Project no.: **619209**

Project full title: **Analysis of Massive Data STreams**

Project Acronym: **AMIDST**

Deliverable no.: **D1.3**

Title of the deliverable: **AMIDST handbook, Initial version**

Contractual Date of Delivery to the CEC:	**31.12.2014**
Actual Date of Delivery to the CEC:	**19.12.2014**
Organisation name of lead contractor for this deliverable:	**NTNU**
Author(s):	**Sigve Hovda, Helge Langseth, Anders L. Madsen, Thomas D. Nielsen**
Participants(s):	**P01, P03, P04**
Work package contributing to the deliverable:	**WP1**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**35**
Start date of project:	**1st January 2014 Duration: 36 month**

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

This document describes the formal basis for evaluating classifiers working on data streams. This basis is utilized to define test and evaluation protocols for the three use case providers, focusing on the testable requirements that were uncovered in Deliverable 1.2: "Requirements for the automotive, oil, and financial data domains". Although guided by these requirements, the scope of the test and evaluation basis extends beyond the three specific use cases considered in AMIDST. The evaluation basis thus also supports setting up test protocols applicable to classifiers in other types of domains.

Keyword list: Evaluation procedures, data streams, use-case requirements

Contents

Document history

Version	Date	Author (Unit)	Description
v0.3	16/12-2014	Sigve Hovda, Helge Langseth, Thomas D. Nielsen	The test and evaluation framework discussed and established
v0.6	17/12-2014	Sigve Hovda, Helge Langseth, Anders L. Madsen, Andrés Masegosa, Ramon Saez Martinez, Thomas D. Nielsen, and Antonio Salmerón, Frode Sørmo, Galia Weidl	Initial draft discussed with industrial partners and reviewed by the PSRG
v1.0	19/12-2014	Helge Langseth, Anders L. Madsen, Thomas D. Nielsen, and Antonio Salmerón	Final version of document

1 Executive summary

The objective of this document is to define the test and evaluation procedures for the different application scenarios described for the three use-case providers: Cajamar, Verdande Technology AS, and Daimler AG. The report constitutes the initial version of the AMIDST Handbook, but considers only initiatives from Work package 1. The Handbook will be continued into a final version (Deliverable 9.6) where other relevant aspects will be examined.

In order to establish a common frame of reference for the procedures, we provide a background section discussing the most commonly used evaluation metrics for both static and streaming classifiers. These techniques are then employed in the subsequent sections of the document, where evaluation procedures for each use-partner are established. The procedures are established based on the requirements that were defined during the requirement engineering phase of the project as documented in AMIDST Deliverable 1.2.

2 Introduction

The number of algorithms designed for learning from streaming data is increasing, but there is still not a single generally accepted framework for evaluating them. One reason for this is that evaluating algorithms that are designed to work on streaming data is in general far more complicated than evaluating their static counterparts, both when computational cost and statistical properties are considered. In this document we will discuss some of the most commonly used evaluation criteria for streaming data, and show how they are used by the three use-case providers in the AMIDST project.

Evaluation of predictive algorithms on streaming data is a quickly evolving field of computer science, and we will only consider the evaluation of *classification* algorithms in this document. For ease of exposition we will focus on binary classification problems, i.e., problems where the predictive variable takes on one of two predefined labels; this is also sufficient in relation to all application scenarios considered by the use-case providers. Nevertheless, we also comment briefly on how the evaluation methods can be generalized to multi-class problems.

Relevant methodologies for evaluation of both static and streaming algorithms are identified and discussed in Section 3. This lays the foundation for the next three sections, where the evaluation routines for the three use case providers are given together with summaries of the requirements and data descriptions.

3 Test and evaluation methodology

As most of the important evaluation metrics applicable for streaming data are generalizations of standard evaluation techniques for static (or "non-streaming") data, we will start our discussion by some relevant performance measures for classification of static data.

3.1 Performance measures for classification of static data

Consider a dataset $\mathcal{D}$ of fixed size N, where each instance is drawn from a joint probability distribution $P(\boldsymbol{X}, Y)$. Here $\boldsymbol{X}$ and Y are random variables; $\boldsymbol{X}$ is the vector of explanatory variables (also called features) and Y is the class label. $\boldsymbol{X}$ and Y 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 the finite set of class labels; note that $|\Omega_Y| = 2$ for a binary classification problem.

3.1.1 Basic metrics

In terms of evaluating the performance of h, we will assume that the observations in $\mathcal{D}$ are independently drawn from $P(\boldsymbol{X}, Y)$. Such a dataset is said to be identically and independently distributed (i.i.d.) because each of the N samples are independent realizations from the same (unknown) distribution $P(\boldsymbol{X}, Y)$. The result of such an experiment with classifier h can be shown in a confusion matrix. An example of a confusion matrix is shown in Table 3.1, where the classifier should distinguish two different oil-well drilling activities, namely drilling and tripping. A global measure of the quality of the algorithm is the classification accuracy, which is calculated as the sum of the diagonal elements divided by the sum of all elements in the table (in this case 21 out of 24, i.e., 0.875).

		Predicted class	
		drilling	tripping
Actual class	drilling	4	3
	tripping	0	17

Table 3.1: Example of a confusion matrix. The accuracy is $21/24 = 0.875$.

It is important to note that the accuracy is not telling the whole story about the classifier. For instance, by looking at the table it is seen that when the activity is drilling, the classifier is unsure and misses on three out of seven examples. On the other hand, all the 17 examples of the tripping-activity are correctly classified. These properties cannot be realized by looking at the accuracy alone. Moreover, the interpretation of a given accuracy value is heavily dependent on how evenly the classes are distributed. For instance, in anomaly detection, which is usually a highly imbalanced classification problem where the number of normal examples will outnumber the abnormal ones significantly, a simple majority classification rule (i.e., one that classifies all new instances as normal) will obtain an impressive accuracy even if the classifier does not show any ability to detect the abnormal elements. We therefore conclude that accuracy alone is not sufficient for evaluating a classifier.

In binary classification it is common to define one of the two classes as the "real" target, and denote examples of that class as positive and instances that do not belong to that class as negative. If we redefine our classifier to be a "system for recognition of the drilling-activity", the information in Table 3.1 can be equivalently expressed as in Table 3.2.

		Prediction	
		positive	negative
Actual class	positive	4	3
	negative	0	17

Table 3.2: Example of a confusion table for recognition of the drilling activity. The true positives and true negatives are on the diagonal, while the two other numbers are the false positives and the false negatives.

From the confusion table it is easy to calculate a variety of numbers that describe the ability of the classification rule: True positive rate, also known as recall, is the number of true positives divided by the total number actual positives (4/7 in the example). The false positive rate, also known as fall-out, is the number of false positives divided by the total number actual negatives (0/17 in Table 3.2). In the same way, one defines true and false negative rates (17/17 and 0/17 in Table 3.2, respectively). Precision is defined as the number of true positives divided by the number of assumed positive (4 out of 4 in this example), and finally, the F1-score is defined as the harmonic mean of the precision and recall, i.e., $2/(\text{precision}^{-1} + \text{recall}^{-1})$, resulting in $F_1 = 8/11$ in Table 3.2. Using a combination of these numbers, we can discuss the abilities of classification algorithms in a meaningful way even when the dataset $\mathcal{D}$ is imbalanced.

3.1.2 Evaluation of families of classification rules

Probabilistic classifiers operate by calculating the probability distribution $P(Y|\boldsymbol{x})$, and then comparing the probability for each class to a certain threshold. The potential classification rules

are in a family of hypothesis functions $\mathcal{H}$, where each element $h_\gamma : \Omega_X \to \Omega_Y$ is defined as

$$h_\gamma(\boldsymbol{x}) = \begin{cases} \mathsf{positive} & \text{if} \quad P(Y = \mathsf{positive}|\boldsymbol{x}) \geq \gamma \\ \mathsf{negative} & \text{otherwise.} \end{cases}$$

It is of interest to evaluate all these classification rules. The receiver operating characteristic (ROC) curve is a plot of the true positive rate as a function of the false positive rate, as the threshold γ varies over $[0, 1]$. An example is seen in Figure 3.1, where the true positive rate (y-axis) is shown as a function of false positive rate (x-axis) for two probabilistic classifiers. The dashed line shows the ROC curve of a classifier that uses random guessing; the solid line gives a typical ROC curve for a more competent classifier.

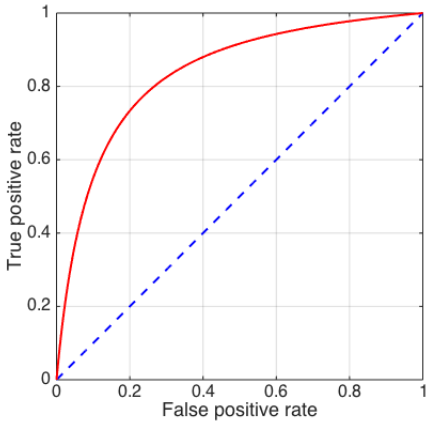

Figure 3.1: Receiver operating characteristics curve.

The ROC curve allows a visual exposition of the family of classifiers $\mathcal{H}$. In particular, it is easy to see which threshold is needed for a certain rate of true positives. This plot is clearly independent of whether or not the class distribution is balanced. Moreover, the information in the ROC curve is commonly reduced to a single number, namely the area under the curve (AUC). Note that AUC is independent of γ. It is a popular method for evaluating classification problems because it is simple but at the same time meaningful also for imbalanced class distributions. As pointed out by for instance [1], AUC is preferable to accuracy for model evaluation.

AUC has the statistical interpretation that it is the probability that a random member of the positive class is found to have a higher probability of being positive than a random member of the negative class, and is therefore a measure of the *ranking ability* of the output function. We will now discuss the underlying statistical properties of AUC. First, let $\boldsymbol{X}_p$ and $\boldsymbol{X}_n$ be random variables with the probability distributions $P(\boldsymbol{X}_p) = P(\boldsymbol{X}|Y = \mathsf{positive})$ and $P(\boldsymbol{X}_n) = P(\boldsymbol{X}|Y = \mathsf{negative})$, respectively. Furthermore, we define the random variables $Q_p = P(Y = \mathsf{positive}|\boldsymbol{X}_p)$ and $Q_n = P(Y = \mathsf{positive}|\boldsymbol{X}_n)$. Q_p and Q_n are random variables because they are the output of the classification procedure evaluated at the *random* variables $\boldsymbol{X}_p$ and $\boldsymbol{X}_n$; Q_p is the probability calculated when a random variable is generated from the distribution for the positive class; similarly Q_n gives the calculated probability for a variable following the distribution of the negative class to be a member of the positive class. The probability $P(Q_p > Q_n)$, which obviously is independent of γ, is called the *concordance probability*. Without going into details,

it is possible to show that

$$\text{AUC} = P(Q_p > Q_n). \tag{3.1}$$

It is interesting to note that Equation (3.1) holds without making additional assumptions about the probability distributions of Q_p and Q_n. Furthermore, it is worth noting that the concordance probability is exactly equal to the so-called *common language effect size* (see, e.g., [2]) of the Mann-Whitney U test [3]. This relationship gives us an efficient estimator for the AUC, and we therefore discuss the Mann-Whitney U test next.

The Mann-Whitney U test [3] is a nonparametric version of the well-known t-test, testing the null hypothesis that two populations are the same against the alternative that one population tends to have larger values than the other. In our context we can use the Mann-Whitney U test to discuss whether or not values of Q_p are consistently larger than values of Q_n, i.e., H_0 is that Q_p and Q_n are identical; H_1 that they have different means but similar shape. Some additional assumptions must be made, notably that all samples of Q_p and Q_n are independent of each other (this is fulfilled as we consider i.i.d. data); a comprehensive discussion can be found in [4].

As before, we consider the data set $\mathcal{D}$ with N observations $\{\boldsymbol{x}_i, y_i\}$, independently drawn from $P(\boldsymbol{X}, Y)$. We separate this dataset into two parts, one containing only the examples from the positive class, the other from the negative class. The sizes of the two subsets are N_p and $N_n = N - N_p$, respectively. For notational simplicity we define the two indexing functions $\text{pos}(\cdot)$ and $\text{neg}(\cdot)$. $\text{pos}(i)$ returns the index of the i'th positive element ($i = 1, \ldots, N_p$) and $\text{neg}(j)$ does similarly for the negative elements ($j = 1, \ldots, N_n$). Next, define $\mathbf{1}\{\cdot\}$ as the the indicator function, so that $\mathbf{1}\{\epsilon\} = 1$ if ϵ is true and 0 otherwise. Now, the U statistic is defined by

$$U = \sum_{i=1}^{N_p} \sum_{j=1}^{N_n} \mathbf{1}\left\{P\left(Y = \text{positive}|\boldsymbol{x}_{\text{pos}(i)}\right) \geq P\left(Y = \text{positive}|\boldsymbol{x}_{\text{neg}(j)}\right)\right\}. \tag{3.2}$$

U is approximately normally distributed with mean $N_p N_n/2$ and variance $N_p N_n(N_p + N_n + 1)/12$ for large N_p and N_u under H_0, which enables the hypothesis test to be conducted. Furthermore, the Mann-Whitney estimator for the concordance probability, and therefore of the AUC itself, is

$$\widehat{\text{AUC}} = \hat{P}(Q_p > Q_n) = \frac{U}{N_p N_n}. \tag{3.3}$$

We note that even though the Mann-Whitney U-test assumes that the shapes of the probability distributions of Q_p and Q_n are similar, this is not necessary for the interpretation of the AUC as a concordance probability, or the estimation of AUC using Equation (3.3).

3.1.3 Empirical loss

The evaluation metrics considered so far do not take into account that some misclassifications might be more costly than others. In a real application this may be crucial, as it might for instance be worse to neglect treating a person who has an infectious disease than it is to erroneously medicate a healthy person. Moreover, the cost of misclassification of an object may depend on the object itself. If the classifier is predicting whether a client in a bank will default on a loan or not, the cost of misclassification is clearly related to the size of the loan in question. We will now consider a procedure to include such costs in a performance measure.

We define a *loss function* as a real and lower-bounded function L on $\Omega_X \times \Omega_Y \times \Omega_Y$. When used for evaluating classification procedures, the value of the loss function at an arbitrary point

$(\boldsymbol{x}, h(\boldsymbol{x}), y)$ is interpreted as the loss of classifying $\boldsymbol{x}$ as being from the class $h(\boldsymbol{x})$ when the correct classification is y. Notice that the loss function is defined to depend also on $\boldsymbol{x}$ so that misclassification costs may depend on the instance being misclassified. From the frequentist perspective, the expected loss is often referred to as the *risk function* [5], and is obtained by calculating the expected loss with respect to $P(\boldsymbol{X}, Y)$:

$$R(h) = \int_{y \in \Omega_y} \int_{\boldsymbol{x} \in \Omega_x} L(\boldsymbol{x}, h(\boldsymbol{x}), y) P(\boldsymbol{x}, y)\, d\boldsymbol{x}\, dy.$$

If the losses are independent of $\boldsymbol{x}$ and the cost of correct classification is zero, the risk function reduces to the well known expected cost of misclassification (ECM) given by

$$\begin{aligned} \text{ECM} = & L(\mathsf{positive}, \mathsf{negative}) \cdot P(\mathsf{positive}|\mathsf{negative}) \cdot p_{\mathsf{negative}} + \\ & L(\mathsf{negative}, \mathsf{positive}) \cdot P(\mathsf{negative}|\mathsf{positive}) \cdot p_{\mathsf{positive}}. \end{aligned} \tag{3.4}$$

Here, $L(\hat{y}, y)$ is the cost of classifying an item from class y as being from class $\hat{y}$ and $P(\hat{y}|y)$ is the probability that the classifier will wrongly classify an object belonging to y as a member of $\hat{y}$. p_{positive} and p_{negative} are the fraction of objects belonging to the positive and negative class, respectively.

$R(h)$ cannot be computed in general because the distribution $P(\boldsymbol{X}, Y)$ is unknown. However, we can compute an approximation known as the *empirical loss* (also known as *empirical error*) by averaging the loss function over the data set $\mathcal{D}$. The empirical loss is given by

$$\hat{R}(h, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} L(\boldsymbol{x}_i, h(\boldsymbol{x}_i), y_i). \tag{3.5}$$

We note that some supervised learning algorithms search directly for a hypothesis h to minimize Equation (3.5) w.r.t. a given labelled dataset. In such applications the data is used for *training* the classifier. In this document, on the other hand, we focus only on *evaluating* classifiers, and the dataset $\mathcal{D}$ is thus a test-set.

3.1.4 Multi class classification

The techniques we have discussed for binary classification generalizes easily to multi-class classification (i.e., classification where Ω_y contains more than two elements). As a first example, consider again the drilling activity classification problem that was analyzed in the confusion matrix of Table 3.1.

		Predicted class		
		drilling	tripping-in	tripping-out
Actual class	drilling	4	3	0
	tripping-in	0	6	2
	tripping-out	0	4	5

Table 3.3: Example of a confusion matrix for a three-class problem. The accuracy is now $15/24 = 0.625$.

Assume now that we want to distinguish between *three* possible actions, after having learned that the tripping activity is better separated into the two sub-classes tripping-in and tripping-out.

An example of what the confusion matrix may look like now is shown in Table 3.3, and similar analysis as was done using Table 3.1 can obviously also be done with Table 3.3.

When it comes to AUC, it is often extended in one of two ways:

- Let $\text{AUC}(i, j)$ be the AUC between class i and j, where samples that do not belong to either i or j are ignored. Then Hand and Till [6] defined the multi-class AUC as the average over all pairwise AUC-values,

$$\text{AUC}_p = \frac{1}{m(m-1)} \sum_i \sum_{j \neq i} \text{AUC}(i, j),$$

  where m is the number of classes. We call this the *pairwise* AUC.

- The *one vs. all* generalization of AUC is equally straight forward. Define $\text{AUC}(i, \neg i)$ as the AUC between class i and all other classes, i.e., samples that do not belong to class i are pooled together in the $\neg i$ class. The multi-class AUC is now

$$\text{AUC}_m = \frac{1}{m} \sum_i \text{AUC}(i, \neg i).$$

Finally, we note that the general definition of the use of loss functions do not require the classification problem to be binary. Thus, $\hat{R}(h, \mathcal{D})$ of Equation (3.5) is immediately useful, even if the simplification in Equation (3.4) is not.

3.2 Performance measures for classification on streaming data

This section includes various evaluation methods for classification of data streams. The data stream is in general observed at time-points $t_1, t_2, \ldots$ where $t_i < t_{i+1}$ for all i. For notational simplicity we shall assume that $t_i = i$ in the following (that is, the sequence is observed at time-points $t = 1, 2, \ldots$). We will use the notation $\boldsymbol{X}_t$ to denote the vector of random variables $\boldsymbol{X}$ at time t, and $\boldsymbol{x}_{t_1:t_2}$ to denote the collection $\{\boldsymbol{x}_{t_1}, \boldsymbol{x}_{t_1+1}, \ldots, \boldsymbol{x}_{t_2}\}$. Evaluation is done w.r.t. a dataset $\mathcal{D}_T = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$.

If the instances in the data stream are i.i.d. (defined to also imply that the autocorrelation in the stream is zero), the only challenge when evaluating a classifier on a stream compared to working on static data is that the data stream is open ended, i.e., continuously grows in size. Evaluation methods related to these streams are closely related to the evaluation methods that are known for static i.i.d. data, and as i.i.d. streams are very rare in applications they will not be described in further detail.

3.2.1 Stationary data streams

A generalization of the i.i.d. data stream is the *stationary* data stream, which contains data that is generated from a stationary process. Let $\{\boldsymbol{X}_t\}$ be a stochastic process, where $p_X(\boldsymbol{x}_{t:t+k})$ is the joint probability distribution for the observations at times $(t, \ldots, t+k)$. Then $\{\boldsymbol{X}_t\}$ is a stationary process if

$$p_X(\boldsymbol{x}_{t:t+k}) = p_X(\boldsymbol{x}_{t+\tau:t+\tau+k})$$

for all t, k and τ. Note that $p_X(\cdot)$ is not a function of time. The mean and variance are (if they exist) constant in time for stationary processes.

Various performance measures on stationary data streams have been proposed in the literature, including those involving loss functions [7–9]. The predictive sequential (or *prequential*) loss is defined as the average loss endured up to and including the current time step T. Using the notation from the previous subsection, the prequential empirical loss is calculated as

$$\hat{R}_T\left(h, \mathcal{D}_T\right) = \frac{1}{T}\sum_{t=1}^{T} L(\boldsymbol{x}_t, h(\boldsymbol{x}_{1:t}), y_t), \tag{3.6}$$

where y_t is the class label at time t. Note that the loss function for the prequential empirical loss includes the object description $\boldsymbol{x}_t$ (similarly to the static case), and that the classifier uses $\boldsymbol{x}_{1:t}$ instead of only $\boldsymbol{x}_t$ to make its classification at time t.

AUC is popular for evaluating probabilistic classifiers also in a streaming context. This is especially the case for imbalanced streams, where classification problems are particularly difficult and effective algorithms for evaluation are vital [10–12]. Different strategies for calculating AUC on data streams exist, one may for instance calculate the AUC from a limited holdout set [10,11] or on the entire stream [12]. Using the whole stream may raise computational problems both related to run-time complexity and memory requirements.

The prequential AUC at time step T can be calculated by

$$\mathrm{AUC}_T = \frac{U_T}{N_p^{(T)} N_n^{(T)}},$$

where $N_p^{(T)}$ and $N_n^{(T)} = T - N_p^{(T)}$ are the number of positive and negative elements up to and including time $t = T$. U_T is defined as was the U-statistic in Equation (3.2), but can also be calculated recursively:

$$U_T = \begin{cases} U_{T-1} + \sum_{j=1}^{N_n^{(T)}} \mathbf{1}\left\{P\left(Y = \mathsf{positive}|\boldsymbol{x}_T\right) \geq P\left(Y = \mathsf{positive}|\boldsymbol{x}_{\mathrm{neg}(j)}\right)\right\} & \text{if} \quad y_T = \mathsf{positive} \\ U_{T-1} + \sum_{i=1}^{N_p^{(T)}} \mathbf{1}\left\{P\left(Y = \mathsf{positive}|\boldsymbol{x}_{\mathrm{pos}(i)}\right) \geq P\left(Y = \mathsf{positive}|\boldsymbol{x}_T\right)\right\} & \text{otherwise.} \end{cases}$$

Clearly, the calculation of AUC_T is $O(T)$, which might be problematic for large T.

Another method for calculating AUC after observing the data at $t = T$ involves maintaining a sorted list of objects. Define $\boldsymbol{o}_t$ as $\boldsymbol{o}_t = \{P(Y_t = \mathsf{positive}|\boldsymbol{x}_{1:t}), y_t\}$ for $t = 1, \ldots, T$, and define $\mathcal{S}_{T-1}$ as the list of objects $\boldsymbol{o}_t$ for $t = 1, \ldots, T-1$ sorted in increasing value of $P(Y_t = \mathsf{positive}|\boldsymbol{x}_{1:t})$. We use $\mathcal{S}_T[i]$ to denote the i'th element in $\mathcal{S}_T$, and let $\mathcal{S}_T[i].y$ denote the class label of that element. Furthermore, $N_p^{(t)}$ and $N_n^{(t)}$ are the counts of positive and negative elements in $\mathcal{S}_t$,

respectively. AUC_T is calculated by Algorithm 1. This method is $O(T)$.

Input: $\mathcal{S}_{T-1}$, $N_p^{(T-1)}$, $N_n^{(T-1)}$
Output: Prequential AUC_T at time T, $\mathcal{S}_T$, $N_p^{(T)}$ and $N_n^{(T)}$.
`// Initialization`
accNegative $\leftarrow 0$;
tmpCount $\leftarrow 0$;
`// Update variables`
$\mathcal{S}_T \leftarrow$`insertElementInSortedList(`$\mathcal{S}_{T-1}$`,` $\boldsymbol{o}_T$`)`;
$N_n^{(T)} \leftarrow N_n^{(T-1)}$;
$N_p^{(T)} \leftarrow N_p^{(T-1)}$;
if $\boldsymbol{o}_T.y$==*positive* **then** $N_p^{(T)}$++ **else** $N_n^{(T)}$++;
`// Iterate through the elements in the sorted list`
for $t = 1, \ldots, T$ **do**
  if $\mathcal{S}_T(t).y$ == *positive* **then**
    tmpCount += accNegative;
  else
    accNegative++;
  end
end
$\text{AUC}_T \leftarrow \text{tmpCount} / \left(N_p^{(T)} \cdot N_n^{(T)} \right)$;

Algorithm 1: Calculation of prequential AUC at time T.

It is worth noting that the interpretation of AUC as the concordance probability is sound also for stationary streams, but the Gaussian approximation utilized by the Mann Witney U-test (Section 3.1) is only valid if the stream is i.i.d.

3.2.2 Data streams with concept-drift

Most streams are not stationary. *Concept drift* is generally understood as changes in the statistical properties of the data stream over time. Typically, it is assumed that data generation process goes through "phases", where the stream is stationary inside each phase.

Two estimators are suggested for handling concept drift within the framework of prequential loss in [9]. The first estimator calculates the prequential empirical loss over a sliding window of w samples:

$$\hat{R}_T(h, \mathcal{D}_T, w) = \frac{1}{w} \sum_{t=T-w+1}^{T} L(\boldsymbol{x}_t, h(\boldsymbol{x}_{1:t}), y_t). \tag{3.7}$$

Notice that $\hat{R}_T(h, \mathcal{D}, w)$ is time-varying by design, and can under some regularity conditions be interpreted as the expected loss at time $t = T - (w-1)/2$. Setting $w = 1$ makes the prequential empirical loss only consider the observation at $t = T$, while choosing $w = T$ results in the evaluation used for stationary data (Equation (3.6)). Notice also that the classification rule itself can vary over time, even if this is not made explicit in the notation.

The second estimator involves a fading factor α, and is defined as follows:

$$\hat{R}_T(h, \mathcal{D}_T, \alpha) = \frac{\sum_{t=1}^{T} \alpha^{T-t} L(\boldsymbol{x}_t, h(\boldsymbol{x}_{1:t}), y_t)}{\sum_{t=1}^{T} \alpha^{T-t}} \quad \text{where} \quad 0 \leq \alpha \leq 1.$$

This reduces to the standard prequential empirical loss (Equation (3.6)) if $\alpha = 1$, while $\alpha = 0$ forces the estimator to only consider the observation at $t = T$.

In the special case when the loss function is zero-one and h is a consistent estimator for y, it can be shown [9] that $\hat{R}_T(h, \mathcal{D}_T)$, $\hat{R}_T(h, \mathcal{D}_T, w)$ and $\hat{R}_T(h, \mathcal{D}_T, \alpha)$ all approximate the Bayes error in the limiting cases.

AUC_T as calculated by Algorithm 1 is not meaningful under concept drift, because the algorithm fails to realize that the AUC could indeed vary with time. To overcome this problem, a prequential AUC with a forgetting factor was proposed in [13]. The approach simply calculates AUC on a sliding window of fixed size w in a similar manner as Algorithm 1, but now letting $\mathcal{S}_T$, $N_p^{(T)}$ and $N_n^{(T)}$ only cover information in the time series from $t = T - w + 1$ to $t = T$.

3.2.3 Evaluation of the prediction's time horizon

The standard formulation for evaluating a classifier in the streaming context considers the precision of a classifier $h(\boldsymbol{x}_{1:t})$ compared to a target variable measured at that same time, y_t. It may sometimes be relevant to examine the system's ability to predict into the future, i.e., use $h(\boldsymbol{x}_{1:t})$ to say something about y_{t+k} for $k > 0$. This is, e.g., relevant if one wants to determine how far into the future the AMIDST system can recognize the lane-change of a car with a predefined level of precision. The procedure will then be to first train a classifier $h_k(\boldsymbol{x}_{1:t})$ which is optimized for k-step prediction, and thereafter evaluate it. The adaption of the evaluation machinery is straight-forward. Equation (3.6) is for instance generalized to

$$\hat{R}_T(h, \mathcal{D}_T, k) = \frac{1}{T} \sum_{t=1}^{T} L(\boldsymbol{x}_t, h_k(\boldsymbol{x}_{1:t}), y_{t+k}).$$

Now, a plot of $\hat{R}_T(h, \mathcal{D}_T, k)$ agains k will highlight how the prediction ability deteriorates with the prediction's time horizon.

3.2.4 Time-aware loss functions

Most evaluation regimes for data streams consider the quality of the classification at each point in time, and then define the total loss as a (potentially weighted) sum of the instantaneous losses over time. Furthermore, a decision that is optimal at time t_1 will also be optimal at a later time $t_2 > t_1$ no matter what happens in the stream between $t = t_1$ and $t = t_2$.

A more realistic scenario is that the users of the system will express views like "*I can tolerate one false positive now and again, but not twice within one minute*" or "*The value of detecting an abnormal situation 2 minutes before the event is worth 5 points; at the time of the event it is only worth one point*". If properly formalized into a decision-theoretic problem, statements like the two examples above transform the decision problem from a sequence of *independent* decisions into a *sequential* decision problem, where the decision at time t_1 also must take potential *future* data and how to respond to that into consideration. Although highly relevant for many real-world applications, we do currently not plan to include solvers for sequential decision problems in the AMIDST toolbox.

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 evaluating credit applications and for the evaluation of marketing campaigns guided by risk profiles.

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 proceed to use the model for establishing risk profiles. 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 the models in Application scenario CAJ1 must be higher than 90% in terms of AUC (CAJ.U3.D3).
3. The quality of the profiles generated in Application scenario CAJ2 is required to improve the benefit of a marketing campaign by at least 5% (CAJ.U6.O3).

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	AUC 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.

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.[1] 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 time line.

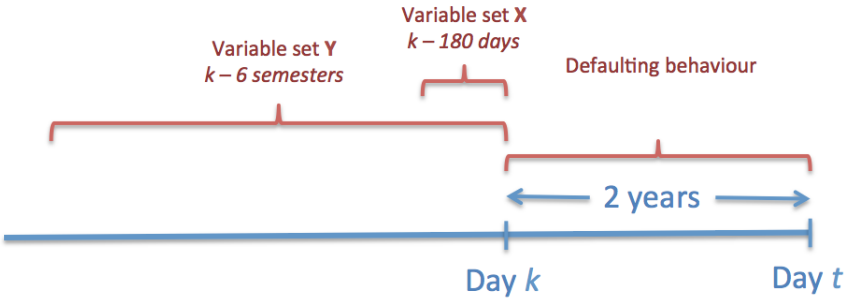

Figure 4.1: Time-line showing the generation of the data set. t refers to the present time and k corresponds to time $t-2$ years. There are two disjoint groups of variables, denoted as $\boldsymbol{X}$ and $\boldsymbol{Y}$, that represent different types of past information; 180 days back (daily) and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

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

The attributes denoted by $\boldsymbol{X}$ refer to the financial activity of a client during the last 180 days. Examples of attributes in the set include "account balance" and "number of credit card oper-

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

ations". These attributes usually change on a daily basis for a customer and are encoded by a corresponding set of variables; one for each day back from time k. Hence, the financial activity of a customer over the entire period is specified by a number of variables equal to 180 times the number of attributes.

For the other attributes, denoted by $\boldsymbol{Y}$, 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 considered. Finally, there are a collection of static variables (mainly encoding socio-demographic aspects) denoted by $\boldsymbol{Z}$. 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 $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ 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 about 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	$\boldsymbol{X}^{(k-180)}$	$\dots$	$\boldsymbol{X}^{(k-1)}$	$\boldsymbol{Y}^{(k-6)}$	$\dots$	$\boldsymbol{Y}^{(k-1)}$	$\boldsymbol{Z}$	Defaulter$^{(k)}$
Client$_1$								
$\vdots$								
Client$_n$								

Table 4.2: The three groups of attributes $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ 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 dataset 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 at Cajamar in the period from day k and up to day t, corresponding to a total of $n = 4.5$M customers.

Every customer in the dataset has been classified as either non defaulter or defaulter (no missing entries). Customers can have missing attribute values in their descriptions though. For example, some clients may not have been clients in the whole three year period before day k. In articular, if a customer was not associated with Cajamar at time $k - 6$ semesters, there will be missing values for some of the variables in $\boldsymbol{Y}^{(k-6)}$. Formally, every member i of the dataset has a vector of explanatory variables denoted by $\boldsymbol{W}_i = \{\boldsymbol{X}_i^{(k-180)}, \dots, \boldsymbol{X}_i^{(k-1)}, \boldsymbol{Y}_i^{(k-6)}, \dots, \boldsymbol{Y}_i^{(k-1)}, \boldsymbol{Z}_i\}$, potentially with some missing values (encoded using special codes). In total, each customer can be

described using 7036 variables, that is, $180\,|\boldsymbol{X}| + 6\,|\boldsymbol{Y}| + |\boldsymbol{Z}| = 7036$, 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 at risk 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 for selecting the customers in the test set.

4.3 Predictive performance: test and evaluation

4.3.1 Application scenario 1: Prediction probability of default

The goal of the first application scenario is to determine if a customer is going to default within two years. This corresponds to a classification problem, for which we denote the class variable by Defaulter^k.

The approach of the AMIDST project is to calculate $r_i = P(\text{Default}_i^k \mid \boldsymbol{w}_i^{(k)})$ for each customer i, where k denotes the date when the prediction is made. These quantities are used to update the risk table in the IC system (see Table 4.3). If, at some point, a customer's probability of defaulting rises above a predefined threshold, the bank may take preventive actions to reduce the risk of defaulting.

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 employed is that a customer i is classified as a defaulter if $r_i \geq c$ for some constant c. The ROC curve can then be computed by plotting the rate of true positives against the rate of false positives for various choices of c. According to the requirement the requested area should 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 has been collected during the crisis, and it is natural to expect that the relationships found by the model are optimized for this particular 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 by definition from the same time period. When 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 may prove problematic 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 final 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 ability of AMIDST in a production setting.

4.3.2 Application scenario 2: Low risk profile extraction

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, but 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 $s_i = P(\text{Signs}_i \mid \boldsymbol{w}_i^{(k)})$ be the probability that customer i (represented by $\boldsymbol{w}_i^{(k)}$) signs on an offer presented to him (calculated by an existing system used by Cajamar) and, as before, let $r_i = P(\text{Default}_i^k \mid \boldsymbol{w}_i^{(k)})$ 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 of that offer for the bank given that the customer does not default, and let Γ_i the cost of the offer if a customer ends up in defaulting after accepting the offer. c is a fixed indirect cost of the campaign. Then, the loss of a customer would be

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

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 the approach above is that the loss-function rests upon (theoretical) probabilities $P(\text{Default}_i^k \mid \boldsymbol{w}_i)$ and $P(\text{Signs}_i \mid \boldsymbol{w}_i)$. Due to the introduction of $P(\text{Signs}_i \mid \boldsymbol{w}_i)$ 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 resulting 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: Prediction probability of default

According to the requirement analysis, the full process starting with SQL queries and ending with the 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 on 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 departments. Data will be obtained via SQL queries. The current Information Center database management system is Oracle, but will be replaced by 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 initializations. 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 only be included as a safety 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 in 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 within 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 application scenario.

4.4.2 Application scenario 2: Low risk profile extraction

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 Verdande Technology: Test and evaluation

5.1 Use-case requirements

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

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 5.1 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 5.1: Testable requirements for VT's use-cases.

5.2 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 has 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.

5.2.1 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 one is used by application scenarios VER1 and VER2, while the second is used by application scenario VER3.

5.2.2 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 units from the *International System of Units* (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.[2]

5.2.3 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.

5.3 Predictive performance: test and evaluation

5.3.1 Application scenario 1: Detection of drill string vibrations

The goal of the first application scenario is to detect abnormal torque states. The output of the AMIDST system is the probability of abnormal torque being present 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 though a domain expert has supplied the labelling of the data (that is, he has marked the parts of the time-series that have abnormal torque), the classifications are not well defined. Two different experts may disagree both on whether or not a time-period actually has abnormal torque, and – if they agree that the torque has been abnormal – they may disagree regarding the start and end points of the erratic period.

Bursts of abnormal torque typically last 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 does not need to be told by a software system. The value in automatically detecting the vibrations is rather to be able to diagnose how much cumulative damage the drill-string has been subject to, and thereby predict equipment wear and potential failure. The evaluation can therefore be seen as inspecting a series of classifications (one at each time-point), where each classification can be evaluated 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.

[2]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.

3. All logs in the test-set will be tagged by AMIDST, and we calculate the AUC for these logs as well.

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.

5.3.2 Application scenario 2: Semi-automatic labelling

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 5.2.1); 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 (Requirements VER.U3.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 a naïve baseline 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 event (existence of an abnormal situation) is determined by looking through the log and compare the calculated probability to a pre-set threshold. If the probability is larger than the threshold, abnormality is flagged. It is a true positive if that time-point is inside a labelling of some 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 Section 3.2.2) and plot this versus time. A proper discussion of the behaviour of this measure is believed to be enlightening, and should be supplied for at least five logs.

5.3.3 Application scenario 3: Automatic formation detection

In this application scenario it is of interest to both find the formation tops and correctly label the formations. 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 5.2.3).

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 a 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 the formation shifts that are present in the true lithology are also found in the correct sequence in a candidate model that is to be evaluated (albeit, maybe not always indicated at the correct depths), the calculation is straight forward. However, if this assumption is not met, a more complex evaluation function than outlined in Deliverable 1.2 must be employed to penalize formation changes that are erroneously added to or removed from the estimated lithology. Our approach will employ a two-step strategy:

1. We first look only at the *sequence* of formations, without considering their depths. We calculate the *edit distance* from the candidate solution to the formation sequence obtained by the expert after drilling. New elements in the candidate sequence are inserted with a length of zero meters.
2. After editing, the sequencing of the formations are identical in the candidate solution and the gold standard. At this point, the mean deviation between the expert identified formation tops identified after drilling and the tops of the solutions is well defined.

The process is exemplified in Figure 5.1. The left part of the figure shows the target lithology (formations named “A”, “B” etc for simplicity) and the candidate solution that is to be evaluated. The first step looks at the correct sequencing (A-B-D-A-C) and compares that to the estimated sequence (A-B-C-B). The edit-distance is 3, because “D” must be inserted, and the two latter letters in the sequence must be re-labelled. The right-hand side of Figure 5.1 shows again the target lithology chart, now together with the amended candidate solution. Note that the “D” formation is given zero depth. The mean deviations are calculated by comparing the depth of each formation top in the target lithology with the corresponding formation top in the candidate model using the mean deviation.

During evaluation, both the AMIDST solution and the prior model (the expert’s lithology-map from the planning phase) will be compared to the expert’s sequence found after drilling. In the addition to the requirements regarding the mean deviations given above (Requirements VER.U4.D2 and VER.U4.D3), it is required that the AMIDST-model should be at least as good as the the prior model in terms of the edit distance on every well.

5.4 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.

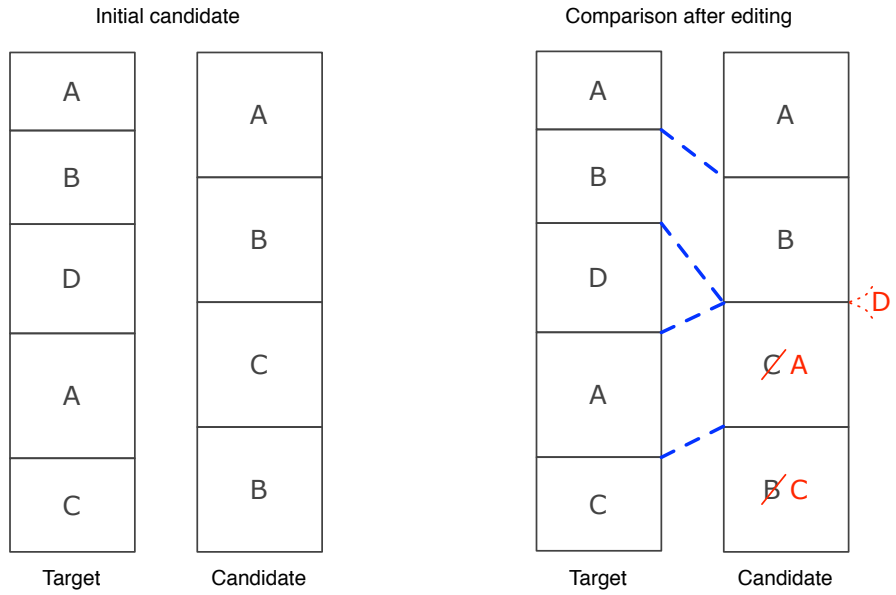

Figure 5.1: Comparing a candidate solution to the target in two steps.

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.

6 Daimler: Test and evaluation

This section introduces the testing procedures for the Daimler use cases. The evaluation is based on the general principles for evaluation already described in Section 3, although some more specific evaluation methods are also employed to deal with the particular requirements of these use cases.

The approach to test and evaluation described in this section corresponds to the method currently employed by Daimler. We use this methodology in order to be able to compare the results produced by the baseline model developed by Daimler before the start of the AMIDST project.

6.1 Use case requirements

In Deliverable 2.1 we defined the following two application scenarios for Daimler:

DAIM1: Early recognition of a lane change manoeuvre

DAIM2: Earlier prediction of the need for a lane change based on relative dynamics [14, 15].

Both application scenarios cover all the use cases defined in Deliverable 1.2, where specific requirements for each of these use cases can also be found. Most requirements are functional in nature

and introduce functionality requests into the system, but some also introduce hard requirements that should be tested quantitatively. The latter are repeated in Table 6.1 for completeness.

ID	Sub-phase	Description	Task(s)
DAI.U7.O2	Testing	Model application should be efficient enough so that the whole maneuver recognition process takes less than 0.1 ms running on Linux platform with I7 CPUs.	6.1, 6.2 (5.2, 5.3)
DAI.U7.O3	Testing	Model application should be efficient enough so that the whole maneuver recognition process is predicting the need for a lane change at least 2 sec. earlier than the actual lane mark crossing. The recognition and prediction accuracy should be measured by the Area under ROC curve (AUC), which must be > 0.96 for 1s prediction and AUC > 0.9 for 2s prediction (due to the safety character of the application).	6.1, 6.2 (5.2, 5.3)
DAI.U7.O4	Testing	Model application should be efficient enough so that the whole maneuver recognition process requires memory space of less than (in the order of) 10^2 kB of RAM and 10^2 kB of ROM for the static model.	6.2 (5.2)

Table 6.1: Testable requirements for the Daimler use-cases.

6.2 Model and data characteristics

Data Characteristics

A rough description of the data sets used in the evaluation can be found below. For confidentiality reasons we cannot give all the details, but a complete specification of the datasets, the setting of the problem, and the proposed models can be found in Deliverables 1.2, 2.1, and 6.1.

- The evaluation dataset is composed of several hundred data sequences; a number which is expected to grow in the future. Each data sequence corresponds to the last seconds of a driving manoeuvre ending with a lane marking crossing.
- Each data sequence consists of a sequence of observations sampled at a fixed rate (in the order of milliseconds), where each observation describes the traffic scene at a given moment in time. These traffic scene descriptions are given by a set of features such as velocity, time to enter the occupancy grid, relative distance to the vehicle in front as seen from either the EGO[3] or an OBJ[3] car, etc. (see D2.1 and D6.1 for further details).
- Each of the manoeuvres have been manually labelled by an expert, or through the use of a heuristic rule, according to three different categories: Lane change to the left (LC-left), Lane change to the right (LC-right), and Lane follow (LF).

[3]As defined in Deliverable 1.2, EGO denotes the vehicle equipped with the prediction system and OBJ is a surrounding vehicle/object.

- Manoeuvres of both the EGO and the OBJ cars are considered in the dataset (see D2.1 and D6.1 for further details). The evaluation dataset contains the following set of manoeuvres:
  - Approximately 20% EGO: LC-left.
  - Approximately 15% EGO: LC-right.
  - Approximately 10% EGO: LF.
  - Approximately 15% OBJ: LC-left.
  - Approximately 20% OBJ: LC-right.
  - Approximately 20% OBJ: LF.

Models

Daimler considers two application scenarios for which dynamic classifiers are proposed. The purpose of this evaluation is to compare the performances of these models to a previously proposed baseline model (a static classifier). Here we just briefly introduce the models (see Deliverable 2.1 for further details):

The Static OOBN model: This is a static object-oriented Bayesian network (OOBN) [16] previously described in [17–19]. It is able to detect a manoeuvre 0.6 seconds before execution. In order to further improve the quality of the on-board adaptive cruise control our goal is to enhance the prediction horizon for manoeuvre recognition using different dynamic extensions.

The Dynamic OOBN model: This model is proposed for the first application scenario. It is a dynamic extension of the above model and involves copies of the static OOBN for a varying number of time-steps.

The Dynamic OOBN model with relative dynamics [14, 15]: This is the model proposed in the second application scenario. It extends the other models by using new data on the relative dynamics between a vehicle and the vehicles in front of it driving on the same lane, thereby trying to improve the prediction horizon.

Other models may be considered as the work in Work package 6 progresses.

6.3 Predictive performance: test and evaluation

Notation

We now describe the evaluation methodology that will be employed for the two application scenarios. This evaluation methodology is formally described assuming a binary classification problem to accommodate the use of the AUC measure (see Section 3.1). To deal with the three-class classification problem we have in this use case (i.e., LC-left, LC-right and LF), we follow the so-called "one vs. all" approach (see Section 3.1.4) and convert the classification problem to three different binary classification problems, to which the evaluation procedure will be applied separately:

- LC-left versus (LC-right or LF);

- LC-right versus (LC-left or LF);
- LF versus (LC-right or LC-left).

We start defining the following notation:

- $\left\{\mathcal{S}^{(i)}\right\}_{i\in\{1,\ldots,M\}}$ denotes the set of M data sequences or driving manoeuvres in the evaluation dataset. We will assume that the provided set of data *sequences* is i.i.d.
- $\left\{\boldsymbol{x}_t^{(i)}\right\}_{t\in\{1,\ldots,T\}}$ denotes the set observations making up data sequence $\mathcal{S}^{(i)}$. To simplify the notation slightly we assume that all data sequences have the same length T. $\boldsymbol{x}_t^{(i)}$ is thus the t-th observation from the i-th sequence, i.e., the description of the traffic scene at time t for the i-th manoeuvre. Similarly to the notation in Section 3.2, we will use $\boldsymbol{x}_{t_1:t_2}^{(i)}$ to denote the sub-sequence $\{\boldsymbol{x}_t^{(i)}\}_{t\in\{t_1,\ldots,t_2\}}$.
- $y^{(i)}$ denotes the class label associated with the i-th data sequence or manoeuvre. As argued above we will treat it as a binary variable, $y^{(i)} \in \{\mathsf{true}, \mathsf{false}\}$.

We use $p_t^{(i)}$ to denote the probability assigned by a classifier to the event that the t-th observation of the i-th data sequence is positive, i.e., the event $\{Y_t^{(i)} = \mathsf{true}\}$. These probabilities are computed differently depending on the type of classifier being used:

- **Static classifiers** provide predictions based on $p_t^{(i)} = P(Y_t^{(i)} = \mathsf{true}|\boldsymbol{x}_t^{(i)})$. The static classifier therefore gives the probability that at time t the manoeuvre happens or is going to happen given the observation at that time.
- **Dynamic classifiers** calculate the conditional probability $p_t^{(i)} = P\left(Y_t^{(i)} = \mathsf{true}|\boldsymbol{x}_{1:t}^{(i)}\right)$. The main difference between a static and a dynamic classifier is that the dynamic one considers all the observations of the sequence up to and including time t to make the prediction. It would also be possible to use dynamic classifiers to make "future" predictions $P\left(Y_{t+k}^{(i)} = \mathsf{true}|\boldsymbol{x}_{1:t}^{(i)}\right)$ for some $k > 0$.

While the classifier will produce a prediction $Y_t^{(i)}$ at each point in time t, the labelling procedure is slightly different. As an example, the beginning of a lane-change sequence always starts with a sequence where the car movements are consistent with a lane follow, and it is difficult to define exactly when the lane change movement actually starts. Consequently, it is also difficult to properly label each time-point in a sequence. The labelling therefore assigns only one label (LC-right, LC-left or LF) to the whole sequence, and we note that this labelling is not subject to interpretation errors. Different approaches will be investigated to improve this labeling process (see the discussion related to Use case 1 in Deliverable 1.2), but for now we follow the approach pursued by [19].

Evaluation Measures

We now detail the evaluation measures used for the comparison of the different classifiers:

ROC-curve and AUC for the last k seconds history: For each manoeuvre we take all the predictions made by the classifier during the last k seconds of the manoeuvre [15], i.e., $p^{(i)}_{T-k:T}$. Each prediction is associated with the class label of that manoeuvre giving the sequence $\left\{\left(p^{(i)}_{T-k}, y^{(i)}\right), \ldots, \left(p^{(i)}_{T}, y^{(i)}\right)\right\}$. By collecting all probability - label pairs across all manoeuvres $i = 1, \ldots, M$ we compute the AUC following the procedure detailed in Section 3.1.

AUC-curve in time: The aim of this evaluation measure is to investigate how the performance of the classifiers improves as their predictions are made closer and closer to the end of the manoeuvre, where it is assumed that predictions are easier. We plot the number of seconds remaining before the end of the manoeuvre on the x-axis and the AUC measure of the classifier at this moment in time on the y-axis [15]. Figure 6.1 illustrates the procedure using the static OOBN model w.r.t. the maneuvers LF, LC-Left and LC-right. Details about the procedure are given in Deliverable 6.1.

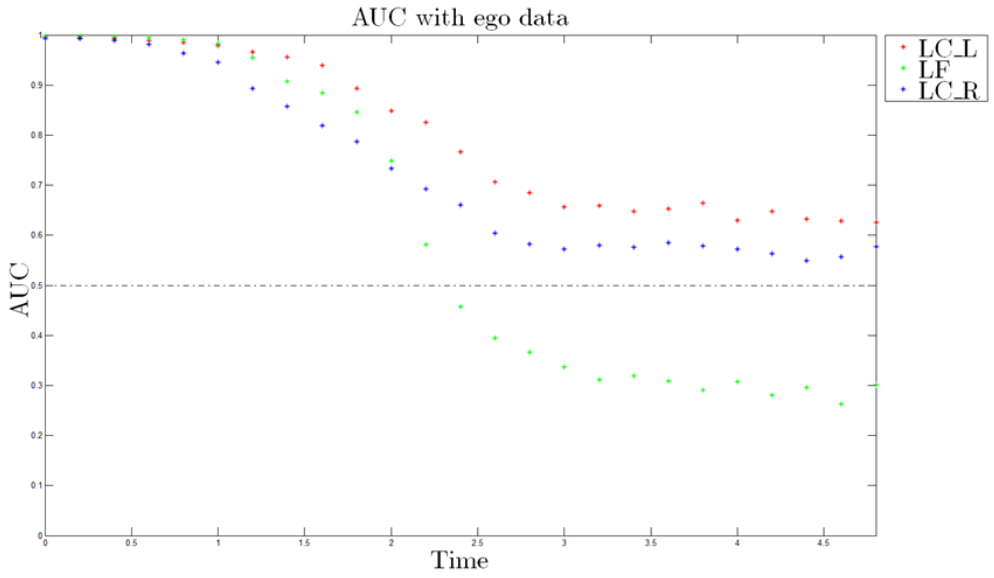

Figure 6.1: An illiustration of an AUC time plot generated by the static baseline classifier.

Static Analysis: The aim of this analysis is to evaluate the performance of the classifiers when making "hard decisions", e.g., when predicting if a manoeuvre is Lane follow or not [15,19]. For this purpose we characterize the predictions of a classifier as either true positives, false positives, true negatives, and false negatives, see Section 3.1. Each classifier provides a sequence of probability predictions, $p^{(i)}_{1:T}$. We say that the sequence is consistently classified as positive if at some moment in time $t = m$, before the end of the manoeuvre, the probability values are consistently over a given threshold τ, i.e., $\min_{t=m}^{T} p_t^{(i)} \geq \tau$ for some $m < T$. Similarly, it is consistently classified as negative if $\max_{t=m}^{T} p_t^{(i)} \leq \tau$ for some $m < T$. These definitions follow:

- **True Positive**: The prediction is a true positive if it is classified as consistently positive and the class label is $y^{(i)} =$ true.

- **False Positive**: We have a false positive if it is classified as consistently positive, but the manoeuvre is labelled $y^{(i)} = \mathsf{false}$.
- **True Negative**: We have a true negative if the manoeuvre is classified as consistently negative and labelled $y^{(i)} = \mathsf{false}$.
- **False Negative**: A false negative is when the manoeuvre is classified as consistently negative, but is labelled$y^{(i)} = \mathsf{true}$.

Time Gain Analysis: The idea here is to look at how far back in time a classifier is consistently classifying correctly. We choose some positively labelled sequence $\mathcal{S}_i$ (that is, $y^{(i)} = \mathsf{true}$), and define $m_h^{(i)}$ as the smallest m for which the classifier h consistently classifies the sequence as positive [15,19]. If no such m exists, we use $m = T$. We then create a histogram of the $m_h^{(i)}$-values for all the positively labelled sequences. Similarly, we can compare two classifiers h_1 and h_2 by making a histogram over their differences $\Delta^{(i)} = m_{h_1}^{(i)} - m_{h_2}^{(i)}$.

6.3.1 Application scenario 1: Early recognition of a lane change manoeuvre

As detailed in Deliverable 2.1, this application scenario mainly consists of the identification of an ongoing manoeuvre before it really happens. As also described in this deliverable, Daimler has already developed a static classifier for this purpose, the so-called **Static OO-BN model** [15, 17–19]. The main aim here is to evaluate if the **Dynamic OO-BN model** proposed in Deliverable 2.1 is able to make better predictions and with an earlier recognition capacity than its static counterpart.

For this purpose, the above evaluation measures will be computed for the three classes. Similarly, these measures will be computed using either only the EGO manoeuvres in the evaluation dataset, only the OBJ manoeuvres, or both together. In that way, we can analyse in which tasks the new classification model works better.

6.3.2 Application scenario 2: Earlier prediction of the need for a lane change based on relative dynamics

In this application scenario we are going to evaluate new classification models following the same evaluation strategy as described above; the only difference being that the new classification models aim at making earlier manoeuvre recognition using an extended set of features based on the relative dynamics between relevant vehicles [14,15] (see Deliverable 2.1 for details). The evaluation procedure is nevertheless identical, and will try to identify if the features encoding relative dynamics incorporated into this model improve the accuracy of the predictions, and whether these predictions are made earlier in time than the models evaluated in the first application scenario.

6.4 Run-time performance: test and evaluation

The requirements for run-time and space performance were detailed in Table 6.1 and apply to the dynamic classifiers proposed in each application scenario. We now detail the procedure that will be employed to verify each of them:

DAI.U7.02: This is a time requirement which specifies that once a new observation $\boldsymbol{x}_t^{(i)}$ arrives it should be classified (requiring inference in the model in order to compute the posterior probability over the class variable) in less than 0.1 ms using a Linux platform with I7 CPUs.

DAI.U7.04: This requirement addresses the memory space complexity of the models, which includes the following elements:

- The representation of the models (that is, the code needed to fully specify the structure and the parameters of the model).
- The inference algorithm (i.e., the code needed to implement the algorithms used to make inferences over these models).
- All auxiliary methods or objects needed to perform the manoeuvre classifications.

Please note that the requirements regarding memory usage should be interpreted not as "less than 100kB", but "less than in the order of hundreds of kB". The precise numbers that will be used are given in Deliverable 6.1.

References

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

[2] Kerby, D.S.: The simple difference formula: An approach to teaching nonparametric correlation. Innovative Teaching **3** (2014) Article 1

[3] 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

[4] 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

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

[6] Hand, D.J., Till, R.J.: A simple generalization of the area under the ROC curve for multiple class classification problems. Machine Learning **45**(2) (2001) 171–186

[7] Gama, J., 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

[8] 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

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

[10] 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

[11] 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

[12] 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

[13] 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)

[14] Weidl, G., Madsen, A.L., Kasper, D., Breuel, G.: Optimizing Bayesian networks for recognition of driving maneuvers to meet the automotive requirements. In: IEEE Multi-Conference on Systems and Control, Nice/Antibes, France, October 8-10. (2014) 1626–1631

[15] Tereshchenko, V.: Relative object-object dynamics for earlier recognition of maneuvers in highway traffic. Master's thesis, University of Stuttgart (October 2014)

[16] Koller, D., Pfeffer, A.: Object-oriented Bayesian networks. In: Thirteenth Conference on Uncertainty in Artificial Intelligence. (1997) 302–313

[17] Kasper, D., Weidl, G., Dang, T., Breuel, G., Tamke, A., Rosenstiel, W.: Object-oriented Bayesian networks for detection of lane change maneuvers. In: IEEE Intelligent Vehicles Symposium (IV), Kongresshaus Baden-Baden, Germany, June 5-9. (2011) 673–678

[18] Kasper, D., Weidl, G., Dang, T., Breuel, G., Tamke, A., Wedel, A., Rosenstiel, W.: Object-oriented Bayesian networks for detection of lane change maneuvers. IEEE Intelligent Transportation Systems Magazine **4**(3) (2012) 19–31

[19] Kasper, D.: Erkennung von Fahrmanövern mit Object-Orrientierten Bayes-Netzen in Autobahnszenarien. PhD thesis, Tübingen University, Germany (2013)