Dynamic Feature Grouping in Anomaly Detection

Karl Walter
karl.walter@hotmail.com
Jonatan Åkerman
jonatan.akerman@hotmail.se

Department of Electrical and Information Technology
Lund University

Supervisors: Christian Gehrmann (LTH), Sara Ramezanian (LTH)
Jakob Folkesson (Backtick), Fredrik Olsson (Backtick)

Examiner: Thomas Johansson (LTH)

June 10, 2024

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

Anomaly detection is important in many different areas, among them web security. When performing anomaly detection in web security there can be hundreds of features in the collected web traffic data to consider, even though all of them might not be necessary. This thesis aims to develop a strategy to dynamically create significant parameter groups from web traffic data containing over 100 features, maximising information retention for effective detection of fraudulent activity. This was achieved by employing a combination of machine learning models and systematically evaluating the predictive ability of different feature groups.

Using simple digital signal processing tools, activity level spikes were detected and used as a proxy for indicating scripted attack behaviour in web traffic. With the use of multiple feature importance methods in an ensemble context, a ranking of the most important features for classifying data points as in-spike or not was constructed. Using this ranking, multiple feature selection algorithms were able to find groups of features that on their own could help simple machine learning models determine whether a data point was benign or not.

Using the pipeline constructed in this work, a simple logistic regression model trained on only the feature groups delivered by the pipeline could classify data points as part of spikes or not, as good or sometimes better than a logistic regression model trained on the complete data set. This shows a high information retention in the feature groups, and a possibility of these groups being helpful in aiding the existing systems used by the web security firm Castle.

Popular Science Summary

In a digital landscape where AI tools improve on the daily, web security is at a threat as a means of authenticating users of web sites. At the same time, the amount of data in web traffic is larger than ever, with hundreds of features describing the users and their behaviour. This thesis aims to develop a strategy to find groups of features that alone can indicate unwanted behaviour, and thus increase the capability and interpretability of web security decision making.

In the digital age, web security is paramount. The simple act of logging into a web site often requires password, multi factor authentication and even a small puzzle to prove the validity of the user. There are many reasons for maintaining strong web security, everything from credit card fraud to account takeovers needs to be prevented. One of the more common ways to commit such crimes is by automating malicious web site usage, like trying thousands of different codes for a stolen credit card. Existing solutions aim to address these problems, but with the rise of highly competent AI used with malicious intent, they may not hold up to such threats in the future.

It is crucial to distinguish real users from ones that come from automated attacks, aiming to exploit vulnerabilities in web sites and their users. To combat malicious activity, an approach relying on more than user authentication is needed, but analysing web traffic exhaustively is not tractable due to the large number of features that describe every single data point in the traffic. What if there was a way to analyse only a select few of the data features and find insights on what reveals an automated user?

This thesis introduces a novel strategy for dynamically finding groups of features that can reveal patterns indicative of automated attacks. It was found that this strategy could boil down the number of features to a select few that retained as much information as the complete web traffic data. This was achieved by using machine learning tools to rank the importance of the features and build groups of them to find the ones that best represented the contents of the web traffic. The results can be used in web security in existing rule based services to easier find patterns that reveal automated activity. By finding small groups of features, it allows for qualified parties to find patterns in the data and make decisions accordingly.

Table of Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Thesis objectives . . . 2
- 1.3 Use case . . . 3
- 1.4 Scope and limitations . . . 3
- 1.5 Report structure . . . 3

2 Background **5**
- 2.1 Machine learning concepts . . . 5
- 2.2 Machine learning models . . . 7
- 2.3 scikit-learn . . . 9
- 2.4 Feature importance . . . 10
- 2.5 Information criterion . . . 11
- 2.6 One-hot encoding . . . 11
- 2.7 Pandas . . . 12
- 2.8 Web traffic . . . 12

3 Related Work **15**
- 3.1 Ensemble strategies for feature importance . . . 15
- 3.2 Feature selection . . . 16

4 Data **19**
- 4.1 Origin . . . 19
- 4.2 Structure and contents . . . 19
- 4.3 Example data sets . . . 20

5 Methodology **25**
- 5.1 Data processing . . . 26
- 5.2 Model training . . . 28
- 5.3 Feature importance . . . 29
- 5.4 Feature selection . . . 29
- 5.5 Evaluation . . . 30

6 Results **33**

6.1 Target labelling 33
6.2 Relevant feature retrieval 37
6.3 Model prediction/information loss 50

7 Discussion **59**
7.1 Target labelling 59
7.2 Relevant feature retrieval 60
7.3 Model prediction/information loss 62
7.4 Performance 63
7.5 Target proxy 63

8 Conclusions and Future Work **65**
8.1 Conclusions 65
8.2 Future work 66

References **69**

List of Figures

2.1 Bias and variance example [22]. 6
2.2 The logistic function $\sigma(t)$ in the interval $t \in [-10, 10]$. 8
2.3 Decision tree to decide whether to play tennis or not. 9
2.4 One hot encoding example [17]. 12

3.1 Majority vote implementation [21]. 16

4.1 Example data set 1: Activity level for all events. 20
4.2 Example data set 2: Activity level for all events. 21
4.3 Example data set 3: Activity level for all events. 22
4.4 Example data set 4: Activity level for all events. 23
4.5 Example data set 5: Activity level for all events. 24

5.1 Pipeline overview. 25

6.1 Detected spike locations in example data set 1. The red in the figure indicates that the web traffic was classified as attack and the blue is normal traffic. 34
6.2 Detected spike locations in example data set 2. The red in the figure indicates that the web traffic was classified as attack and the blue is normal traffic. 34
6.3 Detected spike locations in example data set 3. The red in the figure indicates that the web traffic was classified as attack and the blue is normal traffic. 35
6.4 Detected spike locations in example data set 4. The red in the figure indicates that the web traffic was classified as attack and the blue is normal traffic. 35
6.5 Detected spike locations in example data set 5. The red in the figure indicates that the web traffic was classified as attack and the blue is normal traffic. 36
6.6 Forward selection, example data set 1. 38
6.7 Backward selection, example data set 1. 39
6.8 Bidirectional elimination, example data set 1. 39
6.9 Forward selection, example data set 2. 40

6.10 Backward selection, example data set 2. 40
6.11 Bidirectional elimination, example data set 2. 41
6.12 Forward selection, example data set 3. 41
6.13 Backward selection, example data set 3. 42
6.14 Bidirectional elimination, example data set 3. 42
6.15 Forward selection, example data set 4. 43
6.16 Backward selection, example data set 4. 43
6.17 Bidirectional elimination, example data set 4. 44
6.18 Forward selection, example data set 5. 44
6.19 Backward selection, example data set 5. 45
6.20 Bidirectional elimination, example data set 5. 45

List of Tables

2.1 A DataFrame example. 12

5.1 Models and hyper-parameters. 29
5.2 Attacker indicator features and values (some values hashed for privacy reasons). 31

6.1 Target labelling macro average F1 scores for all data sets 36
6.2 Classification report, example data set 1, automatic spike detection towards Castle's target. 36
6.3 Classification report, example data set 2, automatic spike detection towards Castle's target. 37
6.4 Classification report, example data set 3, automatic spike detection towards Castle's target. 37
6.5 Classification report, example data set 5, automatic spike detection towards Castle's target. 37
6.6 Group size of the groups delivered by each of the feature selection algorithms for each of the data set. 46
6.7 Forward selection groups for all data sets 47
6.8 Backward selection groups for all data sets 48
6.9 Bidirectional elimination groups for all data sets 49
6.10 Macro average F1 scores for all data sets and selection algorithms. . 50
6.11 Classification report, example data set 1, all features, predicted with LRG. 51
6.12 Classification report, example data set 1, top 50 features, predicted with LRG. 51
6.13 Classification report, example data set 1, forward selection group features, predicted with LRG. 51
6.14 Classification report, example data set 1, backward selection group features, predicted with LRG. 51
6.15 Classification report, example data set 1, bidirectional elimination group features, predicted with LRG. 52
6.16 Classification report, example data set 2, all features, predicted with LRG. 52

6.17 Classification report, example data set 2, top 50 features, predicted with LRG. . . . 52
6.18 Classification report, example data set 2, forward selection group features, predicted with LRG. . . . 52
6.19 Classification report, example data set 2, backward selection group features, predicted with LRG. . . . 53
6.20 Classification report, example data set 2, bidirectional elimination group features, predicted with LRG. . . . 53
6.21 Classification report, example data set 3, all features, predicted with LRG. . . . 53
6.22 Classification report, example data set 3, top 50 features, predicted with LRG. . . . 53
6.23 Classification report, example data set 3, forward selection group features, predicted with LRG. . . . 54
6.24 Classification report, example data set 3, backward selection group features, predicted with LRG. . . . 54
6.25 Classification report, example data set 3, bidirectional elimination group features, predicted with a logistic regression model. . . . 54
6.26 Classification report, example data set 4, all features, predicted with LRG. . . . 54
6.27 Classification report, example data set 4, top 50 features, predicted with LRG. . . . 55
6.28 Classification report, example data set 4, forward selection group features, predicted with LRG. . . . 55
6.29 Classification report, example data set 4, backward selection group features, predicted with LRG. . . . 55
6.30 Classification report, example data set 4, bidirectional elimination group features, predicted with LRG. . . . 55
6.31 Classification report, example data set 5, all features, predicted with LRG. . . . 56
6.32 Classification report, example data set 5, top 50 features, predicted with LRG. . . . 56
6.33 Classification report, example data set 5, forward selection group features, predicted with LRG. . . . 56
6.34 Classification report, example data set 5, backward selection group features, predicted with LRG. . . . 56
6.35 Classification report, example data set 5, bidirectional elimination group features, predicted with LRG. . . . 57

Acronyms

ML - Machine Learning
CAPTCHA - Completely Automated Public Turing test to tell Computers and Humans Apart
PI - Permutation Importance
SHAP - SHapley Additive exPlanations
IP - Internet Protocol
ISP - Internet Service Provider
OS - Operating System
CSV - Comma Separated Values
LRG - Logistic Regression
DTC - Decision Tree Classifier
GBC - Gradient Boosting Classifier
MDI - Mean Decrease Impurity
AIC - Akaike Information Criterion
BIC - Bayesian Information Criterion
DSP - Digital Signal Processing
JSON - JavaScript Object Notation

Chapter 1
Introduction

This chapter will discuss the structure of the thesis as a whole, as well as the background of the problem formulation of this thesis, and more specifically the origin of the problem as well as motivation for the solution that this thesis will provide. The chapter will outline the need for this work, examine its main questions, and explain how they can contribute to the development of security in web applications.

1.1 Motivation

Web security is in constant change, and with the growing capacity of Machine Learning (ML) models used with malicious intent, current security solutions aimed at finding automated threats such as CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) are not guaranteed to hold up in the future [11]. At the same time, the amount of data extracted from web traffic is enormous, making extensive analysis impractical. The attackers are also changing their strategies all the time, making the need to adapt security solutions even greater.

A big threat for many web services is scripted attacks [7], which in essence are attacks consisting of repeated, automated interactions with the web service in question, with the purpose of exploiting some vulnerability in the target system. There are many reasons for the need to avoid scripted traffic to websites, especially if the traffic volume is well above the usual levels. If scripted traffic is left, attackers could exploit the website. For example, if an attacker have a list of usernames and passwords leaked from one website, they could use a script to test the same usernames and passwords on other websites.

To combat this fraudulent activity, it is paramount to be able to catch this sort of activity in a data stream that is both extensive and noisy. A particular challenge is sorting out the allowed activity during an attack. It is often critical that regular activity on a website can continue as usual, even when attacks are happening. When manually investigating the web traffic and its features, which are the attributes of the web traffic data points, sometimes patterns in certain features or groups of features can reveal scripted activity. An example of this can be that

many data points occur in the same geographical region and have the same screen resolution. For data with a large number of features however, checking all combinations of features is combinatorially intractable. This means there is a need to capture characterising groups of features that can reveal fraudulent activity upon closer inspection.

1.2 Thesis objectives

This thesis project aims to develop a strategy to dynamically group a number of features into one or more subsets of the original feature space to be able to perform anomaly detection to client sessions in certain feature groups. In this study focus will be on scripted attacks specifically.

1.2.1 Main questions

- Q1: Can a machine learning pipeline dynamically find groups of important features in collected web traffic data in order for existing systems to classify fraudulent activity?
- Q2: Can such a pipeline perform better than a simple baseline solution in predicting fraudulent activity?
- Q3: Can such a pipeline improve interpretability and facilitate decision making?

1.2.2 Precision

For the goals of this work, the precision must be considered. Predicting a false positive in this application would mean locking out a legitimate user from logging in, signing up, or another activity on a website, and this is something that is essential to avoid. Predicting a false negative in this application would mean not detecting an attacker. While false negatives are also a concern, they are not as critical as false positives. Locking out real users would be extremely frustrating for the users and might make the service unusable. However, many attacks are only effective at scale, so by locking out significant portions of the attack, it would be very inefficient.

1.2.3 Interpretability

For the web service owner performing the anomaly detection, interpretability of the underlying system performing anomaly detection can be a valuable tool in order to make more informed decisions regarding web security. It is one thing to be able to classify the traffic as scripted or not, and another to be able to say what that decision is based on. More interpretability can let the owner discover bigger patterns in the malicious activities and configure specific rules based on these patterns. Having an end-to-end data-driven process that can group important features together while simultaneously explaining the process can be an important

step in increasing the owner's understanding of how a certain data point is classified as fraudulent or not, ultimately aiding their decision-making.

1.2.4 Performance

The existing solutions for anomaly detection rely on real-time performance in order to be satisfactory. Simply training an ML model with all available data to predict whether a data point is fraudulent or not might be possible and even yield good accuracy, but not fast enough to be implemented in the existing environment. Network traffic changes rapidly and the anomaly detection needs to operate in real-time. This work aims to find smaller groups of features, that contain information that can be used to determine if a data point has originated from a scripted attack. Doing this allows for real-time analysis.

1.3 Use case

The aim of this work is to aid existing systems in detecting scripted activity in web traffic at the web security company Castle. The main focus is finding groups of features that together can reveal this kind of activity upon closer inspection limited to these features. Existing solutions at Castle can filter the activity based on groups of features, and this can reveal otherwise hidden patterns. In a data set with n features, the number of different combinations of features is $2^n - 1$. Examining all these combinations when n can be in the hundreds is not tractable, so the main purpose of this work is to find groups that capture the essential information of scripted attacks. To put it in simple terms, the result of this study will take web traffic as input and output a number of groups that the existing systems can examine further.

1.4 Scope and limitations

There are some boundaries to this work that are important to establish. The purpose of this study is to develop a strategy to dynamically reduce the features into one or more subsets of the original feature space. However, the application of the resulting feature subsets is left out of the scope of this work. What Castle typically delivers to their customers is not primarily a service that simply blocks some users' access to a website. It is rather a framework with tools like risk scoring that the customers themselves can use to make educated decisions and rule sets to ban or allow certain activities. It is also outside the scope of this work to implement a solution that runs in Castle's system directly, but rather a proof of concept that can be built upon.

1.5 Report structure

The rest of the thesis is structured as follows: Chapter 2, Background, presents the field of research on a shallow level, as well as briefly discussing existing technology in the field. After follows Chapter 3 on Related Work, which will outline

the findings of a literature review, detailing the exploration of existing research and discussing their results.

In Chapter 4, Data, the data used will be described, both in origin, structure and contents. Chapter 5, Methodology, will describe the process of developing a dynamic feature reduction pipeline for anomaly detection, describing all steps in this pipeline. All results derived from this research will be presented in the Chapter 6, Results. In Chapter 7, the results and their importance are analysed and discussed. Finally, in Chapter 8, Conclusions and Future Work, conclusions are drawn and applications in future work is discussed.

Chapter 2
Background

This chapter will explain concepts, terms and frameworks used in the rest of our work. Its purpose is to provide a foundation upon which the rest of the work is built and is helpful to understand the more complex terms and technologies. The chapter will cover machine learning concepts in a broad sense, since machine learning is foundational in this work. It will also cover a few ways to think about feature importance, what it is and different ways of calculating it. Information criterion is explained, as it is an important way of comparing different ML models to each other. Lastly, the chapter will also cover some frameworks and industry details used in this study.

2.1 Machine learning concepts

In this work, much emphasis is put on machine learning and the investigation of its usability in web security applications. There are a few terms that are central in the field of machine learning and will be used to explain the functionality and behaviour of the implementations in this study.

2.1.1 Loss

In statistics, loss is a penalty for incorrect classification of a data point with respect to the correct class label. The concept is central for the *learning* part of machine learning; in order for a ML model to learn, it typically needs some kind of metric to determine how well it is performing in order to adjust itself into a better performing version.

In order for an ML model to perform well, the loss function must typically take into account both bias (oversimplification) and variance (over-fitting) to generalise to new data, see Figure 2.1. For classification purposes, the loss function is typically cross-entropy, which is a measurement of the difference between two distributions for a given random variable [5]. The loss function is used when training ML models as a comparative tool to measure the classification performance.

2.1.2 Regularisation

Regularisation is a part of the training process in ML and is used to restrict a model from overfitting by penalising the complexity of a model. It can be seen as an extension of the loss function [8]. Generally, we can say that $Regularisation = Loss + Penalty$. The goal of regularisation, like the loss function, is to balance bias and variance in the model, so it does not put too much nor too little emphasis on certain parameters, see Figure 2.1. There are different versions of regularisation, and thereby different versions of penalty. The three versions discussed in this work are:

- L1 (Lasso)
- L2 (Ridge)
- Elastic Net (L1 + L2)

L1 regularisation, or Lasso, is expressed as $L1 = Loss + \lambda \sum_{j=1}^{m} |w_j|$, where λ is the regularisation strength and w_j are the model weights. It makes some coefficients zero, which means the model will ignore them. In that sense, L1 performs automatic feature selection to some extent [8].

L2 regularisation, or Ridge, is expressed as $L2 = Loss + \frac{1}{2}\lambda \sum_{j=1}^{m} w_j^2$, where λ is the regularisation strength and w_j are the model weights. Unlike L1, L2 keeps coefficients close to, but not exactly, zero, meaning they have low impact but are not completely set to zero.

Elastic Net is the combination of the two as $Elastic = r \cdot L1 + (1 - r) \cdot L2$, where r controls the ratio between the two. While often better than L1 or L2 alone, it is sometimes more challenging to control, with one extra parameter to handle [8].

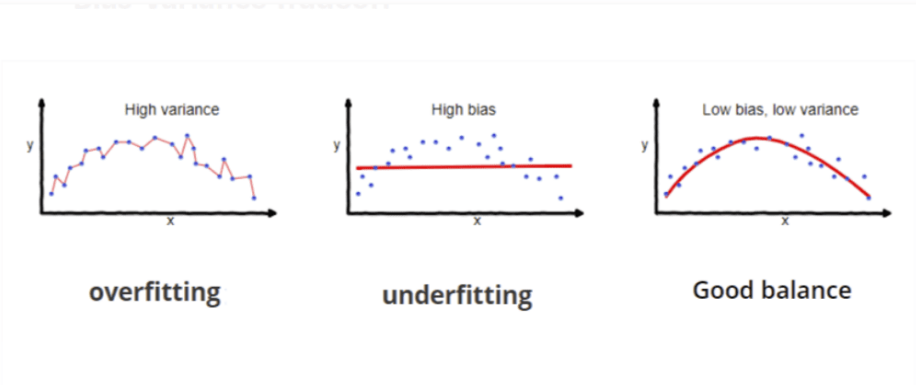

Figure 2.1: Bias and variance example [22].

2.1.3 Supervised learning

The models used in this work are all supervised learning models. Supervised learning is a kind of machine learning, where labelled data is used to train ML models to classify data or predict outcomes [13]. It works by mapping training data to correct outputs, and throughout the training process, the model adjusts its internal weights in order to better predict the correct class of input data based on its own accuracy, measured with a loss function. The opposite of supervised learning is unsupervised learning, where there is no labelled data, or "correct answers", and the model tries to cluster similar data points together.

2.2 Machine learning models

ML models are used in several places in our work. Here the different model types present in this work are presented and explained. Later in this work where the models are used their specific purpose will be explained. The models used are logistic regression, decision trees and gradient boosting.

2.2.1 Logistic regression

Logistic regression (LRG) is a binary classification model, meaning that given a data point, the model will predict which one of two classes the data point belongs to. In our case, the classes might be whether a data point is part of an attack or not, labelled as a binary 1 or 0 corresponding to True or False. LRG is a type of learning method called supervised learning where the algorithm needs to be given a target class for each data point during training [4]. The model is based on the logistic function $\sigma(t) = \frac{1}{1+e^{-t}}$, see Figure 2.2, where t is a linear function of n variables. So $t = \beta_0 + \sum_{k=1}^{n} \beta_k x_k$. Now the function can be interpreted as a probability, the probability of belonging to class 1. A high value for t gives a high probability of belonging to class 1, and a low value for t gives a low probability of belonging to class 1, thereby high probability of belonging to class 0 [4].

To be able to use the model it must be fitted, meaning the weights, β_k, must be adjusted so that data points belonging to class 1 and class 0 will be classified as class 1 and 0 respectively. With a fitted model a previously unseen data point can be processed by the model, and the model will return a probability. If that probability is greater than 0.5 then the data point will be classified as class 1 otherwise class 0 [4].

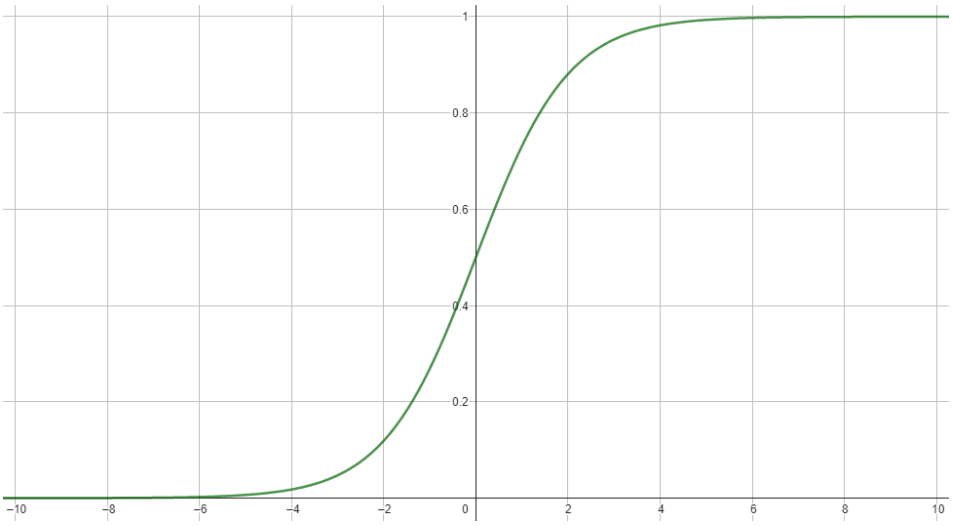

Figure 2.2: The logistic function $\sigma(t)$ in the interval $t \in [-10, 10]$.

2.2.2 Decision tree

A decision tree is a non-parametric supervised learning algorithm [12]. It gets the name from its tree structure which is hierarchical with a root node, internal nodes and leaf nodes, see Figure 2.3. It learns to classify a data point based on a certain rule set that is built up during its training process and is represented as the tree structure, where each rule is represented by a node. The decision tree then splits the data based on how the data points fulfil the rules of the nodes, and when done it has sorted the data set into as many parts as there are classes. A simple example is the decision tree in Figure 2.3, where the classification is a binary question of whether to play tennis or not. The formation of the tree in the training process can be controlled with different criteria, such as stop criteria and split criteria. Split criteria controls how the split points are decided, an example of this can be that a minimal number of data points should be contained in each node after the split. A stop criterion controls when the tree formation is finished, it can be a desired maximal tree depth for example.

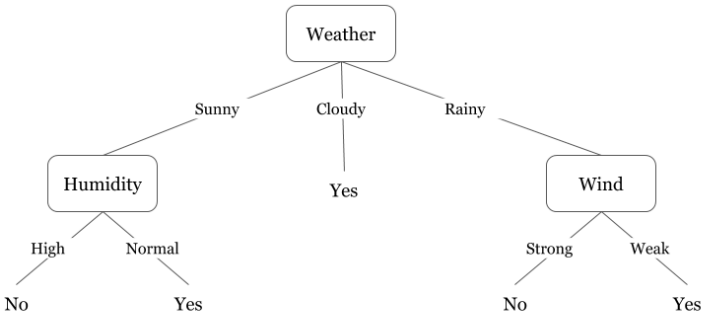

Figure 2.3: Decision tree to decide whether to play tennis or not.

2.2.3 Gradient boosting

Gradient boosting is a type of machine learning model architecture which consist of three elements: [10]

- A loss function
- A weak learner
- An additive model

The loss function depends on the kind of problem, but for classification purposes logarithmic loss is commonly used. The weak learner in gradient boosting is a decision tree. The decision trees are constructed and fitted in sequence. After each iteration the previous are evaluated with the loss function and the next decision tree is trained with that information. The additive model allows the new trees to be added to the model according to this principle, and once added they are not changed. The relative contribution of a tree to the following tree in the additive model is controlled with a parameter called `learning rate`. To make predictions of previously unseen data points the gradient boosting model uses all of the trees created during the training phase [10].

2.3 scikit-learn

scikit-learn is an open-source Python module for machine learning [19]. The models described in section 2.2 are all implemented using scikit-learn.

2.4 Feature importance

When working with ML models, a problem that can occur is that they are treated as black box functions, where data is put into them and class predictions come out, with little to no understanding of what happens in between. This can be especially difficult with complex model types, such as ensemble methods. In order to understand the model predictions better, it is useful to have a metric for how much each individual feature contributed to that prediction. To measure the individual contribution of features, the concept of feature importance is a useful metric with many variations [21].

The typical practice with more complex ML models is to treat the model itself as a black-box function, and apply a model-agnostic feature importance variation after the fact [21]. In this Section, we will describe a few important feature importance variations and also discuss the process of combining them in an ensemble method.

2.4.1 Mean decrease impurity

Many ML models in the scikit-learn module use a built-in feature importance score based on Mean Decrease Impurity (MDI). Depending on the context it is used for, the impurity criterion can be Gini impurity (for classification) or variance reduction (for regression). MDI is advantageous because of its low computational cost. The MDI values are already computed when the model is fit with training data and saved as an attribute of the model object, so no extra calculations are needed after training. The clear disadvantage is that MDI is vulnerable to high cardinality features, that is features with a large number of possible values. MDI tends to overstate the importance of such features [19].

2.4.2 Permutation importance

Permutation Importance (PI) is another feature importance variation that specifically combats the high cardinality feature problem, although being more computationally costly than impurity methods. The basic principle is permuting the rows for each feature column a number of times, while measuring the performance of the model before and after the permutation. The loss from permuting a feature column signifies the importance of that feature. PI is also model agnostic, which makes it applicable to all types of ML models [21] [2].

2.4.3 Shapley Additive Explanations

Another approach used for feature importance is SHAP (SHapley Additive exPlanations). SHAP uses Shapley values to interpret ML models. The Shapley value of a feature is determined by the contribution of that feature in the model. It is calculated by evaluating every possible combination of features and then calculating the features' contribution by comparing the model with and without the feature [15]. To be able to compare the model with and without a specific feature SHAP has to train a ML model for each of the possible feature combinations. This leads to 2^n models trained where n is the number of features. The exponential

nature of this method could be a problem where n is large, but there are methods for speeding up the process. For example when using Python the library SHAP [15] makes this a viable method by approximating, optimising and sampling.

2.5 Information criterion

In order to find a data-driven strategy for feature selection, information criterion comes in handy, since it provides a way to evaluate different machine learning models not only by their prediction accuracy, but also by taking their complexity into account. This means that very accurate but extremely complex models might not be preferable to a slightly worse performing but way simpler model. These metrics don't provide an absolute score of the models, but rather comparative scores that relate different model variations to each other. This will be useful when determining which features should be included in the groups that this work ultimately will deliver as a result, by comparing models trained on different subsets of the data.

2.5.1 Akaike/Bayesian information criterion

For information criterion two versions are considered. The first one is Akaike Information Criterion (AIC), defined as $AIC = 2k - 2ln(\hat{L})$ where k is the number of (internal) parameters and $ln(\hat{L})$ is the logarithm of the maximum likelihood of the model [1]. The second one is Bayesian Information Criterion (BIC), defined as $BIC = kln(n) - 2ln(\hat{L})$ where n is the number of data points, k is the number of parameters and $ln(\hat{L})$ is the log-likelihood [24]. Since a lower AIC/BIC number indicates a better performing model and because the negative terms of both variations are equal, the key difference is that BIC is logarithmic in n, which is the number of data points. As long as we have more than 7 data points then $n > 7 \implies 2k < kln(n)$, which means that BIC will punish complex models (models with high k) more than AIC will. For this reason, BIC is the metric used in this study. Promoting less complex models aligns with the goals of this work, since smaller groups of features will reduce the computational complexity for future applications that might need real-time performance.

2.6 One-hot encoding

In this work, one-hot encoding is a preprocessing step performed to the data by the scikit-learn library. One-hot encoding is a procedure that encodes categorical features into a numerical array [19]. Each feature is expanded into many new "dummy" features that contain binary values corresponding to the values of the original feature, see Figure 2.4. One-hot encoding is necessary for many ML models to work properly, since they need numerical data types rather than categorical.

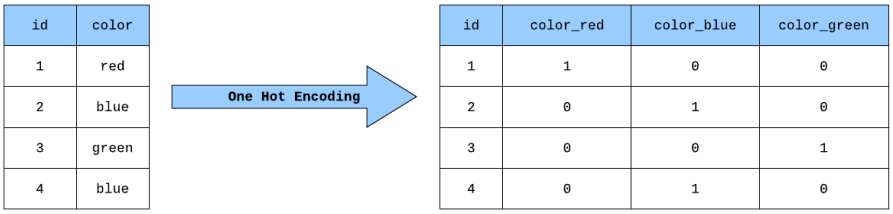

Figure 2.4: One hot encoding example [17].

2.7 Pandas

Pandas is an open-source Python module for data analysis and manipulation. In this work, data is stored in the data structure `DataFrames` from Pandas, which contains two-dimensional, size-mutable, tabular data as well as labelled rows and columns [18]. This data structure supports common matrix operations while also having support for additional built-in operations. Table 2.1 shows a segment of a data frame.

Table 2.1: A DataFrame example.

	FINGERPRINT_TIMEZONE	IP_CITY_NAME	MOBILE_WIFI
0	America/Halifax	Sewell	True
1	America/Bogota	St Louis	False
2	Asia/Almaty	Aurora	False

2.8 Web traffic

Many websites today have millions of concurrent users, and it can be a tough challenge for website owners to distinguish honest, human users from malicious ones, either real people with bad intentions, or bot attackers. What classifies a user as malicious depends on the website and can vary between different attacks, but in general a malicious user is someone who tries to exploit a vulnerability in the website in order to benefit from it for reasons including, but not limited to, monetary, power-related or personal ones.

When interacting with a website there are many things that can be done, and such an interaction is called an `event`. When harvesting this traffic, each event constitutes a data point in the resulting data. This can be the act of logging in to the website, signing up to it, paying for a product or service, etc. Since the communication between the website and user takes place over the internet, there is a possibility for either party to save the communication information in logs. These logs contain information of varying kinds about the users, the geographical location of the connections used by them, as well as information about the hardware

behind the communication. Every such kind of information category is called a `feature`.

Chapter 3

Related Work

An important part of the process of implementing a solution to solve the problem described in Chapter 1 is reviewing the previous work related to this topic. This chapter will describe some of the existing technologies in the field, how they work and outline how they can be helpful when applied in this work. This chapter brings up how different methods for feature importance can be combined, and how groups of features can be selected.

3.1 Ensemble strategies for feature importance

With many feature importance variations, it is desirable to be able to use all of them together in a manner that captures all of the relevant information. The way the combination is performed is not trivial, and that is why there are many ensemble strategies. In this Section, we will discuss a few of those proposed in [21]. Mean value strategy is the simplest ensemble strategy implemented here, it just takes the mean value from all available feature importance values from all variations of all models.

The second strategy is the majority vote. It works by comparing the pairwise similarity between the different feature importance contribution vectors. If the similarity is statistically significant, that is p-value < 0.05, True is inserted into a vote matrix, else False, see Figure 3.1 [21]. This means that only the contribution methods that, in a majority, agree with each other, will contribute to the final feature importance score. Those in a majority are averaged into a final feature importance score.

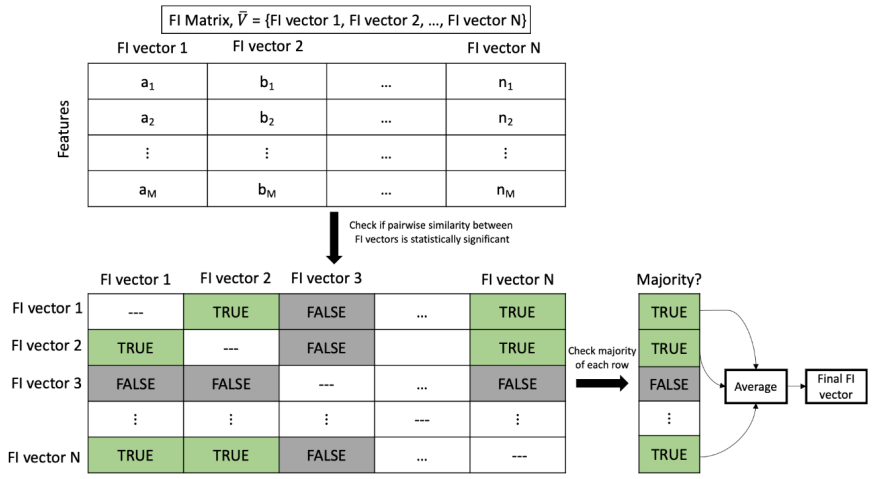

Figure 3.1: Majority vote implementation [21].

3.2 Feature selection

With a list of final feature importance scores, the remaining problem is selecting a subset of the original feature space. A data-driven approach for this is using BIC, as described in 2.5.1. These metrics give a good balance between accuracy and model complexity. They are however only tools in order to pick a model which in turn is trained with a subset of the original feature space. With 204 features in the data set, finding the combination of features which yields a model with the lowest BIC score would mean checking models with every combination of features, and thus training $2^{204} - 1$, or over 2.6×10^{61} models, which is not tractable regardless of hardware. To combat this combinatorial problem there needs to be a way to find relevant feature groupings without checking every variation.

3.2.1 Forward/Backward selection

Forward and backward selection are greedy step-wise regression algorithms. They both work by iteratively building new models with different feature subsets and comparing their performances [9]. In this work, the metric used to compare the different models is BIC. Forward selection starts by training one model for each feature with that feature being the only training data available. It compares all the models to each other, and the one with the lowest BIC score contributes its feature to the new baseline model. Now the model tries every combination of that feature with the other features in a two-feature-model and picks the one with the lowest BIC score to contribute the new feature to the baseline. This process repeats until no features are left. Backward selection works in a similar manner, but starts with a model trained on all features as its baseline and greedily reduces one feature at a time by removing the one where the model has the highest BIC score without it. This is done until no features are left.

3.2.2 Bidirectional elimination

In an effort to combat the shortcomings of forward/backward selection, bidirectional elimination aims to combine the two in a more optimal fashion. In each iteration step all possible additions and deletions are examined and the best option locally is chosen [9].

Chapter 4
Data

This chapter will describe the data sets that were examined in this work. It will present how the data was retrieved, its structure and contents, as well as describing a few of the examples that were worked on. The chapter will provide a foundational understanding of the data in order for the remaining chapters to be understood more clearly.

4.1 Origin

The data sets were provided by Castle and consisted of real web traffic data from one or many of their customers over a chosen period of time. The main goal when choosing a data extraction was that it should cover a time span where attacks occurred, in order to get a picture of what such traffic could look like. This was something that Castle could provide from their archive of web traffic data from different customers.

4.2 Structure and contents

The data was received as a Comma Separated Values (CSV) file, containing columns representing all available features of the data and each row representing an event. This means each event becomes a data point. All values were strings and contained the relevant information in that category. An example of such a value could be `'US'` in row 42 and column `'IP_COUNTRY_CODE'`. The features can be categorised into address- , user- and device-specific, depending on the context of their extraction. IP related features are address-specific, and they are typically harvested from the network traffic information from an event. This can be features such as Internet Service Provider (ISP), Internet Protocol (IP)-address, IP time zone, etc.

There are also device-specific features, which can be harvested from the information provided by the user system itself. This however can easily be obfuscated if the user's intent is malicious. Therefore there are ways to track the hardware itself with Operating System (OS) and hardware-specific features by asking browsers to perform a number of tasks that utilise functionalities from hardware and OS

and thus exposing the underlying computer architecture, as described in [6]. Finally, there are also user agent features, harvested from account email addresses or similar information. Depending on the context, some features are captured in a separate JavaScript Object Notation (JSON) object in the CSV file, such as the column `FINGERPRINT_PARAMS`, which contains all device-specific features, also known as the device fingerprint [3].

4.3 Example data sets

Below are example data set descriptions, outlining a few examples of data extractions that were worked on in order for this work to take shape. For security reasons, Castle anonymised some parts of the data with an unknown hashing algorithm before using in this work so that it would not contain sensitive information. The sensitive columns in the data sets were the IP-addresses and email addresses.

4.3.1 Example data set 1

The first data set came from a 23-hour long window of the web traffic from one customer. During that period, the activity levels rose to very high numbers a few times in quick succession. This data set consisted of 2 578 598 data points. It contained 204 different features. In Figure 4.1 the activity levels of this data set can be seen.

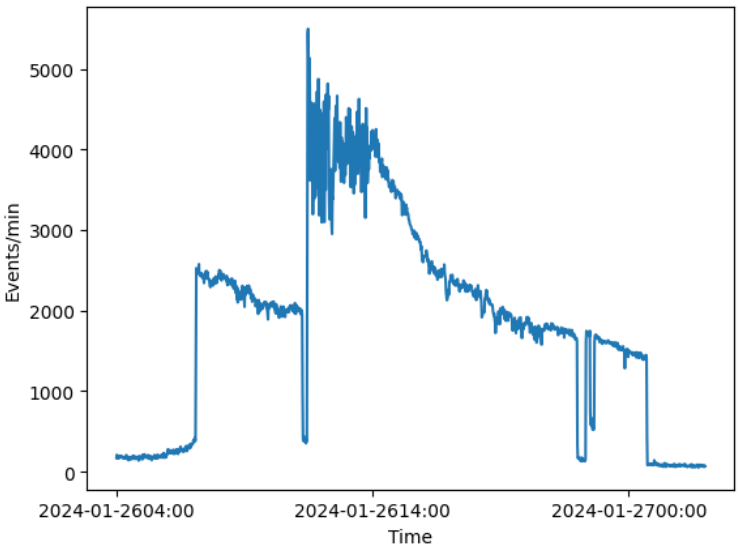

Figure 4.1: Example data set 1: Activity level for all events.

4.3.2 Example data set 2

The second data set came from a two-week period from a specific customer. The normal activity level had a natural 24-hour cycle with moderate activity levels and a few spikes, however, they are more local in scale compared to example 1. This data set contained many different event types and consisted of 1 432 086 data points. It contained 101 different features. In Figure 4.2 the activity levels of this data set can be seen.

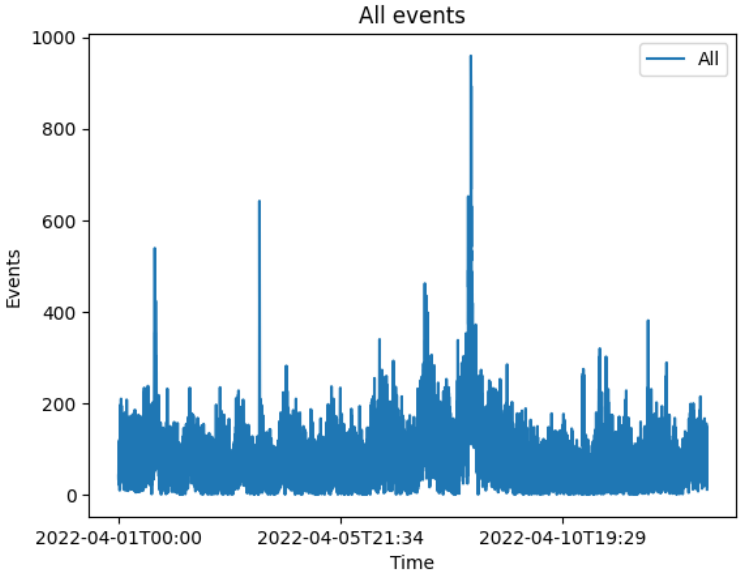

Figure 4.2: Example data set 2: Activity level for all events.

4.3.3 Example data set 3

The third data set came from a four-week period from a specific customer. The normal activity level was very low, with a 24-hour cycle like example 2. There were a few spikes, albeit also low in activity with respect to the amount of events compared to previous examples. This data set contained many different event types and consisted of 21 168 data points. It contained 203 different features. In Figure 4.3 the activity levels of this data set can be seen.

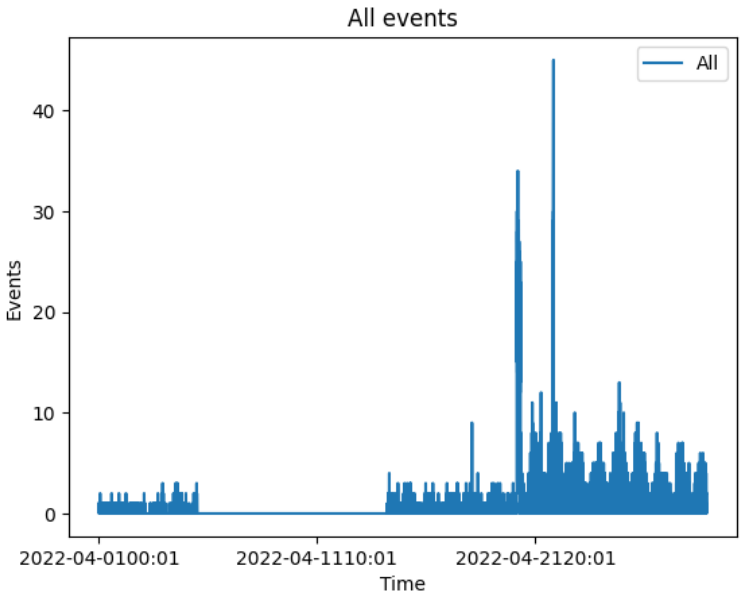

Figure 4.3: Example data set 3: Activity level for all events.

4.3.4 Example data set 4

The fourth data set came from a one-week period from a specific customer. It contained a large peak of activity, but compared to the other data sets the peak was much more spread out over time and less distinct visually. This data set contained many different event types and consisted of 1 215 344 data points. It contained 266 different features. In Figure 4.4 the activity levels of this data set can be seen.

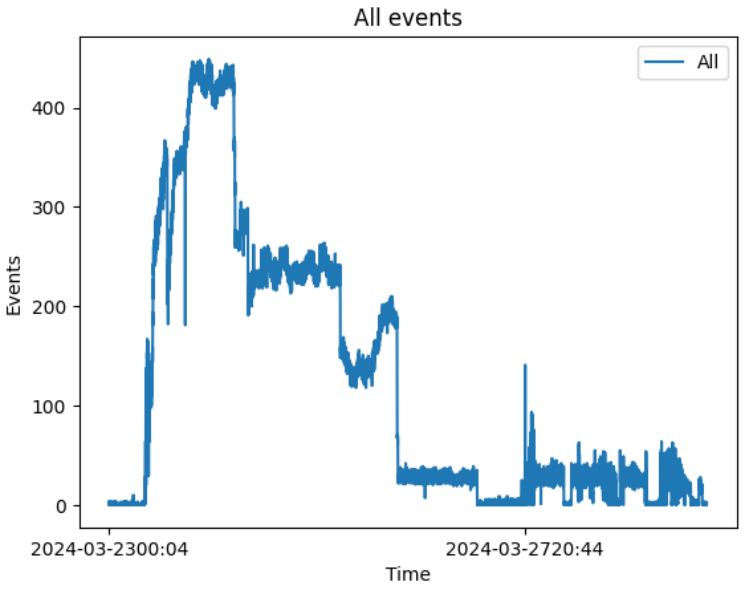

Figure 4.4: Example data set 4: Activity level for all events.

4.3.5 Example data set 5

The fifth data set came from a 90 minute period from a specific customer. It contained a very clear peak of activity. Compared to the other data sets this was a small data set with a short peak of activity. This data set contained many different event types, and consisted of 4336 data points. It contained 243 different features. In Figure 4.5 the activity levels of this data set can be seen.

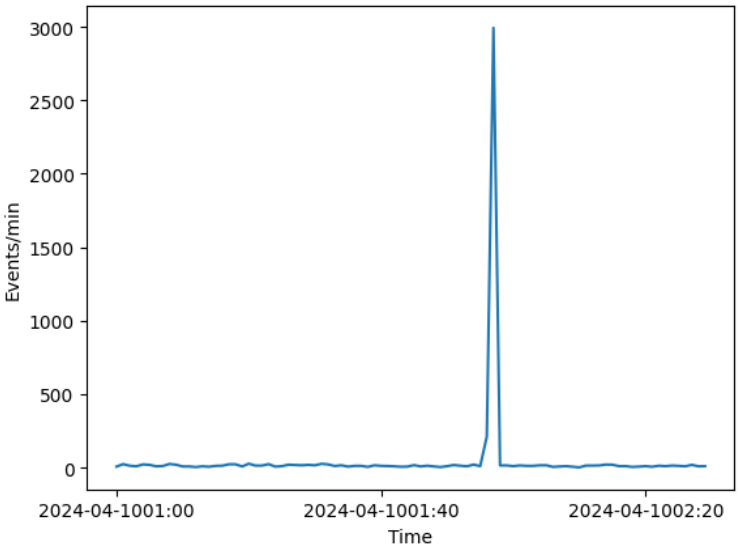

Figure 4.5: Example data set 5: Activity level for all events.

Chapter 5
Methodology

The methodology of this study can best be understood from its pipeline, see Figure 5.1, which consists of seven main steps. Each step will be discussed in this chapter, their inner workings and motivations will be described and their functionality and purpose will be explained.

The fundamental idea is the following:

- Train ML models to predict something that could indicate a scripted attack.
- Extract the information those predictions are based on.
- Use this information to determine which features of the data are the most important in order to make those predictions.
- Build groups of the most important features in order to find groups that capture the characteristics of a scripted attack.

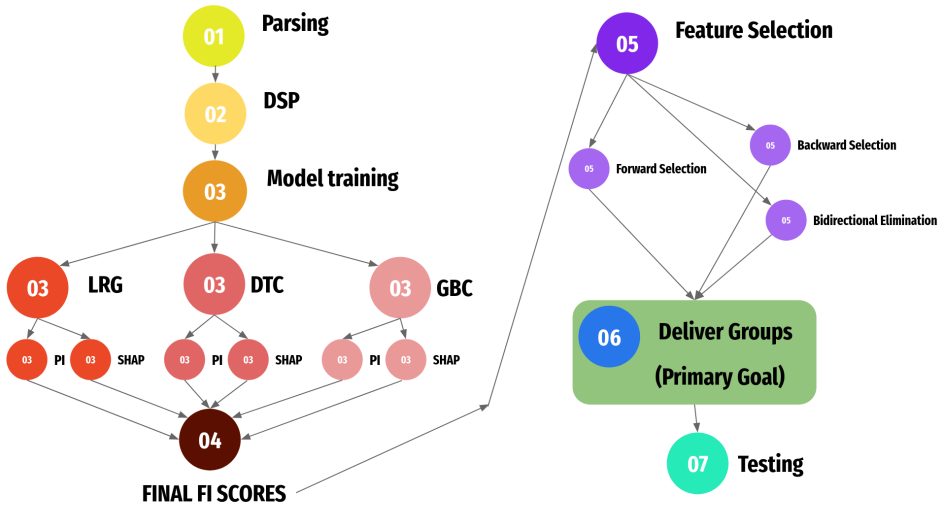

Figure 5.1: Pipeline overview.

All code in this work will be run on a laptop equipped with an Intel Core Ultra 7 155H processor and 32 GB of RAM.

5.1 Data processing

In every data science and machine learning application, data processing is an essential part. It serves a purpose for both understanding the characteristics of the data as well as preparing it to fit the remainder of the pipeline it is going to go through. For these reasons, it is paramount to have a thorough understanding of the data contents as well as parsing it in a manner that works well in tandem with other frameworks and formats. In this section, the data processing will be described in terms of parsing, Digital Signal Processing (DSP) and preparation.

5.1.1 Parsing

The first step in any machine learning pipeline is all about data parsing, in order to convert the data into formats that the rest of the pipeline can understand. The data for development purposes described in Chapter 4 have a CSV format, containing information about every single event in an extracted time series from Castle's data traffic. To read this into a format that most Python libraries and frameworks understand, it is converted into a Pandas data-frame. In this part of the pipeline, the data-frame in question is a matrix with the raw data, each row represents an event and every column represents a feature. It is this data-frame every other part of the pipeline will work with, so for clarity it will from here on be called `X`. From the data-frame `X` all events are aggregated in one-minute intervals and output to a text file which contains a minute-precise time stamp and the corresponding number of events in that minute. This is needed to convert the problem into a supervised problem, which will be described in Section 5.1.2.

5.1.2 Conversion to a supervised learning problem

The data received from Castle has all the information needed about each event in a certain time frame, but no information about whether it is part of a scripted attack or not. This is no surprise, if such information was available, the problem would be solved already. However, we still need to map every data point to a class label, or target value, in order for this problem to be classified as a supervised machine learning problem. An unsupervised approach for this application is still much more of an open problem, as will be discussed in Chapter 8. The approach steps in this work also rely heavily on previous work that all demands a supervised problem.

There needs to be a target label for each data point, but there are none available directly in the data. This is where the concept of a proxy comes in. A proxy target is an attribute that is inherent in the data and can serve as a substitute for the intended target label that is "attacker or not attacker", but is unavailable in this case. In this work, activity level will serve as a proxy target.

5.1.3 Digital Signal Processing

This subsection will describe the process of finding proxy target values from the activity levels. The implementation, however, is carried out in such a manner that any proxy target could be used if necessary, the pipeline would work for such a use case too. This will be discussed more in Chapter 8.

To create proxy target values for all data points there are a few options. In the case that the data shows a clear spike in activity in certain time spans, it is easy to read out the activity level in a time interval.

For data sets with many different event types, and for a general approach, the activity level itself can serves as a proxy for indicating an attack. In general, if we see a spike in traffic where the activity is much higher than normal we assume that there is an attack going on. There could also be natural spikes in the traffic, but we will return to that later. For now, we assume that most of the data points in a spike in traffic are part of an attack, and thus give all the data points in a spike the target value 1 and the others the target value 0.

For this general approach, there are a few ways of detecting activity spikes. For some data sets it is very obvious which parts are attacks upon visual inspection, and simpler solutions produce good results. Such an example can be to simply classify a spike where the moving average of activity levels is higher than a certain global threshold. This threshold can be a fixed value or the mean value of a sliding window plus a number of standard deviations. The solution picked for each data set depended on its characteristics.

5.1.4 Preparation

The data-frame from previous steps is manually cleaned of columns, which correspond to features of the data, that can be deemed unnecessary with some domain-specific knowledge. For example, the column `'EVENT_CUSTOM_NAME'` can be removed at once, since it contains manual annotations made by Castle. These are not readily available in live data, and will therefore not be of much use when this pipeline is running live. Other examples of removed columns are `'EVENT_CREATED_AT'`, which contains the time stamp of the event, a feature that will not be of use after activity levels are extracted. The aggregating of the data in minute intervals is motivated by the fact that for some data sets there were very few events per second, resulting in an overly granular data series that did not yield useful target vectors.

After parsing and DSP, the CSV files delivered by Castle described in Chapter 4 were first vectorised. The vectorisation consisted of one-hot encoding the data into a new binary matrix, using the one-hot encoder in the sklearn pre-processing library, see Section 2.6. For performance improvements described in 2.4, the one-hot encoding was not applied directly to the data, but rather passed to an sklearn pipeline structure for future use.

One thing to consider is the size of the data set that is worked on. In some cases, even after manual cleaning, they are simply too big to be worked with on a consumer computer, in which case there is a need to downsize the data. This can be achieved with sampling. However, this must be done with caution. If the number of samples of each class does not follow an even distribution, imbalance can become a problem. In practice, this means that the classifier tends to classify data points as the majority class more often, and essentially treats the other class data as noise. In this case, the classification process is binary, so it is desirable that the distribution of data points is even between spikes and non-spikes [14]. To combat imbalanced classification, a sampling scheme called random under sampling is implemented. This scheme consists of removing random samples of the majority class until the distribution between classes is even [14].

Lastly, in order for the pipeline to be able to run on different machines with different performance levels, a custom down sampling scheme is implemented to reduce the data set to a selected size so that it runs in a reasonable time frame for any machine, still with the same distribution of data points as discussed earlier. This way, one parameter in the pipeline controls the final data set size so it can be run on a low-end laptop or a strong computer depending on the resources available. This down sampling scheme is controlled with a random seed for reproducibility reasons, but can of course be changed to a secure protocol if necessary.

Now, the vectorised and subsequently one-hot encoded data consist of a matrix where each column is a dummy variable generated from its original feature and each row is a data point. The target vector is a binary vector telling whether that data point comes from a spike in traffic or not. The data is split into training and test data using the sklearn train/test split functionality with a $\frac{80}{20}$ ratio, meaning that 80% of the data is reserved for training and 20% for testing. The split function also shuffles the data so all data points no longer are ordered based on their occurrence in time.

5.2 Model training

When the data is prepared and ready, a few machine learning models are trained with it, in order to predict whether it occurs in a spike and thus gain some valuable information on important features. The models used to determine feature importance are logistic regression (LRG), decision tree classifier (DTC) and gradient boosting classifier (GBC). This way there is a variety of model types that all have some different characteristics. LRG is a linear model, DTC is a tree-based model and GBC is an ensemble model [19]. The hyper-parameters used in the models can be seen in Table 5.1. All model implementations come from the scikit-learn module described in 2.3.

Table 5.1: Models and hyper-parameters.

Models	Hyper-parameters	Values
DTC	Max depth	7
	Min samples split	2
GBC	Max depth	7
	Learning rate	0.1
LRG	Max iterations	200
	Regularisation	L2

5.3 Feature importance

For each trained model, two different feature importance metrics described in 2.4 were calculated. The PI scores were calculated with the permutation importance function in scikit-learn. It essentially takes a trained ML model and makes predictions on the data before and after permuting a feature in it. Using a scikit-learn pipeline structure with the one hot encoder as one part and the trained model in question as another, this works well since the permutation importance permutes the feature before one-hot encoding it, making the process much faster than permuting all different dummy features. The SHAP scores were calculated by training an explainer model from the SHAP module and reading out the importance scores for each feature [15]. Since SHAP did not support one hot encoding with a scikit-learn pipeline like PI did, the data was first one hot encoded and the scores for the dummy features were added to the total sum of their respective origin feature after calculated.

For stability reasons, it is recommended to run 50-100 permutation rounds for PI specifically [19]. This can be difficult to achieve on a consumer-grade computer when data sets are very big, but the goal is to come as close to this benchmark number as possible with the hardware available. Because of the down sampling scheme mentioned in Section 5.1.4, this goal was achieved for a consumer-grade PC. These two scores were merged with the majority ensemble strategy described in 3.1 as long as the number of contributing scores were more than zero. If zero, it meant that the different scores disagreed too much with each other for the majority vote to work. This likely indicates bad scoring but to still output a final feature importance score list, the mean value was used in this special case. The ensemble method delivered final feature importance scores, essentially a list with one number per original feature of the data ranking them in importance.

5.4 Feature selection

The central goal of this work is to group the original feature space into subsets that can reveal patterns of fraudulent activity. For this, a list ranking the most important features is not enough, but a good step in the right direction. The rankings alone do not reveal what features, in a group constellation, contain the

most information. One option is to simply group the highest ranked features, but if are correlated they essentially contain the same information, and the result may not be optimal. To group the features, a few more sophisticated methods were implemented; forward selection, backward selection and bidirectional elimination. The three different groups delivered from these three methods along with the result from exhaustive analysis were the final results of the implemented pipeline.

5.4.1 Forward/Backward selection

Forward and backward selection was implemented and tested as described in Section 3.2. Because of the number of features, it is not practical to use forward or backward selection on all the features. Instead, the top 50 features from the list of feature importance scores are selected. For some implementations of forward and backward selection the algorithm stops if the BIC score can't be decreased by adding/removing a feature, another method is to control the maximum group size, the latter was used in this work. The main reason for this choice is to prevent the algorithms to get stuck in a local optima, which is a risk with greedy algorithms.

5.4.2 Bidirectional elimination

Bidirectional elimination efficiently prevents the process from getting stuck in a sub-optimal state where a choice of feature inclusion/exclusion which is locally optimal is not beneficial globally. However, this method also suffers from a local focus. If a feature is included and in the next iteration the best option is to delete that same feature, the process gets stuck in a closed loop where the same feature is repeatedly added and removed, from which no further progress can take place. This might not be a problem in some situations, but for this purpose, the global minima might not be found if this happens. This means that the stop criterion can be handled in different ways. In this work, inclusion or exclusion stops when the process gets stuck in such a loop such that no new group members can appear, or if a maximum number of iterations, in this work 30 iterations, have been carried out.

5.5 Evaluation

This section will describe the process of evaluating the pipeline and all its components, including target labelling, relevant feature retrieval, model training, generalisation of groups and natural spike avoidance.

5.5.1 Target labelling

For the data sets provided by Castle for testing and evaluation purposes, the characteristics of the attackers were known. For every data set, there was a known value to one or a few of the features, so that Castle with high certainty could classify those as attacker data points. In Table 5.2 the features and values, that with a high likelihood indicated an attacker, are presented. There can of course be other features that could label attacks correctly, but these were given by Castle

because they could classify the data points with high certainty using these feature values, after studying the data closely with their domain knowledge. These mainly functioned as a correct answer table in order for the target labelling procedure to be evaluated. Using this information, the process of target labelling described in 5.1.3 can be evaluated against the closest to a correct answer there is. The predicted labels and the labels provided by Castle are compared using precision, recall and f1-score [20].

Table 5.2: Attacker indicator features and values (some values hashed for privacy reasons).

Data set	Features	Values
1	CLIENT_TOKEN_ID	hashed
2	IP_CITY_NAME	'Lucknow'
3	mm_start-upper-left_b	'True'
5	IP_ADDRESS	hashed

5.5.2 Relevant feature retrieval

Since for all example data sets there is one or a set of features with specific values that indicate an attack, a relevant question is whether the pipeline found these features at all. This evaluation is simply a matter of studying the resulting groups and checking for every data set if the particular features are contained. It is also worth analysing what importance scores they are given.

5.5.3 Model prediction/information loss

When groups of features are retrieved, one way to test their quality is to select a subset of the original data, only containing the columns corresponding to the group member features, and then train a simple ML model, in this work a logistic regression model, with that data subset. The model trained on the group of features will be compared to a model trained on the full feature set, and it will also be compared to a model trained on the top 50 features retrieved from Section 5.3. This test is meant to evaluate if reducing the data set to only the columns in the groups retains enough information to classify the proxy target as well as the complete data set.

Chapter 6

Results

This chapter will present the results of this work. It will contain results of the target labelling, relevant feature retrieval and model training.

6.1 Target labelling

Spikes have been identified in every data set, as can be seen in Figures 6.1 - 6.5. These serve the purpose of giving a visual representation of what parts of the data was labelled as spikes. The quality of these labels will be evaluated with classification reports, but these plots help give a better picture of the target labelling results visually. When looking at the plots the spikes seem to be detected as expected. An exception might be data set 2 in Figure 6.2, where the data is noisy and spikes are harder to distinguish. Classification reports that comparing the detected spikes with the actual scripted attack data points will also be presented, in order to see how accurately spikes work as target proxy for scripted attacks. These can be seen in Tables 6.2 - 6.5. A summary with all macro average F1 scores for all data sets are presented in Table 6.1. This shows that for some data sets the target labelling performed poorly, while it was very accurate for other data sets.

For data set 4, there was no correct label provided by Castle, but the results for this data set will be presented in other sections anyway for completeness purposes and basis for discussion.

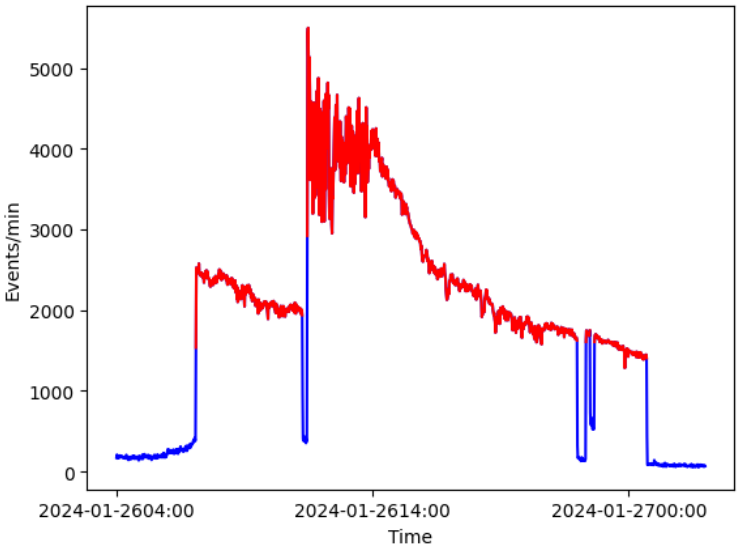

Figure 6.1: Detected spike locations in example data set 1. The red in the figure indicates that the web traffic was classified as attack and the blue is normal traffic.

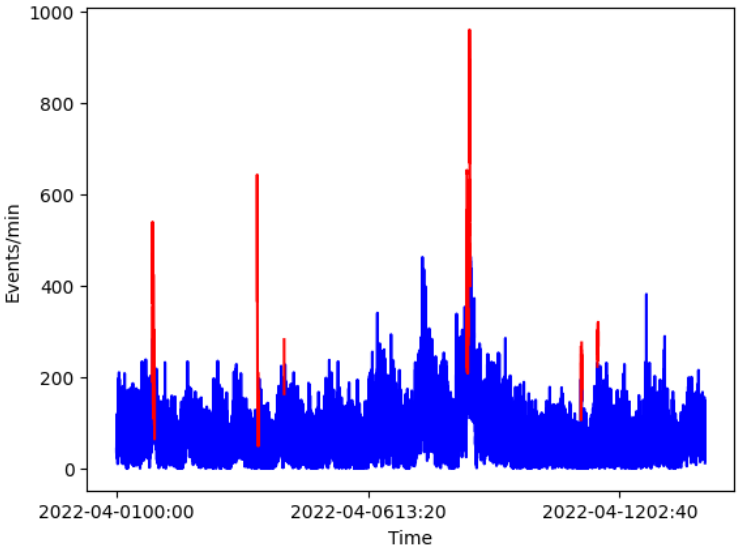

Figure 6.2: Detected spike locations in example data set 2. The red in the figure indicates that the web traffic was classified as attack and the blue is normal traffic.

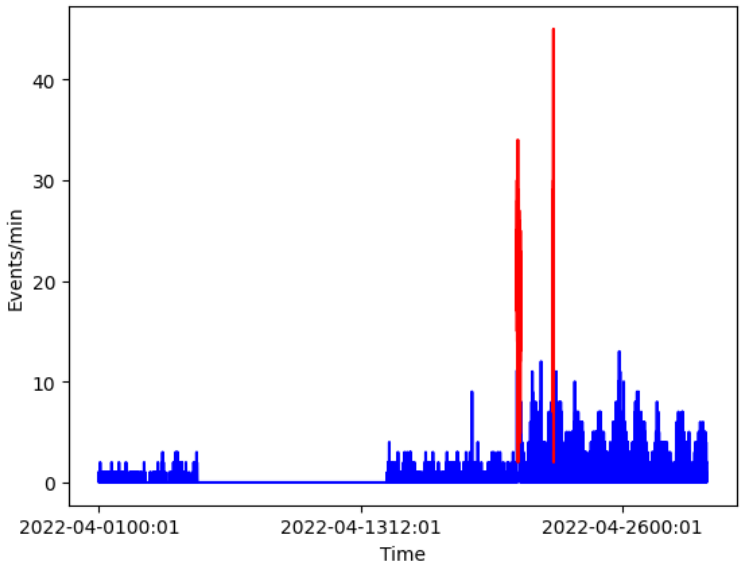

Figure 6.3: Detected spike locations in example data set 3. The red in the figure indicates that the web traffic was classified as attack and the blue is normal traffic.

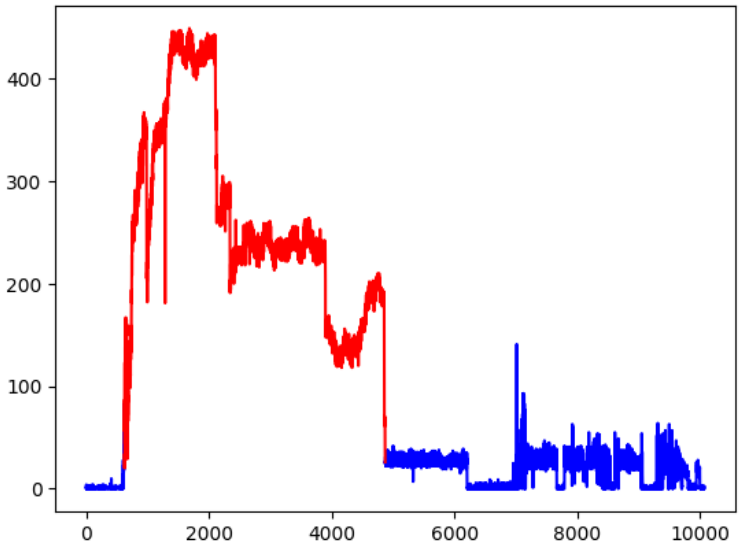

Figure 6.4: Detected spike locations in example data set 4. The red in the figure indicates that the web traffic was classified as attack and the blue is normal traffic.

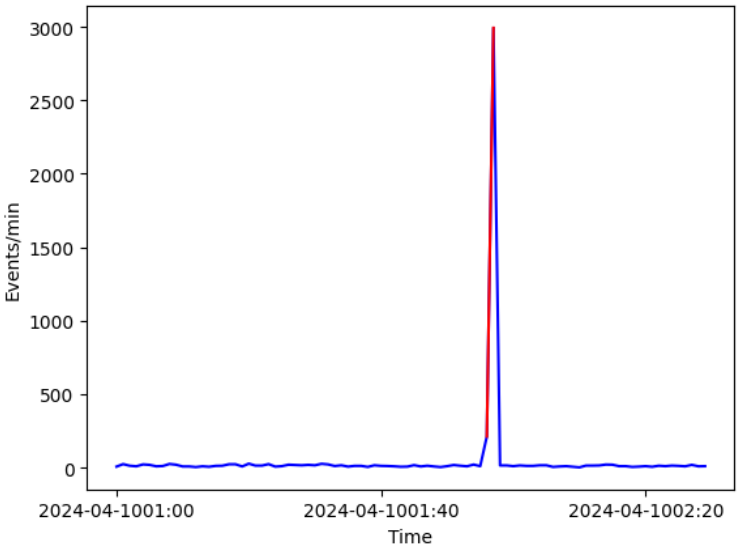

Figure 6.5: Detected spike locations in example data set 5. The red in the figure indicates that the web traffic was classified as attack and the blue is normal traffic.

Table 6.1: Target labelling macro average F1 scores for all data sets

Data set	Macro avg F1-score
1	0.57
2	0.51
3	0.94
4	N/A
5	0.94

Table 6.2: Classification report, example data set 1, automatic spike detection towards Castle's target.

	Precision	Recall	F1-score	Support
0	0.77	0.12	0.21	411864
1	0.86	0.99	0.92	2166734
Accuracy			0.85	2578598
Macro avg	0.82	0.56	0.57	2578598
Weighted avg	0.84	0.85	0.81	2578598

Table 6.3: Classification report, example data set 2, automatic spike detection towards Castle's target.

	Precision	Recall	F1-score	Support
0	0.92	0.96	0.94	1318280
1	0.10	0.06	0.07	113806
Accuracy			0.88	1432086
Macro avg	0.51	0.51	0.51	1432086
Weighted avg	0.86	0.88	0.87	1432086

Table 6.4: Classification report, example data set 3, automatic spike detection towards Castle's target.

	Precision	Recall	F1-score	Support
0	0.99	0.95	0.97	16196
1	0.85	0.98	0.91	4972
Accuracy			0.95	21168
Macro avg	0.92	0.96	0.94	21168
Weighted avg	0.96	0.95	0.96	21168

Table 6.5: Classification report, example data set 5, automatic spike detection towards Castle's target.

	Precision	Recall	F1-score	Support
0	1.00	0.81	0.89	1398
1	0.92	1.00	0.96	2938
Accuracy			0.94	4336
Macro avg	0.96	0.90	0.93	4336
Weighted avg	0.94	0.94	0.94	4336

6.2 Relevant feature retrieval

The plots from the different selection algorithms can be found in Figures 6.6 - 6.20. The resulting groups of features delivered by the pipeline with the different selection algorithms will also be presented. These can be found in Tables 6.7 - 6.9.

6.2.1 Feature selection

The plots from the selection algorithms are presented in Figures 6.6 - 6.20. The plots show the BIC score for the different selection algorithms and how the BIC score changes for each iteration. The group of features from the iteration with the lowest BIC score is the resulting group.

Figures 6.6 - 6.18 show the BIC scores for the different selection algorithms for all data sets. For data set 1, forward selection finds a minimum BIC score after five iterations, corresponding to a group size of 5. Backward also finds a minimum corresponding to a group size of 5, after 25 iterations, and the same is true for bidirectional elimination after five iterations.

For data set 2, forward selection finds a minimum fairly late in the process, corresponding to a fairly large group size of 12. Backward selection finds a minimum after about half of the iterations, again corresponding to a fairly large group size of 16. Bidirectional elimination finds a minimum after nine iterations, in which the process gets stuck because of the local nature of the minimum value, where the group size is 9.

For data set 3, forward selection finds a minimum corresponding to a group size of 5. Backward selection finds a minimum corresponding to a much larger group size of 17. Bidirectional elimination finds a minimum where the group size is 3.

For data set 4, the result follows a similar pattern to data set 3. Forward selection finds a minimum after nine iterations with a group of size 9, backward selection finds a minimum after 25 iterations with a group of size 5 and bidirectional elimination finds a minimum after five iterations with a group of size 5.

For data set 5, all selection algorithms have a linear relationship between number of iterations and BIC-score, leading to all of them having minimum values corresponding to group sizes of 1.

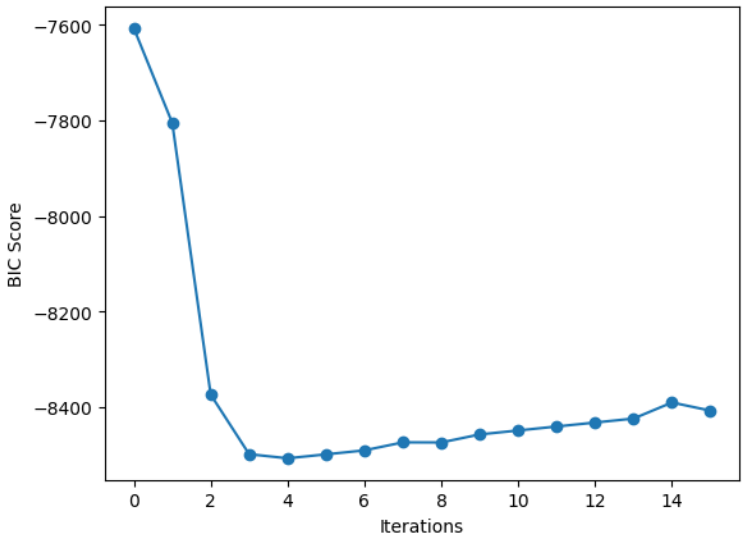

Figure 6.6: Forward selection, example data set 1.

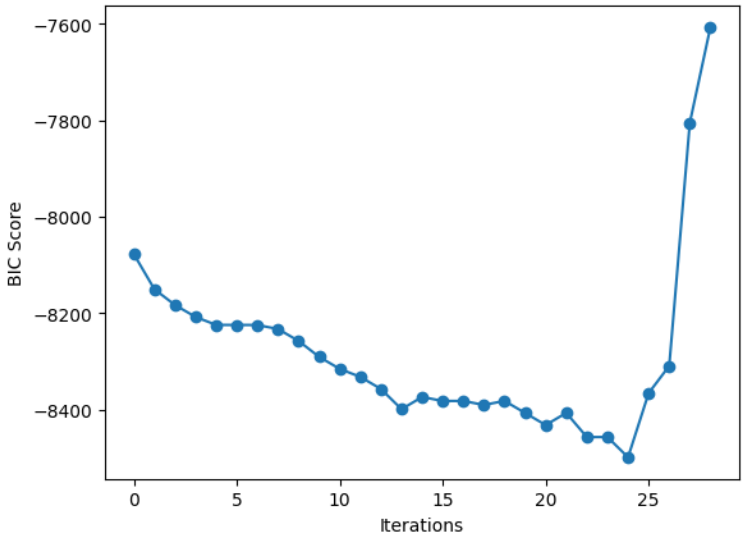

Figure 6.7: Backward selection, example data set 1.

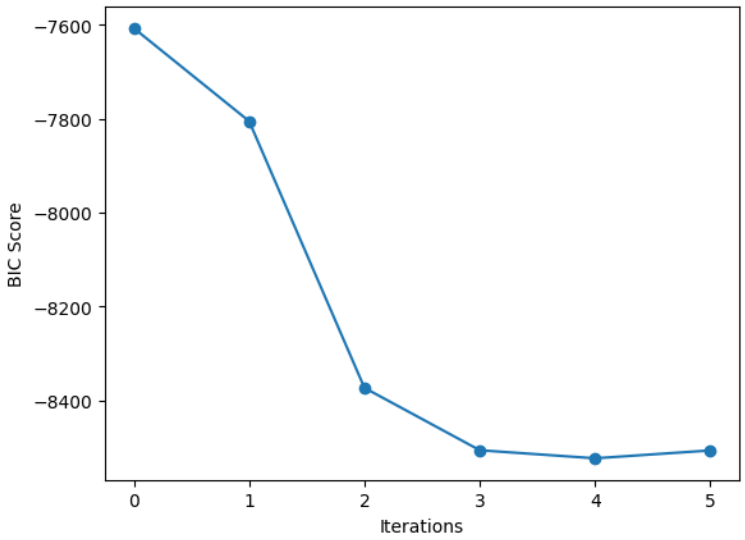

Figure 6.8: Bidirectional elimination, example data set 1.

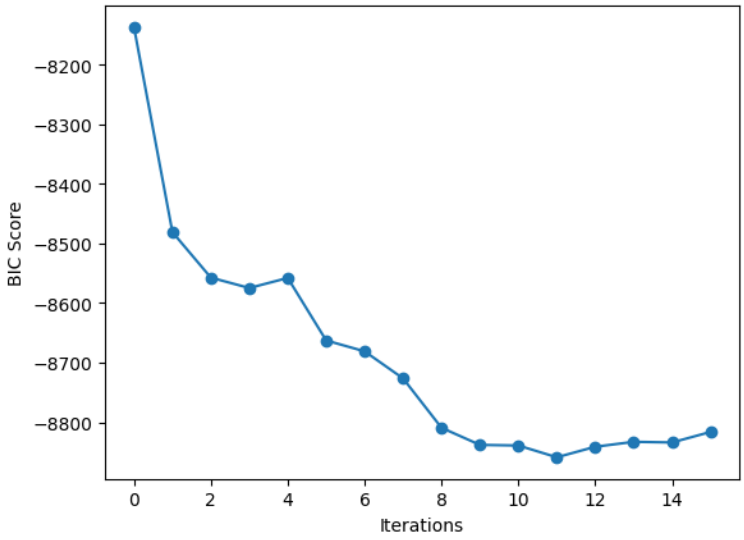

Figure 6.9: Forward selection, example data set 2.

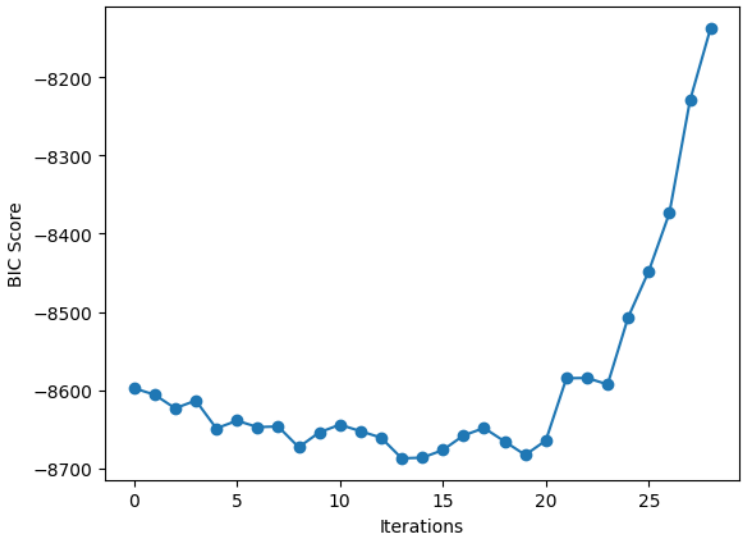

Figure 6.10: Backward selection, example data set 2.

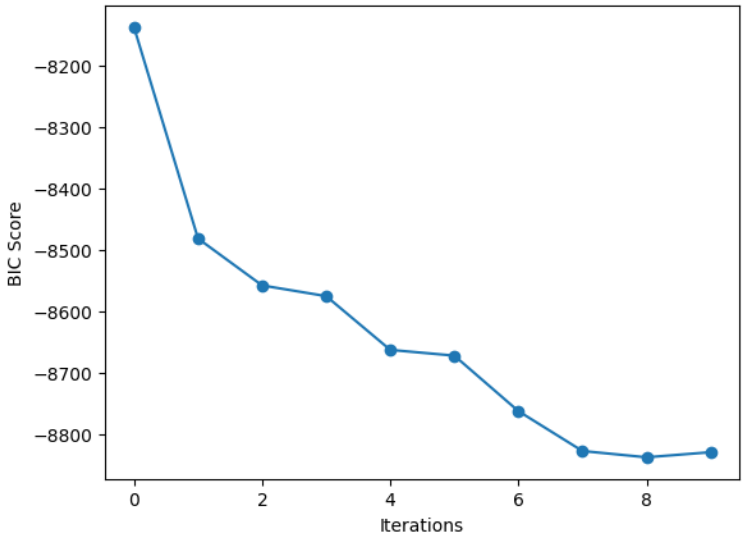

Figure 6.11: Bidirectional elimination, example data set 2.

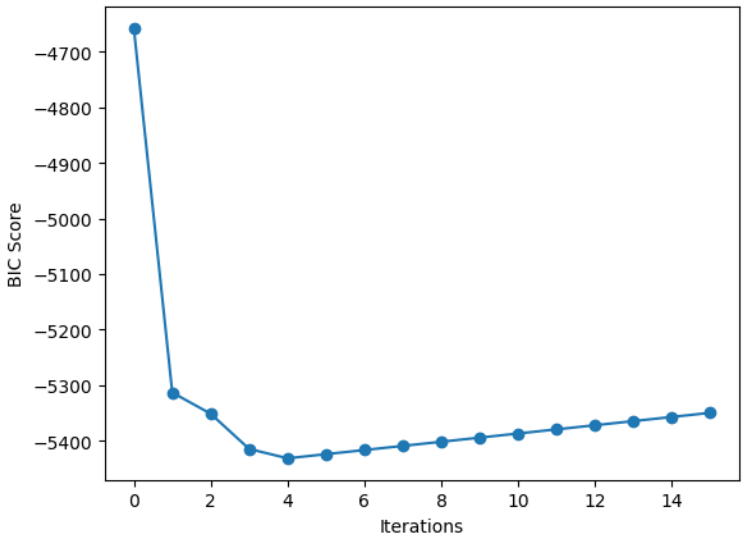

Figure 6.12: Forward selection, example data set 3.

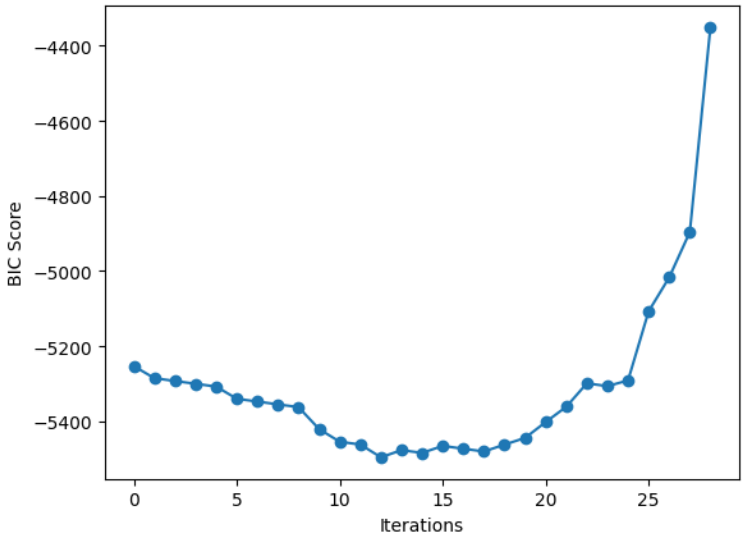

Figure 6.13: Backward selection, example data set 3.

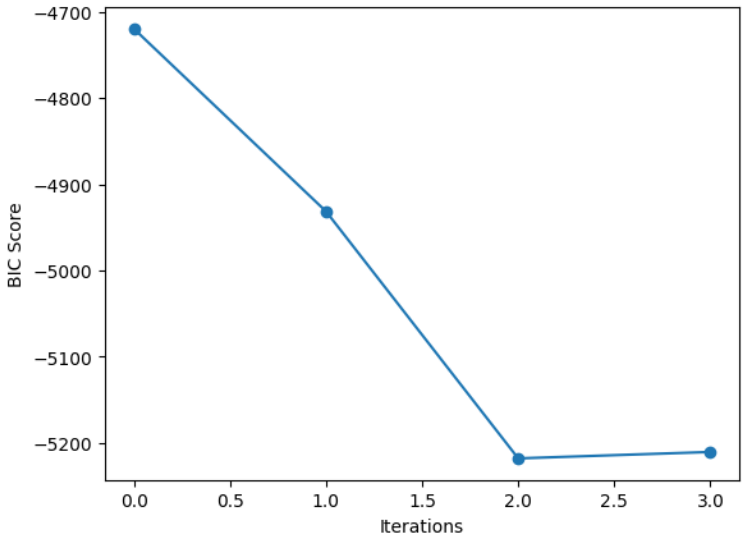

Figure 6.14: Bidirectional elimination, example data set 3.

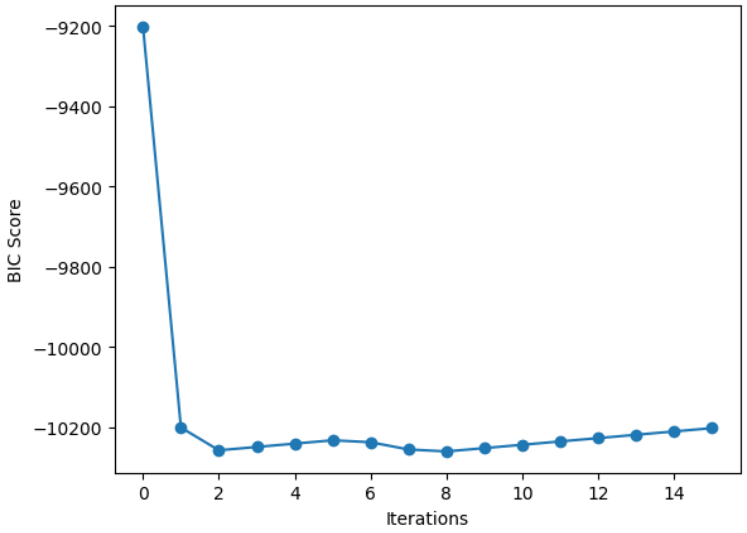

Figure 6.15: Forward selection, example data set 4.

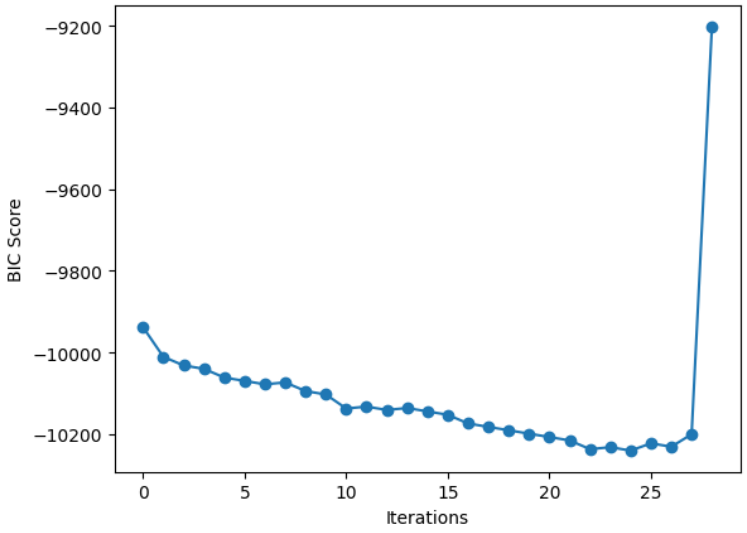

Figure 6.16: Backward selection, example data set 4.

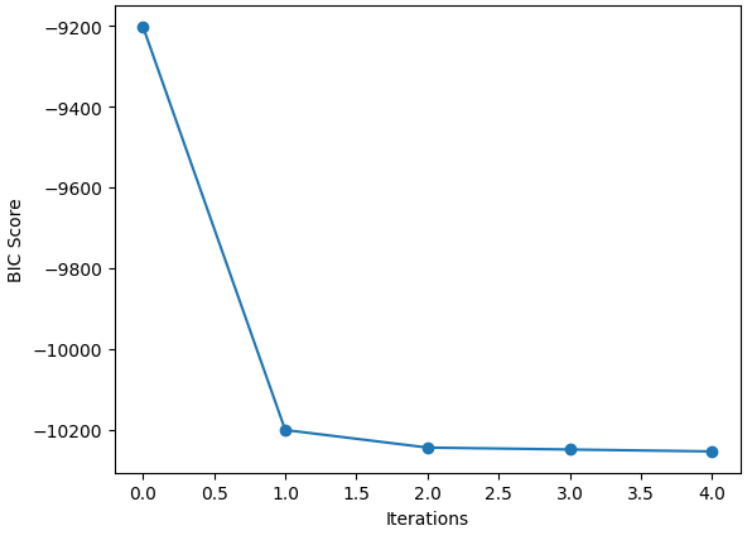

Figure 6.17: Bidirectional elimination, example data set 4.

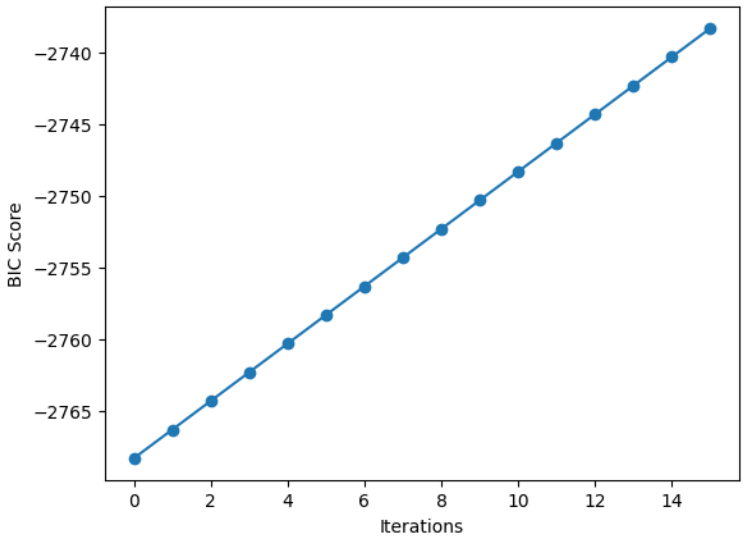

Figure 6.18: Forward selection, example data set 5.

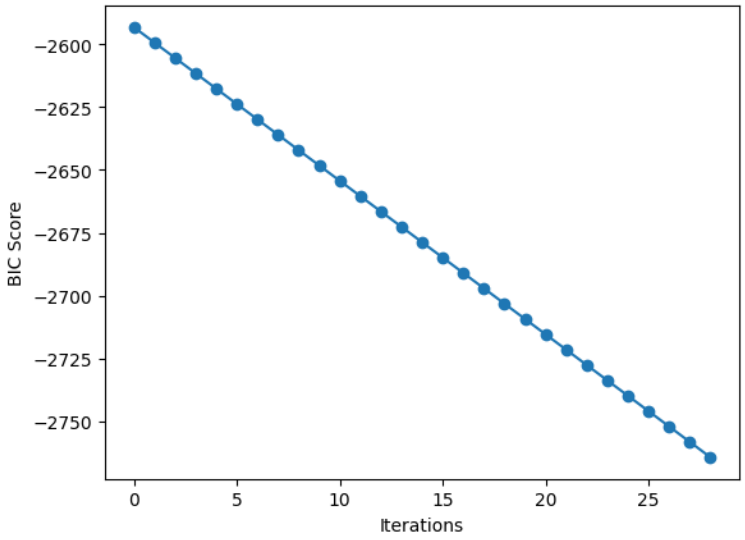

Figure 6.19: Backward selection, example data set 5.

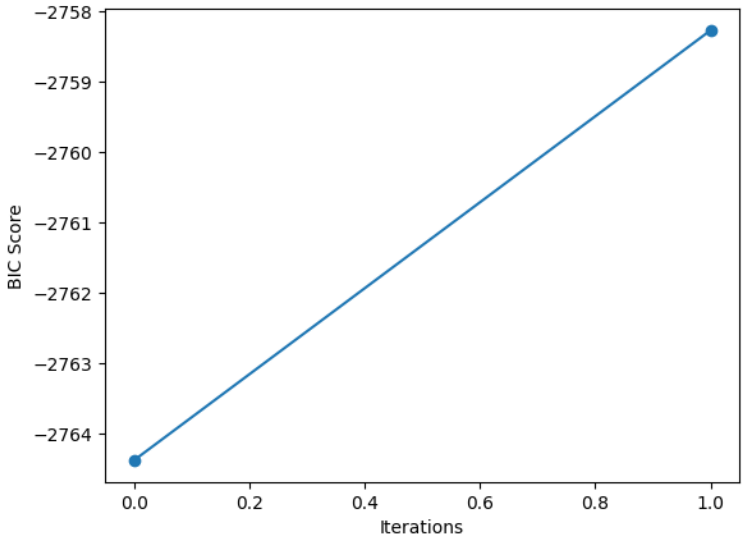

Figure 6.20: Bidirectional elimination, example data set 5.

6.2.2 Group members

In Table 6.7, 6.8 and 6.9 the resulting groups from forward selection, backward selection and bidirectional elimination for the different data sets can be seen respectively. In Table 6.6 a summary of the group size of the groups delivered by each of the feature selection algorithms for each of the data sets can be seen.

Table 6.6: Group size of the groups delivered by each of the feature selection algorithms for each of the data set.

Data set	Forward selection	Backward selection	Bidirectional elimination
1	5	5	5
2	12	16	9
3	5	17	3
4	9	5	5
5	1	1	1

Table 6.7: Forward selection groups for all data sets

Data set	Forward selection group
1	('GLOBAL_DEVICE_ID', 'USER_ID', 'w_alt_n', 'FINGERPRINT_USER_AGENT_OS_VERSION_FULL', 'FINGERPRINT_USER_AGENT_OS_VERSION')
2	('IP_ADDRESS', 'HEADER_USER_AGENT_STRING', 'IP_LATITUDE', 'HEADER_USER_AGENT_SOFTWARE_NAME', 'USER_EMAIL', 'IP_CITY_NAME', 'IP_ISP_ORGANIZATION_NAME', 'ua', 'CLIENT_TOKEN_ID', 'HEADER_USER_AGENT_SOFTWARE_VERSION_FULL', 'IP_SUBDIVISION_CODE', 'FINGERPRINT_USER_AGENT_PLATFORM_BRAND')
3	('IP_ADDRESS', 'eod', 'eol', 'IP_ISP_ORGANIZATION_NAME', 'ku-kd_overlap_b')
4	('CLIENT_TOKEN_ID', 'mc_r_n', 'kd-kd_ln-ll-ti-diff_mad', 'USER_EMAIL_DISPOSABLE', 'USER_ID', 'USER_EMAIL', 'IP_TAGS', 'GLOBAL_DEVICE_ID', 'eol')
5	('tn')

Table 6.8: Backward selection groups for all data sets

Data set	Backward selection group
1	('USER_ID', 'GLOBAL_DEVICE_ID', 'IP_CITY_NAME', 'HEADER_USER_AGENT_STRING', 'npc')
2	('IP_ADDRESS', 'MOBILE_CARRIER_NAME', 'IP_POSTAL_CODE', 'IP_COUNTRY_CODE', 'IP_SUBDIVISION_CODE', 'USER_EMAIL', 'IP_LATITUDE', 'IP_CITY_NAME', 'FINGERPRINT_USER_AGENT_SOFTWARE_VERSION_FULL', 'HEADER_USER_AGENT_OS_VERSION_FULL', 'IP_TIME_ZONE', 'USER_ID', 'CLIENT_TOKEN_ID', 'IP_ISP_ORGANIZATION_NAME', 'dbcs', 'IP_CONNECTION_TYPE')
3	('eol', 'IP_ADDRESS', 'cp', 'mm_start-upper-left_b', 'IP_ISP_NAME', 'IP_LONGITUDE', 'GLOBAL_DEVICE_ID', 'e-e_ti_med', 'IP_ACCURACY_RADIUS', 'kd-kd_ti_med', 'ku_ti_avg', 'pc', 'kd-kd_ln-ll-ti-diff_mad', 'eod', 'cg', 'CLIENT_TOKEN_ID', 'cr')
4	('FINGERPRINT_USER_AGENT_SOFTWARE_VERSION_FULL', 'USER_EMAIL', 'IP_ISP_AUTONOMOUS_SYSTEM_NUMBER', 'mc_r_n', 'e-e_ti_med')
5	('FINGERPRINT_USER_AGENT_SOFTWARE_VERSION_FULL')

Table 6.9: Bidirectional elimination groups for all data sets

Data set	Bidirectional elimination
1	('USER_ID', 'GLOBAL_DEVICE_ID','eum', 'IP_TIME_ZONE', 'Content-Type')
2	('IP_ADDRESS', 'HEADER_USER_AGENT_STRING', 'IP_LATITUDE', 'USER_ID', 'IP_CITY_NAME', 'ua', 'IP_ISP_AUTONOMOUS_SYSTEM_NUMBER', 'USER_EMAIL', 'dosn')
3	('IP_ADDRESS', 'mc_c_b', 'CLIENT_TOKEN_ID')
4	('USER_EMAIL', 'mc_r_n', 'kd-kd_ln-ll-ti-diff_mad', 'Dnt', 'IP_TIME_ZONE')
5	('GLOBAL_DEVICE_ID')

6.3 Model prediction/information loss

For each data set a simple LRG-model predicts whether a data point from the test data set is in a spike or not, while only accessing the features specified in that case. These cases are:

- All features available
- The top 50 features in feature importance
- The forward selection group
- The backward selection group
- The bidirectional elimination group

A summary result Table with macro average F1 scores for all data sets and selection algorithms is presented in Table 6.10. The complete results are presented in classification reports in Tables 6.11 - 6.35.

Table 6.10 shows that for each data set and every selection algorithm, a high macro average F1-score indicates that all those feature groups performed very well in classifying data points as spikes or not. For the data sets where the baseline model performs better the models using the selection algorithms also usually perform better. Tables 6.11 - 6.35 paint the full picture of these results, also showing that the performance was good for both classes, not just one or the other. All classification reports also show an even performance in both precision and recall for all data sets and selection algorithms, meaning that classifying both spikes and non-spikes accurately was accomplished in these tests.

Table 6.10: Macro average F1 scores for all data sets and selection algorithms.

	Macro avg F1-score				
Data set	Baseline	Top 50	Forward	Backward	Bidirectional
1	0.87	0.87	0.87	0.90	0.91
2	0.89	0.89	0.88	0.88	0.88
3	0.95	0.95	0.95	0.96	0.95
4	0.92	0.92	0.92	0.92	0.92
5	1.00	1.00	1.00	1.00	1.00

Table 6.11: Classification report, example data set 1, all features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.87	0.87	0.87	2013
1	0.87	0.87	0.87	1987
Accuracy			0.87	4000
Macro avg	0.87	0.87	0.87	4000
Weighted avg	0.87	0.87	0.87	4000

Table 6.12: Classification report, example data set 1, top 50 features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.86	0.89	0.87	1974
1	0.89	0.86	0.87	2026
Accuracy			0.87	4000
Macro avg	0.87	0.87	0.87	4000
Weighted avg	0.87	0.87	0.87	4000

Table 6.13: Classification report, example data set 1, forward selection group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.85	0.90	0.87	1974
1	0.90	0.85	0.87	2026
Accuracy			0.87	4000
Macro avg	0.87	0.87	0.87	4000
Weighted avg	0.87	0.87	0.87	4000

Table 6.14: Classification report, example data set 1, backward selection group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.86	0.91	0.89	1974
1	0.91	0.85	0.88	2026
Accuracy			0.88	4000
Macro avg	0.88	0.88	0.88	4000
Weighted avg	0.88	0.88	0.88	4000

Table 6.15: Classification report, example data set 1, bidirectional elimination group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.86	0.90	0.88	1974
1	0.90	0.86	0.88	2026
Accuracy			0.88	4000
Macro avg	0.88	0.88	0.88	4000
Weighted avg	0.88	0.88	0.88	4000

Table 6.16: Classification report, example data set 2, all features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.90	0.89	0.90	2006
1	0.89	0.90	0.89	1994
Accuracy			0.90	4000
Macro avg	0.90	0.90	0.89	4000
Weighted avg	0.90	0.90	0.90	4000

Table 6.17: Classification report, example data set 2, top 50 features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.90	0.89	0.89	2006
1	0.89	0.90	0.89	1994
Accuracy			0.89	4000
Macro avg	0.89	0.89	0.89	4000
Weighted avg	0.89	0.89	0.89	4000

Table 6.18: Classification report, example data set 2, forward selection group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.90	0.85	0.88	2006
1	0.86	0.91	0.88	1994
Accuracy			0.88	4000
Macro avg	0.88	0.88	0.88	4000
Weighted avg	0.88	0.88	0.88	4000

Table 6.19: Classification report, example data set 2, backward selection group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.90	0.85	0.88	2006
1	0.86	0.91	0.88	1994
Accuracy			0.88	4000
Macro avg	0.88	0.88	0.88	4000
Weighted avg	0.88	0.88	0.88	4000

Table 6.20: Classification report, example data set 2, bidirectional elimination group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.90	0.86	0.88	2006
1	0.87	0.90	0.88	1994
Accuracy			0.88	4000
Macro avg	0.88	0.88	0.88	4000
Weighted avg	0.88	0.88	0.88	4000

Table 6.21: Classification report, example data set 3, all features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.92	0.98	0.95	884
1	0.98	0.91	0.94	827
Accuracy			0.95	1711
Macro avg	0.95	0.95	0.95	1711
Weighted avg	0.95	0.95	0.95	1711

Table 6.22: Classification report, example data set 3, top 50 features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.92	0.98	0.95	884
1	0.98	0.91	0.94	827
Accuracy			0.95	1711
Macro avg	0.95	0.95	0.95	1711
Weighted avg	0.95	0.95	0.95	1711

Table 6.23: Classification report, example data set 3, forward selection group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.92	0.99	0.95	884
1	0.99	0.90	0.94	827
Accuracy			0.95	1711
Macro avg	0.95	0.95	0.95	1711
Weighted avg	0.95	0.95	0.95	1711

Table 6.24: Classification report, example data set 3, backward selection group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.94	0.99	0.96	884
1	0.99	0.93	0.96	827
Accuracy			0.96	1711
Macro avg	0.97	0.96	0.96	1711
Weighted avg	0.96	0.96	0.96	1711

Table 6.25: Classification report, example data set 3, bidirectional elimination group features, predicted with a logistic regression model.

	Precision	Recall	F1-score	Support
0	0.92	0.99	0.96	884
1	0.99	0.91	0.91	827
Accuracy			0.95	1711
Macro avg	0.96	0.95	0.95	1711
Weighted avg	0.96	0.95	0.95	1711

Table 6.26: Classification report, example data set 4, all features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.87	0.99	0.93	1990
1	0.99	0.85	0.92	2010
Accuracy			0.92	4000
Macro avg	0.93	0.92	0.92	4000
Weighted avg	0.93	0.92	0.92	4000

Table 6.27: Classification report, example data set 4, top 50 features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.87	0.99	0.93	1990
1	0.99	0.85	0.92	2010
Accuracy			0.92	4000
Macro avg	0.93	0.92	0.92	4000
Weighted avg	0.93	0.92	0.92	4000

Table 6.28: Classification report, example data set 4, forward selection group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.87	0.99	0.93	1990
1	0.99	0.85	0.92	2010
Accuracy			0.92	4000
Macro avg	0.93	0.92	0.92	4000
Weighted avg	0.93	0.92	0.92	4000

Table 6.29: Classification report, example data set 4, backward selection group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.87	0.98	0.92	1990
1	0.98	0.86	0.91	2010
Accuracy			0.92	4000
Macro avg	0.92	0.92	0.92	4000
Weighted avg	0.92	0.92	0.92	4000

Table 6.30: Classification report, example data set 4, bidirectional elimination group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	0.87	1.00	0.93	1990
1	1.00	0.85	0.92	2010
Accuracy			0.92	4000
Macro avg	0.93	0.92	0.92	4000
Weighted avg	0.93	0.92	0.92	4000

Table 6.31: Classification report, example data set 5, all features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	233
1	1.00	1.00	1.00	220
Accuracy			1.00	453
Macro avg	1.00	1.00	1.00	453
Weighted avg	1.00	1.00	1.00	453

Table 6.32: Classification report, example data set 5, top 50 features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	233
1	1.00	1.00	1.00	220
Accuracy			1.00	453
Macro avg	1.00	1.00	1.00	453
Weighted avg	1.00	1.00	1.00	453

Table 6.33: Classification report, example data set 5, forward selection group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	233
1	1.00	1.00	1.00	220
Accuracy			1.00	453
Macro avg	1.00	1.00	1.00	453
Weighted avg	1.00	1.00	1.00	453

Table 6.34: Classification report, example data set 5, backward selection group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	233
1	1.00	1.00	1.00	220
Accuracy			1.00	453
Macro avg	1.00	1.00	1.00	453
Weighted avg	1.00	1.00	1.00	453

Table 6.35: Classification report, example data set 5, bidirectional elimination group features, predicted with LRG.

	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	233
1	1.00	1.00	1.00	220
Accuracy			1.00	453
Macro avg	1.00	1.00	1.00	453
Weighted avg	1.00	1.00	1.00	453

Chapter 7
Discussion

This chapter will discuss the results from Chapter 6. It will examine the target labelling accuracy, the feature selection algorithms, the group members delivered, the model predictions, the performance of the pipeline, and the effect of the target proxy.

7.1 Target labelling

This section will discuss the results from the target labelling procedure. It will cover the results of the signal processing and evaluate to which extent the detected spikes corresponded to what Castle labelled as scripted attack data points.

As can be seen in Figures 6.1 - 6.5, fairly simple solutions with a few manually set parameters could detect spikes in the data sets. Upon visual examination, all spikes detected in the data sets look as expected, the red-marked parts of the data set plots line up with the abnormal parts of the data traffic. The one data set where it is not as crystal clear is data set 2, seen in Figure 6.2. Here the spikes are very narrow, corresponding to short time spans, and some detected spikes, more precisely the third, fifth and sixth, do not seem to be as abnormal with regards to the normal activity level as the other spikes are.

The accuracy in spike detection, summarised in Table 6.1, reveals some interesting insights. For data sets like 3 and 5, the target labelling procedure managed to very accurately classify data points as malign or not. However, for data set 2, the labelling was not very accurate, with a macro average F1-score of only 0.54. When inspecting Figure 6.2, it looks like the spikes are very properly marked, but the classification report in Table 6.3 reveals that what is actually found consists of mostly benign data points. The F1-score for class 1, which is what is labelled as malign data points, is only 0.07. The reason why this is the case could be a number of things. Most likely, the malign data points do not occur in an extreme amount in short time spans, but are rather part of the normal-looking ambient data. This could point to a more sophisticated strategy on behalf of the attacker party. Spreading the scripted attempts at exploiting vulnerabilities over time is a clever way of hiding the activity from appearing suspicious upon manual inspection.

For data sets 3 and 5, the target labelling was far more successful. Both resulted in macro average F1-scores of 0.94, with good F1-scores for both classes. In particular, the recall values for class 1 were particularly good, with 0.98 for data set 3 and 1.00 for data set 5. This suggests that the correct time spans for the attacks were found, but contained some benign data points as well. This is to be expected with most web traffic data with a constant number of events per minute, since regular users also use the website in question even when an attack is happening.

7.2 Relevant feature retrieval

This section will discuss the results from the process of feature selection and the groups of features retrieved. It will evaluate the relevancy of those features compared to the features that Castle deemed indicative of scripted attacks.

7.2.1 Feature selection

The plots for data set 1 shown in Figures 6.6 - 6.8 look fairly optimal, forward and backward selection looking like check marks imply that the different models that are compared perform roughly similar but smaller group sizes score better.

The results for data set 2 are indicative of the algorithms struggling to find a small number of indicative features, likely stemming from the fact that the target labelling for this data set performed poorly. With poor targets, the algorithms will struggle to find features that can indicate malign data points simply because there are no clear patterns to be found.

For data set 3, forward selection finds a minimum corresponding to a much smaller group size than backward selection. This is not entirely expected, if a small group size is the most indicative of a scripted attack, backward selection would find it after many more iterations. A probable cause for this is the greedy nature of the algorithm, an early choice of a feature to remove could end up being sub-optimal in the global context. Bidirectional elimination finds a small group of size 3 after three iterations, which is more in line with the results of forward selection, and further strengthens the hypothesis that backward selection did not give a similar result to the other algorithms for this data set because of its greedy nature.

For data set 4, the result follows a similar pattern to the data sets that generated small group sizes. This result points to the fact that a very limited number of features in the groups can indicate a scripted attack very well.

For data set 5, the results are of a completely different character. All of the Figures 6.18 - 6.20 show a completely linear relationship between iterations and BIC score for both forward and backward selection, as well as for bidirectional

elimination, although bidirectional finds a local minimum after only one iteration. This leads to group sizes of 1 for all these algorithms. A linear BIC plot is not typically what is expected, but the explanation can be found by looking at the classification reports in Tables 6.31 - 6.35. These show that no matter the features used by the LRG-model to predict malign data points, the model can classify them with 100% accuracy. This means that all the model variations tested in the selection algorithms will have the same score in every term but the penalty term related to model complexity. This term is linear with respect to the number of parameters in the model, which corresponds to the number of features used to train it, which is why the BIC score increases or decreases linearly by adding or removing features.

It is worth noting that while both data sets 1 and 2 have poor target labelling, they differ in the respect that data set 1 gets much smaller group sizes in the feature selection stage. The reason is not clear, but there can be a possible explanation in the classification report shown in Tables 6.2 and 6.3. The difference between these two is that while they both have a macro average of just over 0.5, they have strengths in completely opposite parts of the data. For data set 1, the target labelling is very successful for class 1, which is the malign data points, while being extremely poor for class 0. For data set 2, the exact opposite is true. Much of this is because of the distribution between the two classes, for both data sets the successful class was the majority class of the data set. This might suggest that as long as the malign class is well represented, the feature selection algorithms will find small groups that encompass the data set characteristics well, especially the malign data points.

7.2.2 Group members

For data set 1, the groups were fairly small and the two reoccurring features from all selection algorithms were `'USER_ID'` and `'GLOBAL_DEVICE_ID'`. This aligns somewhat well with the indicative feature from Castle. While not matching exactly, much of the same information in the indicative feature from Castle; `'CLIENT_TOKEN_ID'`, is shared with these two features, they are essentially related in many ways and also come from the same origin, namely user-related features. The groups are therefore to be considered relevant for this data set.

For data set 2, the groups delivered by the selection algorithms were larger than most other data sets. All selection algorithms delivered groups consisting of many IP related features, such as `'IP_ADDRESS'` and `'IP_LATITUDE'`, among others. While the groups were large and the accuracy of target labels for this data set was poor, IP related features are still relevant in indicating malign data points, as can be seen in Table 5.2. For example, when manually inspecting the data set in the column related to the feature `'IP_LATITUDE'`, the by far most common value is 26.9, which happens to be the latitude of the city Lucknow, the feature value that Castle classified as indicative of scripted attacks. On the one hand, it is positive that the indicative feature is present in all of the groups, on the other hand the groups contain many more features than that, which likely means that the relationships between the groups and malign data points are not that strong.

For data set 3, the indicative feature was `'mm_start-upper-left_b'`. This feature is found by the backward selection algorithm but not the other two selection algorithms. It is worth noting that this feature was found by backward selection specifically, which was the outlier in terms of group size for this data set. The reason for this result could be that the smaller groups found by the other two selection algorithms were actually better in indicating malign data points when used together in a group context, but when such a group was not found by backward selection, `'mm_start-upper-left_b'` was contained in the group simply because it contained much information on its own.

For data set 4, there was no provided indicative feature, so this segment can not be evaluated for this data set specifically. It is worth noting that the group sizes were small, in the same order of magnitude as for the data sets with good target labelling. This could mean that the groups for this data set were relevant and indicative of malign data points.

For data set 5, as discussed earlier, the groups from all selection algorithms consisted of single features. The nature of the selection algorithms was such that these did not necessarily had to be neither the highest ranking in feature importance, nor the same between the algorithms. The indicative feature from Castle was not present in any of the groups for this data set, but that does not mean the features in the group were not relevant, they were simply not the one that Castle explicitly provided.

7.3 Model prediction/information loss

This section will cover the results of the model predictions made using LRG-models trained with different subsets of the training data. It will discuss whether small groups of features can retain enough critical information about the data set to accurately classify data points. Table 6.10 outlines the macro average F1-scores for all different test cases described in 5.5.3. From this table, it is apparent that all methods of feature grouping managed to classify data points with as much, and sometimes even higher accuracy than when using all available features in the data sets. The only data set for which the F1-score dropped slightly was data set 2, for the other the scores were as good or better than the baseline when using the feature groups delivered by the selection algorithms.

It is no surprise that data sets 3, 5 and to some extent 4, can be reduced to a few important features and still classify data points accurately. The target labelling procedures for these data sets were promising, and the group sizes from the selection algorithms were small, hinting at the fact that the groups were both relevant and all-encompassing. The more uncertain results are the ones for data sets 1 and 2. One would assume that since the pipelines for them both had a poor target labelling accuracy, their capabilities in classifying data points would be bad, but surprisingly enough they were on par with the pipelines of the other data sets.

The most likely cause for this is that for these data sets the indicative feature provided by Castle were likely good at indicating a malign data point based on their manual analysis, but rather poor at indicating spikes in the data set. The two don't necessarily have to be the same, but one of the main assumptions in this work was that spikes and malign data points coincided, which it turns out is not always the case.

7.4 Performance

This section will discuss how the pipeline developed in this work compares to an exhaustive test of all possible groups with respect to the number of models needed to be trained. The feature selection algorithms successfully reduce the complexity of the grouping problem to be having to train only $\sum_{r=1}^{k}(n-(r-1)) = \frac{1}{2}k(2n-k+1)$ different models rather than $\sum_{r=1}^{k}\binom{n}{r}$ for an exhaustive test of all groups, where n is the number of features and k is the maximum group size. The first expression is true for the forward selection algorithm and very similar for the other feature selection algorithms. When plugging in $n = 200$ and $k = 30$, approximately the numbers used in this work, the forward selection needs 5565 models. When doing the same for the exhaustive test it needs around 4.96×10^{35} models. This is an enormous improvement and a necessary step to be able to run the pipeline. However, this comes at a cost. These algorithms are greedy, and they cannot guarantee that they will find a globally optimal solution since they only optimise locally in each iteration. The feature importance methods further decrease the number of models needed to train. The feature selection algorithms can be run with the 50 features with the highest feature importance scores. With $n = 50$ and $k = 30$ the number of models needed is only 1065 which is manageable on a modern laptop.

7.5 Target proxy

What is apparent from the previous sections is that the target labelling is a larger part of this process than first anticipated. It was clear that the pipeline performed at a high level in classifying the target that was selected as a proxy, in this case spikes as a proxy for scripted attacks. However, in some cases the target labelling DSP used in this work managed to accurately find activity spikes in the data but failed to properly detect malign data points, simply because they did not coincide with activity spikes. This means that while the pipeline performed in a satisfactory manner, it can only be as good as the target proxy. If the proxy is a good indicator for the actual target, the pipeline will be good at classifying data points as that target or not.

It is also worth discussing the generality of this work, especially within the context of target proxies. The target proxy was chosen as the activity level binarised into two classes 0 and 1 representing whether it was a spike or not. However, the proxy could be something completely different as long as it is intrinsic to the data. For example, if the goal when using this pipeline was to find groups of features that

could indicate whether a data point came from a specific data centre, this would be possible as well. This means that the applications of this work are broader than originally anticipated, and the potential applications in future work are certainly interesting.

Chapter 8
Conclusions and Future Work

In this chapter, the conclusions that can be drawn from this work will be discussed and the main research questions from Chapter 1 will be answered. Furthermore, possible future work in this area will be investigated, including target labelling improvements and handling of correlated features and natural spikes.

8.1 Conclusions

This section will discuss what conclusions that can be drawn. The goal of this work was to answer the three main questions in Section 1.2.1. This was attempted by developing a pipeline that from a data set of web traffic consisting of up to around 200 different features could identify small groups of these features that contain information about whether or not a data point in the data set of web traffic originated from a scripted attack or a real user. The three main questions will be discussed individually below.

Q1: Can a machine learning pipeline dynamically find groups of important features in collected web traffic data in order for existing systems to classify fraudulent activity?

It is possible to use a pipeline of machine learning tools to dynamically find groups of important features that can indicate fraudulent activity, but this pipeline is subject to many improvements and is also likely just a part of the tool chain needed to effectively perform this complex task. This work demonstrated that this kind of pipeline can find groups of features that can classify data points given a proxy target with high certainty.

Q2: Can such a pipeline perform better than a simple baseline solution in predicting fraudulent activity?

The baseline solution tested against in this work was the non-reductive LRG-model trained on all of the features. As discussed in Section 7.3 the models trained on only the features belonging to the group performed very similarly to the models trained on the entire feature set. However, both of these models are trained and

evaluated on the proxy targets. The predicting capability of the models for classifying scripted traffic therefore is highly dependent on the quality of the proxy targets.

It can not be said that the pipeline outperformed the baseline solution with regard to the accuracy of the model. However, the final model presented by the pipeline performs similarly and has other important properties, such as a smaller feature set and thereby a simpler model.

Q3: Can such a pipeline improve interpretability and facilitate decision making?

It is possible that this work can be a valuable addition to the existing solutions at Castle and aid their customers in making decisions about their web services. This work presented a data-driven pipeline that used interpretable machine learning tools and strategies so that every step from data parsing to delivery of groups could be backtracked and explained. This can potentially be a useful complement to the existing risk scoring and other services that Castle provides to their customers.

8.2 Future work

This section will bring up potential future work in this area. The importance of target labelling and how it possibly could be improved will be discussed, as well as how consideration of correlated features could improve the interpretability. The phenomena of natural spikes is also discussed.

8.2.1 Target labelling

The approaches for target labelling in this work were fairly simple as a product of the scope and limitations. It is however clear that if this work is to be further developed, a high emphasis needs to be put on target labelling, if the supervised learning approach is used. It is possible that there are better proxies than spikes in activity, or that an approach not using proxies at all can perform better. All these subjects are interesting and relevant to further research on this topic.

An example of an interesting scenario in which this work could be further developed is detecting spoofed activity. Spoofing is the act of intentionally obfuscating or changing information about who and from where a person is interacting with a web service. It can be done with a number of parts of the information of who one acts out to be, a typical example is IP-address spoofing. If lots of IP-addresses in the web traffic come from the same approximate IP region which happens to be a data centre, it is easy for a web service provider to shut down such activity. This is why such actors can try to spoof their IP information to circumvent such checks. With this pipeline and access to none-spoofed data, groups of features that can indicate this kind of activity can be found, so that when spoofing happens at a later stage, it can be detected even if the IP information is spoofed at that point.

8.2.2 Correlated features

A way to further build on the pipeline developed would be to add an extra step at the end that would look for correlated features. This might be done by looking at the groups delivered by the feature selection algorithms and calculating a correlation metric between the features belonging to the group and the remaining features. Finding features in the group with a high correlation to other features could further improve interpretability by creating more groups containing similar information but maybe meaning something else to the end user looking at the groups.

8.2.3 Natural spikes

Under some special circumstances, natural spikes in activity can occur. This can be because of some event, like the Black Friday sale or the release of a new video game for example. In this case, a spike in activity will work bad as a proxy for scripted attacks because the spikes will likely consist of benign activity. This would present a particular challenge in finding a proper target proxy under these circumstances. The occurrence of such events is low according to Castle based on their customer base, but this would be an interesting research topic nonetheless. Unfortunately, no such data was available from Castle at the time of the publication of this thesis.

References

[1] Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control 19*, 6 (1974), 716–723.

[2] Altmann, A., Toloşi, L., Sander, O., and Lengauer, T. Permutation importance: a corrected feature importance measure. *Bioinformatics 26*, 10 (04 2010), 1340–1347.

[3] Antonio, E., Fajardo, A. C., and Medina, R. P. Browser fingerprint standardization using rule- based algorithm and multi-class entropy. In *International Journal of Scientific and Technology Research* (2020), vol. 9.

[4] Brownlee, J. Logistic regression for machine learning, Accessed 2024-04-15. `https://machinelearningmastery.com/logistic-regression-for-machine-learning/`.

[5] Brownlee, J. A gentle introduction to cross-entropy for machine learning, Accessed 2024-05-02. `https://machinelearningmastery.com/cross-entropy-for-machine-learning/`.

[6] Cao, Y., Li, S., and Wijmans, E. Cross-browser fingerprinting via os and hardware level features. In *Network and Distributed System Security Symposium* (2017).

[7] Cludflare. What is a bot attack?, Accessed 2024-04-09. `https://www.cloudflare.com/learning/bots/what-is-a-bot-attack/`.

[8] Dataquest. Regularization in machine learning, 2022-11-11, Accessed 2024-04-23. `https://www.dataquest.io/blog/regularization-in-machine-learning/`.

[9] Efroymson, M. A. Multiple regression. In *Mathematical Methods for Digital Computers (Anthony Ralston and Herbert S. Wilf (eds.)*. Wiley, New York, 1960, pp. 191–204.

[10] Friedman, J. H. Stochastic gradient boosting. *Computational Statistics Data Analysis 38*, 4 (2002), 367–378. Nonlinear Methods and Data Mining.

[11] Hidalgo, J. M. G., and Alvarez, G. Chapter 3 - captchas: An artificial intelligence application to web security. vol. 83 of *Advances in Computers*. Elsevier, 2011, pp. 109–181.

[12] IBM. What is a decision tree?, Accessed 2024-04-16. https://www.ibm.com/topics/decision-trees.

[13] IBM. What is supervised learning?, Accessed 2024-04-30. https://www.ibm.com/topics/supervised-learning.

[14] Keita, Z. Classification in machine learning: An introduction, 2022-09, Accessed 2024-04-24. https://www.datacamp.com/blog/classification-machine-learning, urldate = 2024-04-24.

[15] Lundberg, S. M., and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774.

[16] Naseer, S., Saleem, Y., Khalid, S., Bashir, M. K., Han, J., Iqbal, M. M., and Han, K. Enhanced network anomaly detection based on deep neural networks. *IEEE Access 6* (2018).

[17] Novack, G. Building a one hot encoding layer with tensorflow, 2020-06-07, Accessed 2024-05-16. https://towardsdatascience.com/building-a-one-hot-encoding-layer-with-tensorflow-f907d686bf39.

[18] pandas development team, T. pandas-dev/pandas: Pandas, 2024.

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

[20] Powers, D. M. W. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *CoRR abs/2010.16061* (2020).

[21] Rengasamy, D., Rothwell, B. C., and Figueredo, G. P. Towards a more reliable interpretation of machine learning outputs for safety-critical systems using feature importance fusion. *Applied Sciences 11*, 24 (2021).

[22] Singh, S. Understanding the bias-variance tradeoff, 2018-05-21, Accessed 2024-05-16. https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229.

[23] Theodoridis, S., and Koutroumbas, K. In *Pattern Recognition (Third Edition)*, S. Theodoridis and K. Koutroumbas, Eds., third edition ed. Academic Press, San Diego, 2006, pp. 1–11.

[24] Wit, E., Heuvel, E. v. d., and Romeijn, J.-W. 'all models are wrong...': an introduction to model uncertainty. *Statistica Neerlandica 66*, 3 (2012), 217–236.

[25] Xiao, C., Freeman, D. M., and Hwa, T. Detecting clusters of fake accounts in online social networks. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security* (2015), AISec '15, Association for Computing Machinery.