
=> text_class=rec.autos (CF = 0.99)
(cars = 1) and (gun = 0) => text_class=rec.autos (CF = 0.97)
(car = 1) and (gun = 0) and (computer = 0) and (question = 0) and (486 = 0)
=> text_class=rec.autos (CF = 0.98)
(god = 0) and (ford = 1) => text_class=rec.autos (CF = 0.96)
(god = 0) and (know = 0) and (does = 0) and (engine = 1) and (32 = 0)
=> text_class=rec.autos (CF = 0.96)
(god = 0) and (does = 0) and (oil = 1) and (18 = 0) => text_class=rec.autos (CF = 0.94)
(god = 0) and (does = 0) and (toyota = 1) => text_class=rec.autos (CF = 0.95)

(god = 1) => text_class=soc.religion.christian (CF = 0.92)
(bible = 1) => text_class=soc.religion.christian (CF = 0.99)
(jesus = 1) => text_class=soc.religion.christian (CF = 0.99)
(church = 1) => text_class=soc.religion.christian (CF = 0.95)
(christians = 1) => text_class=soc.religion.christian (CF = 0.98)
(christian = 1) => text_class=soc.religion.christian (CF = 0.93)
(christ = 1) and (way = 0) => text_class=soc.religion.christian (CF = 0.97)
(christianity = 1) => text_class=soc.religion.christian (CF = 0.98)
(jewish = 1) => text_class=soc.religion.christian (CF = 0.93)
(marriage = 1) => text_class=soc.religion.christian (CF = 0.94)

(gun = 1) => text_class=talk.politics.guns (CF = 0.94)
(government = 1) => text_class=talk.politics.guns (CF = 0.84)
(fbi = 1) and (position = 0) => text_class=talk.politics.guns (CF = 0.97)
(weapons = 1) => text_class=talk.politics.guns (CF = 0.91)
(guns = 1) => text_class=talk.politics.guns (CF = 0.97)
(does = 0) and (compound = 1) => text_class=talk.politics.guns (CF = 0.9)
(god = 0) and (car = 0) and (drive = 0) and (firearms = 1)
=> text_class=talk.politics.guns (CF = 0.97)
(god = 0) and (car = 0) and (rights = 1) => text_class=talk.politics.guns (CF = 0.93)

Number of Rules: 38

FURIA Ruleset with 8 Classes

(atheism = 1) => text_class=alt.atheism (CF = 0.83)
(religion = 1) and (did = 0) and (Christian = 0) and (right = 0) and (freedom = 0) and (person = 0) and (understanding = 0) and (live = 0) and (love = 0) => text_class=alt.atheism (CF = 0.9)
(bobby = 1) => text_class=alt.atheism (CF = 0.88)
(morality = 1) => text_class=alt.atheism (CF = 0.78)
(islamic = 1) and (jews = 0) => text_class=alt.atheism (CF = 0.73) (atheists = 1) => text_class=alt.atheism (CF = 0.75)
(kent = 1) and (cheers = 1) => text_class=alt.atheism (CF = 0.9)
(bobbe = 1) => text_class=alt.atheism (CF = 0.9)

(card = 1) and (controller = 1) and (610 = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(card = 1) and (people = 0) and (mac = 0) and (nubus = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.8)
(pc = 1) and (mac = 0) and (apple = 0) and (think = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.92)
(dos = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(drive = 1) and (controller = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.92)
(486 = 1) and (speed = 0) and (accepted = 0)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(ide = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(people = 0) and (bus = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.81)
(people = 0) and (bios = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.94)
(thanks = 1) and (connect = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.64)
(people = 0) and (vlb = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(people = 0) and (port = 1) and (mac = 0) and (computer = 0) and (hardware = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.79)
(people = 0) and (drives = 1) and (disks = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.83)
(people = 0) and (gateway = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.93)

(mac = 1) and (controller = 0) and (clinton = 0) and (hard = 0) and (agents = 0) and (sin = 0) and (mhz = 0) => text_class=comp.sys.mac.hardware (CF = 0.99)
(apple = 1) and (did = 0) and (controller = 0) => text_class=comp.sys.mac.hardware (CF = 0.94)
(centris = 1) => text_class=comp.sys.mac.hardware (CF = 0.96)
(iisi = 1) => text_class=comp.sys.mac.hardware (CF = 0.94)
(simms = 1) and (486 = 0) and (perfect = 0) and (port = 0) => text_class=comp.sys.mac.hardware (CF = 0.87)
(people = 0) and (powerbook = 1) => text_class=comp.sys.mac.hardware (CF = 0.94)
(thanks = 1) and (lc = 1) => text_class=comp.sys.mac.hardware (CF = 0.81)
(people = 0) and (se = 1) and (car = 0) and (took = 0) and (don = 0) and (14 = 0) => text_class=comp.sys.mac.hardware (CF = 0.95)
(people = 0) and (drive = 1) and (internal = 1) and (thanks = 0) and (000 = 0) => text_class=comp.sys.mac.hardware (CF = 0.9)
(people = 0) and (quadra = 1) and (5mb = 0) => text_class=comp.sys.mac.hardware (CF = 0.95)
(people = 0) and (lciii = 1) => text_class=comp.sys.mac.hardware (CF = 0.9)
(people = 0) and (scsi = 1) and (controller = 0) and (cd = 0) and (ide = 0) and (card = 0) and (5mb = 0) and (bus = 0) and (32 = 0) and (according = 0) => text_class=comp.sys.mac.ha rdware (CF = 0.95)
(people = 0) and (don = 0) and (nubus = 1) => text_class=comp.sys.mac.hardware (CF = 0.87)
(people = 0) and (040 = 1) => text_class=comp.sys.mac.hardware (CF = 0.88)

(car = 1) and (people = 0) and (bike = 0) and (set = 0) and (dealer = 1) => text_class=rec.autos (CF = 0.94)
(car = 1) and (people = 0) and (bike = 0) and (set = 0) and (person = 0) and (use = 0) and (jim = 0) and (ride = 0) and (driver = 0) and (law = 0) and (drivers = 0) and (accept = 0) and (action = 0) and (affairs = 0) and (child = 0) => text_class=rec.autos (CF = 0.99)
(cars = 1) and (way = 0) and (police = 0) and (bike = 0) and (does = 0) and (isn = 0) and (road = 0) and (people = 0) and (majority = 0) => text_class=rec.autos (CF = 0.97)
(car = 1) and (engine = 1) and (bike = 0) and (happened = 0) => text_class=rec.autos (CF = 0.95)
(car = 1) and (say = 0) and (bike = 0) and (drive = 1) and (person = 0) and (called = 0) => text_class=rec.autos (CF = 0.92)
(people = 0) and (ford = 1) and (car = 0) => text_class=rec.autos (CF = 0.88)
(people = 0) and (toyota = 1) => text_class=rec.autos (CF = 0.93)
(people = 0) and (oil = 1) and (bike = 0) and (world = 0) and (don = 0) and (engine = 0) and (internal = 0) => text_class=rec.autos (CF = 0.9)
(does = 0) and (wagon = 1) => text_class=rec.autos (CF = 0.84)
(does = 0) and (god = 0) and (gt = 1) and (16 = 0) => text_class=rec.autos (CF = 0.91)
(does = 0) and (god = 0) and (vw = 1) and (16 = 0) => text_class=rec.autos (CF = 0.87)
(people = 0) and (does = 0) and (chevy = 1) => text_class=rec.autos (CF = 0.84)

(bike = 1) and (accepted = 0) and (civilian = 0) => text_class=rec.motorcycles (CF = 0.99)
(dod = 1) and (death = 0) and (accepted = 0) => text_class=rec.motorcycles (CF = 0.98)
(motorcycle = 1) and (accounts = 0) => text_class=rec.motorcycles (CF = 0.97)
(ride = 1) and (use = 0) and (cars = 0) and (pray = 0) => text_class=rec.motorcycles (CF = 0.97)
(helmet = 1) => text_class=rec.motorcycles (CF = 0.96)
(people = 0) and (bikes = 1) => text_class=rec.motorcycles (CF = 0.97)
(people = 0) and (riding = 1) and (think = 0) and (act = 0) => text_class=rec.motorcycles (CF = 0.96)
(people = 0) and (use = 0) and (dog = 1) and (com = 0) and (accept = 0) => text_class=rec.motorcycles (CF = 0.9)
(people = 0) and (harley = 1) => text_class=rec.motorcycles (CF = 0.92)

(god = 1) and (christ = 1) => text_class=soc.religion.christian (CF = 0.93)
(god = 1) and (christians = 1) and (example = 0) and (does = 1) => text_class=soc.religion.christian (CF = 0.94)
(god = 1) and (atheism = 0) and (christian = 1) and (world = 0) and (mary = 0) => text_class=soc.religion.christian (CF = 0.95)
(god = 1) and (course = 0) and (lord = 1) and (people = 1) => text_class=soc.religion.christian (CF = 0.92)
(god = 1) and (atheism = 0) and (church = 1) => text_class=soc.religion.christian (CF = 0.91)
(god = 1) and (fact = 0) and (paul = 1) => text_class=soc.religion.christian (CF = 0.93)
(god = 1) and (right = 0) and (heaven = 1) and (bobby = 0) => text_class=soc.religion.christian (CF = 0.96)
(jesus = 1) and (god = 0) => text_class=soc.religion.christian (CF = 0.83)
(god = 1) and (make = 0) and (mean = 0) and (question = 1) => text_class=soc.religion.christian (CF = 0.92)
(god = 1) and (love = 1) and (believe = 0) => text_class=soc.religion.christian (CF = 0.85)
(church = 1) and (government = 0) => text_class=soc.religion.christian (CF = 0.88)
(god = 1) and (fact = 0) and (did = 1) => text_class=soc.religion.christian (CF = 0.82)
(christian = 1) and (live = 0) and (believe = 0) and (right = 0) and (given = 0) => text_class=soc.religion.christian (CF = 0.77)
(christians = 1) and (people = 0) and (jew = 0) => text_class=soc.religion.christian (CF = 0.89)

(god = 1) and (understanding = 1) => text_class=soc.religion.christian (CF = 0.89)
(christ = 1) and (way = 0) => text_class=soc.religion.christian (CF = 0.94)
(bible = 1) and (does = 0) => text_class=soc.religion.christian (CF = 0.73)
(christianity = 1) and (think = 0) and (live = 0) and (argument = 0) and (anti = 0) => text_class=soc.religion.christian (CF = 0.81)
(catholic = 1) and (time = 0) and (does = 0) => text_class=soc.religion.christian (CF = 0.84)

(gun = 1) and (guns = 1) => text_class=talk.politics.guns (CF = 0.94)
(gun = 1) and (firearms = 1) => text_class=talk.politics.guns (CF = 0.96)
(gun = 1) and (defense = 1) => text_class=talk.politics.guns (CF = 0.93)
(fbi = 1) => text_class=talk.politics.guns (CF = 0.85)
(gun = 1) and (people = 0) => text_class=talk.politics.guns (CF = 0.85)
(guns = 1) and (kill = 0) => text_class=talk.politics.guns (CF = 0.92)
(weapons = 1) and (world = 0) and (turkey = 0) and (israel = 0) => text_class=talk.politics.guns (CF = 0.89)
(amendment = 1) => text_class=talk.politics.guns (CF = 0.9)
(firearms = 1) and (accounts = 0) => text_class=talk.politics.guns (CF = 0.97)
(davidians = 1) and (american = 0) => text_class=talk.politics.guns (CF = 0.9)
(waco = 1) and (does = 0) => text_class=talk.politics.guns (CF = 0.89)
(batf = 1) => text_class=talk.politics.guns (CF = 0.93)
(atf = 1) => text_class=talk.politics.guns (CF = 0.93)
(jmd = 1) => text_class=talk.politics.guns (CF = 0.91)
(bd = 1) => text_class=talk.politics.guns (CF = 0.88)
(property = 1) and (did = 0) and (leave = 0) => text_class=talk.politics.guns (CF = 0.68)
(firearm = 1) => text_class=talk.politics.guns (CF = 0.95)
(does = 0) and (criminals = 1) and (don = 0) and (society = 0) => text_class=talk.politics.guns (CF = 0.85)

(israel = 1) and (god = 0) and (time = 0) and (assumption = 0) => text_class=talk.politics.mideast (CF = 0.98)
(armenians = 1) => text_class=talk.politics.mideast (CF = 0.98)
(israeli = 1) and (days = 0) and (history = 0) => text_class=talk.politics.mideast (CF = 0.98)
(turkish = 1) and (said = 0) => text_class=talk.politics.mideast (CF = 0.97)
(arab = 1) => text_class=talk.politics.mideast (CF = 0.98)
(turkey = 1) and (automatic = 0) => text_class=talk.politics.mideast (CF = 0.97)
(jewish = 1) and (jesus = 0) and (bible = 0) and (don = 0) and (people = 1) => text_class=talk.politics.mideast (CF = 0.95) (serdar = 1) => text_class=talk.politics.mideast (CF = 0.98)
(don = 0) and (thanks = 0) and (palestinian = 1) => text_class=talk.politics.mideast (CF = 0.94)
(lebanese = 1) => text_class=talk.politics.mideast (CF = 0.94)
(use = 0) and (civilians = 1) and (gun = 0) and (self = 0) => text_class=talk.politics.mideast (CF = 0.94)
(lebanon = 1) => text_class=talk.politics.mideast (CF = 0.93)
(jews = 1) and (god = 0) and (das = 1) => text_class=talk.politics.mideast (CF = 0.85)
(nazi = 1) and (say = 1) => text_class=talk.politics.mideast (CF = 0.89)

Number of Rules: 108

Appendix B

RIPPER-k Ruleset Generated from the Output of the Neural Network (4 classes and 1,000 input features)

(fbi = 1) => text_class=talk.politics.guns (84.0/0.0)
(gun = 1) => text_class=talk.politics.guns (45.0/2.0)
(government = 1) and (want = 0) => text_class=talk.politics.guns (28.0/2.0)
(weapons = 1) => text_class=talk.politics.guns (16.0/1.0)
(batf = 1) => text_class=talk.politics.guns (9.0/0.0)
(house = 1) and (book = 0) and (god = 0) and (pc = 0)
=> text_class=talk.politics.guns (11.0/2.0)
(guns = 1) => text_class=talk.politics.guns (8.0/3.0)
(doubt = -1) => text_class=talk.politics.guns (7.0/3.0)
(come = -1) => text_class=talk.politics.guns (7.0/3.0)
(right = -1) and (meaning = 1) => text_class=talk.politics.guns (3.0/0.0)
(people = 1) and (suicide = 1) => text_class=talk.politics.guns (3.0/0.0)

(god = 1) and (floppy = 0) => text_class=soc.religion.christian (169.0/0.0)
(church = 1) => text_class=soc.religion.christian (48.0/0.0)
(christian = 1) => text_class=soc.religion.christian (27.0/0.0)
(bible = 1) => text_class=soc.religion.christian (17.0/2.0)
(jesus = 1) => text_class=soc.religion.christian (12.0/0.0)
(religion = 1) => text_class=soc.religion.christian (6.0/0.0)
(christians = 1) => text_class=soc.religion.christian (7.0/0.0)
(doctrine = 1) => text_class=soc.religion.christian (6.0/0.0)
(faith = 1) => text_class=soc.religion.christian (6.0/1.0)
(lord = 1) => text_class=soc.religion.christian (4.0/0.0)
(catholic = 1) => text_class=soc.religion.christian (3.0/0.0)
(clh = 1) => text_class=soc.religion.christian (4.0/0.0)

(card = 1) => text_class=comp.sys.ibm.pc.hardware (82.0/1.0)
(drive = 1) and (car = 0) and (think = 0) and (speed = 0) and (engine = 0)
=> text_class=comp.sys.ibm.pc.hardware (55.0/1.0)
(pc = 1) => text_class=comp.sys.ibm.pc.hardware (44.0/0.0)
(thanks = 1) and (car = 0) and (cars = 0) and (possible = 0) and (ford = 0) and (tires = 0)
=> text_class=comp.sys.ibm.pc.hardware (42.0/2.0) (bios = 1) =>
text_class=comp.sys.ibm.pc.hardware (17.0/0.0)
(modem = 1) => text_class=comp.sys.ibm.pc.hardware (17.0/0.0)
(computer = 1) => text_class=comp.sys.ibm.pc.hardware (18.0/1.0)
(program = 1) => text_class=comp.sys.ibm.pc.hardware (9.0/0.0)
(bus = 1) => text_class=comp.sys.ibm.pc.hardware (5.0/0.0)
(problems = 1) => text_class=comp.sys.ibm.pc.hardware (8.0/1.0)
(address = 1) => text_class=comp.sys.ibm.pc.hardware (8.0/2.0)
(gateway = 1) => text_class=comp.sys.ibm.pc.hardware (6.0/0.0)
(cpu = 1) => text_class=comp.sys.ibm.pc.hardware (6.0/0.0)
(sun = -1) => text_class=comp.sys.ibm.pc.hardware (4.0/0.0)
(software = 1) => text_class=comp.sys.ibm.pc.hardware (4.0/0.0)
(motherboard = 1) => text_class=comp.sys.ibm.pc.hardware (5.0/0.0)
(scsi = 1) => text_class=comp.sys.ibm.pc.hardware (4.0/0.0)
(memory = 1) => text_class=comp.sys.ibm.pc.hardware (3.0/0.0)
(machine = 1) => text_class=comp.sys.ibm.pc.hardware (3.0/0.0)
(monitors = 1) => text_class=comp.sys.ibm.pc.hardware (3.0/0.0)
(monitor = 1) => text_class=comp.sys.ibm.pc.hardware (2.0/0.0)
(800 = 1) => text_class=comp.sys.ibm.pc.hardware (2.0/0.0)
(vlb = 1) => text_class=comp.sys.ibm.pc.hardware (2.0/0.0)

```
(hd = 1) => text_class=comp.sys.ibm.pc.hardware (2.0/0.0)
(cards = 1) => text_class=comp.sys.ibm.pc.hardware (2.0/0.0)
(pin = 1) => text_class=comp.sys.ibm.pc.hardware (3.0/0.0)

=> text_class=rec.autos (664.0/99.0)
```

Number of Rules : 50

RIPPER-k Ruleset Generated from the Output of the Neural Network (8 classes and 1,000 input features)

```
(bike = 1) => text_class=rec.motorcycles (104.0/0.0)
(dod = 1) and (version = 0) => text_class=rec.motorcycles (31.0/1.0)
(ride = 1) => text_class=rec.motorcycles (26.0/7.0)
(motorcycle = 1) => text_class=rec.motorcycles (11.0/0.0)
(bikes = 1) => text_class=rec.motorcycles (11.0/0.0)
(helmet = 1) => text_class=rec.motorcycles (7.0/2.0)
(riding = 1) and (car = 0) => text_class=rec.motorcycles (8.0/0.0)
(bmw = 1) and (car = 0) => text_class=rec.motorcycles (12.0/3.0)
(motorcycles = 1) => text_class=rec.motorcycles (4.0/1.0)

(religion = 1) and (christian = 0) and (new = 0) and (belief = 1)
=> text_class=alt.atheism (10.0/0.0)
(god = -1) and (like = 0) => text_class=alt.atheism (29.0/6.0)
(morality = 1) and (christian = 0) => text_class=alt.atheism (23.0/3.0)
(atheism = 1) and (life = 0) => text_class=alt.atheism (21.0/3.0)
(religion = 1) and (does = 0) and (fact = 0) and (government = 0) and (people = 0)
 => text_class=alt.atheism (15.0/3.0)
(atheist = 1) => text_class=alt.atheism (20.0/6.0)
(real = -1) and (people = 1) => text_class=alt.atheism (4.0/0.0)
(bob = 1) => text_class=alt.atheism (13.0/5.0)
(aware = -1) => text_class=alt.atheism (12.0/5.0)
(technical = -1) => text_class=alt.atheism (5.0/1.0)

(fbi = 1) => text_class=talk.politics.guns (83.0/2.0)
(gun = 1) and (took = 0) => text_class=talk.politics.guns (46.0/3.0)
(weapons = 1) and (land = 0) => text_class=talk.politics.guns (25.0/3.0)
(koresh = 1) => text_class=talk.politics.guns (14.0/2.0)
(bd = 1) => text_class=talk.politics.guns (7.0/0.0)
(law = 1) and (god = 0) and (does = 0) and (new = 0) and (car = 0)
=> text_class=talk.politics.guns (19.0/4.0)
(senate = 1) => text_class=talk.politics.guns (8.0/1.0)
(suicide = 1) => text_class=talk.politics.guns (6.0/1.0)
(batf = 1) => text_class=talk.politics.guns (4.0/0.0)
(guns = 1) and (children = 0) => text_class=talk.politics.guns (10.0/3.0)
(texas = 1) => text_class=talk.politics.guns (9.0/3.0)
(jmd = 1) => text_class=talk.politics.guns (5.0/0.0)
(militia = 1) => text_class=talk.politics.guns (4.0/1.0)
(constitution = 1) and (states = 0) => text_class=talk.politics.guns (7.0/2.0)
(waco = 1) => text_class=talk.politics.guns (4.0/1.0)

(card = 1) and (mac = 0) and (lc = 0) and (apple = 0)
=> text_class=comp.sys.ibm.pc.hardware (80.0/6.0)
(pc = 1) and (apple = 0) => text_class=comp.sys.ibm.pc.hardware (51.0/3.0)
(dos = 1) => text_class=comp.sys.ibm.pc.hardware (28.0/4.0)
(computer = 1) and (mac = 0) and (like = 0) and (car = 0)
=> text_class=comp.sys.ibm.pc.hardware (31.0/7.0)
(controller = 1) => text_class=comp.sys.ibm.pc.hardware (22.0/0.0)
(drivers = 1) and (driver = 0) => text_class=comp.sys.ibm.pc.hardware (14.0/2.0)
(bios = 1) => text_class=comp.sys.ibm.pc.hardware (13.0/0.0)
(drive = -1) and (scsi = -1) => text_class=comp.sys.ibm.pc.hardware (4.0/0.0)
(gateway = 1) => text_class=comp.sys.ibm.pc.hardware (8.0/0.0)
(ide = 1) => text_class=comp.sys.ibm.pc.hardware (8.0/0.0)
(help = 1) and (people = 0) and (drives = 1) =>
text_class=comp.sys.ibm.pc.hardware (5.0/0.0)
```

```
(motherboard = 1) and (cpu = 0) => text_class=comp.sys.ibm.pc.hardware (11.0/3.0)
(port = 1) => text_class=comp.sys.ibm.pc.hardware (11.0/5.0)
(monitor = 1) and (apple = 0) and (mac = 0)
=> text_class=comp.sys.ibm.pc.hardware (13.0/4.0)

(israel = 1) and (god = 0) and (belief = 0) => text_class=talk.politics.mideast (74.0/1.0)
(war = 1) and (government = 1) => text_class=talk.politics.mideast (24.0/0.0)
(arabs = 1) => text_class=talk.politics.mideast (21.0/1.0)
(turkey = 1) => text_class=talk.politics.mideast (22.0/0.0)
(killing = 1) => text_class=talk.politics.mideast (16.0/2.0)
(mr = 1) and (does = 0) and (god = 0) => text_class=talk.politics.mideast (14.0/2.0)
(policy = 1) => text_class=talk.politics.mideast (15.0/3.0)
(islamic = -1) and (reason = 0) and (received = 0)
 => text_class=talk.politics.mideast (10.0/1.0)
(arms = 1) => text_class=talk.politics.mideast (12.0/4.0)
(israeli = 1) => text_class=talk.politics.mideast (6.0/1.0)

(population = 1) => text_class=talk.politics.mideast (6.0/1.0)
(mac = 1) => text_class=comp.sys.mac.hardware (84.0/4.0)
(apple = 1) => text_class=comp.sys.mac.hardware (46.0/2.0)
(centris = 1) => text_class=comp.sys.mac.hardware (14.0/0.0)
(modem = 1) => text_class=comp.sys.mac.hardware (15.0/0.0)
(scsi = 1) => text_class=comp.sys.mac.hardware (11.0/0.0)
(cpu = -1) => text_class=comp.sys.mac.hardware (12.0/2.0)
(heat = 1) => text_class=comp.sys.mac.hardware (7.0/2.0)
(software = 1) => text_class=comp.sys.mac.hardware (22.0/7.0)
(apple = -1) => text_class=comp.sys.mac.hardware (11.0/2.0)
(print = 1) => text_class=comp.sys.mac.hardware (12.0/3.0)
(video = 1) => text_class=comp.sys.mac.hardware (6.0/1.0)
(mac = -1) => text_class=comp.sys.mac.hardware (7.0/0.0)
(keyboard = 1) => text_class=comp.sys.mac.hardware (4.0/0.0)
(monitor = -1) => text_class=comp.sys.mac.hardware (9.0/2.0)
(iisi = 1) => text_class=comp.sys.mac.hardware (7.0/0.0)
(mhz = 1) => text_class=comp.sys.mac.hardware (6.0/1.0)
(simms = 1) => text_class=comp.sys.mac.hardware (4.0/0.0)

(god = 1) => text_class=soc.religion.christian (183.0/11.0)
(church = 1) => text_class=soc.religion.christian (55.0/0.0)
(christian = 1) => text_class=soc.religion.christian (34.0/3.0)
(christians = 1) => text_class=soc.religion.christian (17.0/1.0)
(bible = 1) => text_class=soc.religion.christian (17.0/3.0)
(faith = 1) => text_class=soc.religion.christian (12.0/3.0)
(jesus = 1) => text_class=soc.religion.christian (11.0/1.0)
(heaven = 1) => text_class=soc.religion.christian (6.0/1.0)
(catholic = 1) => text_class=soc.religion.christian (5.0/0.0)
(word = 1) and (does = 1) => text_class=soc.religion.christian (3.0/0.0)
(marriage = 1) => text_class=soc.religion.christian (4.0/0.0)
 (revelation = 1) => text_class=soc.religion.christian (3.0/0.0)
(feel = 1) and (world = 1) => text_class=soc.religion.christian (3.0/0.0)
(understanding = 1) => text_class=soc.religion.christian (2.0/0.0)

=> text_class=rec.autos (1260.0/694.0)
```

Number of Rules: 91

FURIA Ruleset Generated from the Output of the Neural Network (4 classes 1,000 input features)

(card = 1) and (look = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(pc = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.98)
(drive = 1) and (car = 0) and (speed = 0) and (cars = 0) and (wheel = 0)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.94)
(thanks = 1) and (car = 0) and (like = 0) and (today = 0) and (post = 0) and (oil = 0)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.94)
(computer = 1) and (like = 0) and (car = 0)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.94)
(modem = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(bios = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(card = -1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.82)
(people = 0) and (cpu = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.94)
(monitor = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(drive = -1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.74)
(address = 1) and (people = 0) and (car = 0)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.87)
(people = 0) and (program = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.92)
(memory = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.83)
(50 = -1) and (like = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.79)
(email = 1) and (god = 0) and (car = 0)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.92)
(scsi = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(people = 0) and (computers = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.92)
(monitors = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.87)
(board = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.93)
(manual = -1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.78)
(dos = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(copy = -1) and (post = -1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.7)
(bios = -1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.85)
(digital = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.89)
(model = -1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.81)

(car = 1) => text_class=rec.autos (CF = 0.99)
(engine = 1) => text_class=rec.autos (CF = 0.98)
(cars = 1) => text_class=rec.autos (CF = 0.98)
(god = 0) and (people = 0) and (radar = 1) => text_class=rec.autos (CF = 0.91)
(god = 0) and (time = 0) and (mustang = 1) => text_class=rec.autos (CF = 0.94)

(god = 1) and (connect = 0) => text_class=soc.religion.christian (CF = 0.99)
(god = -1) and (compound = 0) => text_class=soc.religion.christian (CF = 0.97)
(christian = 1) => text_class=soc.religion.christian (CF = 0.98)
(church = 1) and (day = 0) => text_class=soc.religion.christian (CF = 0.98)
(bible = 1) => text_class=soc.religion.christian (CF = 0.97)
(christians = 1) => text_class=soc.religion.christian (CF = 0.97)
(jesus = 1) => text_class=soc.religion.christian (CF = 0.97)
(lord = 1) => text_class=soc.religion.christian (CF = 0.96)
(year = -1) and (time = 1) and (pc = 0) => text_class=soc.religion.christian (CF
(marriage = 1) => text_class=soc.religion.christian (CF = 0.93)
(faith = 1) and (driver = 0) => text_class=soc.religion.christian (CF = 0.96)
(christ = -1) => text_class=soc.religion.christian (CF = 0.91)
(evil = -1) => text_class=soc.religion.christian (CF = 0.92)
(religion = 1) and (gun = 0) and (guns = 0)
=> text_class=soc.religion.christian (CF = 0.95)
(satan = 1) => text_class=soc.religion.christian (CF = 0.88)

```
(christian = -1) and (news = 0) => text_class=soc.religion.christian (CF = 0.94)
(fbi = 1) => text_class=talk.politics.guns (CF = 0.98)
(gun = 1) => text_class=talk.politics.guns (CF = 0.98)
(government = 1) and (car = 0) => text_class=talk.politics.guns (CF = 0.97)
(weapons = 1) and (car = 0) => text_class=talk.politics.guns (CF = 0.96)
(house = 1) and (god = 0) and (church = 0) => text_class=talk.politics.guns (CF = 0.95)
(texas = 1) => text_class=talk.politics.guns (CF = 0.96)
(com = 1) and (jmd = 1) => text_class=talk.politics.guns (CF = 0.89)
(batf = 1) => text_class=talk.politics.guns (CF = 0.97)
(koresh = 1) => text_class=talk.politics.guns (CF = 0.97)
(compound = 1) => text_class=talk.politics.guns (CF = 0.96)
(com = 1) and (dave = 1) => text_class=talk.politics.guns (CF = 0.8)
(shot = 1) and (god = 0) => text_class=talk.politics.guns (CF = 0.95)
(guns = 1) => text_class=talk.politics.guns (CF = 0.96)
(like = 0) and (investigation = 1) => text_class=talk.politics.guns (CF = 0.89)
(law = 1) and (god = 0) and (long = 0) and (car = 0)
=> text_class=talk.politics.guns (CF = 0.95)
(waco = 1) => text_class=talk.politics.guns (CF = 0.95)
(country = 1) and (god = 0) => text_class=talk.politics.guns (CF = 0.9)
(assault = 1) => text_class=talk.politics.guns (CF = 0.94)
(article = -1) and (sin = 0) and (thanks = 0) => text_class=talk.politics.guns (CF = 0.89)
(know = 0) and (control = 1) and (drive = 0)
=> text_class=talk.politics.guns (CF = 0.89)
(arms = 1) => text_class=talk.politics.guns (CF = 0.91)
```

Number of Rules: 68

FURIA Ruleset Generated from the Output of the Neural Network (8 classes and 1,000 input features)

(religion = 1) and (christian = 0) and (children = 0) => text_class=alt.atheism (CF = 0.77)
(god = -1) and (world = 0) => text_class=alt.atheism (CF = 0.8)
(morality = 1) and (god = 0) => text_class=alt.atheism (CF = 0.87)
(religious = 1) and (law = 0) and (does = 0) and (historical = 0) and (church = 0) and (catholic = 0) and (hardware = 0) and (going = 0) => text_class=alt.atheism (CF = 0.84)
(just = 0) and (atheist = 1) and (does = 0) => text_class=alt.atheism (CF = 0.71)
(use = 0) and (irony = 1) => text_class=alt.atheism (CF = 0.69)

(card = 1) and (pds = 0) and (windows = 1)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.93)
(pc = 1) and (apple = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.93)
(card = 1) and (pds = 0) and (question = 0) and (mac = 0) and (book = 0) and (lc = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.91)
(ide = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(bios = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(thanks = -1) and (does = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.81)
(disk = 1) and (mac = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.79)
(dos = 1) and (computer = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.93)
(gateway = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.89)
(controller = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.94)
(people = 0) and (motherboard = 1) and (cpu = 0) and (mac = 0)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.9)
(windows = 1) and (just = 0) and (called = 0) and (door = 0)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.89)
(people = 0) and (cpu = -1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.75)
(thanks = -1) and (tell = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.71)

(mac = 1) and (said = 0) and (religious = 0) and (formatted = 0)
=> text_class=comp.sys.mac.hardware (CF = 0.98)
(apple = 1) and (bike = 0) and (blue = 0)
=> text_class=comp.sys.mac.hardware (CF = 0.98)
(people = 0) and (thanks = 1) and (does = -1)
=> text_class=comp.sys.mac.hardware (CF = 0.81)
(people = 0) and (lc = 1) => text_class=comp.sys.mac.hardware (CF = 0.9)
(people = 0) and (centris = 1) => text_class=comp.sys.mac.hardware (CF = 0.93)

(car = 1) and (make = 0) and (away = 0) and (bike = 0)
=> text_class=rec.autos (CF = 0.97)
(engine = 1) and (bike = 0) and (just = 0) and (willing = 0)
=> text_class=rec.autos (CF = 0.96)
(cars = 1) and (bike = 0) and (said = 0) and (christ = 0)
=> text_class=rec.autos (CF = 0.97)
(ford = 1) and (source = 0) => text_class=rec.autos (CF = 0.96)
(mustang = 1) => text_class=rec.autos (CF = 0.92)
(toyota = 1) => text_class=rec.autos (CF = 0.89)
(god = 0) and (chevy = 1) => text_class=rec.autos (CF = 0.88)
(god = 0) and (honda = -1) and (doing = 0) => text_class=rec.autos (CF = 0.78)

(bike = 1) => text_class=rec.motorcycles (CF = 0.98)
(dod = 1) and (version = 0) => text_class=rec.motorcycles (CF = 0.96)
(ride = 1) and (small = 0) and (paul = 0) => text_class=rec.motorcycles (CF = 0.96)
(people = 0) and (bikes = 1) => text_class=rec.motorcycles (CF = 0.93)
(motorcycle = 1) => text_class=rec.motorcycles (CF = 0.94)
(just = 0) and (riding = 1) => text_class=rec.motorcycles (CF = 0.91)

(people = 0) and (dog = 1) and (christianity = 0) and (faith = 0) and (cars = 0) and (use = 0) => text_class=rec.motorcycles (CF = 0.87)
(just = 0) and (chain = 1) => text_class=rec.motorcycles (CF = 0.87)

(god = 1) and (jesus = 1) => text_class=soc.religion.christian (CF = 0.97)
(god = 1) and (support = 0) and (christians = 1) =>
text_class=soc.religion.christian (CF = 0.94)
(christian = 1) and (church = 1) => text_class=soc.religion.christian (CF = 0.94)
(church = 1) and (interesting = 0) => text_class=soc.religion.christian (CF = 0.95)
(god = 1) and (life = 1) => text_class=soc.religion.christian (CF = 0.94)
(god = 1) and (religious = 0) and (christian = 1)
=> text_class=soc.religion.christian (CF = 0.96)
(jesus = 1) => text_class=soc.religion.christian (CF = 0.96)
(christian = 1) and (use = 0) => text_class=soc.religion.christian (CF = 0.9)
(god = 1) and (religious = 0) and (interesting = 0)
=> text_class=soc.religion.christian (CF = 0.9)
(lord = 1) and (world = 0) => text_class=soc.religion.christian (CF = 0.96)
(bible = 1) => text_class=soc.religion.christian (CF = 0.89)
(sin = 1) => text_class=soc.religion.christian (CF = 0.97)
(christ = 1) and (drive = 0) => text_class=soc.religion.christian (CF = 0.98)
(faith = 1) and (people = 0) => text_class=soc.religion.christian (CF = 0.91)
(homosexual = 1) => text_class=soc.religion.christian (CF = 0.95)
(doctrine = 1) => text_class=soc.religion.christian (CF = 0.9)
(christianity = 1) and (people = 0) => text_class=soc.religion.christian (CF = 0.96)
(scriptural = 1) => text_class=soc.religion.christian (CF = 0.84)
(translation = 1) => text_class=soc.religion.christian (CF = 0.88)
(clh = 1) => text_class=soc.religion.christian (CF = 0.69)

(gun = 1) and (open = 0) => text_class=talk.politics.guns (CF = 0.9)
(law = 1) and (god = 0) and (world = 0) => text_class=talk.politics.guns (CF = 0.73)
(federal = 1) => text_class=talk.politics.guns (CF = 0.75)
(guns = 1) and (children = 0) and (bike = 0) => text_class=talk.politics.guns (CF = 0.96)
(gas = 1) and (air = 1) and (cars = 0) => text_class=talk.politics.guns (CF = 0.8)

(war = 1) and (population = 1) and (god = 0)
=> text_class=talk.politics.mideast (CF = 0.94)
(jews = 1) and (god = 0) and (catholic = 0) and (religious = 0) =>
text_class=talk.politics.mideast (CF = 0.88)
(war = 1) and (men = 1) => text_class=talk.politics.mideast (CF = 0.9)
(war = 1) and (just = 0) and (hold = 0) and (cars = 0)
=> text_class=talk.politics.mideast (CF = 0.79)
(turkey = 1) => text_class=talk.politics.mideast (CF = 0.95)
(jewish = 1) and (christian = 0) and (god = 0) and (said = 0) =>
text_class=talk.politics.mideast (CF = 0.87)
(said = -1) and (national = -1) => text_class=talk.politics.mideast (CF = 0.74)
(children = -1) => text_class=talk.politics.mideast (CF = 0.55)
(world = -1) and (national = -1) => text_class=talk.politics.mideast (CF = 0.74)
(process = 1) and (middle = 1) => text_class=talk.politics.mideast (CF = 0.74)
(000 = -1) and (american = 1) => text_class=talk.politics.mideast (CF = 0.74)
(people = 1) and (claim = -1) => text_class=talk.politics.mideast (CF = 0.64)

Number of Rules: 78

Appendix C

FURIA Ruleset Generated from the Output of the Neural Network (4 classes and 500 input features)

(card = 1) and (look = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.98)
(pc = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.98)
(drive = 1) and (car = 0) and (engine = 0) and (wheel = 0) and (clutch = 0) and (dealer = 0) and (hot = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.98)
(computer = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.92)
(thanks = 1) and (car = 0) and (god = 0) and (post = 0) and (christians = 0) and (new = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.92)
(bios = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(modem = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(monitor = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(cpu = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(people = 0) and (file = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(disk = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(drivers = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.87)
(motherboard = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.95)
(people = 0) and (bus = 1) and (weapons = 0)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(people = 0) and (port = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.94)
(email = 1) and (believe = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.87)
(upgrade = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.85)
(god = 0) and (plug = -1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.85)
(monitors = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.89)
(memory = 1) and (believe = 0) and (engine = 0) =>
text_class=comp.sys.ibm.pc.hardware (CF = 0.95)

(car = 1) => text_class=rec.autos (CF = 0.99)
(engine = 1) => text_class=rec.autos (CF = 0.98)
(god = 0) and (cars = 1) => text_class=rec.autos (CF = 0.98)
(god = 0) and (does = 0) and (people = 0) and (mustang = 1) =>
text_class=rec.autos (CF = 0.93)

(god = 1) and (fbi = 0) and (drive = 0) => text_class=soc.religion.christian (CF = 0.99)
(church = 1) and (batf = 0) and (car = 0) => text_class=soc.religion.christian (CF = 0.98)
(christian = 1) => text_class=soc.religion.christian (CF = 0.98)
(god = -1) and (email = 0) and (remember = 0) =>
text_class=soc.religion.christian (CF = 0.92)
(bible = 1) and (gun = 0) => text_class=soc.religion.christian (CF = 0.98)
(christians = 1) => text_class=soc.religion.christian (CF = 0.98)
(jesus = 1) => text_class=soc.religion.christian (CF = 0.98)
(marriage = 1) => text_class=soc.religion.christian (CF = 0.95)
(christian = -1) and (news = 0) => text_class=soc.religion.christian (CF = 0.88)
(heaven = 1) => text_class=soc.religion.christian (CF = 0.96)
(catholic = 1) => text_class=soc.religion.christian (CF = 0.97)
(faith = 1) and (driver = 0) => text_class=soc.religion.christian (CF = 0.97)

(fbi = 1) => text_class=talk.politics.guns (CF = 0.98)
(gun = 1) => text_class=talk.politics.guns (CF = 0.96)
(government = 1) and (recently = 0) and (drive = 0) =>
text_class=talk.politics.guns (CF = 0.97)
(weapons = 1) => text_class=talk.politics.guns (CF = 0.96)
(house = 1) and (god = 0) and (long = 0) and (christianity = 0) and (new =

```
(batf = 1) => text_class=talk.politics.guns (CF = 0.97)
(koresh = 1) => text_class=talk.politics.guns (CF = 0.97)
(guns = 1) and (god = 0) => text_class=talk.politics.guns (CF = 0.96)
(waco = 1) => text_class=talk.politics.guns (CF = 0.96)
(constitution = 1) and (church = 0) => text_class=talk.politics.guns (CF =
(suicide = 1) and (new = 0) => text_class=talk.politics.guns (CF = 0.92)
(compound = 1) and (make = 0) => text_class=talk.politics.guns (CF = 0.95)
(shot = 1) and (new = 0) => text_class=talk.politics.guns (CF = 0.94)
(tear = 1) => text_class=talk.politics.guns (CF = 0.86)
(arms = 1) => text_class=talk.politics.guns (CF = 0.86)
(dave = 1) and (com = 1) => text_class=talk.politics.guns (CF = 0.77)
(jmd = 1) => text_class=talk.politics.guns (CF = 0.89)
```

Number of Rules: 53

FURIA Ruleset Generated from the Output of the Neural Network (8 classes and 500 input features)

(god = -1) => text_class=alt.atheism (CF = 0.75)
(religion = 1) and (christian = 0) and (muslims = 0) => text_class=alt.atheism (CF = 0.68)
(atheist = 1) => text_class=alt.atheism (CF = 0.75)
(morality = 1) and (christian = 0) => text_class=alt.atheism (CF = 0.86)
(atheists = 1) => text_class=alt.atheism (CF = 0.78)
(kent = 1) => text_class=alt.atheism (CF = 0.69)
(atheism = 1) and (god = 0) => text_class=alt.atheism (CF = 0.89)

(card = 1) and (apple = 0) and (think = 0) and (ones = 0) and (hello = 0) and (ride = 0) and (driving = 0) and (fixed = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.97)
(pc = 1) and (mac = 0) and (freedom = 0) and (told = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.97)
(thanks = 1) and (drives = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.88)
(ide = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(bios = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.96)
(computer = 1) and (don = 0) and (memory = 0) and (apple = 0) and (people = 0) and (help = 0) and (dos = 0) and (speed = 0) and (interested = 0) and (expensive = 0) => text_class=co mp.sys.ibm.pc.hardware (CF = 0.94)
(thanks = 1) and (windows = 1) and (car = 0)
=> text_class=comp.sys.ibm.pc.hardware (CF = 0.86)
(drivers = 1) and (driver = 0) and (bike = 0) and (vehicles = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.92)
(help = 1) and (appreciated = 1) and (problem = 0) and (bible = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.91)
(thanks = 1) and (think = -1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.71)
(board = 1) and (apple = 0) and (got = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.81)
(port = 1) and (mac = 0) and (don = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.91)
(mode = 1) and (car = 0) and (need = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.92)
(motherboard = 1) and (thanks = 0) and (16 = 0) and (need = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.91)
(monitor = 1) and (apple = 0) and (refresh = 1) => text_class=comp.sys.ibm.pc.hardware (CF = 0.75)
(floppy = 1) and (right = 0) and (mac = 0) and (don = 0) and (bike = 0) => text_class=comp.sys.ibm.pc.hardware (CF = 0.9)

(mac = 1) and (said = 0) and (uiuc = 0) and (transfer = 0) => text_class=comp.sys.mac.hardware (CF = 0.98)
(apple = 1) => text_class=comp.sys.mac.hardware (CF = 0.94)
(iisi = 1) => text_class=comp.sys.mac.hardware (CF = 0.9)
(thanks = -1) and (problem = 1) => text_class=comp.sys.mac.hardware (CF = 0.7)
(computer = -1) and (problem = 1) => text_class=comp.sys.mac.hardware (CF = 0.75)
(thanks = -1) and (buy = 1) => text_class=comp.sys.mac.hardware (CF = 0.7)
(drives = -1) => text_class=comp.sys.mac.hardware (CF = 0.81)
(appreciated = -1) => text_class=comp.sys.mac.hardware (CF = 0.66)
(chip = -1) => text_class=comp.sys.mac.hardware (CF = 0.68)

(car = 1) and (away = 0) => text_class=rec.autos (CF = 0.92)
(engine = 1) and (bike = 0) and (experience = 0) => text_class=rec.autos (CF = 0.95)
(people = 0) and (cars = 1) and (bike = 0) => text_class=rec.autos (CF = 0.96)
(people = 0) and (don = 0) and (honda = -1) => text_class=rec.autos (CF = 0.84)

(bike = 1) and (door = 0) => text_class=rec.motorcycles (CF = 0.98)
(dod = 1) and (60 = 0) and (computer = 0) => text_class=rec.motorcycles (CF = 0.96)
(ride = 1) and (car = 0) and (bus = 0) => text_class=rec.motorcycles (CF = 0.96)
(motorcycle = 1) => text_class=rec.motorcycles (CF = 0.94)
(bikes = 1) => text_class=rec.motorcycles (CF = 0.94)
(people = 0) and (bmw = 1) and (car = 0) and (don = 0) and (cars = 0) and (gt = 0) => text_class=rec.motorcycles (CF = 0.92)
(riding = 1) and (car = 0) => text_class=rec.motorcycles (CF = 0.96)
(people = 0) and (chain = 1) and (scsi = 0) => text_class=rec.motorcycles (CF = 0.91)
(motorcycles = 1) => text_class=rec.motorcycles (CF = 0.9)
(people = 0) and (helmet = 1) => text_class=rec.motorcycles (CF = 0.87)
(people = 0) and (biker = 1) => text_class=rec.motorcycles (CF = 0.86)

(god = 1) and (jesus = 1) => text_class=soc.religion.christian (CF = 0.98)
(god = 1) and (religious = 0) and (right = 0) and (say = 1) => text_class=soc.religion.christian (CF = 0.96)
(church = 1) and (people = 0) and (violence = 0) => text_class=soc.religion.christian (CF = 0.93)
(god = 1) and (right = 0) and (religion = 0) and (atheists = 0) and (perfect = 0) and (new = 1) => text_class=soc.religion.christian (CF = 0.92)
(bible = 1) and (word = 0) and (say = 0) and (ken = 0) and (tek = 0) => text_class=soc.religion.christian (CF = 0.96)
(christian = 1) and (turkish = 0) and (atheist = 0) and (wouldn = 0) and (mac = 0) and (israeli = 0) and (turks = 0) => text_class=soc.religion.christian (CF = 0.98)
(god = 1) and (religious = 0) and (right = 0) and (don = 0) and (recently = 0) and (atheists = 0) and (perfect = 0) and (wait = 0) and (action = 0) and (interesting = 0) and (commen ts = 0) and (define = 0) => text_class=soc.religion.christian (CF = 0.97)
(jesus = 1) and (christian = 0) and (parts = 0) and (money = 0) => text_class=soc.religion.christian (CF = 0.97)
(god = 1) and (punishment = 1) => text_class=soc.religion.christian (CF = 0.85)
(word = 1) and (people = -1) => text_class=soc.religion.christian (CF = 0.83)
(christianity = 1) and (people = 0) => text_class=soc.religion.christian (CF = 0.95)
(word = 1) and (did = -1) => text_class=soc.religion.christian (CF = 0.71)

(guns = 1) and (left = 0) and (word = 0) => text_class=talk.politics.guns (CF = 0.9)

(israel = 1) and (jesus = 0) and (33 = 0) and (die = 0)
=> text_class=talk.politics.mideast (CF = 0.98)
(muslims = 1) and (say = 0) and (different = 0)
=> text_class=talk.politics.mideast (CF = 0.97)
(armenian = 1) => text_class=talk.politics.mideast (CF = 0.97)
(arab = 1) => text_class=talk.politics.mideast (CF = 0.98)
(jews = 1) and (god = 0) and (did = 0) and (young = 0) => text_class=talk.politics.mideast (CF = 0.95)
(jewish = 1) and (say = 0) and (god = 0) and (terms = 0) => text_class=talk.politics.mideast (CF = 0.97)
(peace = 1) and (god = 0) and (don = 0) and (want = 0) and (read = 0) and (wouldn = 0)
=> text_class=talk.politics.mideast (CF = 0.95)
(israeli = 1) => text_class=talk.politics.mideast (CF = 0.97)
(turkey = 1) => text_class=talk.politics.mideast (CF = 0.95)
(troops = 1) and (car = 0) => text_class=talk.politics.mideast (CF = 0.95)

Number of Rules: 70