TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Institute of Informatics
Chair of Information Systems

Outlier Detection on Marinexplore.org Sensor Data

Master's Thesis

Student:	Kristian Allikmaa
Student code:	IABM121772
Supervisors:	Ants Torim André Karpištšenko

Tallinn
2015

Declaration

Herewith I declare that this thesis is based on my own work. All ideas, major views and data from different sources by other authors are used only with a reference to the source. The thesis has not been submitted for any degree or examination in any other university.

-- --

(*date*) (*signature*)

Abstract

This thesis deals with the problem of outlier detection from oceanographic spatiotemporal data from the ARGO program. We present a comprehensive model for the application; we compare and choose the most appropriate outlier detection methods for the current domain.

ARGO is a global array of more than 3 000 free-drifting profiling floats that measures the temperature and the salinity of the upper 2 000 m of the ocean.

Marinexplore.org is the largest free Internet portal for public data in the oceanic, atmospheric and climate domain. It provides powerful search, exploration and visualization tools to access in-situ, raster and model datasets. However, the portal has several problems related to the fact that there is no raw data quality control procedure in place (raw data, AS IS).

The first problem is that the quality control process is too long in the ARGO program - 6 to 12 months. The second problem is that in the Marinexplore.org databases the sensor data from floats is AS IS (due to the first problem). There is a need to improve the data quality. Sensor errors are common in this field. The third problem is that there is no single method for outlier detection and no instruction to choose the method for a specific task or problem. There is no unifying theory available. Applying a technique developed in one domain to another is not straightforward.

This thesis has three main outcomes. Firstly, we express the appropriate data validation method. Secondly, our calculations prove that if a stable waterfront is located lower than 1 500 meters below the sea level, it is possible to use it as a reference. It helps to evaluate the state of the sensor. Thirdly, we demonstrate the comprehensive model for the application.

Our approach is applicable more broadly than only in Marineplore.org - it can also be applied to other actors in the same field.

The thesis is in English and contains 99 pages of text, 6 chapters, 36 figures, 9 tables.

Annotatsioon

Käesoleva magistritöö peamine eesmärk on võrrelda ning välja pakkuda efektiivne anomaaliate tuvastamise meetod, et parandada andmekvaliteeti Marinexplore.org sensorandmetes.

ARGO on rahvusvaheline koostööprojekt, mille hallata on enam kui 3 000 ujuvpoist koosnev sensorite massiiv, mis mõõdab erinevaid parameetreid (veetemperatuur, soolsus).

Marinexplore.org on platvorm, mis koondab andmed ookeanide, atmosfääri ning kliimaga seotud valdkondadest, s.h konsolideerib andmeid ARGO programmist. Puuduvad täiendavad andmekontrollid, seepärast on andmekvaliteediga seotud mitmeid probleeme.

Esimeseks probleemiks on see, et ARGO programmi andmete valideerimisprotsess võtab täna aega 6-12 kuud, selline kestus on selgelt liiga pikk. Tulenevalt eeltoodust on teiseks probleemiks asjaolu, et Marinexplore.org andmekvaliteet ei ole rahuldav. Kolmandaks asjaolu, et valdkonna andmekvaliteedi parandamiseks ei ole kirjanduses välja pakutud ammendavaid juhised.

Töös kirjeldatakse põhjalikult andmekvaliteediga seotud probleemi tausta, tutvustatakse ning võrreldakse potensiaalseid anomaaliate tuvastamise meetodeid probleemi lahendamiseks, analüüsitakse andmeid ning testitakse ja valitakse sobivam valideerimismeetod.

Magistritöö olulisi tulemusi on kolm. Esiteks, pakutakse välja sobiv meetod anomaaliate tuvastamiseks. See tõstab andmekvaliteeti Marinexplore.org sensorandmetes ja lühendab oluliselt andmete valideerimisprotsessi pikkust. Teiseks, näidatakse et ookeanis asub stabiilne kiht 1 500 meetrit veepinnast allpool. Seda saab kasutada referentsina - selle alusel saab hinnata sensori seisundit. Kolmandaks, pakutakse välja mõistlik rakendusmudel disainitud lahendusele.

Välja pakutud lahendust ning selle rakendusmudelit on võimalik implementeerida lisaks Marinexplore.org platvormile ka teistel sensorandmetega tegutsejatel.

Töö on kirjutatud inglise keeles ning sisaldab teksti 99 leheküljel, 6 peatükki, 36 joonist, 9 tabelit.

List of Abbreviations

ANN	***Artificial neural networks***	A class of learning methods, simplified models of the central nervous system.
AUC	***Area under the ROC Curve***	A way to summarize classifier’s performance in a single number.
BIRCH	***Balanced Iterative Reducing and Clustering Using Hierarchies***	An unsupervised hierarchical clustering algorithm.
DAC	***National Data Assembly Center***	Organization, which collects data from regional ocean observing systems, quality controls the data, and distributes it via the Global Telecommunications System in realtime.
DBSCAN	***Density-Based Spatial Clustering of Applications with Noise***	A data clustering algorithm.
FNR	***False Negative Rate***	An error result in which a test result improperly indicates that the result is negative, when in reality it is positive.
FPR	***False Positive Rate***	An error result in which a test result improperly indicates that the result is positive, when in reality it is negative.
GARS	***Global Area Reference System***	A standardized geospatial reference system.
kNN	***k-Nearest Neighbors***	A proximity-based classification technique.
LOF	***Local Outlier Factor***	

	An algorithm for finding anomalous data points by measuring the local deviation of a given data point with respect to its neighbours.
MGRS	***The Military Geographic Reference System***
NN	***Neural network*** See ANN.
PNN	***Probabilistic neural network*** A feedforward neural network.
ROC	***Receiver Operating Characteristic*** A visual tool for comparing two classification models; the display gives the measure of the predictive accuracy of a model.
SVM	***Support Vector Machine*** A geometric method of separating two classes by finding the best hyperplane that puts one class above it and other below.
TNR	***True Negative Rate*** A result in which a test result properly indicates that the result is negative, when in reality it is negative.
TPR	***True Positive Rate*** A result in which a test result properly indicates that the result is positive, when in reality it is positive.

List of figures

List of tables

Table of Contents

1. Introduction

This thesis deals with the problem of outlier detection from oceanographic spatiotemporal data. We present a comprehensive model for the application; we compare and choose the most appropriate outlier detection methods for the current domain.

In this chapter, we define the outlier detection, discuss the importance of this field, and present examples of outliers. We align the terminology used in this thesis. Finally, we present the motivation for the master thesis, define problems and hypothesis in the current domain, and set up objectives of this thesis.

1.1 Introduction

The main focus of this work is the detection of anomalous data points in multidimensional datasets. An outlier is an observation point that is distant from other observations. An outlier may be a result of a measurement error or an erroneous sensor reading; outliers may arise from changes in system behavior or from human error.

We are focused on the sensor data produced by the ARGO program[1]. ARGO is a global array of more than 3 000 free-drifting profiling floats that measures the temperature and the salinity of the upper 2 000 m of the ocean. A float is displayed in Figure 1.

According to Chandola [2] and Han [3] there are several factors that make this apparently simple outlier detection approach very challenging:

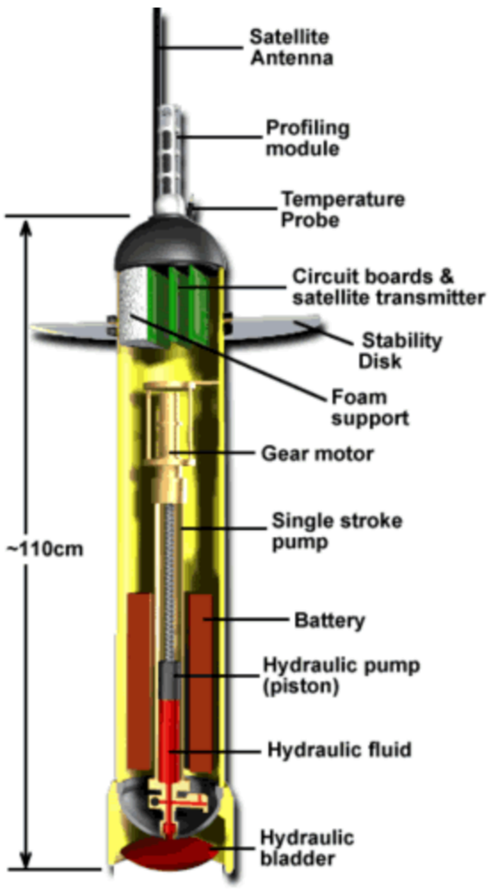


Figure 1: ARGO float [1].

[1] Disclaimer: "A part of these data were collected and made freely available by the International Program and

- The boundary between normal and anomalous behavior is often not precise. Thus an anomalous observation, which lies close to the boundary, can be normal, and vice-versa.

- When anomalies are the result of malicious actions, the malicious appear normal, thereby making the task of defining normal behavior more difficult.

- In many domains, normal behavior keeps evolving and a current notion of normal behavior might not be sufficiently representative in the future.

- The exact notion of an anomaly is different for different application domains. For example, in the medical domain a small deviation from normal (e.g., fluctuations in body temperature) might be an anomaly, while similar deviation in the stock market domain (e.g., fluctuations in the value of a stock) might be considered as normal. Thus applying a technique developed in one domain to another is not straightforward.

- Availability of labelled data for training/validation of models used by anomaly detection techniques is usually a major issue.

- Often the data contains noise, which tends to be similar to the actual anomalies and hence is difficult to distinguish and remove.

So, the main problem is that it is difficult to find appropriate data models to detect outliers.

There is no single best method for outlier detection, and success depends not only on the type of method used but also the nature of data handled. Different problems often have few similarities, and there is no unifying theory available.

The main goal of the research is to investigate outlier detection methods and algorithms that would be suitable for use in sensor data. Outlier detection is a widely explored problem in many fields. There is a lot of literature about this topic (typically targeted towards on some specific task or problem), but it is difficult to compare these algorithms objectively and decide which are suitable for a specific task, problem or domain.

This project helps us to improve the data quality of sensor data in Marinexplore.org; it also helps to shorten the validation period (current length of 6-12 months).

1.2 Background

Improving the data quality involves a broad range of knowledge, namely databases, data analysis, data mining, statistics, machine learning, business analysis, and computer science.

In this chapter, we present relevant background information on anomaly detection and the related topics.

1.2.1 Marinexplore.org [4]

Marinexplore.org is the biggest free Internet portal for public data in the oceanic, atmospheric and climate domain. It provides powerful search, exploration and visualization tools to access in-situ, raster and model datasets from 33 organizations (global, regional and local) from nearly 40 000 in-situ devices and 50 data products, with over 7 700 professionals registered in the community.

There is no raw data quality control procedure implemented by the Marinexplore.org data engineers yet (raw data, AS IS).

This is where automated outlier detection functionality could improve their data quality (in the context of sensor data collected by the ARGO program).

1.2.2 The ARGO program

ARGO is a global array of more than 3 000 free-drifting profiling floats that measures the temperature and the salinity of the upper 2 000 m of the ocean. This allows continuous monitoring of the temperature, the salinity, and the velocity of the upper ocean (sensor data), with all data being relayed and made publicly available within hours after collection.

Most of these floats stay submerged at 1 500 - 2 000 m of depth for about 9 - 10 days, then move up through the water column to take a vertical profile measurement. After taking the measurement, they return to their 'sleeping' depth and drift until the time for the next measurement series comes.

Figure 2 illustrates ARGO cycles.

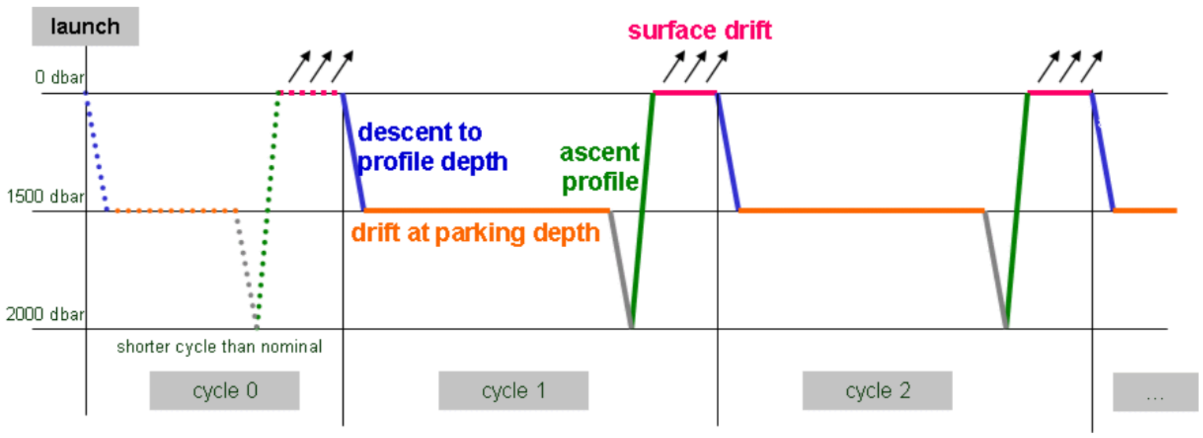


Figure 2: ARGO cycles illustrated.

Since the 2000's, the physical state of the upper ocean is systematically measured, and the data assimilated in near real-time into computer models. The ARGO broad-scale global array of temperature/salinity profiling floats has already grown to be a major component of the ocean observing system. The first ARGO floats were deployed in late 1999, and by November 2012, ARGO has collected its one millioneth profile – double the number obtained by research vessels during all of the 20th century.

Figure 3: An ARGO float make a splash in the Indian Ocean [5].

Global coverage is essential, but for global change applications, ARGO data must also have high accuracy and minimal systematic errors. Therefore, a high priority for ARGO is to continue work aimed at identifying and correcting pressure measurement errors, especially those with systematic impacts.

Figure 4 shows locations of 3 000 randomly chosen positions in waters deeper than 2 000 m (according to the ARGO Science Team, led by Roemmich).

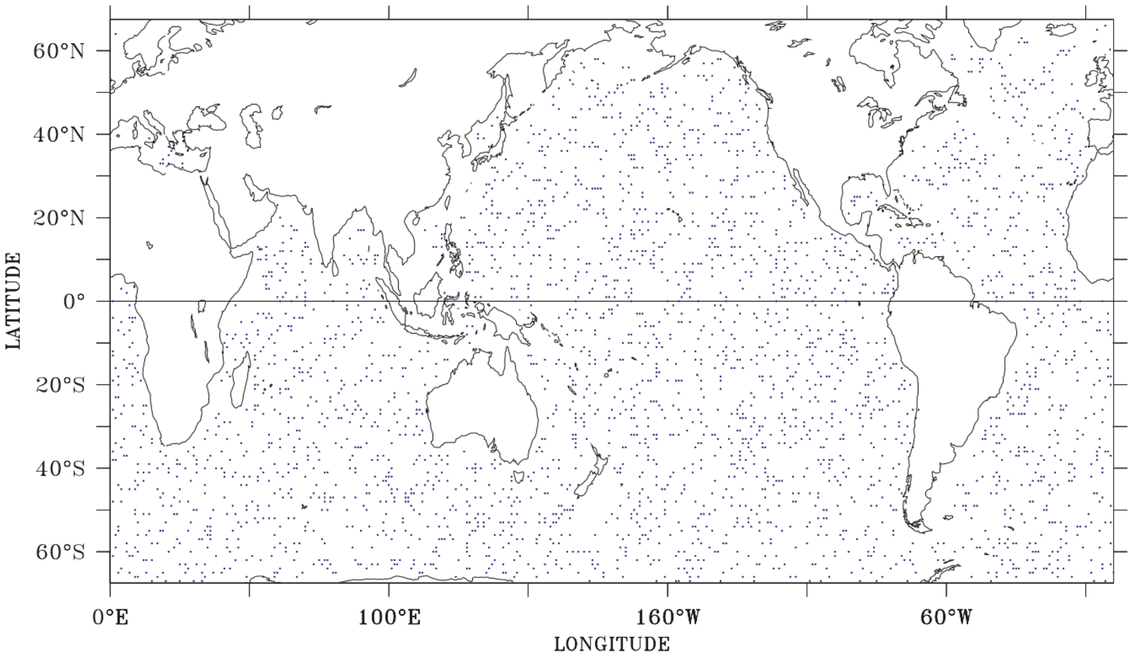


Figure 4: ARGO float coverage (2015) [6].
Salinity is the primary diagnostic variable in the hydrologic cycle [6].

ARGO data is in a wide range of applications: climate and seasonal forecasting, weather (e.g. hurricanes) forecasting, marine safety, maritime transportation, fishery management, offshore industry, and defense [7].

The Gray List (maintained by ARGO) contains floats, which may have problems with one or more sensors [8].

We compiled the Gray List data into a graph (TOP 25 failures, the snapshot from 18.04.2015), and the result is shown in Figure 5.

According to the statistics about known errors, we see that most problems are related to sensor malfunction or drifts. A frozen profile error means that a float repeatedly produces the same temperature or salinity profile (with very small deviations).

The Float Failure Analysis [9] summarises that failures are caused by environmental issues (loss due to grounding and ice), hardware issues (motor backspin, leaking air bladder, failed potentiometer, transmissometer leak), software issues (firmware bugs), and human errors (floats turned on too early, picked up by fishermen).

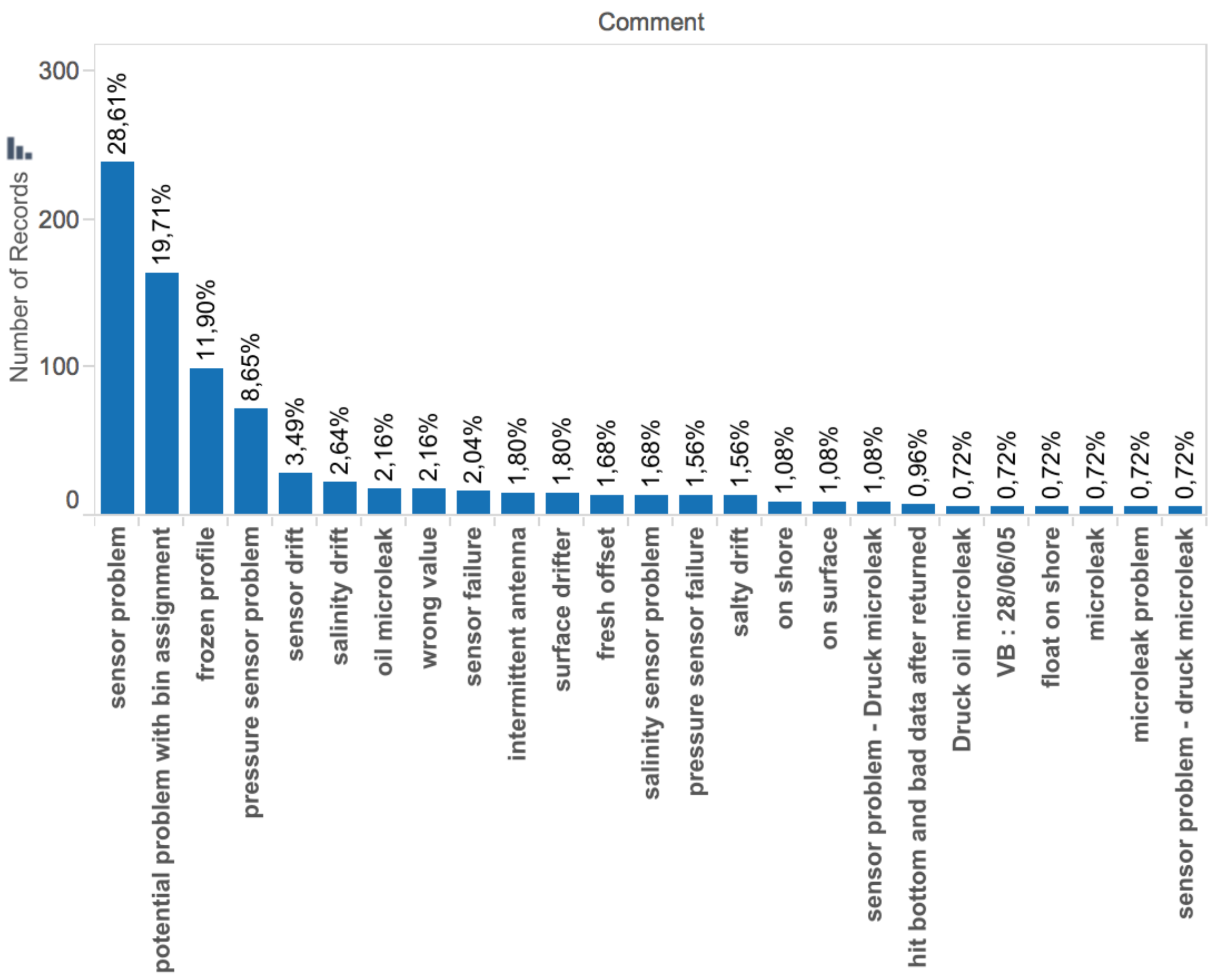


Figure 5: TOP 25 failures (April 2015).

Roemich [6] expresses that the stable salinity sensors for 4-year and longer missions are problematic.

The ARGO data system has three levels of quality control [10]:

- The first level is the real-time system that performs a set of agreed automatic checks on all float measurements. Real-time data with assigned quality flags are available to users within a 24 hr to 48 hr timeframe. These controls are limited and automatic.

- The second level of quality control is the delayed-mode system performed by domain experts.
- The third level of quality control is the regional analysis of all float data.

Since the delayed-mode quality control is a labor intensive job, and it takes 6 to 12 months in data centers in France and the USA, there is a need to significantly shorten the long validation period. This project explores the potential in choosing a suitable method to detect outliers from time-series oceanographic data. That is the main motivation to work on this challenging project.

1.2.3 Outliers

Jiawei Han [3], Charu C. Aggarwal [11] and Robert S. Witte [12] define an outlier as a data point, which is significantly different from the remaining data. Outliers are also defined as deviations from predicted values.

At its broadest, the outlier detection is a classification problem - data points must be assigned to one of a set of classes on the basis of observed attributes or features.

Aggarwal [11] mentions that outliers are also referred to as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants in different application domains and data mining and statistics literature. Of these, anomalies and outliers are two terms most commonly used in the context of anomaly detection; sometimes interchangeably.

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior [3].

Anomaly detection finds extensive use in a wide variety of applications such as fraud detection for credit cards, insurance or health care, intrusion detection for cyber-security, fault detection in safety critical systems, and military surveillance for enemy activities [2].

A straightforward anomaly detection approach, therefore is to define a region representing the normal behavior and declare any observation in the data, which does not belong to this normal region as an anomaly.

In most applications, the data is created by one or more generating processes (it is time-series data, a sequence of data points made over a time interval). Examples of time series data are sea tides, counts of sunspots, and the daily closing value of the Dow Jones Average. Time series data is very frequently plotted via line charts, which could either reflect activity in the system or observations collected about entities. When the generating process behaves in an unusual way, it results in the creation of outliers [11]. Instrument errors, hardware malfunctions, calibration issues, human errors, physical damages, pollution events, etc. may cause anomalies.

It is worth noting that outliers (novelties) are often trendsetters. Therefore, at a later stage, similar data points may no longer be considered outliers (novelties), but may become a normal part of data. However, significant changes from trends continue to be important [11].

Han [3] and Aggarwal [11] divide the outlier detection in a time-series into two categories. This depends on whether the values at specific time stamps are classified as outliers because of sudden changes (contextual anomalies), or whether entire time-series or large subsequences within a time series are classified as outliers because of their unusual shapes (collective or behavioral anomalies).

In this thesis, our main focus will be on finding contextual anomalies.

Aggarwal [11] points out that the core principle of discovering outliers is based on assumptions about the structure of the normal patterns in a given dataset. See the characteristics of our datasets in Chapter 3.3.

Let us explain the difference between the sudden changes in data streams and the trends in the data streams, which change slowly over time.

Aggarwal [11] notes that time-series data contains a set of values, which are typically generated by continuous measurement over time. Therefore, the values in consecutive timestamps do not change very significantly, or change in a smooth way. He also points out that in such cases, therefore, the sudden changes in the underlying data records can be considered anomalous events. Therefore the discovery of anomalous points in time series, is usually closely related to the problem of anomalous event detection (e.g. sensor malfunction).

Let us consider the following time-series of values, along with the corresponding timestamps implicitly defined by the index of the data point. Figure 6 illustrates the example.

$$3, 2, 3, 2, 3, 87, 86, 85, 87, 89, 3, 84, 91, 86, 91, 88$$

There is a sudden change in the data value at timestamp 6 from 3 to 87. This corresponds to an outlier. The data stabilizes at this value, and this becomes a new normal. At timestamp 12, the data value again dips to 3. Even though this data value was encountered before, it is still considered an outlier because of the sudden change in the consecutive data values. It is critical to understand that in that case, treating the data values independent of one another is not helpful for anomaly detection, because the data values are highly influenced by the adjacent values of the data points. Aggarwal [11] concludes that the problem of outlier detection in time-series data is highly related to the problem of change detection.

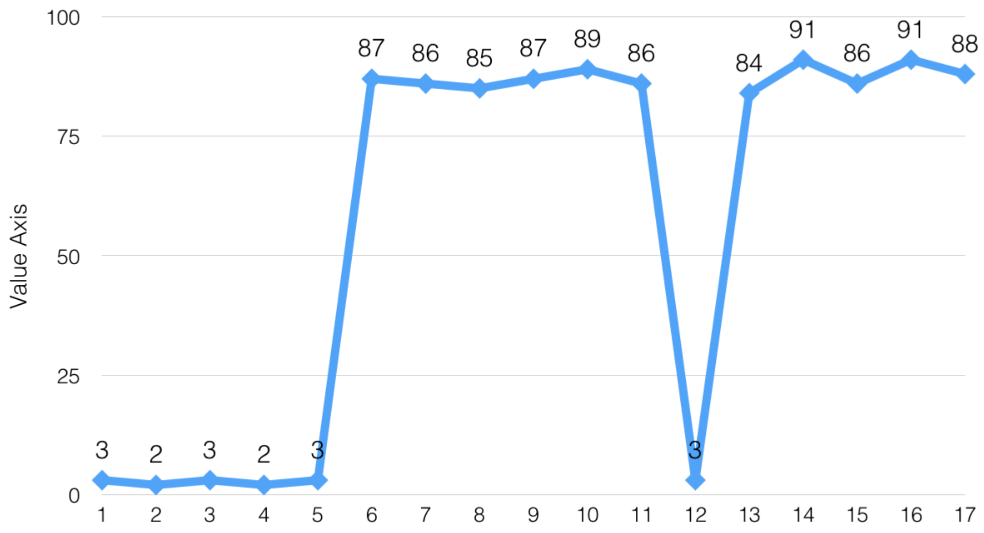


Figure 6: Time-series example illustrated [11].

The values and trends in the data stream change can only be detected slowly over time, a phenomenon, which is referred to as concept drift. The concept drift can only be detected by detailed analysis over a long period. This scenario does not necessarily correspond to outliers. These can be discovered as deviations from forecasted values using window-based analysis.

We are more interested in finding specific sudden changes in data-streams. Such anomalies are also referred as point anomalies.

In figure 7, the anomaly is defined as a missing value (at timestamp 3). It is referred to as a gap.

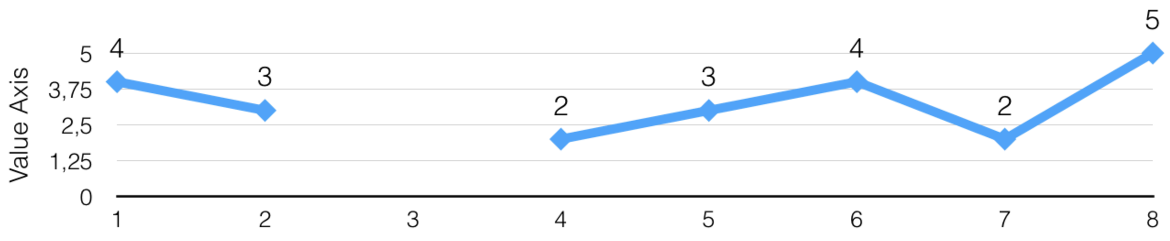


Figure 7: Missing value illustrated.

We note that a high or a low value by itself is not always an anomaly. If we ignore the temporal aspect, the anomaly cannot be detected.

1.2.4 Spatiotemporal data

Spatiotemporal data is a generalization of both spatial and temporal data [13].

Sometimes data points are related to each other temporally and spatially, so the expected values of data points are influenced by their contextual dependencies, and therefore outliers are defined on the basis of such contextually modeled deviations.

In these multidimensional data streams changes in the aggregate distribution of the streaming data may correspond to unusual points [11].

In an ocean dataset, for example, each sample could contain readings taken at a particular time in a particular location. These readings could include seawater temperature, seawater pressure, seawater salinity, seawater electrical conductivity, etc.

Depending on the method, there exists a need to test data points independently of one another.

1.3 Glossary

As several of the terms used in the thesis may have a number of different meanings, we define the terms used throughout this thesis below. This is important to avoid misunderstandings if multiple definitions for one term exist.

Anomaly	A data point, which is significantly different from the remaining data.
Classifier	An algorithm that implements classification.
Clustering	A task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups.

Correlation	A statistical measure of how two intances move in relation to each other.
Gap	A break in continuity.
Instance	An object is an instance of a class.
Labelling	To describe an intance in a short phrase (VALID/INVALID).
Outlier	A very extreme score. See 'anomaly'.
Noise	A difference between noise and anomalies is not pure. Weak outlier.
Scalability	An ability of a system to accommodate the growth.
Spike	A point, which is more than M times the standard deviation from series mean.
Validation	A confirmation that a product or service meets the requirement.

1.4 The importance of outlier detection, examples

Outlier detection is important in many fields. The importance of outlier detection is due to the fact that outliers in data translate to significant and often critical actionable information in a wide variety of application domains [2].

A list of applications that utilize outlier detection:

- Mortgage and loan application processing - detecting potential problematic applications.
- Monitoring aircraft engine performance - analyzes the health of aircraft engine to avoid costly repairs. Any deviation from the typical or normal system behavior is potentially anomalous.
- Anomalous heartbeat pulses using Electrocardiogram (ECG) data from the patient, periodicity is an important aspect here, differences from the rest of the sequence (a gap or an extreme value).
- Image analysis - detecting novelties for surveillance systems.
- Unexpected entries in databases. Data cleansing procedures. To reduce potential human errors.
- Road safety cameras - detecting traffic regulation violations, including speeding, vehicles going trough a red traffic light, unauthorized use of a bus and taxi line.

- Vessel traffic/flight operation monitoring to identify possible unsafe situations. Maritime surveillance. Flight traffic surveillance.

Chandola [2], Han [3], Aggarwal [11], Hastie [14] present some extra examples:

- Intrusion detection systems. In many host-based or networked computer systems, different kinds of data are collected from the operating system calls, network traffic or other activity in the system. This data may show unusual behavior because of malicious activity.

- Credit card fraud. Credit card fraud is quite prevalent, because of how easily sensitive information such as a credit card number, can be compromised. Different unusual patterns can be used to detect outliers in credit card transaction data.

- Interesting sensor events. Sensors are often used to track various environmental and location parameters in many real applications. The sudden changes in the underlying patterns may represent events of interest.

- Industry damage detection.

- Mobile phone fraud detection. The task is to scan a large set of accounts, examining the calling behavior of each, and to issue an alarm when an account appears to have been misused.

- Insider trader inspection. Insider trading can be detected by identifying anomalous trading activities in the market.

- Medical diagnosis. In many medical applications, the data are collected from a variety of devices. Unusual patterns in such data typically reflect disease conditions.

In all these applications, the data has a “normal” model, and anomalies are recognized as deviations from the normal model.

1.5 Outlier definition

As stated earlier, a straightforward anomaly detection approach is to define a region representing the normal behavior and declare any observation in the data which does not belong to this normal region as an anomaly.

First, we are using extreme values to define a data point as an outlier, so, we define outliers as data points that have values that are either too large or too small.

Second, we treat a data point as an outlier if there is a significant deviation from the typical distribution.

Third, gaps are interesting to us as well, and they may refer to missing data values.

In this thesis the terms "outlier" and "anomaly" are used interchangeably.

1.6 Problem definition

1.6.1 Problem

The first problem is that the quality control process is too long in the ARGO program - it takes 6 to 12 months.

The second problem is that in the Marinexplore.org databases the sensor data from floats is AS IS (due to the first problem). There is a need to improve the data quality. Sensor errors are common in this field.

The third problem is that there is no single method for outlier detection and no instruction to choose the method for a specific task or problem. There is no unifying theory available. Applying a technique developed in one domain to another is not straightforward.

1.6.2 Research objectives

This project aims to improve the quality of the Marinexplore.org machine generated sensor data. As the delayed quality control takes too long, we research possibilities for a faster process.

For this purpose, we select a method that could detect anomalous data points in sensor data that come from ARGO floats. We provide an approach to improve the data quality in this domain.

1.6.3 Research hypothesis

- There is no single best method or algorithm ideally suited to detect outliers. It depends on the nature of dataset (the existence of training or labelled data, dataset size, etc.).
- There exists a stable waterfront (e.g. deepwater) in oceans from where we can find a reference data for methods or data for sensor "calibration".
- It is possible to design tests and model validation methods to improve and shorten the data quality process in this domain.

1.6.4 Research approach

In order to facilitate the comparison of different methods and algorithms, we present theoretical aspects first. This helps us to pre-select the methods with most potential that we will later test.

Objectivity is important to us, so, in order to find answers to our questions we will test our pre-selected methods and algorithms in our test environment. We are designing our tests based on real datasets from Marinexplore.org.

To evaluate the performance over multiple datasets, we are using several metrics (including a receiver operating characteristic - ROC - curve analysis and the Friedman test). It gives us an objective way to choose the best method for the current domain.

1.7 Related works

Due to the increase in sensor data generation, outlier detection has attracted the interest of researchers, experts and scientists in the analysis of outliers.

1.7.1 StatLog [15]

The EU funded project, from the early 1990s aims to: 1) review different approaches to classification, 2) compare their performance on a wide range of challenging datasets, 3) draw conclusions on their applicability to realistic industrial problems, 23 Classification Algorithms, 20 Datasets.

The StatLog project was designed to select classification procedures regardless of historical pedigree, testing them on large-scale and commercially important problems, and hence determine to what extent the various techniques met needs of industry [16].

This depends critically on a clear understanding of [16]:

- the aims of each classification/detection procedure;
- the class of problems for which it is most suited;
- measures of performance or benchmarks to monitor the success of the method in a particular application.

StatLog was a large-scale study when it was performed, but since then new learning algorithms have arisen that have better performance. A new similar evaluation of modern learning methods would be useful.

1.7.2 "An Empirical Comparison of Supervised Learning Algorithms" [17]

Caruna has published an academic journal article "An Empirical Comparison of Supervised Learning Algorithms." He presents large-scale empirical comparison between 10 supervised learning methods: SVMs, neural nets, logistic regression, Naïve Bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps. He used eight different performance criteria: accuracy, F-score, Lift, ROC Area, average precision, precision/recall, break-even point, squared error, and cross-entropy. An important aspect of his study is the use of performance criteria to interpret model predictions as probabilities.

1.8 Outline

This thesis is organized as follows:

- In Chapter 1, a broad background of this project is explained. This includes data quality issues in Marinexplore.org, the nature of sensor data, characteristics of time-series data, etc.

- Chapter 2 introduces methods, algorithms, and techniques. Theoretical aspects are discussed.
- Chapter 3 describes the dataset, pre-processing, initial quality control procedures.
- Chapter 4 demonstrates our experimental results, and we give a detailed overview of analysis we have completed. Chapter 4 also makes comparisons between the methods and algorithms.
- Chapter 5 introduces the comprehensive model of the application.
- Finally, Chapter 6 outlines the results, and concludes the thesis with the key results and makes recommendations for future work. The conclusion also provides some concluding remarks on the limitations of the current work.

2 Theoretical aspects

The discussion in the previous sections provided an understanding on the nature of anomalies (temporal data). This section covers the various outlier detection techniques.

All outlier detection algorithms create a model of normal patterns in the data and then compute an outlier score of a given data point on the basis of the deviations from these patterns. The choice of the data model is crucial. An incorrect choice of the data model may lead to poor results. The best choice of a model is often dataset specific [11].

Thus, we have to analyze and evaluate our typical dataset characteristics carefully before we can choose the suitable algorithm for our particular domain. See Chapter 4.3.

Sherrod [18] notes that methods for analyzing and modelling data can be divided into two groups: supervised learning and unsupervised learning. Supervised learning requires input data that has both predictor (independent) variables and a target (dependent) variable whose value is to be estimated. Decision trees, regression analyses are examples of supervised learning. If the goal of an analysis is to predict the value of some variable, then the supervised learning is the recommended approach.

According to Maimon [19] supervised methods attempt to discover the relationship between input attributes (also referred to as independent variables) and a target variable (dependent variable, or output variable), the relationship discovered is represented in a structure referred to as a model.

Supervised methods require pre-labelled data, tagged as normal or abnormal (sometimes referred to as a bag instance).

According to Hodge [20] classification algorithms require a good spread of both normal and abnormal data, i.e., the data should cover the entire distribution to allow generalisation by the classifier.

Unsupervised learning does not identify a target (dependent) variable, but rather treats all of the variables equally. Cluster analysis, correlation, factor analysis and statistical measures are examples of unsupervised learning.

In Chapter 5 we will provide an approach, which allows us to use pre-labelled data so that we can use both learning techniques - unsupervised and supervised algorithms.

Detecting anomalous data points involves determining whether the tested value is significantly different from the underlying reference population.

2.1 Statistical techniques

Han [3] notes that the effectiveness of statistical methods highly depends on whether the assumptions made for the statistical model hold true for the given data. They assume that the normal objects in a dataset are generated by a stochastic process, normal objects occur in regions of high probability and the objects in the regions of low probability are outliers.

Whitney [21] proposes that the statistical approaches to anomaly detection work well on data with a Gaussian distribution (normal distribution). They are focused on discovering how far data points fall from the mean of the data. Before conducting this type of anomaly detection, it is important to first identify the statistical distribution of the dataset. If it is not a Gaussian distribution, but instead has a distribution with a higher probability of points falling further from the mean, the approach will not be as effective.

As we know (see Chapter 4.2) our data distribution is not always Gaussian, so, it can be complicated to perform all methods below.

The key disadvantage of statistical techniques is that they rely on the assumption that the data is generated from a particular distribution. This assumption often does not hold true, especially for high dimensional real datasets [2].

NB! Using some methods below, the data should be modeled from a normal distribution.

2.1.1 Outlier detection using histogram

We can use the histogram as a statistical model to detect outliers. In Figure 8, the birth weight of more than 1.8 kg can be regarded as an outlier as the value falls outside the bins.

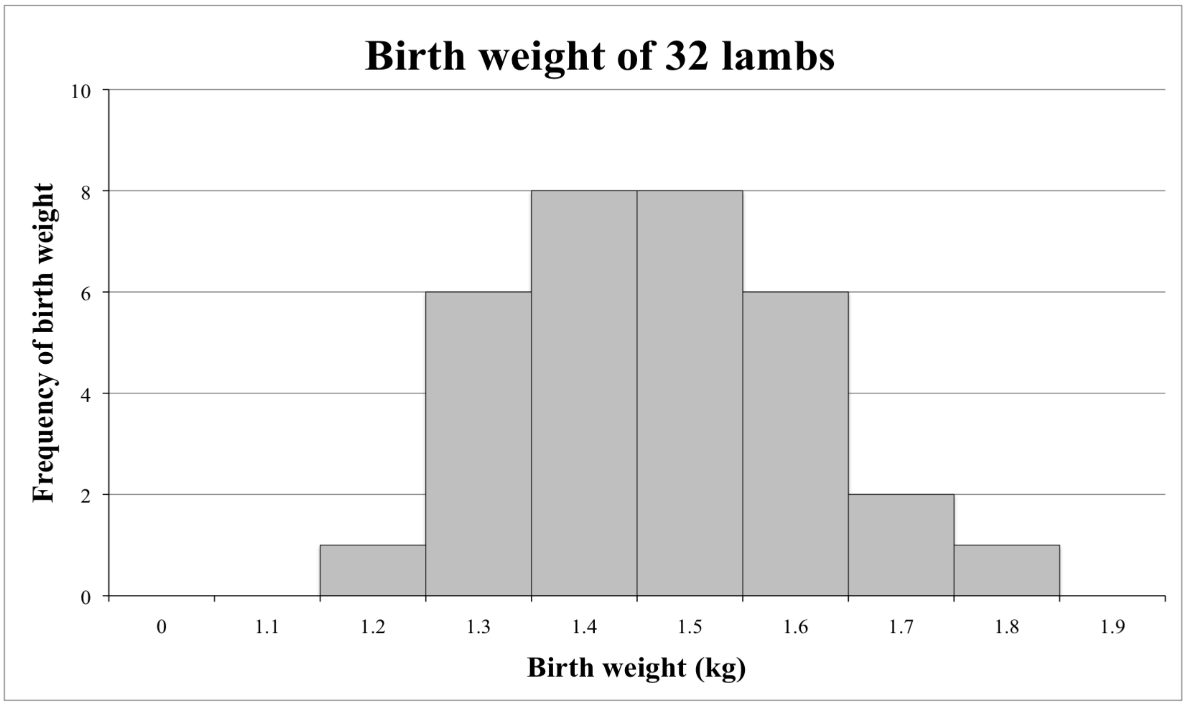


Figure 8: Histogram of birth weights [22].

The x-axis in a histogram represents a scale rather than simply a series of labels, and the area of each bar represents the proportion of values that are contained in that range [23].

To detect whether an instance is an outlier, we can check it against the histogram. In the simplest approach, if the instance falls in one of the bins, the instance is regarded as normal; otherwise it is treated as an outlier.

Han [3] notes that it is hard to choose an appropriate bin size. On one hand, if the bin size is set too small, many normal instances may end up in empty or rare bins, and thus be misidentified as outliers. This leads to a high false positive rate. On the other hand, if the bin size is set too high, outlier instances may infiltrate into some frequent bins and thus be "disguised" as normal, this leads to a high false negative rate.

Next we present some statistical approaches [2] [11] [12] [23] [24].

2.1.2 Excess / Spread statistics

It is a metrics in which the discordancy measure consists of a ratio of the difference between a potential anomaly and its nearest or next-nearest neighbour to the range to as excess / spread statistics. Anomalies are detected if the ratio is in excess of a given reference value.

If $Q > Q_{ref}$, where Q_{ref} is a reference value corresponding to the sample size and the confidence level, then the questionable point is rejected.

$$Q = \frac{(z_n - z_{n-1})}{(z_n - z_1)}$$

2.1.3 Range / Spread statistics

A range is the difference between the largest and the smallest values [12]. These metrics replace the numerator with the sample range. Again, If $Q > Q_{ref}$, where Q_{ref} is a reference value corresponding to the sample size and the confidence level, then the questionable point is rejected. The dominator may also be replaced by a restricted sample analogue, an independent estimate, or a known value of a measure of spread of the population.

Anomalies identified by such statistics do not indicate whether the anomaly is an upper or a lower anomaly, only that the sample is anomalous.

2.1.4 The interquartile range (IQR)

According to this rule outliers are those datapoints lower than 25th-quartile minus $1.5 \times IQR$ or greater than the 75th-quartile plus $1.5 \times IQR$, similarly, extreme outliers are those datapoints lower than 25th-quartile minus $3 \times IQR$ or greater than the 75th-quartile plus $3 \times IQR$ [2] [23].

2.1.5 Sums of squares

The definition of the variance is either the expected value (when considering a theoretical distribution), or the average value (for actual experimental data), of squared deviations from the mean [12].

$$SS_x = \sum_{i=1}^{n} (x_i - \mu)^2$$

μ is the mean of the population, $(x_{i-}\mu)^2$ denotes each of the squared deviation scores, n is the sample size. We are using squaring to eliminate negative signs.

2.1.6 Grubbs' test

Extreme studentized deviation test or maximum normed residual test, is a statistical test used to detect outliers in a univariate dataset assumed to come from a normally distributed population.

This method is applicable only to univariate datasets under the assumption that data is generated by a Gaussian distribution.

According to Chandola [2] the z-score for each test instance x is computed as follows:

$$z = \frac{|x - \mu|}{\sigma}$$

Here μ and σ are the mean and the standard deviation of the data sample, respectively.

A test instance is declared to be anomalous if:

$$z > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\alpha/(2N),N-2}}{N-2+t^2_{\alpha/(2N),N-2}}}$$

Here N is the data size, and $t_{\alpha/(2N),N-2}$ is a threshold used to declare an instance to be anomalous or normal. This threshold is the value taken by a t-distribution at a significance level of $\frac{\alpha}{2N}$. The significance level reflects the confidence associated with the threshold and indirectly controls the number of instances declared as anomalous.

Hodge [20] explains: “It calculates a z value as the difference between the mean value for the attribute and the query value divided by the standard deviation for the attribute where the mean and the standard deviation are calculated from all attribute values including the query value. The z value for the query is compared with a 1% or 5% significance level. The technique requires no user parameters as all parameters are derived directly from data.”

2.1.7 Z-value test

Boslaugh [23] defines a Z-value as the distance of a datapoint from the mean, expressed in units of standard deviation.

The formula is defined as follows:

$$Z_i = \frac{|X_i - \mu|}{\sigma}$$

The Z-value for the data point X_i is denoted by Z_i, μ is denoted as mean, and σ is denoted as standard deviation.

The Z-value test computes the number of standard deviations by which the data varies from the mean, so it transforms a raw value into units of deviation above or below the mean. A good "rule of thumb" is to use $Z_i \geq 3$ as a proxy, for the anomaly.

Where smaller number of samples is available, the results from the Z-value test need to be interpreted more carefully [11].

2.1.8 Extreme location

Another class of test statistics used to test for discordancy are those which take the form of ratios of extreme values to measures of location. Anomalies can be identified if $Q > Q_{ref}$, where Q_{ref} is some measure of spread or deviation of the data.

The formula is defined as follows:

$$Q = \frac{z_n}{\mu}$$

μ is the mean value over data.

Han notes [3] that statistical methods learn from models of data to distinguish normal data instances from outliers. The statistical assumption made about the underlying data meets the constraints in the reality. The distribution of high-dimensional data is often complicated. Thus, statistical methods for outlier detection on high-dimensional data remain a big challenge.

In spite of these reasons, we can use these presented methods to pre-test our datasets. Our initial quality checks are based on these methods. See Chapter 3.4.

2.2 Clustering and classification techniques

2.2.1 k-Means

According to Hastie [14] k-Means clustering is a method for finding clusters and cluster centers in a set of unlabelled data. k-Means is a predictive modeling method, only for classification. Sherrod [18] notes that k-Means is a relatively fast method, but it is also among the least accurate methods. It is assumed that the entire dataset can be accommodated in the main memory.

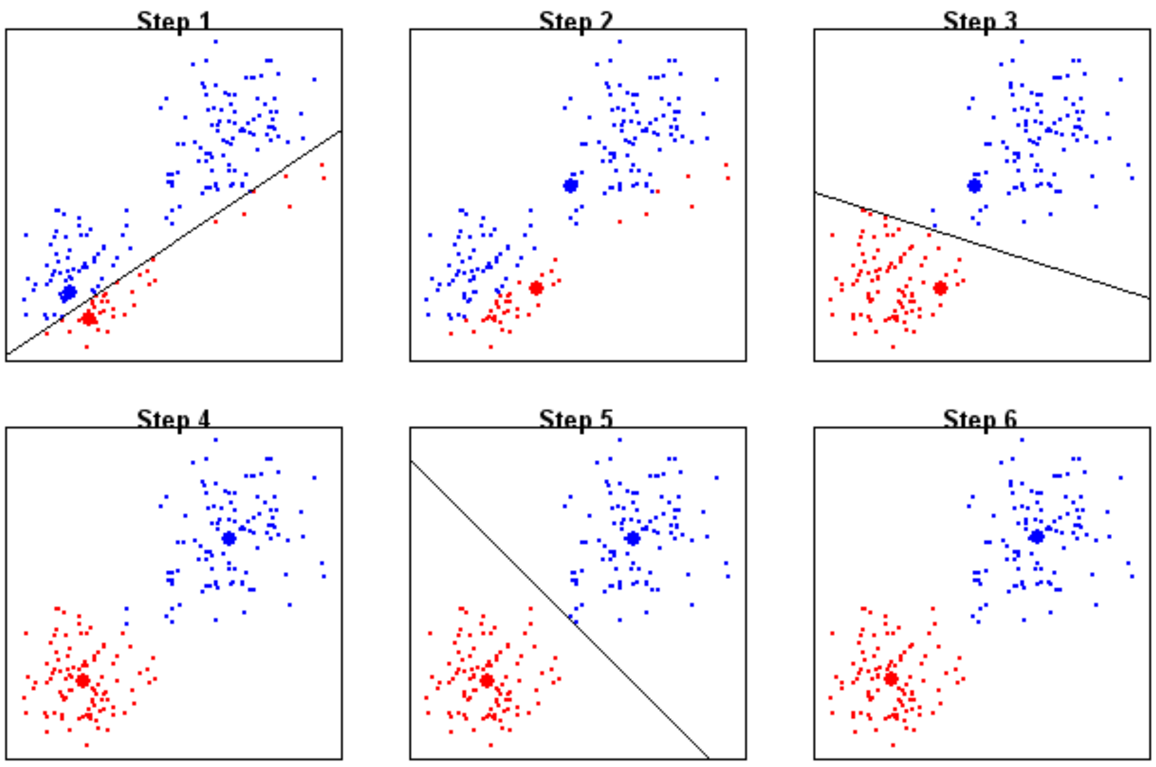


Figure 9: The basic idea of the k-Means clustering [25].

O'Neil and Schutt [26] and Han [3] explain how the algorithm works:

- Initially, you randomly pick k centroids (or points that will be the center of your clusters) in d-space (d is the number of features of each data point). Try to make them near the data but different from another.
- Then assign each data point to the closest centroid.
- Move the centroid to the average location of the data points assigned to it.
- Repeat the preceding two steps until the assignments don't change or change very little.

So, the basic idea of k-Means clustering is that clusters of items with the same target category are identified, and predictions for new data items are made by assuming they are of the same type as the nearest cluster center [26].

They also point out that k-Means has some known issues:

- Choosing k is the number more an art than a science, although there are bounds: $1 <= k <= n$, where n is the number of data points.

- There are convergence issues - the solution can fail to exist, if the algorithm falls into a loop, for example, and keeps going back and forth between two possible solutions, or in other words, there isn't a single unique solution. Interpretability can be a problem - sometimes the answer isn't at all useful. That's often the biggest problem.

k-Means requires the user to specify the value of clusters in advance.

k-Means is an unsupervised learning algorithm, the labels are not known and are instead discovered by the algorithm.

O'Neil and Schutt [26] note that in spite of these issues, it's pretty fast (compared to other clustering algorithms). Maimon [19] also notes that k-Means performs well in terms of execution time.

Maimon [19] proposes that its time complexity is $O(m \times k \times l)$, where m is the number of instances, k is the number of clusters and l is the number of iterations taken by the algorithm to converge (typically k and l are fixed in advance, and the algorithm has a linear time complexity in the size of dataset), the space complexity is $O(k + n)$, this means that the algorithm needs additional space to store the data.

2.2.2 k-Nearest Neighbors

k-Nearest Neighbors (kNN) is a proximity-based classification technique. Chen [27] and Han [3] note that it is also called the "lazy learner", because it stores all training samples, but does not build a classifier until a new sample needs to be classified.

The idea is simple: normal data instances occur in dense neighborhoods, while anomalies occur far from their closest neighbors [2].

The distance of each data point to its k-th nearest neighbor is determined. By picking a value of $k > 1$, small groups of points, which are close together, but far away from remaining dataset are also treated as outliers. It is reasonable to treat such sets of data points as outliers, because small related sets of points can often be generated by an anomalous process [11].

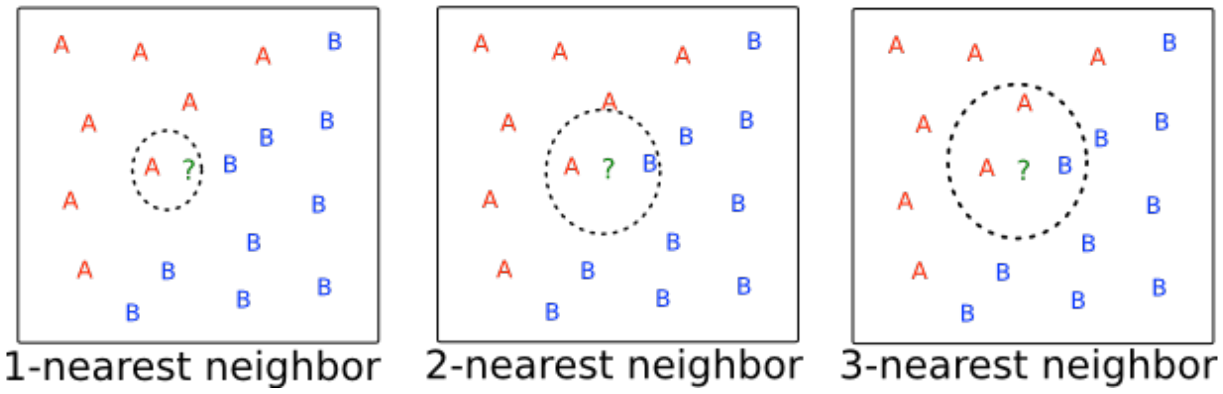


Figure 10: The basic idea of the kNN clustering [21].

Aggarwal [11] and Giudici [28] point out that this method is computationally expensive, because it is required to determine the k-th nearest neighbour of every point in the dataset. It can require $O(n^2)$ time for a dataset containing n points.

As the method is based on the calculation of the distances between all records, there is an exponential computational growth. Flach [29] and Han [3] note that for large datasets, the method is very time consuming for $k > 1$ since all the training data must be stored and examined for each classification.

2.2.3 Local Outlier Factor

Local outlier factor (LOF) is an algorithm for finding anomalous data points by measuring the local deviation of a given data point with respect to its neighbours [3].

Aggarwal [11] defines the LOF as a quantification of the outlierness of the data points, which can adjust for the variations in the different densities.

Due to the local approach, LOF is able to identify outliers in a dataset that would not be outliers in another area of the dataset. For example, a point at a "small" distance to a very dense cluster is an outlier, while a point within a sparse cluster might exhibit similar distances to its neighbors.

According to Chandola [2] the cost of the LOF technique is order of $O(n)$, where n is the total number of data points. He also notes that several techniques have been proposed that improve the efficiency of LOF.

2.2.4 BIRCH

BIRCH is an acronym for “Balanced Iterative Reducing and Clustering Using Hierarchies.”

BIRCH begins by partitioning objects hierarchically using tree structures, where the leaf or low-level nonleaf nodes can be viewed as “microclusters” depending on the resolution scale. It then applies other clustering algorithms to perform macroclustering on the microclusters [3].

A hierarchical scalable unsupervised clustering method works by grouping data objects into a tree of clusters [27].

An advantage of BIRCH is its ability to incrementally and dynamically cluster incoming, multi-dimensional metric data points in an attempt to produce the best quality clustering for a given set of resources (memory and time constraints). In most cases, BIRCH only requires a single scan of the database [30].

According to Hodge [20] the cost of the BIRCH algorithm is an order of $O(n)$, where n is the total number of data points. So, here we have a good scalability of the algorithm with respect to the number of points, insensibility of the output order, and the good quality of clustering data.

2.2.5 Hierarchical clustering

While clustering methods are unsupervised learning methods, users are required to set some parameters for these methods. These parameters vary from one method to another, but most clustering methods require a parameter that specifies the number of clusters. This means that these algorithms may be impractical for real-world data.

According to Han [3], a hierarchical clustering method works by grouping data objects into a hierarchy or “tree” of clusters.

Provost [31] outlines that hierarchical clustering is formed by starting with each node as its own cluster. Then clusters are merged iteratively until only a single cluster remains. The clusters are merged based on the similarity or distance function chosen.

Han [3] and Linoff [32] note that hierarchical clustering methods can encounter difficulties regarding the selection of merge or split points. Such a decision is critical, because once a group of objects is merged or split, the process at the next step will operate in the newly generated clusters. Moreover, the methods do not scale well, because each decision of merge or split needs to examine and evaluate many objects or clusters.

Maimon [19] notes that the complexity of hierarchical clustering is $O(n^3)$.

2.2.6 DBSCAN

DBSCAN is an acronym for “Density-Based Spatial Clustering of Applications with Noise”.

DBSCAN is a nearest-neighbour-based clustering algorithm that automatically determines the number of clusters.

Chen [27] presents DBSCAN as an algorithm, where cluster objects based on the notion of density clusters can grow according to the density of neighborhood objects.

Maimon [19] proposes that discovery of clusters of arbitrary shapes is efficient for large spatial datasets. The algorithm searches for clusters by searching the neigbourhood of each object in the dataset and checks if it contains more than the minimum number of objects.

Thus, it is a density-based clustering algorithm: 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 lie alone in low-density regions (whose nearest neighbors are too far away) [33].

Hodge [20] notes that DBSCAN has two user-specified parameters, one parameter specifies the number of neighbourhood is generally set to 4 but the other parameter (the neighbourhood distance) requires $O(n \times log\, n)$ time to calculate. So, DBSCAN has a running time of $O(n \times log\, n)$.

2.3 Machine learning techniques

Flach [29] defines machine learning as a systematic study of algorithms and systems that improve their knowledge or performance with experience.

Negnevitsky [34] defines machine learning as an approach that involves adaptive mechanisms that enable computers to learn from experience, learn by example, and learn by analogy.

Outlier detection is the task of learning what is “normal” and determining when an event occurs that differs significantly from the expected normal behavior.

2.3.1 Neural network techniques

A class of learning methods that was developed separately in different fields - statistics and artifical intelligence - based on essentially identical models. The central idea is to extract linear combinations of the inputs as derived features, and then model the target as a nonlinear function of these features. The result is a powerful learning method, with widespread applications in many fields [14].

Patterson [35] notes that artificial neural networks (ANN) are simplified models of the central nervous system. ANN provides the best way to overcome the combinatorial explosion associated when using von Neumann computer architectures.

Groth [36] outlines that the greatest strength of neural networks is their ability to predict accurately outcomes of complex problems.

On one hand, neural networks perform very well as measured by error rate. They seem to provide either the best or next to best predictive performance in nearly all cases – the notable exceptions are the datasets with cost matrices. In terms of computational burden, and the level of expertise required, they are majorly more complex than, say, the machine learning procedures [29], but on the other hand a major drawback of neural networks is that they are very difficult to set up to produce good results (StatLog project conclusion) [15].

ANNs have been shown to be effective in many fields, i.e. pattern recognition, classification, forecasting, mapping tasks.

Hodge [20] notes that neural networks are non-parametric and model-based, they generalise well to unseen patterns and are capable of learning complex boundaries. After training the neural network forms a classifier.

Supervised networks require a pre-classified dataset to permit learning [20].

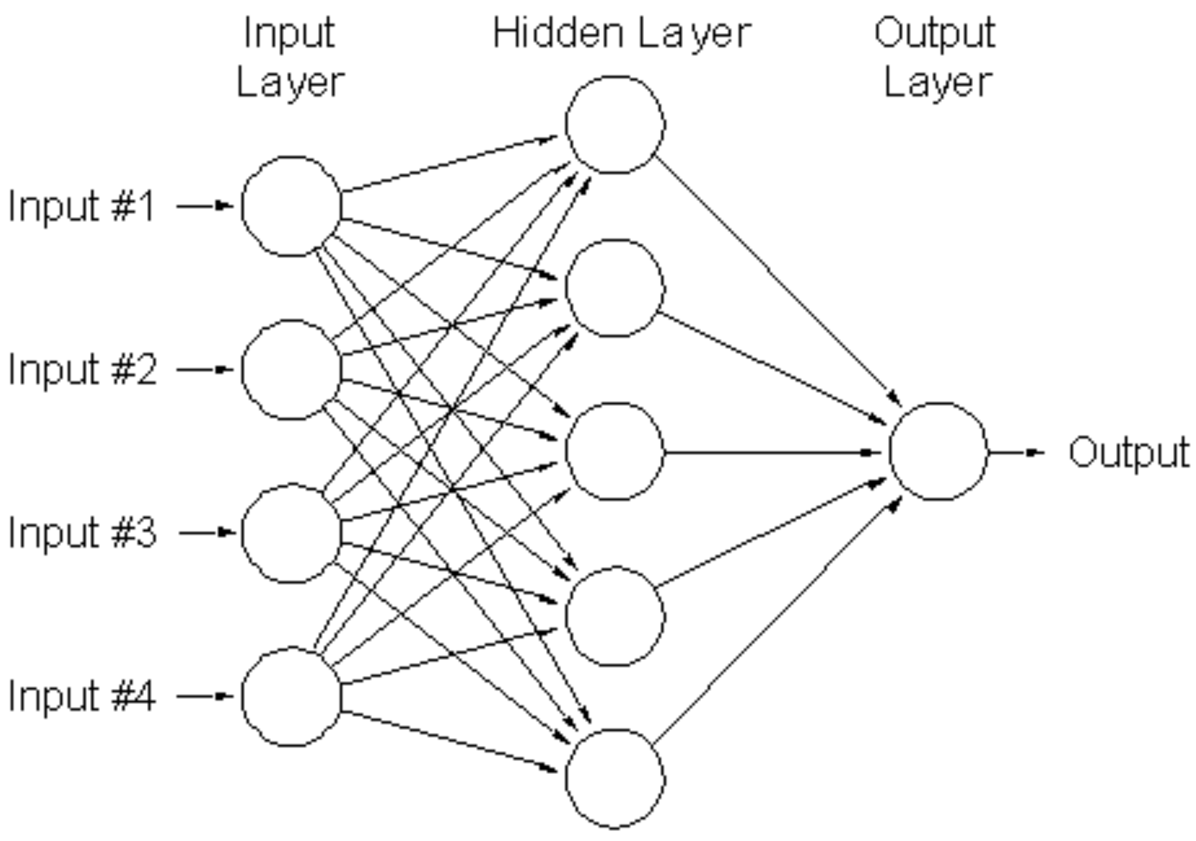


Figure 11: Neural network structure, with inputs, a hidden layer and an output [37].
All neural networks have an input layer and an output layer, but the number of hidden layers may vary [18].

Input layer

A vector of predictor variable values $(x_1 \dots x_p)$ is presented to the input layer. The input layer standardizes these values and distributes the values to each of the neurons in the hidden layer.

Hidden layer

Arriving at a neuron in the hidden layer, the value from each input neuron is multiplied by the weight (W_{ij}), and the resulting weighted values are added together producing a combined value u_j. The weighted sum (u_j) is fed into a transfer function, which outputs value h_j. The outputs from the hidden layer are distributed to the output layer.

Output layer

Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied by the weight (w_{kj}), and the resulting weighted values are added together producing a combined value v_j. The weighted sum (v_j) is fed into the transfer function, which outputs a value y_k. The y values are the outputs of the network.

Sherrod [18] notes that the goal of the training process is to find the set of weight values that will cause the output from neural network to match the actual target values as closely as possible.

According to Sherrod [18] there are several issues involved in designing and training a multilayer perceptron network: selecting how many hidden layers to use in the network, deciding how many neurons to use in each hidden layer, finding a globally optimal solution that avoids local minima, converging to an optimal solution in a reasonable period of time, validating the neural network to test for over fitting.

One of the most important characteristics of a multilayer perceptron network is the number of neurons in the hidden layer. If an inadequate number of neurons is used, the network will be unable to model complex data, the resulting fit will be poor. If too many neurons are used, the training time may become long, and, worse, the network may overfit the data. When over fitting occurs, the network will begin to model random noise in the data.

Hastie [14] also notes that the choice of the number of the hidden layers is guided by background knowledge and experimentation. Sherrod [18] states that there is no theoretical reason for using more than two hidden layers, he recommends designing a three-layer model with one hidden layer.

Thus, these algorithms are usually very demanding on the part of the user. He will have to be responsible for setting up the initial weights of the network, selecting the correct number of hidden layers and the number of nodes in each layer. Adjusting these parameters of learning is often a laborious task [16].

2.3.2 Probabilistic Neural Networks

According to Patterson [35], the probabilistic neural network (PNN) is based on the concept used in classical pattern recognition problems, the PNN models the Bayesian classifier, a technique, which minimizes the expected risk of classifying patterns in the wrong category. One of the main advantages is the speed, which can be trained. In training the PNN, only a single passtrough of the training is needed, thus the decision boundaries can be modified in real-time using new data as it becomes available. One of the main disadvantages of the PNN is that the size of the pattern layer can grow very large when large training sets are used.

Muthukannan [38] points out that:

- PNNs are much faster than multilayer perceptron networks.
- PNNs can be more accurate than multilayer perceptron networks.
- PNN networks are relatively insensitive to outliers.
- PNN networks generate accurate predicted target probability scores.
- PNNs approach Bayes optimal classification.
- PNN are slower than multilayer perceptron networks at classifying new cases.
- PNN require more memory space to store the model.

Hodge [20] also notes that PNNs are faster than multilayer perceptron networks, and they can be more accurate than multilayer perceptron networks.

2.3.3 Support Vector Machines

Hastie, Tibshirani and Friedman [14] present Support Vector Machine (SVM), which produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space.

Linoff [32] defines SVM as a geometric method of separating two classes by finding the best hyperplane that puts one class above it and the other below.

Kernels, such as a radial basis function (RBF) kernel, can be used to learn complex regions [2].

SVM performs classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories. Sherrod [18] notes that the goal of SVM modeling is to find the optimal hyperplane that separates clusters of vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other size of the plane. The vectors near the hyperplane are the support vectors.

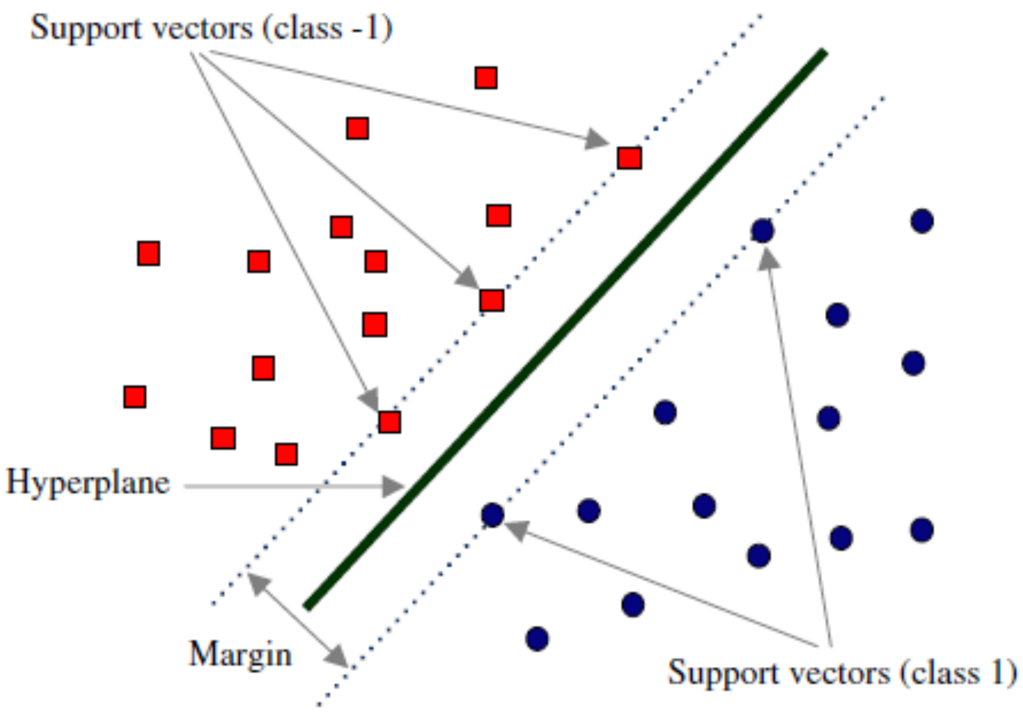


Figure 12: Support Vector Machines [39].
An SVM analysis finds the line (or, in general, hyperplane) that is oriented so that the margin between the support vectors is maximized.

Sherrod [18] proposes that SVM has shown great promise at generating accurate models for a variety of problems. SVM seems to be particularly good at pattern recognation, but it is also applicable to all other types of modeling applications.

Maimon [19] notes that learning here means the estimation of a function from a set of examples. To do this, a learning machine must choose one function from a given a set of functions, which minimizes a certain risk that the estimated function is different from the actual function (yet unknown).

Maimon [19] also suggests that to obtain a high level of performance, some parameters of the SVM algorithm have to be tuned: 1) the selection of the kernel function, 2) the kernel parameter(s), 3) the regularization parameters for the tradeoff between the model complexity and the model accuracy. The kernel here can be viewed as a nonlinear similarity measure. To simplify the tuning process we can use the cross-validation technique to identify the model of the best SVM classifier.

2.3.4 Naïve Bayes

Naïve Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features (attributes). We emphasize that there should be independence between attributes [3].

Maimon [19] explains that Naïve Bayes represents a model in a form of probabilistic summaries.

Bayes' Law is defined as follows [26]:

$$p(y|x) = \frac{p(x|y) \times p(y)}{p(x)}$$

$p(x|y)$ here means the probability that event x happens given y happens; $p(y|x)$ means the probability that event y happens given x happens; $p(x)$ is the probability of event x; $p(y)$ is the probability of event y.

Usually nonparametric methods suffer from the requirements that all of the samples must be stored, that means we have high requirements for memory, so, the performance is poor, appearing near the bottom of the performance for almost all the datasets.

Linoff [32] notes that Naïve Bayes models often perform quite well in terms of accuracy.

Han [3] also found that the method has exhibited high accuracy and speed when applied to large databases.

2.4 Others

2.4.1 Logistic regression

Maimon and Rokach [19] write that statisticians commonly define regression so that the goal is to understand "as far as possible with the available data how the conditional distribution of some response y varies across subpopulations determined by the possible values of the predictors".

According to Sherrod [18] and Linoff [32] logistic regression is a type of predictive model that can be used when the target variable is a categorical variable with two categories or binary problems - for example live/die, wins race/does not win, etc.

Flach [29] expresses that the basic idea is to combine a linear decision boundary with logistic calibration, but to train this in a discriminative fashion by optimizing conditional likelihood.

So, rather than modelling the classes as clouds of points and deriving a decision boundary from those clouds, logistic regression concentrates on areas of class overlap.

2.4.2 Boosting algorithms

Boosting is one of the most powerful learning ideas. The motivation for boosting is a procedure that combines the outputs of many weak classifiers to produce a powerful "committee" [14].

Boosting works by sequentially applying a classification algorithm to reweighted versions of the training data, and then taking a weighted majority vote of the sequence of classifiers thus produced. For many classification algorithms, this simple strategy results in dramatic improvements in performance [40].

Charu C. Aggarwal [11] explains the idea behind the boosting algorithm. Boosting methods are commonly used in classification in order to improve the classification perfomance on difficult instances of the data.

AdaBoost algorithm works by assiociating each training example with a weight, which is updated in each iteration, depending upon the results of the classification in the last iteration (if the instance was incorrectly classified, its weight is increased and if the instance was correctly classified, its weight is decreased). Specifically, instances that are misclassified, are given higher weights in successive iterations. The idea is to give higher weights to "difficult" instances which may lie on the decision boundaries of the classification process. The overall classification results are computed as a combination of the results from different rounds.

AdaBoost.M1 (adaptive boosting) is an ensemble technique to increase overall accuracy by learning and combining a series of individual classifier models [3].

AdaBoost.M1 is a method that is designed especially for classification [41].

3 Data

In this section, we describe topics related to data: the acquisition, the nature, and processing operations.

3.1 Acquisition process

The ARGO Project Office describes the data acquisition process as follows [42]:

“When a float surfaces, the data are transmitted and the float's position is determined either by Système Argos or by GPS. The data from floats using other communications systems may go directly to the float's owner or to the AIC before arriving at the National Data Centers (DAC). At the DACs, they are subjected to initial scrutiny using an agreed upon set of real-time quality control tests where erroneous data are flagged and/or corrected and the data are passed to ARGO's two Global Data Assembly Centers (GDACs) in France and the USA.”

The GDACs are the first stage at which the freely available data can be obtained via the Internet (via FTP-servers).

The target is for these "real-time" data to be available within approximately 24 hours of their transmission from the float.

Given the expanding size of the ARGO data system, there are other ways to access the ARGO data besides downloading the netCDF files from the GDACs. There are gridded fields and velocity products available as well as data viewers developed to look at the ARGO data (i.e. Marinexplore.org).

According to University Corporation for Atmospheric Research (UCAR) the netCDF is an open standard, the netCDF file is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. It is commonly used in climatology, meteorology and oceanography applications and GIS applications.

Marinexplore.org transferring process, for example, is based on FTP-protocol. Their back-end system periodically checks for updates on the FTP-server and if necessary it transfers the new data to their system. At this moment, there are no additional quality control procedures, the data is transferred AS IS.

This study uses real-world sensor data obtained from the fleet of more than 3 000 drifting floats from the ARGO program, which is a collaborative partnership of more than 30 nations from all continents to provide a seamless global array allowing any country to explore the ocean environment.

The data in this dataset is collected periodically from drifting floats and is transferred to the Marinexplore.org database. We have acquired our current datasets from the Marinexplore.org environment (we used Marinexplore.org's Datastudio functions), which are provided in .CSV-format.

3.2 Dataset description, characteristics

Our dataset covers a period of 3 years starting from the beginning of January 2011.

Each .CSV-file contains spatiotemporal data as well as an array of oceanographic data that include sea temperature, depth, pressure, salinity, electrical conductivity. The data is collected in the form of sequences.

As only a few sensors (~10%) measure electrical conductivity, we decided not to focus on this attribute and its values.

Our spatiotemporal sensor data covers the Atlantic Ocean (without the Mediterranean Sea), the area is shown in Figure 13.

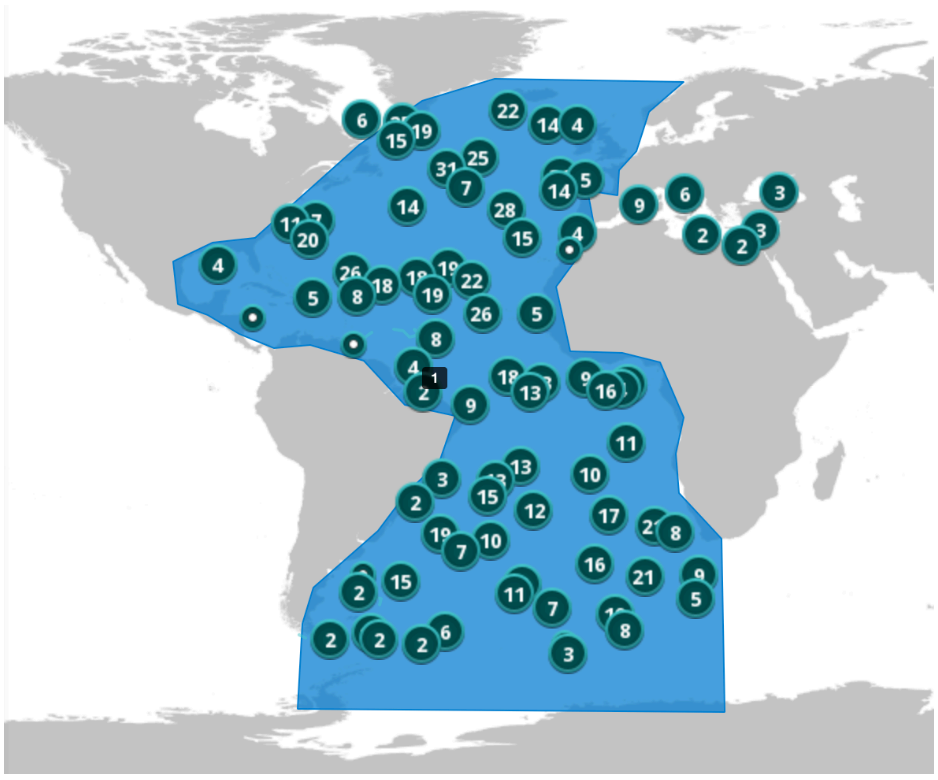


Figure 13: Dataset coverage.

We excluded the Meditarrean because, in terms of behavior, its measured parameters deviate from the parameters of the Atlantic Ocean (salinity, temperature, pressure correlations) [43]. Similar knowledge is also supported by different studies [44].

The Mediterranean Sea measured parameters, and its deviations are shown in Figure 14 (the red rectangle).

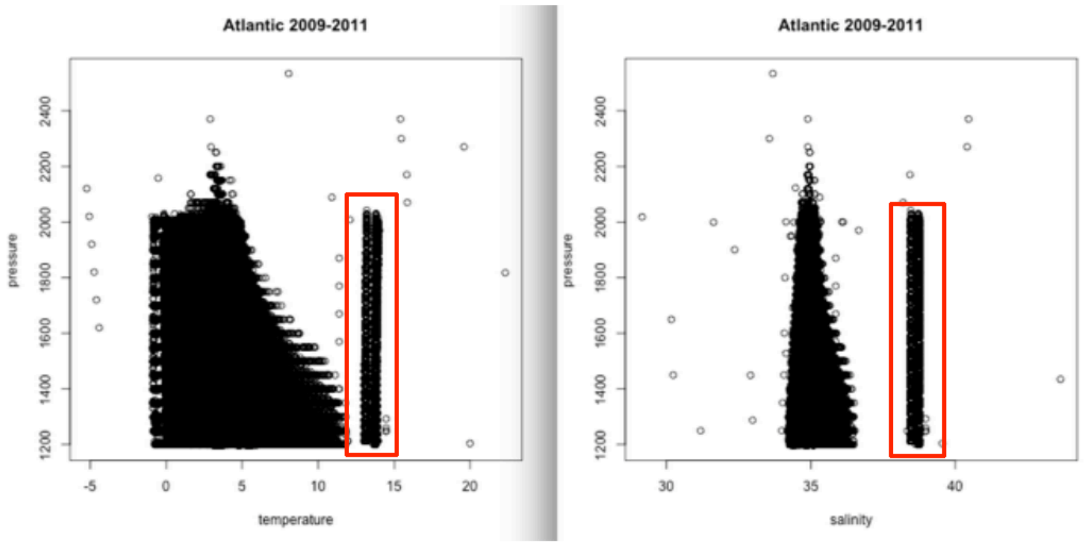


Figure 14: The Mediterranean parameters deviations [43].

A time series is a chronological sequence of observations on a particular variable. The sampling here is irregular.

In spatial data, many non-spatial attributes (i.e. temperature, salinity, etc.) are measured at spatial locations. Unusual local changes in such values are reported as outliers.

Our dataset characteristics:

File size: 306 740 024 bytes = 292.53 MB;
Attributes: 8;
Instances: 4 230 661;
Parameters: Water Pressure, Sea Water Salinity, Water Temperature.

As we use pressure/salinity/temperature values as inputs to our models, we consider an instance without either a pressure, salinity or temperature value an invalid instance. We excluded invalid instances from our dataset. Our initial dataset contains 4 230 661 instances, after removing invalid instances we have a dataset that contains 1 576 251 (37.26%) valid instances (2 654 410 were invalid, it was 62.74%).

To perform our tests, we randomly picked up square: 24U (size = 6° × 8°). The selected square is shown in Figure 18. See Chapter 3.3 which describes latitude and longitude conversion.

After performing initial quality checks, it contains 45 305 instances. We labelled the dataset so that there are 45 182 instances marked as TRUE and 123 instances marked as FALSE.

To analyze the nature of a typical dataset we selected 3 test datasets. Each dataset is produced by a float and it gives us an understanding of what the data looks like.

Our three datasets and their characteristics:

Dataset 1 (data1.csv):
OPeNDAP link: http://dap.marinexplore.org/dap/u/8dc601cd89ee/28b239d2-df99-11e4-8026-22000aaa19e6;
Float: Argo Float ID WMO-6900411 (APEX Profiling Float);
ID: ARGO.WMO-6900411;
Owner: NOAA National Oceanographic Data Center;

Parameters: Water Temperature, Sea Water Salinity, Water Pressure;

Area: Canary Island area, Atlantic Ocean;

Instances: 1208;

Attributes: 8.

Dataset 2 (data2.csv):

OPeNDAP link: http://dap.marinexplore.org/dap/u/8dc601cd89ee/4cd0c77e-df9a-11e4-8026-22000aaa19e6:

Float: Argo Float ID WMO-6900637 (PROVOR Profiling Float);

ID: ARGO.WMO-6900637;

Owner: NOAA National Oceanographic Data Center;

Parameters: Water Temperature, Sea Water Salinity, Water Pressure;

Area: North Atlantic;

Instances: 2457;

Attributes: 8.

Dataset 3 (data3.csv):

OPeNDAP link: http://dap.marinexplore.org/dap/u/8dc601cd89ee/2e378bba-e137-11e4-8026-22000aaa19e6;

Float: Argo Float ID WMO-5903393 (APEX Profiling Float);

ID: ARGO.WMO-5903393;

Owner: US ARGO PROJECT;

Parameters: Water Temperature, Sea Water Salinity, Water Pressure, Sea Water Conductivity;

Area: North Atlantic;

Instances: 212;

Attributes: 9.

CSV-file basic structure [45]:

- A line break (CR/LF) between each line;
- A comma between each value;
- Two commas in a row for a missing value;

- No comma after the last value, unless the last value is missing in which case the preceding comma is retained;

- Double quotes around values that contain commas, spaces, line breaks or double quotes (escaped as “”);

- The first line of every file comprises a list of column headers that provide names for the values in the following rows. If a value has a unit, it shall be specified in parentheses after the column name. This line is known as the header row. All other lines are referred to as data rows.

- Each line of text represents a single point in time and space.

We assume that downloaded dataset contains proper structure and is without syntax errors. No syntax validation is performed.

The dataset structure (float data) explained in the UML notation:

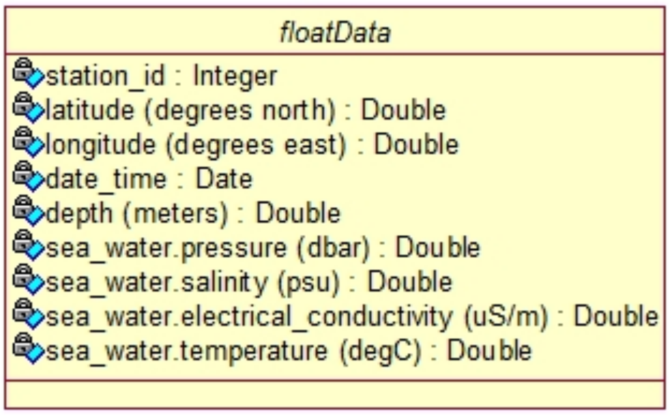
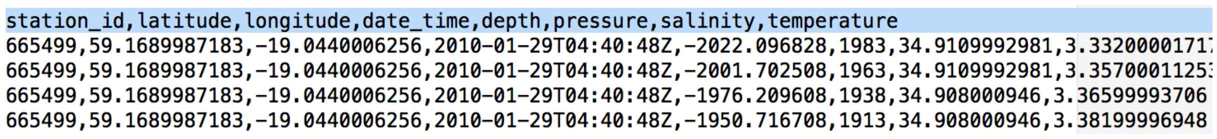

floatData
station_id : Integer
latitude (degrees north) : Double
longitude (degrees east) : Double
date_time : Date
depth (meters) : Double
sea_water.pressure (dbar) : Double
sea_water.salinity (psu) : Double
sea_water.electrical_conductivity (uS/m) : Double
sea_water.temperature (degC) : Double

Figure 15: The dataset (float data) structure explained in the UML notation.

The dataset example (float data) in .CSV-format, header values are renamed to simplify the handling of the file:



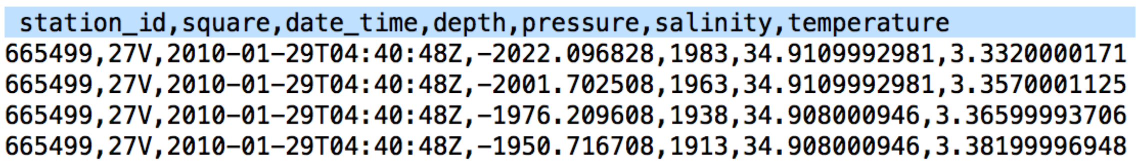
```
station_id,latitude,longitude,date_time,depth,pressure,salinity,temperature
665499,59.1689987183,-19.0440006256,2010-01-29T04:40:48Z,-2022.096828,1983,34.9109992981,3.3320001717
665499,59.1689987183,-19.0440006256,2010-01-29T04:40:48Z,-2001.702508,1963,34.9109992981,3.3570001125
665499,59.1689987183,-19.0440006256,2010-01-29T04:40:48Z,-1976.209608,1938,34.908000946,3.36599993706
665499,59.1689987183,-19.0440006256,2010-01-29T04:40:48Z,-1950.716708,1913,34.908000946,3.38199996948
```

Figure 16: Dataset example in .CSV-format (before processed).

Field characteristics: 1) Station_id is an integer. 2) Latitude and longitude in decimal degrees. 3) Date and time in ISO 8601 format with punctuation (normally

$yyyy - mm - ddThh{:}mm{:}ssZ$. 4) Depth is measured in meters (m), negative below the surface of the water. 5) The pressure is measured in decibars ($dbar$). 6) The unit for salinity is “psu" (Practical Salinity Units). 7) The unit for electrical conductivity is “Siemens per meter” ($\mu S/m$, it is displayed as “uS/m”). 8) Unit for temperature is degree Celsius ($deg\ C$).

After preproccesing the file looks like (see Chapter 3.3):



```
station_id,square,date_time,depth,pressure,salinity,temperature
665499,27V,2010-01-29T04:40:48Z,-2022.096828,1983,34.9109992981,3.3320000171
665499,27V,2010-01-29T04:40:48Z,-2001.702508,1963,34.9109992981,3.3570001125
665499,27V,2010-01-29T04:40:48Z,-1976.209608,1938,34.908000946,3.36599993706
665499,27V,2010-01-29T04:40:48Z,-1950.716708,1913,34.908000946,3.38199996948
```

Figure 17: Dataset example in .CSV-format (after processed).

3.3 Preprocessing

We perform 2 data preprocessing activities during the data transferring process. See Chapter 5.

3.3.1 Data labelling

As mentioned in Chapter 2, supervised methods require pre-labelled data, tagged as normal or abnormal.

The ARGO quality process takes 6-12 months; our data transferring process is responsible for transferring this corrected data from the ARGO FTP-server to our database. During the process we update our database, and we flag those instances as INVALID and we can use it as a training data. These processes ensure that our corrected data is 100% valid, and it is ideal for training processes.

3.3.2 Latitude and longitude conversion

Database systems support spatial data types (sometimes referred as geometry data type) - this allows us to operate on spatial data, but it is complicated to handle latitude and longitude values by machine learning methods. Thus, we have to convert latitude and longitude values into some square grid coordinate system format (i.e. Global Area Reference System (GARS), World Geographic Reference System (Georef), The Military Grid Reference System (MGRS)).

Similarly, tourist maps use some similar form of area referencing with numbers across the top and the bottom and letters along the sides.

It gives us a better way to handle the geospatial data; it is a better input format for supervised machine learning methods.

The Military Grid Reference System (MGRS) is the geocoordinate standard used by NATO for locating points on Earth; it has precision levels of up to 1 meter. It can represent any location on Earth using an alphanumeric string (i.e. "35VLF7210790906").

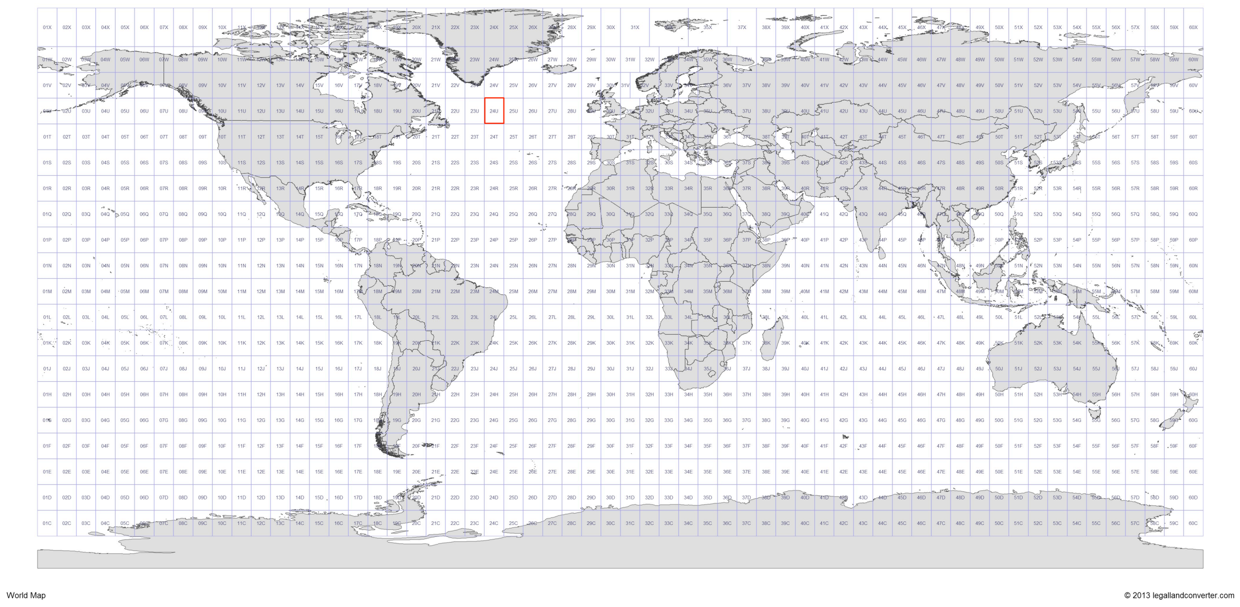


Figure 18: The MGRS. Square 24U is marked with a red square [46].

A MGRS conversion example: 59.437222°, 24.745° (the location of Tallinn) = "35VLF7210790906" (seven parameters), where "35" is a zone number, "V" is a band letter, "L" is a MGRS column letter, "F" is a MGRS row letter, "72107" is a MGRS easting (represents the number of meters east of the origin - southwest corner - of the 100 kilometer square in which it is contained.) and "90906" is a MGRS northing (represents the number of meters north of the origin - southwest corner - of the 100 kilometer square in which it is contained.). Precision levels: 35V - precision level here is 6°×8°, 35VLF - the precision level here is 100×100 km, 35VLF79 - the precision level here is 10×10 km, 35VLF7290 - the precision level here is 1×1 km, 35VLF721909 - the precision level here is 100×100 m, 35VLF72109090 - the precision level here is 10×10 m, 35VLF7210790906 - the precision level here is 1×1 m.

The MGRS covers the entire world and it can refer to a square with a side length of 100 km, 10 km, 1 km, 100 m, 10 m or 1 m. Due to its flexibility we can easily convert latitude and longitude to the MGRS format during the data transferring process. The MGRS system gives us good data management capabilities, see Chapter 5.

3.4 Initial quality check

A variety of tests are performed to evaluate data quality. Some test scenarios are based on the theoretical aspects presented in Chapter 2.1, some are based on the recommendations by ARGO [47] [48] [49].

Based on our findings, TEST7 says that there is a stable waterfront lower than 1 500 meters below the sea level, we use those findings as a reference to evaluate the state of sensor. See Chapter 4.4.

The following guidelines have been developed to perform the initial quality check:

- TEST1: When data is received, it is first checked for gaps and missing data. Criteria: consecutive N is missing data, N is user defined. The checks are based on time tags, which are included in the data stream. Our strategy for handling missing data values handling: Sherrod [18] suggests to exclude a data row, if there are many data rows available and the percentage of rows with missing values is small. It is fast, and it prevents any error from being introduced due to the missing values.

  As many algorithms are unable to handle missing values, it is required to perform this missing data test before our validation process.

- TEST2: Duplicates test.

- TEST3: Flat line test. When some sensors fail, the result can be a continuously repeated observation of the same value. According to Chapter 1.2.2 it is common that a frozen sensor causes the same temperature or salinity profiles. This test compares the present observation n to a number of previous observations. Observation n is flagged if it has the same value as the previous observation within a tolerance value.

- TEST4: A spike test. This test is a single value test.

  Large spikes are easier to identify as outliers and to flag as failures, and smaller spikes may be real and only flagged as they are suspicious. The thresholds can be set dynamically (i.e. a multiple of a standard deviation over a specified period).

  The formula:

$$\frac{\left|V_2 - (V_3 + V_{1)}\right|}{2} - \frac{|V_{1-}V_3|}{2} > THRESHOLD$$

  $THRESHOLD$ is user defined.

  NOAA [47] recommends to flag the temperature value as wrong when the test value exceeds 6.0°C for the pressures less than 500 dbar, if the test value exceeds 2.0°C for pressures greater than or equal to 500 dbar, and the salinity value is flagged as wrong when the test value exceeds 0.9 PSU for pressures lower than 500 dbar, or the test value exceeds 0.3 PSU for pressures greater than or equal to 500 dbar.

- TEST5: Range test. Max/min is user defined. See Chapter 2.1. Parameter envelopes are given in Appendix 3.

- TEST6: Mean shift. The test breaks the time series into N segments of M points each, the segment means are compared to neighboring segments. If the difference in the means of two consecutive segments exceeds $THRESHOLD$, the data is flagged. $THRESHOLD$ is user defined.

- TEST7: Sensor "calibration" test. Calibration is the act of checking or adjusting (by comparison with a standard) the accuracy of a measuring instrument. Thus, it is a method of evolution for the performance of the sensor. We know that deepwater parameters are quite stable (see Chapter 4.3.2), there are no significant daily or seasonal fluctuations, the parameters, in particular geographical location in deepwater are constant. We can use this data as a reference data. It helps us to evaluate the state of the sensor.

Outliers are differences between sensors' expected output and its measured output, which show up consistently every time a new measurement is taken. If the measured value falls outside of the validity range of the model, we have detected an outlier.

4 Empirical evaluation

In this chapter, we investigate the behavior of the potential outlier detection methods discussed in Chapter 2.

The goal of this evaluation is to choose the best method in terms of performance.

Each method will be run on a number of datasets; datasets vary in size, number of instances and the nature of outliers.

Our strategy is simple: It is better to predict a negative class as an outlier (false positive), rather than miss a true outlier (false negative). Similarly, the cost of lending to a “bad” customer is much higher than the sunk opportunity cost of refusing a loan to a “good” customer.

Provost [31] notes that in order to evaluate a classifier properly, it is important to understand the notion of class confusion and the confusion matrix. It is a problem involving n classes where an n class is an $n \times n$ matrix with the columns labelled with actual classes and the rows labelled with predicted classes. The confusion matrix is displayed in Figure 19.

		Predicted class	
		yes	**no**
Actual class	**yes**	True positive	False negative
	no	False positive	True negative

Figure 19: The Confusion Matrix displays actual versus predicted values.

4.1 Pre-selection methods

Hodge [20] notes that in outlier detection, we should select an algorithm that is suitable for the dataset in terms of the correct distribution model, the correct attribute types, the scalability, the speed, any incremental capabilities to allow new exemplars to be stored and the modelling accuracy. We should also consider which of the three fundamental approaches is suitable for our problem - a clustering approach, a classification approach, or a novelty approach (newness). This will depend on: the data type, whether the data is pre-labelled, the amount of data, the ground truth of the labelling (i.e., whether any novel records will be mislabelled as normal), how they wish to detect outliers and how they wish to handle them.

We selected five most promising methods from approaches we covered in Chapter 2, namely, Naïve Bayes, boosting algorithm (AdaBoost.M1), SVM[2], kNN[3], and PNN. These methods are suitable for the dataset in terms of the correct distribution model, the correct attribute types, the scalability, the speed, and the modelling accuracy.

We shortly explain our decisions:

- Several authors have said that Naïve Bayes models often perform quite well in terms of accuracy and speed.
- k-Nearest Neighbor performed well, and it is known to suffer the kind of exponential computational growth. To get more information about its performance, we test its scalability on a larger dataset.
- We know that SVM has been promising in terms of generating accurate models for a variety of problems; we decided to test the method on our datasets.
- Multilayer neural networks can cope with unseen patterns, and their ability to accurately predict outcomes of complex problems is their greatest strength. They have shown long training periods, but it is said that PNNs are faster than multilayer perceptron networks, and they can be more accurate than multilayer perceptron networks.
- Statistical techniques rely on the assumption that the data is generated from a particular distribution. After studying the nature of our typical datasets, we cannot adopt these simple and powerful methods. Thus, we cannot use statistical methods in our validation process. Nevertheless, we figured out that there are several promising methods we can use in our initial validation check procedures. See Chapter 3.4.
- We tested hierarchrical-clustering methods (the *HierarchicalClusterer* implementation in Weka), it generally performed well on small datasets, but the method is not scalable to larger datasets.

[2] We use the LibSVM wrapper implemented in Weka by Yasser El-Manzalawy.
[3] We use the lBk wrapper implemented in Weka by Stuart Inglis, Len Trigg, Eibe Frank.

- We know that k-Means is a relatively fast method, but it is also among the least accurate methods. As our main criteria to choose the appropriate outlier method is based on accuracy, the selected methods should be as accurate as possible, so we cannot rely on k-Means clustering method.
- We performed several accuracy tests on our test datasets, but unfortunately LOF, BIRCH[4] and DBSCAN did not perform well in our tests.
- The scalability of NN didn't satisfy us. Its accuracy is acceptable, but the method for generating and evaluating models for larger datasets is very time-consuming.

4.2 Testing methodology

We express here what heuristic is used to decide if the datapoint is anomalous or not and criteria to choose the best outlier method.

Our objective is to determine whether one method is better or worse than another, across training and test datasets.

Questions: 1) What to measure? 2) How to measure? 3) How to interpret it?

Han [3], Maimon [19], Provost [31], and Linoff [32] note that naturally classification models with high accuracy are considered better. Accuracy is the proportion of true results in the population.

$$accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives}$$

See the calculation equations below.

Similarly, we adopted the idea that this is relevant for our domain, and therefore, our main criteria to choose the appropriate outlier method based on accuracy. It is the most important measure in the current domain, it means that methods should be as accurate as possible. Accuracy is the percentage of examples that the algorithm classifies correctly. Accuracy has a range [0;1].

[4] We use the BIRCHCluster wrapper implemented in Weka by Gabi Schmidberger and Frac Pete.

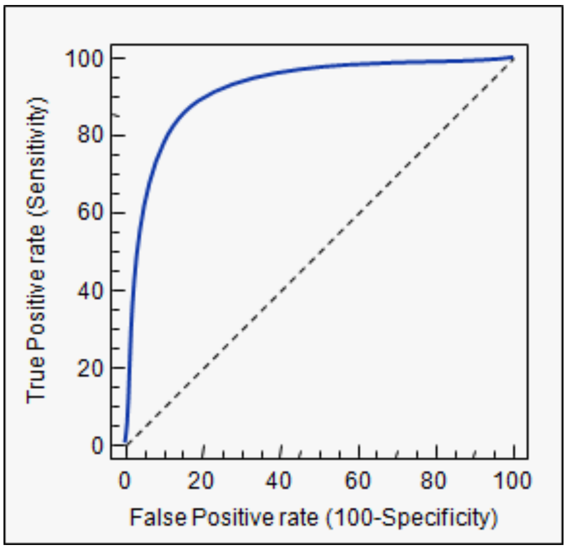


Figure 20: ROC curve [66].

We note that sometimes accuracy is not the best and the most appropriate choice, but it depends on the current domain and the nature of the dataset. Caruna [17] writes that Precision/Recall measures are used in information retrieval; Lift is appropriate for some marketing tasks, etc.

To determine the accuracy of the method, true positive (TP), true negative (TN) rates, as well as false positive (FP) and false negative (FN) rates are measured.

- A true positive rate or sensitivity is defined by: $TPR = \frac{TP}{TP+FN}$;
- A true negative rate or specificity is defined by: $TNR = \frac{TN}{TN+FP}$;
- A false positive rate is defined by: $FPR = \frac{FP}{FP+TN}$;
- A false negative rate is defined by: $FNR = \frac{FN}{FN+TP}$.

Han [3] covers the ROC chart, which is a visual tool for comparing two classification models; the display gives the measure of the predictive accuracy of a model. It displays the sensitivity (true positive TP) and the specificity (false positive FP) of a classifier for a range of cutoffs.

ROC graphs are two-dimension graphs in which TP rate is plotted on the Y axis, and FP rate is plotted on the X axis. A discrete classifier is one that outputs only a class label. Each discrete classifier produces a pair (FP rate, TP rate) corresponding to a single point in ROC space [50].

The lower left point (0,0) represents the strategy of never issuing false positive errors but also gaining no true positives. The opposite strategy, of unconditionally issuing positive classifications, is represented by the upper right point (1,1). The point (0,1) represents the perfect classification. Informally, one point in the ROC space is better than another if it is to the northwest (the TP rate is higher, the FP rate is lower, or both) of the first. Thus, a discrete classifier produces only a single point in the ROC space [50].

Flach [29] proposes that the area under the ROC (AUC) curve is the ranking accuracy.

Lobo [51] expresses that the AUC is currently considered to be the standard method to assess the accuracy of predictive distribution models. Vuk [52] also mentions that the ROC curve performance analysis is the most appropriate and widely used method for a typical machine learning application setting where choosing a classifier with best classification accuracy for a selected problem domain is of crucial importance.

We compare algorithms over dataset. Flach [29] recommends using the Friedman test to rank algorithms. The Friedman test is designed for this situation. In our context, it detects differences in treatment over multiple datasets. Boslaugh [23] notes that the Friedman test is an extension of the matched pairs signed rank test for more than two related samples.

The idea is to rank the performance of all algorithms per dataset, from the best performance to the worst.

The Friedman test first ranks the values in each matched set (each row) from low to high. Each row is ranked separately. It then sums the ranks in each group (column). If the sums are very different, the P value will be small [53].

4.3 Data analysis

First, let us visualize distributions from our test files (actual files from Marinexplore.org, produced by one float; it is possible that uncalibrated devices can be in our selection).

It gives us an opportunity to identify anomalies within a given dataset, it also gives us an understanding of the actual data. Based on this data analysis we design appropriate datasets to evaluate methods and algorithms.

On the x-axis we display depths, on the y-axis we display either temperature or salinity values, date_time attribute values are colour-coded.

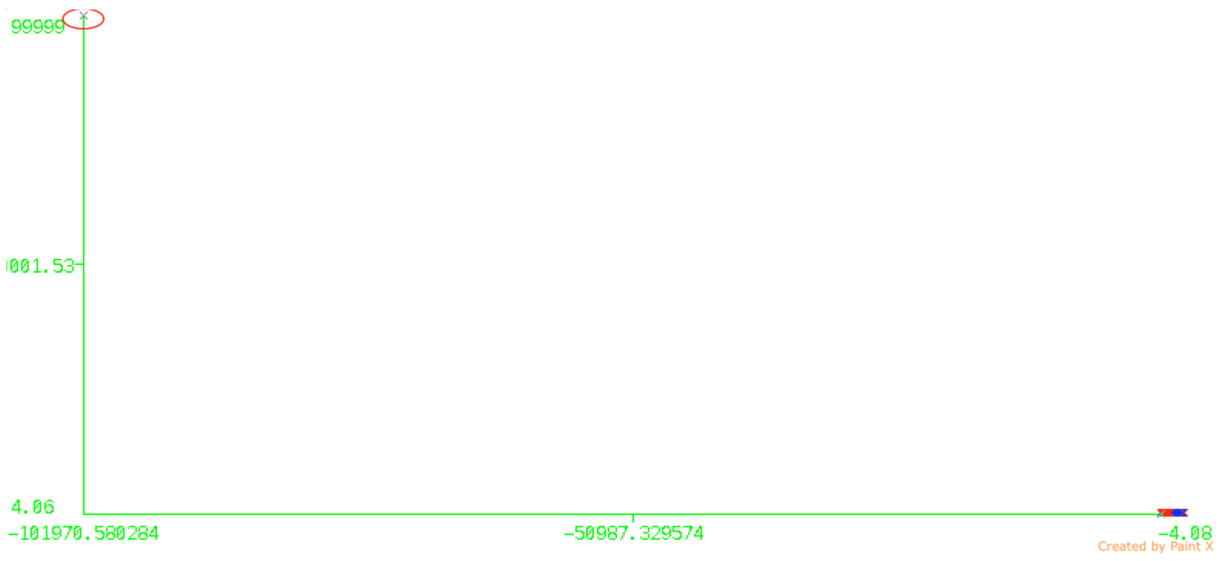


Figure 21: The distribution of dataset 1 (with outliers): depth and temperature.

In this .CSV-file, from row 1 106 to row 1 156 (one full cycle, date_time="2010-12-10T19:33:36Z"), measured values look problematic, a value "99999" is assigned to pressure, salinity, temperature attributes and depth has a value of "-101970.580284".

This may refer to a communication error or a float malfunction, etc. We should treat these values as outliers.

At the end of this chapter, we show known float examples, we see that "potential problem with bin assignment" produces similar output.

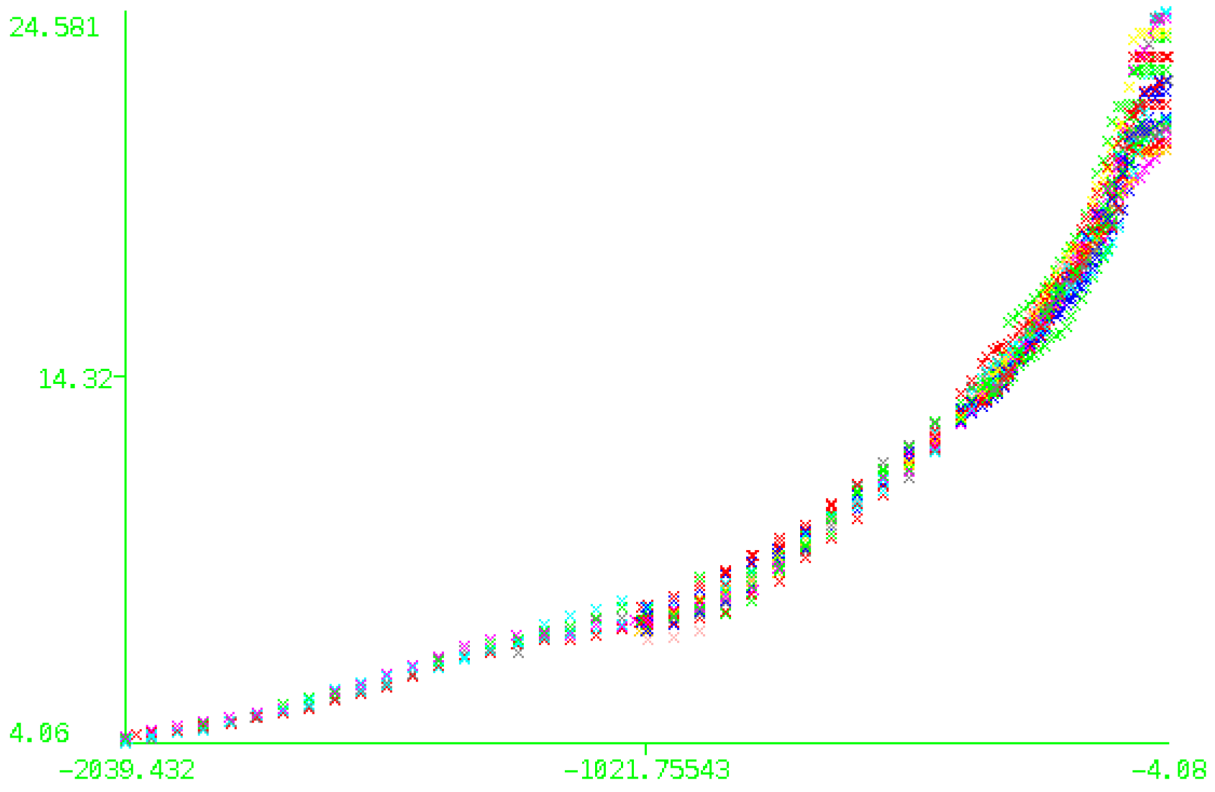


Figure 22: The distribution without outliers of dataset 1: depth and temperature.

After removing rows 1 106 - 1 156 (51 rows, one full cycle), the distribution looks rather normal.

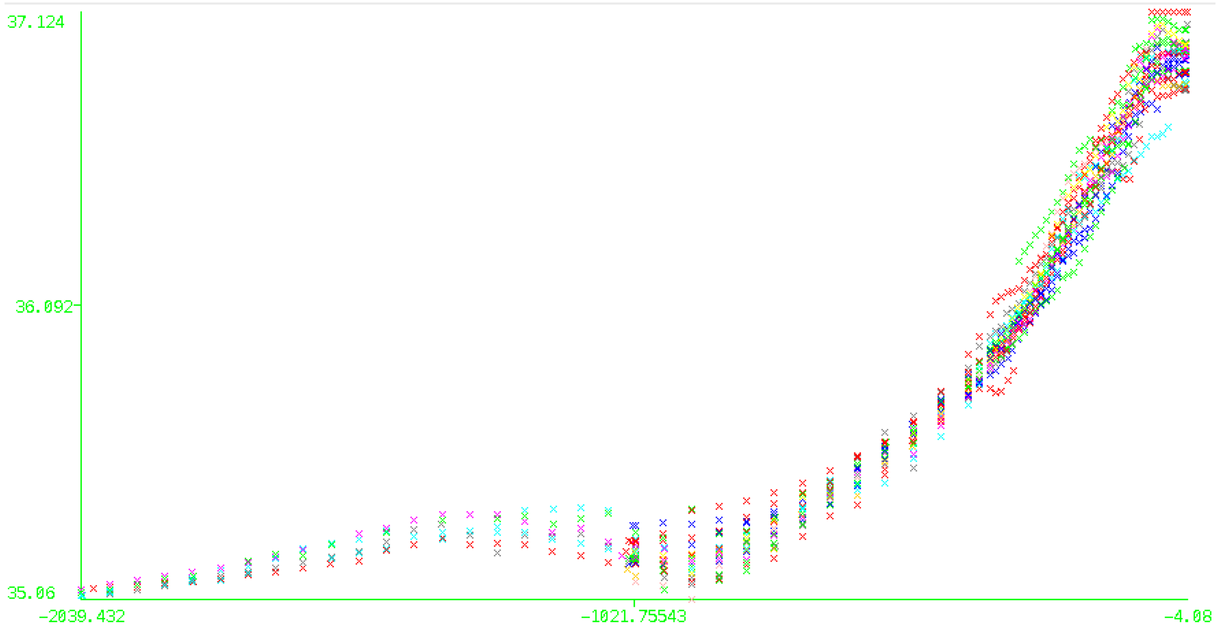


Figure 23: The distribution without outliers of dataset 1: depth and salinity.

The distribution without outliers of dataset 1: depth and salinity.

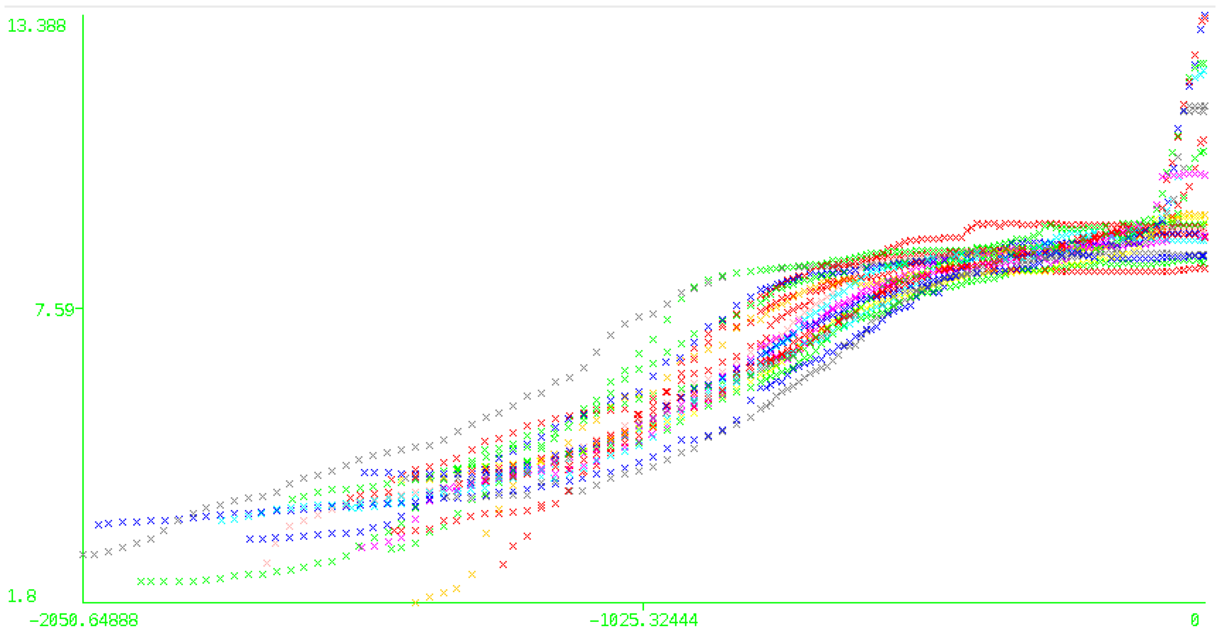


Figure 24: The distribution of dataset 2: depth and temperature.

The sun is probably heating up the sea surface; therefore it is not an anomalous distribution, there is no deviation from the "normal" model.

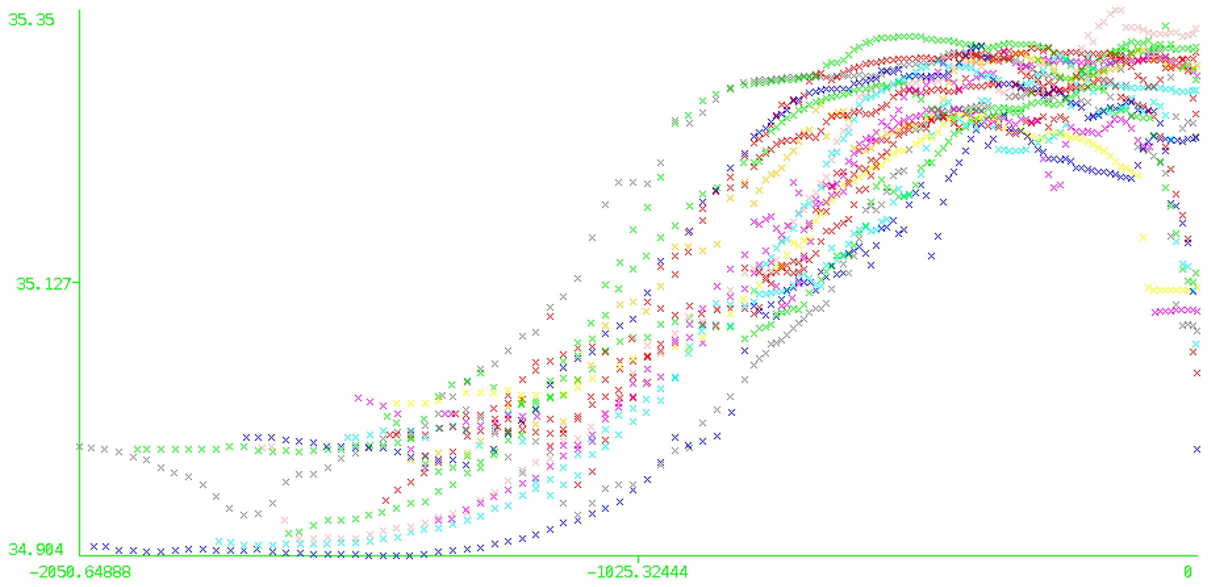


Figure 25: The distribution of dataset 2: depth and salinity.

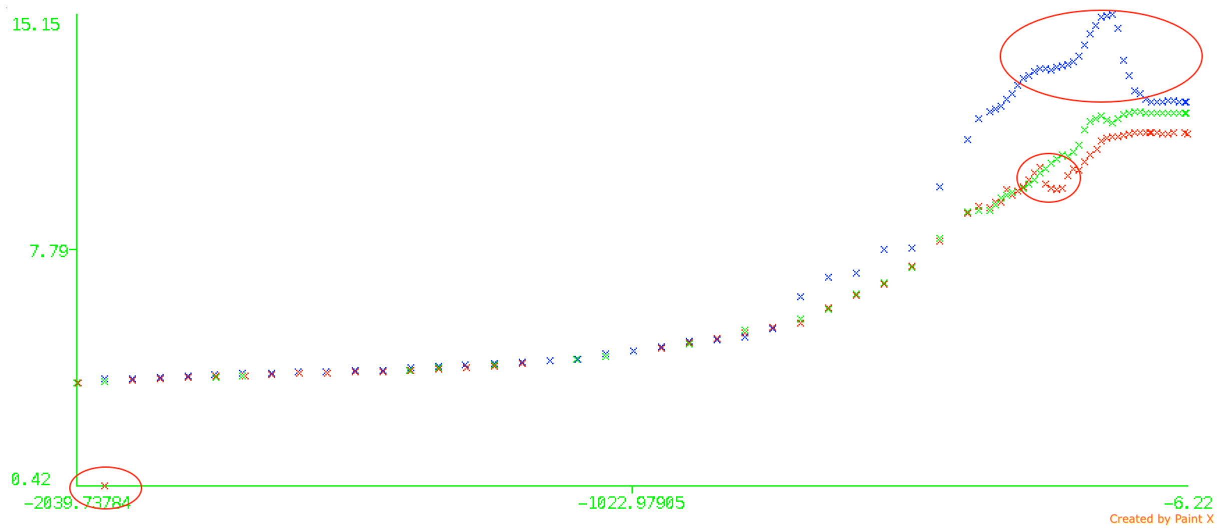


Figure 26: The distribution of dataset 3: depth and temperature.

Here we visually identify 3 potentially anomalous clusters (the first cycle is displayed in blue, the second cycle is displayed in red):

- Firstly, at the beginning of the second cycle, there is a single value that for some reason deviates from the normal behaviour. We should treat this single value as an outlier.

- Then, there is a potential anomaly near the end of the first cycle; the temperature suddenly falls between depths of 220 - 280 m.

- Finally, there is also a strange deviation near the end of the second cycle, where the temperature values are higher than expected between the depths of 100 to 200 m. We should treat these values as outliers.

The ocean, especially in the upper layers, is quite stratified in temperature and salinity, and measured values may be deviate.

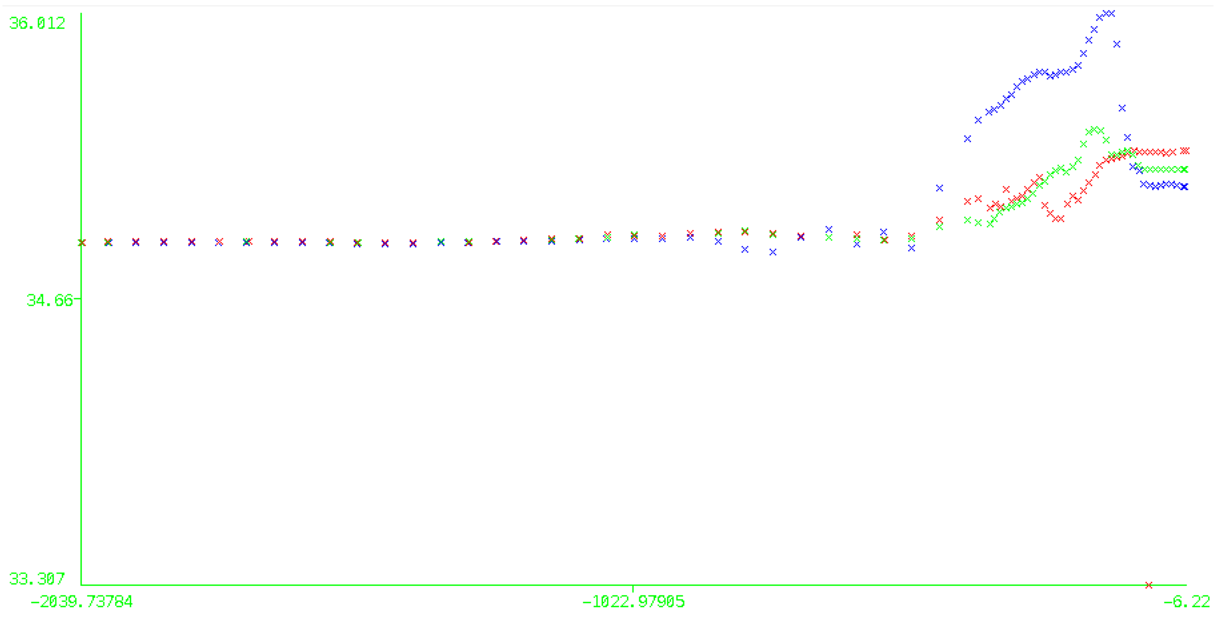


Figure 27: The distribution of dataset 3: depth and salinity.

Let us present potential outputs from the malfunctioned floats; it is based on actual sensor data collected from the ARGO floats (see TOP 25 failures in Chapter 1.2.2):

- Float (ID=5904001) has a problem ("salinity sensor problem"), i.e on 05.04.2015, it is assigned the same salinity value for 27 datapoints in a row (salinity = "36.6510009766).

- Float (ID=1900490) has a problem ("sensor problem"), the result repeatedly observe the same value (during one cycle), latitude="31.0979995728", longitude="-37.21900177", date_time="2006-03-05T11:55:40.000008Z", depth="-101970.580284", pressure="99999", salinity="99999", conductivity="9999900", temperature="99999".

- Float (ID=1900635) has a problem ("potential problem with bin assignment"), all parameters have constant values (during one cycle), latitude="33.2379989624", longitude="-36.6689987183", date_time="2010-12-23T12:23:02.000004Z", depth="-101970.580284", pressure="99999", salinity="99999", conductivity="9999900", temperature="99999".

- Float (ID=3900548) error message says: "on shore", there are no measurements.

According to Suga and Roemmich [54] there is a little drift or a strong negative drift when afflicted by a micro-leak.

We conclude that these TOP 25 errors are producing either frozen datapoints or datapoints that are significantly different from the remaining data.

4.4 Deepwater data analysis

Our aim here is to figure out the waterfront that we can use as a reference. To detect whether a float produces anomalious data, we can check it against the reference data. In the simplest approach, if the instance value falls inside the accepted tolerance threshold, the float is regarded as normal; otherwise it is treated as problematic.

The deviation shows how widely spread the measurements are. By knowing how large the spread is, we can evaluate the quality of the measurement.

The quality of the measurements helps us to evaluate the state of the sensors.

According to the definition [12] the standard deviation of a set of numbers tells us about how different the individual measurements typically are from the average of the set.

According to Figure 28, we see that the standard deviation (both the temperature and the salinity) is rather static lower than 1 500 meters below the sea level.

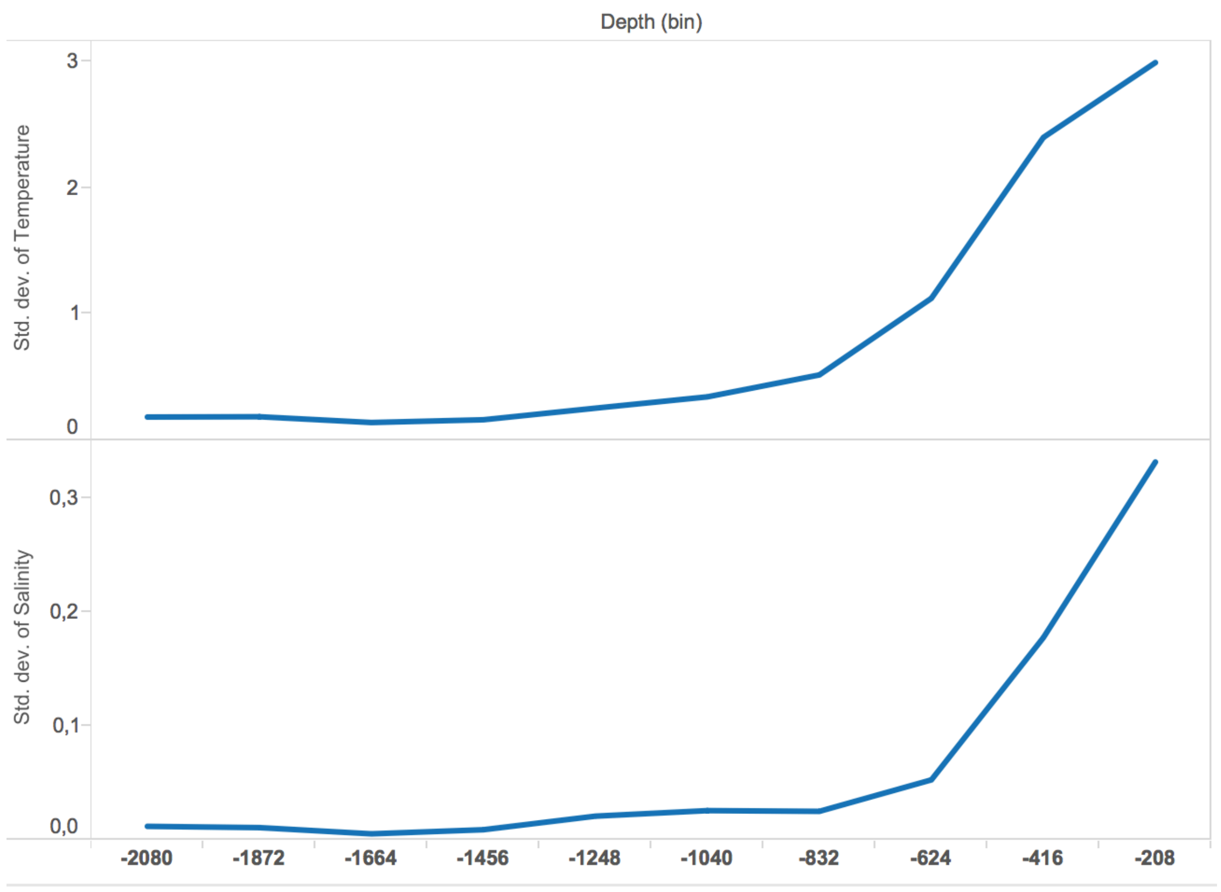


Figure 28: Temperature and salinity standard deviation dependence on depth.

As our aim is to figure out how stable the waterfront is, we only focus on the parameters measured lower than 1500 meters below the sea level next.

The dataset satisfies the following condition: "$DEPTH < -1500$ and $OUTPUT = TRUE$". The size of our reference dataset is 3 794 instances from square 24U. It contains data from 39 floats.

We calculated the standard deviation dependent on depth, our results are shown in Figure 29.

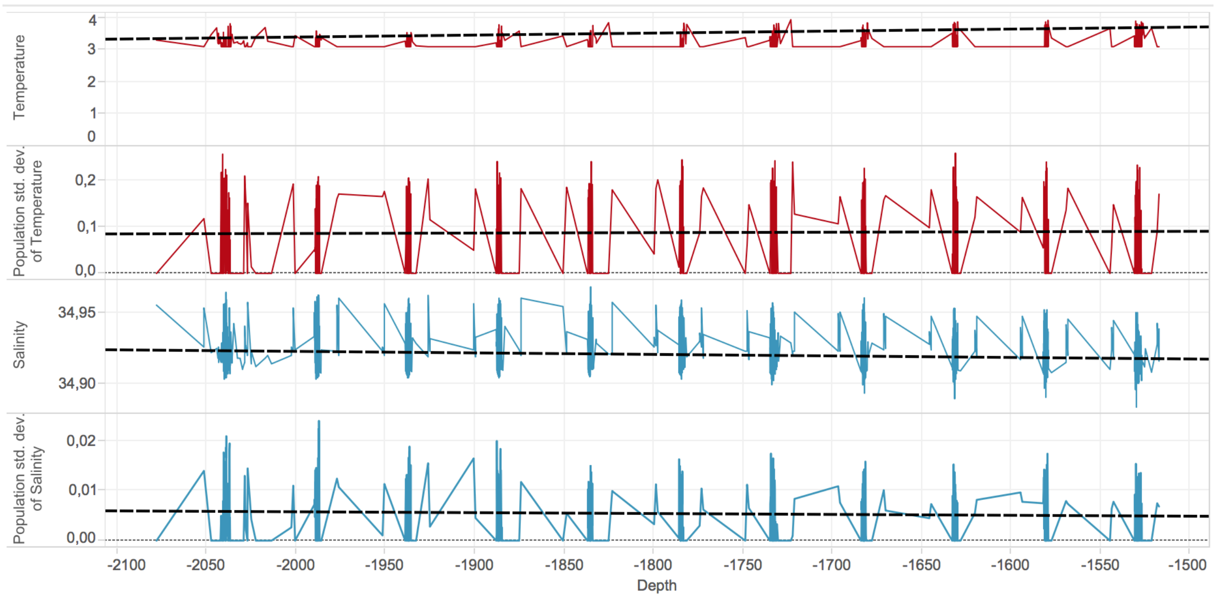


Figure 29: Temperature and salinity values and their standard deviations.

Linear regression lines are computed for temperature and salinity as an attribute derived from depth. In Figure 29 they are shown in dashed black lines. The slope of regression (flat line) indicates a stable trend lower than 1 500 meters below the sea level.

In Figure 29 we see that measured values are relatively stable, so we can use this technique for “calibration”, the thresholds can be set dynamically (i.e. a multiple of a standard deviation over a specified period).

We repeated similar experiments several times (in different part of the Atlantic Ocean), and the results were similar - we didn’t find any significant deviations.

According to this finding, we designed the test for our initial quality check procedure - TEST7 is based on this presented technique (see Chapter 3.4).

4.5 Dataset design

In this section, we explain the test datasets’ generation process.

As the amount of labelled data for training and testing is limited, and we need to use all of it as input to our learning algorithms, we have to design our test datasets by ourselves.

Later, we will use cross-validation methods; these methods are supported by many machine learning software packages. We reserve a certain amount for testing and use the remaining data for training [41].

We split our dataset into ten equal partitions; each turn is used for testing and the remainder is used for training. That is, we use $9/10$ of the data for training and $1/10$ for testing, and we repeat the procedure ten times so that in the end, all instances have been used exactly one time for testing. The ten error estimates are averaged to get an overall error estimate. This is called the 10-fold cross-validation [41].

Sherrod [18] notes that cross-validation is the recommended method for small to medium size data sets where computation time is not significant.

Chen [27] suggests to use the bootstrapping method if the dataset size is relatively small, and it works in a "leave-one-out" fashion.

We know that cross-validation tests are not ideal, theoretically about 2/3 of the results should be within a single standard deviation from the average, and 95% of the results should be within $2x$ standard deviation, so in a 10-fold cross-validation we rarely see results that are better or worse than $2x$ standard deviation. Running cross-validation several times may also result different answers [55].

It is important to note, that each class in the full dataset should be represented in about the right proportion in the training and testing sets. We have to ensure that random sampling is done in a way that guarantees that each class is properly represented in both training and testing sets. According to Witten [41] this procedure is called stratification.

Abu-Mustafa [56] states that if the data is sampled in a biased way, learning will produce a similarly biased outcome.

Flach [29] refers to the "No free lunch theorem": no learning algorithm can outperform another when evaluated over all possible classification problems, and thus the performance of any learning algorithm, over the set of all possible learning problems, is no better than random guessing. So, there is no universally best learning algorithm.

For this reason, the choice of evaluation data determines the outcome of the evaluation, so the set of evaluation datapoints is used, it should contain as representative a selection of relevant data as possible.

As we know the distribution of errors that are potentially producing anomalous data, and we know the nature of this is caused by anomalous data (based on our data analysis), we can efficiently design appropriate testsets. We design appropriate test datasets so that in each outlier class in the dataset is represented.

To ensure that the test data covers typical situations (both valid and invalid), we prepare datasets bearing in mind the following aspects:

- Each class is properly represented in both the training and the testing sets.
- Sampling is performed in a correct way. Right proportion of valid and invalid data.
- Invalid data. Deviations (from very small to very large).
- Different amounts of learning data.
- Calibration error aspects.

Our 5 designed datasets:

- Dataset 1: 51 outliers (very large deviations, coded failure idendificators - an example: “99999”), typical reason: a sensor error, calibration issues.
- Dataset 2: Real sensor data (examples: “depths above the sea level”, impossible seawater salinity values, high seawater temperature values), randomly selected 2 000 instances, manually added outliers.
- Dataset 3: This is valid data with 7 added outliers (minor seawater salinity and temperature deviations), 212 instances.
- Dataset 4: Real sensor data (minor seawater salinity and temperature deviations), randomly selected 2 000 instances, manually added outliers.
- Dataset 5: Real sensor data (“depths above the sea level”, negative seawater salinity values, high seawater temperature values, impossible seawater salinity values, impossible seawater temperature values, minor seawater temperature and salinity deviations), 45 305 instances, 45 182 valid instances and 123 invalid instances.
- Dataset 6: Real sensor data, randomly selected 2 000 instances, manually added outliers (coded failure idendificators, like ‘0’, ‘99999’).

Table 1 shows the characteristics of prepeared datasets, and Appendix 2 contains histograms for the distribution of the value of attributes.

Table 1: Chacarcteristics of designed datasets.

	Data set 1	Data set 2	Data set 3	Data set 4	Dataset 5	Dataset 6
Number of instances	1 208	2 000	212	2 000	45 305	2 000
Number of training instances	1 087.20	1 800	190.80	1 800	40 77.50	1 800
Number of testing instances	120.80	200.00	21.20	200.00	4 530.50	200
Number of attributes (input+output)	4+1	4+1	4+1	4+1	4+1	4+1
Number of valid instances	1 157	1 935	205	1 993	45 182	1950
Known outliers	51	65	7	7	123	50

As our version of DTREG limits the dataset with only 2 000 instances, in DTREG we cannot evaluate the performance on dataset 5. Our knowledge about scalability and time consumption is based on only the information published in the literature.

To test scalability, we prepared a dataset with 45 305 instances; we can test it in Weka, only on Naïve Bayes, AdaBoost.M1, kNN[5] and SVN. As our version of Weka doesn't have PNN implementation, we cannot evaluate its performance. In literature, it has been noted that is a well scalable method.

This dataset has a binary target variable ("output" is labelled either TRUE or FALSE); there are four predictor variables (predictor variable or independent variable - a variable that can be used to predict the value of another variable.), namely date_time, depth, temperature and salinity (all characterised as continuous variables).

The default value for output variable of TRUE (valid data point), FALSE (invalid data point) is assigned only if ARGO experts have flagged the value as an outlier.

[5] The number of neighbours k for kNN, k is a user-defined constant, in our experiments $k = 5$.

4.6 Tests

The experiments were performed in Weka and DTREG running on MacBook Pro. Short description about used software packages is given in Appendix 4.

Our tests were performed on a computer with the following parameters:

- Number of processors: 1;
- Processor: 2.4 GHz Intel Core i5 with 2 processing cores, 64 bit;
- Memory: 8 GB 1600 MHz DDR3;
- Operating system: OS X Yosemite 10.10.2.

In table 5 we show the measured time parameters, in our context this information is not primary to evaluate the method.

To evaluate 5 pre-selected methods, we use 6 different datasets. Table 1 summarizes our datasets we designed for evaluating the performance of our methods.

For all algorithms, we experimented with different parameter settings, Weka and DTREG provide an automatic functionality to adjust parameters to simplify this labour intensive task.

We set up different parameter settings (ranges, search steps, model optimization parameters, error tolerance settings, gradients) before performing these tests (i.e the range of hidden layers for neural networks, the range of neurons on each layer, etc.).

The detailed method configuration is shown in Appendix 5.

To illustrate this adjustment process and its labour intensiveness, the training PNN algorithm for dataset 4 in DTREG automatically performed 178 000 conjugations based on our parameter settings. There were more than 1 633 000 evaluations to build an effective network. Thus, these models require careful parameter selection.

Table 2 shows the accuracy, correctly and incorrectly classified values.

Table 2: Measured accuracy, correctly and incorrectly classified instances.

	Data set 1	Data set 2	Data set 3	Data set 4	Dataset 5	Dataset 6
Naïve Bayes	**100,00% (1 208- 0)**	**99,65% (1 993 - 7)**	97,64% (207 - 5)	**99,90% (1 998 - 2)**	99,82% (45 223 - 82)	98,85% (1 977 - 23)
AdaBoost.M1	**100,00% (1 208- 0)**	99,45% (1 989 - 11)	96,23% (204 - 8)	99,65% (1 993 - 7)	**99,98% (45 295 - 10)**	98,90% (1 978 - 22)
kNN	**100,00% (1 208- 0)**	**99,65% (1 993 - 7)**	**98,11% (208 - 4)**	**99,90% (1 998 - 2)**	99,94% (45 276 - 29)	99,65% (1 993 - 7)
PNN	**100,00% (1 208- 0)**	99,45% (1 989 - 11)	96,70% (205 - 7)	99,80% (1 996 - 4)	Nan	**99,80% (1 996 - 4)**
SVM	**100,00% (1 208 - 0)**	98,50% (1970 - 30)	96,70% (205 - 7)	99,65% (1 993 - 7)	99,93% (45273 - 32)	99,00% (1 980 - 20)

Table 3 shows the measured TP, TN, FP, and FN rates.

Table 3: Measured TP, TN, FP, and FN rates.

	Data set 1	Data set 2	Data set 3	Data set 4	Dataset 5	Dataset 6
Naïve Bayes	1 157 - 0 - 0 - 51	1932 - 4 - 3 - 61	204 - 4 - 1 - 3	1 993 - 2 - 0 - 5	45 104 - 4 - 78 - 119	1 950 - 23 - 0 - 27
AdaBoost.M1	1 157 - 0 - 0 - 51	1 935 - 11 - 0 - 54	204 - 7 - 1 - 0	1 992 - 6 - 1 - 1	45 182 - 10 - 0 - 113	1 950 - 22 - 0 - 28
kNN	1 157 - 0 - 0 - 51	1 935 - 7 - 0 - 58	205 - 4 - 0 - 3	1 993 - 2 - 0 - 5	45 182 - 29 - 0 - 94	1 950 - 7 - 0 - 43
PNN	1 157 - 0 - 0 - 51	1 935 - 4 - 0 - 61	205 - 2 - 5 -0	1 996 - 4 - 0 - 0	NaN	1 996 - 4 - 0 - 0
SVM	1 157 - 0 - 0 - 51	1 935 - 30 - 0 - 35	205 - 7 - 0 - 0	1 993 - 7 - 0 - 0	45 182 - 32 - 0 - 91	1 980 - 20 - 0 - 0

Table 4 shows the measured areas under the ROC curves.

Table 4: Measured areas under ROC curves.

	Data set 1	Data set 2	Data set 3	Data set 4	Dataset 5	Dataset 6
Naïve Bayes	1.000	0.985	0.586	0.861	0.997	0.854
AdaBoost.M1	1.000	0.958	0.546	0.807	0.989	0.982
kNN	1.000	0.946	0.727	0.867	0.869	0.950
PNN	1.000	0.969	0.285	0.969	NaN	1.000
SVM	1.000	0.769	0.500	0.500	0.870	0.800

Table 5 demonstrates the measured time parameters in milliseconds (The *UserCPU_Time_millis_training* and the *UserCPU_Time_millis_testing* attributes in Weka), it is the time to build a model and the time of evaluation. As DTREG doesn't have similar metrics, we can not demonstrate the training and testing times for the PNN. As noted, this information here is just for an overview.

Table 5: Measured times (in milliseconds).

	Data set 1	Data set 2	Data set 3	Data set 4	Dataset 5	Dataset 6
Naïve Bayes	0.54 / 0.79	1.00 / 4.31	0.12 / 0.34	0.98 / 5.99	38.57 / 175.07	0.92 / 3.20
AdaBoost.M1	1.19 / 0.04	17.79 / 0.14	2.41 / 0.02	16.21 / 0.13	705.81 / 3.58	16.27 / 0.13
kNN	0.13 / 4.63	0.20 / 13.16	0.05 / 0.22	0.19 / 12.56	10.28 / 14 815.59	0.19 / 12.03
PNN	NaN	NaN	NaN	NaN	NaN	NaN
SVM	87.76 / 8.31	373.34 / 37.01	12.19 / 1.16	256.86 / 23.92	20 914.80 / 1 075.76	502.89 / 49.58

4.7 Comparison

In this section, we compare our results we presented in the previous chapter.

We used different datasets, with different scenarios; these datasets represent a similar nature of the real datasets. We split our dataset into ten equal partitions, so that each turn was used for testing, and the remainder was used for training – the 10-fold cross-validation was used as the validation method.

Table 1 shows the characteristics of the datasets we tested with 5 pre-selected methods (namely: Naïve Bayes, AdaBoost.M1, kNN, PNN, and SVM). Table 2 displays the accuracy values obtained from our experiments.

The AUC is the ranking of accuracy. Table 4 shows the measured areas under the ROC curve.

Tape [57] gives a rough guide for classifying the accuracy of a diagnostic test in the traditional academic point system; he interprets the AUC score as following:

- 0.90 – 1.00 = excellent;
- 0.80 – 0.90 = good;
- 0.70 – 0.80 = fair;
- 0.60 – 0.70 = poor;
- 0.50 – 0.60 = fail;

According to this guide, our methods' performance were either good or excellent (with some exceptions).

We emphasize that the accuracy and the AUC are different concepts. The AUC and the best overall accuracy may not be consistent, depending on the proportion of the true value of your dataset [52].

We evaluated methods using the Friedman test. It detects differences in treatment over multiple datasets.

First, we ranked each row separately. We assigned the rank of "5" to the best performing method, the rank of “4” to the second best performing method, etc. Finally, the rank of “1” went to the poorest performing method. Then, we summed and calculated the average (mean) ranks in each group (column).

Table 6 shows ranked measures for non-parametric Friedman test:

Table 6: The Friedman test results.

Dataset	Naïve Bayes	AdaBoost.M1	kNN	PNN	SVM
1	1	1	1	1	1
2	1	3	4	2	5
3	2	3	1	5	4
4	3	4	2	1	5
~~5~~	~~1~~	~~2~~	~~4~~	~~Nan~~	~~3~~
6	4	2	3	1	5
Sum	**11**	**13**	**11**	**10**	**20**
Mean	**2,2**	**2,6**	**2,2**	**2**	**4**

As PNN was not implemented in Weka and DTREG has a 2 000 instance limit, we could not evaluate the PNN on dataset 5. Missing values in the Friedman test are allowed, but data on each row that has at least one result as a missing value is ignored.

The results presented in the previous chapter lead to the following observations:

- The experiments showed that the PNN is the best algorithm in terms of accuracy. It performed a little bit better than other methods. The only exception was the experiment on dataset 3 which contains only 212 instances, the method didn’t perform well on this dataset.

- Naïve Bayes, kNN and AdaBoost.M1 showed us also relatively good performance. We emphasize that kNN is dependent on the value of parameter k, in our experiments $k = 5$. kNN running time increases rapidly if $k > 1$, since all the training data must be stored and examined for each classification.

- The PNN and the SVM had computationally longer running times compared to other tested methods (the time to build a model).

- Naïve Bayes and kNN testing phases are significantly longer than their training phases.

- AdaBoost.M1 and SVM testing phases are significantly shorter than their training phases.

- We note that in literature it is mentioned that the PNN's training period is faster compared to the testing period [58].

- Dataset 1 only contained outliers that deviate significantly from the natural distribution, and these invalid instances were typical examples of erroneous sensors. All of our selected methods classified these errors excellently - the accuracy was 100.00%.

- Datasets 2, 4 and 6 contained sensor data, where we manually had added outliers. These datasets covered typical outlier classes we had studied in our data analysis. We saw quite a good performance in terms of accuracy, it seems that 2 000 instances are good enough to learn the nature of outlier patterns, and to have an ability to correctly model a dataset.

- Dataset 3 contained only 190.80 training instances and 21.20 testing instances, so, on 212 instances (205 valid and 7 classified as outliers), the experiments showed that the accuracy was poor on limited amount of training data. 212 instances was clearly too little for the PNN method (it failed, having a very low AUC of only 0.285 and the classification accuracy of 96.7%). kNN was the best performer in terms of accuracy. The less training data we have, the lower the accuracy. This is an indication of the learning algorithm's inability to correctly model the dataset.

- Dataset 5 contained 45 305 instances (45 182 classified as normal, 123 classified as abnormal), the experiments showed again that the more training data we have, the higher the accuracy. We didn't measure the accuracy for the PNN due to the limits of DTREG.
- Models should be trained on the same kind of data they need to analyze if you want high accuracy results.

Our experience confirms, that the amount of training data and its accuracy are correlated - the more training data we have, the more accuracy our methods are. Similarly, a small training set, limited in the number and nature of anomalies, restricts the outlier detection accuracy.

The experimental results demonstrated that the designed models are promising in terms of accuracy and computational time, with a low probability of false alarms.

The PNN as a method has great potential to be the base validation method in our designed application.

5 A comprehensive model for the application

In this chapter, we present a comprehensive model for the application.

5.1 Designed applications

We designed two stand-alone applications: 1) a data transferring application, 2) a data validation application.

5.1.1 Data transferring application

The application is responsible for transferring data from the ARGO FTP-server to the Marinexplore.org database. There are two transferring processes: one for the raw data, and one for the validated data for our learning methods. Relevant learning data is essential for our validation methods.

During the process (raw data context) we convert latitude and longitude into some square grid system format (i.e. MGRS), see Chapter 3.3.

5.1.2 Data validation appliction

The application examines the database to detect outliers. When a value fails a test, the appropriate flag is assigned.

The philosophy behind the flag assignment is that it is inadvisable to change data. Changes should only be made when it is clear what the change should be and that if a change was not made the data would be unusable. It will reserve us the opportunity to re-examine data.

5.2 Requirements

Sommerville [59] outlines that software system requirements are often classified as functional and non-functional requirements:

- Functional requirements. These are statements of services the system should provide, how the system reacts to particular inputs, and how the system should behave in particular situations. Functional requirements for a system describe what the system should do.

- Non-functional requirements. These are constraints on the services or functions offered by the system. They include timing constraints, constraints imposed by standards. Non-functional requirements are requirements that are not directly concerned with specific services delivered by the system to its users.

To choose the most appropriate architecture, we specifed the following non-functional requirements.

Table 7 summarizes non-functional requirements.

Table 7: Non-functional requirements.

Technical enivironment requirements	Application server runs on the Linux OS. Oracle is the database system software. The system works over the www (only the interface for expert/ administrator).
System integration requirements	System operates with other information systems only inside the company. The application shall properly operate with the legacy Oracle database.
Portability requrements	No portability requirements. No need to migrate the application from one operating environment to another in the future.
Maintability requirements	Maximum MTBF (Mean Time Between Failure) for failures shall be 2 weeks averaged over 12 months. Maximum MTTR (Mean Time to Repair) shall be 1 day avegared over 12 months.
Speed requirements	The application shall be able to handle 200 000 datapoint validations per day.
Capacity requirements	The requirement to handle 500 concurrent transactions. The application shall successfully handle a minimum of 1.5 billion datapoints.
Availability and reliability requirements	The planned up time the system's services are available for use is 95%. The application shall not loose persistant data due to power failures.
System value estimates	The system is not mission critical, the system outage influence to the cash flow is not significantly big.
Access control requirements	Only authorized experts/administrators are able to access to the application functions according to their privilege levels.
Encryption and authentication requirements	No need for encryption. The system must log all activities.
Flexibility requirements	Layered architecture. Modular design. Modifiability. Configurability. Adaptability. Extensibility.
Usability requirements	No usability requirements.
Customization requirements	No customazation requirements.
Legal requirements	No legal requirements.

5.3 Scalability

Wiegers [60] defines scalability as "the ability to grow to accommodate more users, data, servers, geographic locations, transactions, network traffic, searches, and other services without compromising performance or correctness." So, it is the ability to continue to function well as it changes in size or volume in order to meet user needs.

To scale horizontally (or scale out) means to add more nodes to a system, such as adding a new computer to a distributed software application. To scale vertically (or scale up) means to add resources to a single node in a system, typically involving the addition of CPUs or memory to a single computer [61].

We address two possible challenges: 1) computing complexity, it can be an important issue, 2) storage problems, dataset readings from storage into the main memory issues.

The architecture has to scale up and scale out.

As the technologies and design of the array will evolve as better instruments are built, and as the float array grows, the forecast is that in the future the amount of ARGO produced sensor data will also grow; thus, the scalability of this architecture has a high priority [42].

Apach Spark is an open-source cluster computing framework which has a fast and general engine for large-scale data processing. Spark is well suited to machine learning algorithms.

Marinexplore.org application servers run on Apache Spark framework. Since Spark's in-memory computing allows efficient performance, Spark is an ideal choice in terms of scalability.

Dennis [62] notes that client-server architectures (our validation application will be based on this type of architecture) have several benefits, i.e. scalability. That means it is easy to increase or decrease the storage and the processing capabilities of the servers. If one server is overloaded, there is a possibility to just add another server to have more memory resources, more disk space, etc.

Information systems that have high performance requirements are best suited to client-server architectures.

5.4 Architecture

5.4.1 Data transferring application

The modular system is implemented in the application server, the server performing all functions. It is referred to as layered server-based architectures. An application server executes these services automatically. As there is no data management by the user (domain expert), no user-interface is implemented.

5.4.2 Data validation application

As the validation application is computationally intensive, Sommerville [59] suggests to design application that is implemented on the two-tier client-server architecture, the presentation layer is implemented on the client and all other layers (data management, application processing, database, etc.) are implemented on the application server (data storage, data access logic, application logic).

A web browser is used as the client (presentation logic only). Most users (administrators, domain experts) interact with the web interface to perform data validation management, or to display statistics/metrics.

The server side is responsible for all the computation. The client side is responsible for all user interactions; relevant information is displayed in a web browser.

A data flow diagram displays how data flows through the system (data validation context). It shows how data is processed by the system in terms of inputs and outputs.

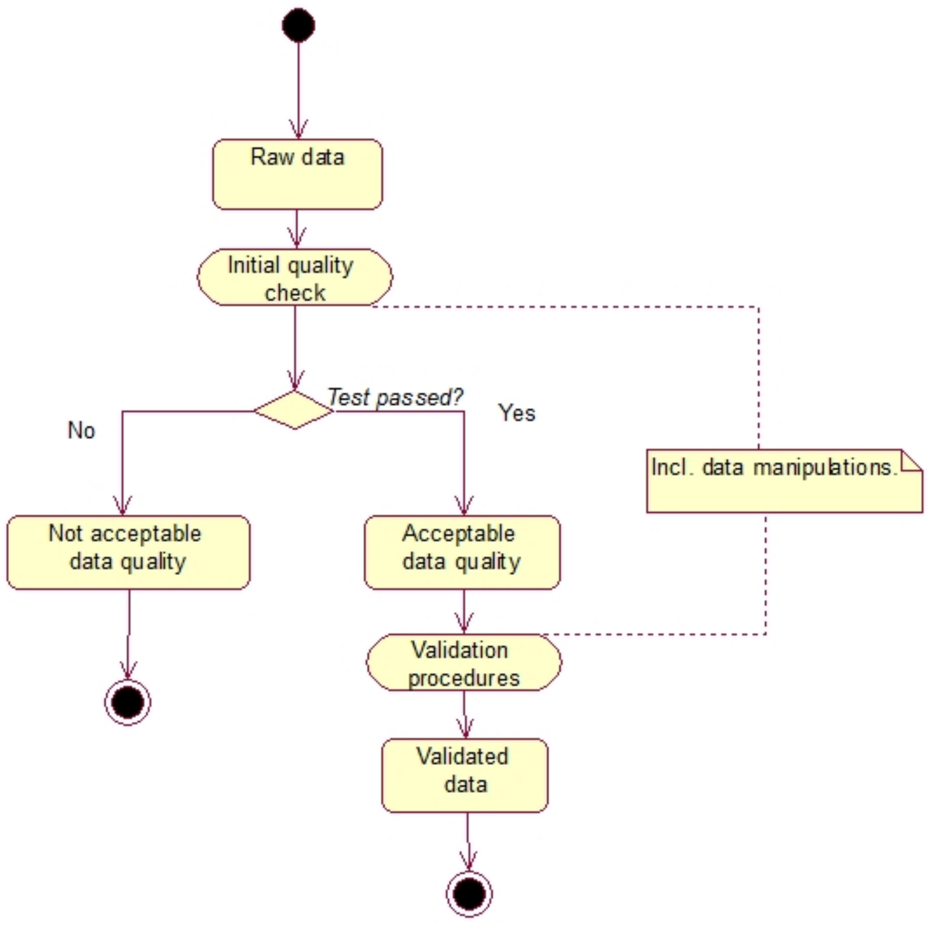


Figure 30: The simplified high-level overview of the data flow (validation process only). We use datasets from the database as input data, and the output is a binary flag (determines a potential outlier) that is set up in the database.

To deal with observations that are spatiotemporally correlated, our model instances are engaged for a particular geographical location and particular timestamps as one of the inputs associated with each data point.

Our approach does not require human actions for either the establishment or the adjustment of outlier detection thresholds.

6 Results

This section presents the results from the Marinexplore.org project on outlier detection methods.

This thesis studied the problem of outlier detection in oceanic sensor data. This project involved a comprehensive comparison between 5 algorithms. In this thesis we provided a comprehensive application model to transfer the data (both, raw and validated data) and to perform outlier validation procedures.

Although any comparative study is by its nature limited, a number of conclusions can be drawn: the main conclusion is that there is no single best algorithm, none of the techniques are superior to others across all datasets, but have features that make them effective for particular types of datasets and we can say that the best technique for a particular dataset depends crucially on the features of a dataset. The universal method is an idealistic fantasy.

The following is a summary of our contributions:

1. We expressed the appropriate data validation method. PNN as a method has great potential to be the base validation method for the oceanic datasets in our designed application.
2. Our calculations proved that a stable waterfront is located lower than 1 500 meters below the sea level. It is possible to use it as a reference. It helps to evaluate the state of the sensor. Our designed initial quality check procedures are partly based on this finding.
3. We demonstrated the comprehensive model for the application (the designed initial quality check procedures, the validation procedures). Our approach does not require human actions for either the establishment or the adjustment of outlier detection thresholds.

We make the following high-level conclusions:

- It is possible to shorten the validation period in Marinexplore.org.

- It is possible to improve the data quality of Marinexplore.org sensor data.
- A stable waterfront is located lower than 1 500 meters below the sea level - it is possible to use it as a reference. It helps to evaluate the state of the sensor.

The results can help a domain expert find the technique, which is best suited for an application domain.

This research does not pretend to be the sole and only way to choose the best method. The main aim of the research instead, is to introduce one possible solution that is usable, competitive, and effective.

Our approach is applicable more broadly than only in Marinexplore.org, it can also be applied to other actors in the same field.

6.1 Future ideas

Due to the time limit, several interesting ideas have not been implemented yet. To name only some of them:

- The model already exists and we plan to apply the same approach to other types of devices (gliders, fixed position stations, research ships, sea animals), so we can broaden the use of the approach.
- We plan to figure out when a nearby float/sensor is determined to have a similar response. We can improve the quality check procedures in the current domain.

The detection accuracy of the PNN can be improved by periodically retraining it. The training set should reflect current trends and outliers detected by ARGO experts.

Tulemused

Käesolev peatükk tutvustab ning võtab kokku magistritöö olulisemad tulemused.

Magistritöö peamine eesmärk oli võrrelda ning välja pakkuda efektiivne anomaaliate tuvastamise meetod, et parandada andmekvaliteeti Marinexplore.org sensorandmetes. Marinexplore.org protsessides puudusid tõhusad andmekvaliteedi kontrollimise mehhanismid, seepärast on andmekvaliteediga seotud mitmeid probleeme.

Magistritöö käsitleb anomaaliate tuvastamist ookeane katva poide võrgustiku sensorite töös. Võrgustikku kuulub enam kui 3000 ujuvpoid, mis perioodiliselt sukeldudes ja pinnale tõustes mõõdavad erinevaid keskkonnaparameetreid.

Töö lahendab andmekvaliteediga seotud probleeme Marinexplore.org sensorandmetes. Viidi läbi põhjalik analüüs sobiva meetodi leidmiseks. Disainiti sobiv rakendusmudel.

Kokkuvõtvalt töö olulisemad tulemused:

- Erinevate andmeanalüüsi meetodite kasutamine anomaaliate leidmiseks sensorandmetest: Naïve Bayes, AdaBoost.M1, k-lähima naabri meetod (kNN), tõenäosuslik närvivõrk (PNN), tugivektor-masinad (SVM).
- 1500 meetrist sügavamal asuva suhteliselt stabiilse keskkonna kasutamine sensorite korrasoleku hindamiseks.
- Nende meetodite võrdlus, kasutades tunnustatud võrdlustehnikaid: 10-kordne ristkontroll (*10-fold cross-validation*), ROC kõverad, Friedmani test.
- Neid meetodeid kasutava rakenduse üldine arhitektuur. Rakendamisega lüheneb andmete valideerimisprotsess märgatavalt, paraneb oluliselt sensorandmete kvaliteet.

Uurimus ei pretendeeri ainsaks ja ainuvõimalikuks lähenemiseks leidmaks mõistlikum meetod andmekvaliteedi parandamiseks. Töö eesmärk oli leida kasutatav, mõistlik ja efektiivne meetod.

Välja pakutud lahendust ning selle rakendusmudelit on võimalik implementeerida lisaks Marinexplore.org platvormile ka teistel sensorandmetega tegutsejatel.

References

[1] ARGO Project Office. (2015, April) ARGO float design. Document.

[2] Arindam Banerjee, Vipin Kumar Varun Chandola. (2009, July) Outlier Detection : A Survey. Report.

[3] Michekine Kamber, Jian Pei Jiawei Han, *Data Mining: Concepts and Techniques*, 3rd ed. Waltham, USA: Morgan Kaufman Publishers, 2012.

[4] Planet OS. (2015, March) Marineplore.org. [Online]. https://planetos.com/company

[5] CSIRO Climate Reponse. (2014, January) Bio robots make a splash in the Indian Ocean. [Online]. https://blogs.csiro.au/climate-response/stories/bio-robots-make-a-splash-in-the-indian-ocean/

[6] Yves Desaubies, Pierre-Yves LeTraon, Robert Molinari, W. Brechner Owens, Stephen Riser, Uwe Send, Kensuke Takeuchi, Susan Wijffels Dean Roemmich, "Argo: The Global Array of Profiling Floats," ARGO, Report 2012.

[7] ARGO Project Information Center, "ARGO, an array of profiling floats observing the ocean real-time," Ramonville, Report 2006.

[8] NOAA, "ARGO Grey list," Silver Spring, Report 2015.

[9] Ann Thresher, Esmee Van Wijk Susan Wijffels, "Argo Australia – 2009 Activities ," The Australian Center for Atmosphere, Weather and Climate Research: a joint partnership between the Australian Bureau of Meteorology and CSIRO, Aspendale, Report 2008.

[10] Robert Keeley, Thierry Carval and the Argo Data Management Team Annie Wong. (2014, November) Argo Quality Control Manual. Document.

[11] Charu C. Aggarwal, *Outlier Analysis*. New York: Springer Science+Media, 2013.

[12] John S. Witte Robert S Witte, *Statistics*, 9th ed. Hoboken, USA: Wiley, 2009.

[13] Donato Malerba, Michael May, Maguelonne Teisseire Gennady Andrienko, "Mining spatio-temporal data," Fraunhofer Institut Autonome Intelligente Systeme, Sankt Augustin, Report 2006.

[14] Robert Tibshirani, Jerome Friedman Trevor Hastie, *The Elements of Statistical Learning*, 2nd ed. Stanford, USA: Springer, 2009.

[15] Michael Siebers. (2013, January) Universität Bamberg. [Online]. http://www.cogsys.wiai.uni-bamberg.de/teaching/ws1213/ml/slides/statlog.pdf

[16] D.J. Spiegelhalter, C.C. Taylor D. Michie, *Machine Learning, Neural and Statistical Classification*. Cambridge, UK: University of Strathclyde, February 1994.

[17] Alexandru Niculescu-Mizil Rich Caruana, "An Empirical Comparison of Supervised Learning Algorithms," *Proceeding ICML '06 Proceedings of the 23rd international conference on Machine learning* , pp. 161-168, May 2006.

[18] Phillip H. Sherrod. (2014, January) DTREG Predictive Modeling Software. Manual.

[19] Lior Rokach Oded Maimon, *The Data Mining and Knowledge Discovery Handbook*. USA: Springer, 2005.

[20] Jim Austin Victoria J. Hodge, "A survey of outlier detection methodologies. Artificial Intelligence Review.," *Artificial Intelligence Review*, pp. 85-126, November 2005.

[21] Trevor Whitney. (2015, March) Classification. [Online]. http://trevorwhitney.com/data_mining/classification

[22] Nicola Ward Petty, Beware of Excel Histograms, November 2012, https://learnandteachstatistics.wordpress.com/2012/11/12/beware-of-excel-histograms/.

[23] Sarah Boslaugh, *Statistics in a Nutshell*, 2nd ed. Sebastopol, USA: O'Reilly Media, 2013.

[24] Vic Barnett, *Outliers in statistical data.*, 3rd ed. New York, USA: John Wiley & Sons Ltd., 1994.

[25] Sherry Towers. (2013, October) K-means clustering. [Online]. http://sherrytowers.com/2013/10/24/k-means-clustering

[26] Rachel Schutt Cathy O'Neil, *Doing Data Science*. Sebastopol, USA: O'Reilly Media, 2014.

[27] Zhengxin Chen, *Data Mining and Uncertain Reasoning*. New York, USA: Wiley-Interscience, 2001.

[28] Paolo Giudici, *Applied Data Mining: Statistical Methods for Business and Industry (Statistics in Practice)*. West Sussex, UK: Wiley, 2003.

[29] Peter Flach, *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge, UK: Cambridge University Press, 2012.

[30] Wikipedia. (2015, March) BIRCH. [Online]. http://en.wikipedia.org/wiki/BIRCH

[31] Tom Fawcett Foster Provost, *Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking*. Sebastopol, USA: O'Reilly Media, 2013.

[32] Michael J. A. Berry Gordon S. Linoff, *managementData mining Techniques for Marketing, Sales, and Customer Relationship* , 3rd ed. Indianapolis, USA: Wiley, 2011.

[33] Wikipedia. (2015, March) DBSCAN. [Online]. http://en.wikipedia.org/wiki/DBSCAN

[34] Michael Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*. Essex, UK: Addison-Wesley, 2004.

[35] Dan W. Patterson, *Artificial Neural Networks*. New Jersey, USA: Prentice Hall, 1998.

[36] Robert Groth, *Data Mining: Building Competitive Advantage*. New Jersey, USA: Prentice Hall, 1999.

[37] Jake Vander-Plas. (2012, October) Neural Network Diagram. [Online]. http://www.astroml.org/book_figures/appendix/fig_neural_network.html

[38] P.Latha, R.Pon Selvi, P.Nisha K.Muthukannan, "Survey on Classification Techniques for Plant Leaf Disease Classification," *Reasearch and Scientific Innovation Society*, vol. II, pp. 15-18, February 2015.

[39] Jyothi R. Tegnoor P. S. Hiremath. (2013, June) Follicle Detection and Ovarian Classification in Digital Ultrasound Images of Ovaries. [Online]. http://www.intechopen.com/books/advancements-and-breakthroughs-in-ultrasound-imaging

[40] Trevor Hastie, Robert Tibshirani Jerome Friedman, "Additive Logistic Regression: a Statistical View of Boosting," *Annals of Statistics*, vol. 28, pp. 337-407, 2000.

[41] Eibe Frank, Mark A. Hall Ian H. Witten, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Burlington, USA: Morgan Kaufman, 2011.

[42] ARGO Project Office. (2015, March) ARGO. [Online]. http://www.argo.ucsd.edu/index.html

[43] Andres Kull, "Argo outliers detection," Marinexplore.org, Tallinn, Report 2013.

[44] Anna Konstantinidou, Leonidas Perivoliotis, Gerasimos Korres Dimitris Kassis, "Comparison of Argo profiles observations against numerical model simulations in Ionian Sea," Attika, Report 2015.

[45] Jeff de La Beaujardière. (2012, January) IOOS Conventions for CSV and TSV Encoding. Document.

[46] Scantek Systems Inc. (2015, March) Military Grid Reference System. [Online]. http://www.legallandconverter.com/p50.html

[47] ARGO Project Office. (2013, January) ARGO Quality Control Manual. Document.

[48] UNESCO. (2010, December) GTSPP Real-Time Quality Control Manual.

[49] U.S. IOOS Program Office Validation. (2014, January) Manual for Real-Time Quality Control of In-situ Temperature and Salinity Data.

[50] Tom Fawcett. (2005, January) An introduction to ROC analysis. Report.

[51] Alberto Jiménez-Valverde, Raimundo Real Jorge M. Lobo, "AUC: a misleading measure of the performance of predictive distribution models," *Global Ecology and Biogeography*, vol. 1, January 2007, http://www2.unil.ch/biomapper/Download/Lobo-GloEcoBioGeo-2007.pdf.

[52] Tomaz Curk Miha Vuk, "ROC Curve, Lift Chart and Calibration Plot," *Metodoloski zvezki*, vol. 3, pp. 89-108, January 2006.

[53] GraphPad Software, Inc. (2015, April) Friedman's test. [Online]. http://www.graphpad.com/guides/prism/6/statistics

[54] Toshio Suga and Dean H. Roemmich. (2010, October) International ARGO: Achievements and Perspectives. Document.

[55] Thomas G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," Oregon State University, 1997, Report.

[56] Malik Magdon-Ismail, Hsuan-Tien Lin Yaser S.Abu-Mostafa, *Learning from Data*. Pasadena, USA: AMLbook.com, 2012.

[57] Thomas G. Tape. (2015, April) Interpreting Diagnostic Tests. [Online]. http://gim.unmc.edu/dxtests/Default.htm

[58] Anna Maria Colla, Stefano Rovetta, Rodolfo Zunino Fabio Ancona, "Implementing probabilistic neural networks," *Neural Comput & Applic*, vol. 7, pp. 37-51, January 1998.

[59] Ian Sommerville, *Software Engineering*, 9th ed. Boston, USA: Pearson, 2009.

[60] Joy Beatty Karl Wiegers, *Software Requirements*, 3rd ed. Redmond, USA: Microsoft Press, 2013.

[61] Wikipedia. (2015, March) Scalability, Horizontal and vertical scaling. [Online]. http://en.wikipedia.org/wiki/Scalability#Horizontal_and_vertical_scaling

[62] Barbara Haley Wixom, David Tegarden Alan Dennis, *System Analysis Design, UML Version 2.0, an Object-Oriented Approach*, 4th ed. New Jersey, USA: Wiley, 2012.

[63] Wikipedia. (2015, March) R. [Online]. http://en.wikipedia.org/wiki/R_(programming_language)

[64] Kentaro Ando Eitarou Oka, "Stability of Temperature and Conductivity Sensors of Argo Profiling Floats," *Journal of Oceanography*, vol. 60, pp. 253-258, 2004.

[65] ARGO Project Group. (2015, April) ARGO. [Online]. http://www.argo.ucsd.edu/About_Argo.html

[66] Scientificsoftware-solutions.com. (2015, March) ROC Curve Analysis. [Online]. http://www.scientificsoftware-solutions.com/pages.php?pageid=71

Appendix 1: ARGO real-time quality check tests

ARGO realtime quality check tests [47]:

1. Impossible date — Tests for sensible observation date and time values.
2. Impossible location — Tests for sensible observation latitude and longitude values.
3. Position on land — Tests whether the observation position is on land.
4. Impossible speed — Tests for a sensible distance travelled since the previous profile.
5. Global range — Tests that the observed temperature and salinity values are within the expected extremes encountered in the oceans.
6. Regional range — Tests that the observed temperature and salinity values are within the expected extremes encountered in particular regions of the oceans.
7. Deepest pressure — Tests that the profile does not contain pressures higher than the highest value expected for a float.
8. Pressure increasing — Tests that pressures from the profile are monotonically increasing.
9. Spike — Tests salinity and temperature data for large differences between adjacent values.
10. Gradient — Tests to see if the gradient between vertically adjacent salinity and temperature measurements are too steep.
11. Digit rollover — Tests whether the temperature and salinity values exceed a floats storage capacity.
12. Stuck value — Tests for all salinity or all temperature values in a profile being the same.
13. Density inversion — Tests for the case where calculated density at a higher pressure in a profile is less than the calculated density at an adjacent lower pressure.

14. Grey list — Tests whether the sensor identifier is present in a list that has been collated to identify sensors, which are experiencing problems.

15. Sensor drift — Tests temperature and salinity profile values for a sudden and important sensor drift.

16. Frozen profile — Tests for the case where a float repeatedly produces the same temperature or salinity profile (with very small deviations).

17. Profile envelope — Tests that temperature and salinity profile values are within an envelope of permitted values within depth ranges. On failure the QC test identifier value 2000000000000H is used.

18. Freezing point — Tests if the profile temperature at a given pressure and salinity is less than the calculated freezing point temperature. On failure the QC test identifier value 4000000000000H is used.

Appendix 2: Distributions of test datasets

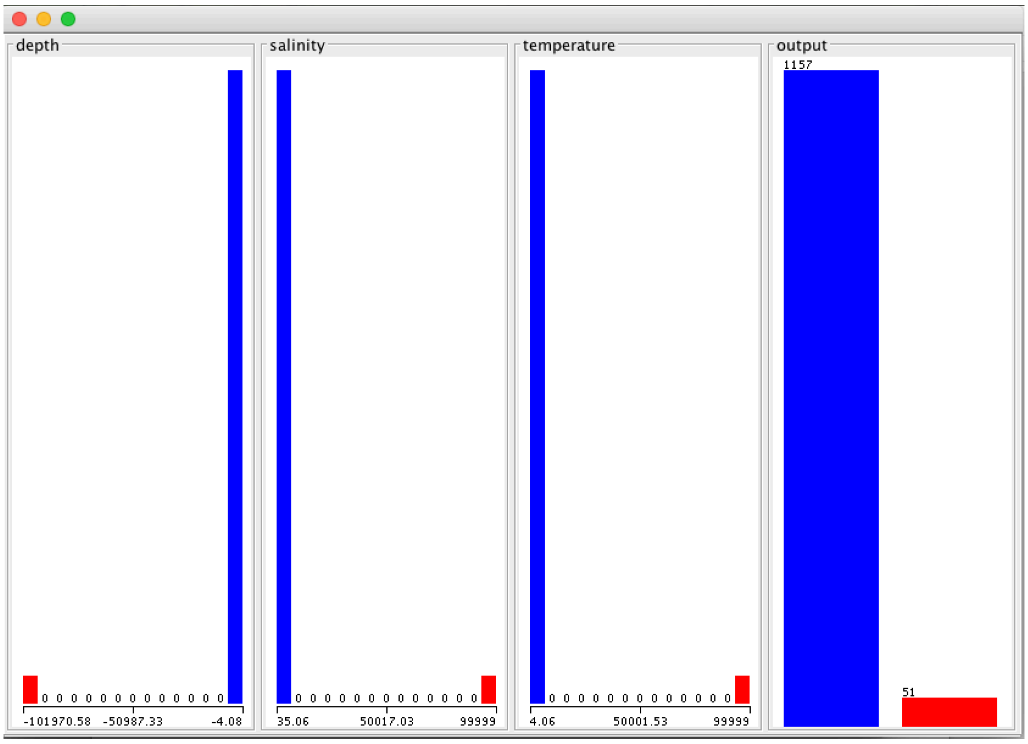


Figure 31: Histogram of dataset 1.

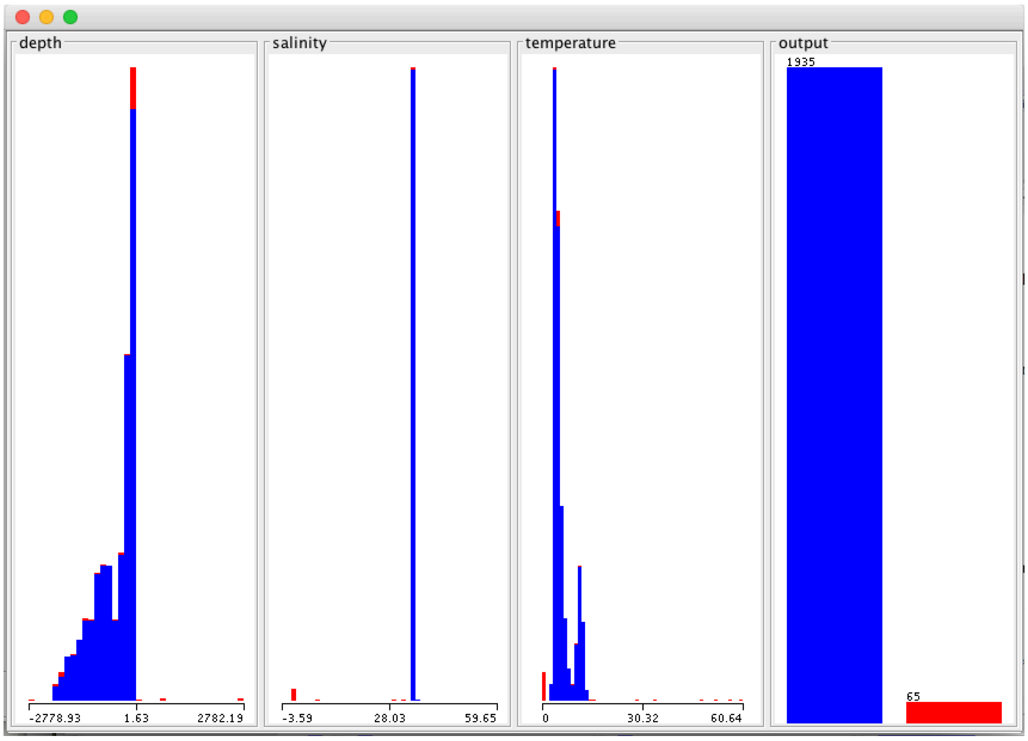


Figure 32: Histogram of dataset 2.

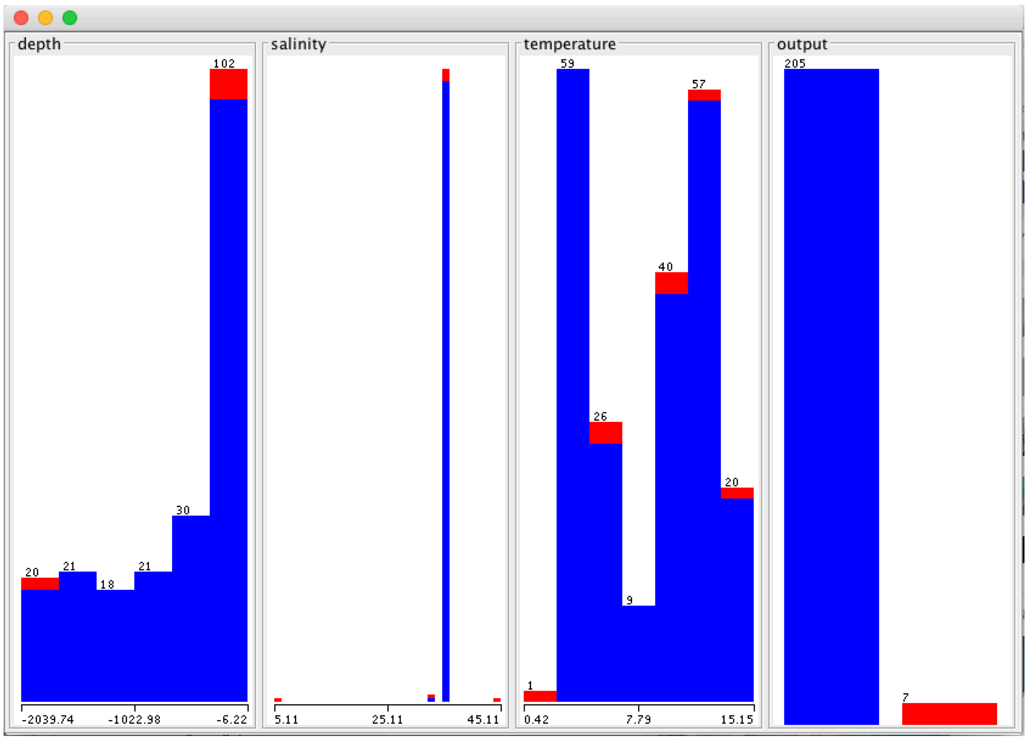


Figure 33: Histogram of dataset 3.

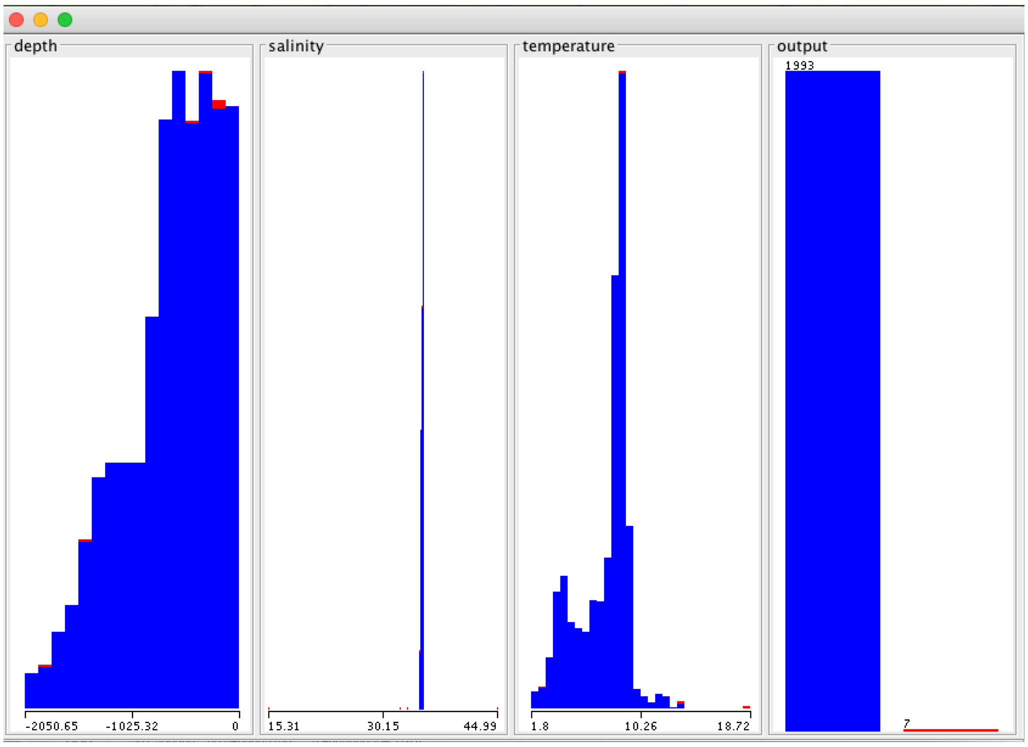


Figure 34: Histogram of dataset 4.

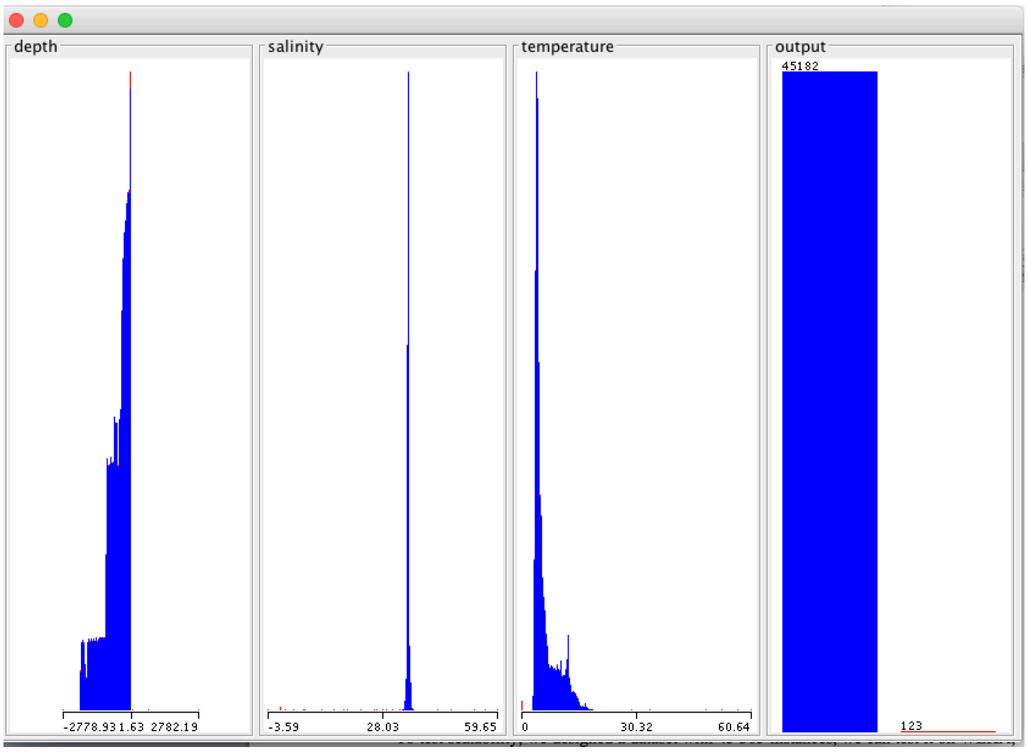


Figure 35: Histogram of dataset 5.

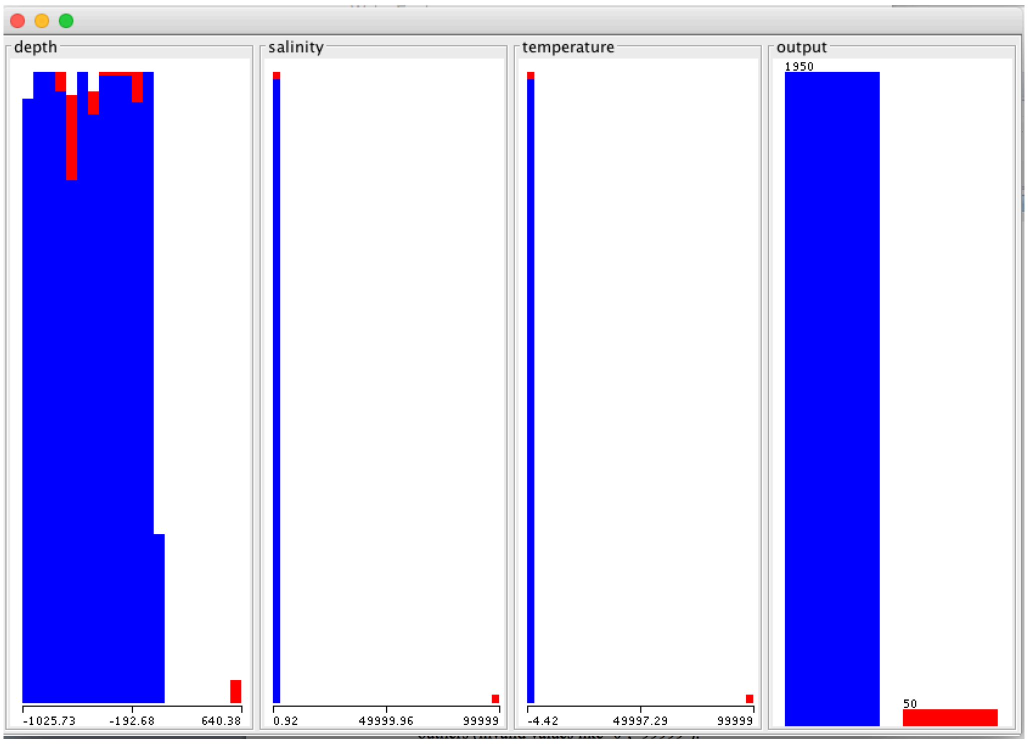


Figure 36: Histogram of dataset 6.

Appendix 3: Parameter envelopes

Table 8: The global impossible parameters [48].

Parameter	Min	Max	Unit
Depth	-10 000	0	m
Salinity	0	41	psu
Water pressure	0	10 000	dbar
Water teperature	-2.0	40	degrees C

Table 9: The parameter envelopes [48].

Depth (m)	Water temperature (degrees C)	Salinity (psu)
0 to 25	-2.0 to 37	0 to 41
> 25 to 50	-2.0 to 36	0 to 41
> 50 to 100	-2.0 to 36	1 to 41
> 100 to 150	-2.0 to 34	3 to 41
> 150 to 200	-2.0 to 33	3 to 41
> 200 to 300	-2.0 to 29	3 to 41
> 300 to 400	-2.0 to 27	3 to 41
> 400 to 1 100	-2.0 to 27	10 to 41
> 1 100 to 3 000	-1,5 to 18	22 to 38

Appendix 4: Tools

Weka

Weka is a collection of machine learning algorithms for solving real-world data mining problems.

There is a metalearner implemented in Weka called CVPParameterSelection; it searches for the best parameter settings by optimizing cross-validated accuracy on the learning data; there is cross-validation used.

We performed our experiments on Weka (exception was the PNN method, it is not implemented for Weka yet).

Marinexplore.org

Marinexplore.org is the largest free Internet portal for public data in the oceanic, atmospheric and climate domain. It provides powerful search, exploration and visualization tools to access in-situ, raster and model datasets from 33 organizations (global, regional and local) from nearly 40 000 in-situ devices and 50 data products, with over 7 700 professionals registered in the community.

We acquired our datasets from the Marinexplore.org database using the Marinexplore.org Datastudio.

Tableau

Tableau Software is an American computer software company, which produces a family of interactive data visualization products focused on business intelligence.

We analyzed our data on Tableau, we also sketched charts in Tableau.

DTREG

DTREG is a program designed for predictive modeling and forecasting.

We performed our experiments for the PNN method in this software program as it is not implemented for Weka.

R

R is a programming language and a software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing statistical software and data analysis [63].

R does statistics and graphics very well, it is superior in that way to many packages.

We used R for our data analysis.

Appendix 5: Configurations

Method configurations:

- PNN configuration in DTREG: the kernel function was Gaussian, sigma values were computed for each variable (sigma values were between 0.0001 and 10), model was optimized to minimize error;

- Naïve Bayes configuration in Weka: “$weka.classifiers.bayes.NaiveBayes$”;

- The kNN configuaration in Weka: “$weka.classifiers.lazy.IBk\ -K\ 5\ -W\ 0\ -$ A "$weka.core.neighboursearch.LinearNNSearch\ -$ A \\"$weka.core.EuclideanDistance\ -R\ first - last$\\"""”;

- The SVM configuration in Weka: “$weka.classifiers.functions.LibSVM\ -S\ 0\ -$ $K\ 2\ -D\ 3\ -G\ 0.0\ -R\ 0.0\ -N\ 0.5\ -M\ 40.0\ -C\ 1.0\ -E\ 0.001\ -P\ 0.1\ -$ $model\ /Users/kristianallikmaa\ -seed\ 1$”;

- AdaBoost.M1 configuration in Weka: “$weka.classifiers.meta.AdaBoostM1\ -$ $P\ 100\ -S\ 1\ -I\ 10\ -W\ weka.classifiers.trees.DecisionStump$”.