ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

University of Pireaus
Digital Systems Department

Anomaly Detection for Industrial Control Systems

Thesis submitted for the MSc Degree of Security in Digital Systems
May 2018

Author:
Kapogianni Eirini
Supervisors:
Prof. Cristina Alcaraz,
Prof. Constantinos Lamprinoudakis

Acknowledgement

I would first like to express my appreciation and thanks to my thesis advisor Prof. Constantinos Lamprinoudakis whose door office was always open whenever I had a question and he steered me in the right direction about my Master's Thesis topic.

I would also like to thank the expert Prof. Christina Alcaraz, who was involved in the validation survey for this research project. Without her participation, input and helpful comments and advice the validation survey could not have been successfully conducted.

Finally, I would like to thank my family and my friends, Maria and Spyros, for encouraging me and supporting me throughout this experience.

Abstract

As Industrial Control Systems (ICSs) become more and more connected it follows that they need to become more secure. Traditional Intrusion Detection Systems (IDSs) do not work well due to the fact that they mostly work on a signature basis and there are not many known signatures to detect attacks on ICSs. Since the network traffic from an ICS is claimed to be static and signatures are scarce, searching for anomalies in the network to detect threats is more effective. This can be achieved using machine learning and other statistical models, teaching the system to tell regular traffic from irregularities. In this thesis we survey different anomaly detection techniques, which based on different parameters, we evaluate and point the one that can fit better. Based on the survey and the risk analysis we analyze the algorithm on which we conclude, and with real-time data-sets (normal and anomalous) we do an implementation. From this work we propose and evaluate methods to be used when creating a more data driven IDS, capable of detecting process semantic tampering within an ICS. Our results from conducted experiments exhibit a static nature of the data originating from the ICS and the result from evaluating many different proposed anomaly detections using proof of concept systems, we deem that the anomaly detection and algorithm that we conclude works well for both semantic tampering as well as on a network basis. Having an IDS using a fusion this proposed method, would benefit the security of an ICS.

Contents

Figures

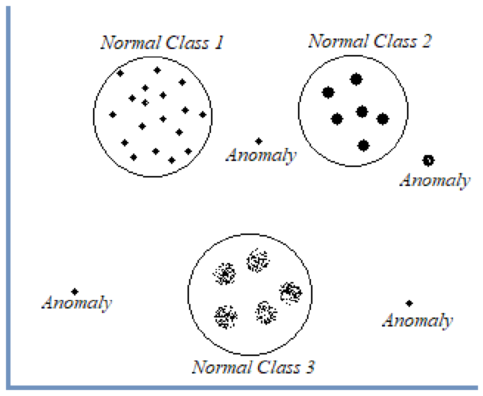
Chapter 1

Machine Learning based Anomaly Detection

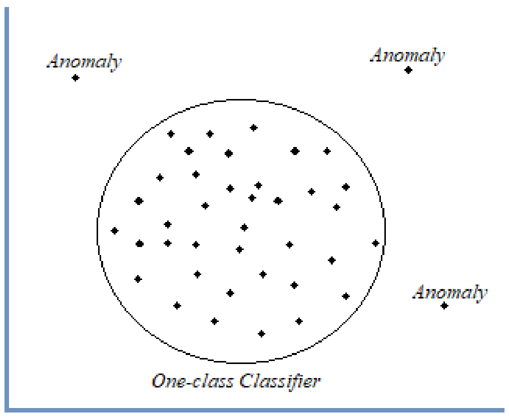
Machine learning is the ability of a program or a system to learn and improve their performance on a certain task or group of tasks over time. Machine learning aims to answer many of the same questions as statistics or data mining. However, unlike statistical approaches which tend to focus on understanding the process that generated the data, machine learning techniques focus on building a system that improves its performance based on previous results. [33]

1.1 Introduction

A model (classifier) is able to be trained from a set of labeled data instances, make classes and then classify a test instance into the mentioned classes. In other words training and then testing. This is classification. As mentioned, classification consists of two steps. The first one is the training phase that the labeled training data can be used by the classifier in order to make a model for standardization. The second step is testing. Through this operation, a test instance can be defined as normal or anomalous. Because there is a variety in the labeled data, the researchers in order to facilitate the procedure have grouped the detection techniques into two different categories. The first one is defined as multi-class anomaly detection techniques. This technique refers to multiple normal classes and teach the classifier to work separately between each normal class and the other classes. According to the result, a test instance will be defined as anomalous or normal. One the other hand, we have one-class anomaly detection technique. One-class classification works as all the training instances belong to one and only class label. They define a boundary around the normal instances so that a classification algorithm will learn and depending on that boundaries a test instance can be declared as anomalous or not.



(1.1) Multi-class Anomaly Detection



(1.2) One-class Anomaly Detection

Overfitting is the phenomenon in which the learning system tightly fits the given training data. This can has as a result the inaccuracy in predicting the outcomes of the training data. Overfitting is more likely with non-parametric and nonlinear models that have more flexibility when learning a target function. As such, many non-parametric machine learning algorithms also include parameters or techniques to limit and constrain how much detail the model learns.

1.1.1 Decision Trees

A Decision Tree is a tree-like graph that can be a very effective method of classification in anomaly detection. It consists of internal nodes which represent a test. More precisely, the input is in general a set of classified data, the output is a tree where every node (leaf) is a decision (class) and each non-final (internal) represents a test. The aim of decision trees is to categorize partitions of data-sets into groups as homogeneous as possible, so that the next variable (or newly coming data) can be predicted. The classification rules are formed by the path selected from the root node to the leaf. To divide each input data, the first root node should be chosen, as it is the most important attribute to separate the data. As a consequence, we can recognize that the main issue is: which value should be chosen for splitting the node of the tree. Starting from a root node to the leaf node visits all the internal nodes in the path depending on the test conditions that are predefined or the attributes of each node.

In general, decision trees are able to analyze data and identify significant characteristics in networks that indicate malicious activities. An advantage is that it can add value to many real-time security systems by analyzing a large data-set of intrusion detection. Besides, it can recognize patterns that are made of development of attack signatures or even other activities of monitoring. The main advantage of using decision trees instead of other classification techniques is that they provide a wealthy set of rules that are understandable and can be combined with real-time technologies.

In anomaly detection that we focus on, the main interest is on attempting to identify patterns or events that do not match the expected ones or they are rarely defined. We use either supervised or unsupervised decision trees. The most common way, is the signature based approach which can be used in intrusion detection systems by creating trained data that can be used by supervised techniques. When, for example, an attack can be detected, its pattern is recorded and classified. These data combined with normal data has as a result the supervised training set. When we have unsupervised decision trees, our training data are not labeled. We have the guess about what is normal or anomaly and based on this guess the decision tree splits. Both supervised and unsupervised decision trees can be very significant, if we can make them in the form of random forests. Finally, we have another category of semi-supervised anomaly detection in which the training data consists only of normal data (without any anomalies). Theoretically, supervised methods provide better detection rate than semi-supervised and unsupervised methods, since they have access to more information.

Decision trees are *non-parametric.* They do not make any assumption about the distribution data. They are combining numeric or categorical or handle any missing data. In addition, because of the fact that decision trees are non-parametric, it has as a result being very flexible and is subject to *overfitting* training data. This problem can be addressed by pruning a tree before or after it has learned in order to remove some of the detail it has picked up. We define tree optimality as a number of leaves of the decision tree. There are used also some other criteria for the optimality, for example the number of nodes. For binary attributes, the criterion the number of leaves and the criterion the number of nodes are the same.

Another commonly used criterion is the average tree depth. This criterion is used when we have to deal with the CPU time. Knowing that we have the log(n) that comes from the average height of the tree after splitting. If we have continuous data, a common technique for finding the best possible split would be to sort the data into ascending order along that data point. Then we consider all possible partition points between those data points and taking the one that minimizes the entropy. This sorting step takes time O(n log(n)), which dominates the runtime. Since we are doing that for each of the O(m) features, the runtime ends up working out to *O(mn log(n))* total work done per node.

Another characteristic we have for decision trees is being multivariate or univariate decision trees. If there is one attribute per node is *univariate* and the opposite *multivariate.* One problem, when using univariate decision trees, is that in the whole path, they test some attributes more than once. Sometimes this prejudices the performance of the system, because with a simple transformation in the data, such as principal components, we can reduce the correlation between the attributes, and with a simple test realize the same classification. But the aim of multivariate decision trees is to perform different tests with the data. The purpose of the multivariate approach is to use more than one attribute in the test leaves.

Whereupon, some of the algorithms based on decision-tree techniques will be analyzed and compared. [1], [2]

ID3 algorithm

ID3 is a learning algorithm that builds a decision tree from a fixed set of examples. It is trained on a data-set to produce a decision tree (which is stored in the memory). The resulting tree is used for classification of future unseen samples and test cases. It builds the tree based on the information that learned from the trained instances, and after that uses the same to classify the test data. ID3 refers to both supervised and unsupervised training sets. In other words, it uses the values of the mentioned test cases in order to arrive at a terminal node, which is the one that can inform us the class that the test case belongs to. It usually uses nominal attributes for classification with no missing values. This model is defined as multi-class anomaly detection technique.

Functionality (input, algorithm, output):

ID3 Algorithm makes the following steps: First of all, we need to find the attribute that will be the root node in our decision tree. This gain is calculated of the entropy (S) of the set. After that, it creates root node for the tree. If all examples are positive it returns leaf node 'positive', else if all examples are negative it returns leaf node 'negative'. It continues by calculating the entropy of the current state H(S.). For each attribute, it calculates the entropy with respectively to the attribute 'x' denoted by H(S,x). It selects the attribute with the maximum value of I.G.(S,x) and removes the attribute that offers highest I.G. from the set of attributes. It repeats, until we run out of all attributes, or the decision tree has all leaf nodes.

Below, the algorithm will be presented.

Inputs: R: a set of non- target attributes, C: the target attribute, S: training data.

Output: returns a decision tree

Pseudo-code:

```
ID3 (Examples, Target_Attribute, Attributes)
Create a root node for the
tree If all examples are
positive
Return the single-node tree Root, with label =+;
If all examples are negative
Return the single-node three Root, with label =-;
If number of predicting attributes is empty,
Return the single node tree Root, with label = most common value of
the target attribute in the examples; else
begin A← The Attribute that best classifies
examples; Decision Tree attribute for Root = A.
For each possible value, vi, of A,
Add a new tree branch below Root, corresponding to the test A=vi;
Let Examples (vi) be the subset of examples that have the value vi
for A; if Examples(vi) is empty,
Add a leaf node with label = most common target value in the examples;
Else below this new branch add the sub-tree ID3 (Examples
      (vi), Target_Attribute, Attributes – {A})
End;
Return Root;
```

Remarks:

Although the algorithm ID3 is the fastest and shortest tree, it does *not ascribe the optimal* solution. It uses a greedy approach by selecting the best attribute to split the data-set, so it can get stuck. One solution is using backtracking during the search for the optimal decision tree. Moreover, another limitation is that is sensitive to features with large numbers, which might be a big problem if we use this algorithm in Internet. Finally, the ID3 algorithm has a difficulty in using it on continuous data because, there are more places to split the data, while searching for the best value split be grinding. ID3 can *overfit* to the training data. To avoid overfitting, smaller decision trees should be preferred over larger ones. This algorithm usually produces small trees, but it does not always produce the smallest possible tree.

ID3 is harder to use on continuous data. If the values of any given attribute is continuous, then there are many more places to split the data on this attribute, and searching for the best value to split by can be time consuming. ID3 is an *univariate* decision tree learner implementation. [3]

C4.5 algorithm

The C4.5 is an extension of ID3 algorithm and is collection of algorithms that are used for performing classifications in machine learning also. It is defined as a *supervised* technique because it is given a set of data representing data that are already classified. The way that the classification model is developed is as a decision tree. C4.5 includes three groups of algorithm: C4.5, C4.5-no- pruning and C4.5-rules. C4.5 algorithm is also a model referring to *multi-class* anomaly detection techniques.

Functionality (input, algorithm, output):

The C4.5 algorithm in order to learn the wanted rules, it combines the divide-and-conquer strategy with the separate-and conquer strategy of rule learning. It builds a decision tree from a set of training data in the same way as ID3 (with the concept of entropy). To be more precise, firstly, the algorithm builds a partial decision tree (sub-tree) on the current set of instances, satisfying the wanted termination criteria. After this step, it computes the information theoretic-criteria for all attributes and creates a new rule (created from the decision tree). C4.5 algorithm chooses the attribute of the data that most effectively splits its set of samples. In other words, the leaf with the largest coverage is the one that should be made into a rule. The operation continues with the third

step, the sub-dataset, that the decision tree (sub-tree) is discarded. Finally, we go again to the first step, calling again the C4.5 algorithm, and while we have already attached the sub-trees mentioned, we repeat. In the end it returns the tree we want.
There is a presentation of the algorithm.
Input: an attribute-valued data-set D *Output:*
returns a decision tree *Pseudocode:*

```
Tree={}
if D is ‘ pure’ OR other stopping criteria met
terminate end if for all attribute a D;
Compute information-theoretic criteria
if we split in an end for
a_best=Best attribute according to above computed criteria
Tree=Create a decision node that tests a_best in the root
D_v=Induced sub-datasets from D based on a_best
or all D_v
do Tree_v=C4.5(D_v);
Attach Tree_v to the corresponding branch of Tree;
end for;
Return Tree;
```

Remarks:
In general, C4.5 algorithm acts similar to ID3 (selects the best attribute based on the concept of entropy and e.g. for developing the tree) but with a little more improved behaviors. Also, as ID3, does *not attribute the optimal* solution. Besides, it has the possibility to use continuous data and using unknown values. Furthermore, attributes with different weights can be used. Being a greedy top-down learning of decision trees it is also vulnerable to *overfitting*. However it can use an algorithm to fix this with pruning the tree before being created. Additionally, it has pessimistic prediction error and the use of sub-trees. Finally, C4.5 creates a node per unique value for discrete values. This defines this tree learner as an *univariate* implementation. [4]

CHAID (Chi-Squared Automatic Interaction Detection) algorithm

It is one of the oldest tree classification methods. This name derives from the basic algorithm that is used to construct non-binary trees (the trees have more than two branches that can attach to a single node), that for classification problems relies on the Chi-square test to determine the best next split at each step. It uses *supervised* data training, however now it has been modified in using more. In addition, because it is based on that relatively simple algorithm is well suited for the analysis of larger data-sets. CHAID is another one *multi-class* anomaly detection classifier.
Functionality (input, output, algorithm):
CHAID algorithm, first of all has to prepare the predictors. It creates categorical predictors out of any continuous predictors by dividing the respective continuous distributions into a number of categories with an approximately equal number of observations. After, the algorithm merge the categories. It functions through the predictors to determine for each predictor the pair of categories that is least significantly different. It continues by selecting the split variable. It chooses the split the predictor variable with the smallest adjusted p-value (e.g. the predictor variable that will yield the most significant split). Finally, the algorithm continues the process until no further splits can be done.
We can see the algorithm below.
Input: independent variable columns, dependent variable
Output: returns a decision tree

Pseudocode:

```
Divide the cases that reach a certain node in the tree;
Cross tabulate the response variable (target) with each of the explanatory
variables;
if there are more than two columns,
find the "best" suitable formed by combining column categories; Compute
Pearson $X^2$ tests for independence for each allowable suitable Look for
the smallest $X^2$ value;
if it is not significant, combine the column categories if
the new table has more than two columns;
Repeat;
Broke apart categories combined previously
For each compound category consisting of at least 3 of the original
categories,
find the "most significant" binary split;
if $X^2$ is significant, implement the split and return Otherwise
retain the compound categories for this variable, move on to
the next variable
You have completed the "optimal" combining of categories for each
explanatory variable;
Find the most significant of these "optimally" merged explanatory variables
Compute a "Bonferroni" adjusted chi-squared test of independence for the
reduced table for each explanatory variable;
Use the "most significant" variable to split the node with respect to the
merged categories for that variable;
Repeat for each of the offspring nodes; Stop
if no more variable is significant;
```

Remarks:
CHAID is an algorithm for classification-type problems. It build non-binary trees that tend to be "wider". CHAID determines for each potential predictor the optimal n-ary split that can produce and then, it selects the predictor on basis of these optimal splits. So, about giving *an optimal* solution depends of the nature of the predictor (e.g. floating predictors).This has made the CHAID method more popular in market research applications. For example, it may yield a split on a variable: income by dividing that into four categories or groups of individuals belonging to these categories that are different with respect to some important consumer-behavior related variable. CHAID, is another algorithm vulnerable to *overfitting*, but it can fix it with pruning the tree using Chi-squared tests. Lastly, CHAID is a multivariate implementation. [5], [6]

POSC4.5 Algorithm

POSC4.5, is a decision tree and it is based on the previously mentioned C4.5. POSC4.5 has the ability to learn from positive and unlabeled data, having a good classification degree of correctness. Because of that data-set POSC4.5 is defined as an *unsupervised* decision tree. Besides because of the fact that POSC4.5 is based on a successful decision tree algorithm there are suggestions with even more improved performance, when POSC4.5 is being an ensemble by bagging, and classify testing samples by majority voting. On the other hand, POSC4.5 is *one-class* anomaly detection technique and it can be very interesting by this point of view.
Remarks:
POSC4.5 because of the unseen training data-sets it *cannot guarantee an optimal* solution. In addition, in order to avoid *overfitting* we should carefully tune the regularization. But in POSC4.5 we have overfitting against unsampled population. This, increases the likelihood of *overfitting* since it replicates the minority class events. We only consider data streams with discrete attribute values. For data streams with continuous attribute values, the continuous attributes could be discretized to

discrete attributes, so as to transform the original data stream into a data stream with discrete attributes only. [7]

OcVFDT (One-class Very Fast Decision Tree) Algorithm

OcVFDT algorithm refers to classification of data streams. Having data streams means that OcVFDT algorithm not uses a training set. It belongs to an extended version of the ultra fast decision tree. It is proved that this algorithm will not be contradictory with the incoming data. This is able to happen because the algorithm can maintain alternative trees. In addition, OcVFDT has better results in *one-class* classification. This can happens because most known one-class methods for data streams are just working in an incremental way and not by having methodologies for each concept. It adapts every time the changing context and forgets the no longer relevant samples. This procedure can increase its overall accuracy. However, OcVFDT has lower training time but also it has a lower accuracy. The algorithm will be presented below.
Input: a stream of samples (S), the size of validating chunk (nvalidate), the probability for a sample to be selected as validating sample (pvalidate), the feature space of stream (A), one minus the desired probability of choosing the correct attribute at any given node (δ), threshold (τ), the number of samples between checks for growth (nmin).
Output: a one-class very fast decision tree.
Pseudocode:

```
ValidateChunk = φ, T =
φ; for each i ∈[1, 9] do
 initialize a tree Ti with only a leaf (the root);
 T = T Ti;
 end for

for each sample s ∈S do
for each i ∈[1, 9] do
PosLevel = i
Grow(Ti, PosLevel, s, A, δ, τ, nmin);
end for
 if Random() ≤ pvalidate then
 ValidateChunk = ValidateChunk
 {s}; if |ValidateChunk| == nvalidate
 then estimate
 (T,ValidateChunk);
 ValidateChunk=φ;
 end if
 end
 if
 end
 for
 if ValidateChunk!=φ then
 estimate (T,ValidateChunk);
 End if;
```

Remarks:
OcVFDT uses data streaming. Because of being unsupervised, it *cannot assure for the optimal* solution. It also should be noted that OcVFDT has much lower training time than our methods, but this is at the cost of a much lower overall accuracy. In addition, since we use data streaming, OcVFDT has increased likelihood of *overfitting*. [8]

Conclusion

For classification-type problems, all these algorithms can be used to build a tree for prediction. Decision trees are one of the few methods that can be presented quickly enough to a non-specialist audience data processing without getting lost in difficult to understand mathematical formulations. [9] There is a table below that we can understand better and more organized the differences between the algorithms.

Algorithms	ID3	C4.5	CHAID	POSC4.5	OcVFDT
Accuracy	Depends on the testing instances	Depends on the testing instances	Depends on the testing instances	Depends on the unseen instances	Depends on the unseen instances
Classification	Multi-class	Multi-class	Multi-class	One-class	One-class
Code Availability	Yes	Yes	Yes	No	No
Complexity	O(logn)	O(logn)	O(logn)	O(logn)	O(logn)
Comprehensibility	Yes	Yes	Yes	Yes	Yes
Control Changes	No	No	No	No	No
Handle Parameters	Categorical/ Continuous	Categorical/ Continuous	Categorical/ Continuous	Categorical/ Continuous	Categorical/ Continuous
Incremental Learning	No	No	No	No	No
Noise	No	No	No	No	No
Number of Parameters	5(size for split, leaf size, gain, depth of tree)	5(size for split, leaf size, gain, depth of tree)	5(size for split, leaf size, gain, depth of tree)	5(size for split, leaf size, gain, depth of tree)	5(size for split, leaf size, gain, depth of tree)
Optimization	Prunning	Prunning	Chi-Squared/ Prunning	-	-
Training Data/ Quick Learning	Supervised	Supervised	Supervised	Unsupervised	Unsupervised
Variables	Multi-variable	Multi-variable	Multi-variable	Multi-variable	Multi-variable

1.1.2 Regression trees

Regression trees are similar to classification trees but in a regression model each node has predicted values. They are simple algorithms but the prediction accuracy often lags and also it can be computationally impracticable. [10]

Regression trees contain an output variable and one or more input variables that are defined as predictors. The output is defined as response and these variables can be both continuous and categorical. A regression tree can be considered as a variant of decision trees but it uses real valued function instead of classification methods. It splits the data into branches and it continues splitting each partition into smaller groups as the method moves up each branch. It has a training set that are grouped into the same partition and then the algorithm begins to allocate the data into the first two partitions or branches using every possible binary split on every field. The split that minimizes the sum of the squared deviations is selected and it is applied to the new branches as a new rule. It continues until we have a terminal node. [11]

The problem that the regression trees has is when they attempt to predict the values of a continuous variable from one or more continuous and categorical predictor variables because there are many different analytic procedures for fitting linear models, various types of nonlinear models etc where we can type in an arbitrary equation containing parameters to be estimate and produces results that are similar in nature. [12]

Because they are also trees they use almost the same pseudocode as the classification trees.

```
Start at the root node
For each X,
find the set S that minimizes the sum of the node impurities in the two
child nodes and
choose the split {X  S} that gives the minimum overall X and S
if a stopping criterion is reached, exit
otherwise, apply step 2 to each child node in turn
```

M5 algorithm

Is a regression tree algorithm that uses a more computationally efficient strategy to construct piecewise linear models. In the beginning it makes a piecewise constant tree and then it fits a linear regression model to the data in each leaf node. The tree structure is the same as that of a piecewise constant model, the resulting trees tend to be larger than those from other piecewise linear tree methods. [10]

GUIDE algorithm

This algorithm uses classification tree techniques to solve the regression problem. In every node it fits a regression model to the data and then it makes the computations. It defines a class variable that takes values depending on the sign of the residual (if it is positive or not). Then it applies the algorithm to the mentioned class variable so that it will split the node into two. It has more advantages because of this approach because the splits are unbiased, only one regression model is fitted at each node and also because that it is based on residuals the method is neither limited to piecewise constant models nor to the least squares criterion. [10]

CART algorithm

This algorithm is structured as a sequence of questions, the answers to which determine what the next question if any should be. The result of these questions is a tree like structure where the ends are terminal nodes at which point there are no more questions. The main elements of CART are the rules for splitting data at a node based on the value of one variable, the stopping rules for deciding when a branch is terminal and cannot be split anymore and the prediction for the target variable in each terminal node. [10]

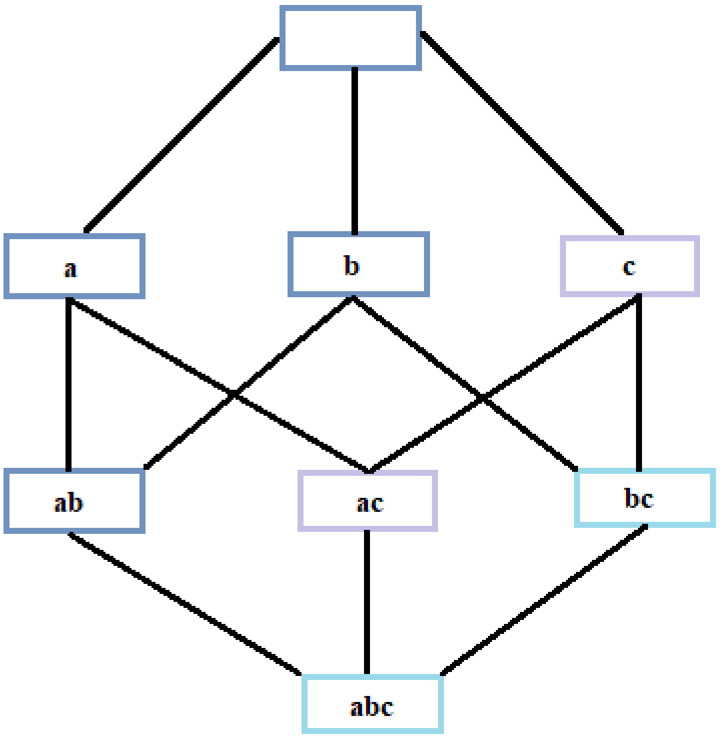
Conclusion

On the basis of the published empirical comparisons of these regression tree algorithms, GUIDE appears to have, on average, the highest prediction accuracy. It has the best linear and quadratic regression. Below, it is presented a comparison table.

Algorithms	MP5'	GUIDE	CART
Accuracy	Depends on the testing instances	Depends on the unseen instances	Depends on the unseen instances
Classification	Multi-class	Multi-class	Multi-class
Code Availability	Yes	Yes	Yes
Complexity	O(logn)	O(logn)	O(logn)
Comprehensibility	Yes	Yes	Yes
Control Changes	No	No	No
Handle Parameters	Categorical/ Continuous	Categorical/ Continuous	Categorical/ Continuous
Incremental Learning	No	No	No
Noise	No	No	No
Number of Parameters	5(size for split, leaf size, gain, depth of tree)	5(size for split, leaf size, gain, depth of tree)	5(size for split, leaf size, gain, depth of tree)
Optimization	Prunning	Prunning or Stopping Rules	Chi-Squared/ Prunning
Training Data/ Quick Learning	Supervised	Supervised/ Unsupervised	Supervised
Variables	Multi-variable	Multi-variable	Multi-variable

1.1.3 Association Rule Learners

Association rules analysis is a technique to uncover how items are associated to each other. It usually has two steps. The first one is to apply a minimum threshold so we can find all the frequent data in a database, and the second steps is to apply minimum confident constraint to these data so that we can form rules. The first step can be really difficult because it requires a lot of search and combination.
It is a method used for finding the relations and connection between the variables when we have large databases. This method is used for many application areas like, web usage mining, intrusion detection, continuous production, bioinformatics. In this method, we do not have to consider about the order of the items within a transaction. [1]



(1.3) Association Rule Learners

In the figure, frequency itself lattice. The color of the box defines the number of transactions that contain the combination of items. The lower levels of the lattice can contain at most the minimum number of their parents' items. [13]
Below, there are some algorithms that have been proposed.

Apriori Algorithm

This algorithm uses a breadth-first search strategy to count the support of itemsets and uses a generation function to find the support. This can reduce the number of itemsets we need to examine. If an itemset is frequent then all its subsets must also be infrequent. [13]
Functionality:

The itemsets that have to be examined can be pruned. We have to find itemsets with high support. The first step for this algorithm is to, start with item sets that contain just one single item. Then we have to determine the support for the itemsets. We have to keep the itemsets that meet your minimum support threshold and can remove itemsets that do not. After that, we use these itemsets and generate all the possible itemset configurations. We repeat until there are no more new itemsets. It has a join step, in which C_k is generated by joining L_{k-1} with itself and a prune Step where any (k- 1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. [14]

Pseudo-code:

C_k: Candidate itemset of size k

L_k: frequent itemset of size k

```
for each transaction t in database
do increment the count of all candidates in $C_{k+2}$
that are contained in t
$L_{k+1}$= candidates in $C_{k+1}$ with min_support
end
return ✉$_k$ $L_k$;
```

Remarks:

This algorithm is *computationally* expensive. Even though the apriori algorithm reduces the number of candidate itemsets to consider this number could still be huge when store inventories are large or when the support threshold is low. Also the analysis of large inventories would involve more itemset configurations and the support threshold might have to be lowered to detect certain associations. However if we lower the threshold we will increase the number of spurious associations detected. A solution could be to have a test data set. [14]

Eclat Algorithm

Eclat is a depth-first algorithm and uses set intersection. It is an algorithm suitable for sequential or parallel execution with locality-enhancing properties. It is another method for frequent itemset generation. [1]

Pseudocode:

```
Get tidlist for each item (DB scan)
Tidlist of {a} is exactly the list of transactions containing {a}
Intersect tidlist of {1} with the tidlists of all other items,
resulting in tidlists of {a,b}, {a,c},{a,d},….{a}-conditional database
(if{a} removed)
repeat from 1 on {a}-conditional database
repeat for all other items;
```

Remarks:

It has to determine the support of any k-itemset by intersecting tid-lists of two its (k-1) subsets. It has three traversal approaches the top-down, the bottom-up and the hybrid.

The advantage that it has is the fact that it has a very fast support counting. The disadvantage is that its intermediate tid-lists may become too large for memory. They are similar apriori and eclat. The differences between these are first of all that the apriori use large data set and eclat small and medium. In addition apriori is able to scan original data-set but eclat scans currently generated. Finally, apriori is slower than eclat. [15]

FP-growth algorithm

FP stands for frequent pattern. At first, the algorithm occurrence of items in the data set and stores them to 'header table'. Then it builds the FP-tree structure by inserting instances. Items in each instance have to be sorted by descending order of their frequency in the data set. Because of that the tree will be processed quickly. The items in each instance that do not have a minimum threshold are discarded. When we have many instances with frequent items the FP-tree provides high compression close to the tree root.

Functionality:

This method, grows large items directly. It starts from the bottom to the header table (having longest branches) by finding all instances matching given condition. After this procedure a new tree is created that counts projected from the original tree corresponding to the set of instances that are conditional on the attribute with each node getting sum of its children counts. This procedure ends when no more individual item condition on the attribute meet minimum support threshold and the process continues on the remaining header items of the original FP-tree. It uses the divide-and- conquer strategy.

It consists mainly of two algorithms. The first as mentioned for the tree construction. It has as an input a transaction database and a minimum support threshold and as an output a FP tree and a frequent-pattern tree of DB. The FP-tree is constructed as follows: Firstly it has to scan the transaction database. Then collect the set of the frequent items and the support of each frequent item. Then we create the root of an FP-tree and label it as null. [16] The FP-tree is constructed in two scans of the database. The first scan collects and sort the set of frequent items and the second constructs the FP-Tree. Referring to the FP algorithm we use it after we construct the tree.

Input: A database DB, represented by FP-tree constructed according to Algorithm 1, and a minimum support threshold.

Output: The complete set of frequent patterns.

Pseudocode:

```
FP-growth (Tree,a){
if Tree contains a single prefix path then {
let P be the single prefix-path of Tree;
let Q be the multipath part with the top branching node replaced by a null root;
for each combination (deloted as ß) of the nodes in the path P
do generate pattern ß ⌧ a with support = minimum support of nodes in ß;
let freq pattern set (P) be the set of patterns so generated; }
else let Q be Tree;
for each item ai in Q do {generate pattern ß = ai ⌧ a with support = ai.support;
contruct ß's conditional pattern-base and then ß's conditional FP-tree Tree ß;
if Tree ß ≠ Ø then call FP-growth(Tree ß , ß);
let freq pattern set(Q) be the set of patterns so generated;}
return(freq pattern set(P)⌧freq pattern set(Q)⌧(freq pattern set(P)×freq
```

Remarks:

When the recursive process will be completed, all the large item sets with minimum coverage that have been found and the association rule creation begins. [13]

Each dataset mined separately. Using the FP- growth it reduces the search costs looking for short patterns, offering good selectivity.

Conclusion

They are unsupervised systems that they try to find the connection between the variables. The algorithms are doing prune in order to reduce the total of the rules. Because of that they can be very effective in learning and even more when we have the right configuration of the parameters. In addition they are easy to understand. We can see the previously mentioned on the table below.

Algorithms	Apriori	Eclat	FP-growth
Accuracy	Depends on the testing instances	Depends on the testing instances	Depends on the testing instances
Classification	-	-	-
Code Availability	Yes	Yes	Yes
Complexity	O(logn)	O(logn)	O(logn)
Comprehensibility	Yes	Yes	Yes
Control Changes	No	No	No
Handle Parameters	Categorical/ Continuous	Categorical/ Continuous	Categorical/ Continuous
Incremental Learning	No	No	No
Noise	No	No	No
Number of Parameters	5(size for split, leaf size, gain, depth of tree)	5(size for split, leaf size, gain, depth of tree)	5(size for split, leaf size, gain, depth of tree)
Optimization	Prunning	Prunning or Stopping Rules	Chi-Squared/ Prunning
Training Data/ Quick Learning	Supervised	Supervised	Supervised
Variables	Multi-variable	Multi-variable	Multi-variable

1.1.4 Bayesian Networks-Based

Bayesian Network (BN) is a probabilistic graphical model that represents a set of random variables with their conditional dependencies through a directed acyclic graph (DAG). Bayesian technique is used for anomaly detection in a *multi-class* setting and uses *uni-variable* categorical data set. It has a test data instance and it can estimate the posteriors probability, via a probability function, of observing a class label from a set of normal class labels and the anomaly class label. The input in the mentioned function is a particular set of values for the node's parent variables. The result that we want as the predicted class for the give test instance, is the one with the largest posterior or the probability of the variable represented by the node. The likelihood for these probabilities is estimated from the training data set. The technique assumes independence between the different attributes (the nodes are not connected) and, because of that, a Bayesian network is often used and able to represent the probabilities of the presence of many diseases (given symptoms). The nodes can represent random variables like, quantities, latent variables, unknown parameters, etc. Another example is if parent nodes are Boolean variables, the probabilistic function would be represented as entries. Every entry should be a possible combination about being true or false of its parents. There are a lot of algorithms in inference and learning Bayesian networks.
Additionally, there is an extension in Bayesian Networks. This is the naive Bayes classifiers. Naive

Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. [17]
Regarding to the complexity in Bayesian their networks are *NP-hard* and their learning can be both *supervised or unsupervised*. In real world the use of Bayesian networks, in large real-life, applications would need to be tempered by either topological structural constraints, such as Bayes networks or by restrictions on the conditional probabilities. [18], [19].
Bayesian Networks have a lot of benefits. Several variants of the basic technique have been proposed for network intrusion detection, for novelty detection in video surveillance, for anomaly detection in text data, and for disease outbreak detection. The general modeling beliefs of Bayesian network are used in computational biology, bioimformatics, medicine, biomonitoring, image processing, data fusion, decision support systems, engineering, sports betting, gaming, geophysics, volcano monitoring, law, study design and risk analysis.
Some algorithms will be presented:

WSARE 3.0 (WSARE: What's Strange about Recent Events?) Algorithm

WSARE 3.0 is an algorithm used for performing early detection of disease outbreaks by searching a database of emergency cases for anomalous patterns. Traditional techniques for anomaly detection are not able to satisfy this problem because they identify individual data points and cannot combine features. These mentioned traditional algorithms can find outliers of strange events but not indicative of a new outbreak. With WSARE 3.0 researchers have the ability to detect groups with specific characteristics that have a recent pattern of illness that is anomalous relative to historical patterns.
The Bayesian Network is able to generalize from days that do not match today precisely, producing an estimation of the desired conditional distribution. On intermediate days, WSARE 3.0 simply updates the parameters of the previously learned network without altering its structure. WSARE 3.0 wants and searches for a variety of anomalous rules or patterns, and thus does not need either a prespecified type or location. Determining the baseline is difficult due to the presence of various trends in data, such as trends caused by the day of week and by seasonal variations in temperature and weather. [20]
Functionality:
The WSARE algorithm consists of three steps. First of all, the baseline data-set is created. Secondly, the algorithm searches for the best scoring rule using both the recent and the baseline data-sets. Finally, a value for the best scoring rule is calculated using a randomization test.
To be more precise, the baseline data-set is created by a Bayesian network. Creating the baseline distribution without taking trends into account can lead to unacceptably high false positive counts and slow detection times. WSARE 3.0 accounts for these trends by using a Bayesian network to model the joint probability distribution of the features of the data. These features can be divided into environmental attributes (features such as the season and the day of week that cause trends in the data) and response attributes (the remaining features such as syndrome and gender). WSARE 3.0 learns a Bayesian network from all data from before the recent period. During this Bayesian network structure learning phase, environmental attributes are prevented from having parents because we are not interested in predicting their distributions, but rather, we want to use them to predict the distributions of the response attributes. Once the Bayesian network structure is learned, we can then produce a conditional probability distribution that represents the baseline behavior given the environmental attributes for the current day. Once the baseline data set is generated, we need to search for the best scoring rule, which characterizes the group with the most unusual shift in proportions between the baseline and recent data sets.
In order to prevent over-fitting, additional components are only added to the best rule so far if the

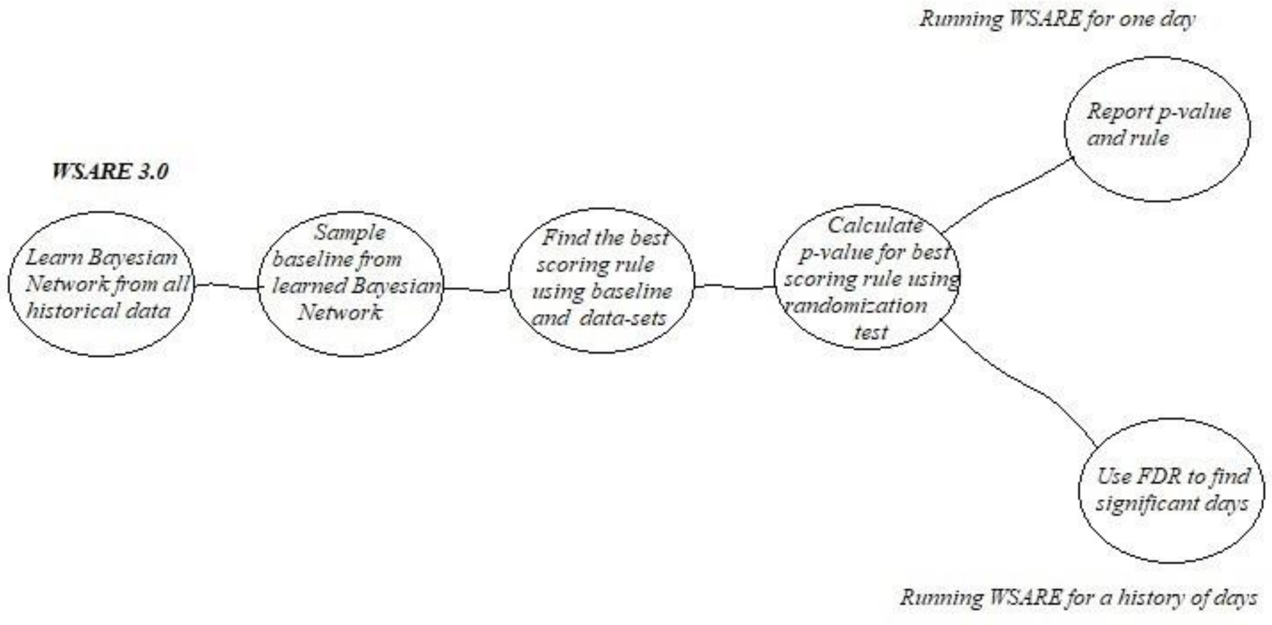
addition of those components is statistically significant. The final step of WSARE accounts for the multiple hypothesis testing problem by calculating a compensated value for the best scoring rule through a randomization test in which the date and the remaining features are assumed to be independent. [20], [21]

Remarks:

WSARE 3.0 is able to successfully identify anomalous patterns in the data. Another advantage that this algorithm has is that has significantly lower detection times than a standard detection algorithms. WSARE 3.0 purpose is to be a general safety net for emerging problems that have not been anticipated. Besides, algorithm innovations have been considered. First of all, it is able to turn the problem of detecting the emergence of new patterns in recent data, into a question (e.g. is it possible to learn a propositional rule that can significantly distinguish whether records are most likely to have come from the recent past or from the more distant past?). In addition, it can incorporate levels of significance tests into rule learning to avoid *overfitting* by multiple testing. Finally, examines the domain of early outbreak detection.

The best rule for a day is found by considering all possible one-component and two-component rules occurring events on that day and returning the one with the best "score." The score is determined by comparing the events on the current day with events in the past. The best scoring rule then has its value estimated by a randomization test. This value is the likelihood of finding a rule with as good a score under the hypothesis that the features of the case and the date are independent. The randomization-based value takes into account the effect of the multiple testing that went on during the rule search. If we were running the algorithm on a day by-day basis, we would end at this step. If we were looking at a history of several days, we would need the additional step of using the false discovery rate to determine which values are significant. The days with significant values are returned as anomalies.

Referring to the *parameters* we have: the max_rule_components (the maximum number of components to a rule, the num_randomizations (the number of iterations to the randomization test), a_{FDR} (the significance level of false discovery rate), environmental_attributes (attributes that account for temporal trends), num_baseline_samples (the number of sampled records from the baseline Bayesian Network). [20], [21]


Running WSARE for one day
WSARE 3.0
Learn Bayesian Network from all historical data
Sample baseline from learned Bayesian Network
Find the best scoring rule using baseline and data-sets
Calculate p-value for best scoring rule using randomization test
Report p-value and rule
Use FDR to find significant days
Running WSARE for a history of days

(1.4) A schematic overview of the steps involved in WSARE algorithm

BNDO (Bayesian Network Outlier Detector) Algorithm

Is an algorithm that is able to find Bayesian Networks using Gaussian Mixture Models and advanced caching. It can be used for massive databases outlier detection and has as a goal to find an explanation about what can define a subset strange.
Input: Bayesian model(BN(N,E)), parameters minconf and maxconf, and a test-set
Output: top n low probability data points in a test-set
Pseudocode:

```
Compute minsup for all parents nodes
X   BN(N,E) for every test case in test sets,
repeat the next steps;
Compute conditional probability in child node given their parent(s)
(P(y)=Pr(y | Pa(y)) where Pa(y)   X);
Apply rules $R_1$ and $R_2$ to uncover anomalous patterns;
Compute joint probability;
Sort join probability;
Output top n low scored data points;
```

Remarks:
In order to implement BNDO we use sparse implementation using Gaussian Mixture Models for *continuous* variables modeling. The difficulty is to find out the number of the clusters (Gaussians) are needed. The functions that are used are likelihood. In addition, because of the low dimension that is needed due to BN factorization into local conditional probabilities the massive data-sets are now practical and all the variables are also *independent*.
The *complexity* that is needed is $O(n^2*m)$ for the operations in order to collect the statistics that are required. Besides, we need more research so that we can find the *optimal* solution. This is because there are equally sized bins that are used in indiscretion as an approximation to the real values of discrepancy. The parent size should be low because is the one responsible for the computation costs. It is important to choose a score which can be decomposed using BN factorization.
The detection mentioned of strange objects is based on each object probability and selecting the worst *defined* parameters. If BN training of data is correct, most objects should be well modeled. [22]

Naive Bayes

Naive Bayes is a classifier that is based on Bayes' Theorem but there is independence among the predictors. In other words, it is assumed that the presence of a particular attribute in a class is not related to any other attribute. Because of this independence, Naive Bayes is considered a *multivariate* classifier but it needs only univariate (conditional) probability distributions. It is a classifier, that is easy to be built and it can handle very large data-sets because it focus on the strong independence assumptions in the model rather than the particular distribution of each feature/attribute. We have three models Gaussian, Multinomial, Bernoulli and we use them depending on the data-set we have. [23]
Functionality:
Firstly, we have our training data-set. Our goal is to classify them. We begin with converting the data-set into a frequency table. After, we create a likelihood table by finding the probabilities needed for our problem. Then, we use the Naive Bayesian equation so we can calculate the posterior probability for each one of our tests. Assume that P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes). P(c) is the prior probability of class and P(x|c) is the

likelihood which is the probability of predictor given class. P(x) is the prior probability of predictor. The class with the highest posterior probability is the outcome of our prediction. [24]

Input: a training data-set S, an unknown instance x'

Output: conditional probability tables, the posterior probability

Pseudocode:

```
LEARNING PHASE: Given a training set S.
For each target value of $c_i(c_1,…c_l)$
$P(C=Ci) \leftarrow$ Estimate $P(C=Ci)$ with examples in S;
Output: conditional probability tables; for $x_j$, $N_j*L$ elements
TEST PHASE: Given an unknown instance $x'(a'_1,… a'_n)$
Look up the tables to assign the label c* to x' if
$[P(a'_1|c^*)… P(a'_n|c^*)]P(c^*)>P(a'_1|c)… P(a'_n|c)]P(c)$, $c!=c^*$, $c=c_1,… c_l$
```

Remarks:

Naive Bayes has a lot of advantages. First of all, it is easy to predict class of test data-set and it can also perform well in a *multi-class* prediction. Given a way to train naive Bayes classifier from labeled data, normally we use a *semi-supervised* training algorithm. Additionally, because of the independence, we need less training data. When we have *categorical* input variables, we have a good performance. Contrary, if categorical variable has a category which was not observed in training data, the model will assign a zero probability so we will not have any prediction. In order to avoid this problem, we use *smoothing technique*. Also, naive Bayes is now always precise because in real life it is almost impossible to have a set of predictors that are completely independent.

Naive Bayes requires a number of parameters linear in the number of variables in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression which takes linear time, thus referring to the *complexity* we have O(n). [23]

Conclusion

Naive Bayes is technically a special case or an extension of Bayesian networks. As mentioned Bayesian represent a set of variables as graph of nodes modeling dependencies between those nodes as edges, and there must be made inherent assumptions about dependence and independence between the variables. In real world, this cannot happen, but in order to be more practical the network is simplified with the approximation that variables which are almost independent, are completely independent. Naive Bayes is almost the same but even more simplified by having all the features conditionally independent of each other. Although, the Bayesian rule for probability is used. Usually this independence assumption works well for most cases. All the dependence in Bayesian Network has to be modeled. They can work well both in *discrete and continuous variables.*

There is a comparison table below:

Algorithms	WSARE 3.0	BNDO	Naive
Accuracy	Yes	No	No
Classification	Multi-class	One-class	Multi-class/One-class
Code Availability	No	Yes	Yes
Complexity	NP-hard	O(n^2+m)	O(n)
Comprehensibility	-	-	-
Control Changes	No	No	No
Handle Parameters	Categorical/ Continuous	Categorical/ Continuous	Categorical/ Continuous
Incremental Learning	Yes	No	No
Noise	-	-	-
Number of Parameters	Depends on the conditional probability tables (CPT)	Depends on the conditional probability tables (CPT)	Depends on the conditional probability tables (CPT)
Optimization	-	-	Smoothing
Training Data/ Quick Learning	Unsupervised/ Supervised	Unsupervised/Supervised	Unsupervised/Supervised
Variables	Multi-variable	Multi-variable	Multi-variable

1.1.5 Clustering Based Anomaly Detection Techniques

The method of identifying similar groups of data in a data-set is called clustering and is one of the most complexes, well-known and most studied problems in data mining theory. In classification, the main concept, as we have already mentioned, is to predict the target class by analysis the training data-set with the use of proper boundaries for each target class. On the other hand, in clustering the main idea is not to predict the target class as a classification but to try grouping the similar kinds considering the most satisfied condition. In other words, all the items that belong to the same group should be similar.
In clustering, because *it is not a classification* we do not have to label points (as previously mentioned in decision trees with supervised). Multivariate analysis in statistics is a set of useful methods for analyzing data when there are more than one variables under consideration. Multivariate analysis techniques may be used for several purposes, such as dimension reduction, clustering or classification. Referring to this analysis we will concentrate on clustering techniques. The goal of clustering analysis is to establish a set of meaningful groups of similar objects by investigating relationships between objects. For example, if you have data from customers, it is possible to segment customers into clusters based on their buying habits. Then, we use clustering results to custom tailor the marketing efforts. But, there are algorithms like k-means, that computes the distance between two points, it is not able to happen with categorical (low, medium, high) variables. As a result a simple workaround for multiple categorical variables is to calculate the percent of times each variable matches in comparison to the cluster centroid.
Clustering and anomaly detection seem to be fundamentally different from each other, even though several clustering-based anomaly detection techniques have been developed. Clustering itself is not one specific algorithm, but a general task to be solved and it can be achieved by various algorithms. The aim of our study is to show which clustering based technique algorithm is more suitable. [25], [26]
Below some algorithms will be presented:

K-Means Algorithm

The K-means clustering algorithm is simple and easy way to classify a given data-set, since it is one of the simplest unsupervised learning algorithms that can solve the well known clustering problem. K-means clustering which has as main idea to define <k> centroids, one for each cluster, is a method of classifying or grouping items into <k> groups (where <k> is the number of pre-chosen groups). These centroids should be placed in a cunning way because of different location that causes different result. The grouping is done by minimizing the sum of squared distances (Euclidean distances) between items and the corresponding centroid. Because of this, the best choice is to place them as much as possible far away from each other.
Functionality (input, algorithm, output):
The first step that the algorithm has to make is to place <k> points into a space. This space is represented by the objects that are being clustered. These points represent initial group centroids. After, each object, that has the closest centroid, has to be assigned to the group. When the operation is finished and all the objects have been assigned, the algorithm has to recalculate the positions of the <k> centroids. The algorithm repeats the steps mentioned until the centroids cannot move anymore. This whole procedure produces the separation of the objects and categorize them into groups from which the metric that should be minimize can be calculated. This marks the completion of the algorithm.
Input: $E=\{e_1,e_2,\ldots,e_n\}$ (set of entities to be clustered), k(number of clusters), MaxIters(limit of

iterations)
Output: $C=\{c_1,c_2,\ldots,c_k\}$(set of cluster centroids), $L=\{l(e)|e=1,2,\ldots,n\}$(set of cluster labels of E)
Pseudocode:

```
for each $c_i$ C
do {$c_i$← $e_j$ E
end
for each ei E
do l($e_i$)← argminDistance($e_i$,$e_j$)j {1...k}
}end
changed←false;
iter←0;
repeat{for each {$c_i$ C
do update cluster ($c_i$);
}end
for each {$e_i$ E
do min Dist←argminDistance($e_i$,$c_j$)j {1….k};
if min Dist l($e_i$) then
l($e_i$) ← minDistl
changed←true;
}end }end
iter++
}until changed =true and iter<=MaxIters;
```

Remarks:
K- means can be viewed as a greedy algorithm for partitioning the n samples into k clusters so as to minimize the sum of the squared distances to the cluster centers. K-means algorithm has a lot of advantages. Firstly, if we have a lot of *variables*, K-means can be computationally faster than other algorithms (e.g. hierarchical clustering) and produce tighter clusters, especially if the clusters are spherical. On the other hand, there can be a difficulty in comparing quality of the clusters produced and in predicting what K should be when we have fixed number of clusters.
Referring to the parameters, the initial step of k-means is to randomly select <k> cluster centroids. Normally they are selected from the data points itself in order to ensure fast convergence. As a result we have n*k parameters: 'n' is the number of dimensions of each data point, and 'k' the number of the clusters. We also have to assign labels to each of the 'p' data points based on a distance measure to the cluster centroids.
To conclude each centroid that has n elements means that for all centroids we have *n*k parameters*.
The complexity that k-means algorithm has is: *O(n*k*i*d),* where:

- 'n'= number of points,
- 'k'=number of clusters,
- 'i'=number of iterations and
- 'd'=number of attributes.

The correct choice of k is not easy because it depends on the shape etc. The problem is computationally difficult (NP- hard), however there are efficient heuristic algorithms that are commonly employed and converge fast to a local optimum. There are several categories of methods for making this decision, *the optimal decision.*
One method to validate the number of clusters is the elbow method. The idea of the elbow method is to run k-means clustering on the dataset for a range of values of <k>, and for each value of <k> calculate the sum of errors(SSE), which is the sum of the squared differences between each observation and its group's mean. However, the elbow method is not always correct (e.g. if the data are not very clustered). Finally, it has been proved that, through the *regulation* of distance metric parameters one can achieve better clustering effects than the traditional k-means, and has an

advantage both in run time and number of iterations. [27], [28]

DBSCAN (Density-based spatial clustering of applications with noise) Algorithm

DBSCAN is a data clustering algorithm which is density-based. In this method clustering is based on density such as density is a connected point. Each cluster has a considerable higher density of points than outside of the cluster. In other words, it is given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that being alone in low-density regions (the nearest neighbors of which are too far away). DBSCAN is one of the most common clustering algorithms. In general, it also consists of two global parameters. The first one is ε-eps (the local radius for expanding clusters): maximum radius of the neighborhood and the second one is minpts: minimum number of points in an eps- neighborhood of that point. Eps can be considered a step size (DBSCAN never takes a step larger than this, but by doing multiple steps DBSCAN clusters can become much larger than eps. [29] *Functionality (input, algorithm, output):*
First of all, the DBSCAN algorithm randomly selects a point <p>. Then it has to find and get back all the points that are density-reachable from ε-eps and minpts. If <p> is a core point, then a cluster is formed, but if p is a border point and there are no points that are density reachable from p then DBSCAN visits the next point of the database. This whole process is continued until all of the points will be processed.
Input: the neighbor list size, eps, minpts
Output: clustering results
Pseudocode:

```
DBSCAN(DB, dist, eps, minPts) {
C=0
for each point P in database DB{
if label (P) ≠ undefined then
continue Neighbors N = RangeQuery(DB, dist, P, eps)
if |N| < minpts then {label(P)= Noise continue}
C=C+1
label (P) =C
Seed set S=N\{P}
for each point Q in S{
if  |N|  ≥  minpts then {  S = S
                `
     }
}
```

Remarks:
Recently, many researchers have combined clustering algorithms with optimization and meta-heuristic algorithms to improve the results of clustering. However, there is no research to solve the problem of automatically choosing input parameters in DBSCAN algorithm. DBSCAN algorithm requires two *parameters* -epsilon, which specifies how close points should be to each other to be considered a part of a cluster and minPts, which specifies how many neighbors a point should have to be included into a cluster. Thus, in order to have the *optimal* result there should be an efficiency very well suited DBSCAN input parameters. The first step, we have to do is to choose a good distance function for our data (we can also use *regulation*). As for "minPts" it is on our data needs, and it is coupled with Epsilon. In general, there is no way of choosing minPts but it depends on

what we want to find (e.g. a low minPts means that we will have more clusters from noise).
Regarding the functionality, firstly, when we use DBSCAN we do not have known the exact number of clusters like in k-means algorithm (the previous one mentioned). Because of that, DBSCAN has the ability to find randomly shaped clusters even if these clusters are surrounded by a different cluster. Besides, DBSCAN has a notion of noise. As in differently, DBSCAN can only result in a good clustering as good as its distance measure is. The most common distance metric used is the Euclidean distance measure (for high-dimensional data, it is hard to find an appropriate value for epsilon). Thus, to conclude, DBSCAN cannot cluster data sets well when they have large differences in densities, because the minpts-epsilon combination cannot be chosen appropriately for all clusters.
The overall average run time complexity of DBSCAN is O (n^2) as if for each point it has to be determined if it is a core point. It can be reduced to O (n*log(n)) in lower dimensional spaces by using efficient data structures (n is the number of objects to be clustered). When to DBSCAN is applied a *regulation* it performs well. It can identify solutions for the inference of regulatory networks. A regulatory factor based on cluster density is proposed to correct the distance measure. [29], [30], [31].

CURE (Clustering Using Representatives-Hierarchical Clustering) Algorithm

CURE is an agglomerative hierarchical clustering algorithm. The CURE algorithm is an efficient data clustering algorithm for large databases. The hierarchical methods group training data into a tree of clusters (called dendrogram), while the merges and splits are determined in a greedy manner. In addition, strategies for hierarchical clustering generally fall into two types the agglomerative and the divisive. CURE is an agglomerative as mentioned. In other words, CURE produces a sequence of clustering of decreasing number of clusters, <k>, at each step. The clustering produced at each step results from the previous one by merging two clusters into one. The representative points it consists of, attempts to capture the physical shape and geometry of cluster which is non-spherical as we have used to. By shrinking the points towards center mitigates the effects of outlier. The larger the movement in the outlier is the more it reduces their ability to cause the wrong cluster to be merged.
Functionality (input, algorithm, output):
The CURE algorithm begins with taking each input as a separate cluster. When it has a successive step it merges the closest pair of clusters. The representative points are stored in order to have the ability to compute the distance between a pair of clusters and also are used to compute the distance from other clusters. Firstly, there should be chosen a set of points well scattered points with in the cluster, and after these are enshrined toward the center of the cluster.
Input: a set of points S
Output: k clusters

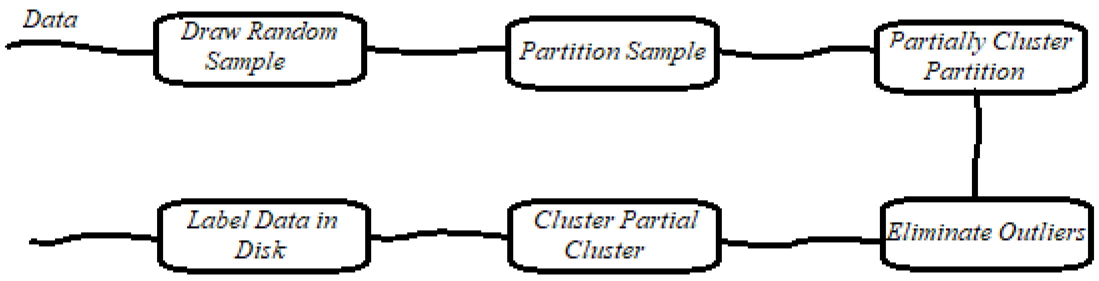
Pseudocode:

```
procedure cluster (S, k)
begin
T:= build_tree(S)
Q:= build_heap(S)
while size(Q) > k
do {
u:= extract_min(Q)
v:= u.closest
delete(Q,v)
w:= merge(u,v)
delete_rep(T,u); delete_rep(T,v);
Insert_rep(T,w)
w.closest:= x
for each x Q do{
if dist (w, x) < dist(w, w.closest)
w.closest:= x
if x.closest is either u or v{
if dist(x, x.closest) < dist(x, w)
x.closest := closest.cluster(T, x, dist(x, w))
else x.closest:=w
relocate(Q,x)
}
else if list (x,x.closest)>dist (x,w){x.closest:=w
relocate(Q,x)}
}
insert(Q,w)
}end
```

Remarks:

This clustering algorithm can recognize arbitrarily shaped clusters and also robust to the presence of outliers. CURE is another algorithm that has some disadvantages. For example, it ignores the information about the aggregate inter-connectivity of objects in two clusters. Referring to the complexity, the algorithm uses space that is linear in the input size n and has a worst-case time complexity of $O(n^2 \log(n))$. For lower dimensions, the complexity can be shown to further reduce to $O(n^2)$. Though many effective clustering algorithms have been developed most of them still lack of automatic decision for optimal number of clusters. An optimal cluster configuration is defined as an outcome of all possible combinations of groupings, which presents a set of the most beneficial associations. As a result, it is defined that clustering gain is considered as clustering *optimal*, which is based on the squared error. Therefore, when we apply the measure to the CURE algorithm we have the optimal number of clusters that can be found. However, we have to note the fact that the optimal clustering configuration discovered by a hierarchical clustering algorithm has maximum clustering gain.

Since clustering gain is minimum at the initial and final clustering stages, an optimal configuration should be found during the middle of the clustering procedure. In order to determine the maximum clustering gain during the middle of the clustering procedure, the clustering gain is considered as an effectiveness criterion. The *parameters* that CURE uses are: sample size, number of representatives in cluster, number of partitions, shrink factor. The quality and effectiveness of CURE can be tuned be varying the mentioned parameters to adapt different input data set. [32]



1.5 *CURE algorithm steps*

Conclusion

There have been mentioned advantages and disadvantages of each algorithm and a brief description. However, on the basis of most researchers found that k-means clustering algorithm is simplest algorithm as compared to other algorithms.
It is presented a comparison table.

Algorithms	K-means	DBSCAN	CURE
Accuracy	No	No	No
Classification	No	No	No
Code Availability	Yes	Yes	Yes
Complexity	O(n*k*i*d)	$O(n^2)$	$O(n^2 logn)$
Comprehensibility	No	No	No
Control Changes	Yes (regulation)	Yes	Yes
Handle Parameters	Categorical	Categorical	Categorical
Incremental Learning	Yes	Yes	Yes
Noise	-	Yes	-
Number of Parameters	5(clusters in measurement data, optimum number of clusters, cluster over consecutive time snapshots, estimate cluster parameters)	-	-
Optimization	Distance Method	Density Method	Hierarchical Method
Training Data/ Quick Learning	Unsupervised	Unsupervised	Unsupervised
Variables	Multi-variable	Multi-variable	Multi-variable

1.1.6 Statistical Methods (Parametric and non-parametric)

When we use statistical methods for anomaly detection, we observe the activity of what we are interested in and then we generate profiles so that there would be a generated behavior. Because of that we have two profiles the current one and the stored one. When the events of the system are processed the intrusion detection system updates the current profile and calculates an anomaly score by comparing the current profile with the stored using a function of abnormality of all measures within the profile and if the anomaly score is higher of a threshold that we have define then we get an alert.

This approach has many advantages. First of all, it is not required prior knowledge of security flaws. Additionally, it can be provided accurate notification of malicious activities that typically occur over extended periods of time. On the other hand, it has also disadvantages. Many attackers are able to train a statistical anomaly detection to accept anomaly behavior as normal. It can also have difficulties in determining the thresholds mentioned. In addition, statistical methods need accurate statistical distributions, but not all behaviors can be modeled using only statistical methods. [34]

The profile of normal behavior is on a selected set of variables that is maintained by the statistical analysis unit. This enables the system to compare the current activity with the expected values of the audited intrusion detection variables that is stored in the profile and then flag an anomaly if the audited activity id far from the expected behavior. Each variable that is in the stored profile reflects the extent of which particular type of behavior is similar to the profile built for it under "normal conditions". [33]

Statistics are more common with parametric rather than non parametric analyses. Non parametric are also called distribution free because they do not assume that our data follow a specific distribution. The differences are the following, first of all parametric tests are able to perform also well with continuous data even if they are not normal (non distributed). In addition, parametric can perform well even when the spread of each group is different contrary to non parametric that all the data of all our groups must have the same spread otherwise we will have not valid results. Finally, parametric tests have more statistical power but non parametric can be used in a very small sample size. Depending of that we can choose which on to use. [33]

1.1.7 Operational Anomaly Detection

The operations is a highly specialized context for anomaly detection. If we check a dataset we have issues of scale or dynamism of data, asymmetric severity of failures, limited time etc. Referring to the scale or dynamism of data the operations that we are looking at a huge dataset is receiving new, streaming data. About the asymmetric severity, there are very different costs that associated with false positives or false negatives in operations. After, about the time limitation we have many heavyweight approaches that cannot be used because they are not trained in an operationally relevant time window especially given the scale of data. It is used normal (or Gaussian distribution). [35]

There is also a comparison table.

Algorithms	Parametric	Non-parametric	Operational
Accuracy	No	No	No
Classification	Multi-class	Multi-class	Multi-class
Code Availability	-	-	-
Complexity	O(n)	O(n)	O(n)
Comprehensibility	Yes	Yes	No
Control Changes	-	-	-
Handle Parameters	Continuous	Continuous	Continuous
Incremental Learning	-	-	-
Noise	-	-	-
Number of Parameters	-	-	-
Optimization	-	-	-
Training Data/ Quick Learning	Unsupervised	Unsupervised	Unsupervised/Supervised
Variables	Statistical	o	Statistical

1.1.8 Smoothing

Smoothing is a simplification of the time series anomaly detection. It is complicated because of practical issues. Time series are either are based on a model or its history and make decisions about the values that arrive later. It is like prediction or forecast by a fitting model to some sample data. Then we use the model to predict the future values. This can be very complicated in real life applications. Because of that smoothing are used.
The kind of time series that we will use vary. It depends on whether we are trying to predict or not. Smoothing refers to the use of an exponentially weighted moving average (EWMA) to smooth a time series. If we have time series x_t we can define a new time series s_t that is a smoothed version of x_t. $st=\alpha xt+(1-\alpha)st-1$. The smoothing amount changes with which is the smoothing weight. The smaller the weight, the less influence each point has on the smoothed time series. If we suppose that we have *xt* with *st* and we want to predict the next value wich is $xt+1$. We can use the last value that we calculated for the EWMA, s_t. This is the way it works because our smoothed time series is the EWMA of our original series and because of the way averages work. This is considered as a good prediction when the time series are stationary.
There are two more categories of time series. Trend and seasonality. Stationary focus on the stability of the values of a time series and they do not have any increasing or decreasing pattern and have approximately the same value. This makes predictions more helpful.
Real time anomaly detection is mainly a forecasting problem because you cannot know what to expect unless you have meet it in the past. Forecasting time series data can be very complicated and sophisticated, but when we make it with more simple techniques like an EWMA we can have most of the benefit with a small fraction op the cost, the effort or the complexity. The complex techniques can be good for specific and more precise cases. [36]
Below, some algorithms will be presented.

STL Algorithm

STL is a smoothing algorithm. The main idea for STL is that at time series can be decomposed into three components: seasonal, trend and remainder. If we should consider it as a Seasonal, Trend or remainder is a matter of opinion and depends on the choice of model and parameters. Key to the STL approach is Loess (Local regrESSion) smoothing but again it is based on the parameters that we will choose for the model choice.
Functionality:
The goal it to separate a time series to trend, seasonal and remainder components. This can be done with two loops. The first one is to assign each data point depending on the size of the remainder which allows for reducing the effects of outliers. The other loop updates the trend and seasonal components. After that will be done the time series are partition into cycleosubseries which are less smoothed and then passed through a low-pass filter. The seasonal components are subtracted from the raw data. The result is loess smoothed which becomes a trend. And the left is the remainder.

Pseudocode:

```
Compute minsup for all parents nodes X  BN(N,E)
Initialize trend as T(0)ν=0Tν(0)=0 and R(0)ν=0Rν(0)=0
Outer loop - Calculate robustness weights
Run n(o)n(o) times
Calculate RνRν
Calculate robustness weightsρν=B(|Rν|/h)ρν=B(|
where  h=6*median  (|Rν|)h=6median(|Rν|)   ans  BB  is  a  bi-square  weight
function
on initial loop, ρνρν = 1
Inner loop- calculate trend and seasonal terms.
Run n(i) n(i) times
```

Remarks:

This model has six main parameters. These are: the periodicity of the seasonality (n(p)n(p) (n.p), the number of cycles through the inner loop (n(i)n(i) (inner)), the number of cycles through the outer loop (n(o)n(o) (outer)), the span in lags for the low-pass filter (n(l)n(l) (l.window)), the smoothing parameter for the seasonal filter ((n(s)n(s) (s.window)), the smoothing parameters of the trend behavior (n(t)n(t) (t.window)). To these six parameters, the degree of the loess smoothing can be changed because of more parameters. The procedure of selecting parameters can be the most important. They control the smoothing of the seasonal data. So what variation is considered seasonal versus the trend or the remainder is the decision. The smaller the value, the less smoothing in each cycle-subseries. [37]

Autoregressive Integrated Moving Average Model (ARIMA)

Is a class of statistical models for analyzing and forecasting time series data. It provides a powerful method for making skillful time series forecasts. Arima is a generalization of an autoregressive moving average model. These models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). Arima are applied in cases where data show evidence of non-stationary where, an initial difference step can be applied one or more times to eliminate the non-stationary.

Functionality:

We have to model the identification. We use plos ans statistics to identify trends, seasonality, and autoregression elements to get an idea of what will be required. Then we have the parameter estimation. The final step is the model checking. We use plots and statistical tests to determine the amount and type of temporal structure not captured by the model.

The process is repeated until either a desirable level of fit is achieved on the in-sample or out-of-sample observations (e.g. training or test datasets).

Pseudocode:

```
Define  the  model  by  calling  ARIMA()and  passing  in  the  p,  d,  and  q
parameters.
The model is prepared on the training data by calling the fit()function.
Predictions can be made by calling the predict ()function and specifying the
index of the time or times to be predicted.
```

Remarks:

With the ARIMA model we can forecast future time steps. It accepts the index of time steps and make predictions as arguments. These indexes are relative to the start of the training data-set used to make predictions. This training data set would return an array with one element contain the prediction. It is also preferred that the forecast values to be in the original scale. We can split the training data set into tests sets and fit the model and generate a prediction for each element. A way that we can perform the forecast is to create the Arima model after each new observation is received. There should be a track keeper as a history that will be seeded with the training data.

The evolving variable of interest is regressed on its own lagged values. The regression error is a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The data values have been replaced with the difference between their values and the previous values. But we have to fit the data as well as possible.

The non-seasonal ARIMA models *parameters* are p: is the order of auto regressive model, d is the degree of difference q is the order of the moving-average model.

Conclusion

In general, time series can be really precise and interesting. The problem is that they are really complicated in finding the anomaly especially if we are referring to a real-time data-set. [38][39]

Below, we have the characteristics of the algorithms.

Algorithms	STM	ARIMA
Accuracy	Yes	Yes
Classification	Multi-class	Multi-class
Code Availability	Yes	Yes
Complexity	-	-
Comprehensibility	No	No
Control Changes	No	No
Handle Parameters	No	No
Incremental Learning	-	-
Noise	-	Yes
Number of Parameters	6(periodicity of personality, number of cycles through the inner loop, number of cycles through the outer loop, span in lags for the low-pass filter, smoothing parameters of the trend behavior)	3(order of auto regressive model, degree of difference, order of the moving average model)
Optimization	-	-
Training Data/ Quick Learning	Unsupervised/ Supervised	Supervised
Variables	One-Variable	One-Variable

1.1.9 Markov Chains

Markov models, just like Bayesian networks, neural networks or support vector machines use mathematical models in the prediction for anomaly detection techniques, so they can decide first the unknown quality of sequence data and after that they can build the prediction models. Referring to these models, we have about Bayesian that the attributes are independent to each other, which cannot be true in real life applications. About neural networks, they require a large number of parameters and because learning time is too long they may fail to achieve the purpose of learning. As for, support vector machines they are difficult to implement large scale training samples because it consumes a lot of memory and computing time. Markov model is widely used in sequence modeling especially because of its advantage that every event is analyzed. Many times, it is not effortless to break up data into contexts, especially when we have time-series and sequence modeling techniques which are extended in order to detect contextual anomalies in the data so we have to utilize the structure data. For time-series data, there are a lot of regression based techniques for time-series that have been developed. These techniques, have been extended to detect contextual anomalies in a set of sequences by modeling the regression as well as correlation between the sequences. One of the time-series anomaly detection is stationary auto-regressive process. The operation is that any observation is tested to be anomalous by comparing it with the covariance matrix of auto-regressive process and if the observation is not agreed with the modeled error then it is defined as anomaly. Another technique to detect a single anomaly in a sequence uses the divide and conquer approach and the one that has the higher complexity is anomaly.
The process stops when a single event is declared as anomaly in the sequence. This idea is extended in other areas by using Markov models in order to determine conditional probabilities for events that are based on history events. This whole transition of normal and anomalous is modeled using a Markov process.
This algorithm has a controllable false alarm rate for fast streaming temporal data. It learn the nominal attributes under possibly varying Markov statistics. Then, an anomaly is declared at a time instant if the observations are statistically sufficiently deviant. So, there is a multi-order Markov Chain. This approach takes a higher order Markov chain and multivariate sequences into account to produce several indicators of anomalies.
Usually, in Markov chain approaches, mostly utilize the short memory property of classical Markov models. Because of that we have two assumptions, that the state probability distribution of time t is only related to the state of time t and the transformation from the state of time is time independent. In addition, the short memory property of classical Markov models in not applicable to the real-world data.

Dynamic Markov Chain

This is an approach based on a dynamic Markov model. It segments sequence data by a sliding window, in which there are defined states of data according to the value of the data and make a higher order of Markov order in order to balance the length of the memory property. In addition, it prevents the detected anomalies from impacting the building of the models and keep the anomaly detection continuously.
The dynamic Markov model that is presented balances the length of the memory property of the Markov models and keep the strong correlation between the memory and the current test data. The mentioned model, first use the sliding window to segment a sequence data and then the correlation analysis of data in the sliding window is used to find out a proper order of a Markov model.

The order of the Markov model is continuously updated with the sliding windows sliding to keep the relationship between the Markov model and current test data. Besides, when the current test data exceed the scope of the previously defined states, the states of data in the sliding window will be redefined and the model will be retrained to follow the changes of the sequence. In addition, at the same time in order to detect anomalies continuously and prevent anomaly points detected from infection to the building of the models an anomaly substitution strategy exists.

Functionality:

The steps are the following. First of all, it has to state a definition that should be clear because it is demanded by the nature of Markov model. After, it should establish an n-order Markov model. Then, there should be a detection and retraining phase. Finally, there is an anomaly substitution strategy, in order to ensure that the detected anomalies cannot infect the detection results of the rest data points in the future. The detection can be carried out dynamically and continuously by substituting the detected anomalous points and sliding the window step by step.

Pseudocode:

```
segment the sequence X(T) using a sliding window Wt₁
define the states of data in the sliding window with the number of states N
use the Pearson correlation analysis approach
determine the order of the higher order Markov model n in the sliding window;
```

```
Establish an n-order Markov model λ(n)={S,Q,P₁,P₂,…, Pₙ} in the slinding window;
```

Remarks:

This approach balances the length of the memory property of higher order Markov model are analyzed by repeatedly utilizing the Pearson correlation that is used so that a proper order of the Markov model in a sliding window should be found. In addition, this approach maintains the establish model reliable and detect the tendency sequences by dynamically defining the states of data and training the models in the sliding window.

Hidden Markov Anomaly Detection

There is another anomaly detection methodology for data with latent dependency structure, the hidden Markov Anomaly Detection that actually extends the regular one-class support vector machine. Hidden Markov anomaly detection (HMAD) is defined as the latent OC-SVM algorithm with the hidden Markov joint feature map.

Pseudocode:

```
input data $x_1,…, x_n$
put t=0 and initialize $w^t$
repeat
t:=t+1
for i=1,…, n do
$z^t_i$ := $\text{argmax}_{z\epsilon z}$ h$w^{t-1}$, $\Psi(x_i, z)$>i+ $\delta(z)$
end for

let $(w^t, \rho^t)$ be the optimal arguments when solving
one-class SVM with $\varphi(x_i) := \Psi(x_i, z^t{}_i)$
until ∀ i = 1,...,n : $z^t_i = z^{t-1}_i$
```

Remarks:

In order to optimize the approach, we have a parameter v which can be used to control the number of outliers in a model. From different researches this approach has higher anomaly detection performance than the regular one-class SVM. This approaches to learning-based anomaly detection represent in a linear way the class and used to predict the anomaly score of new and unseen inputs. The method that is used for hidden Markov chain has a parameter v that controls the fraction of the outliers and enjoys deep theoretical guarantees. It proves that a large deviation bound holds on the generalization error. This method also outputs models as common in label sequence learning, to facilitate the detection of anomalous sequences where changes follow a hidden Markov model. This is better than the other methods that treat each position independently (like SVM).

Referring to the parameters it has v that controls the number of outliers and because of that it can control how optimal or not the algorithm can be because of the boundary we can set.

So, the hidden Markov model combined ideas from structured output learning and kernel-based anomaly detection. It is used in real data from the domains of bioinformatics and computational sustainability.

Conclusion

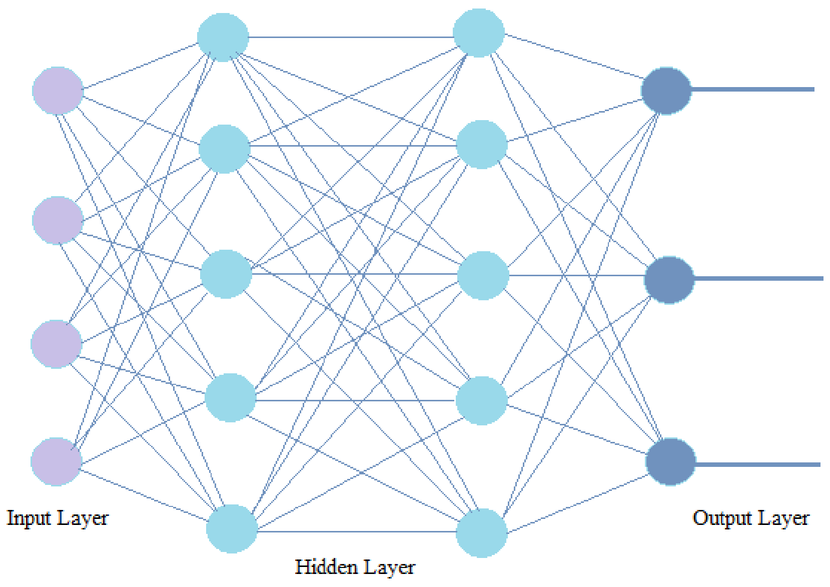
Markov chain techniques are widely accepted because of being simple and with a few parameters. However, the short memory property of a classical Markov model ignores the interaction among data and the long memory property of a higher order Markov model clouds the relationship between the previous data and current test data reduces the reliability of the model. Besides both of these models cannot be successfully describe the sequences changing with a tendency. But in addition it is difficult and complicated to manage this system.

There is a table with the comparison of the mentioned algorithms.

Algorithms	**Dynamic Markov Chain**	**Hidden Markov**
Accuracy	Yes	Yes
Classification	One-class	One-class
Code Availability	Yes	Yes
Complexity	O(nlogn)	O(nlogn)
Comprehensibility	No	No
Control Changes	No	No
Handle Parameters	No	No
Incremental Learning	-	-
Noise	-	-
Number of Parameters	n(n-1)	n(n-1)
Optimization	-	-
Training Data/ Quick Learning	Unsupervised/ Supervised	Unsupervised/Super vised
Variables	Multi-variable	Multi-variable

1.1.10 Artificial Neural Networks (ANNs)

ANNs are computing systems which are inspired by the human (animal) nervous system. It consists of nerve cells that exist in the human brain. They are able to learn and improve their way of considering without task-specific programming or without any previous knowledge. ANNs are weighted directed graphs in which nodes are the artificial neurons and the edges are the connections (which are weighted) between the outputs and the inputs.
In general, the signal at the connection is a real value and the output is calculated by a non-linear function of the sum of its inputs. The values are adjusted by the learning proceed and the neurons are organized in layers. The signals begin from the first (input) to the last (output) layer [40].



(1.6) ANNs architecture

Referring to architecture as mentioned these networks have many artificial neurons called 'units' and they are in a series of layers. A type of method based in a different type of architecture will be presented (Replicator Neural Network).
This technique can be applied to both *multi-class* and *one-class* categories. In multi-class we have two steps. The first one consists of training the neural network learning from the different normal classes, and after that, on the second step, the wanted test instance is used as an input to the neural network. Based on whether the neural network will accept the test input or not, we are able to reach to a decision [3]. Also, can be both *supervised* and *unsupervised*. The learning data sets that are used in ANNs are training (used for learning), validation (used to tune) and test (used to assess the performance).
ANNs have different uses: *classification* (where it is trained to classify the given data into predefined classes), *prediction* (where the expected output is produced) and *clustering* (where an attribute from the data is identified and then classified into

categories) [41].
Below, we will introduce some of the ANNs algorithms.

Gradient Descent Algorithm

This algorithm is a first-order iterative optimization algorithm for finding the minimum of a function. It is repeated many times so that the local minimum will be computed with accuracy. This algorithm as will be explained below is a forward-backward algorithm [44].
Functionality:
The first step that this algorithm has to make it to initialize the weights with random values and calculate Error (SSE). After that it calculates the gradient and when the weights maybe are changed by a small value compare them to their original randomly initialized value. With this we move the values in which the SSE can be minimized. When it finishes this step, it adjust the weights with gradients to reach the optimal values where SSE is minimized as mentioned. With these new weights we predict and calculate the new SSE. Finally, we repeat again the previous steps till further adjustment to weights stops reduce the Error [43].
Input:
random
value
Output:
faster step
Pseudocode
```
# From calculation, it is expected that the local minimum occurs at x=9/4
cur_x = 6 #the algorithm starts at x=6
gamma = 0.01 # step size multiplier
precision = 0.00001
previous_step_size = cur_x
df = lambda x: 4 * x**3-9*x**2
while previous_step_size > precision:
prev_x = cur_x
cur_x += -gamma * df(prev_x)
previous_step_size = abs(cur_x - prev_x)
print("The local minimum occurs at %f" % cur_x)
```

Remarks:
The Gradient Descent algorithm has many challenged to face in order to work properly. First of all, that data challenges. They should be well defined and if the data are not organized then it has not a well optimized problem. Then we have the Gradient Challenges. If the execution is not done properly it may lead to problems like vanishing gradient. Last but not least it has to face the implementation challenges because it is significant to look at the resource utilization by networks.
This procedure is computationally expensive. In order to find the optimal step size on every iteration because it is time consuming we combine it with a line search. This algorithm can be extended constraints. This happens by including a projection onto the set of constraints. This is another reason that makes it

forward-backward algorithm. In order to make this algorithm fast there have been made a lot of extensions. As for the *complexity* it is O(1/n^2) [43] [44].
The algorithm is used for *supervised* method. It is able to find in an actual output and a target output the difference error. After that in order to minimize this error it changes the weights at the connections. Referring to the *parameters* it has the hidden weights, the output weights, the hidden bias and the output bias [41].

Back propagation

This algorithm is used to calculate the error contribution of each neuron. It can also be called automatic differentiation. As mentioned it is also used by the gradient descent optimization algorithm to adjust the weight of the neurons. It is an extension of gradient based delta learning rule. When we find the previously mentioned error, through a hidden layer the error is propagated backward from output to input. We use this method for multi-layer networks. We can also refer to it as the result of the playout that is propagated up [41] [45].
Functionality:
About the first phase which is the propagation, we have the following steps. First of all, we generate the output values and calculate the cost. Then, we have the propagation of the output back through the network (with the use of the pattern target) so that we can generate deltas of all output and hidden neurons. Then, about the second phase which is the weight update, we have the following steps. The weight's output delta and input activation are multiplied in order to find the gradient of the weight. Then a ration of the weight's gradients is subtracted. This ration influences the speed and quality of learning. The greater the ratio, the faster the neuron trains and the opposite. Then the weight must be updated in the opposite direction, "descending" the gradient [45].
Input: 3 layers, 1 hidden
Output: network
Pseudocode:

```
cur_x = 6 #the algorithm starts at x=6
initialize network weights (often small random values)
do for each training example named ex
prediction = neural-net-output (network, ex)
actual = teacher-output (ex)
compute error (prediction - actual) at the output units
compute for all weights from hidden layer to output layer
compute for all weights from input layer to hidden layer backward pass
continued
update network weights
until all examples classifies correctly or another stopping criterion
satisfied
return the network;
```

Remarks:
It requires a desired output for each input. It is more used with *supervised* learning method but also for *unsupervised*. It is more used to trained neural networks with more than one hidden layer [6].
Referring to the cost of the function, it is represented by the map values of one or

more variables. In other words, it calculates the difference between the network output and the expected output after a case propagates through the network. As for the *complexity* it is O (n^3).
As we described it repeats steps the propagation and weight update. The error that is resulted after the cost function is propagated back through the network until each neuron has an associated error value that reflects its contribution to the original output. The learning can be *stochastic*. Each input creates a weight adjustment but it introduces "noise" into the gradient descent process using the local gradient calculated from one data point and it reduces the chance of the network getting stack [6].

Replicator Neural Networks (RNN)

Replicator Neural Networks is used for *one-class* anomaly detection. It is considered as multi-layer feed forward neural network and referring to the data that the input vectors are also used as the output vectors while the RNN attempts to reproduce the input patterns in the output. The aim of this is to minimize the mean reconstruction error for all training patterns. Because of that, common patterns tend to be well reproduced by the trained RNN so their patterns that represent outliers are less well reproduced and have a high reconstruction error.
The function of the RNN is to reproduce the input data pattern at the output layer with error minimized through training. Both input and output layers have n units, corresponding to the number of features of the training data. The number of units in the three hidden layers are chosen experimentally to minimize the average reconstruction error across all training patterns [42].
Remarks:
RNN have a lot of benefits. It can be used in network intrusion detection or at the Wisconsin Breast cancer dataset. An RNN is trained from a sampled data set in order to build a model that predicts the given data. The usage of this classifier is to develop a score for outlyingness (the Outlier Factor) so that the resulted and trained RNN can be applied to the whole data set. After that the aim will be to give a quantitative measure of the outlyingness based on the reconstruction error
About the OF, we can measure the outlyingness, the treatment of *categorical* variables, and the sampling scheme for large data sets. As for the *parameters* we define the Outlier Factor of the i data record OF_i as our measure of outlyingness. It is defined by the average reconstruction error over all feature where n is the number of features. The OF is evaluated for all data records using the trained RNN. When we have data-sets with *categorical variables*, we have to split the data-set into a number of subsets. Every subset has to correspond to a set of particular values of the categorical variables. For example, if we have two categorical variables and each with two categories, we have to split the data set into four subsets so we can achieve a unique combination of the two categorical variables. After that, we have to train RNN for each subset. Although a better method is being developed. After this step, we have to sample and train a portion of data (randomly or by selecting) each subset for every record in order to train the RNN.
Finally, we apply the trained now RNN so we can calculate the Of_i for all the data

points and make a declaration about being anomaly or not [42] [46].

Conclusion

ANNs, are depended from three aspects. The functions of input, the architecture and the weight of the connections. They can be very rapid and precise and also are able to handle the parameters. They are popular enough and easy to handle, so they are also available.
The main goal of the ANN approach is to solve the problems as close as the human brain works. This method is used for many applications like computer vision, speech recognition, machine translation, social network, medical diagnosis, video games. [40]
The input layer it the one that receives input from the outside world on which learn. The output layer is the one that contains units that respond to the information about how it is learned any task. And about the hidden layer, are the units between the input and output and it has to make the transformation between them.
By this way, the machines can study, observe the environment, make decisions, data representation, etc [41].
The table includes the comparison between these algorithms.

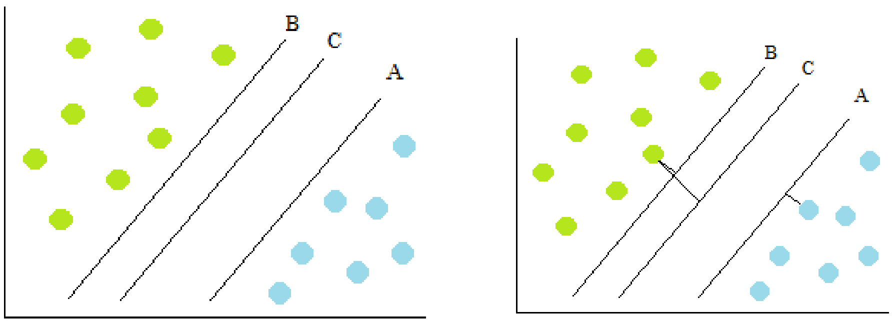
Algorithms	Gradient Descent	Back propagation	Replicator Neural Networks (RNN)
Accuracy	Yes	Yes	Yes
Classification	One-class/ Multi-class	One-class/ Multi-class	One-class
Code Availability	No	Yes	Yes
Complexity	$O(1/n^2)$	$O(n^3)$	O(n)
Comprehensibility	No	No	No
Control Changes	-	-	-
Handle Parameters	No (Categorical)	No (Categorical)	No (Categorical)
Incremental Learning	-	-	-
Noise	-	-	-
Number of Parameters	4 (hidden weights, the output weights, the hidden bias and the output bias)	Free	3(outlier factor of the I data record OF, average reconstruction error, n is the number of features)
Optimization	Combination with linear search	-	Outlier Factor
Training Data/ Quick Learning	Unsupervised/ Supervised	Unsupervised/Supervised	Unsupervised/Supervised
Variables	One-variable	One-variable	One-variable

1.1.11 SVM (Support Vector Machines) Algorithm

In machine learning, support vector machines (SVMs) are (usually) *supervised* learning models with associated learning algorithms that analyze data used for classification and regression analysis (it has a set of training instances and each one of them belongs to one or more other categories). The algorithm builds a model that can assign new instances to one category or to the other, making it a *non-probabilistic* binary linear classifier. So, a SVM model is a representation of the instances as points in space that are mapped in a way that the examples of the separate categories are divided by a gap (as wide as possible). Then, when new examples are mapped into at the same space, are predicted to belong to a category based on which side of the gap they fall. In addition, SVMs can perform *linear* and *non-linear* classification (with kernel trick). It can also use both *supervised* and *unsupervised* learning. In unsupervised learning approach (with not labeled data) attempts to find natural clustering of the data to groups and then map the new data to these formed groups.

Functionality:

Referring to the implementation, we plot each data item as a point in n-dimensional space (where n is number of features we have) with the value of each feature being the value of a particular coordinate. It is a frontier that segregates the two classes. (hyper-plane/line). Then we perform classification by finding the hyper-plane that differentiate the two classes very well. The problem is in identifying the right hyper-plane. There will be presented an example for better understanding the functionality. In this example we have to identify the right hyper-plane. We have to choose the best one of A, B or C. In order to identify the right hyper-plane we are making some steps. The decision will be with the Margin. Margin is the distance that we have if we maximize it between the nearest data point (or class) and hyper-plane.



(1.7) Margin for hyperplane

So, we can see that the margin for hyper-plane C is high if we compare it to A or B. Because of that we consider as the right hyper-plane the C. In addition, because we want to avoid the chance of miss-classification we select the hyper-plane with higher margin is robustness. In SVM is easy to have a linear hyper-plane between two classes but we can also add manually hyper-plane with a technique called kernel trick. These have low dimensional input space and transform it to a higher

dimensional space. In other words, it converts not separable problem to a separable with functions called kernels. SVMs are mostly useful in non-linear separation problems.
Input: X (predictor), Y (target) for training data set, x_test (predictor) of test_dataset
Output: table
Pseudocode:

```
#Create SVM classification object
model = svm.svc(kernel='linear', c=1, gamma=1)
Train the model using the training sets;
check score;
model.fit(X, y)
model.score(X, y)
Predict output;
predicted model=model.predict(x_test)
Import Library;
Train← read.csv(file.choose())
Test← read.csv(file.choose())for A;
#create model
model←  svm  (Target~predictor,  data=train,  kernel='linear',  gamma=0.2,
cost=100)
#Predict Output;
preds← predict (model, Test)
table(preds)
```

Remarks:
SVMs have both advantages and disadvantages. First of all, it can has perfect performance if we have set clear margin of separation. In addition, it is effective in high dimensional spaces and in cases where the number of dimensions are greater than the number of samples. Also, it is memory efficient because it uses a subset of training points in the decision functions (support vectors). On the other hand, there is no good performance when there are large data because of the high training time or when the data set has more noise.
In general, SVMs works better on *small data-sets*, but on them it can be very strong and powerful in building models. Besides, *overfitting* is a SVM's problem. Because every point is a support vector there is too much freedom to bend to fit the training data. SVMs has an automatic way to avoid it but, if one over-fits, then tend to make errors in new data.
Support Vector Machines (SVMs) have been applied to anomaly detection in *one-class* setting, in other words it learns from training data instances (a boundary). If a test instance is included in the training data label then it is defined as normal, otherwise as anomalous. We have already mentioned that one-class classification based anomaly detection techniques presume that all training instances have only one-class label. There are also Robust Support Vector Machines (RSVM) that robust the presence of anomalies in the training data (it is used for intrusion detection). There is another approach of SVMs. This is the *multi-class*. Multi-class SVMs aims to assign labels to instances by using support vector machines, but we will focus on one-class approach. The clustering algorithm that is an improvement of support vector machines is called support vector clustering. It is used in industrial applications either in unsupervised or semi-supervised approach. SVMs are often used in real world. They are helpful in text and hypertext categorization, in

classification of images, in the recognition of hand-written characters, in biological and other sciences. Support vector machines (SVMs) have been considered for real-life machine learning applications referring to security. This field concerns in modern industrial networks, which is also used in critical applications in various fields. SVM can be a powerful tool for realizing a self-configuring monitoring for industrial infrastructures regarding attacks as kind of anomalies.
Referring to the *parameters*, in order to have more effectively learning machine, we have to adjust the parameters correctly. These parameters are: C (cost), kernel, degree, gamma, coef, shrinking, probability, cache size, class weight, verbose, random state.
Finally, about the *complexity*, there are two complexities involved, the training time and the testing time. If we have linear SVMs, for the training time we must estimate the vector and the bias through a quadratic problem and about the test time prediction is linear in the number of features in the training data. If we have kernel SVMs, about the training data we must select the support vectors and for the test time the complexity is linear to the number of the support vectors and on the number of features. Our optimal solution involves the order of n^2 dot products, while solving the quadratic problem directly involves inverting the kernel matrix, which has complexity on the order of n^3, and n is the size of our training set.
There is a table below including the characteristics.

Algorithms	SVMs
Accuracy	Yes
Classification	Multi-class
Code Availability	Yes
Complexity	O(n^3)
Comprehensibility	No
Control Changes	-
Handle Parameters	Yes (Categorical/continuous)
Incremental Learning	No
Noise	-
Number of Parameters	11 (cost, kernel, gamma, coef, shrinking, probability, cache size, class weight, verbose, ransom state)
Optimization	Prunning
Training Data/ Quick Learning	Unsupervised/ Supervised
Variables	Multi-variable

1.1.12 Rule-Based

In a classical *supervised* learning problem, data instances that represent several classes are available for designing a classifier and a learning algorithm uses this information to discriminate between the classes. Rule-based anomaly detection makes rules so that a defined normal behavior of a system can be described and recognized. If there is no rule, the system is defined as anomalous. This technique can be applied in both *multi-class* and *one-class* settings. In the multi-class rule-based technique there are two steps. The first one is to learn the rules (from the training data) and use them with an algorithm. The second is to find for each test instance the best rule. The best rule has the best anomaly score of the test instance.
There have been proposed several rule-based techniques. Association rule mining has been used for one-class anomaly detection. These rules are generated from the data, from a categorical data set. In addition, a threshold is used in order to decide which rules have strong patterns and which not. There have been proposed a lot of anomaly detection algorithm for categorical data sets. There are two ways for building classification rules the direct and indirect. We will focus on the direct way on which the rules are extracted directly from the data.

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) Algorithm

This class implements a propositional rule learner, an optimized version of IREP, a very effective technique used in decision tree algorithms. As in IREP, the training data are split into a growing set and a pruning set. In such way the RIPPER algorithm is used.
Functionality:
The RIPPER algorithm uses the direct method in order to learn one rule. In other words, it begins the procedure by starting from an empty rule and then it add conjuncts as long as they improve the total information gaining. The algorithm through this procedure is learning. They stop learning when the rules can no longer cover any negative examples. The rules are built with accuracy = 1 (if possible) and then they prune the rule immediately using reduced *error pruning*. Referring to the *parameters,* the measure for pruning is: W(R) = (p-n)/(p+n), where p is the number of positive examples covered by the rule in the validation set, and n is the number of negative examples covered by the rule in the validation set. The operation of pruning starts from the last test added to the rule. There is a possibility that may create rules that cover some negative examples (accuracy < 1).
Input:
parameters:
Pos, Neg, k
Output: Rule
set

Pseudocode:

```
Ripper (Pos, Neg, k)

Rule Set ← LearnRuleSet(Pos, Neg)
For k times

~~RuleSet←Opti~~mizeRuleSet(RuleSet, Pos, Neg)

LearnRuleSet(Pos, Neg)
RuleSet←

DL← DescLen(RuleSet, Pos, Neg)

Repeat

Rule← LearnRule(Pos, Neg)

If DL_1<DL DL← DL’
Delete instances covered from Pos and Neg
Until Pos=
Return RuleSet
```

Remarks:

RIPPER is a rule-based learner that builds a set of rules that identify the classes while minimizing the amount of error. The error is defined by the number of training examples misclassified by the rules. When there is a *one-class* problem, the algorithm should choose one of the classes as positive class and the other as negative. The positive class is the one with the rules that should be learned and the other one, the negative class, should be defined as default. On the other hand, for *multi-class* problem the classes should be ordered according to increasing class prevalence. In other words, the rule set for smallest class should be learned first and then the rest should be treated as negative class. Then there should be a repetition with next smallest class defined as positive class. Finally, the time *complexity* of this algorithm is O (N*$\log^2$ N).

PRISM

PRISM is an algorithm that induces modular rules. It is based on ID3 but uses a different way to induce rules which so that many problems that are associated with decision trees can be avoided, because they are difficult to manipulate. On contrary, one of the features of rule-based expert systems is that the mentioned modularity of the rules typically enables a knowledge base to be easily updated or modified. It also provides a means for explanation. There is a requirement, therefore, that rules should be both modular and comprehensible, whether they are elicited from experts or automatically induced from examples. The algorithm refers to *multi-class* and the input that PRISM algorithm has is a training set as ordered sets of attribute values and each set is being terminated by a classification.

Functionality:

In the begging it has to calculate the probability of occurrence of the classification for each attribute-value pair. Then select the one for which there is a maximum and create a subset of the training set comprising all the instances which contain the selected. After you have to repeat until it contains only instances of class. Then, remove all instances covered by this rule from the training set and again repeat until all the instances of the class have been removed. When the rules for one classification have been induced, the training set is restored to its initial state and the algorithm is applied again to induce a set of rules covering the next classification. As the classifications are considered separately, their order of presentation is immaterial. If all instances are of the same classification then that classification is returned as the rule, and the algorithm terminates.

Remarks:

We mentioned that PRISM is based on ID3, but there is a major difference that PRISM concentrates on finding only relevant values of attributes, while ID3 is concerned with finding the attribute which is most relevant overall. PRISM must identify subsets of a specific class. This has the disadvantage of slightly increased computational effort, but the advantage of an output in the form of modular rules rather than a decision tree.

PRISM is based on the strategy which maximizes the information contributed by an attribute-value pair to knowing a particular classification. The accuracy of rules induced from an incomplete training set depends on the size of that training set. A rule will not be induced if there are no examples of it in the training set. In addition, an induced rule may be too general if there are no counter-examples to it in the training set. When PRISM is applied to a complete training set, the resulting set of rules can confidently be expected to be complete and correct. When the training set is incomplete, this confidence is reduced. The smaller the relative number of instances in the training set, the more likely it is that the rule set will contain errors.

If there are two or more rules describing a classification, PRISM tries to induce the most general rule first, because the more general a rule is then the less likely it is to reference an irrelevant attribute. In particular, there is the problem of which attribute or attribute value pair to choose when the results of the respective calculations indicate that there are two or more which are equal.

One Class-DS learning algorithm

In general, on-class learning algorithms are used when the training data belong only for one class (target class). The other data are called outliers. One Class-DS learning algorithm that used both ruled-based classification and greedy search on density because this approach is used to detect outliers or novelty in the data by using the method of density estimation techniques.

Functionality:

One Class-DS algorithm generates rules by performing greedy hill-climbing search. The parameters that uses are: Threshold, MinCoverTager, MinAttribute, MaxAttribute and they are heuristic that can generate rules only from one-class data. The MinAttribute and MaxAttribute are used for accepting the generated rules between this range that should be defined. Then, the selectors in the rule have to satisfy two conditions the MinCoverTarget or the minimum number of target instances and they have to summed-up value equal or larger than Threshold. We can calculate this value by multiplying the number of values a feature takes on by the number of times it appears in the target training data. The process of generating rules is repeated several times with different settings of the four parameters and finally we choose the best model from generating using sensitivity measure.

Input: Data set D with N classes (from 1 to N)

Output: Ruled set model

Pseudocode:

```
Build model;
Chose class I as a target class T
Denote all instances in class i as belonging to class T, and all other instances as belonging to class O-this forms data set D1
Divide data set D1 into k parts Pj (each part may include instances belonging to class T and/or class O)
For each part Pj
Combine (K-1) parts (except for part Pj) to form data set D2
Delete all instances of class O from D2 (becomes one-class problem)
Build model for the reduces data set D2 (now contains only class T instances)
Test on part Pj(may contain instances belonging to T and O)
Choose the best model and test it on D1
```

Remarks:

The main advantage that this algorithm has is that oit has high efficiency and robustness to missing data. If we use higher values for the mentioned parameters we have the generation of stronger rules that can predict *optimal* new instances with higher accuracy. If we use lower values, we have weaker rules. Finally, referring to the computational *complexity* of One Class-DS algorithm is approximately O(R*K*N*log(N)), where R is the number of rules in rules, N is the number of instances, and K is the number of attributes. To summarize, One Class-DS is simple, greedy, effective rule learner that generates a set of production rules. The rules are compact, easy to comprehend, and have accuracy on par with rules (models) generated by other one-class rule-based algorithms.

Conclusion:

To conclude, this technique has a lot of advantages. Because of the parameters, allows encapsulating knowledge and expansion of the expert system done in an easy way. In addition, it is easy to build explanation facilities. Also, they are precise and they have low complexity. However, they are fond of noise which can make difficult of the results in the anomaly detection.

A table is presented with the mentioned characteristics.

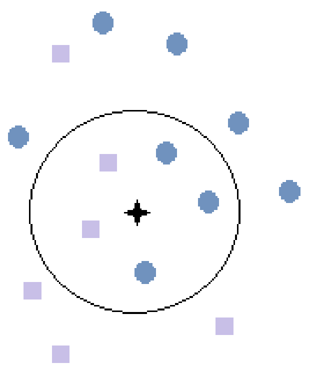
Algorithms	RIPPER	PRISM	One Class-DS Learning
Accuracy	Yes	Yes	Yes
Classification	One-class/ Multi-class	Multi-class	One-class
Code Availability	Yes	No	Yes
Complexity	O(nlog n^2n)	-	O(r*k*n*log(n))
Comprehensibility	Yes	Yes	Yes
Control Changes	-	-	-
Handle Parameters	Yes	Yes	Yes
Incremental Learning	No	No	No
Noise	No	No	No
Number of Parameters	3 (pos, neg, k)	-	4 (threshold, mincovertager, minattribute, maxattribute)
Optimization	Prunning	Prunning	-
Training Data/ Quick Learning	Supervised	Supervised	Supervised
Variables	Multi-variable	Multi-variable	Multi-variable

1.1.13 Nearest Neighbor-Based

This method is used for many anomaly detection techniques which are ruled by assumptions. When we have normal data instances occur in dense neighborhoods but when we have anomalies we have the opposite result which is being far from their neighbors.
In this technique we have a distance between the data instances and this can be computed in different ways (e.g. when we have continuous attributes we use the Euclidean distance). It can accept continuous or categorical attributes. In addition it can work with multivariate data instances. They are similar to clustering based technique, both of them do not require strictly metric. We can have two groups, the techniques that use the distance of a data instance to its k th nearest neighbor as the anomaly score, and the one that compute the relative density of each data instance to compute its anomaly score. [47]
In general, the algorithm uses the following steps: first of all it has to start on an arbitrarily vertex as current vertex and then to find the shortest edge connecting to the current vertex and an non visited vertex V. After that we set a current vertex to V. Then we mark the V as visited. If all the vertices in domain are visited we terminate and we repeat. The final output of the algorithm is the sequence of the visited vertices. [48]
Some algorithms will be introduced and compared:

K th Nearest Neighbor (k-NN) Algorithm using distance

The k-NN classification is used as a base classifier in many pattern classification problems. It is based on *distance*. We measure the distance between the test data and each of the training data to decide the final classification output. Is a non-parametric classifier.
Because it is a distance metric in k-NN we use Euclidean distance function as mentioned, but also there are more functions like Ch square, cosine etc. The chosen distance function can affect the classification accuracy of the k-NN classifier. E.g. for medical data-sets that has categorical data Chi square is preferred or when we have *categorical and numerica*l we prefer the Euclidean distance. It depends on the data set. [49]
Functionality:
This algorithm is a *non-parametric* classifier. When we want to have *unsupervised* instances the k- NN calculates the distances between the points and the points in the training data set. As mentioned the most usual function to find out this distance id the Euclidean. After that it assigns the point to the class among its k nearest neighbor, where k is an integer. On the following figure, * is the point. If k=1 the point belongs to the square class and if k=5 to the small circle class which is the majority class of five nearest points.



(1.8) Non-parametric Classifier

This algorithm does not require training stage. The computation is on-line searching for nearest neighbors of a given testing example. [49]

Input: Let (X_i, C_i) where i = 1, 2……., n be data points.

Output: labels for X_i for each i. Pseudocode:

```
Calculate "d(x, x_i)" i =1, 2, …..,n;
where d denotes the Euclidean distance between the points
Arrange the calculated n Euclidean distances in non-decreasing order
Let k be a +ve integer,
take the first k distances from this sorted list
Find those k-points corresponding to these k-distances.
Let k_i denotes the number of points belonging to the i^th class among k points
Let i.e. k ≥ 0
If k_i>k_j  i ≠ j then put x in class i.
```

Remarks:

In general, Knn is a simple algorithm and rapid for small training data-sets. It does not need any prior knowledge about the data in the training set and even id a new training pattern is added to the existing no training is required. On the other hand, if the training set is big then it takes a lot of time. For every test data the distance is computed between the test data and the training data which requires even more time. [4]

The anomaly score of the data instance is the distance to its k^{th} nearest neighbor from the given data set. This technique has three ways of functioning. The first modifies the definition so that we can have the anomaly score of the data instance. The second used the different distance to handle different data type. And the third focuses in improving the complexity in the basic technique.

The *complexity* is $O(n^2)$ where n is the data size. Another way to compute the anomaly score is to count the number of the nearest neighbors that are no more than the distance is given. Some techniques use pruning to search space to have better complexity. They ignore instances that are not anomalous or they focus on instances that are most likely to be anomalous. With this the complexity becomes

O(MN), where M is the sample size chosen. [47]

Kth Nearest Neighbor (k-NN) Algorithm using density

This technique is the same as the previous one but it estimates the density of the neighborhood of each data instance. An instance with low density is defined as anomalous, and the opposite are defined as normal. [47]
Functionality:
When we have a data instance, this distance is equal to the kth nearest neighbor. This distance of the kth nearest neighbor can be view as an estimate of the inverse of the density of the instance in the data set and the basic nearest neighbor-based technique can be considered as *density*-based anomaly detection technique. [47]
Remarks:
Density-based techniques do not have a good performance if the data has a lot of different densities. In order to face that there are a lot of different techniques. One of them is the Local Outlier Factor (LOF). When we have a data instance the LOF score is equal to ratio of average local density of the k nearest neighbors of the instance and the local density of the data instance itself. If we want to find the local density. The authors first find the radius of the smallest hyper-sphere centered at the data instance that contains its k nearest neighbors. At the normal instance, the density will be similar to the one of its neighbors. For the anomalous will be lower. Again, referring to the *complexity* will be O (N^2), where N is the data size.
Another technique is the Connectivity-based Outlier Factor (COF). In COF, the neighborhood for an instance is computed in an incremental mode. So, the difference between them is the way that the k neighborhood for an instance is computed. In COF, the closest instance to the given instance is the one with what we start. Then the next instance is added to the neighborhood set is such that its distance to the existing neighborhood set is minimum among all remaining data instances. The distance that is between an instance and a set of instances is defined as the minimum distance between the given instances. The neighborhood is growing like this until it gets the size k. The anomaly score is computed as LOF. Referring to the *complexity* it is again O(N^2).
The nearest neighbor-based technique refer to *unsupervised* learning mostly. However, it has some problems. One of them is that if the data has not enough close neighbor the technique might fail to label correctly. Also, for semisupervised techniques, if we do not have enough similar normal instances in the training data, the false positive rate is high.[47]

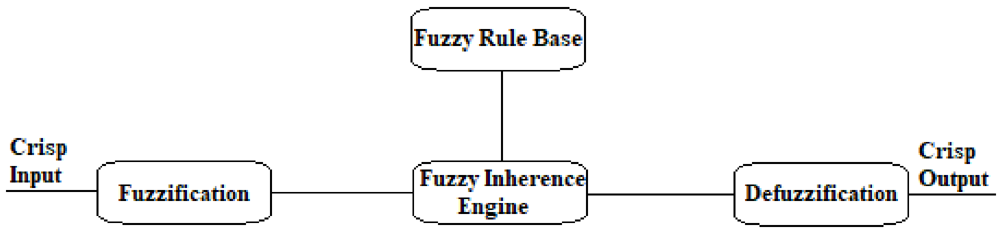
Conclusion

These are easy and quick algorithms, but because they have greedy nature sometimes cannot have the optimal solution and miss the shorter route. [48]
There is a table presented:

Algorithms	Kth Nearest Neighbor (k-NN) Algorithm using distance	Kth Nearest Neighbor (k-NN) Algorithm using density
Accuracy	No	No
Classification	Multi-class	Multi-class
Code Availability	Yes	No
Complexity	O((n^2)	O((n^2)
Comprehensibility	-	-
Control Changes	-	-
Handle Parameters	Categorical	Categorical
Incremental Learning	Yes	Yes
Noise	-	-
Number of Parameters	Non-parametric	Non-parametric
Optimization	Non optima solution	Non optimal solutions
Training Data/ Quick Learning	Unsupervised	Semi-supervised
Variables	Multi-variable	Multi-variable

1.1.14 Fuzzy Logic

Fuzzy logic was considered as a tool for linguistic uncertainty and vagueness ubiquitous in the imprecise meaning of words. It has four main parts, the input fuzzification, the fuzzy inference engine, the fuzzy rule base and the output defuzzification. [51]



(1.9) Fuzzy Logic System

Fuzzy logic algorithms can be used on sensors in order to find any anomaly. The cyber sensor is able to create a secure zone around the control system. After that, the learning algorithm is building a fuzzy rule base that includes the previously seen normal network behavioral patterns. The mentioned fuzzy rule base is made from the stream of incoming packets. It also might uses the nearest neighbor clustering algorithm as we will see below. Firstly, a crisp set of input data are gathered and converted to a fuzzy set using fuzzy linguistic variables, fuzzy linguistic terms and membership functions. This step is known as fuzzification. Afterwards, an inference is made based on a set of rules. Lastly, the resulting fuzzy output is mapped to a crisp output using the membership functions, in the defuzzification step. [52]

Pseudocode:

```
Define the linguistic variables and terms (initialization)
Construct the membership functions (initialization)
Construct the rule base (initialization)
Conver crisp input data to fuzzy values using the membership
functions(fuzzification)
Evaluate the rules in the rule base (inference)
Combine the results of each rule (inference)
Convert the output data to non-fuzzy values (defuzzification)
```

Fuzzy Logic Algorithm

At first we use a learning algorithm for the fuzzy logic anomaly based anomaly detection. We will use a low-cost online rule extraction technique and the algorithm learns directly from the stream of the incoming packets. So it is unsupervised. Because of that there is memory problem for storing the information. The final normal network behavior model is composed of a set of fuzzy rules. The algorithm maintains additional information about the spread of data points associated with each cluster throughout the clustering process.

When the rule extraction phase of the learning process is finalized the learning algorithm that we have mention maintains a final set of clusters that describe the normal communication behavioral patterns observed in the provided training data. Finally, each cluster is converted into a fuzzy logic rule. Each fuzzy rule describes a particular region of a multi-dimensional input to space to the class of normal behavior. A dimensional cluster is transformed into fuzzy rule.

The algorithm extracts fuzzy rules using an adapted version of the online nearest neighbor clustering algorithm directly to the stream of packets. [51]

Fuzzy Q-Learning Algorithm

Q-Learning Algorithm tries to optimize the fuzzy rule learning. Three fuzzy sets are defined as an input for the Q-Learning algorithm and they are called TBC. These are time response that has to do with the time duration of response between sensor nodes, the Buffer Size that relies on the size of the buffer for processor storage in sensor node by sending a huge number of fake messages and the count that is the number of connection to the same host at current connection in a past two second. As for the FLC output, it represents the action of the sink node. It is the detect confidence. This input and output variables shows us the traffic connection and how it changes. For this algorithm is not required to have a prior knowledge of data distribution, this is its main advantage. The method that uses to combine the classifiers is fuzzy rule base and q learning. However, it has low speed of detection comparing it to the other algorithms and because of that it can fail to high volume of traffic. The rules that are made are a logical combination of the time response, buffer size count signals while the consequent is a control pattern output. [53]

Conclusion

In general fuzzy logic, are simple rules. These rules can be used manually or automatic. They are also flexible, rapid and capable of handle big data without having any special prior knowledge.
There is a table presented with the mentioned characteristics:

Algorithms	Fuzzy Logic Algorithm	Fuzzy Q-Learning Algorithm
Accuracy	No	Yes
Classification	One-class/ Multi-class	Multi-class
Code Availability	Yes	No
Complexity	NP	NP
Comprehensibility	Yes	Yes
Control Changes	-	-
Handle Parameters	Yes	Yes
Incremental Learning	-	-
Noise	-	-
Number of Parameters	3(name of the linguistic variable, minimum and maximum values of the variable, membership function)	3(minimum and maximum values of the linguistic variable, name of the variable, membership functions)
Optimization	-	-
Training Data/ Quick Learning	Unsupervised	Unsupervised
Variables	Multi-variable	Multi-variable

1.1.15 Knowledge-based

These are techniques that are used based on the previous knowledge they have about some systems, events, attacks etc. It is usually used as IDES. IDES is a system for is computer intrusion detection. The recent behavior of a subject of a computer system is compared with observed behavior and any significant difference is considered anomalous. The main goal of IDES is to provide a system-

independent mechanism for real-time detection of security violations.
A general pseudocode about IDES so that we can realize the knowledge-based technique is the following:

```
Def rule programmed login attack
"Failed login attempts are being received at a rate too great to be from
normal human interaction"
With justification ("There were multiple login failures within second which
is an extremely short span of time. This may indicate a programmed login
attempt"
(-?t1 ?t2) states if there exists a login item error in marked called old
session
such that session id IS?sid and
such that error code is invalid' pass word and such that time' stamp is? T1
and there exists A login 'item' error unmarked called old' session
such that session 'id IS?sid and such that error code is invalid 'password
and such that time' stamp IS? T2 with (and(not (equal ?t1?t2))(equal (car?
t1)(car?t2)) (i(cadr?tw))(i(-(cadr?t1)(cadr?t2)login' delay)) and
such that session 'id IS?sid and
such that suspicion IS? Suspicion and there exists an active 'session called
session1 then execute (format t n "%Programmed Attack.") and mark old'
session and remember an active' session which inherits from session1 except
that suspicion equals (+? suspicion) and forger session1)
```

The main methods that knowledge-based uses is automated and Petri nets.

Knowledge-Based Temporal Abstraction (KBTA) method

This method is used for temporal data based on predefined domain knowledge. KBTA method has a temporal pattern mining method for anomaly detection. These patterns are then analyzed in order to identify number of normal patterns. By the term temporal-abstraction (TA) we mean the integration of time-stamped data in order to extract and summarize from these data. So, in general KBTA method is the computational framework for TA. It provides *automated* means from raw time- stamped data by using a domain-specific knowledge base.

Functionality:

As an input we have a set of raw, time-stamped events. And as for the output, we have a set of interval-based concept at the same or higher level of abstraction. (E.g. a period of two hours of high FTP connections while no user activity was detected. This indicates a Trojan horse that leaks information via FTP connection.) The domain of knowledge that is relevant to the TA task by using a temporal-abstraction ontology includes five KBTA entities (concepts, events, contexts, abstractions and patterns) and the relations among them. In general, we have six steps. First we apply the KBTA method on the raw data on order to derive basic abstractions. Then we discrete duration for each result (e.g. short, medium, long). Then we apply the temporal pattern mining on the discrete abstraction and we identify the temporal patterns that are repeated sufficiently enough and considered frequent.
Then we select a sub-set of normal behavior patterns that we have already identify before. After we add these patterns to domain of KBTA ontology. We apply the KBTA pattern matching mechanism on the test set in order to identify the

patterns that are selected. Finally, we detect abnormal time periods. Then, the temporal pattern mining algorithm is applied and new meaningful patterns are presented to the expert which decides if we can add it to the knowledge base.
Remarks:
KBTA is used for clinical analysis in medicine, for intrusion detection in computer-network security, fraud detection and decision-making in the financial domains, and analysis of information and decision-making in military intelligence. This happens when meaningful patterns are defined. Anomaly detection in KBTA can happen with two approaches: specifying abnormal pattern in the KBTA domain ontology and specifying temporal pattern representing normal behavior. The first approach feeds the domain ontology with known abnormal patterns, e.g. it can be a malicious behavior of a user. The domain expert has to knows and indicate a not normal event and the goal is to manual extract temporal patterns that leads or represents the event. As for the temporal pattern mining it is a similar approach but the assumption is that an indication of a known anomalous event is provided. When the algorithm is applied some temporal patterns are extracted. Such patterns represent the anomalous event. The challenge is to identify these significant patterns.

Internal Mining Algorithm

This algorithm is knowledge based. A fact here is defined as triplet: concept name, concept value, concept duration. A temporal pattern is defined recursively as if we have a single fact and a temporal pattern is called as an atomic pattern. The size of the temporal pattern is the number of atomic patterns in the temporal pattern. The algorithm is divided into two main steps. The first where the data are transforming into an item-list representation and the second that the temporal patterns are being mined.
Functionality:
The first step is to transform the database into an item-list representation. Then each tuple contains subject id, fact, item. In the first step we transform the database into an item list database where each fact is associated with a list of tuples. Then, mining temporal patterns is the step that we have. We start with a seed set that was found. We use the seed for generating large items. After that, we get a temporal coverage that is the sum of time intervals of all instances. The temporal coverage is satisfied if it is equal or greater than the temporal coverage support threshold. After that the algorithm determines temporal relations between the composite pattern and atomic pattern that have sufficient vertical support as well as horizontal/ temporal coverage support. The algorithm terminates when it cannot find any large k-items after the end of the current pass.

Petri Net

Petri net is one of several mathematical modeling languages so that we can describe distributed systems. In a Petri net the nodes represent transitions and places like conditions. It has directed arcs that places are pre- or post- conditions for which transitions are described. Petri nets offer a graphical notation for stepwise processes that include choice iteration, execution. They have precise mathematical definition of their execution semantics with a well-developed mathematical theory for process analysis.
They are applied in many areas like: office automation, work-flows, flexible, manufacturing, programming languages, protocols and networks, hardware structures, real-time systems, performance evaluation, operations research, embedded systems, defense systems, telecommunications, Internet, e-commerce and trading, railway networks, biological systems. The mathematical properties of Petri Nets are interesting and useful.
Functionality:
Arcs have capacity 1 by default. If other than 1, the capacity is marked on the arc. The places have infinite capacity by default. If other the 1 the capacity is marked on the arc. Places have infinite capacity by default and transitions have no capacity and cannot stroke tokens at all. With the rule that arcs can only connect places to transitions by using only Petri Nets.
A transition is enabled when the number of tokens in each of its inputs places is at least equal to the arc weight going from place to the transition. An enabled transition may fire at any time. When fired, the tokens in the input places are moved to output places, according to arc weights and place capacities. This results in a new marking of the net. In this nets are sequences that are obvious (things happen in order) and conflicts which are not so obvious.
Pseudocode:

```
Petri net synthesis
repeat/* Generation of pre-regions and label splitting*/
split:=false;
for each e belongs E do
e=generate_min_pre-regions(e);
if excitation_closure (e) then
split_labels(e);
split:=true;
end if
end for
until - split
find_irredundant_cover;
map_to_PN;
```

Conclusion

In general, this method has simple rules and its main advantage is the fact that the rules are easy. They are precise and not doing a lot of mistakes. But in order to achieve this there should be a lot of knowledge.
Below, it is presented a table with the mentioned characteristics:

Algorithms	KBTA	Internal Mining Algorithm	Petri Net
Accuracy	Yes	Yes	Yes
Classification	Multi-class	Multi-class	Multi-class
Code Availability	Yes	No	Yes
Complexity	NP	NP	NP
Comprehensibility	-	-	-
Control Changes	-	-	-
Handle Parameters	Continuous	Continuous	Continuous
Incremental Learning	Yes	Yes	Yes
Noise	-	-	-
Number of Parameters	-	-	-
Optimization	-	-	-
Training Data/ Quick Learning	Supervised/Unsupervised	Supervised/ Unsupervised	Supervised/Unsupervised
Variables	Multi-variable	Multi-variable	Multi-variable

Chapter 2:

2.1 Control System Categories

As mentioned in the introduction, Internet of Things (IoT) is a very fast-growing industry that can give the ability to the devices or the people to communicate with each other in every place or in every time or 24/7 service (the numerals for 24 hours a day, 7 days a week). Because of all this wide of the IoT which is used, we have to divide them into categories, which concern the sensors that integrate the devices, the CPU and the memory in order to define the complexity that is preferred to be used etc. Additionally, these categories had been made from some samples of different companies.

The type that every processor we should use for the connected devices can be influenced by the type of sensing needed for the target application. If some devices are able to perform a limited amount of data sets (e.g. temperature, humidity, pressure etc.) then more complicated systems should be needed in order to manage more multiple data sets (e.g. sound, video). Until recently, IoT devices have been reusing semiconductor products which are designed for other markets like the mobile applications. Nowadays, it has been recognizable that it is a different market area with different needs.

Two main "trends" are spread and used. The first one is about integrating the radio baseband on chip to become a standard practice because many IoT devices needs some form of wireless connectivity. If we have many communications standards, like Wi-Fi or Bluetooth etc. costs much power consumption which lead to affordable devices in order to have better battery life. The second, is to have a form of hardware security at the system so IoT devices could ensure users and companies that can feel secure with their sensitive data. As a result, we have conclude in five categories of IoT.

The first type refers to smart sensors and low-power M2M applications. Smart sensors are connected to microcontrollers that integrate analog interfaces for sensing. Their CPU performance requirements are between 50-100 DMIPS and connectivity includes energy-efficient standards such as low power Wi-Fi (802.11n or 802.11ah, Bluetooth Smart, low power cellular (e.g.Cat-M LTE), or 802.15 4-based protocols (e.g. ZigBee, Thread etc.).

The second type refers to connected audio and video. This category includes everything from Bluetooth-based speakers to home cinema systems. Depending on the target applications, the CPU needs to scale from a high-performance MCU delivering 300 to 500 DMIPS to an area-efficient application processor capable of around 1000 DMIPS. As for the connectivity, most semiconductor devices for wireless audio applications integrate Bluetooth (smart or not) and Wi-Fi.

As for the third and forth type, when we have to choose a specific flavor of the 802.11 protocol we have to check on the target application. A connected speaker like Amazon Echo Dot does not require more than 802.11n Wi-Fi, or when we have more complicated setups like multi-room home theater, 802.11ac is required in order to ensure that there is enough bandwidth to feed the entire network. When we have connected video segment it refers to Chromecast-type devices and connected of IP cameras which are used for streaming or recording video. IP cameras have a very basic user interface and do not require an embedded GPU capable of rendering 3D graphics. The Quanta WebRTC IP camera supports VP8 and H.264 encoding and integrate a MIPS CPU. Wireless displays are terminals that mirror or extend the surface of a more sophisticated smart device.
Finally, we have high-performance computing devices. The configuration of each chip it depends on the application.
One example analysis is with video analytic platforms is following. Video analytic platforms, usually process the data locally before sending the results to the cloud. These requires a very powerful and high integrate multimedia pipeline that includes a multicore GPU or specialized vision hardware capable of handling advanced algorithms. The connectivity probably will be covered by high-speed Wi-Fi such as 802.11 ac 2x2. When we have connected home devices with high-resolution displays and rich user interface has its balance between processing requirements, power consumption and cost. For example smart TV where the overall system costs, is one of the main drivers. In such cases, a more graphics-oriented GPU that is optimized for area efficiency is more suitable. The same can be said for multi core CPU or the radio processor. The cost can be reduced by not sourcing dedicated chips for each protocol. As for wearable is all about minimizing the power consumption. It depends on the target market and use case and wearable can be configured as a standalone device or tethered to a smartphone. This means that it reduces the range of wireless standard to a combination of blue tooth, Wi-Fi and using a CPU and GPU configuration that is optimized for energy efficient above anything else.
Finally, high-density compute nodes is a recent category. It includes applications with green computing-type systems like network and storage systems, cloud computing or large-scale data centers which must deliver ultra-high performance capabilities in a power efficient envelope. High-density compute nodes feature a heterogeneous CPU architecture. Usually can be defined by a more specialized architecture that integrates a many core CPU alongside other hardware accelerators (e.g. GPUs) used for compression, data filtering and other complex algorithms. A special particular feature of many core CPU is hardware multi reading. This allows the computational node to scale much better in terms of performance by turning on individual threads inside each of the CPUs before powering on multiple cores. This heterogeneous CPU computing solutions leads to significant gains in efficiency, both in terms of area and power.

IoT create new models and address global challenges that our society face. Typical problems are managing wireless communications bandwidth to balance control and data transfer needs, providing sufficient local processing resources to enable rich graphical user interfaces or video and image processing, controlling power consumption and efficiency and ensure appropriate levels of network and data security.

2.2 Selecting the best one

There are a lot of papers written about anomaly detection using machine learning, where SVM is the algorithm used as the classification algorithm. This happens because of the high accuracy it offers to real data-sets while having low complexity where we can see from the table above. Many of the papers can show accuracy in the numbers close to a 100% which would be the wanted outcome, but is almost impossible to achieve that in a real life scenario. In a real life scenario, where the attacks are so many and the threats just keeps developing and getting harder to detect. It would require a lot of training data in order for the machine learning could replace the normal IDS. Just even talking about numbers that many of the researchers have achieved in their experiments is a high achievement and is a big step for machine learning. In most of the papers listed in the background chapter the researchers have gotten results as high as 97%, which is a very good result. SVM can be used in many ways and there are several means explained in a very simple matter is that the hyper planes (which explained in the background chapter, both visually and in theory) are linear. There is also some other kernels that can be used like radial, polynomial etc.
The SVM algorithm is also fed with some other parameters that can e crucial when it comes to the accuracy of the classification. These are gamma and cost, which is two parameters that the calculation of the model that SVM uses to create the predictions. These two parameters have to be optimized before the classification process can be done. The gamma and cost parameters have different functions. The gama parameter or γ which is the mathematical representation of it is used to define how much influence on training example has on the model that is being created. Low values will mean that it does not have too much on effect each and every one of the training examples. We will see the results on a next chapter.

Chapter 3

3.1 Weka Tool

Waikato Environment for Knowledge Analysis (Weka) is a suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. It is free software licensed under the GNU General Public License. Weka contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions.

3.2 Results

3.2.1 Overview

In this chapter all of the results from the experiments will be presented. This chapter should give an overview of how the different experiments performed. The results from the different experiments will be gone through and explained thoroughly. There is a total of experiments as explained earlier. The experiments will be performed in the same order as planned and the results of each and every experiment will be analyzed.
How efficient the algorithms are in time and how much resources they use when running. When saying resources, what is actually meant is the CPU usage. There has been good accuracy on the algorithms, and the parameters has been optimized by testing with ranges of values before it was set as the best values.
Also, using multiclass classification, it can be more like a real life scenario. When saying real life scenario, what is meant is that there will be more than two classes the classifier can classify the data into. It will be a total of five different classes, which is mentioned earlier. For the algorithm this means that it has to recognize the patterns and classify it as one out of the five different classes. This will be a more complicated process then just to classify it as either true or false, as with binary classification. Even when using 60 % of the data set for training, it is not given that the all of the different attacks will be in the training data. Therefore it can be hard for the model to adapt to these attacks and learn them, while being tested.
There will be used some diagrams, tables and figures to explain flows and some logical examples. In this chapter it will be explained what has been done and how it became so. All of the different plans mentioned in the approach will be discussed and there will be an thorough explanation both visually and in text. There is some sort of standards that the data set must follow or be optimized for. So this standard will be utilized when design the project, including the implementation and the scripts that are made. WEKA reads in the data set from a .csv, .txt or .arff file which also is the case when using the data set that is used. A .csv and a .arff file looks very much a like and both can be read in by WEKA. The only difference between an .arff file and an

.csv file is that the .arff in this case also contains the different attribute names, or information that can be read in together with the data set in the same operation. Attributes names is the name of the features listed in the approach. When using the .txt file that comes with the NSL-KDD one have to manually set the column names (or feature names).
The data set is also the same as planned. There are still being used the same features in the data set, so saying that it is just like a network log would be incorrect.
As long as all of the values is continuous and the values are removed or changed into values. This goes for most of the algorithms using numeric calculations to do the predictions, there is also some that uses text for pattern recognition. But in both of the algorithms that is being used in this project there is factors or numeric values that should be in the data set for the algorithm to work properly. When using an algorithm there is often or in most cases a need to give the actual model that is going to made one or more parameters
Support Vector Machine (SVM) is also a very widely used algorithm when doing classification, this is mainly due to it is a good accuracy and that it can be used to do multi-calss classification

3.2.2 SVM Multiclass Classification

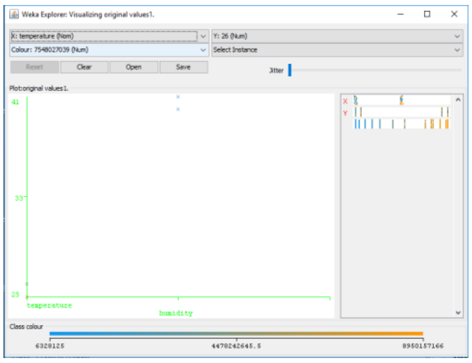
The experiment done is using the algorithm SVM to do multiclass classification. This will be done using supervised learning, a subcategory of machine learning as explained in the background chapter. It also shows how many it actually is in the data set of different types of attacks, that it should have recognized.
We conducted experiments with live, real-world data-sets with different characteristics, normal or anomaly. We mentioned above the information that these data-sets provide.
The first table has the nine normal results as monitored and compared to time. We can realize that there is nothing suspicious in this evaluation and that the signals provided are "healthy". We can also recall that the algorithm we choose to be evaluated, was compared with other through a range of types such as number of threads, memory and CPU usage, PLC, smart sensors and other characteristics needed for control systems evaluation as for the security needed. Health alerts are raised according to rules for detecting suspicious results as for the anomalous data-set. These faults were detected by rule-based mechanism that the detection technique has.
The techniques that have been applied are the same with the same choice of parameters. But because of the different nature of results, each has different results. We applied the same techniques to all services, using the same choice of parameters. Yet, due to their different nature, we discuss the results for each scenario separately.

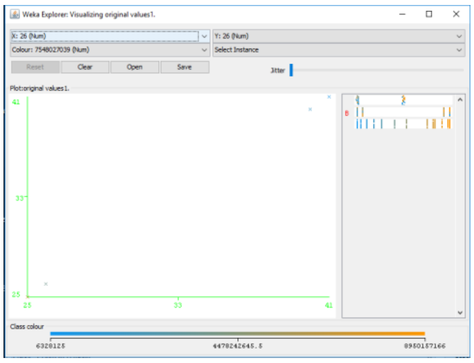
Precision

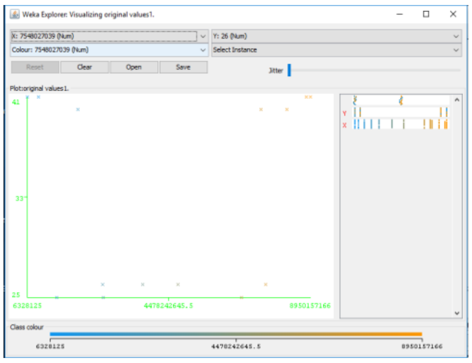
This detects any performance of the different versions if there is something as an incidence and for example if the data-sets are figured very close to each other they are reported as failures. The methods were able to detect and predict machine failures; therefore, latent faults do exist in this service as well, albeit to a lesser extent.

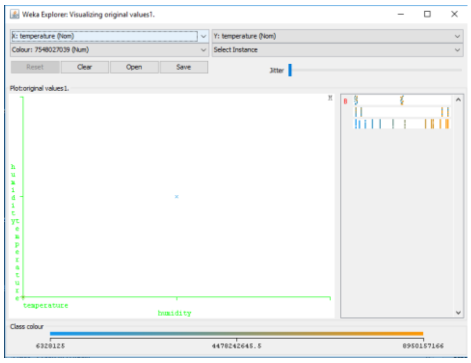
3.2.3 The Results

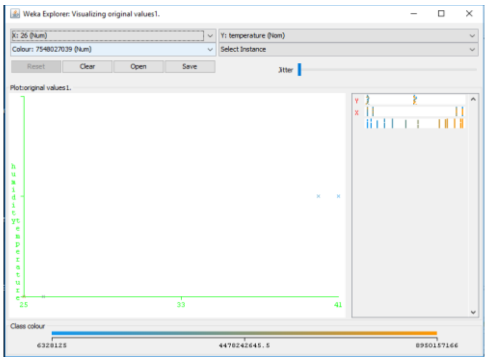
The results are the below. As for the normal:

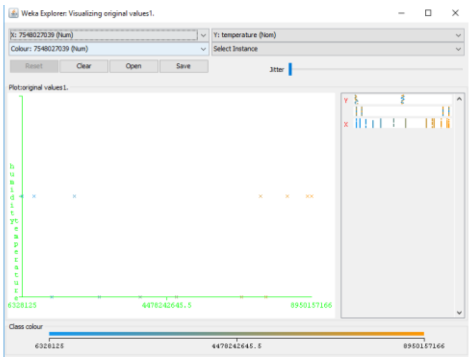

Weka Explorer: Visualizing original values1.
X: temperature (Nom) Y: 26 (Num)
Colour: 7548027039 (Num) Select Instance

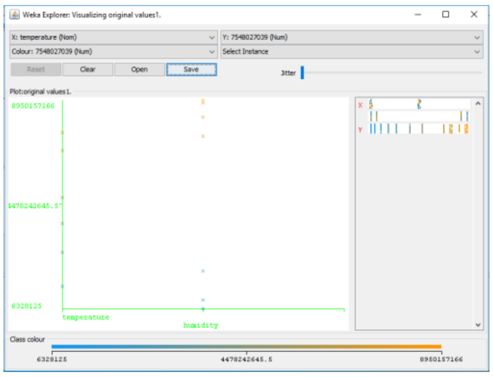

Weka Explorer: Visualizing original values1.
X: 26 (Num) Y: 26 (Num)
Colour: 7548027039 (Num) Select Instance


Weka Explorer: Visualizing original values1.
X: 7548027039 (Num) Y: 26 (Num)
Colour: 7548027039 (Num) Select Instance


Weka Explorer: Visualizing original values1.
X: temperature (Nom) Y: temperature (Nom)
Colour: 7548027039 (Num) Select Instance

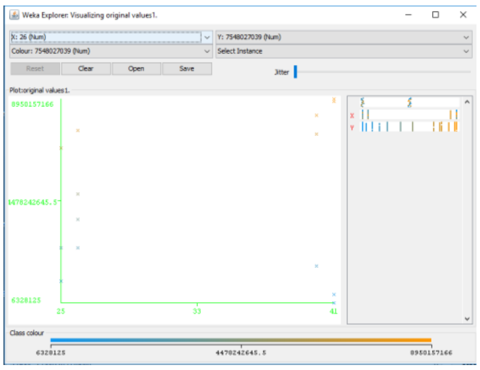

Weka Explorer: Visualizing original values1.
X: 26 (Num) Y: temperature (Nom)
Colour: 7548027039 (Num) Select Instance


Weka Explorer: Visualizing original values1.
X: 7548027039 (Num) Y: temperature (Nom)
Colour: 7548027039 (Num) Select Instance


Weka Explorer: Visualizing original values1.

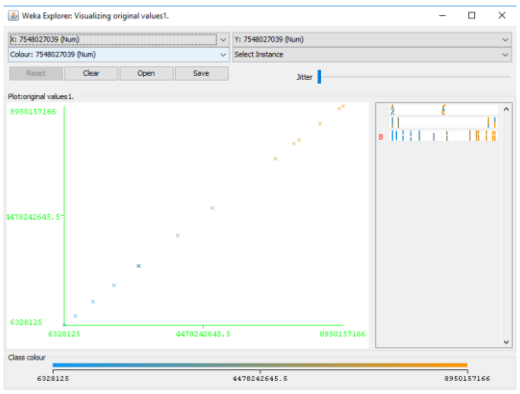
X: temperature (Nom)
Y: 7548027039 (Num)
Colour: 7548027039 (Num)
Select Instance

Plot:original values1.


Weka Explorer: Visualizing original values1.

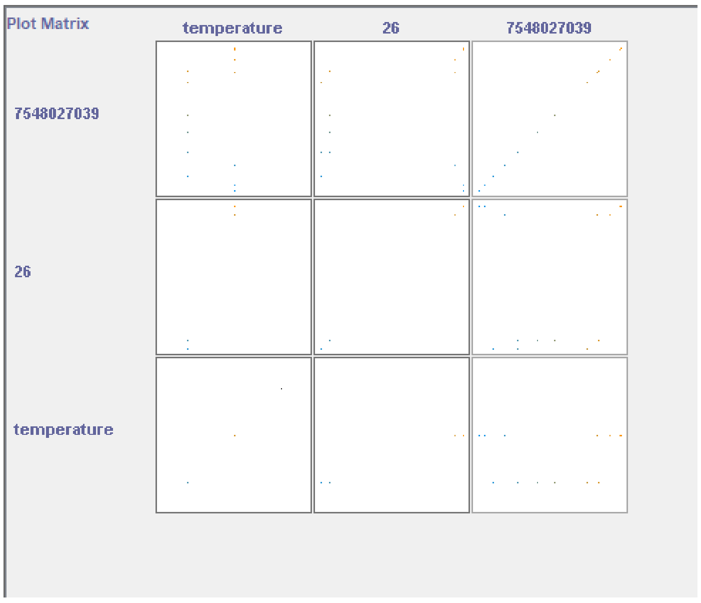
X: 26 (Num)
Y: 7548027039 (Num)
Colour: 7548027039 (Num)
Select Instance

Plot:original values1.

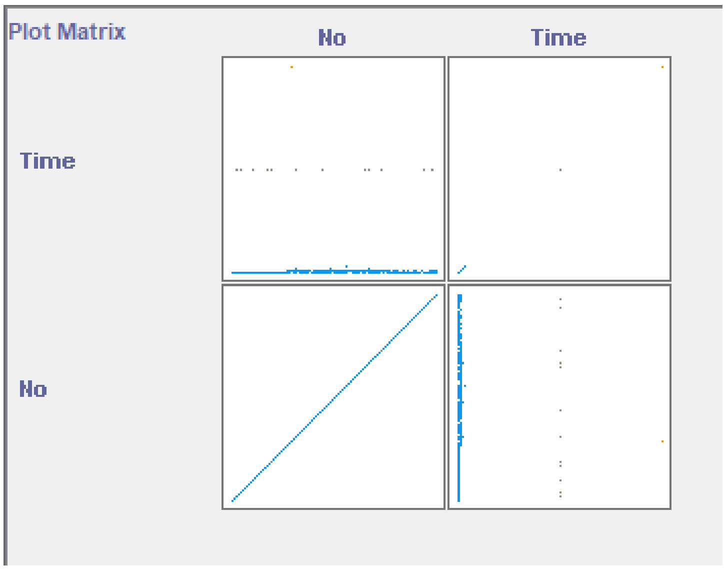

Weka Explorer: Visualizing original values1.

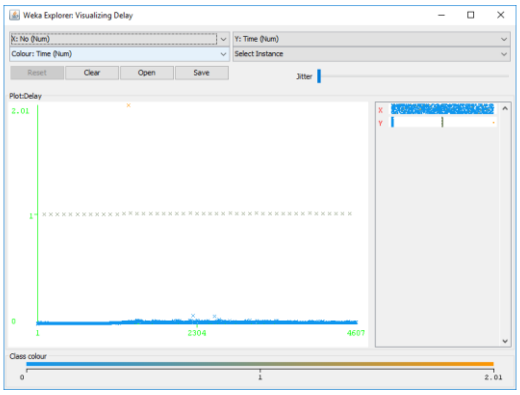
X: 7548027039 (Num)
Y: 7548027039 (Num)
Colour: 7548027039 (Num)
Select Instance

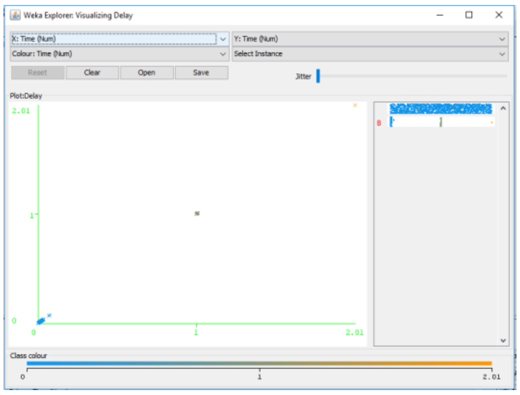
Plot:original values1.

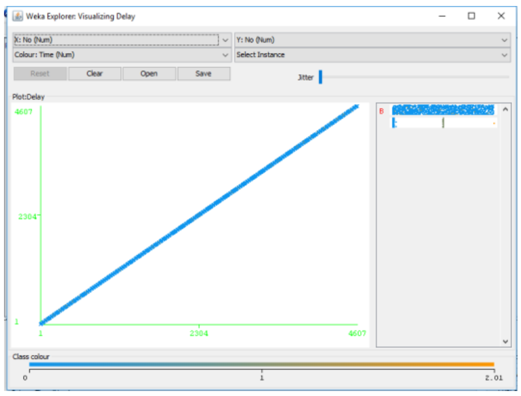


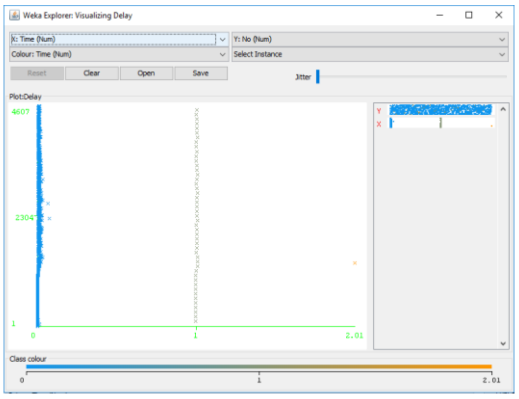
- As for the anomaly input data-set we have the following results.











When having a high cost value there will be a much more strict classification of the training data and the aim is to classify all the training data correctly. The importance of getting these two parameters correct can be crucial as said, if i.e. C is too big it could lead to over-fitting the model. Over-fitting the model means that the hyperplanes will not be placed correct and it could lead to unnecessary incorrect classification. The hyper planes should be placed in the optimal place between the different classes and for this to happen the cost and gamma needs to be correct.

Categorical values vs. continuous values

One important thing needed to be done before the algorithms can be applied and run is the preparation of the data set. It needs to be done as the data set contains categorical values such as protocol type, service and flags. These are categorical values which means that they are not continuous, something they have to be for the algorithm to work. What is meant by categorical values is values in the column or feature, is a value that is not numeric, factors or vectors which are continuous values. There is packages that can be used to convert these categorical values into numeric like the package model. The package model has a function that can read in all the different values and create a numeric value for the categorical.

Partitioning the data set

When doing anomaly detection using machine learning techniques the data has to be divided into at least two data sets. This is in order to have a training data set and at least a testing data set. At least in this context is used since there is very often divided into three data sets as well. The reason for dividing a data set into three data sets is since there should be a training data set, a testing data set and a data set for validation. Which is a good thought, but in this project the validation will be done using cross validation as mentioned in the approach. So having a part of the data set for validation will not be necessary.
The words validation and testing is also mixed up a lot in different papers found, when the word testing the algorithm is used in this paper it refers to the actual predictions that is presented in the results. Training set is the data set that the algorithm will use to learn the patterns from and should after the training data has been applied be able to recognize the different attacks in the test data set. After all of this is done the data sets is almost ready to be used. There are some modifications that still needs to be made before it is ready to be used by the algorithms. There are several steps before the actual calculation can be made as explained in this section of the chapter. It is not just to use any data set at hand and do the predicting of the algorithms. There is several steps and modifications that has to be in place just for the algorithms to be applied. The cost and gamma, are two values that has to be optimized after all the modifications is done. There is a lot to be done and to remember when working with classification algorithms. It is enough to grasp and if one of the steps above has not been preformed there will be errors. Of course there are many data sets that is already optimized and modified, so that one can just go ahead and do the implementation of the algorithms.

Chapter 4

4.1 Discussion

The aim of this thesis has been to compare the overall performance of classification algorithms, when doing anomaly detection using machine learning techniques with both anomalous and normal samples. This is what has been done and described in the previous chapters. There will also be discussed how the results of the experiments has affected the end result that will be discussed in this chapter.
The experiments are designed in accordance with a purposed answer to the problem statement. The problem statement is what the whole thesis is built on, and is the reason behind this project. The problem statement gives a clear problem or question that is attempted giving an answer to in this thesis. The goal of this thesis is as stated in the problem statement, to try and come up with an answer to which classification algorithm who has the best overall performance. The main focus in the papers found is as said on the classification rates/detection rates.
If there should be any possibility for machine learning to be implemented into network security, then time is of importance. The same is the resource consumption in a larger scale, where there is the need for multiple cores for doing the classification and so on. These parameter was therefore found to interesting metrics that could determine the overall performance of the algorithms. In this thesis there is made two different implementations of the algorithms. These implementations are based on the two different type of sample that we had. If the actual results can be considered representable for the algorithms still have to be determined. This will be further discussed in the algorithms and experiments part, along with some other considerations. In the thesis, the part that we focus is about the accuracy that the algorithm can have compared to the others in order to detect the anomaly detection that exist on the samples.

4.1.1 The project

The project as a whole has been a very interesting experience. It started out as a very complex project. After gaining the information needed to understand how the algorithms actually works and how they can be applied and used, the complex parts of the project were more understandable and did not any longer look impossible to manage. Machine learning is a very big and wide field of study, which just keeps on growing. It is used for a lot of things these days and there are researchers developing new ideas on how it can be used for even more than it used for today. Just understanding what machine learning is can take time, since it used for so many purposes and there are so many ways it can be used. So trying to apply machine learning techniques, when doing anomaly detection was a challenge to overcome. When trying to get the information needed to make the actual implementations, there were so many different suggestions on how and what to do technically. Choosing the best algorithm, part was the most time consuming part to understand and implement, because for each control system the requirements change. Each data set or parameters needs to be optimized in different ways. Some data set does only

have continuous values and some only categorical values. This is just some of the modification that has to be considered about the selection of the best algorithm.
There were several problems encountered throughout the project, some were easy fixes others were not. When it comes to machine learning was at the beginning of the project very limited, so there were many small problems along the way and some that were harder to overcome. The most difficult problems were the ones that were with the developing, these were tough to problem solve. Trying to Google the errors can give many different suggestions to what the errors can be about, and might make it even more confusing. Therefore it was very good to have some help from supervisor and others, and there is also some forums that can be of help if needed.

4.1.2 Future work

There are so many different algorithms that could be used to do classification and in this thesis the most common classification algorithms were chosen to be compared. What is actually meant by most common, is that they are widely used in research. But then again there are not many comparisons of the just the two. Many chose to go with a whole bunch of algorithms to see more results, but the positive thing about just using one is that these can be optimized and are the only focus. This would have to be studied more in order for it to be implemented in this project. From the results there is a bit of a gap between the results found in the binary versus the multiclass classification, when using the algorithm SVM. If this is a common problem could have been a study that could enlighten the results gotten in this paper. The parameters should be optimized when running the scripts. Another thing that is a little off about the result is that it performs so well when using the binary data, but when shifting to the multiclass it gains a very low accuracy. So the research would gain a lot from studying this even more than it has been in this project. There has been attempted to do some research in order to see if this an error made by the author or if this is a common thing when using SVM. The answers found were very ambiguous, there were some researchers that stated that it was very good using binary data. Others said that there are so many different versions of the original SVM and that the different versions could have different outcomes.

4.2 Conclusion

The main goal of this thesis was to study the better anomaly detection technique and implement it using machine learning techniques. The study has utilized a machine learning technique called supervised learning. It was implemented through usage of algorithm SVM which is classification algorithm. The results gained from implementing these algorithm with real time data-set, both normal and anomalous, in order to check if our assumption for best suited algorithm is valid. The data set contains different indicators for multiple attacks blended in with normal traffic. This was also done in order to observe how well the algorithms performed, when exposed to multiclass data.

The experiments conducted in this thesis was developed in order to give a proposed solution for the problem statement. The parameters listed in the problem statement were: classification performance, time, resource consumption and others mentioned and analyzed before. The experiments conducted in this thesis is based on these parameters. However, conclusions drawn from the results in this thesis, needs further research to determine the validity and trustworthiness of the research presented.

References:

[1] Jason Brownlee. (2016). *How To Implement The Decision Tree Algorithm From Scratch In Python.* Available: https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/. Last accessed 16/1/2018.

[2] Mayur Kulkarni. (2017). *Decision Trees for Classification: A Machine Learning Algorithm.* Available: https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning- algorithm.html. Last accessed 16/1/2018.

[3] Faraj A. El-Mouadib. (2009). Conference: Conference: In the 8th WSEAS International Conference on DATA NETWORKS, COMMUNICATIONS, COMPUTERS. *New Implementation of Unsupervised ID3 Algorithm (NIU-ID3) Using Visual Basic.net*. - (-), 1-14.

[4] Arinto Murdopo. (2013). *C4.5 Decision Tree Implementation.* Available: http://www.otnira.com/2013/03/25/c4-5/. Last accessed 16/1/2018.

[5] Gilbert Ritschard. (2010). *CHAID and Earlier Supervised Tree Methods.* Available: http://www.unige.ch/ses/metri/cahiers/2010_02.pdf. Last accessed 16/1/2018.

[6] *CHAID and Exhaustive CHAID Algorithms.* Available: ftp://ftp.software.ibm.com/software/analytics/spss/support/Stats/Docs/Statistics/Algorithms/13.0/T REE-CHAID.pdf. Last accessed 16/1/2018.

[7]Chen Li, Yang Zhang. (2008). Fifth International Conference on Fuzzy Systems and Knowledge Discovery. *Bagging One-Class Decision Trees*. - (-), 2-5.

[8] Chen Li, Yang Zhang, Xue Li. (2009). *OcVFDT: One-class Very Fast Decision Tree for One- class Classification of Data Streams*

[9] Ms. A. Sivasankari, Mrs. S. Sudarvizhi, S. Radhika Amirtha Bai. (2014). COMPARATIVE STUDY OF DIFFERENT CLUSTERING AND DECISION TREE FOR DATA MINING ALGORITHM. *International Journal of Computer Science and Information Technology Research*. 2 (3), 221-232

[10]Wei-Yin Lon. (2011). Classification and regression trees. 1-10.

[11] (2018). *REGRESSION TREES* . Available: https://www.solver.com/regression-trees. Last accessed 9/1/2018.

[12] (2018). *Classification and Regression Trees.* Available: http://www.statsoft.com/Textbook/Classification-and-Regression-Trees. Last accessed 9/1/2018. [13] *Association rule learning.* Available: https://en.wikipedia.org/wiki/Association_rule_learning. Last accessed 21/12/2017

[14] Annalyn Ng. *Association Rules and the Apriori Algorithm: A Tutorial.* Available: https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html/2.

Last accessed 21/12/2017.
[15] Trieu Anh Tuan (2012). *A Vertical Representation for Parallel dEclat Algorithm in Frequent Itemset Mining*. Ritsumeikan: Ritsumeikan University. 1-42.
[16] *The FP-Growth Algorithm.* Available: https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_FP- Growth_Algorithm. Last accessed 21/12/2017.
[17]Abdallah Abbey Sebyala, Temitope Olukemi, Dr. Lionel Sacks. (-). Active Platform Security through Intrusion Detection Using Naïve Bayesian Network for Anomaly Detection. 1-5.
[18] Paul Dagum, Micahel Luby . An Optimal Approximation Algorithm for Inference Bayesian . 1- 41.
[19] Antonio Cansado, Alvaro Soto. (2005). Unsupervised anomaly detection using Bayesian Networks and Gaussian Mixture Models. 1-24.
[20] Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. (2003). WSARE: What's Strange About Recent Events?.*Journal of Urban Health: Bulletin of the New York Academy of Medicine*. 80 (2), 1-10.
[21] Weng-Keen Wong, Andrew Moore, Gregory Cooper, Michael Wagner . (2005). What's Strange About Recent Events (WSARE): An Alg. *Journal of Machine Learning Research 6*. 12, 1-38.
[22] Sakshi Babbar and Sanjay Chawla. (2006). On Bayesian Network and Outlier Detection. 1-12.
[23] Rahul Saxena. (2017). *Naive bayes classifier machine learning.*Available: http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/. Last accessed 15/1/2018.
[24]Sunil Ray. (2017). *Naive Bayes explained.* Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/. Last accessed 15/1/2018. [25] Muhammad Ilyas. *A Clustering Based Study of Classification Algorithms/.* Available: https://www.researchgate.net/profile/Muhammad_Ilyas24/publication/295862330_A_Clustering_Ba sed_Study_of_Classification_Algorithms/links/5905e3634585152d2e964bc5/A-Clustering-Based- Study-of-Classificat. Last accessed 15/1/2018.
[26] Saurav Kaushik. (2016). *An introduction to clustering and different methods of clustering.* Available: https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and- different-methods-of-clustering/. Last accessed 15/1/2018.
[27] *K-Means Clustering.* Available: https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html. Last accessed 15/1/2018.
[28] *Data Mining Algorithms In R/Clustering/K-Means.* Available: https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/K-Means. Last accessed 15/1/2018.
[29] *DBSCAN.* Available: https://en.wikipedia.org/wiki/DBSCAN. Last accessed 16/1/2018. [30] Siddharth Agrawal. (2013). *MACHINE LEARNING – DBSCAN.* Available: https://algorithmicthoughts.wordpress.com/2013/05/29/machine-learning-dbscan/. Last accessed 16/1/2018

[31] Jiawei Han. *DBSCAN: A Density-Based Clustering Algorithm.* Available: https://www.coursera.org/learn/cluster-analysis/lecture/MWDS8/5-2-dbscan-a-density-based- clustering-algorithm. Last accessed 16/1/2018.
[32] Sudipto Guha, Rajeev Rastogi, Kyuseok Shim. (2001). Cure: an efficient clustering algorithm for large databases. *Information Systems*. 26 (1), 35-58.
[33] Animesh Patcha, Jung-Min Park. (16/2/2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Science Direct*. 1-23.
[34] Jim Frost . (19/2/2015). *Choosing between a non parametric test and a parametric test.* Available: http://blog.minitab.com/blog/adventures-in-statistics-2/choosing-between-a- nonparametric-test-and-a-parametric-test. Last accessed 11/1/2018
[35] Dev Nag. (2016). *Why is operational anomaly detection so hard.* Available: https://www.wavefront.com/why-is-operational-anomaly-detection-so-hard/. Last accessed 11/1/2018.
[36] Preetam Jinka. (2017). *Exponential Smoothing for Time Series Forecasting.* Available: https://www.vividcortex.com/blog/exponential-smoothing-for-time-series-forecasting. Last accessed 19/12/2017
[37] Dillon R. Gardner. (2017). *STL Algorithm Explained: STL Part II.* Available: http://www.gardner.fyi/blog/STL-Part-II/. Last accessed 19/12/2017.
[38] Jason Brownlee. (2017). *How to Create an ARIMA Model for Time Series Forecasting with Python.* Available: https://machinelearningmastery.com/arima-for-time-series-forecasting-with- python/. Last accessed 19/12/2017.
[39] *Autoregressive integrated moving average.* Available: https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average. Last accessed 19/12/2017.
[40]Wikipedia. *Artificial neural network*. Available: https://en.wikipedia.org/wiki/Artificial_neural_network. Last accessed 18/12/2017. [41]XenonStack. (2016). *Overview of Artificial Neural Networks and its Applications*. Available: https://medium.com/@xenonstack/overview-of-artificial-neural-networks-and-its-applications- 2525c1addff7. Last accessed 18/12/2017.
[42]Varun Chandola, Arindam Banerjee, Vipin Kumar. *Anomaly Detection: A Survey*. University of Minnesota: 1-58.
[43] Jahnavi Mahanta. (2017). *Keep it simple! How to understand Gradient Descent algorithm*. Available: https://www.kdnuggets.com/2017/04/simple-understand-gradient-descent- algorithm.html. Last accessed 18/12/2017.
[44]*Gradient descent*. Available: https://en.wikipedia.org/wiki/Gradient_descent. Last accessed 18/12/2017.
[45] *Backpropagation*. Available: https://en.wikipedia.org/wiki/Backpropagation. Last accessed 18/12/2017.
[46]Simon Hawkins, Hongxing He, Graham Williams and Rohan Baxter. *Outlier Detection Using Replicator Neural Networks*. Australia: CSIRO Mathematical and Information Sciences. 1-10. [47]Varun Chandola, Arindam Banerjee, Vipin Kumar.

Anomaly Detection: A Survey. University of Minnesota: 1-58.
[48] *Nearest neighbour algorithm.* Available:
https://en.wikipedia.org/wiki/Nearest_neighbour_algorithm. Last accessed 19/12/2017.
[49] Li-Yu Hu, Min-Wei Huang, Shih-Wen Ke and Chih-Fong Tsai (2016). *The distance function effect on k-nearest neighbor classification for medical datasets*: Springerplus.
[50] Rahul Saxena. (2016). *Knn Classifier, Introduction to k-nearest neighbor algorithm.* Available: http://dataaspirant.com/2016/12/23/k-nearest-neighbor-classifier-intro/. Last accessed 19/12/2017. [51]Ondrej Linda, Milos Manic, Todd Vollmer, Jason Wright. (April 2011). Fuzzy Logic Based Anomaly Detection for Embedded Network Security Cyber Sensor.*2011 IEEE Symposium on Computational Intelligence in Cyber Security*. 1-10.
[52] (April 8,2010).*A Short Fuzzy Logic Tutorial.* Available:
http://cs.bilkent.edu.tr/~zeynep/files/short_fuzzy_logic_tutorial.pdf. Last accessed 8/1/2018.
[53] Shahaboddin Shamshirband, Nor Badrul Anuar, Miss Laiha Mat Kiah, Sanjay Misra. (2014). Anomaly Detection using Fuzzy Q-learning Algorithm. 11 (8), 1-24.