Contents

Document history

Version	Date	Author (Unit)	Description
v0.3			The test and evaluation framework discussed and established
v0.6			Initial draft finished and reviewed by the PSRG
v1.0			Final version of document

1 Executive summary

This document is by no means ready, so I haven't written anything here.

2 Introduction

Even though the number of algorithms designed for learning on streaming data is increasing, there is still not a unified and well accepted way for evaluating them. This is because testing and evaluating algorithms that are designed to work on streaming data are generally more complicated than those designed to work on non streaming data. There are both statistical and computational reasons for this.

If the data instances in a data stream are identically and independently distributed (i.i.d.), the only challenge compared to static i.i.d. data is that the data streams are open ended, hence can be seen as a static i.i.d data that continuously grow in size. Evaluation methods related to these streams are closely related to the evaluation methods that are known for static i.i.d. data.

A generalization of an i.i.d. data stream is a *stationary* data stream, which is data that is generated from a stationary process. A stationary process is a stochastic process where the joint probability distribution over any sized time window do not change when shifted in time.[1] Various performance measures on stationary data streams have been proposed in the literature, including those involving loss functions [1–3]. The loss function for a regression problems is a function of both the predicted value and the ground truth, while the loss function on classification problems is a function of predicted and real class labels. The holdout error is basically the average zero-one loss on a holdout dataset of fixed size. The predictive sequential, or *prequential* error is defined as the average loss function up to time step t, where t is the current time step.

[TODO: HL says: I've removed a part here as I couldn't get any sense out of it, and believe we agreed to kill it. Check, please. Other parts have been moved to the next section.]

[TODO: HL says: I've also removed a part here on AUC that used stationarity, concept drift etc, that was not yet defined and later was used again (in context). Check.]

In the next section, relevant methodologies for evaluation of both batch and streaming algorithms are identified and discussed. For ease of exposition we focus on binary classification in this document, although many of the approaches we discuss can be generalized to multi classification or regression. This section forms the foundation of the next three sections, where the exact evaluation routines for each use case provider is given. These sections contains a description of the requirements related to evaluation as described in Deliverable 1.2 and a short description of the algorithms and the data. At the end of each of these sections, the methods for evaluating predictive and runtime performances are exposed and discussed. Section 7 concludes the report.

3 Test and evaluation methodology

As discussed in the introduction, finding appropriate performance measures on streaming data is more difficult than finding performance measures on non streaming data. We will therefore start by discussing some relevant performance measures for classification of i.i.d. non streaming data.

[1] Consequently, parameters such as the mean and variance do not change over time for a stationary data stream.

3.1 Performance measures for classification on i.i.d. data

Consider a dataset of fixed size n, where each instance is independently drawn from a joint probability distribution $P(\boldsymbol{X}, Y)$,. Here $\boldsymbol{X}$ and Y are random variables; $\boldsymbol{X}$ is known as the explanatory variable and Y is the class label. These two variables have output spaces Ω_X and Ω_Y, respectively. In classification, we typically consider a hypothesis function $h : \Omega_X \to \Omega_Y$, where Ω_Y is a finite set of class labels; note that $\Omega_Y = \{0, 1\}$ for a binary classification problem. Ω_X is a space of all possible explanatory vectors. In terms of evaluating the performance of h, we use a dataset of input-output pairs $(\boldsymbol{x}_i, y_i)$ that are independently drawn from $\Omega_X \times \Omega_Y$. The result of such an experiment can be shown in a confusion matrix. An example of a confusion matrix is shown Table 3.1, where the classifier is attempting to distinguish cats, dogs and rabbits.

		Predicted class		
		Cat	Dog	Rabbit
Actual class	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

Table 3.1: Example of a confusion matrix. A classifier is labelling instances as either cats, dogs or rabbits. The accuracy is the sum of the diagonal elements divided by the total number (in this case 19/27).

A global measure of the classification algorithm is the classification accuracy which is calculated as the sum of the diagonal elements divided by the sum of all numbers in the table (in this case 19/27). It is important to note that the accuracy is not telling the whole story of the classification rule. For instance, by looking at the table it is seen that the algorithm distinguishes cats from rabbits quite easily, but it is much harder to distinguish cats from dogs. This property cannot be realized by looking at the accuracy alone.

Moreover, accuracy is also very dependent on how evenly the classes are distributed. For instance, when there are a lot more instances of one class compared to the others, a naive classification rule that always predict the majority class will get a high accuracy, even though the method is not using any of the information that is contained in the explanatory variables. These are reasons for inspecting the full confusion matrix for interpreting the classification rule.

In order to inspect various properties of a classification rule, various performance measures can be calculated from the confusion matrix. We have decided to limit this exposition to binary classification, while being aware that most of the performance measures can be expanded to discuss multi class classifiers.

For a binary classifier, such as classifying cats and non-cats, the confusion matrix is of size two - by - two; an example is shown in Table 3.2. It is common to introduce positives and negatives instead of the class labels. A true positive is therefore an actual cat that has been predicted to be a cat, while false positives, true negative and false negatives are defined in an equivalent manner.

The results are commonly shown in a confusion table (see Table 3.3), which is not the same as a confusion matrix. From a confusion table it is easy to calculate numerous numbers that describes the classification rule. Specifically, we mention the true positive and false positive

		Predicted class	
		Cat	Not cat
Actual class	Cat	5	3
	Not cat	2	17

Table 3.2: Example of a confusion matrix for a classifier of cats and not cats. The accuracy is 22/27.

		Actual Condition	
		Positive	Negative
Test outcome	Positive	5	2
	Negative	3	17

Table 3.3: Example of a confusion table for a classifier of cats and not cats. The true positives and true negatives are on the diagonal, while the two other numbers are the false positives and the false negatives.

rates. True positive rates, also known as recall, is the number of true positives divided by the total number actual positives. The false positive rates, also known as fall-out, is the number of false positives divided by the total number actual positives. By investigating various numbers that can be deduced from the confusion table, it is possible to discuss classification rules, even when the datasets are unevenly distributed. That is, when some class label have more instances than others.

However, these numbers do not take into account that some misclassifications might be more costly than others. For instance, a false positive might be more costly than a false negative. For instance in the case of diagnosing cancer, it might be more costly to not treat a person that is sick, compared to treating a healthy person. Moreover, the cost of each false positive (or false negative) may not be constant either. For instance, if the classifier is predicting whether a client in a bank will default a loan or not, the cost is clearly related to the size of the loan in question. The next subsection includes a procedure to include such costs in a performance measure.

3.1.1 Empirical risk

In mathematical optimization, statistics, decision theory and machine learning, a loss function or a cost function is a function that maps an event or values of one or more variables onto a real number that is intuitively representing some *cost* associated with the event. Loss functions can be used on optimization problems, when an algorithm or a method is optimized by minimizing the loss function. Moreover, loss functions are frequently used to diagnose and compare various algorithms or methods. We define the *loss function* as a real and lower-bounded function L on $\Omega_X \times \Omega_Y \times \Omega_Y$. The value of the loss function at an arbitrary point $(x, h(\boldsymbol{x}), y)$ is interpreted as the loss, or cost, of taking the decision $h(\boldsymbol{x})$ at $\boldsymbol{x}$, when the right decision is y. Notice that the loss function is defined to also depend on $\boldsymbol{x}$. This is of high practical use, because a certain misclassification might be more expensive than another.

In the frequentist perspective, the expected loss is often referred to as the risk function [4]. It is obtained by taking the expected value over the loss function with respect to the probability distribution $P(\boldsymbol{X}, Y) : \Omega_X \times \Omega_Y \to \mathbb{R}^+$. The *risk function* is given by

$$R(h) = \int_{\Omega_X, \Omega_Y} L(x, h(\boldsymbol{x}), y) dP(\boldsymbol{x}, y). \tag{3.1}$$

In the case when the costs are independent of x and also that there is no cost related to correct classification, the risk function reduces to the well known expected cost of misclassification (ECM)

$$ECM = c(1|0)p(1|0)p_0 + c(0|1)p(0|1)p_1. \tag{3.2}$$

Here, $c(1|0)$ is the cost for misclassifying an item of class zero as class one and $p(1|0)$ is the misclassification probability given class zero. The quantities $c(0|1)$ and $p(0|1)$ are defined equivalently, while p_0 and p_1 are the priors.

In general, the risk $R(h)$ cannot be computed because the distribution $P(\boldsymbol{X}, Y)$ is unknown. However, we can compute an approximation, called empirical risk, by averaging the loss function on the data set $\mathcal{D}$ of size N, where each element in $\mathcal{D}$ is $(\boldsymbol{x}_i, y_i)$. The empirical risk is given by

$$R_{emp}(h, \mathcal{D}) = N^{-1} \sum_{i=1}^{N} L(x_i, h(\boldsymbol{x}_i), y_i). \tag{3.3}$$

Many supervised learning algorithms are optimized by finding the h in a hypothesis space $\mathcal{H}$ that minimizes the empirical risk on a data set that is used only for training the classification rule. In this case, a separate data set is needed for testing the classifier. In this document, we focus only on testing classifiers. We only use empirical risk to quantitatively describe the methods.

3.1.2 Evaluation of families of classification rules

So far we have discussed how to evaluate a single classification rule. However, probabilistic classifiers operate by calculating the probability distribution $P(Y|\boldsymbol{x})$, and then comparing this probability to a certain threshold. We will call this estimated probability the output function $q : \Omega_X \to \mathbb{R}^+$. In the probabilistic framework the range of q is $[0, 1]$, but this restriction is not necessary for this theory to be well-defined. The potential classification rules are the family of hypothesis functions $\mathcal{H}$, where each element $h_T : \Omega_X \to \Omega_Y$ has the form

$$h_T(\boldsymbol{x}) = \begin{cases} 0 & \text{if} \quad q(\boldsymbol{x}) \leq T \\ 1 & \text{otherwise.} \end{cases} \tag{3.4}$$

It is of interest to evaluate all these classification rules. The receiver operating characteristic (ROC) is a plot of the true positive rate as a function of the false positive rate, as the threshold T varies over all relevant values. An example is seen in figure 3.1.

The ROC curve allows a visual exposition of the family of classifiers $\mathcal{H}$, in the sense of true positive rates and false positive rates. In particular, it is easy to see which threshold is needed for a certain hit rate or true positive rate. This plot is clearly independent on class distributions and the whole cost content.

Moreover, it is also of interest to reduce all this information into a single number. The area under the ROC curve, also known as AUC, is such a number. AUC is independent of T. The AUC is a popular method for evaluating classification problems where class imbalance is vital,

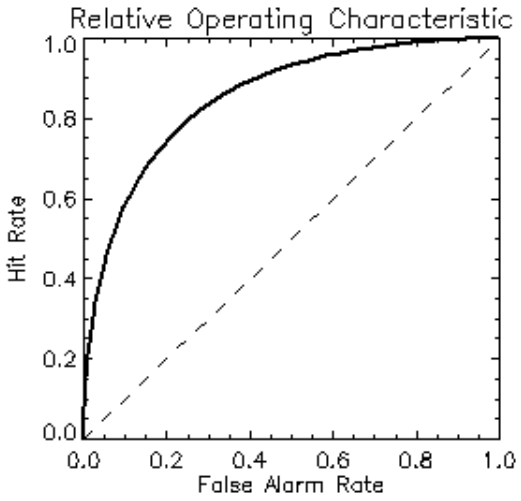

Figure 3.1: Receiver operating characteristics curve. True positive or hit rate is shown as a function of false positive or false alarms rate. The dashed line shows the ROC curve of the random guess.

because it is invariant to the class distribution. AUC has the statistical interpretation that it is the probability that a random member of the "positive" class is scored higher than a random member of the "negative" class. AUC is therefore a measure of the *ranking ability* of the output function. This is particularly relevant if one wants to change the classification threshold as a consequence of changing class distributions or changing misclassification costs [5]. Moreover, it has also been pointed out that AUC is more preferable than accuracy for model evaluation [6].

We will now discuss the underlying statistical properties of AUC. First, let $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$ be random variables with probability distributions $P(\boldsymbol{X}|Y=0)$ and $P(\boldsymbol{X}|Y=1)$, respectively. We define the random variables $Q_0 = q(\boldsymbol{X}_0)$ and $Q_1 = q(\boldsymbol{X}_1)$. To summarize, Q_0 is basically the output value you get when you pick a random sample of class zero and calculate the probability for it to be a member of class one; similarly for Q_1.

The probability $P(Q_1 > Q_0)$ is of interest, because this says what is the probability that if you take one sample by random from each of the populations, what is the chance that the output value from sample one is higher than the output value of sample zero. This question is independent of T, meaning that the discussion about priors and costs are not needed. This probability is called the concordance probability.

Without exposing the details, it is possible to show that

$$\text{AUC} = P(Q_1 > Q_0). \tag{3.5}$$

It is important to note that nothing need to be assumed about the probability distributions of Q_0 and Q_1. Furthermore, it is worth noting that the concordance probability is exactly equal the common language effect size of the Mann-Whitney U test. We discuss the Mann-Whitney U test next.

3.1.3 Mann-Whitney U test

The Mann-Whitney U test [7] is a nonparametric test of the null hypothesis that two populations are the same against an alternative hypothesis where one population tends to have larger values than the other. It is the nonparametric version of common t-test, where both populations are assumed to be of normal distributions. It is nearly as efficient as the t-test on normal data and far more efficient on non normal data. In our context, we can use the Mann-Whitney U test to discuss whether values of Q_1 are consistently larger than values of Q_0.

A relevant assumption for our problem is that all samples of Q_0 and Q_1 are assumed to be independent of each other. On stationary streams, we know this is not true as two consecutive measurements are clearly dependent. However, in the context of interpretation, we can imagine that the samples of randomized.

A more important assumption is that under the null hypothesis, the probability distributions of Q_0 and Q_1 are exactly equal, while the alternative hypothesis is that the distributions have different means, but similar shapes. It is clearly a problem if for instance Q_0 is one sided exponentially distributed, while Q_1 is normally distributed with a larger mean. Other assumptions can be stated for this test as well. A more thorough discussion can be found in [8].

We define a data set $\mathcal{D}$ with n input-output pairs (x_i, y_i), independently drawn from $P(\boldsymbol{X}, Y)$. From the data set we have two populations $\boldsymbol{q}_0 = \{q(\boldsymbol{x}_i), | \, y_i = 0\}$ and $\boldsymbol{q}_1 = \{q(\boldsymbol{x}_i), | \, y_i = 1\}$. Their sizes are n_0 and n_1 so that $n_0 + n_1 = n$. Calculating the U statistics is straightforward, where these two values are obtained

$$U_0 = \sum_{i=1}^{n_0} \sum_{j=1}^{n_1} H(\, q_{1,j} - q_{0,i} \,) \quad \text{and} \quad U_1 = \sum_{i=1}^{n_0} \sum_{j=1}^{n_1} H(\, q_{0,i} - q_{1,j} \,). \tag{3.6}$$

Here, $H(\cdot)$ is the heaviside step function and notice that $U_0 + U_1 = n_0 n_1$. For large samples, each U (U_0 or U_1) is approximately normally distributed. In that case, the standardized value

$$z = \frac{U_0 - m_U}{\sigma_U}, \tag{3.7}$$

where m_U and σ_U are the mean and standard deviation of U given by

$$m_U = \frac{n_0 n_1}{2} \quad \text{and} \quad \sigma_U = \sqrt{\frac{n_0 n_1 (n_0 + n_1 + 1)}{12}}. \tag{3.8}$$

Significance of test can be checked in tables of the normal distribution. Although, such an hypothesis test is interesting by itself, we are more interested in the concordance probability $P(Q_1 > Q_0)$ which is defined by

$$P(Q_1 > Q_0) = \frac{U_1}{n_0 n_1}. \tag{3.9}$$

It is important to note that even though the Mann-Whitney U test is dependent on whether the shapes of the probability distributions of Q_0 and Q_1 are similar, this requirement is not necessary for the concordance probability. To be more explicit, the problem when Q_0 is for instance one sided exponentially distributed and Q_1 is normally distributed, is not a violation of preliminary assumptions.

The concordance probability or AUC are therefore a measure of the ranking ability of the output function. The Mann Whitney U-test allows one to test whether the result that is obtained is not just be chance in the way the data set is drawn. It can also be used for designing experiments in terms of reasoning about how large data sets are needed for making an effective test.

Moreover, equation (3.9) shows a simple way of calculating $P(Q_1 > Q_0)$ and AUC.

[TODO: multi class AUC.]

3.2 Performance measures for classification on streaming data

This section includes various evaluation methods for classification on data streams. Central to choices of evaluation methods are stationarity versus concept-drift, class imbalance, whether a scoring function exists, computational constraints and also the importance of early detection when a data stream changes behaviour.

3.2.1 Stationary data streams

Let $\{X_t\}$ be a stochastic process, where $p_X(\boldsymbol{x}_{t_1+\tau}, ..., \boldsymbol{x}_{t_k+\tau})$ is the probability distribution function of the joint distribution at $t_1+\tau, ..., t_k+\tau$. Then $\{X_t\}$ is a stationary process, or a stationary stream, if, for all k, for all τ and for all $t_1 + \tau, ..., t_k + \tau$,

$$p_X(\boldsymbol{x}_{t_1+\tau}, ..., \boldsymbol{x}_{t_k+\tau}) = p_X(\boldsymbol{x}_{t_1}, ..., \boldsymbol{x}_{t_k}), \tag{3.10}$$

where the x_ts are vectors of fixed size. Clearly, $p_X(\cdot)$ is not a function of time.

In stationary streams, the mean and the variance, if they exist, do not vary. The covariance structures to neighboring instances are also preserved. To summarize, stationary streams are more general than i.i.d. streams, where the covariance to all other instances are zero.

In [3], the prequential empirical risk is defined as

$$R_{emp}(h, \{X_{t_k}\}) = k^{-1} \sum_{i=1}^{k} L(\boldsymbol{x}_{t_i}, h(\boldsymbol{x}_{t_i}), y_{t_i}), \tag{3.11}$$

where y_{t_i} is the class label at time t_i and n is the number of samples up to t_k. In the original paper, the notation is different and the measure is called the prequental error Adaptations are made to comply with notation and definitions in section 3.1. Notice, that the loss function in the prequential empirical risk, is generalized compared to [3], to also include dependency of x_{t_i}, which may in fact be just t_i. This generalization means that it is possible to model evolving loss as well.

AUC is also popular in a streaming context, in the case when the classification is based on a scoring function, which is common for Bayesian networks. Specialized learning algorithms for imbalanced streams are proposed in [9–11], where it is also pointed out that this problem is particularly difficult and effective algorithms for evaluation is vital. AUC is calculated on limited holdout sets are used in [9,11], while AUC is calculated on the entire stream in [10]. If the whole stream is used, it may have computational problems related to memory and cpu time.

The prequential AUC at time step t_k can be calculated by

$$\text{AUC}_{t_k} = \frac{U_{1,k}}{n_{0,k} n_{1,k}}, \tag{3.12}$$

where $n_{0,k}$ and $n_{1,k}$ are the number of negatives and positives up to t_k. To be precise, $n_{0,k}+n_{1,k} = k$. The quantity $U_{1,k}$ is calculated by the second U-statistic as of equation (3.6) (using $n_{0,k}$ and $n_{1,k}$).

On a data stream there is an opportunity of calculating AUC recursively with the following algorithm.

$$U_{1,k} = \begin{cases} U_{1,k-1} + \sum_{j=1}^{n_{1,k}} H(\, q_{0,n_{0,k}} - q_{1,j}\,) & \text{for} \quad y_{t_k} = 0 \\ U_{1,k-1} + \sum_{i=1}^{n_{0,k}} H(\, q_{0,i} - q_{1,n_{1,k}}\,) & \text{for} \quad y_{t_k} = 1. \end{cases} \tag{3.13}$$

Clearly AUC calculation is $O(k)$, which might be a problem for very long streams.

Another method for calculating AUC involves sorting the scores of all instances and iteration over this list. It is essential that the sorted list always keeps the mapping from each score to either a positive or a negative. At each time step, the list of scores, the number of positives and the number of negatives are updated. The AUC is calculated by algorithm 1. This method is also $O(k)$.

Data: s: Current score, *sortedList*: sorted list of all scores in a stream, p: number of positives in sorted list, n: number of negatives in sorted list
Result: Prequential AUC at the current time step
sortedList.update(s);
p.update(s);
n.update(s);
$c \leftarrow 0, AUC \leftarrow 0$;
forall the *scores* $t \in$ *sortedList* **do**
  if *isPositive(t)* **then**
    $c \leftarrow c + 1$
  end
  else
    $AUC = AUC + c$;
  end
end
$AUC = AUC/(p \cdot n)$;

Algorithm 1: Calculation of prequential AUC at each time step in a stream. Here, *sortedList.update*(s) takes in the current score s and places it correctly in the sorted list. The method *isPositive*(s) computes whether the score s is positive or not, while *p.update*(s) and *n.update*(s) updates the number of positives p and negatives n.

It is worth noting a few points about the statistical interpretability of AUC on stationary streams. Unless the stream is also i.i.d. the $q_{0,i}$s and the $q_{1,i}$s are generally dependent on neighboring points. This fact is however not contradicting the fact that AUC is equal to the concordance probability $P(Q_1 > Q_0)$. This is because the definitions of Q_0 and Q_1 are $Q_0 = q(X_0)$ and $Q_1 = q(X_1)$ and that X_0 and X_1 are random variables with probability distributions $P(X|Y = 0)$ and $P(X|Y = 1)$. The key point is that $P(X|Y = 0)$ and $P(X|Y = 1)$ are not dependent on time in stationary streams. This is not true on non stationary streams.

3.2.2 Data streams that involve concept-drift

Concept drift is generally understood as unforeseen changes in statistical properties of the target variable. In the case of regression this can basically be understood as non stationary of the dependent variable. In the case of classification, it is understood as the class distribution is *not* constant in time.

However, this definition is not always clear cut in the literature. For instance in [12], concept drift is understood broader, where non stationarity of explanatory variables are present in addition to time dependent class distributions. For clarity, in this paper we refer to concept drift in the sense of [12].

In [3], two estimators are suggested for dealing with concept drift and/or non stationary explanatory variables. The first estimator involves calculating the prequential empirical risk over a sliding window of w samples.

$$R_{emp}(t_k, \{h_{t_k}\}, \{X_{t_k}\}, w) = w^{-1} \sum_{i=k-w+1}^{k} L(\boldsymbol{x}_{t_i}, h_{t_i}(\boldsymbol{x}_{t_i}), y_{t_i}). \tag{3.14}$$

Notice that $R_{emp}(t_k, \{h_{t_k}\}, \{X_{t_k}\}, w)$ is a function of time and not just a single number. Hence, the measure reflects the expected loss at $t_{k-\text{floor}(w/2)}$. Notice also that the classification rule itself h_{t_i} is allowed to be a function of time.

The second estimator involves fading factors and is defined as follows

$$R_{emp}(t_k, \{h_{t_k}\}, \{X_{t_k}\}, \alpha) = \frac{\sum_{i=1}^{k} \alpha^{i-k} L(x_{t_i}, h_{t_i}(x_{t_i}), y_{t_i})}{\sum_{i=1}^{k} \alpha^{i-k}} \quad \text{where} \quad 0 \leq \alpha \leq 1. \tag{3.15}$$

When α is equal to one, this reduces to the prequental empirical risk with no forgetting factor, while in the case when α is equal to zero, the estimator forgets all effects before t_k.

In the special case when the loss function is zero-one and h, or h_k, is consistent, then it is shown in [3] that $R_{emp}(t_k, \{h_{t_k}\}, \{X_{t_k}\}, \alpha)$, $R_{emp}(t_k, \{h_{t_k}\}, \{X_{t_k}\}, w)$ and $R_{emp}(h, \{X_{t_k}\})$ approximates the Bayes error in the limiting cases.

As said in 3.2.1, $P(X|Y = 0)$ and $P(X|Y = 1)$ are not dependent on time in stationary streams. However, in our definition of concept-drift, which includes non stationary explanatory variables, $P(X|Y = 0)$ and $P(X|Y = 1)$ are generally dependent on time. This means that an AUC that is calculated by either equation (3.13) or algorithm 1 at the current time step, may be severely biased by very old measurements.

To overcome this problem, it is suggested in [12], a prequantial AUC with a forgetting factor. The method involves calculating AUC on a sliding window of fixed size in a similar manner as

algorithm 1. This algorithm is explained in algorithm 2.

Data: s: Current score, *sortedList*: sorted list of scores in the sliding window, p: number of positives in sorted list, n: number of negatives in sorted list
Result: Prequential AUC at the current time step
sortedList.update(s);
p.update(s);
n.update(s);
$c \leftarrow 0, AUC \leftarrow 0$;
forall the *scores* $t \in$ *sortedList* **do**
  if *isPositive(t)* **then**
    $c \leftarrow c + 1$
  end
  else
    $AUC = AUC + c$;
  end
end
$AUC = AUC/(p \cdot n)$;

Algorithm 2: Calculation of prequential AUC at each time step in a stream. Here, *sortedList.update*(s) takes in the current score s and throws out the oldest score before resorting the list. The method *isPositive*(s) computes whether the score s is positive or not, while *p.update*(s) and *n.update*(s) updates the number of positives p and negatives n.

3.2.3 Risk related to early and late warnings

When working on data streams it may sometimes be of interest to give as early warnings as possible. Another way of saying this is to penalize late warnings more and more. We therefore suggest this measure, where the notation is the same as in equation (3.14)

$$R_{emp}(t_k, \{h_{t_{k+m}}\}, \{X_{t_k}\}, w) = w^{-1} \sum_{i=k-w+1}^{k} L(\boldsymbol{x}_{t_i}, h_{t_{i+m}}(\boldsymbol{x}_{t_i}), y_{t_{i+m}}). \tag{3.16}$$

Here, m denotes some time steps into the future and $h_{t_{k+m}}$ denotes prediction at t_{k+m}. **[TODO: Sigve says: Reflect more and relate to Cajamar. Helge has promised to help out.]**

4 Cajamar: Test and evaluation

This section introduces the testing procedures for the Cajamar use cases. It builds on the general principles for evaluation already described in the previous sections of this report, and exemplifies how these principles can be employed in the setting of the credit evaluation application scenarios.

4.1 Use-case requirements

The test and evaluation procedures for Cajamar will be developed along the lines introduced in Deliverable 2.1 (D2.1): Instead of testing each use case separately, we utilize the notion of *application scenarios*. An application scenario is defined by a sequence of use cases that combined

constitutes a full interaction procedure leading to a verifiable result. In D2.1 we defined two scenarios:

CAJ1: Prediction probability of default: The application scenario covers the first five use cases defined in Deliverable 1.2 (D1.2):

- UC1: Data reading and attribute construction
- UC2: Feature selection
- UC3: Model construction
- UC4: Model application
- UC5: Result checking and risk update

CAJ2: Low risk profile extraction: This application area uses the same model that is developed in the first scenario, but then progresses to use the model differently. It covers the following use cases:

- UC1: Data reading and attributes construction
- UC2: Feature selection
- UC3: Model construction
- UC6: Profile extraction

Requirements for the different use-cases were defined in D1.2. Most requirements are functional in nature, but some also introduce hard requirements that can be tested quantitatively. The latter are repeated in Table 4.1 for completeness. We note that the requirements center around three issues:

1. The whole process covered by Application scenario CAJ1, starting with SQL queries and ending with report generation must take less than 3 hours for the 5.6M clients (requirements CAJ.U1.O1, CAJ.U2.O1, CAJ.U4.D1, CAJ.U4.O1, and CAJ.U5.O1).
2. The prediction quality for Application scenario CAJ1, evaluated by AUROC, must be higher than 90% (CAJ.U3.D3).
3. The quality of the profiles generated in Application scenario CAJ2 is required to improve the benefit of at least 5% (CAJ.U6.O3).

4.2 Model and data characteristics

4.2.1 The data generation process

Both application scenarios use the same dataset, containing the defaulting behaviour of the Cajamar clients[2]. We now briefly describe the data generation process (the description is adapted from D2.1, where a more comprehensive description can be found). Please refer to Figure 4.1 for the timeline.

[2]The second application scenario will use only a subset of the total number of features.

ID	Sub-phase	Description	Task(s)
CAJ.U1.O1	Interface	SQL queries should be efficient enough so that the whole process takes less than 3 hours.	8.2
CAJ.U2.O1	Interface	The feature selection should be efficient enough so that the whole process takes less than 3 hours.	4.3
CAJ.U3.D3	Testing	AUROC should be higher than 90%.	8.3
CAJ.U4.D1	Develop.	Model application should be efficient enough so that the whole process takes less than 3 hours.	2.3, 3.3, 4.1, 4.4
CAJ.U4.O1	Testing	Model should be able to evaluate daily about 5.6M clients.	2.3, 3.3, 4.1, 4.2
CAJ.U5.O1	Interface	The risk data update process should be efficient so that the whole process takes less than 3 hours.	8.2
CAJ.U6.O3	Testing	Expected benefits of a marketing campaign using obtained profiles should be 5% higher than with current methods.	8.3

Table 4.1: Testable requirements for the Cajamar use-cases.

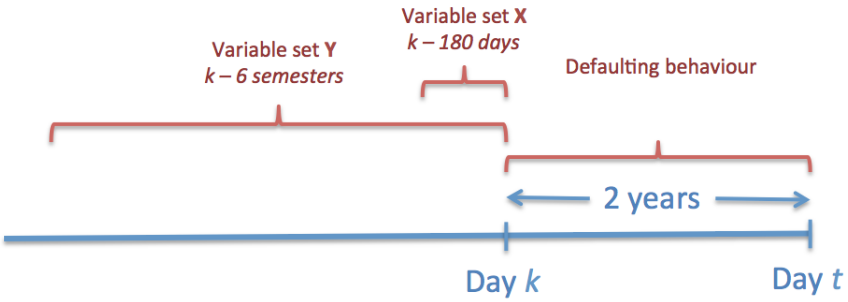

Figure 4.1: Time-line showing the generation of the data set. t refers to the present time and k corresponds to time $t - 2$ years. There are two disjoint groups of variables, denoted as $\mathbf{X}$ and $\mathbf{Y}$, with different past information considered, 180 days back (daily) and 6 semesters back (aggregated by semester), respectively.

The dataset is created at time k, and contains a record for every client to be evaluated. Predictive variables refer here to the financial activity and payment behaviour of the customers in recent past as well as to their socio-demographic information, which usually does not change over time.

Attributes denoted by $\mathbf{X}$ refer to the financial activity during the last 180 days. Examples of these features include "account balance" and "number of credit card operations". They usually change daily for a customer and are encoded by introducing a set of variables for each attribute – one for each day back from time k. Hence, the financial activity of a customer is specified by a number of variables equal to 180 times the number of attributes.

For others attributes, denoted by $\mathbf{Y}$, we are interested in information from the last 36 months. Examples of variables in this set include payments inside Cajamar (loans, mortgages, credits, etc.). The information from the last 36 months is grouped by semester, giving 6 summary variables per attribute that is considered. Finally, there are some static variables (mainly encoding socio-demographic aspects) denoted by $\mathbf{Z}$. These are not included in Figure 4.1 as they are not time-indexed.

The objective of the data analysis is to detect if a customer with profile $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ will default

within the next two years. This corresponds to the class label in the dataset, *Defaulter*, and is determined by inspecting the user's behaviour from time k and 2 years into the future, i.e., in the period from k to t (see Figure 4.1). We obtain this information directly from Cajamar's databases simply by selecting the time k to be two years back in time, and thereby letting t be the current time. Note that, at the present time (time t), we have information of the *Defaulter* variable in the period of time from k to t. Thus, Defaulter$^{(k)}$ indicates if at some point in this period the customer was a defaulter.

The format of the data set for training/updating the model is depicted in Table 4.2. Each record contains the values for all predictive variables and a class variable. The class variable is labelled as *non-defaulter* only when there is no defaulting in the period from k to t (2 years).

	Days			Semester				
Time t	$\mathbf{X}^{(k-180)}$	$\dots$	$\mathbf{X}^{(k-1)}$	$\mathbf{Y}^{(k-6)}$	$\dots$	$\mathbf{Y}^{(k-1)}$	$\mathbf{Z}$	Defaulter$^{(k)}$
Client$_1$								
$\vdots$								
Client$_n$								

Table 4.2: Three groups of attributes $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ are distinguished according to the past information required. Current time is denoted as t. The data set is built at time t with $k = t - 2$ years.

4.2.2 The generated dataset

The existing data set was generated at t equal to December 31$^{\text{st}}$ 2013, thus simulating the calculations as if the AMIDST system was run two years before that (k corresponds to December 31$^{\text{st}}$ 2011). The dataset includes all customers who have been a client of Cajamar in the period k and up to day t, corresponding to $n = 4.5$M customers.

Every member of the dataset has been classified as either non defaulter or defaulter (no missing entries). Customers can have missing attribute values in their description. In particular, some of the clients were not clients in the whole three year period before day k. If a customer was not associated to Cajamar at some point in time $k - 6$ semesters, there will be missing values for some of the variables in $\mathbf{Y}^{(k-6)}$. Formally, every member i of the dataset has a vector of explanatory variables denoted by $\mathbf{W}_i = \{\mathbf{X}_i^{(k-180)}, \dots, \mathbf{X}_i^{(k-1)}, \mathbf{Y}_i^{(k-6)}, \dots, \mathbf{Y}_i^{(k-1)}, \mathbf{Z}_i\}$, potentially with some missing values (encoded using special codes). In total, each customer can be described using 7036 variables, that is, $180\,|\mathbf{X}| + 6\,|\mathbf{Y}| + |\mathbf{Z}| = 7036$, where $|\cdot|$ is the number of variables in each group.

Of all the customers in the dataset, 76% are considered to be of no risk because they do not have a loan in the bank, approximately 20% of the customers were exposed but did not default, and 3.79% of the customers defaulted in the two-year period of interest.

This data set will be used both for training and testing purposes. The test-set is defined by randomly selecting 20% of the customers. For reproducibility, a documented procedure including a fixed random seed is used to select the customers in the test set.

4.3 Predictive performance: test and evaluation

4.3.1 Application scenario 1

The goal of the first application scenario is to determine if a customer is going to be a defaulter after two years. This corresponds to a classification problem, where the class variable is denoted as Defaulter^k in Table 4.2.

The approach of the AMIDST project is to calculate $r_i = P(\text{Default}_i^k \mid \mathbf{w}_i)$ for each customer i. These quantities update the risk table in the system (see Table 4.3). If, at some point, the probability of default of a customer rises above a predefined threshold, the bank may take preventive actions to reduce the risk of defaulting by this customer.

Time t	Risk of being defaulter
Client_1	r_1
$\vdots$	$\vdots$
Client_n	r_n

Table 4.3: Risk for the bank customers where r_i represents the probability of being defaulter for customer i.

According to requirement CAJ.U3.D3, the risk prediction quality should be assessed using the area under the receiver operating characteristic (ROC) curve. The classification rule is that a customer i is classified as a defaulter if $r_i \geq C$ for some constant C, and the ROC curve is computed by plotting the rate of true positives against the rate of true negatives for various choices of C. The requested area can then be found directly, and should, according to the requirement, be larger than 0.90 (see also Section 3.1.2).

There are two main issues with the outlined approach:

Changes in the economic climate: A Cajamar customer's chance of defaulting is to some extent determined by external factors, like the economic climate in Spain. During the economic crisis, the rate of defaulters was significantly higher than what was observed prior to the onset of the crisis. The data set we work with, corresponds to the period of the crisis, and it is natural to expect that the relationships found by the model are optimized for that economic climate. The AUC criteria is chosen to remedy global effects that affect all customers in a similar way, but we can still not guarantee that the model will perform at a similar level for a fundamentally different economic climate. In some cases these differences in the economic climate might be overcome by some of the socio-economic variables, for instance, a civil servant's personal economy with a permanent job should be less influenced by the economic climate than a casual/seasonal worker.

Stable versus volatile climate: Customers in the training and test sets are per definition from the same time period. Learning from the training data, we will therefore be able to detect the economic climate to which the customers will be exposed (e.g., simply by detecting the fraction of defaulters in the training data). If put in production, the predictions the AMIDST model is asked to make will be about the future, i.e., shifted two years in time when compared to the training data. This is not a problem if the economic climate is static or only slowly varying, but will be disastrous if, for instance, AMIDST is asked to make predictions about future customers immediately before the onset of a new economic crisis.

To partly account for these shortcomings of the test procedure we have collected another dataset with k equal to December 31st 2013, and where the correct class labels will be discovered two years later (December 31st 2015). As of today, the class labels for this dataset are unknown, but will be supplied at the end of 2015, and will therefore be available for the formal testing in Task 8.3. An AUC of more than 90% when using this new dataset for testing (and the original dataset for training) would be seen as a strong indication of the applicability of AMIDST in production.

4.3.2 Application scenario 2

Currently, a marketing campaign in Cajamar involves two steps:

- The first step is conducted by the marketing department, and results in a list of candidate customers. The list contains clients that have a high probability of signing what is offered to them (for instance a credit card).
- The second step, conducted by the risk modellers, is to filter out clients that are risky in terms of defaulting.

The task described in Use case 4 is to find relevant users profiles. The profiles can, for instance, be used to conduct marketing campaigns. The profiles should contain customers that are likely to be non-defaulters, and cover only attributes that are found to be relevant by the domain experts. It is required (CA.U6.O3) that the expected benefits of a marketing campaign using the obtained profiles should be 5% higher than with current methods.

Direct quantitative evaluation of the AMIDST-generated profiles is difficult to perform in a formal way mainly for two reasons. The first reason is that the application scenarios generate a *user profile* and not a *set of users*. We cannot value a profile in itself; it is the application of the profile to generate user sets that can potentially be monetized. The second reason is that the AMIDST profile defines users that are not likely to default, not users that are likely to sign a contract (and therefore not necessarily users who are valuable as marketing objects). For instance, it seems natural to expect the AMIDST profile to prefer solvent customers living in their own homes without any mortgage and with a sizeable cash-account. On the other hand, a customer like that may not be relevant to target for a campaign selling small-sized cash-loans without security requirements. This leads to a two-tier evaluation of the profiles, first considering the profiles as generators of profitable marketing campaigns, next as a way to evaluate the ability to find customers that will not default.

1) EVALUATION OF A CAMPAIGN'S PROFIT: Cajamar currently uses theoretical measurements for evaluating marketing campaigns. Let $s_i = P(\text{Signs}_i \mid \mathbf{w}_i)$ be the probability that a customer i signs on an offer presented to him (calculated by an existing system used by Cajamar) and, as before, let $r_i = P(\text{Default}_i^k \mid \mathbf{w}_i)$ be the probability that customer i defaults on a loan within the next two years (given that the offer is accepted). Furthermore, let γ_i be the net present value for the bank of that offer given that the customer does not default, and Γ_i the cost of the offer if a customer ends up in defaulting. c is a fixed indirect cost of the campaign. Then, the loss of a customer would be

$$L(\mathbf{w}_i, s_i, r_i) = \begin{cases} s_i r_i \Gamma_i + (1 - s_i)(1 - r_i)\gamma_i + c & \text{if a loan is offered;} \\ s_i(1 - r_i)\gamma_i & \text{otherwise.} \end{cases} \tag{4.1}$$

We therefore propose that the profile extraction is evaluated as follows:

1. A marketing campaign is selected, and the set of customers contacted are listed. The set of customers is called $\mathcal{C}$.
2. The AMIDST system is used to generate a profile for non-defaulting customers, and a fixed number of customers fitting the profile (comparable to the number of elements in $\mathcal{C}$) are selected. The marketing department selects a subset of the customers in this set based on their probability to contract. Call this reduced set of customers $\mathcal{A}$. Note that the set $\mathcal{A}$ now defines a fictitious campaign (chosen by AMIDST) and has not been employed in a real campaign.
3. The two sets $\mathcal{C}$ and $\mathcal{A}$ are compared qualitatively and quantitatively (using the theoretical measure in Equation (4.1)).

In Equation (4.1) all customers that are not included in the campaign will contribute with the loss $s_i(1 - r_i)\gamma_i$. In practice, contributions will only be collected from the set of customers that is selected by either method, i.e., the set of customers in $\mathcal{C} \cup \mathcal{A}$ because a customer selected by neither system (i.e., not in $\mathcal{C} \cup \mathcal{A}$) will contribute equally to the loss of each method, and is therefore not helpful to establish the difference between them.

2) EVALUATION OF DEFAULTING BEHAVIOUR: A drawback of this approach is that the loss-function rests upon (theoretical) probabilities $P(\text{Default}_i^k \mid \mathbf{w}_i)$ and $P(\text{Signs}_i \mid \mathbf{w}_i)$. Due to the introduction of $P(\text{Signs}_i \mid \mathbf{w}_i)$ in the loss function, we cannot in general guarantee that the system that is best at predicting the defaulting behavior will obtain the lowest loss. To target the quantitative requirement without using the theoretical probabilities we therefore also propose to utilize the AMIDST risk prediction capability from application Scenario 1 directly in the marketing setting, where the following procedure will be performed:

- Select a historical marketing campaign that is at least two years old, and remove all customers that did not sign. The remaining set of customers is called $\mathcal{C}$.
- Filter out clients that the AMIDST system deem too risky. The set of customers is called $\mathcal{A}$. Note that $\mathcal{A} \subseteq \mathcal{C}$.
- Calculate the empirical loss of the set $\mathcal{A}$ compared to that of $\mathcal{C}$. The requirement is that the loss of $\mathcal{A}$ should be at least 5% lower than the loss of the set $\mathcal{C}$. Note that for a direct comparison, only the difference in the two sets, $\mathcal{C} \setminus \mathcal{A}$, will contribute, and the costs of applying the AMIDST solution will be γ_i for the customers that do not default and $-\Gamma_i$ for those that do.

It should be noted that this procedure only evaluates the AMIDST system's ability to remove poor customers from the list of customers that were included in the original campaign, as we are unable to quantify the effect of AMIDST potentially wanting to send marketing material to customers that were not selected for the historical campaign without using the theoretical construct of Equation (4.1). The two evaluation approaches must therefore be seen in combination to give the full picture.

4.4 Run-time performance: test and evaluation

4.4.1 Application scenario 1

According to the requirement procedure, the full process starting with SQL statements and ending with validation of the new risks should be completed in no more than three hours (CAJ.U1.O1,

CAJ.U2.O1, CAJ.U4.D1, CAJ.U5.O1).

The AMIDST solution will be installed in a server (IBM System x3690 X5) with the following characteristics:

- 2 Intel Xeon 10C Processors (Model E7-2870 130w 2.40GHz/30MB)
- 256GBRAM,16x16GB(1x16GB,4Rx4,1.35V)PC3L-8500CL7ECCDDR3 1066MHz LP RDIMM
- 2 internal disks SAS IBM 146 GB 2.5in SFF Slim-HS 15K 6Gbps SAS HDD RAID, one of them hot swap. 10 Disks IBM 600GB 2.5in SFF 10K 6Gbps HS SAS HDD.

This server is mainly used by credit risk models and marketing models departments. Data will be obtained via SQL queries. The current Information Center database management system is Oracle, but is being changed to a Teradata solution.

Learning the AMIDST model, however, requires an iterative process that is performed until convergence, and whose number of steps, and hence time, might vary due to random initialization. One way to enforce that the 3 hour upper bound is not violated, is to specify a *timeout* counter for the learning algorithm. Ideally, there should be enough time for the learning algorithm to reach convergence, and the timeout counter should be only included as a safe mechanism.

In order to provide a reliable estimation of the average time employed by the learning process and its expected performance, the following two graphs will be plotted:

- A first graph whose x-axis shows the number of tests and y-axis the total time employed by each of the experiments. Apart from the mean, a confidence interval at e.g. a 99% confidence level, will be provided to guarantee that the expected time will be included among this interval with a margin of error of 1%.
- A second graph in which the x-axis corresponds to the number of iterations/time and y-axis to the performance of the learning algorithm (indicating how close it is to convergence, e.g. lower bound in variational Bayes).

Joint interpretation of the two graphs will give us knowledge about the expected running time and performance of the process. We want to provide a mechanism to ensure that with a level of confidence of 99%, the learning algorithm will converge at the desired time limit. In the worst case, the algorithm will provide a valid outcome that might not be optimal. Note that the same analysis should also be performed for any other stochastic algorithms utilized in the process of the application scenario.

4.4.2 Application scenario 2

There are no specific run-time requirements for this application scenario. Specifically, it is stated that *"[...] execution time of this process is not relevant because the marketing campaigns are not launched so frequently."*.

5 Daimler: test and evaluation

6 Verdande Technology: Test and evaluation

6.1 Use-case requirements

In Deliverable 2.1 the following application scenarios for Verdande Technology (VT):

VER1: Detection of drill string vibrations covering these use cases:

- UC1: Parsing data sets
- UC2: Erratic torque detection (model construction)
- UC5: Model application
- UC6: Result generation

VER2: Semi-automatic labelling covering these use cases:

- UC1: Parsing data sets
- UC3: Semi-automatic labelling (model construction)
- UC5: Model application
- UC6: Result generation

VER3: Automatic formation detection covering these use cases:

- UC1: Parsing data sets
- UC4: Formation detection (model construction)
- UC5: Model application
- UC6: Result generation

Requirements for the different use-cases were defined in Deliverable 1.2. Those that can be evaluated quantitatively are repeated in Table 6.1 for completeness. We summarize the requirements for each application scenario as follows:

VER1: The evaluation process for this application scenario was not quantified, but postponed to this document, as requirement VER.U2.D2 states that *"[. . .]Formalisation of the test procedure will be developed in task 1.2"*.

VER2: The result of the system is to be compared to the prior rates and the results of tagging performed by an inexperienced driller (requirements VER.U3.D2, VER.U3.D3, VER.U3.D4 and VER.U2.D5).

VER3: Quality is estimated as distance from the estimated formation shifts to the ground truth, and improvement over the a priori formation chart is required (VER.U4.D2 and VER.U4.D3).

ID	Sub-phase	Description	Task(s)
VER.U1.O1	Develop.	Data import should take less than 10 minutes on a single desktop computer or equivalent	7.1
VER.U3.D2	Testing	The rate of missed abnormalities must be no larger than the prior probability of abnormality.	7.2
VER.U3.D3	Testing	The rate of missed normal states must be no larger than the prior probability of normality.	7.2
VER.U3.D4	Testing	The rate of missed abnormalities could be below what is obtained by an inexperienced drilling engineer or a software program at that level.	7.2
VER.U3.D5	Testing	The rate of missed normalities could be below what is obtained by an inexperienced drilling engineer or a software program at that level.	7.2
VER.U4.D2	Testing	MD_algorithm must be less than MD_prior (defined in Use Case Scenario 3 in Delivery 1.2), on at least 1 randomly chosen drilling log.	7.3
VER.U4.D3	Testing	MD_algorithm should be 10 percent less than MD_prior (defined in Use Case Scenario 3 in Delivery 1.2) on at least 3 randomly chosen drilling logs.	7.3
VER.U5.D4	Testing	Agents need to run fast enough for real-time implementation. In practice, this means that once probability tables are calculated (post learning), the agent once provided a single new data point must be able to calculate probabilities, values or classifications and returning these in under 1 second on a single desktop computer or equivalent hardware	3.3, 7.3

Table 6.1: Testable requirements for VT's use-cases.

6.2 Model and data characteristics

In order to make this document self contained, a summary of VT's three application scenarios are given below:

Detection of drill string vibrations: This task aims to better diagnose the shape of the well-bore and the state of the equipment. Currently, VT's solution is a simple algorithm based on comparing root mean squared differences in a time window between the measured torque and a smoothed version of the torque, thereby measuring the local amplitude variations. This approach is rather limited and does not inherently take the existing uncertainty in the streaming data into account. The main objective for scenario VER1 is to design a probabilistic graphical model for erratic torque monitoring and detection of abnormal situations.

Semi-automatic labelling: VT's solution uses supervised learning for detecting undesired events, and therefore requires *labelled* datasets where undesired events or episodes are identified. VT already have extensive datasets labelled in this way, but manually labelling new data sequences, or adapting the existing labelling to new definitions, can be a very time-consuming activity. VT is therefore interested in looking into techniques for semi-automatic labelling. Given unlabelled data streams collected over time from typical drilling conditions, this application scenario aims to compute a normality score for each drilling situation, then label it as either *normal* or *abnormal*. As for the previous task, a probabilistic graphical model will be designed, taking into account the temporal dynamics of the drilling process and continuously adapting to changes in the incoming streaming data.

Automatic formation detection: This scenario aims to predict in real time the formation tops from the MWD (measurements while drilling) data using a probabilistic graphical model. Once again, this should be performed taking the temporal dynamics of the drilling process into account. The automatic formation detection is vital for dealing with several issues such as hole instability and vibrations, and also important for reducing the costs and the overall non-productive time.

6.2.1 The data generation process

The datasets available from VT are very complex, and contain a high number of variables (a number that varies between the different logs). Many of the variables will not be used by the application scenarios developed in AMIDST, and feature selection both based on expert knowledge and using data-driven methods will be required. Another complicating factor is that mnemonics and units are generally not consistent between logs. In order to simplify these issues, all the drilling logs are preprocessed before given to the AMIDST software, resulting in two separate datasets: The first data-set is used by application scenarios VER1 and VER2, while the second is used by application scenario VER3.

6.2.2 Dataset one

This data set consists of 100 drilling logs of various sizes; typically one log covers one drilling section (that is a few weeks of data). The data is cleaned and contains a fixed number of variables defined using common mnemonics and SI units. Each log is represented in two XML-files, one that is *holistic*, the other *drilling-focused*. The first file contains a stream of data, with new observations coming with an update frequency between 0.1Hz and 1Hz. In addition to the actual drilling data, the time-indexed XML-file is amended with three calculated fields:

Torque vibration index: This index indicates whether or not there was abnormal torque at the time the data was captured. The index is not available in every log. This variable is central for the application scenario VER1.

Normality index: This index is related to the application scenario VER2. It is generated automatically by VT's DrillEdge as follows: A point in time is defined as normal if there are no cases on the case radar, no events have fired in a 10 minutes time-window (looking to the past *and* to the future), and no events have fired in a depth window of 5 meters (upwards and downwards).

Depth-log indicator: This indicator points out which time stamps are used for drilling. As such it is used to link to the elements in the second XML file.

The second XML file is made by simply filtering out the drilling periods from the first XML-file (using the the depth log indicator). This data-set is *depth-indexed*, as the drilling activity is defined through the increasing depth of the wellbore.[3]

[3]Some care must be taken when depth is adjusted manually. This is also performed during preprocessing of the data, and is not part of the AMIDST solution.

6.2.3 Dataset two

The second dataset is related to application scenario VER3. The data originates from OMV Aktiengesellschaf, but VT has unlimited access to the data also for use in the AMIDST project. This data is an extension of the logs in the previous dataset, as it contains down hole measurements such as gamma ray and resistivity. Moreover, lithology charts from the planning phase (i.e., before drilling) and after the well is drilled are also included.

6.3 Predictive performance: test and evaluation

6.3.1 Application scenario 1

The goal of the first application scenario is to detect abnormal torque states. The output of the AMIDST system is a probability of abnormal torque at each time step, and during evaluation these numbers will be compared to the torque vibration index from the logs that are tagged with this information. We note that even if domain experts have supplied the labelling of the data (that is, they have marked the parts of the time-series that are abnormal), the classifications are not well defined: Two different experts may disagree both whether or not a time-period actually has abnormal torque, and – if they agree that the torque has been abnormal – they may still disagree regarding the start and end points of the erratic period.

Bursts of abnormal torque typically lasts for a period of from two to ten minutes, but we note that detecting torque vibrations *early* is generally not important. The driller feels the vibrations himself, and is need not be told what to him is obvious by a software system. The value in automatically detecting the vibrations is rather to be able to diagnose how much cumulative damage the equipment has been subject to, and thereby predict wear and potential equipment failure. The evaluation can therefore be seen as inspecting a series of classifications (one at each time-point), where each can be seen in isolation. In other words, it is not important to evaluate the streaming fashion of the classification for this application scenario. We have therefore decided to use the AUC as the evaluation metric, only taking whether or not the tag is detected at each point in time into account.

There are two reasons why it is not possible to determine a quantitative requirement for the AUC before further examining the data: *i*) the effect of the before-mentioned ambiguities in the expert-provided tagging has not yet been quantified, and *ii*) it is unknown how well-defined the start and end points of each period of abnormal torque are in the data. We will therefore proceed in the following manner:

1. Logs containing the torque-information will be separated into training and test-set using a documented procedure (including a fixed random seed).
2. 10 logs that have already been tagged by a domain expert will be re-tagged by a different expert. The quality of the domain expert's tags will be calculated using AUC.
3. All logs in the test-set will be tagged by AMIDST, and the AUC for each log is calculated.
4. The poorest results from AMIDST (in terms of AUC) should be be at least as good as the poorest result from the expert. Similarly, the mean AUC result obtained by AMIDST must be at least as good as the mean result obtained by the domain expert.

.

6.3.2 Application scenario 2

The goal of application scenario VER2 is to detect "abnormal states". The output from AMIDST is a normality-index at each time step, which is to be combined into intervals of time where the system is seen as "abnormal". The length of the time intervals with an abnormal state will typically be in the range of hours. We note that the tags used for learning are clearly not well-defined (see Section 6.2.1); they are automatically generated using VT's existing software system and not evaluated by a domain expert. The supplied tags are nevertheless taken to be the ground truth during the evaluation of this application scenario, both because having a domain expert verify the tagging on all 100 logs would be very time consuming and because this application-scenario, as argued in Deliverable 7.1, is designed to be purely data-driven.

The requirements for testing this scenario (Requirement D2, VER.U3.D3, VER.U3.D4, and VER.U3.D5) all point towards calculating false negatives and false positives, and the results of the AMIDST system should improve both random labelling and the output of an inexperienced drilling engineer. Note that it is the *detection* of an event that is highlighted, and the AMIDST system is not given extra credit for being able to locate an abnormal event very precisely in time. In order to meet these requirements it is important to define what false positives and false negatives are. A positive tag (existence of a tag) is determined by looking through the log and compare the calculated probability for that tag to a pre-set threshold. If the probability is larger than the threshold, the event is flagged. It is a true positive if that time-point is inside a labelling of the same event in the gold-standard labelling, and a false positive otherwise. Similarly, events that are not detected are false negatives.

We note that the above measures are global and can be estimated after the logs have been processed. The performance measures do not take concept drift into account, even though this is clearly present in the data. A drilling operation will almost always start out as normal, but as the open hole gets longer, more unstable and more dirty, the probability of abnormality increases. Furthermore, the geology as such varies with the depth of the wellbore, and often gets more challenging (e.g., due to higher pressures and temperatures). Finally, equipment, which may have been well maintained at the start of the operation, will degrade during drilling. It is therefore also of interest to compute the prequential AUC with a forgetting factor (see Equation (**??**)) and plot this versus time. A proper discussion of this is believed to be of interest, and should be supplied for at least five logs.

6.3.3 Application scenario 3

In this application scenario it is of interest to both find the formation tops and correctly label them. The input to the learning is the prior knowledge in form of a lithology chart from the planning of the well as well as the drilling-related data captured during operation (see the description of Dataset 2 in Section 6.2.3).

From Delivery 1.2 we have defined two important quantities: MD_prior is the mean deviation between the expert identified formation tops before drilling and after drilling, and MD_algorithm is the mean deviation between the formation tops that are detected by the algorithm and the formation tops that are identified by the experts after the well is drilled. Now, it is required that MD_algorithm must be less than MD_prior on randomly chosen log (Requirement VER.U4.D2), and also that MD_algorithm should be 10 percent less than MD_prior when evaluated on on at least 3 randomly chosen logs (Requirement VER.U4.D3).

If we can assume that all formation shifts that are present in the true lithology is also found in lithology chart from the planning phase (albeit, maybe not always indicated at the correct depths), evaluation is straight forward. However, if this assumption is not met, a more complex evaluation function than outlined in the requirement must be employed to penalize formation changes that are erroneously added to or removed from the estimated lithology.

[TODO: Sigve says: In general, I do not have the answer for this. Helge suggested looking at sequence alignment on wiki. I tried, but did not figure it out. Will think more. A simple option would be to assume that sequences from planning are the same as the ground truth, by some manipulation of the data.]

6.4 Runtime performance: test and evaluation

The requirements related to runtime performance are

VER.U1.O1: Data import should take less than 10 minutes on a single desktop computer or equivalent.

VER.U5.D4: Agents need to run fast enough for real-time implementation. In practice, this means that once probability tables are calculated (after learning), the agent once provided a single new data point must be able to calculate probabilities, values or classifications and returning these in under 1 second on a single desktop computer or equivalent hardware.

The requirements are considered to be fairly straightforward to accomplish, but will nevertheless be rigorously tested. The requirements are considered fulfilled if the test is passed at least 99% of the time.

7 Conclusion

References

[1] Gama, J.a., Rodrigues, P.P., Sebastião, R.: Evaluating algorithms that learn from data streams. In: Proceedings of the 2009 ACM Symposium on Applied Computing. SAC '09, New York, NY, USA, ACM (2009) 1496–1500

[2] Gama, J., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (2009) 329–337

[3] Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Machine Learning **90**(3) (2013) 317–346

[4] Vapnik, V.: The Nature of Statistical Learning Theory. Information Science and Statistics. Springer Verlag (2000)

[5] Wu, S., Flach, P., Ramirez, C.: An improved model selection heuristic for auc. In: ECML PKDD. Volume 4701 of Lecture Notes in Computer Science. (2007) 478–489

[6] Gama, J.: Knowledge Discovery from Data Streams. Chapman and Hall (2010)

[7] Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. Annals of Mathematical Statistics **18**(1) (1947) 50–60

[8] Fay, M.P., Proschan, M.A.: Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules. Statistics Surveys **4** (2010) 1–39

[9] Ditzler, G., Polikar, R.: Incremental learning of concept drift from streaming imbalanced data. IEEE Transactions of Knowledge and Data Engineering **25**(10) (2013) 2283–2301

[10] Hoens, T., Chawla, N.: Learning in non-stationary environments with class imbalance. In: 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (2012) 168–176

[11] Lichtenwalter, R., Chawla, N.: Adaptive methods for classification in arbitrarily imbalanced and drifting data streams. In: New Frontiers in Applied Data Mining. Volume 5669 of Lecture Notes in Computer Science. (2010) 53–75

[12] Brzezinski, D., Stefanowski, J.: Prequential auc for classifier evaluation and drift detection in evolving data streams. In: Proceedings of the 3rd International Workshop on New Frontiers in Mining Complex Patterns, Nancy, France, September 19, 2014. (2014)