**Research questions**

How should an architecture that analyses logs be designed?

(1) Can on-line anomaly detection be performed on log data while limiting the number of false positives?

(2) Can the anomalies be placed in context so that we can clearly state why they are classified as anomalies?

(3) Can the anomaly detection model handle changes over time without future predictions being influenced by incidental anomalies?

The goal of this research is to provide an overview of techniques currently used in outlier-detection. To accomplish this goal we pose two knowledge-questions:

1. What approaches are currently used in outlier-detection according to published literature?

2. What algorithms are currently used in outlier-detection approaches according to published

literature?

1. Can the anomaly detection quality be improved by incorporating architectural

knowledge?

* Anomaly detection quality does not change or is reduced by incorporating

architectural knowledge.

* Anomaly detection quality is increased by incorporating architectural

knowledge.

2. Can the anomaly detection quality be improved by incorporating event knowledge?

*-* Anomaly detection quality does not change or is reduced by incorporating

event knowledge.

*-* Anomaly detection quality is increased by incorporating event knowledge.

main contributions of this thesis:

\_ We have made a feature selection of unlabeled network tra\_c data in order to improve

anomaly detection and have a better understand of the generated data.

\_ We ran machine learnings algorithms for anomaly detection based on di\_erent outlier detection

approaches on the same dataset.

\_ We identi\_ed alerts of network data by using density and statistical machine learning algorithms.

\_ We performed a comparison of proposed anomaly detection algorithms and examined their

performance according to evaluation metrics.

\_ We have evaluated alerts by constructing access control request regarding access control

policies and veri\_ed the reliability of those alerts, for a given use case of situation awareness.

**Goal (must rephrase)**

Analyzing the time series of those properties across multiple subsystems, one can recognize patterns which correspond to system health. It is hence the goal of this thesis to reliably identify system anomalies by those patterns using machine learning algorithms. While known anomalies will form the basis for the training data, the trained system should also identify yet unknown anomalies,

Challenge 1: How to design methods that are able of coping with time series data

that includes anomalous behavior? In other words, how to distinguish between

normal and abnormal behavior of data automatically?

Knowledge questions:

◦ What approaches are currently used in outlier-detection according to published

literature?

◦ What are the algorithms used in outlier-detection?

◦ What approaches are applicable for analysing access-logs?

◦ What algorithms are most applicable for our research-problem?

|  |  |  |  |
| --- | --- | --- | --- |
| Data-mining |  |  |  |
| Outlier detection | Anomaly detection | Deviant behaviour  detection |  |
| Approach | Method | Technique |  |
| Survey | Literature study | Overview | Review |
| Pattern recognition | Behaviour patterns |  |  |
| Logs | Access-logs | Event-logs |  |

Types of Anomalies

In the context of continuously changing services the following kinds of anomalies were identified. Some of them are to be expected when service instances are deleted and replaced with newer version. However, some of the anomalies represent unexpected behavior and should trigger alerts.

**Research method**

References

|  |
| --- |
| **Isolation Forest**  - F. T. Liu, K. M. Ting, and Z.-H. Zhou, ``Isolation forest,'' in *Proc. 8th IEEE Int. Conf. Data Mining (ICDM)*, Dec. 2008, pp. 413\_422.  - Z. Ding and M. Fei, “An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data Using Sliding Window,” *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, 2013. 3rd  IFAC Conference on Intelligent Control and Automation Science ICONS 2013.  F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation-Based Anomaly Detection,” *ACM Trans. Knowl. Discov.*  *Data*, vol. 6, pp. 3:1–3:39, Mar. 2012.  Wu, K. et al. (2014).  “RS-Forest: A Rapid Density Estimator for Streaming Anomaly Detection”.  In: *Data Mining (ICDM), 2014 IEEE International Conference*  *on*. IEEE, pp. 600–609  Tan, S. C., K. M. Ting, and T. F. Liu (2011).  “Fast anomaly detection for streaming data”. In: *IJCAI Proceedings-*  *International Joint Conference on Arti\_cial Intelligence*. Vol. 22. 1.  Citeseer, p. 1511 |
| One of the machine learning approaches to detect anomalies in time series is isolation forest using sliding window. Isolation forest, also known as iForest, was introduced by Lui et al. [49]. It builds an ensemble of Isolation Trees *iTrees*, which are binary trees isolating data points. As anomalies are  more likely to be isolated than non-anomalous points, it is more likely that they are closer to the root of an iTree [50]. Figure 3.4 shows how an iTree is generated on a sample data structure. An anomaly is detected after two partitions while the first normal point is detected after the fourth partition:  Thus, this method considers points with shorter path lengths as candidates that are highly likely to be anomalies.  The anomaly detection process using Isolation Forests is performed generally in two steps:  a) Training: Create *n* iTrees for the given training set.  b) Evaluation: Pass the test instance through the isolation trees to determine the anomaly score.  There are some methods that have extended iForest to detect anomalies on time series. One main approach to detect anomalies in univariate data is to analyse the dataset in sequences defined by the window length *w*. Thus, let *{XT }* = (*x*1*, x*2*, ..., xT* ) be a univariate time series and *w* the window length. Then:  *W* :=(*W*1*,W*2*, ...,Wp*)  =((*x*1*, ..., xw*)*,* (*x*2*, ..., xw*+1)*, ...,* (*xp, ..., xw*+*p−*1))  (3.24)  After that, the anomaly score on each sequence is computed, which is proportional to the average path length of an instance. Using supervised learning, the threshold can be computed on the training set and used for test set later.  Ding et al. [51] have extended the concept to compute the anomaly score *S*(*x, p*) where *x* is the data point and *w* is size of the window:  where *hi*(*x*) denotes the length of the i-th iTree, *E*(*h*(*x*)) the average of *h*(*x*) from a collection of iTrees and *c*(*p*) is the average of *h*(*x*) given *w* and *L* the number of iTrees.  The main challenges of the isolation forest algorithm are the following parameters:  • Window length *w*: If the length is too short, then there will not be enough data to construct an appropriate model. On the other hand, if the length is too long, older and sometimes less relevant data will be considered as much as more recent data points. Ding et al. [51] have shown in their experiment results that fixed sliding window for different datasets result in bad performance.  • Number of iTrees in the iForest: The higher the number of iTrees, the closer the average value is to the expected value. The downside is that the higher number of iTrees will increase the computation time [50]. For *w* as the window length and *L* the number of iTrees, iForest has a time complexity of *O*(*L · w*2) and a space complexity of *O*(*L · w*).  • Contamination: Many implementations of iForest like the implementation in *sklearn* have a contamination parameter where the proportion of anomalies in the dataset is set. This also marks the threshold for the anomaly. Improper assignment of this parameter could result in a higher rate of false positive or false negatives.  Isolation Forest [39] is an anomaly detection algorithm designed to detect outliers in high dimensional data. This algorithm does not make assumptions about the distribution of the data and does not require the data to be normally distributed. Additionally, it does not require any labels to detect anomalies and therefore, it  is a suitable choice for this purpose. The Isolation Forest algorithm is very suitable and effective for detecting  anomalies in high dimensional data with lots of irrelevant features and noise which has a small number of anomalies or no anomalies at all, since it the majority of these anomalies will be very close to the root of the tree. This algorithm also performs well on normal unbiased data which does not have a lot of noise, is  non-parametric which makes it suitable for unsupervised anomaly detection [3] and it is computationally efficient. With using Isolation Forest algorithm even the few true positives that are present will be found. However, this technique is effective in detecting local outliers [13]. The algorithm partitions the domain space. These partitions are created totally at random by selecting a feature and then creating a split value between the minimum and maximum value of the feature The algorithm keeps creating partitions until all data points are isolated. This is done by creating random decision trees. The score for each point in then calculated based on the length of the path needed to isolate that point as shown in 2.7.  𝑠(𝑥, 𝑛) = 2- E(h(x))  c(n) (2.7)  The anomaly score for one point is calculated by taking the average anomaly scores across all trees for that point. A score closer to 1 indicates that this is an anomaly. A score much smaller than 0.5 indicates that this is a normal observation. If all scores are around 0.5 then the data doesn’t contain any obvious anomalies. An anomalous point would have a short path to travel to its root. The decision tree is constructed such that first one feature is chosen and is split based on the min and max value observed. In these splits the next feature is split in the same manner until all features are passed. For anomalous instances, these paths will be close to the minimal length since they would have unique and few splits. For normal instances these splits would be larger than the minimum path length because the splits. Isolation Forest might work better with one-hotencoding because the uniqueness of observed values within a feature is preserved, whereas with frequency encoding the feature is seen as a continuous variable and the uniqueness of some observed instances is lost.  Isolation Forest (iForest) [41] is a model based anomaly detection technique, which is built on the idea of random trees. Here, `isolation' means separating an anomalous instance from the rest of the instances. iForest isolates instances by random partitioning of a tree followed by random selection of the features. This random partitioning produces shorter paths for anomalies. The path length from the root node to the terminating node is averaged over a forest of random trees.  The isolation forest is an algorithm that generates outliers by using a decision  (‘sklearn.ensemble.IsolationForest — scikit-learn 0.19.1 documentation’, 2018). It also assigns an anomaly score to every outlier it finds. This score gives insight in whether the request is very deviant or barely outside the normal spectrum. This anomaly scores allows for the filtering of outliers by setting a minimal score specified by the users. Some potential false positives can be filtered out of the collection of outliers that is send to the user for further investigation.  The input used for the algorithm is the output of the preprocessing process, which is a collection of users with similar job descriptions. The user can specify two formats for input. The first format is a dissimilarity matrix, this gives a more broad overview that allows for a comparison of users. The second format consists of all the encoded requests made by users in the group. This allows for detecting specific requests as outliers. Which user made the request is then not taken into account. This results in two types of outputs. The first format results in users who are deemed outliers, while the second format results in request which are deemed as outliers. In both cases however anomaly scores are returned. These scores can be used to filter the outliers and only return the truly interesting outliers.  The *Isolation Forest*37 (*i*Forest) is an anomaly detection algorithm based on randomtrees. The idea is to separate (isolate) all instances using a binary tree structure called isolation tree (*i*Tree). The assumption is that anomalies are rare and that they obey di\_ferent properties than normal instances. Hence it should be easier to isolate them which implies that anomalies are closer to the *i*Tree’s root and therefore have a shorter path length, denoted by h(x). On the contrary, normal instances are more difficult to isolate since they lie in dense regions of the input space and are therefore deeper located within  the tree.  Isolation forests achieve a constant training time and space complexity by subsampling the training set to a \_\_xed size and can therefore be applied to datasets with a large number of samples. However, the random attribute and split point selection can become problematic in high dimensions as the splitting can occur at suboptimal attributes. It would be better to exploit the knowledge contained in the data to guide this selection. |
| **HBOS**  M. Goldstein and A. Dengel, ``Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm,'' in *Proc. KI-2012:* *Poster Demo Track*, 2012, pp. 59\_63.  [23] Eric Falk, Ramino Camino, Radu State, and Vijay K Gurbani. On  non-parametric models for detecting outages in the mobile network. In  2017 IFIP/IEEE Symposium on Integrated Network and Service Management  (IM), pages 1139–1142. IEEE, 2017. doi: 10.23919/INM.2017.  7987448.  [24] Markus Goldstein and Andreas Dengel. Histogram-based outlier score  (hbos): A fast unsupervised anomaly detection algorithm. KI-2012: Poster  and Demo Track, pages 59–63, 2012.  [49] Nerijus Paulauskas and Algirdas Baskys. Application of histogram-based  outlier scores to detect computer network anomalies. Electronics, 8(11):  1251, 2019. doi: 10.3390/electronics8111251.  [87] Tommaso Zoppi, Andrea Ceccarelli, and Andrea Bondavalli. On algorithms  selection for unsupervised anomaly detection. In 2018 IEEE 23rd Pacific  Rim International Symposium on Dependable Computing (PRDC), pages  279–288. IEEE, 2018. doi: 10.1109/PRDC.2018.00050. |
| Histogram-Based Outlier Score (HBOS) [16] is another statistical unsupervised anomaly detection algorithm. This algorithm is computationally far less expensive as compared to nearest neighbor and clustering based anomaly detection methods. HBOS works on arbitrary data by offering a standard \_xed bin width histogram as well as dynamic bin width (\_xed amount of items in each bin).  2.5. Histogram-based Outlier Score  The Histogram Based Outlier Score (HBOS) is a statistical based anomaly detection technique [24]. This technique has a linear time complexity 𝒪(𝑛) for a static fixed bin width and a 𝒪(𝑛 ∗ 𝑙𝑜𝑔𝑛) complexity for a dynamic bin width. This techniques models the feature densities using histograms with a bin width that can be both static or dynamic dependent on the scenario that it is used for. The created models are subsequently used for the calculation of an anomaly score for each data point. The HBOS algorithm constructs an uni-variate histogram for every feature. This algorithm can work with both categorical and numerical  features. If the features are categorical, a frequency is calculated based on the 18 2. Preliminaries counts of each feature category and if the features are numerical static or dynamic bin width histograms are used. For the static bin width histogram, k equal bins  are used where the density is determined based on the frequency of points that belong to one bin. The dynamic bin widths are determined by sequential values with a fixed amount of 􁍍􁍤 that are placed into one bin. The number of bins, 𝑘, is often chosen as the square root of the total number of data points 𝑁, so 𝑘 = √𝑁.  The HBOS for every data point 𝑝 can be calculated with the following formula:  (2.5)  where 𝑘 represents the number of bins, 𝑁 represents the number of data points and 𝑑 represents the dimension. All histograms are normalised in a way so that a maximum value of 1 for the height of each histogram is obtained. The advantage of HBOS is that it does not require labelling of the data, is effective in detecting global outliers and is very fast and efficient which makes it easier to apply this technique to larger data sets [23, 24, 49, 87]. Also since this technique assigns outlier scores it is easier to estimate how reliable the predictions are. However, this technique does not perform well for the detection of local outliers [24] and it is difficult to get context from the obtained results [87].  Figure 2.5: Example of Histogram Based Outlier Score  An example of how an anomalous instance would be found by HBOS is the following: The first bin in the range [10, 20] has a height of 13. The score for each point in this bin is thus computed as log(1/13) = -1.1139. The score for the histogram in range [70-80] would be equal to log(1/2) = -0.30102. This score is closer to 0 and thus these points are considered more anomalous than the points in bin [10-20]. For each attribute included in the training phase a histogram is build and the final score for a point is computed as the sum of each attribute score. Higher scores represent more abnormal points that would be assigned to  bins with a low density.  Histogram-based Outlier Score (HBOS) HBOS calculates an outlier score by creating a histogram with a \_xed or a dynamic binwidth [18]. This operator calculates a separate univariate histogram for every column in the dataset. There are two modes, one with a static and one with a dynamic bandwidth. In the static mode, every bin has the same binwidth equally distributed over the value range. In the dynamic mode, the binwidth can vary, but it is possible to specify a minimum number of examples contained in a bin.  De\_nition 3.3.2. To compute the outlier score, the histograms are normalized to one in height  \_rst. Then, the score is inverted, so that anomalies have a high score and normal examples a low  score. |
| **CBLOF**  [50] Z. He, X. Xu, and S. Deng, Discovering cluster-based  local outliers. Pattern Recognition Letters, 24(9):1641–1650, 2003.  [21] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. Pattern Recognition Letters, 24(9):1641{1650, 2003. |
| Cluster-Based Local Outlier Factor  (CBLOF) [15] is a clustering based anomaly detection algorithm, in which data points are clustered using *k*-means (or any other) clustering algorithm. The anomaly score of an instance is the distance to the next large cluster. As this approach is based on clustering algorithm, the problem of choosing the right number of clusters arises, and reproduction of the same anomaly score also becomes impossible due to non-deterministic nature of clustering algorithms.  Cluster-Based Local Outlier Factor (CBLOF): receives as an input the dataset and generates the cluster model according to a clustering algorithm. The outliers score is calculated according to the size of the cluster and the distance to the nearest big cluster centroid as proposed by He et al. [21]  Cluster-based Local Outlier Factor  The Cluster Based Local Outlier Factor (CBLOF) [28] is a technique that assigns an outlier factor to each data point which is determined by the distance of the data point to the nearest neighbouring clusters and the size of its own cluster. In order to assign these factors, the data points are first divided into clusters.  A similarity function is computed for every tuple and a data point is added to the cluster with the highest similarity value. The complexity of this techniques  is 𝒪(𝑛). The cluster-based outlier factor for a point 𝑡 can be calculated with the  following formula:  𝐶𝐵𝐿𝑂𝐹(𝑡) = {  |𝐶∗  􁍢 | min(distance (𝑡, 𝐶􁍣)), where t ∈ 𝐶􁍢, 𝐶􁍢 ∈ SC and 𝐶􁍣 ∈ 𝐿𝐶 𝑓𝑜𝑟 𝑗 = 1 𝑡𝑜 𝑏  |𝐶∗  􁍢 | min(distance (𝑡, 𝐶􁍣)), where t ∈ 𝐶􁍢 𝑎𝑛𝑑 𝐶􁍢 ∈ LC  (2.6)  where 𝐶􁍢 and 𝐶􁍣 represent clusters, SC = 𝐶􁍢, | 𝑗 > 𝑏 represents set with small  clusters and LC = 𝐶􁍢, | 𝑗 ≥ 𝑏 represent the set with large clusters and 𝑏 is the  Figure 2.6: Example of CBLOF Score Calculation  boundary of these clusters.  The advantages of using CBLOF for outlier detection are that this technique has a linear complexity which makes it perform fast computations and is easily scalable for large data sets [28].  Because point A is clustered in a small cluster, the score for this point is computed as 𝑠𝑐𝑜𝑟𝑒􁍀 = min(D1, D2). Where D1 and D2 are the respective distances from the point to the cluster centres of the two large clusters. The score for point B, which is part of a large cluster, is computed as the score from the point to its  own cluster’s centre. So the score for point B is 𝑆𝑐𝑜𝑟𝑒􁍁 = 𝐷3. |
| KNN  [11] S. Ramaswamy, R. Rastogi, and K. Shim, ``Effcient algorithms for mining outliers from large data sets,'' *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 427\_438, 2000.  14] Stephen D. Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03, pages 29{38, New York, NY, USA, 2003. ACM.  [46] Amol Ghoting, Srinivasan Parthasarathy, and Matthew Eric Otey. Fast mining of  distance-based outliers in high dimensional datasets. In Proceedings of the Sixth SIAM  International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA,  pages 609{613, 2006.  [120] Mingxi Wu and Christopher Jermaine. Outlier detection by sampling with accuracy  guarantees. In Proceedings of the 12th ACM SIGKDD International Conference on  Knowledge Discovery and Data Mining, KDD '06, pages 767{772, New York, NY,  USA, 2006. ACM.  18 Knorr and Ng, *Algorithms for mining distance-based outliers in large datasets*;  Knorr, Ng, and Tucakov, *Distance-based outliers: algorithms and applications*.  Knorr, E. M. and R. T. Ng (1998).  “Algorithms for mining distance-based outliers in large datasets”. In:  *Proceedings of the International Conference on Very Large Data Bases*.  Citeseer, pp. 392–403 (cited on page 33).  Knorr, E. M., R. T. Ng, and V. Tucakov (2000).  “Distance-based outliers: algorithms and applications”. In: *The VLDB*  *Journal—The International Journal on Very Large Data Bases* 8.3-4,  pp. 237–253 (cited on page 33).  [55] Leif E Peterson. K-nearest neighbor. Scholarpedia, 4(2):1883, 2009.  [56] Edwin M Knorr and Raymond T Ng. Finding intensional knowledge of  distance-based outliers. In VLDB, volume 99, pages 211–222, 1999.  [57] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms  for mining outliers from large data sets. In ACM Sigmod Record,  volume 29, pages 427–438. ACM, 2000.  [58] Varun Chandola, Shyam Boriah, and Vipin Kumar. Understanding categorical  similarity measures for outlier detection. Technical report 08–008,  University of Minnesota, 2008.  [59] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures  for categorical data: A comparative evaluation. In Proceedings of the  2008 SIAM International Conference on Data Mining, pages 243–254.  SIAM, 2008.  [60] Matthew Eric Otey, Amol Ghoting, and Srinivasan Parthasarathy. Fast  distributed outlier detection in mixed-attribute data sets. Data Mining  and Knowledge Discovery, 12(2):203–228, May 2006.  Bibliography 123  [61] Stephen D Bay and Mark Schwabacher. Mining distance-based outliers  in near linear time with randomization and a simple pruning rule.  In Proceedings of the ninth ACM SIGKDD international conference on  Knowledge discovery and data mining, pages 29–38. ACM, 2003.  [62] Amol Ghoting, Matthew Eric Otey, and Srinivasan Parthasarathy.  Loaded: Link-based outlier and anomaly detection in evolving data sets.  In Data Mining, 2004. ICDM’04. Fourth IEEE International Conference  on, pages 387–390. IEEE, 2004.  [63] Mingxi Wu and Christopher Jermaine. Outlier detection by sampling  with accuracy guarantees. In Proceedings of the 12th ACM SIGKDD  international conference on Knowledge discovery and data mining, pages  767–772. ACM, 2006. |
| *k*-NN anomaly detection method is the simplest and most widely used unsupervised global anomaly detection method for point anomalies. This distance based algorithm calculates the anomaly score based on *k*-nearest-neighbors distance [11]. This technique is computationally expensive, highly dependent on the value of *k*, and may fail if normal data points do not have enough neighbors.  earest Neighbor-based techniques: Techniques that are based on nearest neighbor approaches  assume that all normal data points are located close to each other while all other data points are considered anomalous [Chandola et al., 2009]. Therefore, simple nearest neighbor based approaches are not capable of detecting contextual or even collective anomalies in data sets. However, there are approaches that use complex metrics in order to enhance the process of finding contextual anomalies [Boriah et al., 2008]. There are also approaches that first cluster the data set with an appropriate clustering algorithm, for example the k-nearest neighbor algorithm and prune all  clusters that cannot be anomalous. The anomalies are then calculated based on this condensed set [Ramaswamy et al., 2000].  For example, given a data set, Ramaswamy et al. [96] considered data objects with  the top-k highest Dks as outliers. The Dk of a data object o is the distance between o and ist kth nearest neighbor. The distance between two data objects can be measured by di\_erent distance metrics, such as manhattan distance, euclidean distance, and so on. In this paper, the authors use the square of the euclidean distance instead of the euclidean distance itself, for the sake of reducing computational costs. Angiulli and Pizzuti [9] provided another way of de\_ning the k-NN distance based outliers. Speci\_cally, the outlier score of a data object o is de\_ned as the sum of the distances from o to all its k nearest neighbors. The outliers are data objects with top-k highest outliers scores. Many methods [14, 46, 120] have been  developed to improve the e\_ciency of the distance-based outlier detection methods.  The e\_ectiveness of proximity-based methods highly depend on the proximity measure  Nearest neighbor-based anomaly detection techniques assume that normal instances occur in dense neighborhoods, while anomalies are isolated. There are di\_ferent ways to de\_\_ne such a *dense neighborhood*. Most common is to declare the distance to the k-nearest neighbor of an instance as ist neighborhood. The number of points within this neighborhood is then used to approximate a density. The advantage of these algorithms is that they only require a metric. They can therefore be applied to a variety of domains such as videos, images, trajectories or more complicated structures like graphs. This section reviews some frequently applied techniques and discusses their advantages and disadvantages.  *Distance-based Outlier Detection*  A straightforward way to build an anomaly detector based on the neighborhood  is to use the k-nearest neighbors distance as an anomaly score    as suggested by Ramaswamy et al.17 The assumption made here is that the k-nearest neighbor distance for an anomaly is much large than for normal instances.There exists di\_ferent variations of this simple idea. Knorr et al. propose a *distance-based outlier detection*18 algorithm which classi\_\_es an instance as  anomalous (distance-based outlier) if at least a fraction of \_ of the objects in the training have a distance from this instance that is larger than \_. Distance-based outlier detection methods determine the label of an instance based on the distance to its neighbors, and they are also known as Nearest Neighbor(NN)-based algorithms [55]. As non-parametric approaches, they make no assumptions about the underlying distribution from which the datasets are generated. Instead, the most important motivation is the local similarity. It suggests that when the distance between instances in feature space are small under a specific distance metric, they share similar mode of behavior. In other words, the proximity of an outlier object to its neighbors is very di↵erent from that of a normal object. The region of the neighbors is determined by the top k instances of the ascending sequence of distances to the reference point. k-NN based outlier detection algorithms was first proposed by Ng and Knorr [56]. It takes a local view of the data, inferring the behavior based on the local neighbors. It assumes that normal data points have close neighbors in the normal training set, while outlier points are located far from those points. An outlier instance is then declared if it is located far from its neighbors. Although being simple and flexible, k-NN based methods in practical applications face  some difficulties. Firstly, the key component in k-NN is the distance metric, commonly Euclidean distance, Mahalanobis distance and Hamming distance. Ramaswamy et al. [57] considered data objects with the top-k highest Dks as outliers. The Dk of a data object x is the distance between x and the first kth nearest neighbor surrounding the location of x in the feature space. For categorical features, a simple matching coefficient is used, while some other well designed metric [58, 59] have been proposed. For a data set of mixed features,  it is challenging to choose a universal metric to generate meaning distance for reference. Except for [60], less research has been done in this direction in spite of its importance in practical application. Secondly, k-NN based algorithm utilizes the ’majority voting’ for assigning a label to an instance referring to ist neighbors’ labels. It is required to compute the distances between every point in the whole data set. The computational complexity increase as the cardinal of the data set increases. To address this problem, extra assumptions are required to prune the scope of target space. Bay and Schwabacher [61] introduced the  distance-based outlier detection algorithm called ORCA and showed that for sufficiently randomised data, a simple pruning step could result in the average complexity of the nearest neighbor search being approximately linear. Ghoting et al. [62] further improve the performance of ORCA by proposing Recursive  Binning and Reprojection. Wu [63] introduced sampling techniques to improve the efficiency.  **3.2.1 Nearest neighbour-based algorithms**  The common idea behind nearest neighbour-based algorithms for anomaly detection is to compute an anomaly score using some relation (for example, the distance) to a number of nearest neighbours. We will start the discussion with the KNN outlier detection algorithm by Ramaswamy et al. [21].  **K-nearest neighbours**  In 1998, Knorr and Ng proposed an anomaly detection algorithm that used a simple and intuitive definition [16]. Their definition considers every point that has no more than k points within a distance d to be an anomaly. In their definition, k and d are user defined parameters. The paper by Knorr and Ng  was the basis for the first K-nearest neighbour (KNN) anomaly detection algorithm by Ramaswamy et al. in 2000 [21]. The changes to the original idea that were made by Ramaswamy et al. were made to solve a few important shortcomings. The disadvantages Ramaswamy et al. identified were the following:  1. The user has to specify a distance, d, which can be difficult to determine.  2. There is no ranking among the anomalies.  3. The algorithm is too inefficient to scale to higher numbers of dimensions.  Especially the first two points can be a problem in our application domain. This is the reason we will experiment with the algorithm by Ramaswamy et al. rather than the algorithm by Knorr and Ng.  Before looking at the solution Ramaswamy et al. have for the problems mentioned above, we will first discuss the algorithm they came up with. They built their algorithm based on the following definition of an anomaly:  “Given a k and n, a point p is an outlier if no more than n 􀀀 1 other points in the dataset have a higher value for Dk than p”. In this definition, Dk is the distance to the k-th nearest neighbour of a point. This distance, Dk, is used as the anomaly score for this algorithm. The definition that is given requires two parameters for the algorithm, namely, k and n. k indicates the number of neighbours that is taken into account, while n specifies the number of anomalies that has to be found. However, setting the expected number of anomalies beforehand may not be desirable. We will therefore work with a ranking of all the points based  on their anomaly scores. Based on the relative anomaly scores that are found, we can then determine which of the points are outliers and which are not. We can, for example, separate the anomalies from the normal data by putting a boundary between the two at twice the mean anomaly score. This is just an example, and the experiments will determine what kind of threshold works well. There is a trivial way to convert the above definition into an algorithm. The simple algorithm Ramaswamy et al. propose is a block nested-loop  algorithm which has a running time that is quadratic in the size of the input. Alternatively, they also proposed two more efficient algorithms to find anomaly scores matching the above description. One uses an R\*-tree to perform KNN queries, while the other partitions the data and prunes partitions that cannot contain anomalies thereby reducing the number of points to check. |